ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

TRÍ TUỆ NHÂN TẠO (CS106.P21)

BÁO CÁO

EVALUATION FUNCTIONS FOR MINIMAX/ALPHABETA/EXPECTIMAX

GIẢNG VIÊN HƯỚNG DẪN: TS. LƯƠNG NGỌC HOÀNG

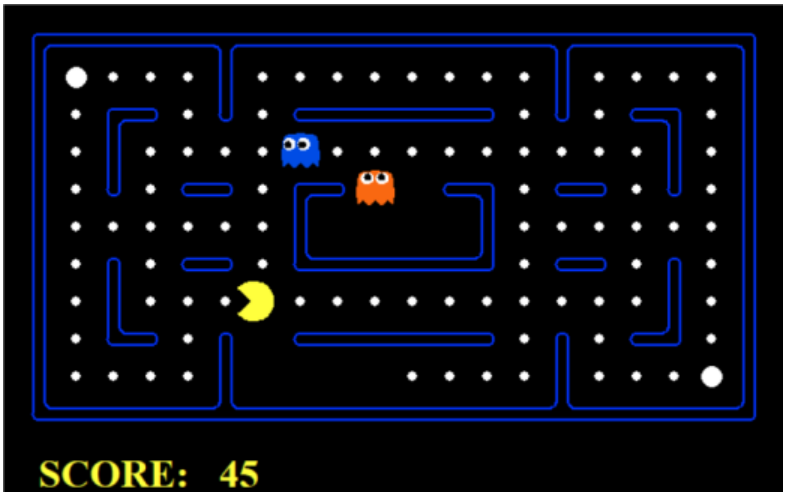
STT	Họ tên SV	MSSV
1	Trần Minh Tiến	23521587

Nội dung

1 Tổng quan về trò chơi Pacman

Pacman là một trò chơi cổ điển, nơi người chơi điều khiển một nhân vật có hình dạng tròn màu vàng – gọi là Pacman – với mục tiêu đơn giản: ăn hết toàn bộ food trong bản đồ mà không bị ghost bắt. Trong mỗi bản đồ, ta sẽ thấy ba thành phần chính:

- **Pacman:** Nhân vật người chơi điều khiển, được biểu diễn bằng hình tròn màu vàng bị khuyết một phần. Nhiệm vụ của Pacman là di chuyển qua bản đồ để ăn hết các viên Food nhằm hoàn thành màn chơi và tối đa hóa điểm số.
- **Ghost:** Các đối thủ của Pacman, thường có màu như xanh dương hoặc cam. Nếu Pacman chạm phải ghost khi chúng đang ở trạng thái bình thường, trò chơi sẽ kết thúc. Tuy nhiên, nếu Pacman đã ăn capsule trước đó thì ghost sẽ chuyển sang trạng thái bị dọa (có màu trắng), và lúc này Pacman có thể quay lại ăn ghost để ghi thêm điểm.
- **Food và Capsule:** Food là những chấm trắng nhỏ rải rác trên bản đồ, cần ăn hết chúng để hoàn thành màn chơi. Capsule là các viên đặc biệt mà khi ăn vào, toàn bộ ghost trên bản đồ sẽ rơi vào trạng thái "scared" (sợ hãi) trong một khoảng thời gian ngắn.



Hình 1: Minh họa một màn chơi Pacman.

2 Mô tả hàm lượng giá

Trong môi trường Pacman, hàm đánh giá quyết định hành vi di chuyển của agent tại mỗi trạng thái. Thay vì chỉ dựa vào `currentGameState.getScore()`, `mybetterEvaluationFunction` tổng hợp đồng thời nhiều đặc trưng để phản ánh chính xác cả lợi thế và nguy cơ trong trò chơi.
Hàm đánh giá được xây dựng dựa trên các đặc trưng sau:

- **Điểm số nền tảng (Score(s)):** Đây là giá trị điểm mà hệ thống trò chơi cung cấp, đại diện cho tiến trình tổng thể của game. Điểm số này phản ánh mức độ thành công hiện tại của agent trong việc hoàn thành các mục tiêu của trò chơi.
- **Khoảng cách đến ghost và trạng thái ghost (ghostState.scaredTimer):** Khoảng cách giữa Pacman và các ghost, cũng như trạng thái "sợ" của ghost (scaredTimer), là yếu tố quan trọng để quyết định hành vi tấn công hoặc phòng thủ.
- **Khoảng cách đến capsule (getCapsules()):** Capsule có thể cung cấp cơ hội để Pacman tấn công ghost, do đó việc tìm capsule gần nhất và ăn nó là rất quan trọng.
- **Số lượng food còn lại và khoảng cách đến food gần nhất (getFood().asList() và manhattanDistance):** Các yếu tố này thúc đẩy Pacman hoàn thành mục tiêu bằng cách ăn hết food, đồng thời giúp tối ưu hóa hành trình di chuyển của Pacman.

2.1 Score(s):

Điểm số tại trạng thái hiện tại được lấy từ hàm `getScore()` của `currentGameState`. Đây là thành phần gốc và phản ánh tiến trình trò chơi tổng thể. Trạng thái có điểm số cao hơn thể hiện rằng Pacman đang tiến gần hơn đến mục tiêu của trò chơi.

$$\text{Score}(s) = \texttt{currentGameState.getScore()}$$

2.2 RewardGain:

Phần thưởng được tính dựa trên các cơ hội có lợi cho Pacman, bao gồm các yếu tố liên quan đến ghost, food, và capsule.

- **Ghost bị dọa và gần:** Nếu ghost đang bị dọa (scaredTimer > 0) và ở gần Pacman (khoảng cách ≤ 2), Pacman sẽ được cộng thêm phần thưởng 100 điểm để khuyến khích hành vi tấn công. Nếu ghost bị dọa nhưng ở xa hơn, phần thưởng giảm dần theo khoảng cách:

$$\text{reward} = \frac{10}{\text{distance}}$$

- **Food gần:** Phần thưởng được cộng thêm khi Pacman tiếp cận food gần nhất. Khoảng cách đến food gần nhất được tính bằng manhattanDistance, và phần thưởng được tính theo công thức:

$$\text{reward} = \frac{10}{\text{minFoodDist} + 1}$$

  trong đó `minFoodDist` là khoảng cách ngắn nhất từ Pacman đến food. Nếu không còn food nào, phần thưởng sẽ được tăng lên 100 điểm để khuyến khích Pacman hoàn thành game.

- **Capsule gần:** Phần thưởng được cộng thêm nếu Pacman di chuyển đến capsule gần nhất. Khoảng cách đến capsule gần nhất cũng được tính bằng manhattanDistance:

$$\text{reward} = \frac{5}{\text{minCapsuleDist} + 1}$$

2.3 PenaltyRisk:

Phạt được áp dụng khi Pacman ở gần ghost mà không có trạng thái bị dọa. Nếu ghost không bị dọa và ở gần Pacman (khoảng cách ≤ 2), Pacman sẽ bị phạt 100 điểm để thúc đẩy hành vi phòng thủ, tránh nguy cơ bị ghost ăn.

$$\text{penaltyRisk} = \begin{cases} 100 & \text{nếu distance} \leq 2 \text{ và ghost không bị dọa} \\ 0 & \text{ngược lại} \end{cases}$$

2.4 PenaltyIncompletion:

Phạt được tính dựa trên số lượng food và capsule còn lại trong game. Mục tiêu của phạt này là khuyến khích Pacman hoàn thành nhiệm vụ và tránh việc trì hoãn quá lâu.

- **Phạt food còn lại:** Mỗi food còn lại trong game sẽ bị phạt 2 điểm:

$$\text{penaltyIncompletion\_food} = 2 \times \text{len(foodList)}$$

- **Phạt capsule còn lại:** Mỗi capsule chưa được ăn sẽ bị phạt 10 điểm:

$$\text{penaltyIncompletion\_capsule} = 10 \times \text{len(capsules)}$$

2.5 Công thức tính giá trị đánh giá:

Giá trị đánh giá cuối cùng được tính dựa trên các đặc trưng đã nêu. Công thức tổng quát của hàm đánh giá là:

$$\text{Evaluation}(s) = \text{Score}(s) + \text{RewardGain} - \text{PenaltyRisk} - \text{PenaltyIncompletion}$$

Trong đó:

- **Score(s):** Điểm số tại trạng thái s.
- **RewardGain:** Tổng phần thưởng từ các cơ hội có lợi như ghost bị dọa, food gần, và capsule gần.
- **PenaltyRisk:** Phạt khi Pacman ở gần ghost không bị dọa, tạo nguy cơ bị ăn.
- **PenaltyIncompletion:** Phạt nhẹ nếu còn nhiều food hoặc capsule chưa ăn.

Các trọng số được lựa chọn thông qua thực nghiệm để đạt được sự cân bằng giữa các yếu tố tấn công, phòng thủ, và tốc độ hoàn thành game. Hàm đánh giá từ đó giúp Pacman hành động linh hoạt, tránh hành vi quá tham lam hoặc phòng thủ cực đoan.

3 Kết quả thực nghiệm

Hai hàm lượng giá được kiểm tra trên 5 bản đồ khác nhau với ba thuật toán: `MinimaxAgent`, `AlphaBetaAgent` và `ExpectimaxAgent`. Mỗi thuật toán được chạy với độ sâu tìm kiếm là 3. Để đảm bảo kết quả không bị ảnh hưởng bởi tính ngẫu nhiên trong game, mỗi bản đồ được chạy 5 lần với các random seed khác nhau. Sau mỗi lần chạy, hai chỉ số chính được ghi nhận:

- **Điểm số trung bình**: trung bình tổng điểm Pacman đạt được trên mỗi lượt chạy.
- **Tỉ lệ thắng**: tỉ lệ Pacman ăn hoàn thành trò chơi.

3.1 scoreEvaluationFunction

Layout	MinimaxAgent		AlphaBetaAgent		ExpectimaxAgent	
	Scores	Win rate	Scores	Win rate	Scores	Win rate
minimaxClassic	-293.8	0.2	-293.8	0.2	-293.2	0.2
capsuleClassic	-417.4	0	-417.4	0	-385.4	0
smallClassic	765.6	0.6	765.6	0.6	809	0.8
contestClassic	290	0.2	290	0.2	856.4	0.2
testClassic	520	1	520	1	526.4	1

Bảng 1: Kết quả thực nghiệm với scoreEvaluationFunction

3.2 mybetterEvaluationFunction

Layout	MinimaxAgent		AlphaBetaAgent		ExpectimaxAgent	
	Scores	Win rate	Scores	Win rate	Scores	Win rate
minimaxClassic	-294.4	0.2	-294.4	0.2	108.8	0.6
capsuleClassic	140.4	0.2	104.4	0.2	436.2	0.2
smallClassic	1369.4	1	1369.4	1	1434.4	1
contestClassic	2508.6	1	2508.6	1	2257	0.8
testClassic	544	1	544	1	554.6	1

Bảng 2: Kết quả thực nghiệm với mybetterEvaluationFunction

3.3 Nhận xét

Dựa trên hai bảng kết quả thực nghiệm với `scoreEvaluationFunction` và `mybetterEvaluationFunction`, ta có thể rút ra các so sánh về hiệu suất như sau:

- Với tất cả các layout, `mybetterEvaluationFunction` đều cho kết quả điểm số (*Scores*) và tỉ lệ thắng (*Win rate*) cao hơn so với `scoreEvaluationFunction`, chứng tỏ rằng hàm đánh giá mới giúp Pacman đưa ra quyết định hiệu quả hơn.
- Trên layout `contestClassic`, điểm số của `ExpectimaxAgent` tăng từ 856.4 lên 2257, và tỉ lệ thắng tăng từ 0.2 lên 0.8. Đây là sự cải thiện rõ rệt, đặc biệt ở một layout phức tạp như `contestClassic`.
- Agent `ExpectimaxAgent` cho thấy khả năng khai thác tốt nhất từ hàm đánh giá mới, với tỉ lệ thắng đạt 1.0 ở hai layout `smallClassic` và `testClassic`, đồng thời luôn có điểm số cao nhất trong các agent.
- Cả `MinimaxAgent` và `AlphaBetaAgent` cũng được cải thiện về hiệu suất, nhưng không vượt trội như `ExpectimaxAgent`, do hai agent này giả định ghost hành động tối ưu thay vì ngẫu nhiên.

Tổng kết: `mybetterEvaluationFunction` đã chứng minh tính hiệu quả khi giúp tất cả các agent đạt được kết quả tốt hơn đáng kể. Điều này cho thấy hàm đánh giá mới đã phản ánh chính xác hơn giá trị của một trạng thái trong trò chơi, đồng thời phù hợp hơn với chiến lược của Pacman trong môi trường nhiều tác nhân. Đặc biệt, sự kết hợp giữa `mybetterEvaluationFunction` và `ExpectimaxAgent` cho thấy hiệu năng vượt trội, rất phù hợp cho các layout như `contestClassic`.

3.4 Ván chơi có kết quả tốt nhất

Ván chơi có kết quả tốt nhất được ghi nhận trên layout `contestClassic`, sử dụng `MinimaxAgent` cùng với hàm lượng giá `mybetterEvaluationFunction`. Dưới đây là video ghi lại ván chơi này:

https://drive.google.com/file/d/14aTwJFA-lBR1uwec-HQcgS8LMlyPkghx/view?usp=drive_link

Các lý do khiến ván chơi này đạt điểm số vượt trội:

- **Chiến lược tối ưu với ghost:** Do sử dụng `MinimaxAgent`, Pacman có thể dự đoán và tránh được các hành vi nguy hiểm từ ghost, nhờ đó giảm thiểu rủi ro bị thua.

- **Hàm lượng giá hiệu quả:** `mybetterEvaluationFunction` đã kết hợp tốt các yếu tố như khoảng cách đến thức ăn gần nhất, số lượng capsule còn lại, vị trí và trạng thái của ghost, từ đó đánh giá chính xác giá trị của từng trạng thái trong trò chơi.
- **Tránh rủi ro tốt:** Với khả năng đánh giá tốt các trạng thái nguy hiểm, Pacman giữ được mạng sống trong toàn bộ ván chơi, từ đó tận dụng tối đa thời gian để ghi điểm.
- **Tận dụng ghost bị "scared":** Khi ghost ở trạng thái bị sợ, Pacman chủ động săn ghost để ghi điểm thưởng, thay vì chỉ né tránh.