
Problem 1. *Thông tin chung về trò chơi Pacman và các thuật toán Minimax, AlphaBeta và Expectimax*

Answer. Pac-Man là một trò chơi điện tử kinh điển ra mắt vào năm 1980, trong trò chơi ta điều khiển Pacman trong một mê cung và ăn các chấm thức ăn Nếu người chơi ăn hết các chấm thì Pacman được đưa qua màn chơi mới.

Lối chơi

- Mê cung: Pac-Man di chuyển trong một mê cung được chia thành các ô vuông, mỗi ô có thể chứa chấm thức ăn hoặc tường.
- Chấm thức ăn: Mục tiêu chính của Pac-Man là ăn hết tất cả các chấm trong mê cung. Khi ăn đủ số lượng chấm, Pac-Man sẽ sang màn chơi tiếp theo.
- Ma: Tối đa Bốn con ma di chuyển tự do trong mê cung và cố gắng bắt Pac-Man. Nếu Pac-Man bị bắt, người chơi sẽ mất mạng và thua cuộc.
- Capsule: Một số ô vuông trong mê cung chứa viên năng lượng, khi ăn viên này, Pac-Man sẽ có khả năng "ăn" ma trong một khoảng thời gian ngắn.

⇒ Dây là một dạng bài toán Adversarial.

Thuật toán

Trong bài tập này, chúng ta sẽ thiết kế các agent cho Pacman. Các thuật toán được sử dụng là Minimax, AlphaBeta và Expectimax. Đồng thời, việc thiết kế một hàm đánh giá (Evaluation Function) để ước tính giá trị của các trạng thái trong trò chơi cũng đóng vai trò quan trọng

Mục tiêu

Mục tiêu của việc thiết kế agent cho Pac-Man là:

- Ăn hết tất cả các chấm thức ăn trong mê cung.
- Né tránh hoặc tiêu diệt ma để sống sót.
- Đạt được số điểm cao nhất khi kết thúc trò chơi.

Problem 2. *Thiết kế các đặc trưng và hàm Evaluation Function*

Answer. Hàm đánh giá đóng vai trò quan trọng trong việc đánh giá giá trị của các trạng thái trong trò chơi Pac-Man, từ đó giúp agent đưa ra quyết định di chuyển hiệu quả nhất. Ta sẽ thiết kế một hàm đánh giá mới (betterEvaluationFunction) được cải tiến dựa trên những đặc trưng và kinh nghiệm thu thập được khi chơi game.

Điểm số và các hành động
Mỗi hành động của Pac-Man ảnh hưởng đến điểm số và kết quả cuối cùng của trò chơi. Do đó, hàm đánh giá cần được thiết kế dựa trên các kết quả mà hành động đó mang lại. Các điểm số tương ứng với các hành động trong trò chơi như sau:

- Thời gian: -1 điểm mỗi giây.
- Chấm thức ăn: 10 điểm mỗi chấm.
- Chiến thắng: 500 điểm.
- Ăn ma: 200 điểm.
- Thua cuộc: -500 điểm.
- Capsule: 0 điểm

Hàm đánh giá cũ (scoreEvaluationFunction) chỉ sử dụng điểm số của trạng thái để ước lượng giá trị trạng thái. Tuy nhiên, điểm số không phản ánh đầy đủ thông tin về tình trạng hiện tại của trò chơi. Do đó, cần bổ sung thêm các đặc trưng khác để đánh giá chính xác hơn.

Các đặc trưng được sử dụng trong hàm đánh giá mới (betterEvaluationFunction) bao gồm:

- '**ghost_dir**': Khoảng cách đến con ma gần nhất.
- '**closest_food**': Khoảng cách đến chấm thức ăn gần nhất.
- '**closest_capsule**': Khoảng cách đến capsule gần nhất.
- '**currentGameState.getScore()**': Điểm số hiện tại của trò chơi.
- '**no_food**': Số lượng chấm thức ăn còn lại.
- '**no_capsule**': Số lượng capsule còn lại.

Kinh nghiệm và chiến lược Qua quá trình chơi và thử nghiệm, một số kinh nghiệm sau đây được rút ra:

- Điểm số hiện tại (currentGameState.getScore()) là yếu tố gần nhất với kết quả cuối cùng của trò chơi. Do đó, cần ưu tiên tăng điểm số hiện tại để tối ưu hóa kết quả.
- Số lượng thức ăn và capsule còn lại (no_food và no_capsule) càng nhỏ thì càng gần đến chiến thắng. Do đó, cần khuyến khích giảm giá trị của hai đặc trưng này.
- Khoảng cách đến chấm thức ăn và capsule (closest_food và closest_capsule) càng nhỏ thì Pac-Man càng có lợi thế. Do đó, cần ưu tiên di chuyển đến những mục tiêu này.

```
def betterEvaluationFunction(currentGameState):
    """
    Your extreme ghost-hunting, pellet-nabbing, food-gobbling,
unstoppable
    evaluation function (question 5).
    DESCRIPTION: Inverse sums of nearest food distances and
capsule distances, adding game score,
    subtracting ghost distance and remaining food.
    """

    ghostStates = currentGameState.getGhostStates()
    pacmanPos = currentGameState.getPacmanPosition()
    foodList = currentGameState.getFood().asList()
    capsuleList = currentGameState.getCapsules()
    numFood = len(foodList)
    numCapsules = len(capsuleList)

    stateScore = 0

    # Feature 1: distances from ghosts if they exist
    if currentGameState.getNumAgents() > 1:
        ghostDistances = [manhattanDistance(pacmanPos, ghost.
getPosition()) for ghost in ghostStates]
        minGhostDist = min(ghostDistances)
        if minGhostDist <= 0.05:
            return -10000
        stateScore -= 1.0 / minGhostDist

    # Feature 2: food positions
    currentFood = pacmanPos
    while foodList:
        closestFood = min(foodList, key=lambda x:
manhattanDistance(x, currentFood))
        stateScore += 1.0 / manhattanDistance(currentFood,
closestFood)
        currentFood = closestFood
        foodList.remove(closestFood)

    # Feature 3: capsule positions
    currentCapsule = pacmanPos
    while capsuleList:
        closestCapsule = min(capsuleList, key=lambda x:
manhattanDistance(x, currentCapsule))
        stateScore += 1.0 / manhattanDistance(currentCapsule,
closestCapsule)
        currentCapsule = closestCapsule
        capsuleList.remove(closestCapsule)

    # Feature 4: Score of the game
    stateScore += 8 * currentGameState.getScore()

    # Feature 5: remaining food and capsules
    stateScore -= 6 * (numFood + numCapsules)

    return stateScore
```

Problem 3. *Thực nghiệm và Thống kê*

Answer. Kết quả thực nghiệm so sánh hiệu suất của các thuật toán Minimax, AlphaBeta và Expectimax kết hợp với hai hàm đánh giá (Evaluation Function) cũ (scoreEvaluationFunction) và mới (betterEvaluationFunction) trên trò chơi Pac-Man.

Phương pháp

- Thử nghiệm được thực hiện trên 5 layout (capsuleClassic, contestClassic, mediumClassic, minimaxClassic và trappedClassic) và ta sẽ nhìn trước với depth là 2.
- Hai loại ma được sử dụng là Random Ghost và Directional Ghost.
- Mỗi layout với mỗi hàm đánh giá được chạy 5 lần (random seed 21521413 → 21521417).
- Các thông tin bao gồm tỷ lệ chiến thắng (Win count), thời gian chạy trung bình (Time) và điểm số trung bình (Average Score).

Random Ghost - betterEvaluationFunction

Layout	Minimax			AlphaBeta			Expectimax		
	Score	Win	Time	Score	Win	Time	Score	Win	Time
capsuleClassic	-472.6	0/5 (0.00)	3.21	-472.6	0/5 (0.00)	3.08	26.8	1/5 (0.20)	6.30
contestClassic	1445	3/5 (0.60)	21.01	1445	3/5 (0.60)	18.79	1402.8	4/5 (0.80)	20.54
mediumClassic	1196.8	4/5 (0.80)	26.03	1196.8	4/5 (0.80)	23.57	1553.8	5/5 (1.00)	22.53
minimaxClassic	112	3/5 (0.60)	0.14	112	3/5 (0.60)	0.14	313.4	4/5 (0.80)	0.16
trappedClassic	118.4	3/5 (0.60)	0.05	118.4	3/5 (0.60)	0.05	118.4	3/5 (0.60)	0.06

Random Ghost - scoreEvaluationFunction

Layout	Minimax			AlphaBeta			Expectimax		
	Score	Win	Time	Score	Win	Time	Score	Win	Time
capsuleClassic	-348.2	0/5 (0.00)	4.11	-348.2	0/5 (0.00)	3.94	-354.2	0/5 (0.00)	4.49
contestClassic	8.8	0/5 (0.00)	9.82	8.8	0/5 (0.00)	9.13	397.2	1/5 (0.20)	18.23
mediumClassic	-66.6	1/5 (0.20)	31.84	-66.6	1/5 (0.20)	30.39	-388.2	2/5 (0.40)	58.60
minimaxClassic	-92.8	2/5 (0.40)	0.21	-92.8	2/5 (0.40)	0.19	108.6	3/5 (0.60)	0.22
trappedClassic	118.4	3/5 (0.60)	0.05	118.4	3/5 (0.60)	0.05	118.4	3/5 (0.60)	0.05

Directional Ghost - betterEvaluationFunction									
Layout	**Minimax**			**AlphaBeta**			**Expectimax**		
	Score	Win	Time	Score	Win	Time	Score	Win	Time
capsuleClassic	-307.8	0/5 (0.00)	2.55	-307.8	0/5 (0.00)	2.41	-286.2	0/5 (0.00)	2.40
contestClassic	42.4	0/5 (0.00)	13.76	42.4	0/5 (0.00)	11.85	1.2	0/5 (0.00)	13.07
mediumClassic	441.6	1/5 (0.20)	14.80	441.6	1/5 (0.20)	13.04	689.8	2/5 (0.40)	14.16
minimaxClassic	-492	0/5 (0.00)	0.11	-492	0/5 (0.00)	0.08	-291.2	1/5 (0.20)	0.09
trappedClassic	-295.2	1/5 (0.20)	0.05	-295.2	1/5 (0.20)	0.04	-295.2	1/5 (0.20)	0.04

Directional Ghost - betterEvaluationFunction									
Layout	**Minimax**			**AlphaBeta**			**Expectimax**		
	Score	Win	Time	Score	Win	Time	Score	Win	Time
capsuleClassic	-420.4	0/5 (0.00)	0.88	-420.4	0/5 (0.00)	0.82	-420.4	0/5 (0.00)	0.89
contestClassic	-258	0/5 (0.00)	2.89	-258	0/5 (0.00)	2.48	-260.4	0/5 (0.00)	2.91
mediumClassic	465.4	0/5 (0.00)	5.03	465.4	0/5 (0.00)	4.76	-388.2	2/5 (0.40)	58.60
minimaxClassic	-492	0/5 (0.00)	0.07	-492	0/5 (0.00)	0.07	-291.2	1/5 (0.20)	0.09
trappedClassic	-295.2	1/5 (0.20)	0.03	-295.2	1/5 (0.20)	0.03	-295.2	1/5 (0.20)	0.04

So sánh hai Evaluation Function

- Random Ghost: Hàm đánh giá mới (betterEvaluationFunction) vượt trội hơn hẳn hàm đánh giá cũ (scoreEvaluationFunction) với tỷ lệ chiến thắng gấp đôi (44/75 so với 21/75).
- Directional Ghost: Sự chênh lệch giữa hai hàm đánh giá không đáng kể (8/75 so với 6/75).

So sánh các thuật toán tìm kiếm

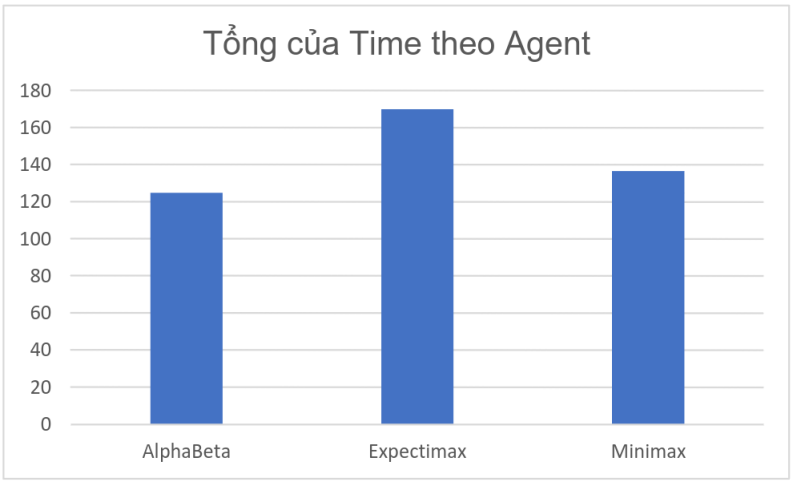
- Expectimax: Nhìn chung, Expectimax có hiệu suất tốt nhất với tỷ lệ chiến thắng cao hơn và thời gian chạy tương đương hoặc thấp hơn so với Minimax và AlphaBeta.
- Minimax và AlphaBeta: Hai thuật toán này có hiệu suất tương đương nhau với Minimax có thời gian chạy thấp hơn một chút.

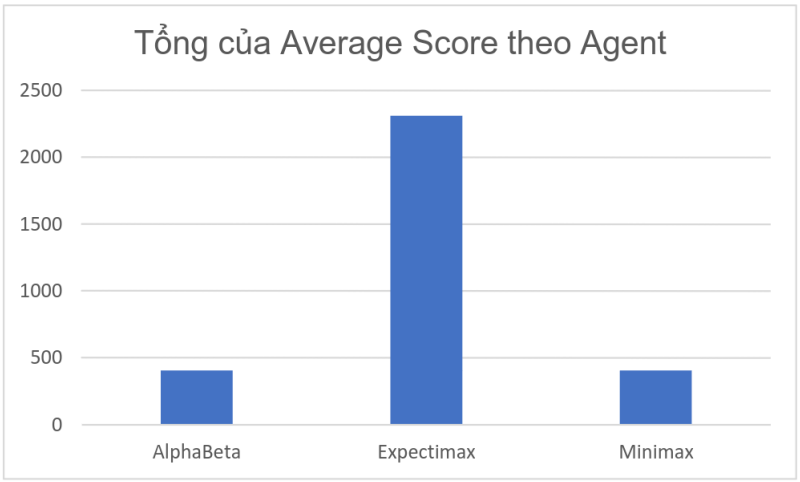
Phân tích chi tiết

- CapsuleClassic: Hàm đánh giá mới không phát huy hiệu quả do chiến lược ưu tiên ăn thức ăn thay vì ăn ma không phù hợp với layout này.
- ContestClassic, MediumClassic và MinimaxClassic: Expectimax có hiệu suất tốt nhất với cả hai hàm đánh giá.
- TrappedClassic: Kết quả tương tự nhau cho cả ba thuật toán với cả hai hàm đánh giá do tính chất ngẫu nhiên của Random Ghost.
- Directional Ghost: Hàm đánh giá mới có thể chưa đủ hiệu quả để khai thác lợi thế của thuật toán Expectimax khi đối mặt với ma thông minh hơn.

Hướng cải thiện hàm đánh giá

- Khoảng cách: Thay vì sử dụng khoảng cách Manhattan (khoảng cách theo hàng ngang và dọc), cần sử dụng khoảng cách thực tế giữa Pac-Man và các đối tượng (ma, thức ăn, capsule) để phản ánh chính xác hơn độ dài đường đi. Có thể sử dụng thuật toán tìm kiếm đường đi như BFS (Breadth-First Search) hoặc Floyd-Warshall để tính toán khoảng cách thực tế.
- Trạng thái ma: Hàm đánh giá nên cân nhắc trạng thái của ma (ví dụ: đang di chuyển hay đuổi theo Pac-Man) để đưa ra quyết định phù hợp. Ví dụ, nếu ma đang đuổi theo Pac-Man, Pac-Man nên ưu tiên né tránh ma thay vì ăn thức ăn.
- Số lượng thức ăn và capsule còn lại: Cần điều chỉnh cho các đặc trưng này để phản ánh tầm quan trọng của chúng trong từng giai đoạn của trò chơi. Ví dụ, khi còn ít thức ăn và capsule, Pac-Man nên ưu tiên tìm kiếm chúng hơn so với giai đoạn đầu trò chơi.





Problem 4. *Resources*

Videos record các màn chơi được lưu tại đây.