KHOA TOÁN TIN - TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA TP.HCM

BÁO CÁO DỰ ÁN CUỐI KỲ

Môn học: Quy hoạch tuyến tính

Đề tài: Viết chương trình giải bài toán Quy hoạch tuyến tính tổng quát.

Giảng viên: Nguyễn Lê Hoàng Anh
Trợ giảng:

Sinh viên:
20280028 – Lê Thị Mỹ Hằng
20280064 – Mai Thị Thảo Ly
20280088 – Nguyễn Thị Hồng Thi

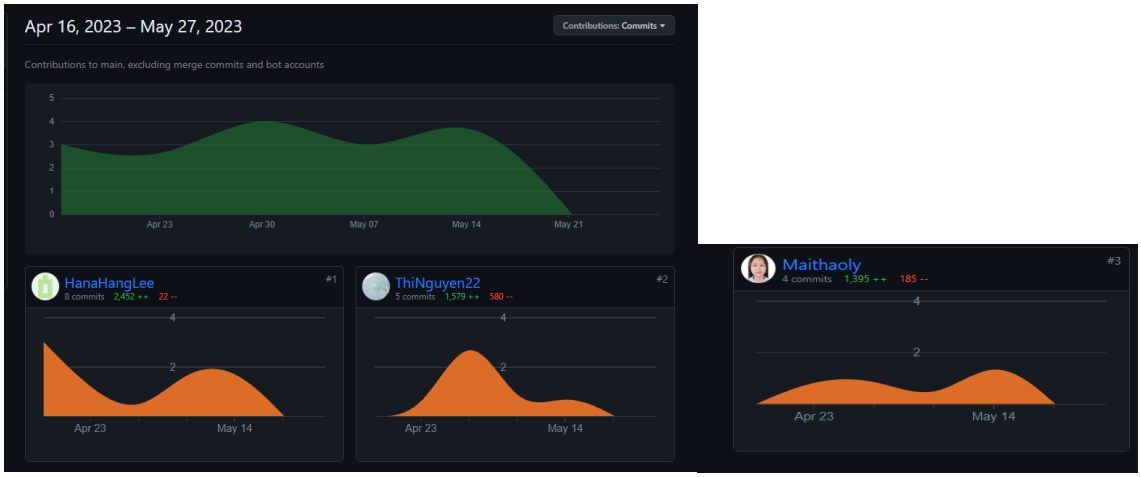
Tp Thủ Đức – Tháng 5/2023

MỤC LỤC

I. Thông tin nhóm:

Tên nhóm: BabiesChillWithDeadline

STT	MSSV	Tên	Email	Nhóm trưởng	Nhiệm vụ	Đánh giá	Chữ ký
1	20280028	Lê Thị Mỹ Hằng	20280028@student.hcmus.edu.vn	x	Thuật toán Hai pha, Giao diện, Xử lý nghiệm	Hoàn thành	
2	20280064	Mai Thị Thảo Ly	20280064@student.hcmus.edu.vn		Chuẩn hóa, Giao diện, Báo cáo	Hoàn thành	
3	20280088	Nguyễn Thị Hồng Thi	20280088@student.hcmus.edu.vn		Thuật toán Dantzig, Bland, Báo cáo	Hoàn thành	

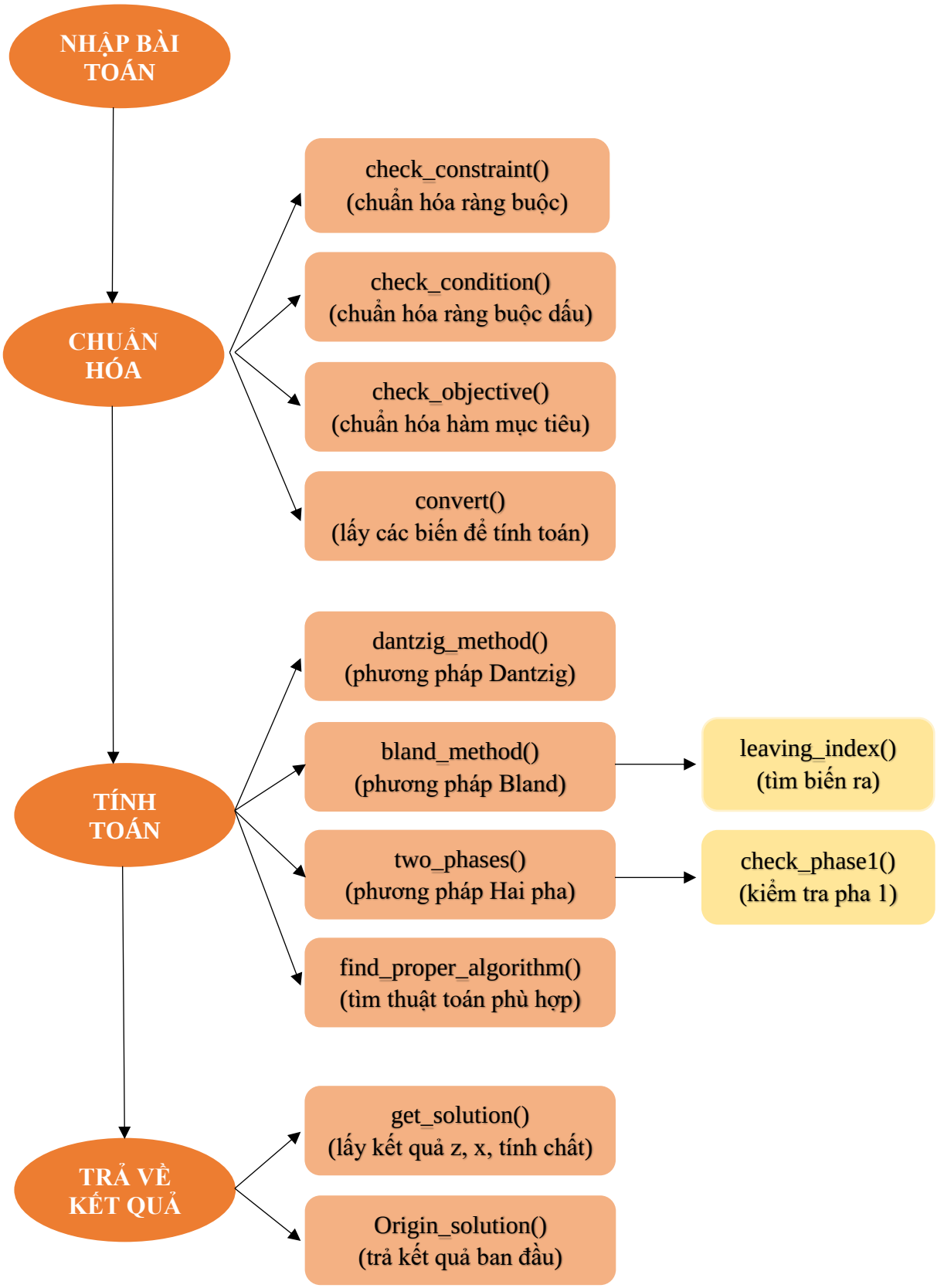


(*Đóng góp của các thành viên trên Github*)

Đánh giá chung: Các thành viên đều hoàn thành nhiệm vụ đúng thời hạn và hỗ trợ nhau hết mình khi gặp khó khăn.

Link Github : https://github.com/HanaHangLee/Linear-Programming

II. Thông tin bài làm:



Trên đây là sơ đồ tóm tắt quy trình sử dụng các hàm giải bài toán quy hoạch tuyến tính và chức năng của các hàm.
Nhóm đã viết được chương trình giải bài toán Quy hoạch tuyến tính tổng quát và viết file thực thi exe để tạo giao diện giải bài toán trên. Trong đó, nhóm đã hoàn thành việc chuẩn hóa bài toán đầu vào tổng quát và các hàm tính toán sử dụng thuật toán đơn hình: Dantzig, Bland, Hai pha để giải quyết các trường hợp có thể xảy ra của bài toán như: nghiệm duy nhất, vô số nghiệm, vô nghiệm, không giới nội, và trường hợp thuật toán lặp vô hạn. Các hàm đó được thực hiện như sau:

1. Chuẩn hóa bài toán quy hoạch tuyến tính:

- **Hàm chuẩn hóa ràng buộc dấu check_condition():**

objective: Hàm mục tiêu của bài toán gốc.
constraint: Ràng buộc biến của bài toán gốc.
condition: Ràng buộc dấu của bài toán gốc.
Ta kiểm tra ràng buộc dấu của các biến:

Nếu x '≤' 0:

- Thay biến x thành x', với x' = -x.
- Đổi dấu ràng buộc dấu '≤' thành '≥'.

Nếu x '≥' 0:

- Không thay đổi gì.

Nếu x '≥' hoặc '≤' b (với b != 0):

- Thay biến x = u – v.
- Ràng buộc biến ta thêm một ràng buộc: x '≥' hoặc '≤' b
- Trong ràng buộc dấu: xóa ràng buộc dấu của x, thêm 2 dòng ràng buộc dấu tương ứng của u và v.
- Ràng buộc dấu của u:
  + Giá trị tại u: 1.
  + Giá trị tại các biến còn lại :0.
  + Nếu là '≤' thì chuyển thành '≥'.
  + Nếu là '≥' thì giữ nguyên.
  + Giá trị tại b: 0.
- Ràng buộc dấu của v:
  + Giá trị tại v: 1.
  + Giá trị tại các biến còn lại :0.
  + Nếu là '≤' thì chuyển thành '≥'.
  + Nếu là '≥' thì giữ nguyên.
  + Giá trị tại b: 0.

Nếu x '?' 0 (x là biến tự do):

- Thay biến x = u – v.
- Trong ràng buộc dấu: xóa ràng buộc dấu của x, thêm 2 dòng ràng buộc dấu tương ứng của u và v.
    - Ràng buộc dấu của u:
        + Giá trị tại u: 1.
    + Giá trị tại các biến còn lại :0.
    + Nếu là '≤' thì chuyển thành '≥'.
    + Nếu là '≥' thì giữ nguyên.
        + Giá trị tại b: 0.
    - Ràng buộc dấu của v:
        + Giá trị tại v: 1.
    + Giá trị tại các biến còn lại :0.
    + Nếu là '≤' thì chuyển thành '≥'.
        + Nếu là '≥' thì giữ nguyên.
    + Giá trị tại b: 0.

Trả về kết quả tất cả ràng buộc dấu đều '≥' 0.

Ví dụ:

```
objective=np.array([['max',5,6]], dtype=np.object)
objective
```

```
array([['max', 5, 6]], dtype=object)
```

```
constraint=np.array([[1,-2,'>=',2],[-2,3,'>=',2]], dtype=np.object)
constraint
```

```
array([[1, -2, '>=', 2],
       [-2, 3, '>=', 2]], dtype=object)
```

```
# Nếu là biến tự do thì ta dùng "?"
condition=np.array([[1,0,'?',0],[0,1,'?',0]], dtype=np.object)
condition
```

```
array([[1, 0, '?', 0],
       [0, 1, '?', 0]], dtype=object)
```

Kết quả sau khi gọi hàm:

```
objective:
 [['max' 5 -5 6 -6]]
constraint:
 [[1 -1 -2 2 '>=' 2]
 [-2 2 3 -3 '>=' 2]]
condition:
 [[1 0 0 0 '>=' 0]
 [0 1 0 0 '>=' 0]
 [0 0 1 0 '>=' 0]
 [0 0 0 1 '>=' 0]]
```

- **Hàm chuẩn hóa ràng buộc check_constraint():**

objective: Hàm mục tiêu sau khi đi qua hàm check_condition().
constraint: Ràng buộc biến sau khi đi qua hàm check_condition().
condition: Ràng buộc dấu sau khi đi qua hàm check_condition().
Ta kiểm tra ràng buộc biến:
Nếu dấu của ràng buộc biến là '≥' b:
- Ta đổi dấu tất cả các biến và b.
- Đổi '≥' thành '≤'.
Nếu dấu của ràng buộc biến là '=':
- Ta xóa dòng chưa ràng buộc biến này và tạo thêm 2 dòng tương ứng:
+ Dòng thứ nhất:

- Giữ nguyên tất cả các giá trị của ràng buộc biến trên.
- Thay '=' thành '≤'.

+ Dòng thứ hai:

- Đổi dấu tất cả các giá tri của ràng buộc biến trên.
- Thay '=' thành '≤'.

Trả về kết quả tất cả ràng buộc dấu đều '≥' 0, tất cả ràng buộc biến đều '≤' 0.
Ví dụ:

```
objective:
 [['max' 5 -5 6 -6]]
constraint:
 [[-1 1 2 -2 '<=' -2]
 [2 -2 -3 3 '<=' -2]]
condition:
 [[1 0 0 0 '>=' 0]
 [0 1 0 0 '>=' 0]
 [0 0 1 0 '>=' 0]
 [0 0 0 1 '>=' 0]]
```

- **Hàm chuẩn hóa hàm mục tiêu check_ objective ():**

objective: Hàm mục tiêu sau khi đi qua hàm **check_constraint()**.
constraint: Ràng buộc biến sau khi đi qua hàm **check_constraint()**.
condition: Ràng buộc dấu sau khi đi qua hàm **check_constraint()**.
Nếu 'max':
- Đổi dấu tất cả các giá trị của hàm objective.
- Thay 'max' thành 'min'.
Nếu 'min':
- Không thay đổi gì.
Trả về kết quả tất cả ràng buộc dấu đều '≥' 0, tất cả ràng buộc biến đều '≤' 0, và hàm mục tiêu là 'min'.
Ví dụ:

```
objective:
 [['min' -5 5 -6 6]]
constraint:
 [[-1 1 2 -2 '<=' -2]
 [2 -2 -3 3 '<=' -2]]
condition:
 [[1 0 0 0 '>=' 0]
 [0 1 0 0 '>=' 0]
 [0 0 1 0 '>=' 0]
 [0 0 0 1 '>=' 0]]
```

- **Hàm chuyển đổi dữ liệu đầu vào convert():**
  Sau khi có bài toán đầu vào và đã chuyển về dạng chuẩn, ta trích ra các biến mới là tham số đầu của các hàm thuật toán A, b, c. Trong đó:
  - A: ma trận chứa hệ số của của các biến trong các ràng buộc
  - b: mảng chứa các giá trị b_i
  - c: mảng chứa hệ số các biến trong hàm mục tiêu

2. Thực hiện giải bài toán:

Ý tưởng chung:

- Khởi tạo các biến là đầu ra của thuật toán: biến giá trị tối ưu (z), biến tính chất bài toán (status), biến các giá trị cơ sở ban đầu (basis).
- Tạo các biến là các ma trận bổ sung để chưa các từ vựng.
- Copy các tham số đầu vào thành các biến mới và thao tác với các biến đó.
- Sử dụng các thuật toán Đơn hình Dantzig, Bland và Hai pha để giải bài toán Quy hoạch tuyến tính. Các thuật toán đó được thực hiện như sau:

- **Phương pháp đơn hình Dantzig dantzig_method():**
  Sau bước khởi tạo các biến sử dụng vòng lặp để:
  - Tìm biến vào trong từ vựng được khởi tạo ban đầu với tiêu chí chọn biến có hệ số âm nhất trên hàm mục tiêu (cột j).
  - Tìm biến ra bằng cách lập các tỷ lệ $\{b_i / | a_{ij} |\}$ với a_{ij} là các hệ số của của x_j trên cột j, và chọn tỷ lệ nhỏ nhất tương ứng là biến ra $x_{ij.}$
  - Thực hiện phép xoay bằng cách khử cột x_j.
  - Lúc này từ vựng mới xuất hiện, với biến cơ sở mới và biến không cơ sở mới. Ta cập nhật lại từ vựng mới, biến gía trị tối ưu mới và lưu lại vị trí của biến cơ sở ban đầu.
  - Lặp lại các bước trên cho đến khi thỏa điều kiện dừng.
  - Điều kiện dừng vòng lặp: khi xuất hiện từ vựng mà tất cả các hệ số của biến x_i trên hàm mục tiêu không âm (từ vựng tối ưu).
  - Ngoài ra, trong quá trình thực hiện vòng lặp có các trường hợp xảy ra như: bài toán có biến vào nhưng không có biến ra thì bài toán Không giới nội (Unbounded), hay từ vựng mới trùng với từ vựng ban đầu thì bài toán không thể áp dụng thuật toán đơn hình Dantzig vì không tìm được từ vựng tối ưu.

Kết thúc vòng lặp, khởi tạo biến x chứa nghiệm tối ưu của bài toán.
Kết quả trả về của bài toán ở dạng chuẩn là giá trị tối ưu (z), nghiệm tối ưu (x) và tính chất của bài toán (status).

- **Phương pháp đơn hình Bland bland_method():**
  Sau bước khởi tạo các biến sử dụng vòng lặp để:
  - Tìm biến vào trong từ vựng được khởi tạo ban đầu với tiêu chí chọn biến có hệ số âm nhất trên hàm mục tiêu (cột j).
  - Tìm biến ra bằng cách lập các tỷ lệ $\{b_i / | a_{ij} |\}$ với a_{ij} là các hệ số của của x_j trên cột j, và chọn tỷ lệ nhỏ nhất tương ứng là biến ra $x_{ij.}$
  - Thực hiện phép xoay bằng cách khử cột x_j.
  - Lúc này từ vựng mới xuất hiện, với biến cơ sở mới và biến không cơ sở mới. Ta cập nhật lại từ vựng mới, biến gía trị tối ưu mới và lưu lại vị trí của biến cơ sở ban đầu.
  - Lặp lại các bước trên cho đến khi thỏa điều kiện dừng.
  - Điều kiện dừng vòng lặp: khi xuất hiện từ vựng mà tất cả các hệ số của biến x_i trên hàm mục tiêu không âm (từ vựng tối ưu).
  - Ngoài ra, trong quá trình thực hiện vòng lặp có các trường hợp xảy ra như: bài toán có biến vào nhưng không có biến ra thì bài toán Không giới nội (Unbounded), hay từ vựng mới trùng với từ vựng ban đầu thì bài toán không thể áp dụng thuật toán đơn hình Dantzig vì không tìm được từ vựng tối ưu.

  Kết thúc vòng lặp, khởi tạo biến x chứa nghiệm tối ưu của bài toán.
  Kết quả trả về của bài toán ở dạng chuẩn là giá trị tối ưu (z), nghiệm tối ưu (x) và tính chất của bài toán (status).
- **Phương pháp đơn hình Hai pha two_phases():**
  Pha 1: Lập bài toán bổ trợ chứa x_0 và xây dựng từ vựng với bài toán bổ trợ và giải bài bài toán đó. Ở vòng lặp đầu tiên, chọn biến vào là biến x_0, biến ra tương ứng với dòng có b_i âm nhất. Thực hiện phép xoay đầu tiên. Tiếp tục dùng thuật toán Dantzig cho từ vựng tiếp theo cho đến khi có từ vựng tối ưu (sử dụng hàm check_phase1() để kiểm tra các trường hợp nghiệm của pha 1). Nếu từ vựng tối ưu chỉ chứa biến x_0 trên hàm mục tiêu thì chuyển sang pha 2. Ngược lại, kết luận bài toán vô nghiệm.
  Pha 2: Cho x_0 bằng 0 ở các ràng buộc, và thiết lập từ vựng mới với hàm mục tiêu mới và các ràng buộc ở từ vựng tối ưu trong pha 1. Tiếp tục sử dụng phương pháp đơn hình bình thường để giải bài toán.
- **Hàm tìm thuật toán phù hợp cho bài toán find_proper_algorithm():**
  Kiểm tra các giá trị trong biến b chứa các giá trị b_i:
  - Nếu trong b tồn tại giá trị b_i âm thì dùng thuật toán Hai pha.
  - Nếu trong b tồn tại giá trị $b_i = 0$ thì dùng thuật toán Bland.
  - Ngược lại, các giá trị b_i đều dương thì dùng thuật toán Dantzig.

3. Trả về kết quả bài toán ban đầu:

- **Hàm lấy kết quả get_solution():**

  Hàm này tạo ma trận lưu trữ các biến cở sở và không cơ sở giúp xét các trường hợp nghiệm của bài toán như nghiệm duy nhất, vô số nghiệm, vô nghiệm… và trả về các nghiệm ban đầu khi các biến đã được đưa về dạng chuẩn.
- **Hàm trả về nghiệm ban đầu origin_solution():**

  Hàm trả về nghiệm của bài toán ban đầu bằng cách xét điều kiện min/max trên hàm mục tiêu và dấu của của các ràng buộc dấu (trừ các trường hợp biến ra là None).
  Nếu bài toán ban đầu có hàm mục tiêu là min thì giữ nguyên giá trị tối ưu. Ngược lại, nếu là max thì đổi dấu giá trị tối ưu.
  Bài toán có vô số nghiệm thì trả về nghiệm được biểu diễn theo biểu thức.
  Nếu bài toán có nghiệm duy nhất: mà là nghiệm tự do ('?') trả về hiệu của hai biến phụ, ràng buộc dấu là '<=' thì đổi dấu của các nghiệm, còn ràng buộc dấu là '>=' thì không biến đổi thêm.

III. Hướng dẫn code:

1. Phần mềm sử dụng:

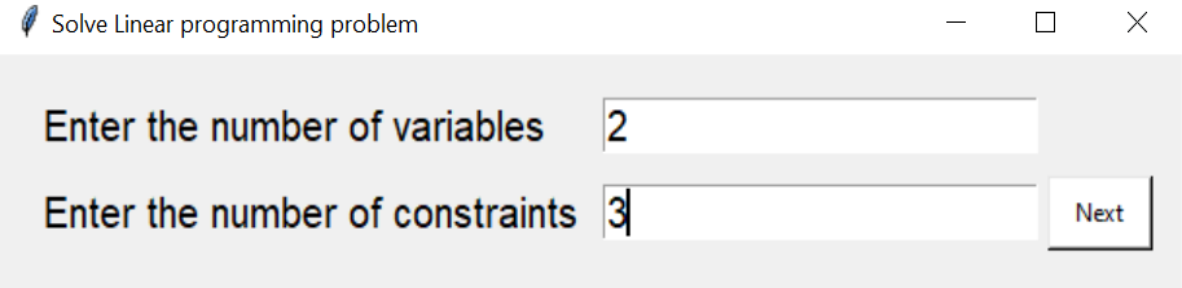
Chương trình được viết trên Jupyter Notebook sau đó chuyển qua Visual Studio Code bằng ngôn ngữ Python phiên bản 3.9.7. Các thư viện sử dụng chủ yếu là Numpy phiên bản 1.23.5 và Tkinter phiên bản 8.6

2. Hướng dẫn nhập:

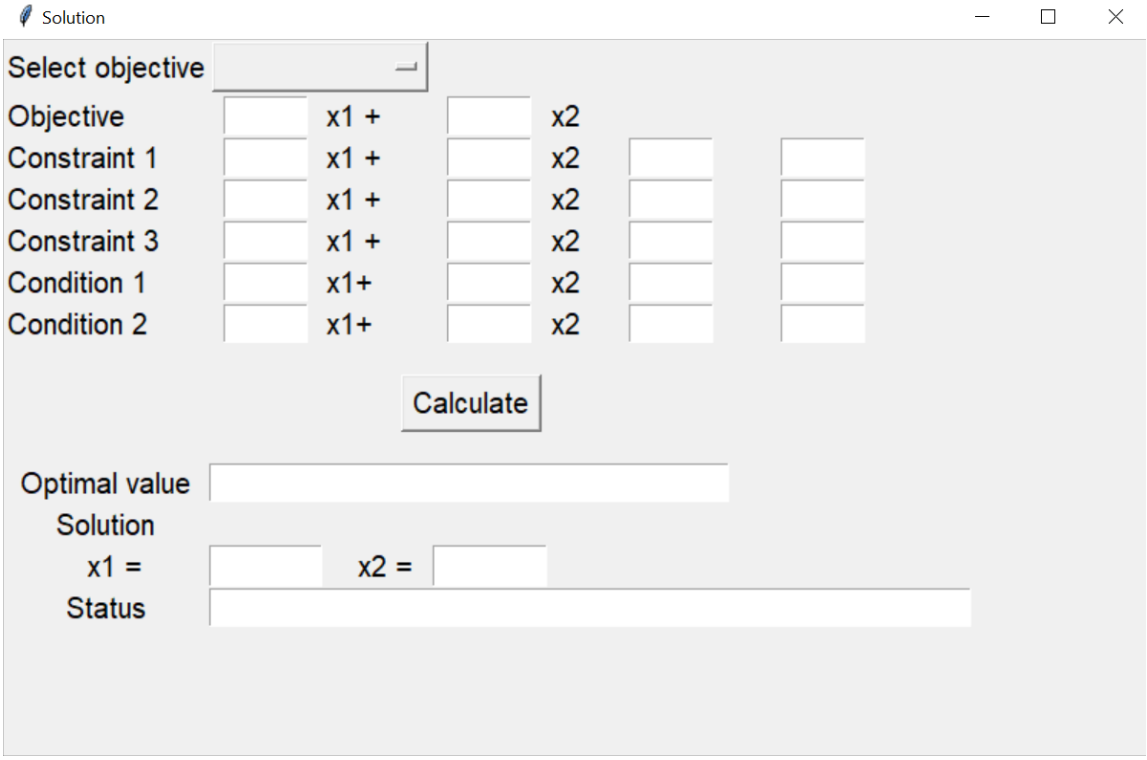
Ví dụ bài toán Quy hoạch tuyến tính:

$$\text{Max } 3x_1 + 5x_2$$
$$x_1 + 2x_2 <= 5$$
$$x_1 <= 3$$
$$x_2 <= 2$$
$$x_1, x_2 >= 0$$

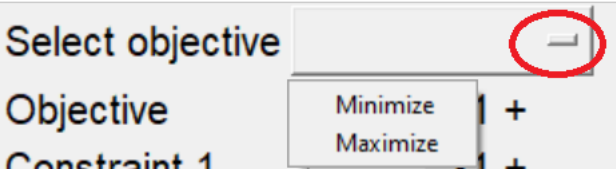
Nhập từ bàn phím số lượng biến (variables) và số lượng ràng buộc (constraint).
Theo ví dụ trên bài toán có 2 biến x_1, x_2 và có 3 ràng buộc thì nhập như sau:



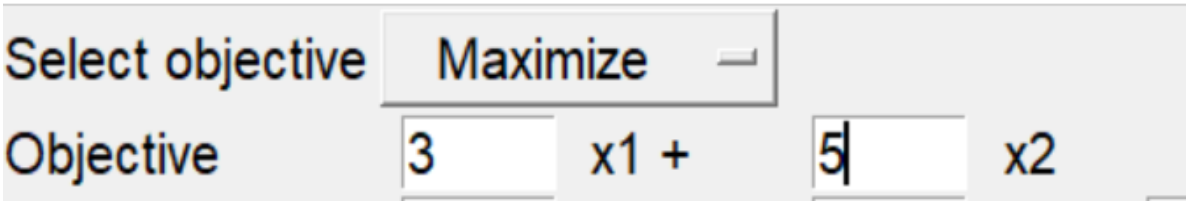
Nhấn ‘Next’ để chuyển tới giao diện nhập bài toán Quy hoạch tuyến tính:



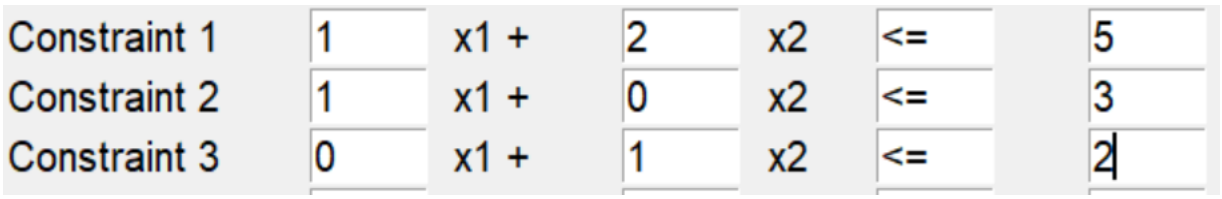
Tại đây, chọn nhấn vào biểu tượng được khoanh tròn trên dòng ‘Select objective’ để chọn hàm mục tiêu là Minimize hoặc Maximize:



Tiếp theo, nhập các hệ số của các biến ở hàm mục tiêu trên dòng ‘Objective’

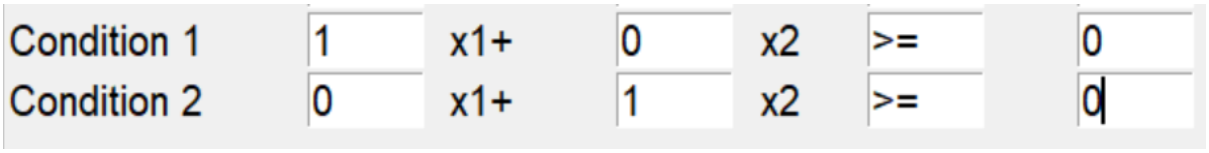


Tiếp tục nhập các ràng buộc của bài toán tương ứng với mỗi dòng ‘Constraint’ bằng cách nhập lần lượt các hệ số của biến, dấu của ràng buộc (>=, <=, =) và các giá trị b_i tương ứng với các cột như sau:



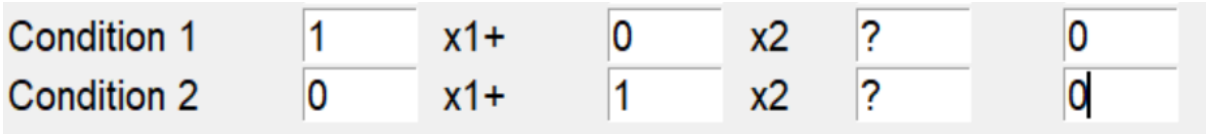
Tiếp tục nhập các ràng buộc dấu của các biến tương ứng với mỗi dòng ‘Condition’ (>=, <=, lưu ý nếu là biến tự do ta nhập dấu ‘?’).

+ Giá trị tại biến x: 1.
+ Giá trị tại các vị trí còn lại: 0.

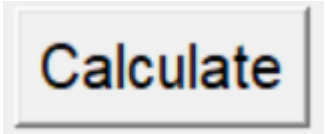


Quy ước: Điều kiện của biến tự do x ta nhập như sau:

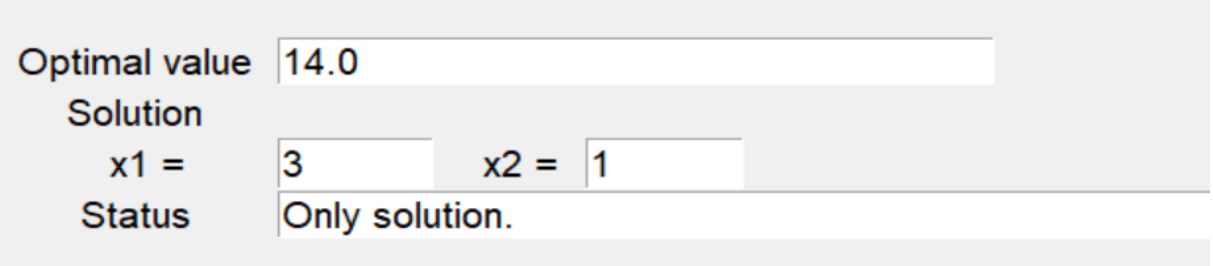
+ Giá trị tại biến x: 1.
+ Tại ô nhập dấu ta nhập ‘?’.
+ Giá trị tại các vị trí còn lại: 0.



Sau khi đã nhập xong bài toán Quy hoạch tuyến tính, nhấn nút ‘Calculate’ để thực hiện tính toán và trả về kết quả:



Kết quả hiển thị ra màn hình gồm: giá trị tối ưu của bài toán (Optimal value), nghiệm tối ưu (Solution: x_1, x_2, …) và tính chất của bài toán (Status: Only solution, Infinite solution, Unbounded)



Link video hướng dẫn cách nhập chi tiết:
https://www.youtube.com/watch?v=XUixqZMAgvw

IV. Tài liệu tham khảo:

Sách giáo trình Quy hoạch tuyến tính GS.TSKH Phan Quốc Khánh – TS. Trần Huệ Nương.
https://python.quantecon.org/lp_intro.html
https://radzion.com/blog/operations/problems
https://uomustansiriyah.edu.iq/media/lectures/6/6_2022_01_08!08_05_56_PM.pdf

Ngoài ra nhóm còn tham khảo ý tưởng code trên các trang Stackoverflow, Geeksforgeeks, Github…

Hết. Cảm ơn thầy!