ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

TRÍ TUỆ NHÂN TẠO

BT4: Evaluation functions for Minimax/AlphaBeta/Expectimax

GVHD: Lương Ngọc Hoàng

Lớp: CS106.O21

Tên: Lê Minh Nhựt

MSSV: 22521060

I. Thiết kế hàm lượng giá

a. Mục tiêu thiết kế

Mục tiêu là thiết kế 1 hàm lượng giác nhằm đạt được số điểm cao nhất trong trò chơi. Trong đó, Pacman cần né tránh các con ma để không bị con ma ăn thịt, tránh dẫn đến trạng thái thất bại; ăn capsule để săn các con ma trong trạng thái sợ hãi nhằm gia tăng điểm số và cố gắng ăn hết các chấm thức ăn để chiến thắng trò chơi.

b. Hàm lượng giác mới

Hàm lượng giá mới được thiết kế có công thức cuối cùng như sau:

current_state_score + food_score + (ghost_score + scared_ghost_score) * len(newGhostStates) + capsule_score + remain_food_score

Source code của betterEvaluationFunction:

```
def betterEvaluationFunction(currentGameState):


    newPos = currentGameState.getPacmanPosition()
    newFood = currentGameState.getFood()
    newGhostStates = currentGameState.getGhostStates()
    newCapsules = currentGameState.getCapsules()
    newScaredTimes = [ghostState.scaredTimer for ghostState in newGhostStates]


    #current state score
    current_state_score = 2*currentGameState.getScore()


    #food score
    foodList = newFood.asList()
    if foodList:
        closestFood = min([manhattanDistance(newPos, food) for food in
foodList])
        if closestFood:
            food_score = -closestFood
        else:
            food_score = 10
```

```
    else:
        food_score = 500


    #ghost score
    closestGhost = min([manhattanDistance(newPos, ghost.getPosition()) for
ghost in newGhostStates])
    if closestGhost:
        ghost_score = -3/closestGhost
    else:
        ghost_score = -500


    #scared ghost score
    if max(newScaredTimes) > 0:
        if closestGhost:
            scared_ghost_score =  - ghost_score + 3/closestGhost
        else:
            scared_ghost_score = 50
    else:
        scared_ghost_score = 0


    #capsule score
    capsule_score = None
    if newCapsules:
        closestCapsule = min([manhattanDistance(newPos, caps) for caps in
newCapsules])
        if closestCapsule:
            capsule_score = -3*closestCapsule
        else:
            capsule_score = 0
    else:
        capsule_score = 0
```

```
    #remain food score
    remain_food_score = -7*len(foodList)


    #final score
    return current_state_score + food_score + (ghost_score +
scared_ghost_score)*len(newGhostStates) + capsule_score + remain_food_score
```

c. Phân tích đặc trưng

- Đặc trưng đầu tiên là ***current_state_score*** vì chúng ta cần tối ưu hóa trò chơi nên cần phải tính điểm dựa trên hành động của Pacman và trạng thái hiện tại của trò chơi nhằm gia tăng điểm số. Nếu không có đặc trưng này có thể làm Pacman bị mất phương hướng hoặc di chuyển 1 cách ngớ ngẩn. Ở đây em thiết lập *current_state_score = 2*currentGameState.getScore()* vì qua quá trình thử nghiệm trọng số 2 là khá tốt để pacman di chuyển ổn định.

- Đặc trưng thứ 2 là ***food_score*** được tính dựa trên *closestFood* (khoảng cách đến chấm thức ăn gần nhất theo độ đo Manhattan). Theo đó, nếu còn thức ăn trên bản đồ thì khi *closestFood != 0*, em sẽ đặt trọng số là **-1** (*food_score = - closestFood*) có nghĩa là càng gần chấm thức ăn thì bị trừ càng ít điểm, càng xa thì bị trừ càng nhiều và khi chọn như vậy, Pacman sẽ không bị trừ quá nhiều ở các vị trí gần thức ăn, từ đó ưu tiên đi đến thức ăn để chiến thắng. Khi *closestFood = 0* tức là Pacman ăn chấm thức ăn thì *food_score = 10*, xem như là điểm thưởng khi nó ăn được chấm thức ăn nào đó. Trong trường hợp không còn chấm thức ăn nào thì đặt *food_score = 500* nhằm thúc đẩy Pacman ăn hết thức ăn để chiến thắng trò chơi.

- Đặc trưng thứ 3 là ***ghost_score*** được đánh giá dựa trên *closestGhost* (khoảng cách đến con ma gần nhất dùng độ đo manhattan). Trong đó, hệ số được đặt là *ghost_score = -3/closestGhost,* tức là càng gần con ma điểm sẽ bị trừ càng nhiều, càng xa con ma điểm sẽ bị trừ ít đi, **-3** là nhằm tăng điểm số bị trừ để Pacman sợ ma nhiều hơn và tránh xa chúng. Còn nếu *closestGhost = 0* (bị ma ăn) thì *ghost_score = -500* như một điểm phạt lớn để Pacman tránh bị ma ăn và thua trò chơi.

- Đặc trưng thứ 4 là ***scared_ghost_score*** đánh giá bằng trạng thái sợ hãi của con ma, đặc trưng này được xác định phụ thuộc vào *newScaredTimes* (thời gian các con ma ở trạng thái sợ hãi). Cụ thể nếu *max(newScaredTimes) > 0* thì ta đặt *scared_ghost_score = - ghost_score + 3/closestGhost*. Công thức này có nghĩa là khi có ít nhất 1 con ma nào đó trong trạng thái sợ hãi thì pacman sẽ không còn sợ chúng mà chuyển sang tấn công các con ma gần nó, *-ghost_score* là nhằm triệt tiêu đi việc sợ các con ma và *3/closestGhos*t là để tiến lại gần con ma trong trạng thái sợ hãi khi càng gần thì cộng càng nhiều điểm và nếu ăn được con ma sẽ được thưởng *50 điểm*.

Còn khi *max(newScaredTimes) = 0* thì *scared_ghost_score* được đặt là 0, tức là nó đang không ảnh hưởng tới trạng thái hiện tại.

- Đặc trưng thứ 3 và thứ 4 được kết hợp lại và nhân với *len(newGhostStates)* nhằm tăng độ ảnh hưởng của 2 đặc trưng này khi có nhiều ma xuất hiện, nếu càng nhiều ma thì Pacman sẽ càng thận trọng, còn khi ma ở trạng thái sợ hãi thì càng nhiều ma thì độ chiến của Pacman càng lớn để săn ma nhằm gia tăng điểm số. Cách hoạt động của 2 đặc trưng này có thể mô tả như sau: Khi ma ở trạng thái bình thường Pacman sẽ tránh xa các con ma, khi Pacman ăn capsule lúc này ma bị sợ hãi và Pacman sẽ chuyển từ trạng thái né tránh các con ma sang trạng thái săn ma để tìm kiếm điểm số.

- Đặc trưng thứ 5 là ***capsule_score*** được tính dựa vào *closestCapsule* (khoảng cách đến capsule gần nhất dùng độ đo manhattan). Đặc trưng này được đặt trọng số là **-3** cũng có nghĩa là càng gần capsule bị trừ càng ít và càng xa capsule bị trừ càng nhiều nhằm khuyến khích Pacman ăn capsule để ăn ma và gia tăng điểm số, ngoài ra **-3** là để *capsule_score* bị trừ điểm nặng hơn *food_score,* nhằm ưu tiên ăn thức ăn để chiến thắng thay vì mải mê ăn capsule để săn ma. Nếu không có capsule nào trên bản đồ thì đặc trưng này được đặt *capsule_score = 0* (không ảnh hưởng đến trạng thái).

- Đặc trưng thứ 6 là ***remain_food_score*** nhằm giúp Pacman có thể ưu tiên thức ăn hơn để chiến thắng dựa trên *len(foodList)*. Trọng số được chọn là **-7** vì qua thử nghiệm nó khá là tốt để Pacman ưu tiên thức ăn, nếu còn nhiều thức ăn trên bản đồ sẽ bị trừ nhiều điểm và lúc này Pacman cần ăn các chấm thức ăn để giảm số điểm bị trừ và kết thúc trò chơi. Ví dụ còn 10 viên thức ăn sẽ bị trừ thêm 70 điểm, còn 1 viên thức ăn sẽ chỉ bị trừ 7 điểm.

Nhận xét: Hàm lượng giá này vẫn chưa thật sự tốt vì khó khăn trong quá trình cân bằng giữa các trọng số. Sẽ có lúc Pacman bị vây quanh bởi các con ma và không thể xác định đường đi vì chấm thức ăn ở xa hoặc bị các con ma cản trước mặt làm cho Pacman bị ma ăn và thất bại. Nhưng nhìn chung nó hoạt động khá tốt với các random state và các màn đã thử nghiệm.

II. Thống kê và so sánh 2 hàm lượng giá

Chú thích:

Loss

Win

scoreEvaluation					
Layout	**Random State**	**Depth**	**Minimax**	**AlphaBeta**	**Expectimax**
capsuleClassic	22521060	3	-428	-428	-428
	22521061		-481	-481	-70
	22521062		-453	-453	-420
	22521063		-454	-454	-118
	22521064		-433	-433	-433
Average score			**-449.8**	**-449.8**	**-293.8**
Game win			**0**	**0**	**0**
contestClassic	22521060	3	-401	-401	1688
	22521061		-170	-170	1245
	22521062		664	664	-548
	22521063		-168	-168	-289
	22521064		708	708	555
Average score			**126.6**	**126.6**	**530.2**
Game win			**0**	**0**	**0**
mediumClassic	22521060	3	193	193	1497
	22521061		-233	-233	-108
	22521062		-93	-93	201
	22521063		-790	-790	-3027
	22521064		-56	-56	943
Average score			**-195.8**	**-195.8**	**-98.8**
Game win			**0**	**0**	**3**
openClassic	22521060	3	-inf	-inf	-inf
	22521061		-inf	-inf	-inf
	22521062		-inf	-inf	-inf
	22521063		-inf	-inf	-inf
	22521064		-inf	-inf	-inf
Average score			**-inf**	**-inf**	**-inf**
Game win			**0**	**0**	**0**
powerClassic	22521060	3	682	682	1241

	22521061		1396	1396	2701
	22521062		1377	1377	1298
	22521063		1292	1292	2120
	22521064		687	687	584
Avarage score			**1086.8**	**1086.8**	**1588.8**
Game win			**0**	**0**	**2**

betterEvaluation					
Layout	**Random State**	**Depth**	**Minimax**	**AlphaBeta**	**Expectimax**
capsuleClassic	22521060	3	1323	1323	238
	22521061		-430	-430	-430
	22521062		-450	-450	-451
	22521063		1819	1819	-239
	22521064		850	850	850
Average score			**622.4**	**622.4**	**-6.4**
Game win			**3**	**3**	**1**
contestClassic	22521060	3	2439	2439	1924
	22521061		-111	-111	2372
	22521062		1947	1947	2870
	22521063		2484	2484	2601
	22521064		2305	2305	3237
Average score			**1812.8**	**1812.8**	**2600.8**
Game win			**4**	**4**	**5**
mediumClassic	22521060	3	1664	1664	1851
	22521061		1601	1601	1604
	22521062		1660	1660	632
	22521063		2098	2098	1886
	22521064		1602	1602	1649
Average score			**1725**	**1725**	**1524.4**
Game win			**5**	**5**	**4**
openClassic	22521060	3	1411	1411	1422
	22521061		1423	1423	1409
	22521062		1420	1420	1406
	22521063		1417	1417	1423
	22521064		1417	1417	1401

Average score			**1417.6**	**1417.6**	**1412.2**
Game win			**5**	**5**	**5**
powerClassic	22521060	3	2314	2314	4552
	22521061		4842	4842	4750
	22521062		3067	3067	3649
	22521063		3465	3465	3840
	22521064		3229	3229	3437
Average score			**3383.4**	**3383.4**	**4045.6**
Game win			**5**	**5**	**5**

*Link video màn powerClassic, random state là 22521061 dùng AlphaBeta cho số điểm là 4842:

python pacman.py -l powerClassic -p AlphaBetaAgent -a depth=3,evalFn=betterEvaluationFunction -s 22521061 --frameTime 0

a. So sánh giữa 2 hàm lượng giá

So với scoreEvaluation chỉ có đặc trưng là điểm số ở trạng thái hiện tại, hàm betterEvaluation được thiết kế mới có thêm nhiều đặc trưng quan trọng như là khoảng cách với con ma gần nhất, thức ăn gần nhất, capsule gần nhất, số lượng thức ăn còn lại và thời gian con ma rơi vào trạng thái sợ hãi. Điều này giúp Pacman có nhiều thông tin hơn để tính toán nước đi tốt nhằm đạt được điểm số tối ưu hơn thay vì chỉ dựa vào điểm số ở trạng thái hiện tại. Nhìn vào 2 bảng ở trên ta có thể thấy điểm số trung bình và số game win ở hàm betterEvaluation đều tốt hơn rất nhiều so với hàm scoreEvaluation. Trong quá trình thử nghiệm, ta cũng thấy được hàm score thường xuất hiện những tình huống phân vân làm con ma không biết đi hướng nào chỉ đứng yên hoặc di chuyển qua lại 1 chỗ, còn hàm better hầu như di chuyển rất tốt và hiếm khi rơi vào trường hợp nêu trên do có nhiều đặc trưng để quyết định, ngoài ra hàm better cũng tận dụng rất tốt việc ăn capsule và săn ma để gia tăng điểm số.

b. So sánh giữa Minimax, AlphaBeta Prunning và Expectimax:

- Minimax có thể tìm kiếm toàn bộ cây trò chơi nếu không có cắt tỉa, điều này có thể tốn nhiều thời gian và bộ nhớ đối với các màn chơi phức tạp. Minimax sẽ đảm bảo tìm ra nước đi tốt nhất.

- AlphaBeta sẽ giảm bớt số lượng nút được kiểm tra bằng cách loại bỏ các nút không cần thiết khỏi cây tìm kiếm. Nhờ vậy nó tìm kiếm nhanh hơn so với Minimax nhưng vẫn đảm bảo tìm ra nước đi tốt nhất ở nút gốc.
- Expectimax xem xét các hành động dựa trên yếu tố ngẫu nhiên nên không chắc chắn với các nước đi của nó, do đó thường dẫn đến các kết quả không ổn định so với AlphaBeta/Minimax. Ngoài ra do tính ngẫu nhiên nên Expectimax cũng không thể cắt tỉa nên cũng tốn nhiều tài nguyên xử lý với các màn phức tạp.

Nhận xét: Nhìn vào điểm số ở 2 bảng ta có thể thấy Minimax và AlphaBeta đều cho kết quả như nhau ở tất cả các màn, trong quá trình thử nghiệm rõ ràng AlphaBeta cho lời giải nhanh hơn, cũng như Pacman di chuyển mượt mà hơn so với Minimax. Cả 2 thuật toán này đều cho lời giải tốt nhất với hàm lượng giá có được và nhìn vào bảng ta cũng thấy được điểm số của chúng ổn định hơn Expectimax vì thuật toán này phụ thuộc vào các trạng thái ngẫu nhiên từ đó quá trình tìm ra lời giải có thể thay đổi và dẫn đến thất bại một cách không thể tránh khỏi.

Ngoài ra, việc thiết kế hàm lượng giá tốt có thể làm tăng đáng kể kết quả cho các thuật toán tìm kiếm. Tuy nhiên đánh đổi lại là chi phí tính toán tăng lên vì vậy ngoài việc tìm ra 1 hàm lượng giá tốt ta cũng cần để ý đến chi phí tính toán để đạt được hiệu suất tốt nhất.