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

THIẾT KẾ HÀM LƯỢNG GIÁ VỚI MINIMAX/ALPHABETA/EXPECTIMAX

Sinh viên thực hiện: Nguyễn Duy Đạt

MSSV: 20520435

Lớp: CS106.M21.KHTN

Giảng viên: Lương Ngọc Hoàng

1.Thiết kế hàm lượng giá

- Những đặc trưng sử dụng để ước lượng giá trị trạng thái:

Đặc trưng sử dụng	Mục đích
CurrentGameState.getScore()	Chọn các state giúp gia tăng điểm số. Tuy nhiên, nếu chỉ có một hệ số này, sẽ có thể làm Pacman không di chuyển khi cây tìm kiếm có độ sâu thấp và các không gian trạng thái xung quanh không giúp gia tăng điểm.
Len(foodList)	Việc tối đa điểm đạt được và tránh kéo dài thời gian thắng là rất quan trọng. Muốn có được 2 trạng thái này phải kết thúc trò chơi càng sớm càng tốt. Phải ăn hết thức ăn càng nhanh càng tốt.
foodDistance	Pacman cần được chỉ hướng đi khi bị đói, độ sâu tìm kiếm thì có hạn nên nếu thức ăn ở xa Pacman có thể bị đói.
ghostDistance	Pacman được đặt trong môi trường Adversarial, nếu chạm phải Ghost sẽ bị thua. Do đó cần phải tránh xa Ghost để không bị thua game.

Capsule	Việc tiêu thụ Capsule không giúp tăng điểm nhưng giúp tối ưu điểm số có thể đạt được lên đến 200. Tiêu thụ Capsule sẽ làm cho Pacman phân tâm vào việc kết thúc trò chơi, và nếu tiêu thụ Capsule không đúng lúc, Ghost ở quá xa, sẽ làm cho chúng ta tốn một lượng vô ích. Trong code có thể nhắm đến Capsule như sau: if closetCapsule > 10: closetCapsule = 100. Pacman sẽ chỉ tập trung vào Capsule khi ở đủ gần.
effectGhostDistance	Khi tiêu thụ Capsule, các Ghost bị dính hiệu ứng đặc biệt và Pacman có thể ăn chúng trong khoảng thời gian là 40 giây. Pacman hướng vào Ghost nhằm lấy điểm bonus khi dính phải hiệu ứng này.

➔ Công thức lượng giá tự thiết kế:

$$Score = 2 * currentGameState.getScore() - (len(foodList)) - foodDistanceEuclid - ghostDistance - 10 * (closestCapsule) - effectGhostDistance$$

2.Nhận xét:

- Tùy vào layout mà các đặc trưng trên có thể thay đổi rất khác do việc xung đột giữa các kết quả. Có một số layout thì số Ghost, food, Capsule rất nhiều làm tính toán bị chậm hoặc map nhỏ nên khả năng di chuyển của Pacman bị hạn chế.

- Việc khởi tạo giá trị rất quan trọng nếu đặt không đúng, Pacman có thể xảy ra bị đói.

3.So sánh

** (Bảng bên dưới win = 1 là thắng game, win = 0 là thua game)*

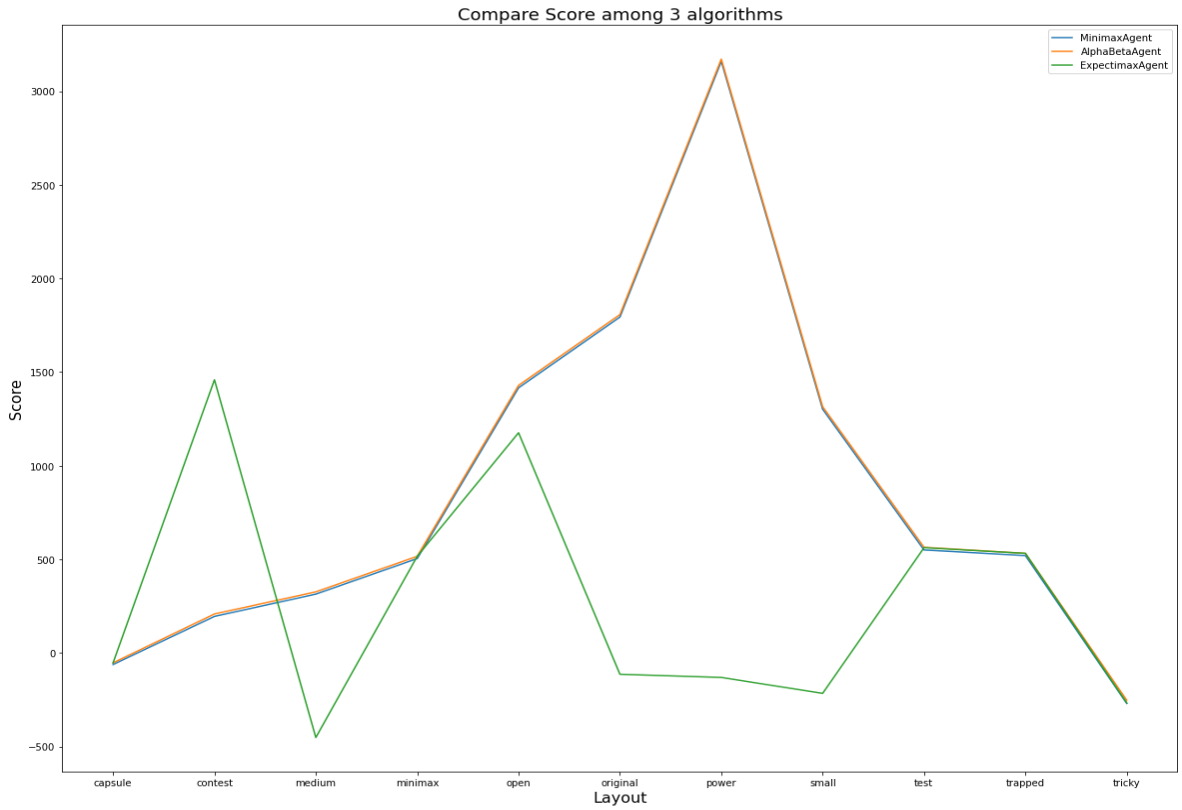
- Thử nghiệm các thuật toán đã cài đặt với evaluation function thiết kế ở trên (betterEvaluationFunction):

	MinimaxAgent		AlphaBetaAgent		ExpectimaxAgent	
Layout	score	win	score	Win	score	win

capsuleClassic	-53	0	-53	0	-53	0
contestClassic	208	0	208	0	1459	0
mediumClassic	326	0	326	0	-452	0
minimaxClassic	516	1	516	1	516	1
openClassic	1429	1	1429	1	1176	1
originalClassic	1807	1	1807	1	-114	0
powerClassic	3172	1	3172	1	-131	0
smallClassic	1315	1	1315	1	-216	0
testClassic	564	1	564	1	562	1
trappedClassic	532	1	532	1	532	1
trickyClassic	-253	0	-253	0	-267	0

+ Hầu như ở các layout, thuật toán MinimaxAgent và AlphaBetaAgent tốt hơn ExpectimaxAgent, nguyên nhân do ExpectimaxAgent phân chia xác suất làm cho Pacman đôi khi không nhận ra được nguy hiểm có thể dẫn đến việc kết thúc game do Ghost.

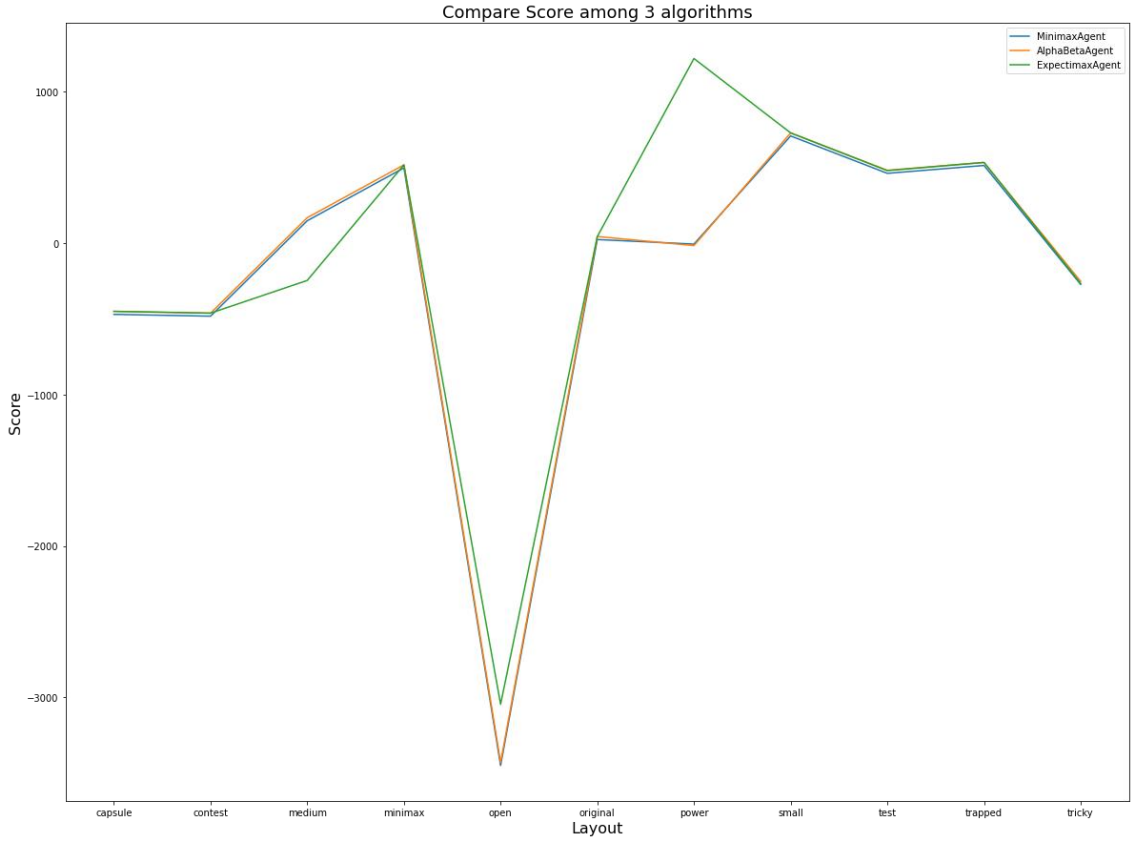
+ Score ở 2 thuật toán MinimaxAgent và AlphaBetaAgent tương tự nhau, do 2 thuật toán này tương tự nhau nhưng thời gian xử lí của AlphaBeta có thể sẽ tốt hơn do cơ chế pruning.



-Thực nghiệm các thuật toán với hàm lượng giá có sẵn (scoreEvaluationFunction):

	MinimaxAgent		AlphaBetaAgent		ExpectimaxAgent	
Layout	score	win	score	Win	score	win
capsuleClassic	-451	0	-451	0	-451	0
contestClassic	-463	0	-463	0	-463	0
mediumClassic	167	0	167	0	-247	0
minimaxClassic	516	1	516	1	516	1
openClassic	-∞	…	-∞	…	-1045	0
originalClassic	43	0	43	0	42	0
powerClassic	-17	0	-17	0	1218	0
smallClassic	727	1	727	1	727	1
testClassic	480	1	480	1	478	1
trappedClassic	532	1	532	1	532	1
trickyClassic	-253	0	-253	0	-267	0

+ Dựa vào bảng trên và biểu đồ bên dưới có thể thấy dù hàm tự thiết kế ở có thể vẫn chưa được tối ưu nhất, nhưng kết quả so với hàm có sẵn đã tốt hơn rất nhiều. Ví dụ như ở layout “openClassic”, MinimaxAgent và AlphaBetaAgent không giải được ở hàm lượng giá có sẵn nhưng hàm lượng giá tự thiết kế đã giải được .



4.Kết luận

- Thuật toán MinimaxAgent và AlphaBetaAgent tuy luôn có cùng kết quả thực nghiệm, nhưng ở AlphaBetaAgent thời gian xử lí thấp hơn so với MinimaxAgent nếu các nhánh có thứ tự sắp xết tốt hơn và độ sâu lớn hơn nhờ cải tiến Pruning so với MinimaxAgent.

- Thuật toán ExpectimaxAgent, việc thiết kế ước lượng theo giá trị kì vọng, có thể chưa phù hợp trong các layout hiện tại của Pacman, dẫn đến kết quả chưa thực sự tốt so với MinimaxAgent và AlphaBetaAgent.

- Việc thiết kế hàm lượng giá tốt có thể làm tăng đáng kể kết quả cho các thuật toán tìm kiếm. Độ sâu càng thấp, giúp tối ưu bộ nhớ thì lại làm cho hàm ước lượng trở nên phức tạp hơn, mất nhiều thời gian để tính toán hơn. Do đó, việc đánh đổi này cần được xem xét kĩ để có được hiệu suất tối ưu nhất.

5.Link drive 1 ván chơi tốt nhất (3172 điểm)

- https://drive.google.com/file/d/1ghF4sl2_-RzX_LFRr3DMFGiAM2hu--1A/view?usp=sharing