ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

Nhập môn trí tuệ nhân tạo

Báo cáo mở rộng

Game PACMAN

GVHD: Nguyễn Đức Dũng
SV: Lê Thành Sơn - 1810481

TP. HỒ CHÍ MINH, THÁNG 1/2021

Mục lục

1 Giới thiệu về game Pacman

Pacman là một trò chơi arcade được phát triển bởi Namco Bandai. Người chơi sẽ điều khiển một nhân vật tên là Pacman để ăn hết các viên điểm để có thể hoàn thành màn chơi. Trên màn chơi có thể có các đối thủ (gọi là ghost) sẽ di chuyển ngẫu nhiên hoặc đuổi theo Pacman trên màn chơi. Khi chạm vào ghost, màn chơi sẽ kết thúc và Pacman bị thua. Trong một số màn, sẽ có các viên to hơn, khi ăn các viên này, các ghost sẽ nằm trong trạng thái hoản sợ. Lúc đó, Pacman có thể ăn các con ghost này, ghost bị ăn sẽ quay về vị trí ban đầu của nó.

2 Pacman framework

Ta sẽ sử dụng framework Pacman được cung cấp tại UC Berkeley CS188 Intro to AI. Framework này định nghĩa các lệnh đồ hoạ cũng như các agent di chuyển cho ghost. Người dùng chỉ cần hiện thực các hàm tìm kiếm và các agent phù hợp cho Pacman. Bên cạnh đó, các thiết kế của màn chơi có thể nhập được các màn chơi từ các file .lay, trong đó, tường được ký hiệu bởi ký tự %, các chấm điểm được ký hiệu bởi ., các viên to được ký hiệu bởi o, vị trí của Pacman và các ghost lần lượt được ký hiểu bởi các ký tự P và G.

Đối với framework sử dụng, có một số màn chơi đã được định nghĩa tương đối phù hợp để thử nghiệm. Khi Pacman ăn được một viên điểm, điểm số tăng lên 10, khi ăn các viên tăng sức mạnh, các ghost sẽ bị vào trạng thái hoảng sợ, lúc đó, khi ăn chúng, Pacman sẽ có thêm 200 điểm. Khi chạm phải ghost không hoảng sợ, Pacman sẽ bị trừ 500 điểm và trò chơi kết thúc.

3 Khi không có ghost

Khi không có thức ăn, việc chơi trò chơi tương tự việc tìm kiếm một mục tiêu từ các nhánh có thể trong màn chơi. Có thể giả sử độ dài của mỗi đoạn đường thẳng trong màn chơi chính là chi phí cho nhánh đó. Các ngã rẽ tại cuối mỗi đoạn đường có thể coi là các nhánh của đoạn đường đó.

Ta có thể thực hiện việc tìm kiếm mục tiêu với các thuật toán được giới thiệu trong chương trình như sau.

3.1 Deep-first search (DFS)

Thuật toán tìm kiếm DFS sẽ tìm kiếm đường đi có thể dài nhất từ một nhánh bắt đầu để tìm được mục tiêu. Nếu nhánh đó không thể tìm được sẽ chuyển sang nhánh khác chưa đi qua.

3.2 Breath-first search (BFS)

Thuật toán tìm kiếm BFS sẽ tìm kiếm từ gốc và lần lượt các nhánh con của mỗi node cho tới khi không còn tìm được ở độ sâu đó sẽ chuyển sang độ sâu tiếp theo.

3.3 Uniform cost search (UCS)

Các cách tìm kiếm DFS hay BFS có thể cho ra đường đi ngắn nhất tuy nhiên khi xét đến chi phí cho các lần di chuyển, đường đi ngắn nhất có thể chưa phải là đường đi tối ưu nhất. Giải thuật UCS cũng tìm kiếm tương tự 2 giải thuật đã được trình bày trên, tuy nhiên, nó có tính toán chi phí cần thiết để di chuyển đến đỉnh khác trước khi tiếp tục tìm kiếm.

Hai giải thuật tìm kiếm gọi là tương đương khi 2 giải thuật này tìm kiếm cùng số lượng các node với số lượng giống nhau, các node được tìm kiếm với cùng một thứ tự và đường đi trả về là giống nhau. Khi chi phí dàng cho mỗi node tìm kiếm là một hằng số dương, giải thuật sẽ tương đương với giải thuật tìm kiếm BFS, còn khi chi phí là một hằng số âm, giải thuật UCS sẽ tương đương với giải thuật DFS.

3.4 Tìm kiếm A*

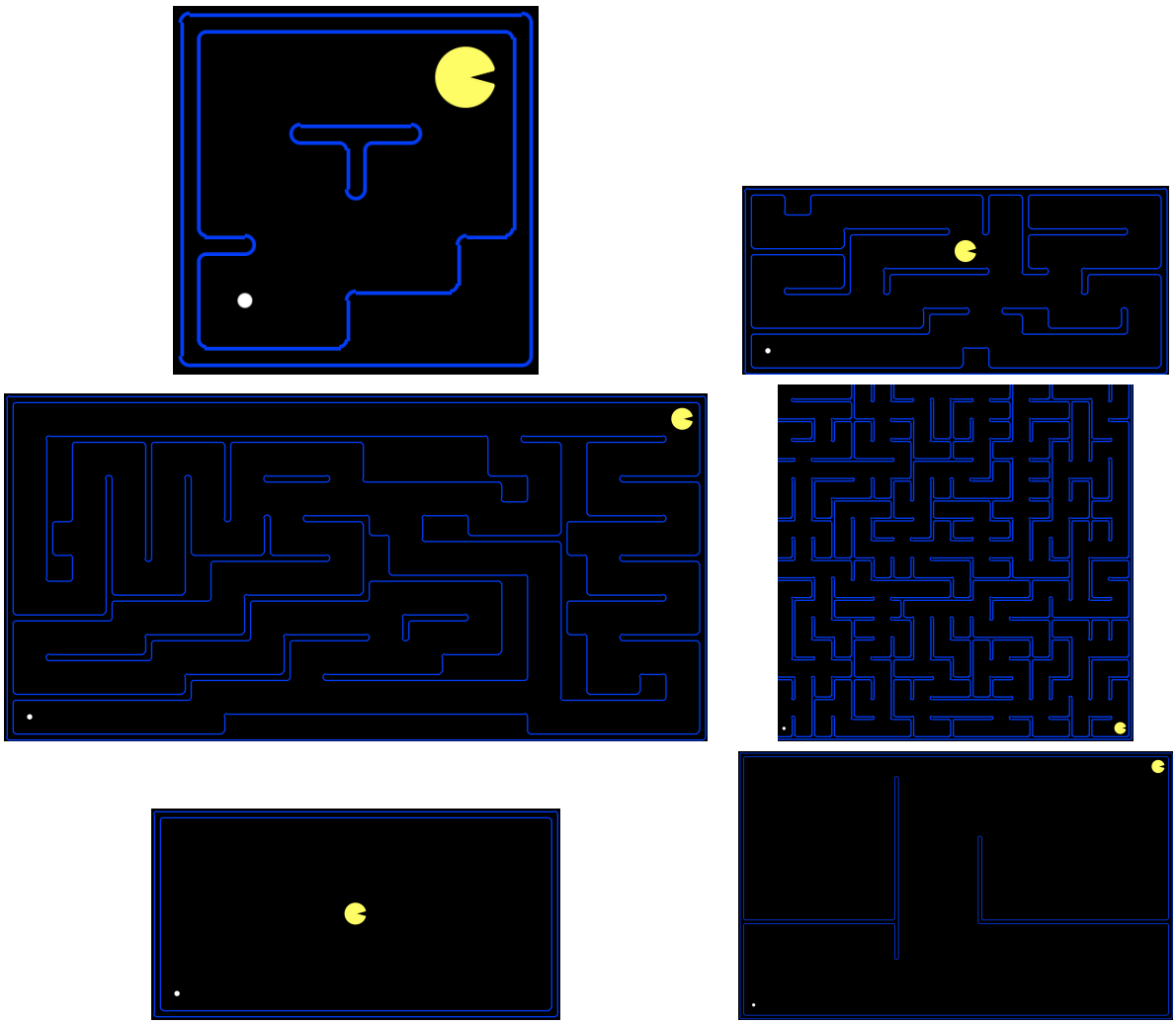
Giải thuật tìm kiếm A* dựa trên một hàm tìm kiếm kinh nghiệm $f(n) = g(n)+h(n)$ trong đó, $g(n)$ là chi phí cho đường đi từ vị trí xuất phát đến vị trí hiện tại, $h(n)$ là hàm ước lượng chi phí từ đỉnh hiện tại đến đỉnh trạng thái đích, thường sẽ được mong muốn càng thấp càng tốt. Giải thuật tìm kiếm A* luôn tìm được đường đi ngắn nhất nếu tồn tại đường đi như thế.

Trong mặt phẳng 2 chiều nơi Pacman di chuyển, có 2 hàm heuristics được định nghĩa: Manhattan và Euclid.

- Manhattan: dựa trên khoảng cách Manhattan giữa 2 điểm trên mặt phẳng toạ độ: Khoảng cách giữa 2 điểm $A(x_1, y_1)$ và $B(x_2, y_2)$ được tính bởi $AB = |x_1 - x_2| + |y_1 - y_2|$.
- Euclid: dựa trên khoảng cách Euclid giữa 2 điểm trên mặt phẳng toạ độ: Khoảng cách giữa 2 điểm $A(x_1, y_1)$ và $B(x_2, y_2)$ được tính bởi $AB = \sqrt{(x_1^2 - x_2^2) + (y_1^2 - y_2^2)}$.

3.5 Kết quả chạy

Thực hiện kiểm tra việc chạy các giải thuật đối với các layout dưới đây, ta thu được kết quả.



Hình 1: Một số layout không ghost

Kết quả chạy được của các giải thuật được cho như bảng dưới đây.

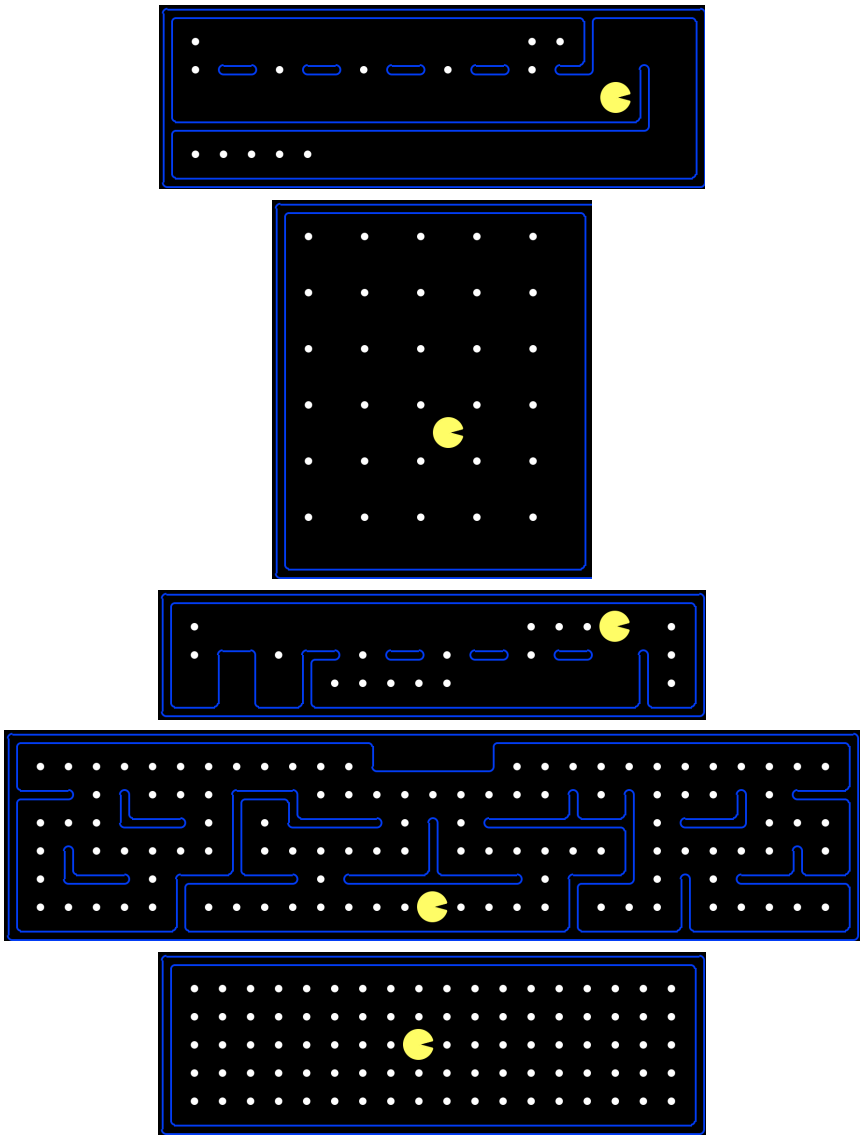
	DFS	BFS	UCS	A*(Mahattan Heuristic)	A*(Euclid Heuristic)
Tiny	10	8	8	8	8
Small	26	26	26	26	26
Medium	130	68	68	68	68
Big	210	210	210	210	210
Contours	85	13	13	13	13
Open	298	54	54	54	54

Bảng 1: Chi phí cần thiết để thực hiện tìm kiếm kết quả

	DFS	BFS	UCS	A*(Mahattan Heuristic)	A*(Euclid Heuristic)
Tiny	15	15	15	14	13
Small	76	86	86	45	48
Medium	146	269	269	221	226
Big	390	620	620	549	557
Contours	85	170	170	49	60
Open	806	682	682	535	550

Bảng 2: Số node cần tìm kiếm trước khi đưa ra kết quả

Ta sẽ tiếp tục thử nghiệm đối với các layout có nhiều điểm hơn.



Hình 2: Một số layout với nhiều thức ăn hơn

3.6 Nhận xét

Đối với trường hợp có nhiều hơn 1 điểm cần đến, các phương pháp bình thường có thể được dùng để giải quyết đối với trường hợp đơn giản trong trường hợp đầu tiên. Tuy nhiên, khi số lượng các điểm tăng lên hay cấu trúc trở nên phức tạp hơn, các phương thức như BFS, DFS, UCS, A* tỏ ra thiếu hiệu quả khi cần rất nhiều thời gian để thực hiện công việc tìm kiếm. Đặc biệt, A* có thể giải quyết đối một số layout đặc biệt (như tricky search) trong thời gian dưới 2 phút và ăn hết toàn bộ các điểm, trong khi các giải thuật khác

Đối với các layout có một điểm cần đến, giải thuật A* với hàm Mahantan Heuristic thường xét với số lượng node ít hơn so với số lượng node ở hàm Euclid Heuristic, điều này là do Pacman chỉ di chuyển trên các điểm nguyên trên mặt phẳng toạ độ, việc tính toán khoảng cách Euclid giữa 2 điểm có thể tạo ra nhiều điểm nguyên phù hợp.

Nhìn chung, các số liệu có thể thay đổi tuỳ theo layout được lựa chọn; giải pháp tìm kiếm A* sẽ mang đến kết quả tốt hơn (về số lượng node cần tìm và về chi phí) so với các phương pháp như UCS, BFS, DFS, việc chọn hàm heuristic phù hợp sẽ khiến cho bài toán tối ưu hơn. Bên cạnh đó, như đã nhận xét ở UCS, khi cost là một hằng số dương, nó tương đương với giải thuật BFS. Giải thuật BFS thường cho ra đường đi có số lượn node ngắn hơn nhưng chi phí không tối ưu nhất, đặc biệt, khi số lượng tường trở nên ít đi, đường đi của DFS sẽ trở nên dài hơn các phương pháp khác do tính chất của DFS.

4 Khi có ghost

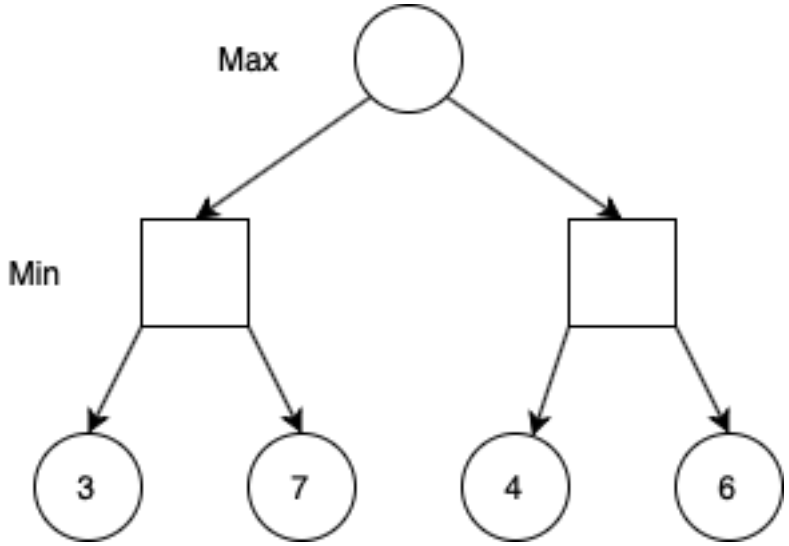
Khi có ghost, bài toán tương đương với việc có 2 người chơi lần lượt thực hiện các nước đi của mình để đạt được mục đích của mình. Một số giải thuật được trình bày bao gồm.

4.1 Minimax

Giải thuật minimax là một giải thuật để tìm nước đi tối ưu nhất cho mỗi người chơi với giả sử rằng các người chơi đều có cách chơi tối ưu nhất. Có 2 thành phần được chơi gọi là Maximizer và Minimizer, Maximizer sẽ cố gắn được điểm cao nhất trong khi Minimizer sẽ cố gắn được điểm thấp nhất. Nếu Maximizer được đi trước, có xác xuất cao giá trị cuối cùng là một số dương và ngược lại nếu Mimimizer đi trước.

Việc thể hiện các trường hợp có thể xảy ra đối với giải thuật này thường được biểu diễn bởi cây quyết định có ví dụ như sau.

Với ví dụ trên, nếu Maximizer đi bên trái, Minimizer sẽ đi về phía nhỏ hơn và



Hình 3: Ví dụ đơn giản về Minimax

sẽ là 3, kết quả sẽ đạt được là 3. Nếu Maximizer đi bên phải, Minimizer sẽ đi về phía nhỏ hơn và là 4, kết quả đạt được là 4. Do đó, bước đi tối ưu nhất cho cả 2 là đi về bên phải và giá trị tối ưu là 4.

Giải thuật Minimax luôn tìm được một lời giải nếu tồn tại trên một cây hữu hạn.

4.2 Alpha-Beta Pruning

Việc thực hiện giải thuật minimax đòi hỏi người dùng phải duyệt hết tất cả các nhánh có thể có của cây. Đối với cây có node nhiều nhánh và có độ sâu lớn, số lượng tìm kiếm có thể lên hàng mũ. Do đó, có phương pháp để loại bỏ đi những thông tin dư thừa trong quá trình tìm kiếm gọi là Alpha-Beta.

Alpha-Beta Pruning là một biến thể của giải thuật Minimax có thể tỉa đi lá hoặc một cây con của cây tìm kiếm nếu thoả điều kiện $\alpha \geq \beta$. Trong đó, α, β là các giá trị được cập nhật trong quá trình duyệt cây theo quy luật:

1. Maximizer sẽ cập nhật giá trị α, Minimizer sẽ cập nhật giá trị β.
2. Khi thực hiện backtracking, giá trị của node sẽ được chép vào node trên nó.
3. Chỉ truyền giá trị α, β xuống node con khi duyệt.

Trong trường hợp tối ưu nhất, giải thuật này sẽ đi sâu gấp đôi Minimax trong cùng một khoảng thời gian và loại bỏ rất nhiều thông tin thừa nhưng cũng có thể hoạt động tương tự Minimax nhưng chậm hơn do có các yếu tố α, β.

4.3 Expectimax

Bên cạnh các giải thuật được giới thiệu trong chương trình học, một giải thuật được gợi ý trong framework là Expectimax.

Giải thuật Minimax giả sử cả hai phía đều đưa ra các nước đi tối ưu. Tuy nhiên, trong trường hợp Minimizer có sai sót hoặc chơi không tối ưu, có thể dẫn đến một số kết quả khác với tính toán nếu dùng Minimax. Giải thuật Expectimax sẽ tính đến trường hợp này thông qua việc tính giá trị có thể đi vào nhánh đó. Giá trị này là trung bình có trọng số của nhánh đó với xác xuất đi vào nhánh đó.

Khi người chơi không đưa ra nước đi tối ưu, đôi khi, giá trị đạt được cuối cùng có thể cao hơn so với tính toán để cả 2 phía đều tối ưu như trong Minimax. Việc dùng Expectimax có thể khiến cho agent thua cuộc. Ngoài ra, nó cần phải duyệt toàn bộ cây nên không thể thực hiện các thuật toán cắt tỉa, có thể dẫn đến chậm.

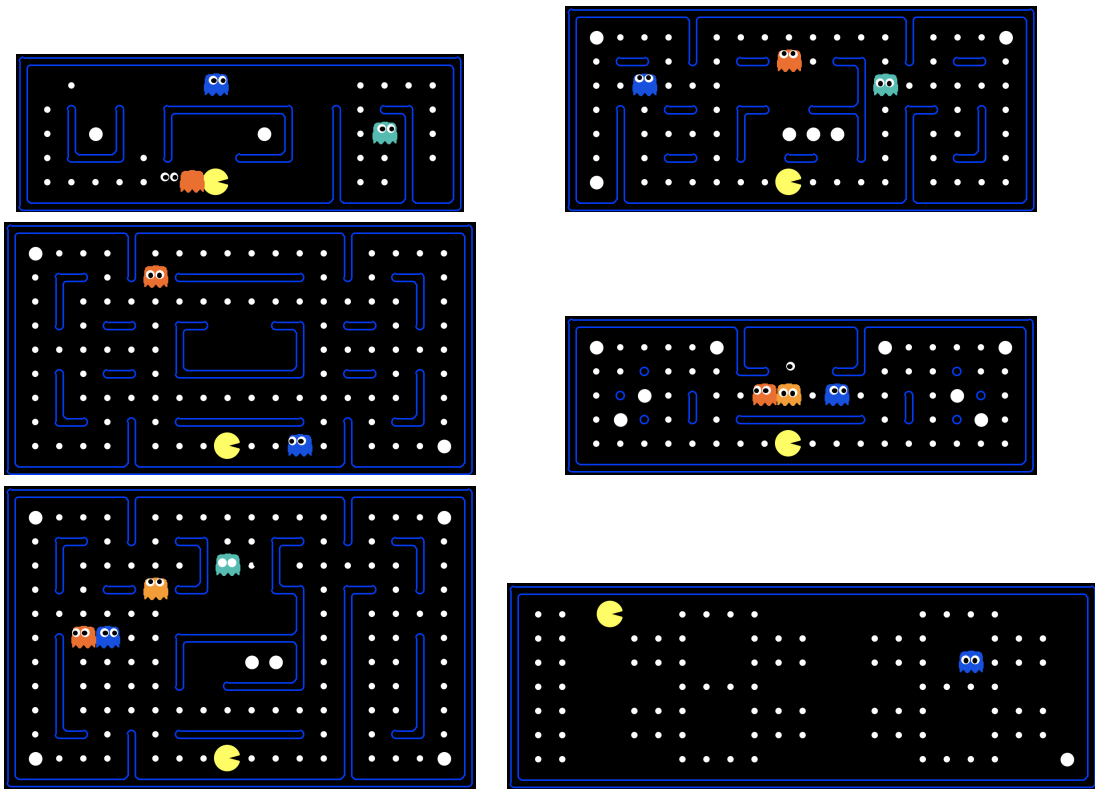
Trong trò chơi Pacman này, Pacman sẽ muốn di chuyển tối ưu nhất còn các ghost có thể di chuyển ngẫu nhiên, do đó Pacman sẽ là Maximizer còn các ghost sẽ là Minimizer có thể di chuyển không tối ưu trong giải thuật Expectimax.

4.4 Reflect agent

Framework cũng giới thiệu một agent đơn giản dành cho Pacman để có thể chơi tự động gọi là reflect. Reflect agent được hiện thực sẽ cố gắng đi gần với điểm và tránh xa các ghost nhất. Khi đi gần điểm, biến score sẽ tăng một số lượng còn khi gần ghost, biến đó sẽ giảm một số lượng. Biến này được cộng với giá trị của nước đi tiếp theo để xác định có thực hiện bước đó hay không.

4.5 Kết quả chạy

Đối với các layout to và khó đoán, việc chạy thử có thời gian quá lâu, các kỹ thuật được trình bày không phù hợp với các layout này nên chỉ có thể kiểm tra được trên các layout nhỏ. Một số kết quả thu được được trình bày bởi các bảng dưới.



Hình 4: Một số layout có ghost

	Lần 1	Lần 2	Lần 3	Lần 4	Lần 5	Trung bình
Reflex	-476	-476	-476	-476	-440	-468
Minimax (depth=3)	-40	-184	-167	-52	-461	-180.8
Alpha-Beta (depth=3)	-48	-448	-443	-481	-148	-313.6
Expectimax (depth=3)	223(thua)	57(thua)	-332	-441	-475	-193.6

Bảng 3: Kết quả 5 lần chạy của các agent đối với layout capsule với 3 ghost

	Lần 1	Lần 2	Lần 3	Lần 4	Lần 5	Trung bình
Reflex	-259	-476	-476	-250	-440	-387.4
Minimax (depth=3)	-765	-791	-501	-721	-381	-631.8
Alpha-Beta (depth=3)	-759	-1013	-1119	-451	-1195	-907.4
Expectimax (depth=3)	-546	-256	-514	-324	-832	-494.4

Bảng 4: Kết quả 5 lần chạy của các agent đối với layout capsule với 1 ghost

	Lần 1	Lần 2	Lần 3	Lần 4	Lần 5	Trung bình
Reflex	479(thua)	-66	467(thua)	-384	1344	368
Minimax	-396	234(thua)	-200	-200	-229	-158.2
Alpha-Beta	-221	-343	140(thua)	-409	-863	-339.2
Expectimax	573	90(thua)	-247	-161	-217	7.6

Bảng 5: Kết quả 5 lần chạy của các agent đối với layout small

	Lần 1	Lần 2	Lần 3	Lần 4	Lần 5	Trung bình
Reflex	-504	532	531	-503	-502	-89
Minimax (depth=3)	-501	-501	-501	-501	-501	-501
Alpha-Beta (depth=3)	-501	-501	-501	-501	-501	-501
Expectimax (depth=3)	-502	532	-502	532	532	118.4

Bảng 6: Kết quả 5 lần chạy của các agent đối với layout trapped

4.6 Nhận xét

Trong các bài kiểm tra đối với các layout tương đối phức tạp, có nhiều trường hợp thuật toán sẽ dẫn đến việc thất bại cho dù đang có một số điểm cao. Ngoài ra, đa phần trung bình các kết quả sẽ dẫn đến một số âm.

Các thuật toán đã viết dựa trên xác định bước di chuyển của Pacman dựa trên các bước di chuyển của các ghost. Do các ghost di chuyển ngẫu nhiên, các kết quả đối với các giải thuật Minimax cũng như Alpha-Beta thường không được tối ưu nếu so sánh với kết quả của thuật toán Expectimax. Việc dựa trên các bước di chuyển ngẫu nhiên để đưa ra bước di chuyển cho Pacman cũng khiến cho trong nhiều trường hợp, Pacman không thể di chuyển do khoảng cách giữa Pacman và các ghost là quá xa, Pacman có thể gặp khó khăn trong việc phán đoán nước đi tiếp theo là ăn hay tránh xa các ghost. Do đó, có thể ưu tiên tìm kiếm các điểm gần Pacman để tiến tới.

Trong bài test với layout trapped, đối với thuật toán Minimax cũng như Alpha-Beta, Pacman sẽ có xu hướng lao thẳng vào các ghost để kết thúc sớm màn chơi khi nhận thấy không thể có lối thoát khác. Điều này xảy ra do các thuật toán ưu tiên tăng điểm cao nhất cho Pacman hoặc thấp điểm nhất đối với các ghost, nên kết thúc sớm nhất sẽ giải quyết việc này. Đối với thuật toán ExpectiMax, do đã xét các xác xuất ghost đi vào các nhánh không thuận lợi nên Pacman sẽ cố gắn đạt được điểm cao nhất trước khi hy sinh.

Thuật toán Expectimax tuy có thể cho ra kết quả tương đối tốt hơn so với các thuật toán khác nhưng thời gian chạy tương đối cao hơn so với thuật toán Alpha-Beta.

Đối với agent reflex dựa trên khoảng cách đối với ghost và điểm, đôi khi nó cho kết quả tốt hơn và đôi khi nó cho kết quả không tốt bằng các thuật toán khác do việc di chuyển của các ghost là ngẫu nhiên

5 Tổng kết

Bài báo cáo đã thực hiện cài đặt và thử nghiệm một số quan sát trên các layout có mẫu cũng như layout chỉnh sửa dựa trên framework đã trình bày.

Đối với trường hợp không có ghost, chỉ thực hiện tìm kiếm mục tiêu cụ thể, các giải thuật đã trình bày nhìn chung đều có thể tìm kiếm được kết quả dù có tối ưu (về chi phí hay đường đi hay không). Đối với trường hợp có ghost, với các giải thuật đã được trình bày trong chương trình, có thể giải quyết một số màn chơi của game Pacman trong một số trường hợp cụ thể, tuy nhiên Pacman có thể gặp phải các vấn đề về cách chơi khi đi đến những màn chơi phức tạp hơn.

Dự kiến có thể sử dụng các phương pháp như học máy, học tăng cường để có thể cải thiện về cách chơi game Pacman.

6 Tài liệu tham khảo

Tài liệu

[1] UC Berkeley CS188 Intro to AI – Course Materials-Pacman project

[2] Depth First Search or DFS for a Graph

[3] Breadth First Search or BFS for a Graph

[4] Uniform-Cost Search (Dijkstra for large Graphs)

[5] Các thuật toán cơ bản trong AI

[6] Mini-Max Algorithm in Artificial Intelligence

[7] Alpha-Beta Pruning

[8] Expectimax Algorithm in Game Theory

Các trang web được truy cập lần cuối lúc 16:00 ngày 10/1/2021.