BÁO CÁO CUỘC THI TOÁN MÔ HÌNH 2024

ĐỘI THI: DK ALGEBROS

1. Ta có biến nhị phân $x_{i,j}$ với a là 1 trong 8 nhân viên, b là ca làm việc (sáng hoặc chiều). Trong đó:

i) Nhân viên (i) gồm An, Bình, Châu, Dương, Linh, Kiệt, Giang, Hiếu.

ii) Ca làm việc (j) gồm ca sáng (1) và ca tối (2) từ 05/08/2024 đến 01/09/2024.

iii)iii) $x_{i,j} = 1$ có nghĩa là nhân viên a làm việc trong ca j, $x_{i,j} = 0$ có nghĩa là nhân viên a không làm việc trong ca b.

2. Dựa theo đề bài, ta có các ràng buộc (hard constraints) sau:

a) Số ca làm việc trong 1 tháng của mỗi nhân viên

- Nhân viên full time (An, Bình, Châu Dương) làm 18 ca 8 tiếng trong 4 tuần, biểu diễn dưới dạng công thức như sau:

$$\sum_{j=1}^{56} x_{i,j} = 18$$

(do 1 ngày có 2 ca, 1 tháng có 28 ngày nên b sẽ chạy từ 1 đến 56)

- Nhân viên part time (Linh, Kiệt, Giang, Hiếu) làm 10 ca 8 tiếng trong 4 tuần, biểu diễn dưới dạng công thức như sau:

$$\sum_{j=1}^{56} x_{i,j} = 10$$

b) Số lượng nhân viên mỗi ca:

- Số lượng nhân viên ca ngày:

$$\sum_{i=1}^{8} x_{i,j} = 3$$

- Số lượng nhân viên ca đêm:

$$\sum_{i=1}^{8} x_{i,j} = 1$$

Để tìm sắp xếp lịch biểu cho nhân viên, nhóm mình sẽ sử dụng phương pháp Simulated Annealing cho cả câu 3 và câu 4. Đây là một thuật toán metaheuristic lấy cảm hứng từ quá trình tôi luyện kim loại, nơi mà vật liệu được nung nóng và sau đó làm nguội để đạt được trạng thái cấu trúc thấp năng lượng hơn. Trong bối cảnh bài toán này, SA được sử dụng để tìm kiếm một giải pháp tối ưu hoặc gần tối ưu bằng cách cho phép "nhảy" qua các giải pháp kém hơn với xác suất giảm dần.

3. Lịch trình thoả mãn hai hard constraints ở trên

Đầu tiên, chúng ta sẽ tạo một lịch trình ngẫu nhiên chia ca cho nhân viên trong 28 ngày, trong đó mỗi nhân viên sẽ được phân công làm ca sáng và ca tối (lưu ý lịch trình ban đầu này chưa được tối ưu để thoả mãn các hard constraints)

```python
import numpy as np
import random

# Định nghĩa hàm initialize_schedule
def initialize_schedule():
    schedule = np.zeros((8, 28), dtype=int)
    for day in range(28):
        for _ in range(3):
            i = random.randint(0, 7)
            while schedule[i, day] != 0:
                i = random.randint(0, 7)
            schedule[i, day] = 1
        i = random.randint(0, 7)
        while schedule[i, day] != 0:
            i = random.randint(0, 7)
        schedule[i, day] = 2
    return schedule

# Khởi tạo lịch trình
initial_schedule = initialize_schedule()

# In lịch trình
print("Lịch trình ban đầu ngẫu nhiên:")
print(initial_schedule)

```

Kết quả sẽ hiện ra một lịch trình với 1 list lớn, trong đó gồm 8 list con lần lượt của An, Bình, Châu Dương, Linh, Kiệt, Giang, Hiếu. List này sẽ hiện ra kết quả ca được phân công trong 28 ngày của mỗi nhân viên:

```
Lịch trình ban đầu ngẫu nhiên:
[[0 1 0 0 2 0 0 0 0 0 1 1 0 0 0 2 1 0 1 0 2 1 0 1 2 1 0 1]
 [0 1 0 1 1 1 2 0 1 2 0 1 2 1 0 0 0 0 0 0 1 0 0 1 1 0 1 1]
 [1 0 1 1 0 1 1 0 1 1 2 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 2 0]
 [1 0 1 2 1 0 0 2 2 1 0 0 0 1 1 1 2 0 0 2 1 2 1 0 0 0 1 0]
 [2 2 2 0 0 1 0 1 0 1 0 0 1 0 2 1 0 1 0 1 0 0 2 0 0 1 0 1]
 [1 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 0 1 0 1 0 1 0 2 1 2 0 2]
 [0 0 0 0 1 2 1 0 0 0 1 1 0 2 1 0 1 0 2 1 1 0 1 0 0 0 1 0]
 [0 1 1 0 0 0 0 1 0 0 1 2 1 0 0 0 0 2 1 0 0 0 0 0 0 0 0 0]]
```

Sau đó, chúng ta sẽ xác định nhiệt độ ban đầu và tỉ lệ giảm nhiệt độ. Trong đó, nhiệt độ ban đầu sẽ quyết định phạm vi tìm kiếm của thuật toán, nhiệt độ cao thì thuật toán có khả năng thử nghiệm nhiều giải pháp hơn. Tỉ lệ giảm nhiệt độ sẽ quyết định tốc độ giảm nhiệt độ trong quá trình tìm kiếm. Tỉ lệ giảm thường nằm ở khoảng 0.8 đến 0.99 để tối ưu nhất. Trong bài toán này, chúng ta sẽ xác định nhiệt độ ban đầu là **5000** và tỉ lệ giảm là **0.97**.

Ngoài ra, chúng ta sẽ khởi tạo công thức tính chi phí (cost) của giải pháp. Mục đích của việc này là nhằm tìm kiếm giá trị tối ưu cho một hàm mục tiêu bằng cách khám phá các giải pháp khác nhau và dần dần điều chỉnh các tham số để cải thiện giá trị của hàm mục tiêu. Chi phí càng thấp thì giải pháp càng tối ưu. Trong bài toán này, hàm chi phí sẽ xác định trường hợp vi phạm các hard constraints. Với mỗi trường hợp vi phạm sẽ có "điểm phạt" cộng vào chi phí ban đầu.

```python
def cost(schedule):
    cost = 0
    # Kiểm tra số lượng nhân viên cho ca sáng và tối mỗi ngày
    for day in range(28):
        num_s = np.sum(schedule[:, day] == 1)
        num_t = np.sum(schedule[:, day] == 2)
        cost += abs(num_s - 3)  # Phạt số nhân viên ca sáng không đúng
        cost += abs(num_t - 1)  # Phạt số nhân viên ca tối không đúng

    # Kiểm tra số ca làm việc của từng nhân viên
    for i in range(8):
        num_s = np.sum(schedule[i, :] == 1)
        num_t = np.sum(schedule[i, :] == 2)
        if i < 4:  # Nhân viên toàn thời gian
            cost += abs(num_s + num_t - 18)  # Phạt số ca không đúng
        else:  # Nhân viên bán thời gian
            cost += abs(num_s + num_t - 10)  # Phạt số ca không đúng

    return cost
```

Sau khi xác định xong hàm chi phí, chúng ta sẽ tạo **neighbor solution** *(giải pháp lân cận)* nhằm giúp thuật toán khám phá không gian tìm kiếm và tìm kiếm các giải pháp tốt hơn. Chúng ta sẽ hoán đổi phân công ca trực trong một ngày của hai nhân viên ngẫu nhiên do hệ thống lựa chọn.

Khi nhiệt độ hiện tại lớn hơn mức nhiệt độ dừng (stopping temparature), mà ở trong bài toán này là **0.1**, ta sẽ tính chi phí của giải pháp lân cận để so sánh với chi phí của giải pháp hiện tại. Nếu giải pháp lân cận có chi phí thấp hơn (tốt hơn) so với giải pháp hiện tại, thì chấp nhận giải pháp lân cận. Nếu giải pháp lân cận có chi phí cao hơn, quyết định có chấp nhận giải pháp lân cận hay không dựa trên một xác suất, tùy thuộc vào nhiệt độ hiện tại (xác suất được xác định bằng công thức *exp*(*−delta _ cost* / *temperature*) trong đó *delta_cost* là hiệu của chi phí giải pháp lân cận và giải pháp hiện tại). Nếu được chấp nhận, giải pháp hiện tại sẽ được thay bằng giải pháp lân cận. Còn nếu bị từ chối, ta sẽ tiếp tục tạo giải pháp lân cận mới cho đến khi nhiệt độ giảm đến mức dừng đã định.

Sau khi cho code chạy, ta sẽ có nhiều kết quả lịch trình khác nhau. Chú ý nên cho file chạy nhiều lần để có phương án tối ưu nhất. Dưới đây là một lịch trình mẫu cụ thể với chi phí tốt nhất là **18**:

Nhân viên	5/8	6/8	7/8	8/8	9/8	10/8	11/8	12/8	13/8	14/8	15/8	16/8	17/8	18/8	19/8	20/8	21/8	22/8	23/8	24/8	25/8	26/8	27/8	28/8	29/8	30/8	31/8	1/9
An	0	1	1	0	1	1	1	2	0	0	1	0	1	2	1	1	0	1	2	1	0	1	0	0	0	0	0	1
Bình	1	0	1	1	1	1	0	1	0	0	0	1	0	1	0	0	2	2	0	1	1	2	1	1	1	1	0	0
Châu	0	0	0	2	1	0	1	0	1	2	0	1	1	1	1	1	1	0	0	1	1	0	0	1	1	0	1	0
Dương	0	1	0	0	0	0	1	0	0	1	1	0	0	1	2	2	0	0	1	2	0	0	2	0	2	2	2	2
Linh	2	0	2	1	0	2	0	0	1	0	0	2	1	0	1	0	1	0	0	0	2	0	0	1	1	0	0	0
Kiệt	1	0	1	0	2	0	0	0	2	1	2	0	2	0	0	0	0	1	1	0	0	1	0	0	0	1	1	0
Giang	0	1	0	0	0	1	2	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	0	1
Hiếu	1	2	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0	1	1	0	1	1	1	2	0	0	1	1

4. Hướng giải bài toán khi có thêm 2 soft constraints mới

Dựa theo dữ kiện đã cho ở đề bài, ta có thể viết công thức mô tả các soft constraints như sau:

i) Nhân viên full-time làm 4-5 ca một tuần, nhân viên part-time là 2-3 ca một tuần:

$$FT: 4 \le \sum_{j=7(k-1)+1}^{7k} x_{j,k} \le 5 \forall k \in \{1,2,3,4\}, \forall i \in \{1,2,3,4\}$$

$$PT: 2 \le \sum_{j=7(k-1)+1}^{7k} x_{i,j} \le 3 \forall k \in \{1,2,3,4\}, \forall i \in \{1,2,3,4\}$$

ii) Chuỗi ngày làm việc liên tiếp nhân viên full-time là từ 4 đến 6 ngày, nhân viên part-time là 2 đến 3 ngày:

$$\text{FT:consecutive\_day}(x_{i,j}) \in [4,6] \forall i \in \{1,2,3,4\}$$
$$\text{PT:consecutive\_day}(x_{i,j}) \in [2,3] \forall i \in \{1,2,3,4\}$$

Đầu tiên, chúng ta sẽ xây dựng hàm tính chi phí mới bao gồm cả các điểm phạt cho hard constraints và soft constraints:

i) Đối với hard constraints, về cơ bản công thức vẫn giữ nguyên so với bài 3. Tuy nhiên mỗi điểm phạt ta sẽ nhân với trọng số là 10.

ii) Đối với soft cost:

- Yêu cầu về số ca làm việc hàng tuần:
  + Đối với nhân viên full-time (FT):
    - Phạt nếu số ca ít hơn 4:

$$P_w^{FT} = c(4-s) \Leftrightarrow s < 4$$

    - Phạt nếu số ca nhiều hơn 5:

$$P_w^{FT} = c(s-5) \Leftrightarrow s > 5$$

  + Đối với nhân viên part-time (PT):
    - Phạt nếu số ca ít hơn 2:

$$P_w^{PT} = c(2-s) \Leftrightarrow s < 2$$

    - Phạt nếu số ca nhiều hơn 3:

$$P_w^{PT} = c(s-3) \Leftrightarrow s > 3$$

Trong đó: s là số ca làm việc trong tuần

c là hệ số phạt (ở bài giải dưới đây ta để là 1)

- Yêu cầu về chuỗi ngày làm việc tối đa liên tiếp:
  + Đối với nhân viên full-time (FT):
    - Phạt nếu chuỗi làm việc liên tiếp ít hơn 4 ngày:

$$P_c^{FT} = d(4 - d_c) \Leftrightarrow d_c < 4$$

- Phạt nếu chuỗi làm việc liên tiếp nhiều hơn 6 ngày:

$$P_c^{FT} = d(d_c - 6) \Leftrightarrow d_c > 6$$

+ Đối với nhân viên part-time (PT):

- Phạt nếu chuỗi làm việc liên tiếp ít hơn 2 ngày:

$$P_c^{FT} = d(2 - d_c) \Leftrightarrow d_c < 2$$

- Phạt nếu chuỗi làm việc liên tiếp nhiều hơn 3 ngày:

$$P_c^{FT} = d(d_c - 3) \Leftrightarrow d_c > 3$$

Trong đó: d_c là số ngày làm việc liên tiếp

d là hệ số phạt, tương tự như c (trong bài toán dưới đây là 1)

Tổng chi phí của soft constraints:

$$\text{Total Soft Cost} = \sum_{i=1}^{N} (P_w^{(i)} + P_c^{(i)})$$

Trong đó: $P_w^{(i)}$: chi phí phạt cho nhân viên thứ iii về số ca làm việc hàng tuần.

$P_c^{(i)}$: chi phí phạt cho nhân viên thứ iii về số ca làm việc hàng tuần.

Tổng chi phí của giải pháp:

Total Cost = Hard Cost + Total Soft Cost

Sau khi xây dựng xong hàm chi phí, chúng ta sẽ tiếp tục sử dụng phương pháp *Simulated Annealing* bằng cách xây dựng giải pháp lân cận, so sánh với giải pháp hiện tại hoặc dựa vào nhiệt độ để tìm ra giải pháp tối ưu nhất. Quy tắc giải thuật bằng Python được triển khai tương tự như câu 3.

Nhân viên	5/8	6/8	7/8	8/8	9/8	10/8	11/8	12/8	13/8	14/8	15/8	16/8	17/8	18/8	19/8	20/8	21/8	22/8	23/8	24/8	25/8	26/8	27/8	28/8	29/8	30/8	31/8	1/9

An	0	1	2	0	0	1	1	1	1	2	1	0	0	0	0	1	1	1	2	1	0	1	1	1	1	0	0	0
Bình	0	1	1	1	1	0	0	1	2	0	0	1	1	0	1	1	0	1	0	0	1	0	0	1	0	1	0	1
Châu	0	0	0	1	0	2	2	0	0	0	1	0	1	2	1	0	1	2	1	0	0	0	1	0	2	0	1	0
Dương	0	0	1	2	1	0	0	1	0	0	2	1	2	0	2	1	0	0	0	1	1	1	2	0	1	0	0	1
Linh	1	2	1	0	1	0	0	0	0	1	1	2	1	0	1	0	0	0	0	2	2	2	0	0	0	0	2	0
Kiệt	2	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	2	0	0	1	1	0	0	0	0	2	1	1
Giang	1	0	0	1	0	1	1	0	0	1	0	1	0	1	0	2	0	0	1	0	0	1	1	1	1	1	1	0
Hiếu	1	1	0	0	2	1	1	2	1	0	0	0	0	1	0	0	1	1	1	0	0	0	0	2	0	1	0	2

Mỗi lần cho code chạy ta sẽ có nhiều kết quả lịch trình khác nhau. Chú ý nên cho file chạy nhiều lần để có phương án tối ưu nhất. Sau khi cho file chạy thử 150 lần, dưới đây là một lịch trình mẫu cụ thể với chi phí tốt nhất là **29**:

Để đánh giá lời giải này tốt hơn lời giải khác, ta sẽ dựa theo tiêu chí khách quan là chi phí tốt nhất (best cost). Chi phí càng thấp thì lời giải càng tốt. Tuy nhiên, chúng ta cũng có thể cân nhắc một số các tiêu chí chủ quan sau:

- **Tính công bằng**: Giữa các nhân viên có sự phân công công bằng. Không có nhân viên nào bị sắp xếp quá sát lịch hay quá thoáng.

- **Năng lực của nhân viên:** có thể bố trí lịch phân công dựa theo năng lực để mỗi ca đều có những nhân viên có năng lực tốt.

I. TÀI LIỆU THAM KHẢO

1. P. De Bruecker, J. Van den Bergh, J. Beli n, E. Demeulemeester, Workforce planning incor-porating skills: State of the art, European Journal of Operational Research, 243(1), 1-16,2015.
2. R. Soto, B. Crawford, E. Monfroy, W. Palma, F. Paredes, Nurse and paramedic rosteringwith constraint programming: A case study, Romanian Journal of Information Science andTechnology, 16(1), 52-64, 2013