ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

BÁO CÁO ASSIGNMENT 1 BỘ MÔN AI

GVHD: Vương Bá Thịnh
SV: Nguyễn Văn Sơn 1612980

TP. HỒ CHÍ MINH, THÁNG 3/2019

Mục lục

Danh sách hình vẽ

1 Mục tiêu của Assignment:

- Củng cố các kiến thức của môn học Trí Tuệ Nhân Tạo (AI)
- Rèn luyện thêm về kỹ năng lập trình, đặc biệt là đối với Python
- Rèn luyện cách đọc tài liệu (document)
- Tăng cường khả năng nghiên cứu

2 Mô tả bài toán đặt ra:

2.1 Vấn đề của bài toán:

Xếp lịch thi đấu cho môn thể thao X với n vận động viên và k trận đấu sao cho thoã yêu cầu cho trước.

2.2 Đặt tả của bài toán:

- Một giải đấu có n vận động viên (vđv)
- Một trận đấu gồm 2 vđv thi đấu đối kháng
- Mỗi vđv có 1 điểm số trong bảng xếp hạng của môn thể thao X
- Mỗi vđv thi đấu chính xác với k vđv khác

2.3 Yêu cầu:

- Lịch đấu cần tối ưu hóa mục tiêu sau:

Điểm số trung bình các đối thủ của 2 vđv bất kì (tbinh-i và tbinh-j) không quá chênh lệch.

3 Phân tích bài toán:

Ta thấy rằng yêu cầu của bài toán là xếp lịch thi đấu cho n vận động viên, mỗi vận động viên cần k trận sao cho thống nhất với nhau và thoã mãn tối ưu theo yêu cầu của bài toán.

Ta nhận thấy rằng có sự liên hệ tương xứng giữa bài toán và một ma trận kích thước (nxn) các giá trị 1 trong ma trận đánh dấu trận đấu giữa các vận động viên, các ô còn lại thì có giá trị 0.

Dựa vào tính chất của ma trận ta thấy rằng các giá trị ma trận sẽ đối xứng nhau qua đường chéo chính nên ta chỉ cần nữa ma trận trên cùng là đủ để mô tả các trận đấu của bài toán.

Mặt khác ta thấy nếu n*k mà là một số lẻ thì ma trận sẽ mất tính đối xứng qua đường chéo chính. Từ đây ta suy ra n*k phải là một số chẵn và k $<$ n thì ta mới có thể thực hiện bài toán.

4 Giải quyết bài toán

4.1 Xác định không gian mẫu:

Ta nhận thấy rằng các số n,k không thể ngẫu nhiên mà phải tuân theo một quy luật nhất định thì ta mới tạo được một ma trận tương xứng để xếp lịch cho trân đấu. Dựa vào điều rút ra ở trên, n*k phải là số chẵn và n < k thì ta có các phát biểu sau để xác định không gian mẫu của bài toán rõ hơn:

- Khi số vận động viên là số chẵn thì số trận đấu có thể chẵn hoặc lẻ, và ngược lại.
- số trận đấu của mỗi vận động viên phải ít hơn số vận động viên.(tối đa là n-1).
- Số vận động viên không thể ít hơn hoặc bằng 1(n>=2).
- Số trận đấu không thể bằng 0(k>=1).
- Số trận đấu của vận động viên thứ i và vận động viên thứ j phải bằng nhau($k_{(i)} = k_{(j)} = k$).

4.2 Tạo trạng thái khởi tạo(Initial State)

Phần này được tạo bằng hàm initial_state(k,n) hàm này chả về danh sách đối thủ của từng vận động viên và ma trận thể hiện trận đấu tương ứng!

Ta nhận thấy rằng có một số giá trị k,n đặc biệt có thể giúp ta tạo ma trận cho trạng thái khởi tạo một cách thuận lợi và nhanh chóng:

Với k = 1: Ta điền giá trị 1 sao cho cách 1 ô thì điền 1 ô trên đường chéo song song sát với đường chéo chính thì ta điền số 1 vào như hình:

	0	1	2	3	4	5
0	0	1				
1		0				
2			0	1		
3				0		
4					0	1
5						0

Hình 1: Với n = 6, k=1

Với k = 2: Ta điền giá trị 1 vào các ô của đường chéo song song và sát với đường chéo chính và điền số 1 vào ô đỉnh vuông của tam giác phía trên đường chéo như hình:

	0	1	2	3	4	5
0	0	1				1
1		0	1			
2			0	1		
3				0	1	
4					0	1
5						0

Hình 2: Với n = 6, k =2

Với n chẵn và k = n/2: Ta điền giá trị 1 bằng cách chia các hàng và cột trong ma trận thành hai thành phần từ $0 <= i <= k$ **và** $k + 1 <= j <= n - 1$**, với các ô tương ứng với hàng** $0 <= i <= k$ **và cột** $k + 1 <= j <= n - 1$ **thì ta điền giá trị 1 vào như hình:**

	0	1	2	3	4	5
0	0			1	1	1
1		0		1	1	1
2			0	1	1	1
3				0		
4					0	
5						0

Hình 3: Với n = 6, k=3

Với k = n-1: Ta điền giá trị 1 vào các ô sao cho tạo thành một tam giác vuông cân với cạnh huyền là đường chéo chính như hình:

	0	1	2	3	4	5
0	0	1	1	1	1	1
1		0	1	1	1	1
2			0	1	1	1
3				0	1	1
4					0	1
5						0

Hình 4: Với n = 6, k=5

Với các trường hợp còn lại thì ta làm theo cách sau, ta sinh tổ hợp ngẫu nhiên nCk $(defrandom\_combination(iterable, r))$ **trả về tập hợp đối thủ của vận động ứng với vị trí i tương ứng mỗi hàng i của ma trận. Mỗi lần sinh tổ hợp cho hàng i+1 tiếp theo ta phải xem xét xem vận động viên thứ i+1 đó có nằm trong danh sách đối thủ của các vận động trước đã được sinh tổ hợp trận đấu đầy đủ hay không. Nếu có ta phải loại bỏ và sinh tổ hợp sao cho phù hợp để trận đấu của các vận động viên là tương xứng và đối xứng trong ma trận.Sinh tổ hợp ngẫu nhiên ở các hàng diễn ra liên tiếp và tương hỗ nhau cho đến khi đạt được trạng thái thoả mãn yêu cầu thì dừng. Phần này được hiện thực trong hàm** $(defsubinitial\_state(n, k))$**.**
Chú thích: Với các trường hợp đặc biệt trên để cho thuận tiện cho việc hiện thực thì ta dùng list để ghi danh sách đối thủ của từng vận động viên, sau đó ta cập nhật từ danh sách vào ma trận để thực hiện.

4.3 Giải thuật áp dụng và hàm lượng giá:

Vì số trận đấu của các vận động viên là như nhau(đều bằng k) nên ta có thể xem điểm trung bình của các đối thủ ứng với vận động viên như là tổng điểm các đối thủ của vận động viên.

Tại đây em áp dụng giải thuật leo đồi với hàm lượng giá **H = max(tổng số điểm đối thủ của các vận động viên) - min(tổng số điểm đối thủ của các vận động viên** . H càng nhỏ thì kết quả càng tối ưu. Vì yêu cầu của bài toán là: **"Điểm số trung bình các đối thủ của 2 vđv bất kì (tbinh-i và tbinh-j) không quá chênh lệch."**Điều này tương đương với

tổng điểm đối thủ của hai vận động viên bất kì không quá chênh lệch theo điều ta rút ra được ở trên.Gọi Sum_i là tổng điểm đối thủ của vận động viên thứ i (với $0 \leq i < n$). Gọi max_p, min_p là vận động viên có tổng điểm đối thủ cao nhất và thấp nhất.
Ta nhận thấy rằng: Với mọi i,j ($0 \leq i, j < n$) thì ta luôn có:

$$Sum_{max\_p} - Sum_{min\_p} \geq Sum_i - Sum_j \tag{1}$$

Và ta cũng có được 2 điều sau:

$$Sum_{max\_p} - Sum_j \geq Sum_i - Sum_j \tag{2}$$
$$Sum_i - Sum_{min\_p} \geq Sum_i - Sum_j \tag{3}$$

Từ điều (1),(2) và (3): ta rút ra được rằng để giảm được **sự chênh lệch tổng điểm của hai vận động viên bất kì** thì đầu tiên ta phải giảm đi sự chênh lệch giữa hai vận động viên max_p và min_p, vì nó là độ chênh lệch lớn nhất của mỗi trạng thái.

Luật di chuyển: Lựa chọn đối thủ có điểm số cao nhất của vận động viên đạt max(tổng số điểm đối thủ của các vận động viên) đổi với đối thủ có điểm số thấp nhất của vận động viên đạt min(tổng số điểm đối thủ của các vận động viên. Sau đó cập nhật lại ma trận rồi lại tính hàm lượng giá. Nếu giá trị lượng giá nhỏ hơn thì nhận và sinh bước nhảy tiếp theo. Nếu không thì bỏ qua và lặp lại cho đến khi đối thủ của vận động viên đạt max(tổng số điểm đối thủ của các vận động viên) hoặc đối thủ của vận động viên đạt min(tổng số điểm đối thủ của các vận động viên) đã trao đổi hết. Phần này được xử lí và hiện thực tại hàm $process(mt\_, n, k, ls, ls\_p)$.

Chú ý: Ở đây ta dùng hàm check_(mt_,i,n,k) để chuyển đổi từ ma trận qua danh sách đối thủ để tính hàm lượng giá, và sử dụng hàm update_matrix_2(mt_, list_,i) và hàm update_matrix(mt_,list_,i) để chuyển từ danh sách đối thủ sang ma trận để cập nhật cho mỗi trạng thái mới!

4.4 Ví dụ:

Giả sử số vận đông viên là n = 6 và số trận đấu của mỗi vận động viên là k = 2. Với danh sách điểm như sau:

vđv	0	1	2	3	4	5
điểm	7	18	5	25	12	8

Sau khi thực hiện hàm $initial_state(k, n)$ để sinh ra trạng thái khởi tạo thì được danh sách các đối thủ của các vận động viên theo thứ tự của các vận động viên:

[[1, 5], [0, 2], [1, 3], [2, 4], [3, 5], [0, 4]]

Dựa vào danh sách đối thủ và danh sách điểm ta có thể xác định được số 2 có tổng điểm đối thủ cao nhất(=25) và số 1 có tổng điểm đối thủ thấp nhất(=12).Nên ta sẽ dổi đối thủ của 1 và 2 cho nhau.Ta thấy 2 cũng là đối thủ của 1 trong danh sách hiện tại nên ta chỉ có thể đổi các đối thủ còn lại, đấy là đổi 0 của 1 và 3 của 2. Sau khi đổi và tuỳ chỉnh để các vận động viên có được đối thủ tương ứng ta được:

[[2, 5], [3, 2], [1, 0], [1, 4], [3, 5], [0, 4]] Ta thấy giờ số 0 tổng điểm đối thủ thấp nhất và số 4 tổng điểm đối thủ cao nhất nên ta tiếp tục đổi chúng cho nhau như làm ở trên. Cứ thế lặp đi lặp lại đến khi tổng điểm đối thủ chênh lệch ít nhất thì dừng lại. Nhưng ở đây sau khi đổi 0 và 4 thì hàm lượng giá không tốt hơn nên ta dừng lại và kết quả:

$$[[2, 5], [3, 2], [1, 0], [1, 4], [3, 5], [0, 4]]$$

5 Hàm sinh input và kết quả chạy của một số trường hợp: