ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

TRÍ TUỆ NHÂN TẠO

LỚP: CS106.O21

Evaluation functions for Minimax/AlphaBeta/Expectimax

Họ và tên: Trần Đình Khánh Đăng

MSSV: 22520195

I. MÔ HÌNH HÓA Ý TƯỞNG

Trong hàm evaluation, em dùng chiến thuật ưu tiên thức ăn trước, trong trường hợp Pac-Man ở gần viên capsule thì nó sẽ đợi ma lại gần thay vì di chuyển, lúc ma ở rất gần thì Pac-Man sẽ ăn capsule và quay lại tấn công ma để được nhiều điểm hơn. Còn nếu Pac-Man không ở gần viên capsule và ma ở rất gần thì Pac-Man sẽ ưu tiên né ma.

Trong hàm evaluation, em dùng những đặc trưng bao gồm:

Đặc trưng	Trọng số
closestFood	-1.5
ghost_distance	1
len(foodList)	-50
closest_capsule	1
currentGameState.getScore()	2
maxScaredTimes	10
alphaFactor	100

- closestFood: Khoảng cách tới chấm thức ăn gần nhất.
  - Trọng số: -1.5 * closestFood nếu còn chấm thức ăn, ngược lại là 0.
  - Ý nghĩa: Nếu tồn tại chấm thức ăn ở gần, Pac-Man sẽ ưu tiên tiến lại gần hơn nữa, như thế khoảng cách sẽ giảm, dẫn tới việc penalty sẽ giảm (đồng nghĩa việc sẽ được điểm cao hơn). Còn nếu không tồn tại chấm thức ăn nào, trò chơi kết thúc nên điểm nhận được sẽ là 0.
- ghost_distance: Khoảng cách từ Pac-Man đến con ma gần nhất.
  - Trọng số: (-2 / closestGhost) nếu khoảng cách của ma và Pac-Man >= 2, ngược lại là 1 số vô cùng bé.

- Ý nghĩa: Nếu ma ở càng gần thì sẽ bị trừ càng nhiều điểm, từ đó ưu tiên né ma thay vì lại gần. Còn nếu khoảng cách của Pac-Man và ma < 2, Pac-Man sẽ có tỉ lệ chết cực kì cao nên bắt buộc phải né.

- `len(foodList)`: Số lượng thức ăn còn lại.
  - Trọng số: -50 * len(foodList)
  - Ý nghĩa: Nếu số thức ăn càng ít thì sẽ bị trừ ít điểm hơn, từ đó Pac-Man sẽ ưu tiên đi tới trạng thái còn ít thức ăn (đồng nghĩa việc ưu tiên ăn chấm thức ăn).
- `closest_capsule`: Capsule gần nhất.
  - Trọng số: -3 / `closestCapsule` nếu còn Capsule, ngược lại gán giá trị 100.
  - Ý nghĩa: Pac-Man sẽ không ưu tiên ăn Capsule, vì có thể dẫn tới tình trạng bị ma bao vây. Còn nếu không còn Capsule nào, giá trị sẽ là 100.
- `currentGameState.getScore()`: Điểm số của trạng thái hiện tại.
  - Trọng số: 2 * `currentGameState.getScore()`
  - Ý nghĩa: Tiến tới trạng thái có điểm cao hơn.
- `maxScaredTimes`: Thời gian lớn nhất của các con ma bị yếu đi.
  - Trọng số: 10 * `maxScaredTimes`
  - Ý nghĩa: Tiến tới trạng thái mà thời gian ma bị yếu là lớn nhất.
- `alphaFactor`: Dựa trên 2 thành phần con.
  - `min_ghost_distance`: Khoảng cách tới ma không bị yếu gần nhất.
  - `foodNearby`: Khoảng cách tới chấm thức ăn gần nhất.
  - Trọng số: 100 * 3 nếu 2 điều kiện đều thoả, ngược lại 100 * -1.
  - Ý nghĩa: Nếu `min_ghost_distance` đủ lớn và `foodNearby` đủ nhỏ, Pac-Man sẽ ưu tiên tiến về chấm thức ăn thay vì đi ra xa ma.

II. BẢNG THỐNG KÊ

Agent	Classic		RandomSeed				
			22520195	**22520196**	**22520197**	**22520198**	**22520199**
Minimax	power	sEF	2429	701	402	594	2055
		bEF	3296	3998	3426	3593	2639
	small	sEF	1487	941	-264	305	-168
		bEF	1686	1661	1382	1692	1661
	medium	sEF	577	1565	291	-781	53
		bEF	1921	1474	1704	1924	1718
	open	sEF	-INF	-INF	-INF	-INF	-INF
		bEF	1394	878	1343	1079	1338
	capsule	sEF	-445	-463	-49	-443	-454
		bEF	1402	372	-475	415	-517

1. Bảng thống kê Minimax

Agent	Classic		RandomSeed				
			22520195	22520196	22520197	22520198	22520199
AlphaBeta	power	sEF	2429	701	402	594	2055
		bEF	3296	3998	3426	3593	2639
	small	sEF	1487	941	-264	305	-168
		bEF	1686	1661	1382	1692	1661
	medium	sEF	577	1565	291	-781	53
		bEF	1921	1474	1704	1924	1718
	open	sEF	-INF	-INF	-INF	-INF	-INF
		bEF	1394	878	1343	1079	1338
	capsule	sEF	-445	-463	-49	-443	-454
		bEF	1402	372	-475	415	-517

2. Bảng thống kê AlphaBeta

Agent	Classic		RandomSeed				
			22520195	22520196	22520197	22520198	22520199
Expectimax	power	sEF	2011	1018	1074	2268	3104
		bEF	3838	2495	2599	3075	3808
	small	sEF	92	941	-265	1044	1323
		bEF	1565	1713	1743	1759	1309
	medium	sEF	1692	1565	291	-474	-146
		bEF	1906	1719	1534	1680	547
	open	sEF	-INF	-INF	-INF	-INF	-INF
		bEF	1374	927	1318	1285	1322
	capsule	sEF	-445	-463	-49	-445	-454
		bEF	178	-281	160	313	-517

3. Bảng thống kê Expectimax

Chú thích:

sEF: scoreEvaluationFunction

bEF: betterEvaluationFunction

: Win

: Lose

: Không thể giải trong vòng 10p

Lưu ý: *Mọi màn chơi đều được chơi với depth = 3.*

III. NHẬN XÉT

Hàm betterEvaluationFunction sẽ cho kết quả chơi tốt hơn scoreEvaluationFunction vì nhờ những đặc trưng khác được thêm vào mà Pac-Man có nhiều lựa chọn dẫn đến kết quả tốt hơn. Tuy nhiên hàm trên vẫn chưa thật sự tối ưu với một vài màn có tính chất đặc thù (ví dụ như capsuleClassic).

IV. KẾT LUẬN

- Minimax: Minimax là một thuật toán tìm kiếm cây trạng thái của trò chơi, trong đó người chơi cố gắng tối ưu hóa kết quả cho chính họ trong tình huống tối thiểu nhất (tức là người chơi đối thủ cố gắng tối đa hóa kết quả cho họ). Thuật toán này đảm bảo tìm ra nước đi tốt nhất có thể cho người chơi, nhưng có thể trở nên chậm chạp đối với các trò chơi có không gian trạng thái lớn.
- AlphaBeta: AlphaBeta là một cải tiến của thuật toán Minimax. Nó loại bỏ các nhánh không cần thiết trong cây tìm kiếm bằng cách sử dụng các giá trị "alpha" và "beta" để cắt tỉa các nhánh không quan trọng. Nhờ vào cách này, AlphaBeta thường nhanh hơn Minimax với cùng một số lượng trạng thái.
- Expectimax khác biệt với Minimax và AlphaBeta ở điểm này: thay vì giả định rằng đối thủ sẽ chọn động thái tối ưu nhất, Expectimax xem xét tất cả các động thái có thể của đối thủ và tính toán giá trị kỳ vọng của mỗi động thái. Điều này thích hợp cho các trò chơi mà đối thủ có thể hành động theo các chiến lược ngẫu nhiên hoặc không hoàn toàn tối ưu. Expectimax có thể hiệu quả hơn khi xử lý các trò chơi có sự không chắc chắn như trò chơi với xác suất hoặc các trò chơi nơi các đối thủ thực hiện hành động ngẫu nhiên.
  - Như vậy, về hiệu suất, Expectimax có thể nhanh hơn trong các trường hợp mà có nhiều sự không chắc chắn hoặc khi đối thủ thực hiện các động thái ngẫu nhiên. Tuy nhiên, AlphaBeta thường nhanh hơn Minimax và Expectimax trong hầu hết các trường hợp, nhất là khi không gian trạng thái lớn.

Video