ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ASSIGNMENT 4

HỌC PHẦN: TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI: EVALUATION FUNCTIONS FOR MINIMAX/ALPHABETA/EXPECTIMAX PACMAN

Giảng viên hướng dẫn: Lương Ngọc Hoàng

Lớp: Trí tuệ nhân tạo - CS106.O22

Sinh viên thực hiện: Nguyễn Vũ Khai Tâm – 22521293

TP. Hồ Chí Minh, tháng 5, năm 2023

I. Thiết kế hàm evaluation function mới

Mục tiêu của trò chơi là làm sao để có thể giúp Pacman đạt được điểm số cao nhất có thể và ăn hết toàn bộ food để kết thúc trò chơi. Mỗi hành động mà Pacman thực hiện sẽ ảnh hưởng đến điểm số hiện tại và điểm kết thúc của trò chơi, do đó ta cần thiết kế hàm lượng giá dựa trên các kết quả mà các hành động này mang lại.

1. Ý tưởng sử dụng MST (Minimum Spanning Tree)

- Cây khung nhỏ nhất (MST) là một khái niệm trong lý thuyết đồ thị, dùng để tìm một tập hợp các cạnh trong một đồ thị liên thông, không có hướng sao cho chúng tạo thành một cây bao gồm tất cả các đỉnh, và tổng trọng số (trong trường hợp này là khoảng cách) của các cạnh là nhỏ nhất. Trong bối cảnh của Pacman, việc tính toán MST giúp xác định đường đi tối ưu nhất để thu thập tất cả các điểm thức ăn còn lại trên bản đồ, giảm thiểu khoảng cách tổng thể mà Pacman cần di chuyển.
- Ứng dụng của MST (em được một anh khóa trên gợi ý ý tưởng này): Tính toán hiệu quả khoảng cách di chuyển: Khi Pacman cần thu thập nhiều điểm thức ăn, việc sử dụng MST cho phép tính toán một lộ trình gần như tối ưu để thu thập tất cả mà không cần đi quá xa từ một điểm này sang điểm khác.

2. Các Đặc Trưng Được Sử Dụng trong Hàm Đánh Giá

Hàm đánh giá mới *betterEvaluationFunction2*:

- Tổng quát:

+ Khi có ma sợ hãi, ta có công thức của hàm đánh giá như sau:

```
if len(foodList) <= 2:
    return currentScore + 200/distToFood + 200/distToScaredGhost - 50/distToNonScaredGhost
else:
    return currentScore + 1000/distToFood + 200/distToScaredGhost - 50/distToNonScaredGhost
```

+ Khi không có ma sợ hãi, ta có công thức của hàm đánh giá như sau:

```
return currentScore + 1000/distToFood - 50/distToNonScaredGhost + 100/distanceToCapsules
```

_ Các trọng số được chọn thông qua thử nghiệm nhiều lần để cho ra kết quả tốt

- Chi tiết:

*** CurrentScore: currentScore = currentGameState.getScore()**

+ Mỗi hành động (kể cả đứng yên): **-1 điểm**

+ Ăn chấm thức ăn: **+10 điểm**

+ Ăn Capsule: **+0 điểm** (lúc này các con ma chuyển sang trạng thái sợ hãi trong thời gian ngắn và Pacman có thể ăn ma để kiếm điểm)

+ Ăn ma: **+200 điểm**

+ Thắng (ăn hết các chấm thức ăn): **+500 điểm**

+ Thua (bị ma ăn): **-500 điểm**

**** Thức ăn (distToFood):***

- Ý tưởng chính của trong thiết kế của em nằm ở việc tính khoảng cách đến thức ăn dựa trên MST (Minimum Spanning Tree). Đây là một chiến lược để tính toán khoảng cách tối ưu mà Pacman cần di chuyển để ăn hết thức ăn dựa trên cách tiếp cận MST, giúp đảm bảo rằng Pacman di chuyển hiệu quả và tiết kiệm thời gian nhất có thể. Dưới đây là phân tích chi tiết về cách thức hoạt động của hàm này (các trường hợp đặc biệt như không còn thức ăn hay chỉ còn 1 thức ăn sẽ có trọng số khác):

- Bước đầu sử dụng khoảng cách Manhattan để ước lượng từ vị trí hiện tại của Pacman đến từng vị trí thức ăn và lựa chọn khoảng cách ngắn nhất.
- Thuật toán bắt đầu bằng cách chọn một đỉnh làm gốc, sau đó mở rộng cây khung bằng cách thêm đỉnh kề có trọng số cạnh nhỏ nhất cho tới khi cây bao gồm tất cả các đỉnh:
  1. Lấy thức ăn gần nhất với Pacman (sử dụng Manhattan ở trên) để làm gốc với trọng số (khoảng cách) là 0, và các đỉnh khác có khoảng cách khởi tạo là 999999, biểu thị rằng chúng chưa được kết nối với cây khung.
  2. Tiến hành xây dựng MST: Vòng lặp tiếp tục miễn là còn đỉnh trong heap, nơi heap được sử dụng để lưu trữ các đỉnh chưa được thêm vào cây khung với trọng số là khoảng cách ngắn nhất từ cây khung hiện tại đến mỗi đỉnh. *heapq.heappop(h)* lấy ra đỉnh u từ heap, đây là đỉnh có trọng số (khoảng cách) nhỏ nhất đến cây khung hiện tại, và khoảng cách này được cộng dồn vào *MSTDistance*. Sau đó, vòng lặp qua mỗi đỉnh v còn lại trong heap để xem liệu khoảng cách từ u đến v

```
while len(h) > 0:
    u = heapq.heappop(h)
    MSTDistance += u[0]
    for v in h:
        if util.manhattanDistance(u[1], v[1]) < v[0]:
            v[0] = util.manhattanDistance(u[1], v[1])
    heapq.heapify(h)
```

có thể cập nhật khoảng cách hiện tại của v trong heap hay không. Nếu khoảng cách mới nhỏ hơn, khoảng cách này sẽ được cập nhật. Để duy trì tính chất của min-heap, *heapq.heapify(h)* được gọi sau mỗi lần cập nhật để đảm bảo rằng đỉnh với trọng số nhỏ nhất luôn nằm ở đầu heap.

➔ Kết quả cuối cùng, MSTDistance, là tổng khoảng cách mà Pacman cần để thu thập tất cả các điểm thức ăn theo lộ trình tối ưu nhất.

**** Ma và Capsules:***

- Các hàm phụ kiểm tra trạng thái của ma:

  + *anyGhostIsScared*: Trả về True nếu có bất kỳ ma nào có thời gian sợ hãi lớn hơn không.

  + *noneOfGhostIsScared*: Trả về True nếu không có ma nào đang sợ hãi (tất cả các giá trị scaredTimer đều là không).

  + *allGhostAreScared*: Trả về True nếu mọi ma đều đang sợ hãi (tất cả các giá trị scaredTimer lớn hơn không).

- Hàm chính:

  + *minDistanceToNonScareGhost*: Hàm này tính khoảng cách Manhattan nhỏ nhất từ Pacman đến con ma gần nhất không đang trong trạng thái sợ hãi.

  Trọng số: **-50/distToNonScaredGhost**

  Các bước thực hiện trong hàm như sau:

  1. Kiểm tra tất cả các ma có đang sợ hãi không: Nếu hàm allGhostAreScared trả về True, tức là tất cả các ma đều đang sợ hãi và không phải là mối đe dọa. Hàm sau đó trả về một giá trị cao (999999).
  2. Nếu không phải tất cả các ma đều sợ hãi, hàm sẽ lặp qua từng ma và tính khoảng cách Manhattan từ vị trí của Pacman đến những ma không sợ hãi để tìm ra ma không sợ hãi gần nhất, và cộng thêm một khoảng nhỏ (0.1) để đảm bảo giá trị không bằng không, thuận tiện cho các phép tính sau có thể chia cho giá trị này.

  Trọng số là **-50/distToNonScaredGhost**, vì khoảng cách càng nhỏ, tức ma càng gần, mối đe dọa càng lớn, và điểm trừ càng nhiều trong đánh giá tổng thể. Điều này thúc đẩy Pacman tránh xa các ma không sợ hãi. Ngược lại, trả về một giá trị rất cao giúp đảm bảo rằng trọng số -50/minDistanceToNonScareGhost là một số rất nhỏ, phản ánh đúng tình huống không cần quan tâm đến ma không sợ hãi.

+ *minDistanceToScareGhost*: Hàm này tương tự như hàm trên nhưng nhắm vào các ma sợ hãi, tính khoảng cách nhỏ nhất từ Pacman đến con ma sợ hãi gần nhất:

Trọng số: **+200/distToScareGhost**

Các bước thực hiện trong hàm như sau:

1. Kiểm tra nếu noneOfGhostIsScared trả về True, tức là không có ma sợ hãi để Pacman săn điểm, và nó trả về giá trị cao (999999), điểm thưởng cho tiếp cận nhằm ăn ma sợ hãi lúc này được coi như gần bằng 0
2. Nếu có ma sợ hãi, tính toán khoảng cách đến các ma sợ hãi để tìm ma sợ hãi gần nhất. và cộng thêm một khoản nhỏ (0.1) để đảm bảo kết quả không bằng không.

Trọng số là **+200/distToScareGhost**, khuyến khích Pacman tiếp cận và ăn các ma sợ hãi để kiếm điểm thưởng. Trọng số sẽ càng lớn khi khoảng cách giảm, điều này có nghĩa là càng gần ma sợ hãi, điểm thưởng cộng thêm vào điểm số tổng thể càng lớn.

+ *minDistanceToCapsules*: Hàm khuyến khích Pacman di chuyển gần các capsules, đặc biệt khi các ma không sợ hãi.

Trọng số: **+100/distanceToCapsules**

Các bước thực hiện trong hàm như sau:

1. Nếu không còn capsule nào, hàm trả về giá trị cao (999999), trọng số lúc này gần như gần bằng 0, không ảnh hưởng đến tổng điểm.
2. Nếu còn, dùng Manhattan để ước lượng khoảng cách từ Pacman đến mỗi capsule và sau đó xác định capsule gần nhất, cộng thêm một khoản nhỏ (0.1) để đảm bảo khoảng cách không bằng không.

Trọng số **+100/distanceToCapsules** *(chỉ được cộng vào tổng điểm khi không có ma nào sợ hãi)*, khuyến khích Pacman di chuyển gần các capsules. Khoảng cách giữa Pacman và capsule càng thấp, Pacman sẽ càng nhận được nhiều điểm thưởng, khuyến khích việc thu thập capsules để tăng khả năng sống sót và điểm số.

II. Thống kê kết quả thực nghiệm

*** Thống kê:**

- Kết quả được thử nghiệm trên 5 layout capsuleClassic, contestClassic, mediumClassic, minimaxClassic và trappedClassic với số depth là 3, được chạy với cả 3 thuật toán (Minimax,

AlphaBeta, Expectimax) với lần lượt hàm lượng giá **scoreEvaluationFunction**, **betterEvaluationFunction** có sẵn (của thầy) và **betterEvaluationFunction2** mới. Thử nghiệm trên ma Random Ghost.

- Mỗi layout đều được chạy với các thuật toán và hàm lượng giá khác nhau 5 lần (5 random seed là 22521293+0 -> 22521293+4). Qua đó trích xuất win rate và điểm số trung bình để so sánh hiệu suất của các thuật toán và các hàm lượng giá.

Bảng thống kê kết quả của các giải thuật với các hàm lượng giá

Random Ghost						
scoreEvaluation Function	**Minimax**		**AlphaBeta**		**Expectimax**	
Layout	**Average Score**	**Game win**	**Average Score**	**Game win**	**Average Score**	**Game win**
capsuleClassic	-416.8	0/5 (0.00)	-416.8	0/5 (0.00)	-419.8	0/5 (0.00)
contestClassic	-178.4	0/5 (0.00)	-178.4	0/5 (0.00)	1008.8	0/5 (0.00)
mediumClassic	36.6	1/5 (0.20)	36.6	1/5 (0.20)	377.6	2/5 (0.40)
minimaxClassic	-495.8	0/5 (0.00)	-495.8	0/5 (0.00)	-93.2	2/5 (0.40)
trappedClassic	-501.0	0/5 (0.00)	-501.0	0/5 (0.00)	-88.4	2/5 (0.40)

betterEvaluation Function (sẵn)	**Minimax**		**AlphaBeta**		**Expectimax**	
Layout	**Average Score**	**Game win**	**Average Score**	**Game win**	**Average Score**	**Game win**
capsuleClassic	-464.2	0/5 (0.00)	-464.2	0/5 (0.00)	-474.4	0/5 (0.00)
contestClassic	1122.6	3/5 (0.60)	1122.6	3/5 (0.60)	1232.4	4/5 (0.80)
mediumClassic	359.6	3/5 (0.60)	359.6	3/5 (0.60)	218.6	5/5 (1.00)
minimaxClassic	-497.0	0/5 (0.00)	-497.0	0/5 (0.00)	-94.0	2/5 (0.40)
trappedClassic	-88.4	2/5 (0.40)	-88.4	2/5 (0.40)	-88.4	2/5 (0.40)

betterEvaluationFunction2	**Minimax**		**AlphaBeta**		**Expectimax**	
Layout	**Average Score**	**Game win**	**Average Score**	**Game win**	**Average Score**	**Game win**
capsuleClassic	1387.8	3/5 (0.60)	1387.8	3/5 (0.60)	1709.0	4/5 (0.80)
contestClassic	3023.4	5/5 (1.00)	3023.4	5/5 (1.00)	2476.4	4/5 (0.80)
mediumClassic	1977.8	5/5 (1.00)	1977.8	5/5 (1.00)	1920.4	5/5 (1.00)
minimaxClassic	-496.8	0/5 (0.00)	-496.8	0/5 (0.00)	112.2	3/5 (0.60)
trappedClassic	-88.4	2/5 (0.40)	-88.4	2/5 (0.40)	-88.4	2/5 (0.40)

Agent	**Số ván thắng**	**Tổng điểm trung bình**
Minimax	24/75	4681
AlphaBeta	24/75	4681
Expectimax	37/75	7708.8

EvaluationFunction	**Số ván thắng**	**Tổng điểm trung bình**
scoreEvaluationFunction	8/75	-2325.8
betterEvaluationFunction	29/75	1671.4
betterEvaluationFunction2	48/75	17737.2

*** Nhận xét:**

- **Các thuật toán *Minimax, Alpha-Beta và Expectimax***

- Hai thuật toán MinimaxAgent và AlphaBetaAgent gần như tương tự nhau, nhưng ở AlphaBeta sẽ cho thời gian xử lí thấp hơn so với Minimax nếu các nhánh có thứ tự sắp xết tốt hơn và độ sâu lớn hơn nhờ vào bước cải tiến Pruning so với minimax.
- Thuật toán Expectimax thường đạt điểm số trung bình cao hơn so với Minimax và AlphaBeta ở tất cả các hàm đánh giá. Điều này do với Expectimax, hành động của đối thủ là ngẫu nhiên không phải đối thủ chơi hoàn hảo - tìm kiếm đường đi tốt nhất giả định như Minimax và AlphaBeta.

- **Các hàm đánh giá *scoreEvaluationFunction, betterEvaluationFunction, betterEvaluationFunction2***

- Ta có thể thấy, hàm lượng giá mới tuy chưa đạt đến ngưỡng tối ưu nhất, nhưng đã tốt hơn rất nhiều so với hàm lượng giá cũ, có nhiều layout mà hàm lượng giác cũ vẫn chưa giải quyết được nhưng với hàm lượng giác mới đã có thể giải khá suôn sẻ.
- Khi sử dụng hàm betterEvaluationFunction2, cả ba thuật toán đều có số lần thắng cao hơn đáng kể so với hai hàm đánh giá còn lại. Hàm betterEvaluationFunction2 là một hàm đánh giá mạnh mẽ hơn, tốt hơn trong việc xác định các trạng thái có lợi và có chiến lược chơi tối ưu hơn.
- Layout minimaxClassic là khó khăn nhất đối với tất cả các thuật toán và hàm đánh giá, điều này có thể phản ánh mức độ phức tạp của layout này.

KẾT LUẬN

- **betterEvaluationFunction2** tuy đã hoạt động tốt hơn 2 hàm cũ, nhưng vẫn cần cải thiện.
- Đánh giá hiệu năng: Minimax < AlphaBeta < Expectimax.
- Link video record lại 1 ván chơi với câu lệnh “python pacman.py -l capsuleClassic -p AlphaBetaAgent -a depth=3, evalFn=betterEvaluationFunction2 -g RandomGhost -s 22521295 --frameTime 0”: **https://drive.google.com/file/d/1Ii4DD1ednsjn1pjPqlSsEQCV1a64ed8T/view?usp=sharing**