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

KHOA: KHOA HỌC MÁY TÍNH

MÔN HỌC: CS106.O21.KHTN - TRÍ TUỆ NHÂN TẠO

Pacman MultiAgents

Sinh viên:
Nguyễn Tuấn Anh – 21520142

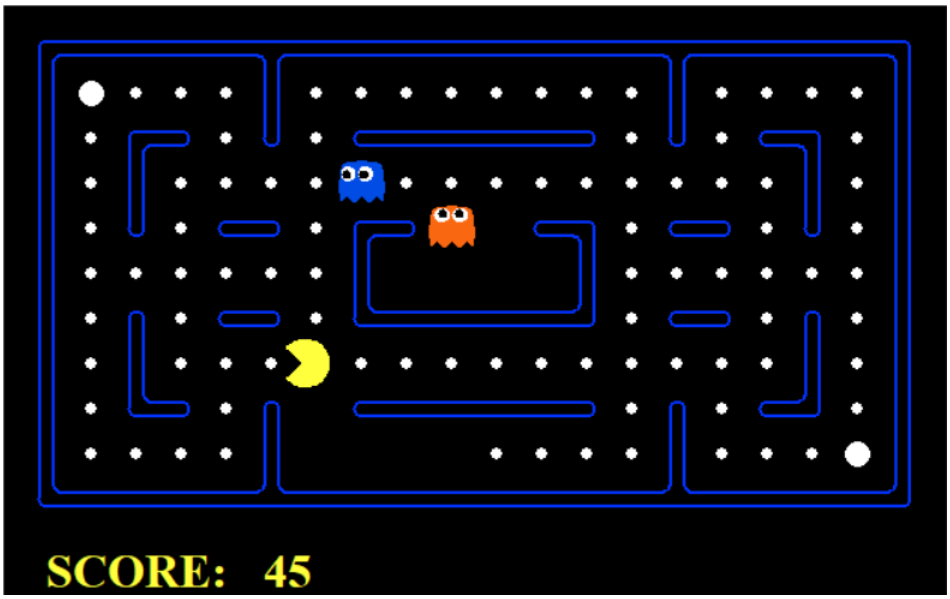
Giảng viên:
TS. Lương Ngọc Hoàng

Mục lục

Danh sách hình vẽ

1. Tổng quan về Pacman

Một màn chơi Pacman sẽ gồm có các Ghost, Food và Pacman. Như Hình 1 dưới đây, các Ghost có màu xanh và cam, còn Pacman là hình tròn khuyết màu vàng. Mục tiêu của Pacman là ăn tất cả Food có trong map và đạt được Score càng cao càng tốt, Score sẽ bị trừ theo thời gian. Để làm được điều này, đầu tiên Pacman phải không chạm Ghost trên đường tìm Food. Một lưu ý rằng nếu Ghost đang suy yếu, có màu trắng, thì Pacman có thể ăn được Ghost.



Hình 1: Màn chơi Pacman cơ bản.

2. Evaluation Function

Bảng 1: Điểm số tương ứng với mỗi hành động tác động lên các đối tượng của Pacman.

Object	Score
Food	10
Eat Ghost	200
Win	500
Capsule	0
Time per second	-1
Loss	-500

Hàm đánh giá là rất cần thiết để Pacman có thể đưa ra những quyết định hợp lí. Một ý tưởng rất tự nhiên đó là sử dụng càng nhiều đặt trưng có sẵn, sau đó tổ hợp chúng lại. Cụ thể, các đặc trưng được sử dụng là:

- `currentGameState.getScore()`: Đặt là F_1. Hướng về state ở đó điểm số sẽ tăng. Tuy nhiên, nếu chỉ sử dụng đặc trưng này, thì Pacman sẽ bất động trừ khi Ghost tiếp cận bởi vì cây tìm kiếm có độ sâu thấp và các trạng thái xung quanh không khả thi.
- `mindisGhost`: Đặt là F_2. Khoảng cách ngắn nhất từ Pacman đến Ghost. Tránh trường hợp Pacman xao nhãng vào việc giữ an toàn.
- `ghostState.scaredTimer`: Hỗ trợ quyết định Eat Ghost. Sau khi thực hiện hành động sẽ được đặt là F_3.
- `numCap`: Đặt là F_4. Số lượng Capsule hiện tại. Nên ăn Capsule để có khả năng Eat Ghost, từ đó nhận được bonus. Sau khi thực hiện hành động sẽ được đặt là F_5.
- `numFood`: Đặt là F_6. Số Food hiện tại. Nên ăn hết Food để kết thúc màn chơi, tránh để thua cuộc.
- `closestFood`: Đặt là F_7. Hỗ trợ quyết định ăn Food gần nhất.
- `closest_capsule`: Đặt là F_8. Hỗ trợ quyết định ăn Capsule gần nhất.

Definition

Evaluation Function cuối cùng chính là

$$\texttt{score} = F_1 - 0.5F_2 + F_3 + F_5 - 0.5F_7 + F_8 + \frac{3F_2}{2F_4 + 3F_6 + 1}[F_2 < 6]$$

Dấu dương trong biểu thức trên tương ứng với các hạng tử F_1, F_3, F_5 và F_7 với mục đích hướng về state ở đó điểm số sẽ tăng, Eat Ghost nhiều nhất có thể, ăn Capsule càng nhiều càng tốt, nên ăn Capsule gần nhất.

Dấu âm trong biểu thức trên tương ứng với các hạng tử F_2 và F_7. Mục đích là để cực tiểu khoảng cách giữa Pacman và Ghost (tránh việc Pacman trốn tránh quá xa), cũng như khoảng cách giữa Pacman và Food.

Hạng tử cuối cùng trong biểu thức chỉ được sử dụng trong trường hợp Pacman cách Ghost một khoảng rất gần (< 6). Đây là tỉ số quan trọng nhất, có nhiệm vụ trung hòa giữa việc giữ khoảng cách an toàn trong phạm vi nào và việc tiêu thụ Capsule, Food ở mức độ nào. Có thể thấy được chẳng hạn, khi Pacman chú trọng giữ an toàn, thì việc ăn Capsule và Food (F_4, F_6) sẽ khó khăn hơn bởi vì lúc này chúng sẽ trở nên rất thưa thớt. Số 1 được cộng vào mẫu số để tránh việc mẫu bằng 0 khi chiến thắng.

Ngoài ra, F_3 và F_5 được gọi là các điểm số khuyến khích. Nghĩa là Pacman nên ăn Capsule và Ghost trong trường hợp Ghost bị suy yếu. Cụ thể, F_3 sẽ được cộng thêm một lượng `50/manhattan(Pacman, Ghost)` nếu Ghost bị suy yếu, ngược lại sẽ trừ một lượng `10/manhattan(Pacman, Ghost)`. F_5 sẽ được cộng thêm một lượng 50 để khuyến khích Pacman ăn hết Capsule, ngược lại sẽ trừ đi một lượng `10/F_4`.

Tất cả các trọng số khi cập nhật F_3, F_5 và hàm `score` ở trên đều được điều chỉnh một cách thủ công khi thực nghiệm với điều kiện đảm bảo tiêu chí về dấu như đã đề cập trước đó.

> **Remark.** Hàm lượng giá trên sẽ hiệu quả cao nếu với các màn chơi có nhiều Ghost và Capsule, đồng thời không gian phải rộng để có nhiều vị trí an toàn. Bởi vì khi đó các Ghost có vùng bao phủ rộng, nhờ vậy Pacman sẽ có khả năng đi khắp bản đồ và đưa ra quyết định hợp lí. Tuy nhiên, hàm này vẫn không phải tối ưu, các đặc trưng sử dụng vẫn còn quá lỏng lẻo. Hơn nữa, các trọng số đều được tinh chỉnh thủ công bằng việc thử sai, điều này cũng không được đảm bảo bởi lẽ để thiết kế được bộ trọng số thực sự tốt là vấn đề không đơn giản. Ngoài ra, trong quá trình hiện thực hóa, các xung đột từ các quy tắc tối ưu trong hàm số cũng có thể xảy ra.

3. Thực nghiệm

Bảng 2: Kết quả thực nghiệm 2 Evaluation Function trên 7 Layout với 3 thuật toán MinimaxAgent, AlphaBetaAgent và ExpectimaxAgent. Mỗi Layout được tiến hành 5 lần, mỗi lần với một random seed khác nhau, độ sâu là 2. Trung bình điểm số và tỉ lệ win thu được dưới đây. https://github.com/nAuTahn

Layout	MinimaxAgent		AlphaBetaAgent		ExpectimaxAgent	
	Scores (std)	Win rate	Scores (std)	Win rate	Scores (std)	Win rate
Default Evaluation Function						
trappedClassic	-88.4 (506.5)	2	-88.4 (506.5)	2	-88.4 (506.5)	2
capsuleClassic	-473.4 (31.6)	0	-473.4 (31.6)	0	-483.8 (51.2)	0
mediumClassic	-805.6 (1471.0)	1	-805.6 (1471.0)	1	-872.8 (1579.9)	1
minimaxClassic	109.6 (493.6)	3	109.6 (493.6)	3	109.6 (493.6)	3
testClassic	528.0 (28.5)	5	528.0 (28.5)	5	528.0 (28.5)	5
smallClassic	-57.8 (162.9)	0	-57.8 (162.9)	0	171.6 (271.2)	1
trickyClassic	176.6 (273.3)	0	176.6 (273.3)	0	465.0 (401.1)	0
New Evaluation Function						
trappedClassic	-88.4 (506.5)	2	-88.4 (506.5)	2	-88.4 (506.5)	2
capsuleClassic	149.4 (736.7)	2	149.4 (736.7)	2	-61.2 (568.6)	1
mediumClassic	1750.0 (214.1)	5	1750.0 (214.1)	5	1741.2 (196.6)	5
minimaxClassic	109.6 (494.0)	3	109.6 (494.0)	3	109.6 (494.0)	3
testClassic	562.8 (0.9)	5	562.8 (0.9)	5	563.6 (0.4)	5
smallClassic	1036.4 (151.8)	5	1036.4 (151.8)	5	867.8 (514.2)	4
trickyClassic	1715.8 (719.9)	1	1715.8 (719.9)	1	951.2 (396.9)	0

Remark. Có thể thấy, Evaluation Function mới tốt hơn Default Evaluation Function trên gần như tất cả Layout. Layout `mediumClassic` và `smallClassic` cho thấy hiệu quả rõ rệt nhất khi tăng Win rate lên 5/5 đối với MinimaxAgent và AlphaBetaAgent. Ngoài ra, trừ `trappedClassic`, các layout còn lại đều có điểm số trung bình lớn hơn rất nhiều. Ở layout khó như `trickyClassic`, việc tinh chỉnh trọng số có thể làm tăng đáng kể Win rate, tuy nhiên để giữ tính tổng quát cho đa số layout, bộ trọng số đã đề cập trong phần trước sẽ được sử dụng.

Remark. MinimaxAgent và AlphaBetaAgent cho ra kết quả tương đương nhau và chúng tốt hơn ExpectimaxAgent trên tất cả layout. Điều này có thể giải thích bởi vì MinimaxAgent và AlphaBetaAgent chỉ khác ở cơ chế pruning, về cách thức hoạt động thì giống nhau. Với ExpectimaxAgent, do bản chất của hàm lượng giá là để Pacman bám sát Ghost, nên việc tính trên kỳ vọng có thể làm Pacman dễ bị Ghost ăn dẫn đến kết thúc sớm. Bảng kết quả trên cũng thể hiện phần nào điều này, điểm số sau cùng tương đối cao nhưng Win rate thấp.

Bảng 3: Thời gian hoàn thành cả 5 lần thực nghiệm của ba thuật toán MinimaxAgent, AlphaBetaAgent và ExpectimaxAgent trên hàm lượng giá mới.

Layout	MinimaxAgent	AlphaBetaAgent	ExpectimaxAgent
trappedClassic	6.2	5.9	5.9
capsuleClassic	35.3	32.5	32.2
mediumClassic	57.3	42.1	52.2
minimaxClassic	6.5	6.4	6.4
testClassic	7.0	6.7	6.8
smallClassic	19.8	18.0	19.7
trickyClassic	251.7	180.3	349.5

Remark. Trong tất cả layout, AlphaBetaAgent cho thấy sự cải thiện đáng kể về thời gian. Cụ thể, so với MinimaxAgent thì AlphaBetaAgent với cơ chế pruning nên giới hạn trên cây tìm kiếm được giảm, do đó thời gian thực thi cũng giảm. Còn đối với ExpectimaxAgent, do sự khác nhau về Win rate nên khó để so sánh được. Nhưng nếu không xét đến Win rate, thì hiệu xuất của ExpectimaxAgent có thể tương đương với MinimaxAgent, riêng với `mediumClassic` và `testClassic` thì tốt hơn.

Demo cho layout `mediumClassic` với điểm số 2126: demo

Hình 2: Ván chơi tốt nhất.