ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

Bài kiểm tra giữa kỳ

Môn học: Quy hoạch tuyến tính

Sinh viên thực hiện:

Lê Công Đắt (20120454)

Giảng viên hướng dẫn:

ThS. Lê Phúc Lữ

ThS. Trần Hà Sơn

Ngày 6 tháng 5 năm 2024

Mục lục

1 Bài 1 (3.0 điểm) **2**

1.1 Đề bài 2

1.2 Bài giải 2

2 Bài 2 (4.0 điểm) **4**

2.1 Đề bài 4

2.2 Bài giải 5

3 Bài 3 (3.0 điểm) **10**

3.1 Đề bài 10

3.2 Bài giải 11

1 Bài 1 (3.0 điểm)

1.1 Đề bài

Cho bài toán quy hoạch tuyến tính có hàm mục tiêu $f = 2x_1 - 3x_2 \to min$ với các ràng buộc như sau:

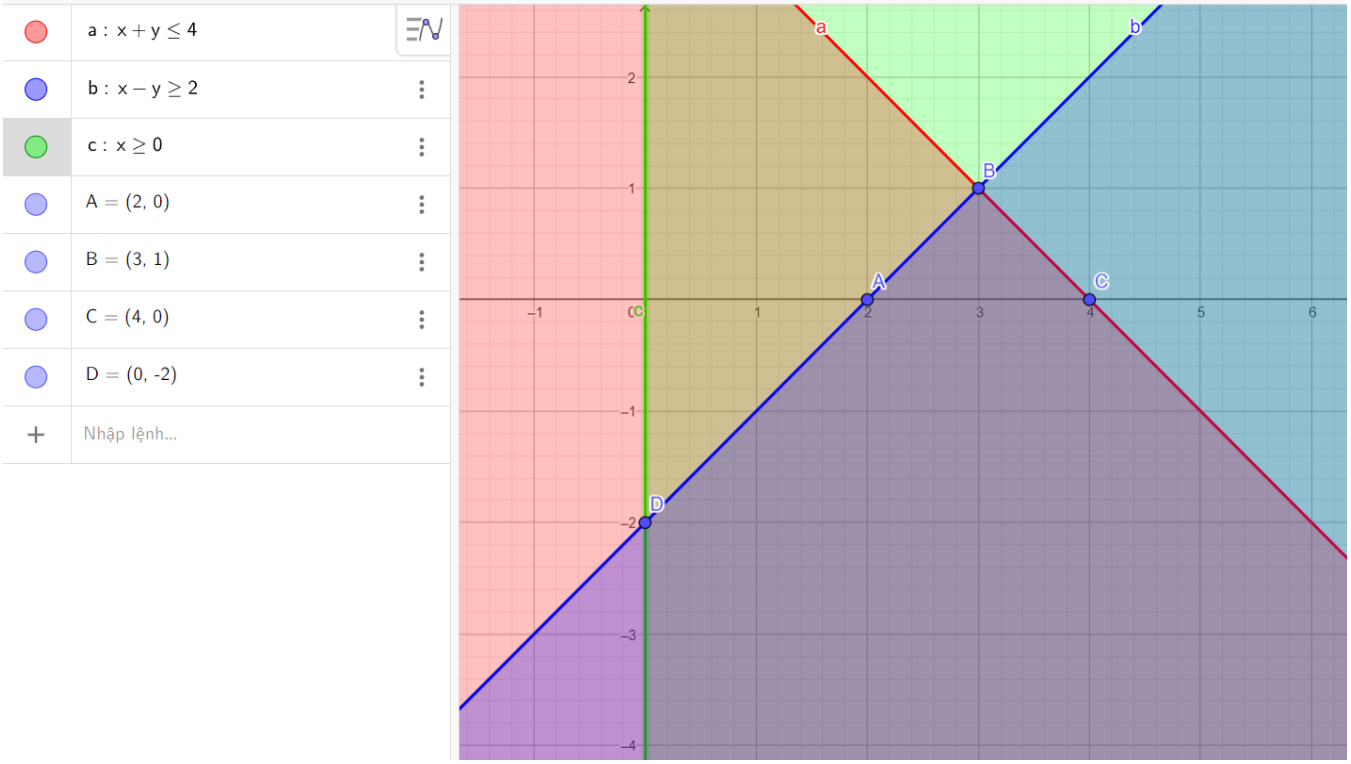
$$\begin{cases} x_1 + x_2 \leq 4 \\ x_1 - x_2 \geq 2 \end{cases} \quad \text{và } x_1 \geq 0,\ x_2 \in \mathbb{R}$$

a) Hãy giải bài toán trên bằng phương pháp hình học.

b) Đưa bài toán trên về dạng chính tắc và giải lại nó bằng phương pháp đơn hình

1.2 Bài giải

a)



Hình 1: Vẽ các điều kiện lên hệ trục tọa độ

Ta có các điểm cực biên là:

- $A(2,0) \to f(2,0) = 4$

- $B(3,1) \to f(3,1) = 3$
- $C(4,0) \to f(4,0) = 8$
- $D(0,-2) \to f(0,-2) = 6$

Quan sát Hình 1 (trục hoành là x_1, trục tung là x_2), ta thấy rằng, miền ràng buộc không bị chặn về phía $x_2 \to -\infty$ và $x_1 \geq 0$

$$\begin{cases} x_2 \to -\infty \Rightarrow -3x_2 \to +\infty \\ x_1 \geq 0 \Rightarrow 2x_1 \geq 0 \end{cases}$$
$$\Rightarrow f = 2x_1 - 3x_2 \to +\infty$$

Từ các kết quả trên, ta có thể kết luận: $f_{min} = 3$ tại B(3, 1)

b)

Ta thêm hai biến tạm x_3, x_4 để tạo đẳng thức và biến giả x_5

Hàm mục tiêu trở thành $f = 2x_1 - 3x_2 + Mx_5 \to min$ với các ràng buộc như sau:

$$\begin{cases} x_1 + x_2 + x_3 = 4 \\ x_1 - x_2 - x_4 + x_5 = 2 \end{cases} \quad \text{và } x_1 \geq 0,\ x_2 \in \mathbb{R},\ x_3 \geq 0,\ x_4 \geq 0,\ x_5 \geq 0$$

Lập bảng đơn hình

x_B	c_B	PA	x_1 2	x_2 -3	x_3 0	x_4 0	x_5 M
x_3	0	4	1	1	1	0	0
x_5	M	2	**1**	-1	0	-1	1
f_{min}		$2M$	$M-2$	$-M+3$	0	$-M$	0

Xét $x_1 : \frac{2}{1}(M-2) = 2M - 4$

x_1 vào, x_5 ra, vị trí số **1** là phần tử xoay

Dòng $x_3 \to$ Dòng x_3 - Dòng x_1

x_B	c_B	PA	x_1 2	x_2 -3	x_3 0	x_4 0	x_5 M
x_3	0	2	0	**2**	1	1	-1
x_1	2	2	1	-1	0	-1	1
f_{min}		4	0	1	0	-2	$2-M$

Xét $x_2 : \frac{2}{2} \times 1 = 1$

x_2 vào, x_3 ra, vị trí số **2** là phần tử xoay

Dòng $x_2 \to$ Dòng $x_2 : 2$

Dòng $x_1 \to$ Dòng x_1 + Dòng x_2

x_B	c_B	PA	x_1 2	x_2 -3	x_3 0	x_4 0	x_5 M
x_2	-3	1	0	1	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
x_1	2	3	1	0	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
f_{min}		3	0	0	$-\frac{1}{2}$	$-\frac{5}{2}$	$\frac{5}{2}-M$

Đến đây, tất cả Δ đều ≤ 0

Vậy $f_{min} = 3$ tại $(3, 1)$

2 Bài 2 (4.0 điểm)

2.1 Đề bài

Cho bài toán quy hoạch tuyến tính có hàm mục tiêu $f = 2x_1 + 4x_2 + \lambda x_3 \to max$ với các ràng buộc sau:

$$\begin{cases} x_1 + 3x_2 + x_4 = 4 \\ 2x_1 + x_2 + x_3 = 3 \qquad \text{và } x_i \geq 0,\ \forall i = 1, 2, 3, 4 \\ x_2 - 3x_4 \leq 3 \end{cases}$$

Gọi a, b lần lượt là chữ số lớn nhất và chữ số nhỏ nhất trong MSSV của anh/chị.

a) Với $\lambda = a$ hãy phát biểu bài toán đối ngẫu của bài toán trên. Hãy giải chi tiết một trong hai: bài gốc hoặc bài đối ngẫu rồi dùng định lý độ lệch bù để tìm phương án tối ưu cho bài còn lại. Tại sao anh/chị lại lựa chọn như thế?

b) Chứng minh rằng $\mathbf{x}^\top = (1; 1; 0; 0)$ là một phương án cực biên của bài toán với mọi giá trị λ. Hỏi nếu chọn $\lambda = b$ thì phương án $\mathbf{x}^\top$ có là tối ưu hay không?

c) Xác định điều kiện của λ để bài toán đã cho có vô số phương án tối ưu.

2.2 Bài giải

$MSSV : 20120454 \Rightarrow a = 5,\ b = 0$

a)

$\lambda = a = 5$

Bài toán đối ngẫu của bài toán gốc là:

Cho bài toán quy hoạch tuyến tính có hàm mục tiêu $g = 4y_1 + 3y_2 + 3y_3 \to min$ với các ràng buộc sau:

$$\begin{cases} y_1 + 2y_2 \geq 2 \\ 3y_1 + y_2 + y_3 \geq 4 \\ y_2 \geq \lambda \\ y_1 - 3y_3 \geq 0 \end{cases} \quad \text{và } y_1 \in \mathbb{R},\ y_2 \in \mathbb{R},\ y_3 \geq 0$$

Nhận xét: Chỉ nên chọn giải bài toán đối ngẫu khi bài toán gốc có số ràng buộc nhiều hơn số biến. Trong khi đó:

- Bài toán gốc có 4 biến, 3 ràng buộc; Bài toán đối ngẫu có 3 biến, 4 ràng buộc.
- Trong 3 ràng buộc của bài toán gốc, có 2 phương trình, có sẵn 1 biến cơ sở ở ràng buộc thứ 2; Trong 4 ràng buộc của bài toán đối ngẫu, không có phương trình nào và không có sẵn biến cơ sở ở ràng buộc nào.

Do đó, ta sẽ chọn **giải bài toán gốc** thay vì bài toán đối ngẫu, rồi dùng định lý độ lệch bù để tìm phương án tối ưu cho bài đối ngẫu.

Tìm phương án tối ưu cho bài toán gốc:

Ta thêm biến tạm x_5 để tạo đẳng thức và biến giả x_6.

Bài toán gốc có hàm mục tiêu $f = 2x_1 + 4x_2 + \lambda x_3 - Mx_6 \to max$ với các ràng buộc sau:

$$\begin{cases} x_1 + 3x_2 + x_4 + x_6 = 4 \\ 2x_1 + x_2 + x_3 = 3 \\ x_2 - 3x_4 + x_5 = 3 \end{cases} \quad \text{và } x_i \geq 0,\ \forall i = 1,2,3,4,5,6$$

Lập bảng đơn hình

x_B	c_B	PA	x_1 2	x_2 4	x_3 λ	x_4 0	x_5 0	x_6 $-M$
x_6	$-M$	4	1	3	0	**1**	0	1
x_3	λ	3	2	1	1	0	0	0
x_5	0	3	0	1	0	-3	1	0
f_{max}		$-4M + 3\lambda$	$-M + 2\lambda - 2$	$-3M + \lambda - 4$	0	$-M$	0	0

Xét x_1: $\frac{3}{2}|-M + 2\lambda - 2| = \frac{3}{2}M - 3\lambda + 3 = \frac{3}{2}M - 3 \times 5 + 3 = \frac{3}{2}M - 12$

x_2: $\frac{4}{3}|-3M + \lambda - 4| = 4M - \frac{4}{3}\lambda + \frac{16}{3} = 4M - \frac{4}{3} \times 5 + \frac{16}{3} = 4M - \frac{4}{3}$

x_4: $\frac{4}{1}(-M) = 4M$

$4M > 4M - \frac{4}{3} > \frac{3}{2}M - 12$

Suy ra x_4 vào, x_6 ra, vị trí số **1** là phần tử xoay.

Dòng $x_5 \to$ Dòng $x_5 + 3 \times$ Dòng x_4.

x_B	c_B	PA	x_1 2	x_2 4	x_3 λ	x_4 0	x_5 0	x_6 $-M$
x_4	0	4	1	3	0	**1**	0	1
x_3	λ	3	2	1	1	0	0	0
x_5	0	15	3	10	0	0	1	3
f_{max}		$3\lambda = 15$	$2\lambda - 2 = 8$	$\lambda - 4 = 1$	0	0	0	M

Tất cả Δ đều ≥ 0.

Vậy $f_{max} = 15$ tại $(0, 0, 3, 4)$.

Tìm phương án tối ưu cho bài đối ngẫu:

$x_3 \neq 0 \Rightarrow$ Ràng buộc thứ 3 của bài đối ngẫu xảy ra dấu bằng $\Rightarrow y_2 = \lambda = 5$

$x_4 \neq 0 \Rightarrow$ Ràng buộc thứ 4 của bài đối ngẫu xảy ra dấu bằng $\Rightarrow y_1 - 3y_3 = 0$

Ràng buộc thứ 3 của bài toán gốc $x_2 - 3x_4 \leq 3 \Leftrightarrow 0 - 3 \times 4 \leq 3 \Leftrightarrow -12 \leq 3$ không xảy ra dấu bằng $\Rightarrow y_3 = 0$

Thay $y_3 = 0$ vào phương trình $y_1 - 3y_3 = 0$ ở trên $\Rightarrow y_1 = 0$

Thay $y_1 = 0, y_2 = 5, y_3 = 0$ vào hàm mục tiêu của bài đối ngẫu, ta được:

$g = 4y_1 + 3y_2 + 3y_3 = 4 \times 0 + 3 \times 5 + 3 \times 0 = 15$

Vậy $g_{min} = 15$ tại $(0, 5, 0)$

b)

Chứng minh rằng $\mathbf{x}^\top = (1; 1; 0; 0)$ là một phương án cực biên của bài toán với mọi giá trị λ

Ta thay $(1, 1, 0, 0)$ vào các ràng buộc của bài toán:

$$\begin{cases} x_1 + 3x_2 + x_4 = 4 \\ 2x_1 + x_2 + x_3 = 3 \qquad \text{và } x_i \geq 0,\ \forall i = 1, 2, 3, 4 \\ x_2 - 3x_4 \leq 3 \end{cases}$$

$$\Leftrightarrow \begin{cases} 1 + 3 \times 1 + 0 = 4 \\ 2 \times 1 + 1 + 0 = 3 \qquad \text{và } x_i \geq 0,\ \forall i = 1, 2, 3, 4 \\ 1 - 3 \times 0 \leq 3 \end{cases}$$

Ta thấy rằng $\mathbf{x}^\top = (1; 1; 0; 0)$ thỏa tất cả ràng buộc của bài toán nên $\mathbf{x}^\top$ là một phương án của bài toán.

Định lý:

Ký hiệu ma trận hàng $C = (c_1, c_2, ..., c_n)_{1 \times n}$ và các ma trận:

$X = (x_1, x_2, ..., x_n)^\top_{n \times 1}$

$b = (b_1, b_2, ..., b_m)^\top_{m \times 1}$

$A = (a_{ij})_{m \times n}$

Ta có thể viết bài toán quy hoạch tuyến tính dưới dạng ma trận:

$z = CX \to max$

với điều kiện: $\begin{cases} AX \leq b \\ X \geq 0 \end{cases}$

Phương án $X = (x_i)$ là điểm cực biên khi và chỉ khi hệ vector cột $\{A_i\}$ ứng với các $x_i > 0$ độc lập

tuyến tính.

Xét ma trận:

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 1 \\ 0 & 1 \end{bmatrix}$$

Ta có $rank(A) = 2$ bằng với số $x_i > 0$ của $\mathbf{x}^\top \Rightarrow A$ độc lập tuyến tính.

Do đó, $\mathbf{x}^\top$ là một phương án cực biên của bài toán với mọi giá trị λ.

Nếu chọn $\lambda = b = 0$ thì phương án $\mathbf{x}^\top$ có là tối ưu hay không?

Giả sử: $\mathbf{x}^\top$ là một phương án tối ưu của bài toán gốc ứng với $\lambda = b = 0$

Từ phương án tối ưu của bài toán gốc, ta suy ra phương án tối ưu của bài toán đối ngẫu:

$x_1 \neq 0 \Rightarrow$ Ràng buộc thứ 1 của bài đối ngẫu xảy ra dấu bằng $\Rightarrow y_1 + 2y_2 = 2$

$x_2 \neq 0 \Rightarrow$ Ràng buộc thứ 2 của bài đối ngẫu xảy ra dấu bằng $\Rightarrow 3y_1 + y_2 + y_3 = 4$

Ràng buộc thứ 3 của bài toán gốc $x_2 - 3x_4 \leq 3 \Leftrightarrow 1 - 3 \times 0 \leq 3 \Leftrightarrow 1 \leq 3$ không xảy ra dấu bằng $\Rightarrow y_3 = 0$

Thay $y_3 = 0$ vào 2 phương trình trên, ta có hệ phương trình:

$$\begin{cases} y_1 + 2y_2 = 2 \\ 3y_1 + y_2 = 4 \end{cases}$$

Giải hệ trên, ta được $y_1 = \frac{6}{5}$ và $y_2 = \frac{2}{5}$

Thay $\mathbf{y}^\top = (\frac{6}{5}, \frac{2}{5}, 0)$ vào bài toán đối ngẫu ứng với $\lambda = b = 0$:

$$\begin{cases} y_1 + 2y_2 \geq 2 \\ 3y_1 + y_2 + y_3 \geq 4 \\ y_2 \geq 0 \\ y_1 - 3y_3 \geq 0 \end{cases} \quad \text{và } y_1 \in \mathbb{R},\ y_2 \in \mathbb{R},\ y_3 \geq 0$$

$$\Leftrightarrow \begin{cases} \frac{6}{5} + 2 \times \frac{2}{5} \geq 2 \\ 3 \times \frac{6}{5} + \frac{2}{5} + 0 \geq 4 \\ \frac{2}{5} \geq 0 \\ \frac{6}{5} - 3 \times 0 \geq 0 \end{cases} \quad \text{và } y_1 \in \mathbb{R},\ y_2 \in \mathbb{R},\ y_3 \geq 0$$

$$\Leftrightarrow \begin{cases} 2 \geq 2 \\ 4 \geq 4 \\ \frac{2}{5} \geq 0 \\ \frac{6}{5} \geq 0 \end{cases} \quad \text{và } y_1 \in \mathbb{R},\ y_2 \in \mathbb{R},\ y_3 \geq 0$$

Ta thấy $\mathbf{y}^\top$ thỏa mãn ràng buộc của bài toán đối ngẫu

Do đó, với $\lambda = b = 0$ thì phương án $\mathbf{x}^\top$ là phương án tối ưu.

c) **Xác định điều kiện của λ để bài toán đã cho có vô số phương án tối ưu.**

Ta thấy rằng với $y_2 \geq \lambda \Leftrightarrow \lambda \leq \frac{2}{5}$ thì $\mathbf{y}^\top = (\frac{6}{5}, \frac{2}{5}, 0)$ là phương án của bài toán đối ngẫu.

Mặt khác, ta có:

$$f(\mathbf{x}^\top) = f(1, 1, 0, 0) = 2 \times 1 + 4 \times 1 + \lambda \times 0 = 6$$

$$g(\mathbf{y}^\top) = g(\frac{6}{5}, \frac{2}{5}, 0) = 4 \times \frac{6}{5} + 3 \times \frac{2}{5} + 3 \times 0 = 6$$

$\Rightarrow f(\mathbf{x}^\top) = g(\mathbf{y}^\top) \Rightarrow \mathbf{y}^\top$ là phương án tối ưu của bài toán đối ngẫu.

Với $\lambda \leq \frac{2}{5}$ thì $\mathbf{y}^\top$ là phương án tối ưu của bài toán đối ngẫu.

Do đó, với $\lambda \leq \frac{2}{5}$ thì $\mathbf{x}^\top$ là phương án tối ưu của bài toán gốc.

Gọi $X = (x_1, x_2, x_3, x_4)$ là phương án tối ưu của bài toán.

Ta thấy $y_1 \neq 0, y_2 \neq 0$ và với $\lambda < \frac{2}{5}$ thì ràng buộc thứ 3 ($y_2 \geq \lambda$) không xảy ra dấu bằng.

Do đó X thỏa mãn hệ phương trình:

$$\begin{cases} x_1 + 3x_2 + x_4 = 4 \\ 2x_1 + x_2 + x_3 = 3 \\ x_3 = 0 \end{cases}$$

Hệ trên có nghiệm $X = (1 + \frac{x_4}{5}, 1 - \frac{2x_4}{5}, 0, x_4)$

X còn phải thỏa các điều kiện còn lại của bài toán:

$$\begin{cases} x_1 \geq 0 \\ x_2 \geq 0 \\ x_4 \geq 0 \\ x_2 - 3x_4 \leq 3 \end{cases}$$

$$\Leftrightarrow \begin{cases} 1 + \dfrac{x_4}{5} \geq 0 \\ 1 - \dfrac{2x_4}{5} \geq 0 \\ x_4 \geq 0 \\ 1 - \dfrac{2x_4}{5} - 3x_4 \leq 3 \end{cases}$$

$$\Leftrightarrow \begin{cases} x_4 \geq -5 \\ x_4 \leq \dfrac{5}{2} \\ x_4 \geq 0 \\ x_4 \geq -\dfrac{10}{17} \end{cases}$$

$$\Leftrightarrow 0 \leq x_4 \leq \frac{5}{2}$$

Vậy khi $\lambda < \frac{2}{5}$ thì bài toán có vô số phương án tối ưu thuộc tập:

$X = (1 + \frac{x_4}{5}, 1 - \frac{2x_4}{5}, 0, x_4)$ với $0 \leq x_4 \leq \frac{5}{2}$

3 Bài 3 (3.0 điểm)

3.1 Đề bài

Xét bài toán sau đây: Một nhà hàng lớn tại TPHCM cần tuyển nhân viên làm việc theo các ca: sáng/trưa/chiều/tối với các quy tắc như sau:

- Một người khi ứng tuyển thì sẽ chọn hai ca nào đó trong bốn ca với lương mỗi ngày như sau: sáng+trưa là 230k, trưa+chiều là 250k, chiều+tối là 280k, sáng+chiều là 250k và trưa+tối là 300k.

- Theo đặc thù thì buổi sáng cần ít nhất 9 nhân viên, buổi trưa cần ít nhất 20 nhân viên, buổi chiều cần ít nhất 11 nhân viên và tối cần ít nhất 17 nhân viên. Hỏi nhà hàng cần có phương án tuyển như thế nào để tổng chi phí trả cho nhân viên là ít nhất?

Yêu cầu: Phát biểu bài toán trên ở dạng quy hoạch tuyến tính nguyên và sử dụng thư viện của Python giải theo 2 cách sau đây để đối chiếu: (1) dùng điều kiện số nguyên ngay từ đầu hoặc (2) giải với số thực rồi điều chỉnh thành số nguyên dần dần thông qua phương pháp nhánh cận.

SV code trên Notebook/Colab rồi chụp ảnh đưa vào file bài làm.

3.2 Bài giải

Phát biểu bài toán trên ở dạng quy hoạch tuyến tính nguyên:

Gọi x_1, x_2, x_3, x_4, x_5 lần lượt là số người làm việc theo ca sáng+trưa, trưa+chiều, chiều+tối, sáng+chiều, trưa+tối.

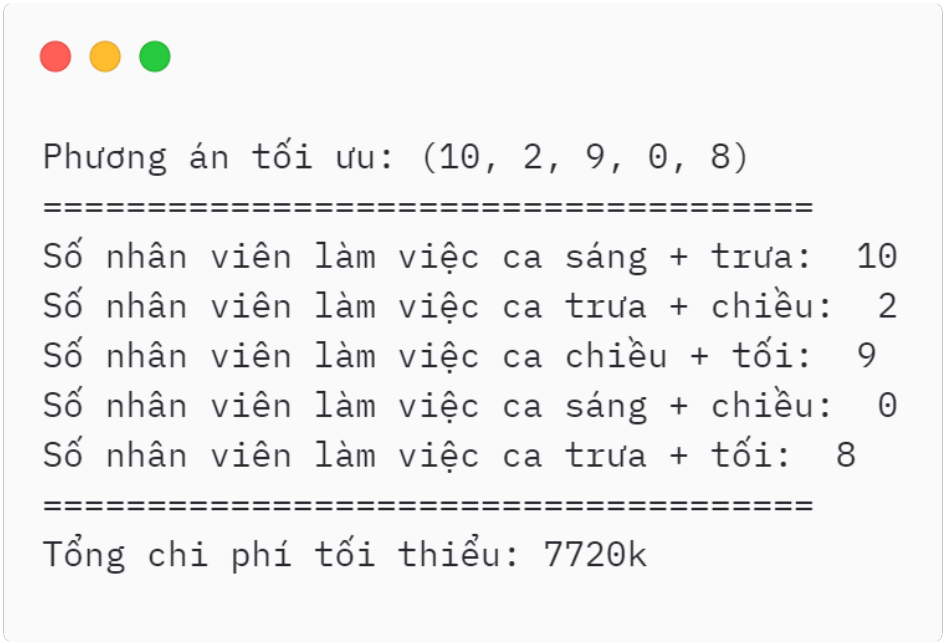
Hàm mục tiêu: $f = 230x_1 + 250x_2 + 280x_3 + 250x_4 + 300x_5 \to min$ với các ràng buộc sau:

$$\begin{cases} x_1 + x_4 \geq 9 \\ x_1 + x_2 + x_5 \geq 20 \\ x_2 + x_3 + x_4 \geq 11 \\ x_3 + x_5 \geq 17 \end{cases} \quad \text{và } x_i \in \mathbb{N},\ \forall i = 1, 2, 3, 4, 5$$

Giải bằng cách dùng điều kiện số nguyên ngay từ đầu:

```python
from pulp import *

# Tạo bài toán
problem = LpProblem("TuyenNhanVien", LpMinimize)

# Khai báo biến: 5 biến x1, x2, x3, x4, x5
items = range(5)
variables = LpVariable.dicts(name="CaLamViec", indices=items, lowBound=0, cat=LpInteger)

# Hàm mục tiêu
problem += lpSum(variables[i] * [230, 250, 280, 250, 300][i] for i in items)

# Ràng buộc
problem += lpSum(variables[i] * [1, 0, 0, 1, 0][i] for i in items) >= 9
problem += lpSum(variables[i] * [1, 1, 0, 0, 1][i] for i in items) >= 20
problem += lpSum(variables[i] * [0, 1, 1, 1, 0][i] for i in items) >= 11
problem += lpSum(variables[i] * [0, 0, 1, 0, 1][i] for i in items) >= 17

# Giải bài toán
problem.solve()

# In phương án tối ưu
print("Phương án tối ưu: (", end="")
for i in range(0, items.__len__()-1):
    print(int(variables[i].value()), end=", ")
print(int(variables[items.__len__()-1].value()), end=")\n")
print("====================================")
print("Số nhân viên làm việc ca sáng + trưa: ", int(variables[0].value()))
print("Số nhân viên làm việc ca trưa + chiều: ", int(variables[1].value()))
print("Số nhân viên làm việc ca chiều + tối: ", int(variables[2].value()))
print("Số nhân viên làm việc ca sáng + chiều: ", int(variables[3].value()))
print("Số nhân viên làm việc ca trưa + tối: ", int(variables[4].value()))
print("====================================")
# In chi phí tối thiểu
print("Tổng chi phí tối thiểu:", int(value(problem.objective)), end="k")
```

Hình 2: Giải bằng cách dùng điều kiện số nguyên ngay từ đầu



Hình 3: Kết quả cách dùng điều kiện số nguyên ngay từ đầu

Giải với số thực rồi điều chỉnh thành số nguyên dần dần thông qua phương pháp nhánh cận:

```python
import scipy
import numpy as np

def is_integer(n):
    """Kiểm tra một số có phải là số nguyên hay không"""
    if int(n) == n:
        return True
    else:
        return False

def branch_and_bound(c, A, b, integer_variables, low_bound, up_bound, is_max = False, depth=1):
    """
    Giải bài toán quy hoạch tuyến tính nguyên bằng phương pháp nhánh cận

    Args:
        c: Hệ số mục tiêu (dạng numpy array)
        A: Ma trận hệ số vế trái của các ràng buộc (dạng numpy array)
        b: Số ở vế phải của các ràng buộc (dạng numpy array)
        integer_variables: Danh sách các biến cần là số nguyên (dạng numpy array boolean)
        low_bound: Giới hạn dưới cho các biến (dạng numpy array)
        up_bound: Giới hạn trên cho các biến (dạng numpy array)
        is_max: Hàm mục tiêu tìm max hay min (mặc định False - tìm min)
        depth: Độ sâu của nhánh (dùng để theo dõi quá trình đệ quy)

    Returns:
        x: Nghiệm khả thi (dạng numpy array)
        f: Giá trị mục tiêu tại nghiệm (float)
        depth: Độ sâu của nhánh tìm thấy nghiệm
    """

    # Giải bài toán quy hoạch tuyến tính (LP relaxation) để lấy nghiệm ước lượng
    if(is_max):
        optimized_lp_relaxation = scipy.optimize.linprog(-c, A, b, method="highs", bounds=list(zip(low_bound, up_bound)))
    else:
        optimized_lp_relaxation = scipy.optimize.linprog(c, A, b, method="highs", bounds=list(zip(low_bound, up_bound)))

    x_candidate = optimized_lp_relaxation.x # Nghiệm ước lượng
    f_candidate = optimized_lp_relaxation.fun # Giá trị mục tiêu tại nghiệm ước lượng

    print(f"Depth: {depth}, x: {x_candidate}, f: {f_candidate}")
```

Hình 4: Phương pháp nhánh cận - Phần 1

```
var_constraint_fulfilled = True # Biến kiểm tra các biến có thỏa mãn là số nguyên hay không

    # Nếu không tìm thấy nghiệm ước lượng thì trả về giá trị không thể đạt được
    if(f_candidate == None):
        if(is_max):
            return [], -np.inf, depth
        else:
            return [], np.inf, depth

    # Kiểm tra xem nghiệm ước lượng có thỏa mãn các biến phải là số nguyên hay không
    for index, bool in enumerate(integer_variables):
        if(bool and not is_integer(x_candidate[index])):
            var_constraint_fulfilled = False
            break
        else:
            var_constraint_fulfilled = True

    # Nếu nghiệm ước lượng thỏa mãn thì trả về luôn
    if(var_constraint_fulfilled):
        if(is_max):
            return x_candidate, -f_candidate, depth
        else:
            return x_candidate, f_candidate, depth

    # Nếu nghiệm ước lượng không thỏa mãn, tiến hành nhánh
    else:
        left_low_bound = np.copy(low_bound)
        left_up_bound = np.copy(up_bound)

        right_low_bound = np.copy(low_bound)
        right_up_bound = np.copy(up_bound)

        max_coeff_index = -1 # Chỉ số của biến có hệ số mục tiêu lớn nhất để nhánh
        for idx, val in enumerate(integer_variables):  # Tìm biến không nguyên có hệ số lớn nhất
            if(val and not is_integer(x_candidate[idx])):
                if(max_coeff_index == -1):
                    max_coeff_index = idx
                elif(c[max_coeff_index] < c[idx]):
                    max_coeff_index = idx

        # Tạo nhánh trái: giới hạn trên của biến được chọn bằng phần nguyên của nghiệm ước lượng
        left_up_bound[max_coeff_index] = np.floor(x_candidate[max_coeff_index])
        # Tạo nhánh phải: giới hạn dưới của biến được chọn bằng phần trần của nghiệm ước lượng
        right_low_bound[max_coeff_index] = np.ceil(x_candidate[max_coeff_index])
        print(f"Branching on x{max_coeff_index+1} at {x_candidate[max_coeff_index]}")

        # Giải đệ quy với các nhánh trái và phải
        x_left, f_left, depth_left = branch_and_bound(c, A, b, integer_variables, left_low_bound, left_up_bound, is_max, depth + 1)
        x_right, f_right, depth_right = branch_and_bound(c, A, b, integer_variables, right_low_bound, right_up_bound, is_max, depth + 1)

        # So sánh giữa 2 nhánh trái và phải để chọn ra nghiệm tốt nhất
        if(is_max):
            if(f_left > f_right):
                return x_left, f_left, depth_left
            else:
                return x_right, f_right, depth_right
        else:
            if(f_left < f_right):
                return x_left, f_left, depth_left
            else:
                return x_right, f_right, depth_right
```

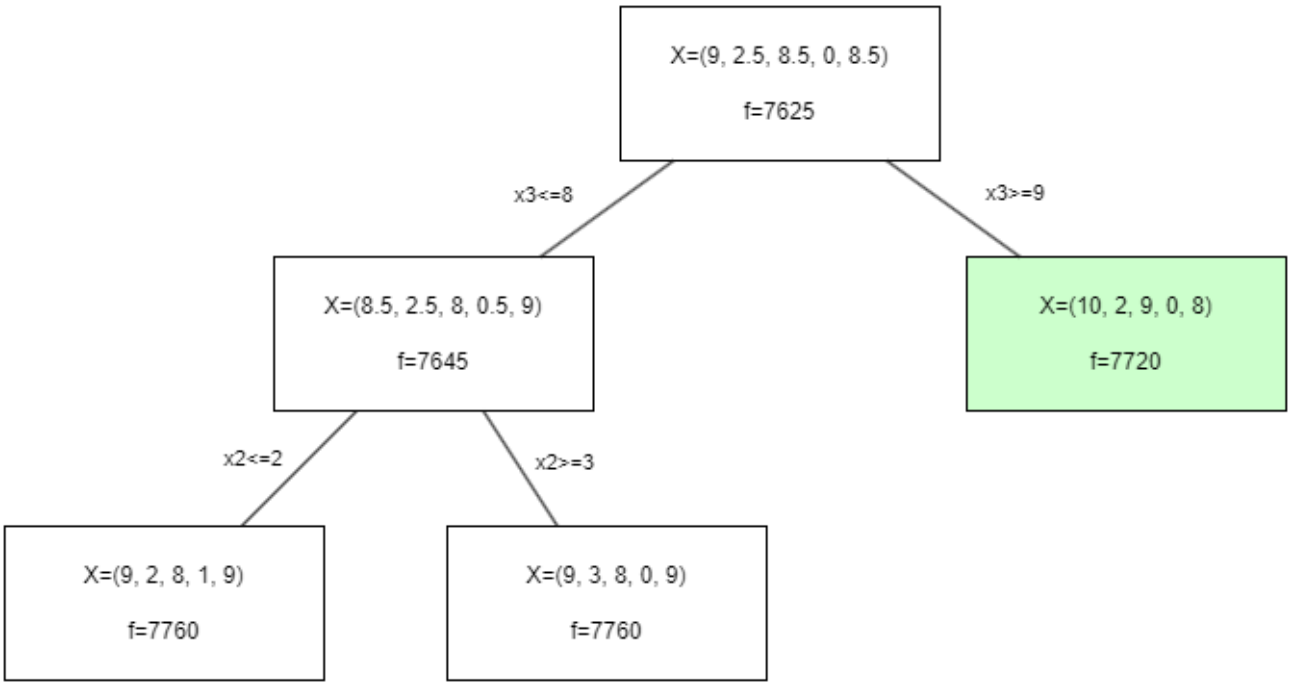
Hình 5: Phương pháp nhánh cận - Phần 2

```python
# Hệ số hàm mục tiêu
c = np.array([230, 250, 280, 250, 300])

# Hệ số ràng buộc, ở đây ràng buộc ban đầu là Ax>=b, ta chuyển về Ax<=b bằng cách nhân -1 cho cả 2 vế
A = np.array([[-1, 0, 0, -1, 0], [-1, -1, 0, 0, -1], [0, -1, -1, -1, 0], [0, 0, -1, 0, -1]])
b = np.array([-9, -20, -11, -17])

# Các biến phải là số nguyên
integer_variables = [True, True, True, True]

# Giới hạn dưới cho mỗi biến
low_bound = [0, 0, 0, 0, 0]
# Giới hạn trên cho mỗi biến
up_bound = [None, None, None , None, None]

x_optimal, f_optimal, depth_optimal = branch_and_bound(c, A, b, integer_variables, low_bound, up_bound, False)
print("====================================")

print("Phương án tối ưu: (", end="")
for i in range(len(x_optimal)-1):
    print(int(x_optimal[i]), end=", ")
print(int(x_optimal[-1]), end=")\n")
print("====================================")
print("Số nhân viên làm việc ca sáng + trưa: ", int(x_optimal[0]))
print("Số nhân viên làm việc ca trưa + chiều: ", int(x_optimal[1]))
print("Số nhân viên làm việc ca chiều + tối: ", int(x_optimal[2]))
print("Số nhân viên làm việc ca sáng + chiều: ", int(x_optimal[3]))
print("Số nhân viên làm việc ca trưa + tối: ", int(x_optimal[4]))
print("====================================")
print(f"Tổng chi phí tối thiểu: {int(f_optimal)}k")
print("====================================")
print(f"Tree depth: {depth_optimal}")
```

Hình 6: Phương pháp nhánh cận - Phần 3

```
Depth: 1, x: [9.  2.5 8.5 0.  8.5], f: 7625.0
Branching on x3 at 8.5
Depth: 2, x: [8.5 2.5 8.  0.5 9. ], f: 7645.0
Branching on x2 at 2.5
Depth: 3, x: [9. 2. 8. 1. 9.], f: 7760.0
Depth: 3, x: [9. 3. 8. 0. 9.], f: 7760.0
Depth: 2, x: [10.  2.  9.  0.  8.], f: 7720.0
====================================
Phương án tối ưu: (10, 2, 9, 0, 8)
====================================
Số nhân viên làm việc ca sáng + trưa:  10
Số nhân viên làm việc ca trưa + chiều:  2
Số nhân viên làm việc ca chiều + tối:  9
Số nhân viên làm việc ca sáng + chiều:  0
Số nhân viên làm việc ca trưa + tối:  8
====================================
Tổng chi phí tối thiểu: 7720k
====================================
Tree depth: 2
```

Hình 7: Kết quả chạy bằng phương pháp nhánh cận



Hình 8: Sơ đồ nhánh cận

Nhận xét. 2 cách giải cho kết quả giống nhau

Xem toàn bộ mã nguồn tại đây