**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**


# BÁO CÁO

## MÔN HỌC: TRÍ TUỆ NHÂN TẠO

**ĐỀ TÀI:**

## Áp dụng thuật toán Value Iteration và Policy Iteration cho một số trò chơi trên OpenAI Gym

**Lớp**: CS106.L21.KHCL

**Giảng viên hướng dẫn**:

TS. Lương Ngọc Hoàng

**Sinh viên thực hiện**:

Phan Nguyễn Thành Nhân 19521943

Thành phố Hồ Chí Minh, ngày 07 tháng 06 năm 2021

# Mục lục

# I. Giới thiệu thuật toán Value Iteration và Policy Iteration

## 1. Thuật toán Value Iteration

- Ý tưởng chính của thuật toán Value Iteration là tìm chiến lược tối ưu dựa trên giá trị của chiến lược tối ưu.
- Thuật toán Value Iteration bao gồm 2 vòng lặp:
  + Value iteration: đầu tiên, thuật toán sẽ tạo khởi tạo giá tri v bằng 0 cho mọi trạng thái. Sau đó thuật toán sẽ quét qua từng trạng thái, với mỗi trạng thái thuật toán sẽ quét qua từng chiến lược, lúc này thuật toán sẽ tính giá trị q bằng công thức $Q^*(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^*_{t-1}(s')]$ cho mỗi chiến lược. Sau khi tính giá q xong, ứng với mỗi trạng thái, thuật toán sẽ cập nhập giá trị v bằng cách chọn giá trị q lớn nhất trong mỗi chiến lược $V^*_t(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^*_{t-1}(s')]$. Vòng lặp kết thúc khi giá trị v hội tụ.
  + Policy extraction: sau khi tính giá trị của chiến lược tối ưu xong, lúc này ta sẽ có thể tìm được chiến lược tối ưu bằng cách tính giá trị q cho mỗi chiến lược ứng với mỗi trạng thái, ta sẽ chọn chiến lược có giá trị q bằng với giá trị v của trạng thái đó, và chiến lược cũng là chiến lược có giá trị q lớn nhất cũng là chiến lược tối ưu cho trạng thái đó.

    $\pi^*(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^*(s')]$

## 2. Thuật toán Policy Iteration

- Nhận thấy thuật toán Value Iteration có nhiều vấn đề bất cập, như:
  + Tốc độ tìm ra chiến lược tối ưu chậm vì độ phức tạp lớn: $O(s^2a)$ với mỗi vòng lặp.
  + Giá trị 'max' ở mỗi trạng thái hiếm khi thay đổi.
  + Chiến lược thường sẽ hội tụ trước khi giá trị v hội tụ.

- Do đó thuật toán Policy Iteration được đề ra để giải quyết những vấn đề của thuật toán Value Iteration.
- Thuật toán Policy Iteration sẽ bao gồm 2 vòng lặp:
  + Policy evaluation: đầu tiên, thuật toán sẽ khởi tạo một chiến lược ngẫu nhiên cho mỗi trạng thái. Sau đó sẽ dùng policy evaluation bằng công thức $V_t^{\pi}(s) \leftarrow \sum_{s'} P(s'|s,\pi(s))[R(s,\pi(s),s') + \gamma V_{t-1}^{\pi}(s')]$ để đánh giá cho chiến lược đó. Vòng lặp sẽ dừng lại cho đến khi giá trị v hội tụ, do đó độ phức tạp của Policy Iteration chỉ là $O(s^2)$ cho mỗi vòng lặp.
  + Policy improvement: sau khi đánh giá chiến lược xong, thuật toán sẽ quét qua từng trạng thái trong trò chơi. Tại mỗi trạng thái, thuật toán sẽ cập nhập chiến lược bằng cách chọn chiến lược có giá trị q lớn nhất $\pi_{i+1}(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^{\pi_i}(s')]$. Lúc này ta sẽ có một chiến lược mới, tiếp tục đánh giá và cải thiện chiến lược đó cho đến khi nào chiến lược hội tụ.

# II. Áp dụng thuật toán Value Iteration và Policy Iteration cho một số trò chơi trên OpenAI Gym

- Trong bài cáo này, chúng ta sẽ chọn 3 trò chơi trên OpenAI Gym bao gồm FrozenLake-v0, FrozenLake8x8-v0, Taxi-v3 để áp dụng thuật toán Value Iteration và PolicyIteration.
- Đầu tiên chúng ta sẽ khởi tạo môi trường cho cả 3 trò chơi, sau đó chúng ta sẽ kiểm tra số trạng thái và số hành động của mỗi trò chơi.

```
env_F4x4_v0 = gym.make('FrozenLake-v0')
```

```
env_F4x4_v0.observation_space.n
```

```
16
```

```
env_F4x4_v0.action_space.n
```

```
4
```

```
env_F8x8_v0 = gym.make('FrozenLake8x8-v0')
```

```
env_F8x8_v0.observation_space.n
```

```
64
```

```
env_F8x8_v0.action_space.n
```

```
4
```

```
env_taxi_v3 = gym.make('Taxi-v3')
```

```
env_taxi_v3.observation_space.n
```

```
500
```

```
env_taxi_v3.action_space.n
```

```
6
```

- Khi ta có môi trường của cả 3 trò chơi, chúng ta sẽ bắt đầu áp dụng 2 thuật toán cho mỗi trò chơi.

## 1. Áp dụng thuật toán Value Iteration

- Đầu tiên đã sẽ gọi hàm value iteration với đầu vào là env, max_iters và gamma để tính giá trị chiến lược tối ưu cho mỗi trạng thái.

```
v_values_F4x4_v0, v_exe_time_F4 = value_iteration(env_F4x4_v0, max_iters=1000, gamma=0.9)
```

```
v_values_F8x8_v0, v_exe_time_F8 = value_iteration(env_F8x8_v0, max_iters=1000, gamma=0.9)
```

```
v_values_taxi_v3, v_exe_time_T3 = value_iteration(env_taxi_v3, max_iters=1000, gamma=0.9)
```

- Sau khi hàm value_iteration thực hiện xong, hàm sẽ trả về giá trị chiến lược tối ưu cho mỗi trạng thái và thời gian thực hiện.

```
v_values_F4x4_v0
```

```
array([0.06888615, 0.06141054, 0.07440682, 0.05580409, 0.09185022,
       0.        , 0.11220663, 0.        , 0.14543286, 0.2474946 ,
       0.29961593, 0.        , 0.        , 0.3799342 , 0.63901926,
       0.        ])
```

```
v_exe_time_F4
```

```
0.06053519248962402
```

```
v_values_F8x8_v0
```

```
array([0.00641104, 0.00854808, 0.01230044, 0.01778942, 0.02508214,
       0.03247089, 0.03957134, 0.04297844, 0.00602405, 0.00764512,
       0.01091162, 0.01642654, 0.02605411, 0.03619409, 0.0493547 ,
       0.05730461, 0.00509024, 0.0058532 , 0.00677534, 0.        ,
       0.02557084, 0.03882139, 0.06763973, 0.08435607, 0.0042256 ,
       0.00476954, 0.00581968, 0.0078541 , 0.02036065, 0.        ,
       0.09175501, 0.12919111, 0.00318093, 0.00319659, 0.00270488,
       0.        , 0.0344439 , 0.06195145, 0.10901921, 0.20969093,
       0.00186915, 0.        , 0.        , 0.01085079, 0.03250092,
       0.06304172, 0.        , 0.36008773, 0.00118046, 0.        ,
       0.00137717, 0.00366839, 0.        , 0.11568671, 0.        ,
       0.63051379, 0.00088531, 0.00077462, 0.00092218, 0.        ,
       0.13824885, 0.32258065, 0.61443932, 0.        ])
```

```
v_exe_time_F8
```

```
0.30692100524902344
```

```
v_values_taxi_v3
```

```
array([ 89.47323891,  32.81971401,  55.26423891,  37.57755845,
         8.43222921,  32.81971401,   8.43222921,  15.28447953,
        32.81971401,  18.09386122,  55.26423891,  21.2154998 ,
        12.75594298,  18.09386122,  12.75594298,  37.57755845,
       100.52591945,  37.57755845,  62.51591945,  42.86394891,
        79.52591945,  28.53774704,  48.73781945,  32.81971401,
```

```
v_exe_time_T3
```

```
1.6266841888427734
```

- Khi đã có giá trị chiến lược tối ưu của mỗi trạng thái, chúng ta sẽ tìm chiến lược cho mỗi trạng thái bằng hàm policy_extraction.

```
v_policy_F4x4_v0 = policy_extracion(env_F4x4_v0, v_values_F4x4_v0, gamma=0.9)
print(v_policy_F4x4_v0)
```

```
[0 3 0 3 0 0 0 0 3 1 0 0 0 2 1 0]
```

```
v_policy_F8x8_v0 = policy_extracion(env_F8x8_v0, v_values_F8x8_v0, gamma=0.9)
print(v_policy_F8x8_v0)
```

```
[3 2 2 2 2 2 2 2 3 3 3 3 2 2 2 1 3 3 0 0 2 3 2 1 3 3 3 1 0 0 2 1 3 3 0 0 2
 1 3 2 0 0 0 1 3 0 0 2 0 0 1 0 0 0 0 2 0 1 0 0 1 1 1 0]
```

```
v_policy_taxi_v3 = policy_extracion(env_taxi_v3, v_values_taxi_v3, gamma=0.9)
print(v_policy_taxi_v3)
```

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

## 2. Áp dụng thuật toán Policy Iteration

- Đối với thuật toán chúng ta chỉ cần gọi hàm policy_iteration, hàm sẽ trả về cho ta chiến lược tối ưu, vì trong hàm policy_iteration đã gọi hàm policy_evaluation để đánh giá chiến lược và sau đó gọi hàm policy_improvement để cải thiện chiến lược.
- Kết quả sau khi chạy hàm policy_iteration:

```
p_policy_F8x8_v0, p_exe_time_F8 = policy_iteration(env_F8x8_v0, max_iters=1000, gamma=0.9)
print(p_policy_F8x8_v0)
```

```
Policy evaluation convergence at 1-th iteration
Policy evaluation convergence at 28-th iteration
Policy evaluation convergence at 92-th iteration
Policy evaluation convergence at 93-th iteration
Policy evaluation convergence at 91-th iteration
Policy evaluation convergence at 93-th iteration
Policy evaluation convergence at 96-th iteration
Policy evaluation convergence at 101-th iteration
Policy evaluation convergence at 113-th iteration
Policy evaluation convergence at 118-th iteration
-----------------------------------
Policy iteration converge at 10-th iteration
[3 2 2 2 2 2 2 2 3 3 3 3 2 2 2 1 3 3 0 0 2 3 2 1 3 3 3 1 0 0 2 1 3 3 0 0 2
 1 3 2 0 0 0 1 3 0 0 2 0 0 1 0 0 0 0 2 0 1 0 0 1 1 1 0]
```

```
p_policy_F4x4_v0, p_exe_time_F4 = policy_iteration(env_F4x4_v0, max_iters=1000, gamma=0.9)
print(p_policy_F4x4_v0)
```

```
Policy evaluation convergence at 23-th iteration
Policy evaluation convergence at 29-th iteration
Policy evaluation convergence at 63-th iteration
Policy evaluation convergence at 80-th iteration
Policy evaluation convergence at 81-th iteration
-----------------------------------
Policy iteration converge at 5-th iteration
[0 3 0 3 0 0 0 0 3 1 0 0 0 2 1 0]
```

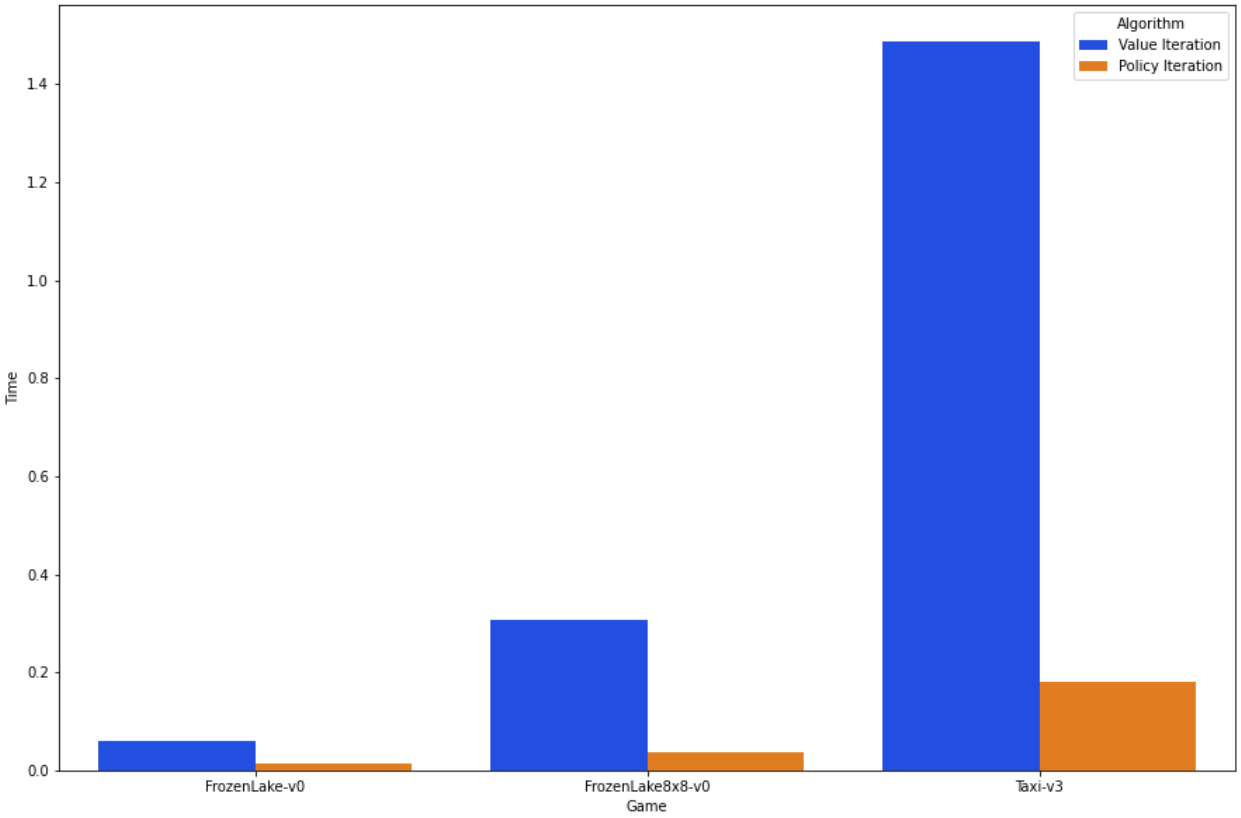
```
p_policy_taxi_v3, p_exe_time_T3 = policy_iteration(env_taxi_v3, max_iters=1000, gamma=0.9)
print(p_policy_taxi_v3)
```

```
Policy evaluation convergence at 95-th iteration
Policy evaluation convergence at 100-th iteration
Policy evaluation convergence at 103-th iteration
Policy evaluation convergence at 104-th iteration
Policy evaluation convergence at 105-th iteration
Policy evaluation convergence at 110-th iteration
Policy evaluation convergence at 111-th iteration
Policy evaluation convergence at 112-th iteration
Policy evaluation convergence at 113-th iteration
Policy evaluation convergence at 116-th iteration
Policy evaluation convergence at 117-th iteration
Policy evaluation convergence at 117-th iteration
Policy evaluation convergence at 117-th iteration
Policy evaluation convergence at 117-th iteration
Policy evaluation convergence at 117-th iteration
Policy evaluation convergence at 117-th iteration
Policy evaluation convergence at 117-th iteration
Policy evaluation convergence at 117-th iteration
----------------------------------
Policy iteration converge at 18-th iteration
[4 4 4 4 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 3 3 3 3 0 0 0 0 0 0 0 0 0 0 0 3
 0 0 0 0 0 0 0 2 2 2 2 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 2 2 2 2 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 4 4 4 4 0 0 0 0 0 0 0 0 0 5 0 0 1 1 1 1 0 0 0 0 0 0
 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1
 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 2 2 2 2 0 0 0 0 2 2 2 2 1 2 0 2 1 1
 1 1 2 2 2 2 3 3 3 3 2 2 2 2 1 2 3 2 3 3 3 3 1 1 1 1 3 3 3 3 2 2 2 2 3 1 3
 2 3 3 3 3 1 1 1 1 3 3 3 3 0 0 0 0 3 1 3 0 3 3 3 3 1 1 1 1 3 3 3 3 0 0 0 0
 3 1 3 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 0 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 1 1 1
 1 4 4 4 4 1 1 1 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 1 1 1 5 1
 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 1 1 1 3]
```

# III. So sánh thuật toán Value Iteration và Policy Iteration với 3 trò chơi

- Sau khi áp dụng 2 thuật toán Value Iteration và Policy Iteration cho 3 trò chơi, chúng ta sẽ nhận được thời gian chạy của 2 thuật toán ứng với mỗi trò chơi.

| | FrozenLake-v0 | FrozenLake8x8-v0 | Taxi-v3 |
|---|---|---|---|
| Value Iteration | 0,06 | 0,3 | 1,62 |
| Policy Iteration | 0,01 | 0,03 | 0,18 |




- Khi chúng ta vẽ biểu đồ so sánh thời gian chạy, chúng ta nhận thấy thời gian thực hiện của thuật toán Policy Iteration nhanh hơn đáng kể so với thuật toán Value Iteration, bởi vì độ phức tạp của thuật toán Policy Iteration chỉ là $O(s^2)$ trong khi đó của Value Iteration là $O(s^2a)$.

# IV. Nhận xét chung

- Nhìn nhận chung, ta thấy thuật toán Policy Iteration tốt hơn so với Value Iteration; vì Policy Iteration vừa đảm bảo tìm được chiến lược tối ưu để giải quyết bài toán, vừa đảm bảo được về mặt thời gian thực hiện.
- Vì vậy, Policy Iteration sẽ thuật toán tốt để thay thế Value Iteration.