Nguyễn Lê Quang

19120121

**HEXWORLD PROBLEMS**

1. phát biểu bài toán

-Giới thiệu bài toán Hex World:

  ⏺Mỗi ô trong bản đồ đại diện cho 1 trạng thái và số trạng thái là hữu hạn

  ⏺Từ một trạng thái ta có thể di chuyển theo 6 hướng ( hướng theo 6 cạnh của lục giác) để chuyển đổi trạng thái nếu hướng đi đó khả thi (trường hợp đụng tường hoặc không có ô bên cạnh thì không thể di chuyển đến)

  ⏺Quá trình chuyển đổi hành động là hoàn toàn ngẫu nhiên

  ⏺Tại một số ô nhất định sẽ có phần thưởng riêng đặc biệt được quy định bởi 1 giá trị(có thể âm hoặc dương).

- Bài toán Hex World

  ⏺Với tập dữ liệu đầu vào bao gồm:

  +Tập các trạng thái S  (state)

  +Tập các phần thưởng R (reward) tương ứng ở mỗi trạng thái

  ⏺Vấn đề cần giải quyết là quá trình quyết định cách mà tác nhân (agent) thực hiện các hành động ở mỗi một trạng thái để nhận được phần thưởng nhiều nhất có thể và tránh các phần thưởng âm trong bản đồ.

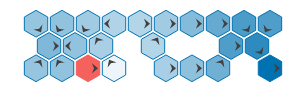
⏺Dữ liệu đầu ra: tập các hành động ứng với mỗi trạng thái

-Minh họa về bài toán



Hình 1

⏺ Hình 1 trên là một bản đồ hex world với 24 trạng thái, 2 phần thưởng dương là 5,10 và 1 phần thưởng âm là -10.



Hình 2

⏺ Hình 2 trên là một bản đồ hex world sau khi chạy xong thuật toán và kết quả trả về sẽ cho ta biết hướng đi của từng trạng thái trong 24 trạng thái ban đầu.

1. thách thức

-Tập dữ liệu S,R lớn

-Tập dữ liệu S gồm các trạng thái không thể chuyển đổi nhau bằng bất cứ hành động nào ( không có 2 ô nào có chung cạnh)

-Thiếu tập kiểm định

1. thực nghiệm
2. Mô hình tính toán
   1. MDP (Markov Decision Process)

|  |  |  |
| --- | --- | --- |
| Tên | Biến | Mô tả |
| Tập trạng thái (State) | S | Tập các trạng thái (vị trí) khả thi của  agent |
| Tập hành động (Action) | A | Tập các hành động (hướng đi ) khả thi của 1 agent |
| Hàm chuyển tiếp (Transition) | T(s’|s,a) | xác suất chuyển tiếp từ  trạng thái s sang trạng thái s’ bởi hành động a |
| Hàm phần thưởng (Reward) | R(s,a) | Phần thưởng có được khi thực hiện hành động a từ trạng thái s |
| Hệ số chiếc khấu (discounted rate) | γ ∊ [0, 1] | Hệ số kiểm soát mức độ quan tâm phần thưởng  Nếu γ→0: agent chỉ quan tâm đến phần thưởng trước mắt  Nếu γ→1: agent quan tâm đến phần thưởng trong tương lai |

* 1. DiscreteMDP

|  |  |  |
| --- | --- | --- |
| Tên | Biến | Mô tả |
| Hàm chuyển tiếp (Transition) | T(s’|s,a) | xác suất chuyển tiếp từ  trạng thái s sang trạng thái s’ bởi hành động a |
| Hàm phần thưởng (Reward) | R(s,a) | Phần thưởng có được khi thực hiện hành động a từ trạng thái s |
| Hệ số chiếc khấu (discounted rate) | γ ∊ [0, 1] | Hệ số kiểm soát mức độ  quan tâm phần thưởng |

* 1. HexworldMDP

|  |  |  |
| --- | --- | --- |
| Tên | Biến | Mô tả |
| Tập các tọa độ | hexes | Các vị trí mà agent có thể đứng |
| DiscreteMDP | mdp | Mô hình DiscreteMDP |
| Tập các giá trị phần thưởng | special\_hex\_rewards | Giá trị phần thưởng tại các ô tương ứng có trong tập, còn lại mặc định bằng 0 |

Sử dụng mô hình Markov Decision Process để giải quyết bài toán hexworld Với tập các Transition là một cấu trúc dữ liệu T (transition) là mảng 3 chiều còn R (rewards) là mảng 2 chiều. Giá trị của T nằm trong khoảng [0,1] còn giá trị R là 1 số thực tùy ý.

1. Phương pháp giải quyết

: chính sách đưa ra quyết định hành động tại trạng thái s

: hàm giá trị đánh giá ứng với chính sách

1. Policy evaluation

Chúng ta sẽ đánh giá giá trị của **value function** theo 2 cách sau:

**Cách 1:**  Sử dụng vòng lặp

Giá trị U: Với mỗi trạng thái s, hành động a, sử dụng để tính

Cách 1: Sử dụng vòng lặp

-Giá trị U : Với mỗi trạng thái s, hành động a

🞾

🞾

-Lặp k\_max lần để tính ra giá trị của : kết quả trả về là 1 mảng các giá trị ứng với mỗi hành động tại một trạng thái s tương ứng.

**Cách 2**: Sử dụng ma trận (One-step learning)

Chúng ta có thể tính toán bằng cách không sử dụng vòng lặp nhiều lần. Mô hình tính toán được thể hiện như sau:

Trong đó , lần lượt là hàm giá trị U và hàm phần thưởng được thể hiện bằng một mảng có |S| phần tử. Còn là 1 ma trận gồm có |S|x|S| phần tử trong đó là xác suất khi chuyển từ trạng thái thứ i sang trạng thái thứ j.

Phương trình trên có thể được biến đổi như sau:

(\*)

Từ phương trình (\*) trên ta có thể tính dựa trên T,R, chỉ 1 lần nhưng độ phức tạp thời gian yêu cầu lên đến O(|S|3)

1. Policy improvement

Chúng ta sử dụng và policy để tối ưu hóa cực đại chính sách đường đi dựa theo U.

1. Optimal policy

Còn một cách khác để tính chính sách đi (policy) là sử dụng action value function hay còn gọi là *Q-function .* Hàm này sẽ trả về 1 giá trị khi đang đứng tại trạng thái s mà đi theo hành động a và vẫn tiếp tục sử dụng thuật toán tham lam với Q.

Từ hàm Q ở trên ta có thể tính value function U(s) như sau:

Do đó, hàm policy sẽ được tính như sau:

Khi lưu trữ Q thì ta cần đến O(|S|x|A|) không gian lưu trữ thay vì O(|S|) không gian lưu trữ như U, nhưng thay vì đó chứng ta sẽ không cần trích xuất T và R để tính được policy

1. Policy Iteration

Sử dụng vòng lặp để được giá trị hội tụ của  policy.Sau mỗi vòng lặp sẽ cho ra một giá trị policy mới.Kết quả trả về sẽ là một policy tương ứng cho các trạng thái trong hexworld.

Mã giả:

for k = 1:M.k\_max

U = policy\_evaluation(𝒫, π)      // tính giá trị value function U

π′ = ValueFunctionPolicy(𝒫, U)  // tính chính sách

if all(π(s) == π′(s) for s in 𝒮)   // điều kiện hội tụ

break

end

π = π′

end

return π

1. Code
2. Flow chính

HexWorld()::**HexworldMDP**

Solve(mdp,policy,k\_max)

createMDP(mdp::HexworldMDP)::**MDP**

policy::**ValueFunctionPolicy**

-Funcion HexWorld(): dùng để khởi tạo ra các giá trị như : trạng thái,phần thưởng,hệ số chiếc khấu,…và trả về 1 kiểu HexWorldMDP đã được định nghĩa từ trước

-Function createMDP(mdp::HexWorldMDP) : dùng để ép kiểu 1 HexworldMDP về thành 1 MDP

- ValueFunctionPolicy policy: Đây là 1 biến được khởi tạo với constructor gồm 2 tham số là MDP và U (ban đầu sẽ được khởi tạo bằng một mảng các phần tử 0), đây là 1 chính sách giúp cho thuật toán đưa ra hành động tại mỗi trạng thái ( có 6 hành động tất cả). policy(s) là hàm trả về chính sách tại trạng thái s.

-Function Solve(mdp::MDP,policy,k\_max) : đây là hàm xử lý chính để đưa ra kết quả của bài toán, kết quả trả về là 1 mảng các hành động sao cho mỗi hành động tương ứng với một trạng thái trong hexworld.

1. Hàm hỗ trợ

Lookahead(mdp,U,s,a)

greedy(mdp,U,s)

policy(s)

-Function lookahead(mdp,U,s,a) : trả về giá trị là 1 số thực được tính toán dựa trên giá trị reward,transition và U.

-Function greedy(mdp,U,s) : trả về giá trị là 1 tuple gồm 2 giá trị là u(max) và hành động tương ứng với hàm hỗ trợ là lookahead

-Function policy(s) : trả về kết quả là hành động (1 trong 6 hướng đi) tại trạng thái s với hàm hỗ trợ là greedy

iterative\_policy\_evaluation(mdp,policy,k\_max)

lookahead(mdp,U,s,a)

policy(s)

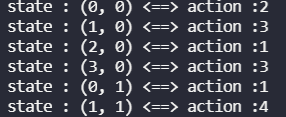
Solve(mdp,policy,k\_max)

-Function iterative\_policy\_evaluation (mdp,policy,k\_max): kết quả trả về là 1 mảng U sau khi lặp k\_max lần với policy tương ứng với 2 hàm hỗ trợ là lookahead và policy(s) đã nói ở trên

-Function Solve(mdp,policy,k\_max) kết quả trả về là 1 mảng các hành động thông qua việc gọi hàm iterative\_policy\_evaluation

1. Thực thi

-Để chạy code, ta vào file “./src/hexworld/main.jl” để chạy. (click vào hình tam giác bên góc phải trên để chạy). Kết quả trả ra là các hành động được đánh số từ 1 đến 6 ứng với mỗi trạng thái được lưu dưới dạng tọa độ (x,y)



4

3

(x,y)

5

2

6

1

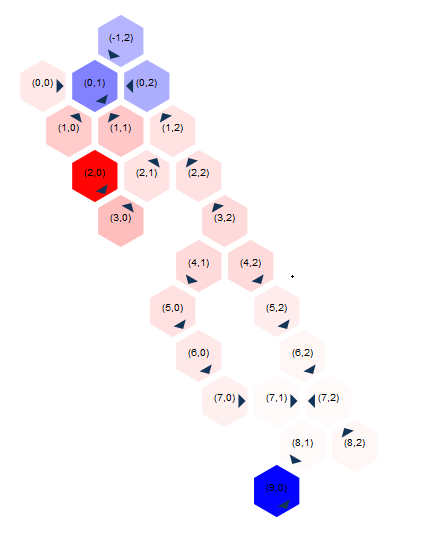
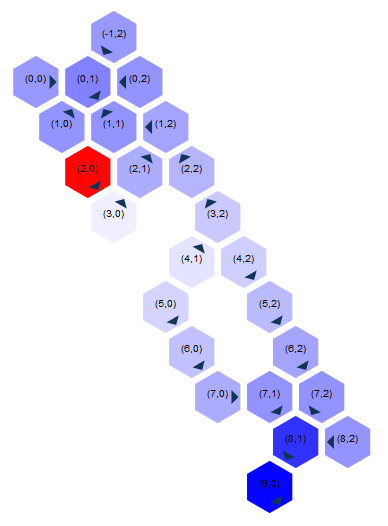
+Các số tương ứng với các hướng đi tại trạng thái (x,y)

-Các hình ảnh visualize thì được lưu tại folder “src\hexworld\visualization” dưới dạng .png

\*Lưu ý: import package Luxor để vẽ hình

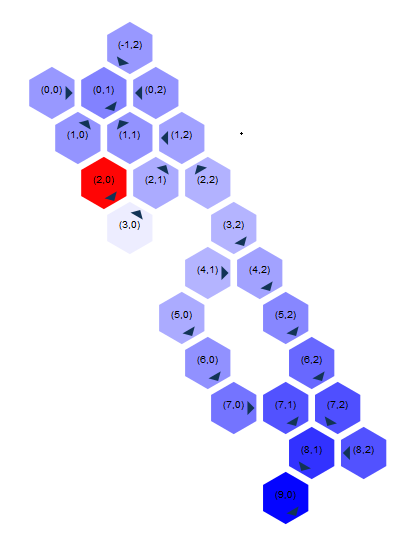
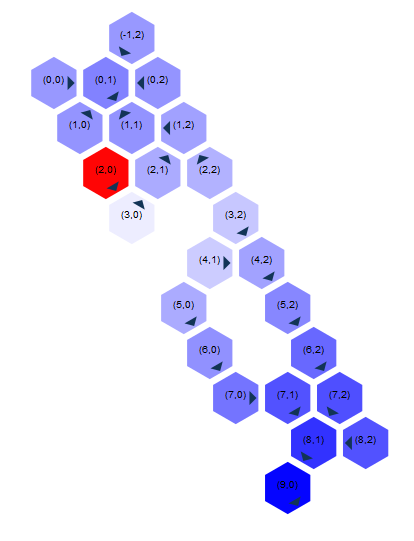
1. Visualization

Kết quả nhận được sau 4 vòng lặp tại hàm Solve như sau:



Iteration 2

Iteration 1



Interation 4

Interation 3

1. Phân tích

-Kết quả trả về đúng với kết quả mong muốn.

-Thuật toán chạy đúng và hội tụ với số vòng lặp thấp

1. tóm tắt kết quả
2. Điểm mạnh:

-Hiểu được mô hình thuật toán hexworld

-Tìm ra được policy để giải quyết bài toán

-Visualize được kết quả mô hình chạy của thuật toán

-Phân chia code thành nhiều file thuận tiện cho việc cải tiến hoặc chỉnh sửa sau này

1. Điểm yếu:

-Policy định hướng chưa thật sự tốt để tìm ra được phần thưởng tốt nhất.

Vd: input phần thưởng có là 5 và 10 thì các ô lân cận 5 thường sẽ tiến về 5 hơn là so với việc tìm đường đi hướng đến 10 là giá trị phần thưởng cao hơn

1. Tài liệu tham khảo

Code : <https://github.com/algorithmsbooks/DecisionMakingProblems.jl>

Document: Algorithms for decision making .pdf (sách thầy Bắc)

Luxor: <https://juliagraphics.github.io/Luxor.jl/stable/>