TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────

BÀI TẬP LỚN

**TRÍ TUỆ NHÂN TẠO**

Đề tài: Game Pacman

|  |  |  |
| --- | --- | --- |
| **Nhóm số 3** | Phan Vũ Hồng Hải | 20141394 |
|  | Ngô Quang Hòa | 20141840 |
|  | Chu Tiến Khoa | 20142344 |
|  | Đỗ Mạnh Khoa | 20142347 |

**Mã lớp**: 91620

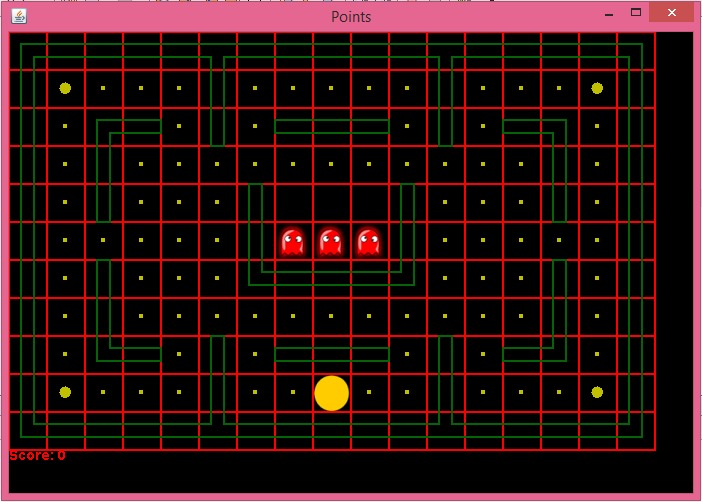
**Giảng viên**: TS. Nguyễn Nhật Quang

**Hà Nội, 11/2016**

**I, Giới thiệu về game Pacman**

**1, Luật chơi**

* Pacman và Ghost không thể di chuyển qua tường.
* Pacman phải ăn hết các viên đậu đề giành chiến thắng.
* Mỗi lần di chuyển Pacman bị trừ 1 điểm.
* Pacman thắng nếu ăn hết các viên đậu và được thưởng 500 điểm.
* Pacman bị Ghost bắt sẽ bị thua và trừ 500 điểm.
* Mỗi lần ăn đậu Pacman được thưởng 10 điểm.
* Pacman có thể bắt Ghost nếu như ăn được viên năng lượng, điểm thưởng là 200.
* Ghost không được dừng và chỉ có thể quay đầu nếu gặp ngõ cụt.
* Ghost sẽ chuyển sang trạng thái sợ nếu như Pacman ăn viên năng lượng, thời gian duy trì trạng thái là 40 bước đi.

**2, Bản đồ**

**Hình 1:** Bản đồ dạng lưới 11 x 17

Bản đồ dạng lưới 11 x 17 ô. Pacman được đánh chỉ số là 0, Ghost đánh chỉ số bắt đầu từ 1.

Có tất cả 96 vị cho Pacman và Ghost di chuyển, có 90 ô có thể thay đổi trạng thái (có hoặc không có đậu, viên năng lượng trên đó).

Pacman luôn nằm trong một ô trong lưới . Ghost luôn nằm trong một ô trong lưới nếu như ở trạng thái bình thường và có thể nằm ở giữa hai ô trong lưới nếu như đang sợ (đi chậm), lúc đó Ghost chỉ có thể di chuyển tiếp theo hướng cũ cho tới khi nằm trọn vẹn trong một ô.

Chi phí dịch chuyển giữa các ô trong lưới luôn là 1.

**3, Yếu tố môi trường**

* Môi trường quan sát được toàn bộ, tất cả các tác tử tham gia đều được quyền truy cập đến trạng thái đầy đủ của môi trường tại mỗi thời điểm.
* Chiến lược, trạng thái tiếp theo của môi trường được xác định hoàn toàn dựa trên trạng thái hiện tại và hành động của tác tử, ngoại trừ đối với hành động của tác tử khác.
* Liên tiếp, việc lựa chọn hành động của tác tử phụ thuộc vào các hành động trước đó.
* Tĩnh, khi các tác tử cân nhắc đưa ra hành động thì môi trường không thay đổi.
* Rời rạc, các nhận thức và hành động là hữu hạn và được phân biệt rõ ràng.
* Đa tác tử vừa cạnh tranh (Ghost và Pacman) vừa hợp tác (Ghost và Ghost).

**4, Nhận xét**

* Pacman là một dạng Grid game.
* Luật chơi đơn giản và dễ chơi.
* Tìm đường đi cho Pacman thuộc dạng toán NP – hard [[1]](#footnote-1).
* Số trạng thái cỡ rất lớn và không thể ước lượng chính xác do phụ thuộc vào việc di chuyển của Pacman.
* Áp dụng trí tuệ nhân tạo giúp tăng tính đối kháng với người chơi, thể hiện sự thông minh cho máy tính.

**5, Mục tiêu và phương pháp áp dụng**

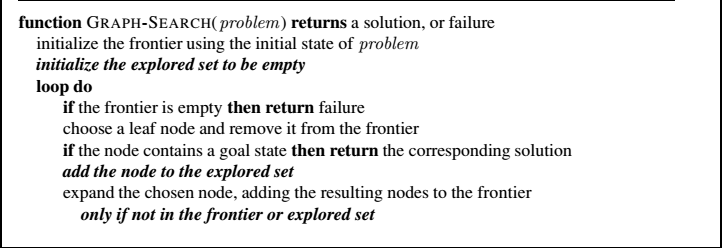
* Cài đặt các thuật toán thông minh cho cả Ghost và Pacman.
* So sánh hiệu quả giữa các thuật toán được cài đặt

**II, Phương pháp giải quyết**

**1, A\* áp dụng với Ghost**

A\* là thuật toán tìm đường đi ngắn nhất được áp dụng rất phổ biến do tính tối ưu của nó so với các thuật toán tìm kiếm khác.Tuy nhiên, trong môi trường đa tác tử thì việc tìm kiếm nước đi bằng A\* không phù hợp để áp dụng do tính biến đổi của môi trường. Vì thế, trong bài toán này, A\* được sử dụng như một phương tiện để đánh giá trạng thái của trò chơi.

Trong môi trường lưới, các trạng thái luôn có khả năng lặp lại. Do đó, trong bài này chúng em sẽ áp dụng thuật toán A\* dựa trên chiến lược Graph Search.



Hình 1. Thuật toán Graph Search

Cấu trúc explored lưu trữ các trạng thái đã được duyệt, cấu trúc frontier lưu trữ các trạng thái trong hàng đợi để duyệt. Nếu như một trạng thái đã ở trong explored thì nó sẽ không được thêm vào frontier và do đó loại được các trạng thái lặp. Trong cài đặt của chúng em, tên tương ứng của explored là Close và frontier là Open. Open được triển khai cấu trúc hàng đợi ưu tiên, Close được triển khai trên danh sách liên kết.

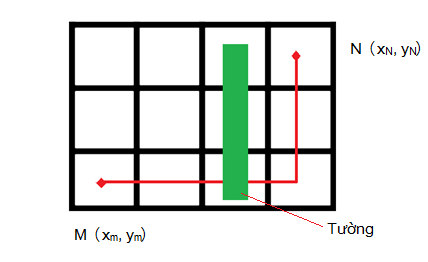
Mỗi nút trên cây tìm kiếm sẽ lưu trữ tọa độ của vị trí hiện tại, nút cha của mình, hành động của nút cha để chuyển sang nút hiện tại, chi phí từ nút đích đến nút hiện tại.Để tìm lại đường đi sau khi tới đích chỉ cần truy ngược lại cho tới khi đến được nút nguồn, các hành động được đưa vào stack. Từ stack ta sẽ lấy được đường đi từ vị trí đầu tới đích.

Cấu trúc hàng đợi ưu tiên Open lấy mức độ ưu tiên dựa trên hàm:

Trong đó, là chi phí từ vị trí gốc đến vị trí đang xét

là ước lượng chi phí từ vị trí hiện tại tới đích.

Giá trị càng nhỏ thì nút tương ứng có được lấy ra càng sớm.

 Tuy nhiên, để có được kết quả tối ưu, chiến lược Graph Search cần phải có hàm ước lượng thỏa mãn hai điều kiện là chấp nhận được (admissible) và kiên định (consistent). Trong môi trường dạng lưới, hàm ước lượng phù hợp nhất là ước lượng khoảng cách Manhattan (Manhattan Distance).

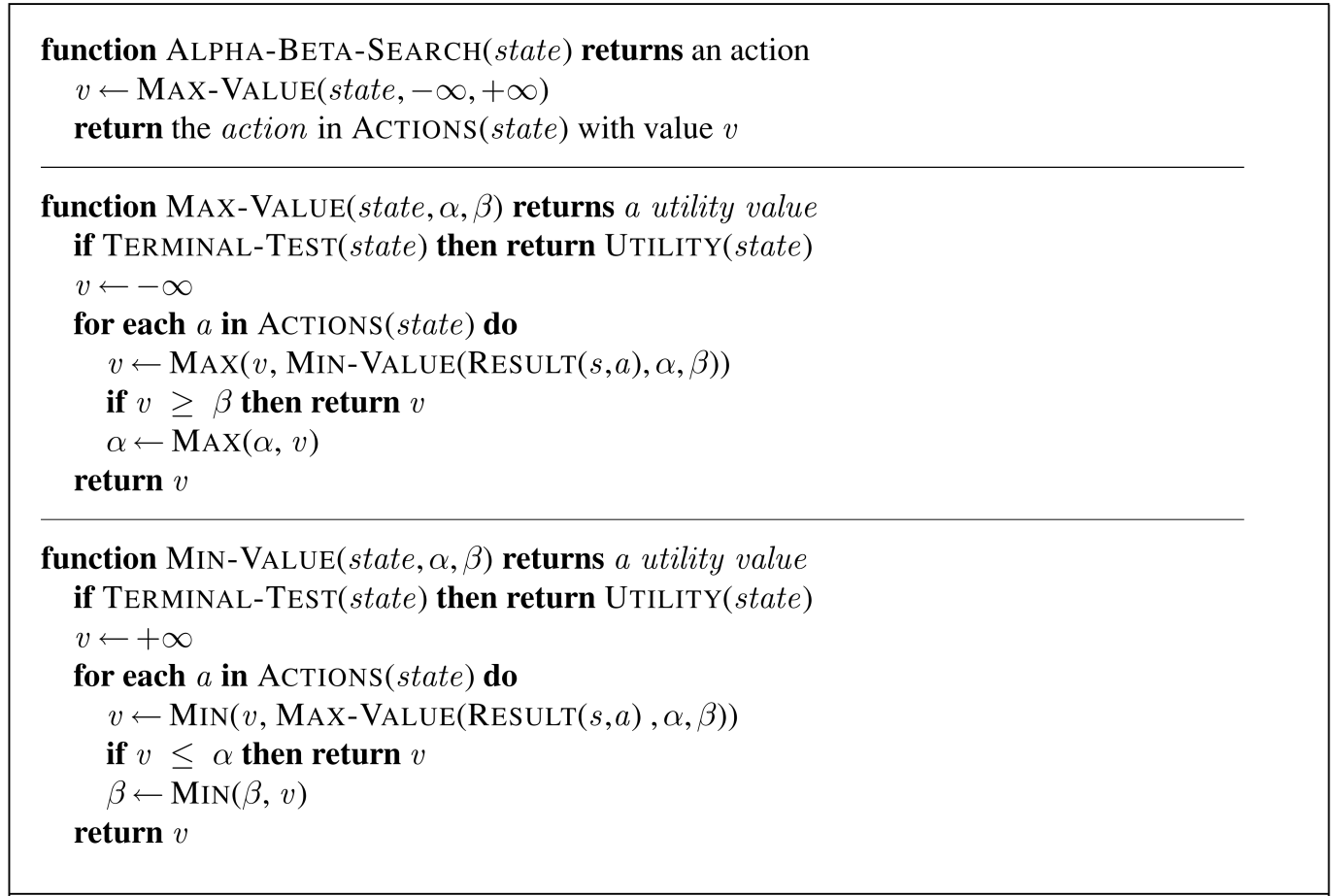
Hình 2. Minh họa Mahattan Distance

Giả sử A(xA, yA), B(xB, yB). Ước lượng khoảng cách Manhattan của A và B là

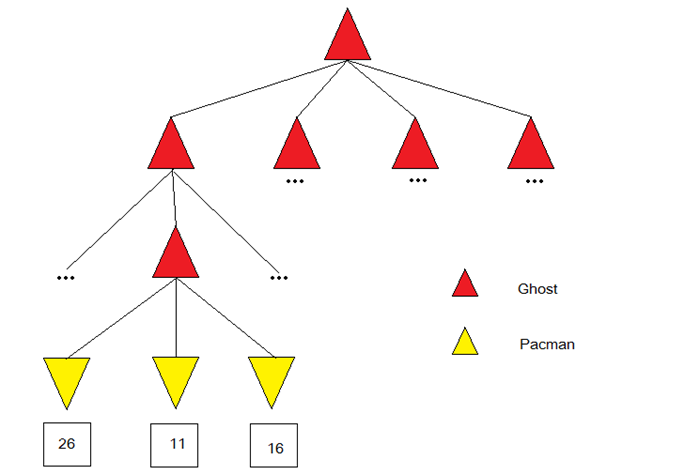
Trên hình vẽ, khoảng cách Manhattan giữa M và N là 5.

Ước lượng khoảng cách Manhattan là chấp nhận được vì chi phí dịch chuyển giữa các ô trong lưới của game tối thiểu là bằng với khoảng cách Manhattan. Ước lượng khoảng cách Manhattan cũng là kiên định vì chi phí mỗi bước đi là 1 nên có với là con của nút được sinh ra theo chiến lược tìm kiếm. Đồng thời, ưu điểm của ước lượng khoảng cách Manhattan so với các ước lượng khác là nó khá chặt do trong một số trường hợp chính là chi phí thực tế giữa điểm đầu và điểm đích. Ưu điểm tiếp theo chính là chi phí tính toán của ước lượng khoảng cách Manhattan thấp, nên chi phí tính toán giảm.

**2, Cắt tỉa α – β áp dụng với Ghost**

 Cắt tỉa α – β được phát triển dựa trên thuật toán Minimax. Ý tưởng cơ bản của thuật toán là loại bỏ việc xét các nhánh của cây tìm kiếm Minimax không đem lại kết quả tốt hơn các kết quả đã xét trước đó.

Hình 3. Thuật toán cắt tỉa α – β

Tuy thuật toán α – β đã loại bỏ các nước dư thừa, nhưng với bài toán thực tế thì việc tìm kiếm vét cạn là bất khả thi do không gian trạng thái rất lớn. Do đó cần phải dừng tìm kiếm tiếp ở độ sâu nhất định và đánh giá trạng thái thể hiện mức độ lợi thế của người chơi. Việc cắt cây tìm kiếm sử dụng một hàm để kiểm tra số lần xuất hiện của mỗi tác tử trong cây tìm kiếm. Ví dụ cây tìm kiếm có độ sâu 2 thì mỗi tác tử sẽ xuất hiện 2 lần trong cây tìm kiếm theo độ sâu. Dựa vào tổng số tác tử, ta có thể tính xem được có bao nhiêu tác tử theo độ sâu đã xuất hiện và cắt cây khi đã quá số lượng đó.

Hình 4. Minh họa cây tìm kiếm cắt tỉa α – β của Ghost với độ sâu 1

Khi thiết kế mô hình cây như trên, các tác tử Ghost sẽ trở thành cộng tác một các tự nhiên. Có thể diễn giải điều này qua việc Ghost ở vị trí nông hơn sẽ chọn hành động sao cho với hành động tối ưu của đồng đội, hành động của bản thân cũng là tối ưu.

Việc đánh giá trạng thái có tốt hay không phụ thuộc vào từng trò chơi và kinh nghiệm của người thiết kế thuật toán. Hàm đánh giá trạng thái trả về đánh giá theo cách nhìn của tác tử tìm kiếm. Khi cài đặt thuật toán cắt tỉa α – β cho Ghost, Ghost sẽ đóng vai trò là MAX, Pacman sẽ đóng vai trò là MIN. Vì thế, thiết kế hàm lượng giá cho Ghost sẽ là việc làm sao để cho lợi ích của Ghost là tối ưu, tức là điểm của Pacman phải là thấp nhất có thể.

Ở trạng thái bình thường, Ghost sẽ đạt được mục tiêu khi săn đuổi được Pacman, cho nên cần cho điểm cao những trạng thái mà Ghost gần Pacman. Nếu như Ghost ở trạng thái sợ, mục tiêu của Ghost là không để bị Pacman ăn, do đó cần cho điểm cao những trạng thái mà Ghost ở xa Pacman. Mặt khác, Ghost không muốn Pacman được điểm cao cho nên nó phải cản trở việc Pacman giành điểm. Qua phân tích trên, và dựa trên kết quả thử nghiệm, chúng em đã đưa ra hàm đánh giá trạng thái cho Ghost như sau:

Trong đó

scaredTime > 0 nếu Ghost ở trạng thái sợ, sacredTime = 0 nếu Ghost ở trạng thái bình thường.

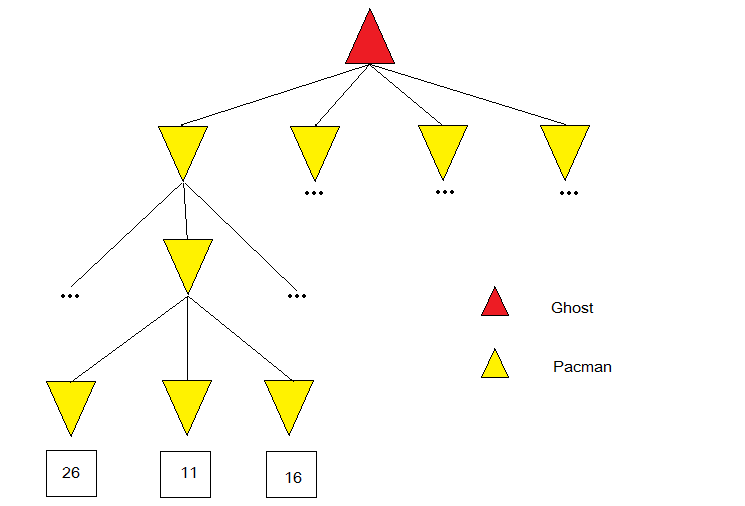
) là điểm của Pacman ở trạng thái S.

Bảng 1. So sánh giữa Ghost random và Ghost cắt tỉa α - β

|  |  |  |
| --- | --- | --- |
| Ghost | Điểm | Số lần chơi |
| 2 Random | 730 | 100 |
| 1 Random + 1 cắt tỉa α – β độ sâu 1 | 89 | 100 |

Hàm đánh giá trạng thái chạy tương đối tốt, nhưng chưa phải tốt nhất. Ở chương sau, chúng em sẽ trình bày về hàm đánh cải tiến sử dụng A\* và quan hệ khoảng cách giữa các Ghost.

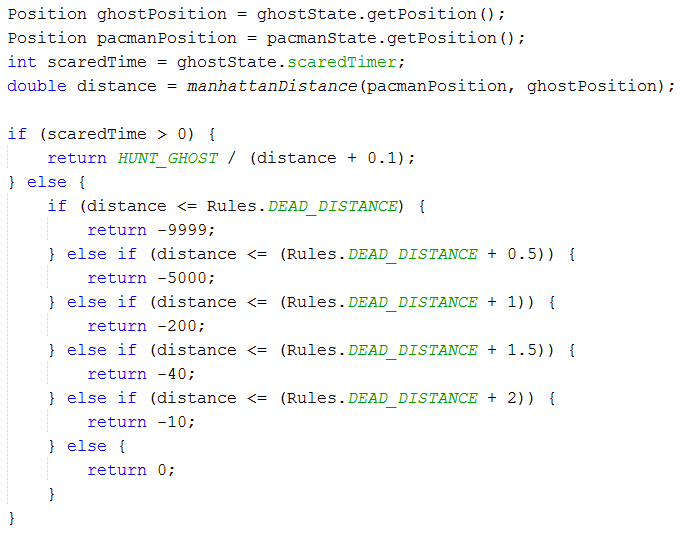
**3, Cắt tỉa α – β áp dụng với Pacman**

**Tương tự với việc áp dụng thuật toán cắt tỉa α – β với Ghost. Việc áp dụng thuật toán cắt tỉa α – β với Pacman coi Pacman là MAX, Ghost là MIN. Và mặc nhiên, hàm đánh giá trạng thái của Pacman dựa vào cái nhìn của Pacman để đánh giá độ tốt xấu của trạng thái.

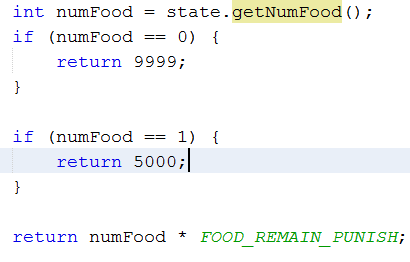
Tuy nhiên, thay vì hàm đánh giá đơn giản như của Ghost, hàm đánh giá của Pacman rất phức tạp do Pacman không chỉ muốn tránh bị ăn mà còn muốn thắng trò chơi với điểm cao nhất có thể. Vì thế, hàm đánh giá trạng thái của Pacman cần xem xét nhiều yếu tố như: khoảng cách đến tất cả các Ghost, số đậu và viên năng lượng còn lại, khoảng cách tới viên đậu và viên năng lượng gần nhất, điểm của trò chơi, … Dựa trên các phân tích đó và tinh chỉnh tham số qua quá trình kiểm tra thực tế, chúng em đã tìm được hàm đánh giá phù hợp.

Hình 5. Minh họa cây tìm kiếm cắt tỉa α – β của Ghost với độ sâu 1

Trong đó:

* m là tổng số Ghost
*  được biểu diễn như sau

Với HUNT\_GHOST = 300; DEAD\_DISTANCE = 0.7

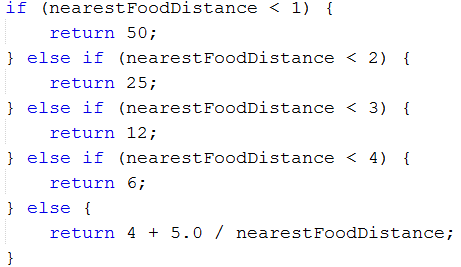
*  được biểu diễn như sau

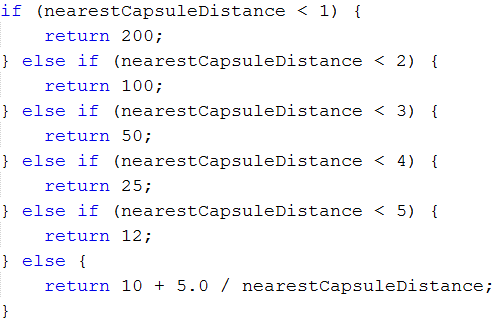
Với FOOD\_REMAIN\_PUNISH = -150.

* được biểu diễn như sau

Với CAPSULE\_REMAIN\_PUNISH = -150.

* được biểu diễn như sau



*  được biểu diễn như sau
* là điểm mà Pacman đang có ở trạng thái S.

Khi áp dụng thuật cả α – β cho Ghost và Pacman, ta gặp phải trường hợp tất cả các lần chạy thì kịch bản đều lặp lại như nhau. Điều này dễ hiểu vì chúng đều chọn nước đi tối ưu nhất, do đó mỗi lần chạy lại, nước đi tối ưu cũ lại được lặp lại. Do đó, để tránh trùng lặp, chúng em thêm một tác tử Ghost đi ngẫu nhiên vào Game và thống kê điểm số sau một số lớn lần chạy (1000 lần với Pacman máy và 100 lần với Pacman do người điều khiển) để có cái nhìn tổng quan nhất về tính hiệu quả của các thuật toán.

Sau khi cài đặt thuật toán thông minh cho Pacman, chúng em đã so sánh với người chơi và kết quả được mô tả ở bảng sau.

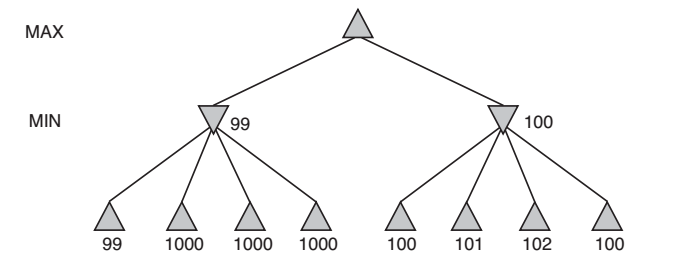
Bảng 2. So sánh người chơi với máy chơi

|  |  |  |
| --- | --- | --- |
| Ghost | Người chơi | Pacman máy độ sâu 3 |
| 1 Random | 1218 | 1405 |
| 1 Random + 1 cắt tỉa α – β độ sâu 1 | 89 | 1359 |
| 1 Random + 1 cắt tỉa α – β độ sâu 1 + 1 cắt tỉa α – β độ sâu 2 | -5 | 1303 |

Qua thống kê trên, ta có thể thấy thuật toán α – β cắt tỉa áp dụng cho Pacman cho kết quả rất cao so với người, đặc biệt khi chơi với các đối thủ “chuyên nghiệp”.

**3, Tăng hiệu quả của Pacman với Expectiminimax**

Cắt tỉa α – β được cài đặt với mục đích là chọn nước đi tối ưu với giả thiết là đối thủ cũng chọn nước đi tối ưu của họ. Điều gì xảy ra nếu đối thủ không thật sự “thông thái”? Ví dụ kinh điển dưới đây được trích trong quyển sách Artificial Intelligence: A Modern Approach.



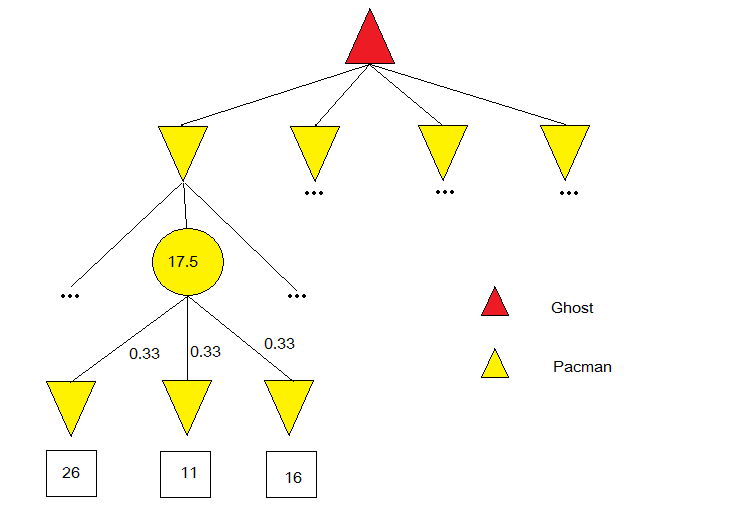
Hình 6. A two-ply game tree for which heuristic minimax may make an error.

Theo cây tìm kiếm, MAX sẽ chọn đi sang bên phải do có điểm số cao hơn. Tuy nhiên, nếu MIN không chọn nước đi tối ưu, MAX có thể sẽ nhận được lợi ích lớn hơn nếu đi sang bên trái. Xuất phát từ đó, chúng em muốn cải thiện hiệu quả của Pacman máy khi áp dụng trong môi trường có đối thủ không hành động tối ưu. Cụ thể với bài toán này là Ghost di chuyển ngẫu nhiên.

Thuật toán Expectiminimax chúng em đã áp dụng được trình bày như sau:

|  |
| --- |
| value(state):  if the state is a terminal state: return the state’s utility  if the next agent is MAX: return max-value(state)  if the next agent is MIN: return min-value(state)  if the next agent is EXP: return exp-value(state) |
| exp-value(state):  initialize v = 0  for each successor of state:  p = probability(successor)  v += p \* value(successor)  return v |
| max-value(state):  initialize v = -∞  for each successor of state:  v = max(v, value(successor))  return v |
| min-value(state):  initialize v = +∞  for each successor of state:  v = min(v, value(successor))  return v |

Cây tìm kiếm Expectiminimax của Pacman có dạng như sau



Hình 7. Minh họa cây tìm kiếm expectiminimax của Ghost với độ sâu 1

Có thể thấy thuật toán Expectiminimax rất giống với Minimax. Tuy nhiên, do việc tính kỳ vọng tại nút exp-value nên không thể cắt tỉa cây tìm kiếm, do đó độ sâu đạt được sẽ thấp hơn cây tìm kiếm dùng cắt tỉa α – β.

Kết quả so sánh với thuật toán cắt tỉa α – β với đối thủ là 3 Ghost:1 random, một Ghost cắt tỉa α – β độ sâu 2, một Ghost cắt tỉa α – β độ sâu 3 và dùng hàm lượng giá như mục II,2. Hàm lượng giá của Pacman giống ở mục II,3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Độ sâu | Thuật toán Pacman áp dụng | Điểm trung bình | Thời gian chạy (ms) | |
| Tối đa | Trung bình |
| 3 | Cắt tỉa α - β | 849 | 31 | 2 |
| 3 | Expectiminimax | 885 | 151 | 4 |
| 4 | Cắt tỉa α - β | 1113 | 100 | 8 |
| 4 | Expectiminimax | 1586 | 1008 | 56 |

Qua thống kê, kết quả của thuật toán Expectiminimax đưa ra rất tốt. Tuy nhiên, chi phí tính toán lại cao, với độ sâu 4 thì thuật toán Expectiminimax không hợp lí khi áp dụng vào game. Một lần nữa thực tế chứng minh độ thông minh càng cao thì chi phí tính toán càng lớn.

**5, Cải tiến hàm lượng giá của Ghost**

Áp dụng hàm lượng giá đã trình bày ở II,2 cho Ghost cho kết quả tốt, tuy nhiên chúng em muốn cải thiện hàm ước lượng này bằng cách áp dụng thuật toán A\* cùng với việc xem xét đến mối quan hệ giữa vị trí của Ghost trên bản đồ.

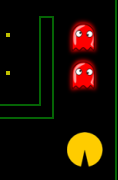
Hàm ước lượng mới được trình bày như sau

Trong đó:

AstartDistance(ghost, pacman) là khoảng cách được ước lượng bằng thuật toán A\*, đây sẽ là khoảng cách nhỏ nhất từ Ghost tới Pacman.

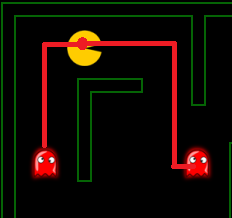
* n là tổng số Ghost ở trạng thái bình thường
* là điểm của game ở trạng thái S.

Chúng em nghĩ đến vì trong một số trường hợp, Ghost sẽ đi đi theo một hàng phía sau Pacman và chỉ tập trung ở một khu vực và điều đó không mai lại hiệu quả cao.



Hình 8. Ghost đi săn theo hàng

Đề giải quyết tình huống này, chúng em sẽ cho điểm cao nếu như các Ghost đi săn cách xa nhau, do đó sẽ giúp cải thiện hiệu quả hoạt động của Ghost.



Hình 9. Ghost đi săn cách xa nhau

Chúng em tiến hành chạy thử nghiệm với ước lượng mới. Pacman thử nghiệm dùng cắt tỉa α – β độ sâu 3. Gồm 3 ghost, một ghost random, hai ghost cắt tỉa α – β với độ sâu và hàm ước lượng nêu trong bảng

|  |  |  |
| --- | --- | --- |
| Ghost cắt tỉa α - β | Hàm lượng giá | Điểm |
| Độ sâu 1 + 2 | Cũ trong II, 2 | 1303 |
| Độ sâu 1 + 2 | Mới | 852 |
| Độ sâu 2 + 3 | Cũ trong II,2 | 846 |
| Độ sâu 2 + 3 | Mới | 735 |

**III, Chức năng và hướng dẫn sử dụng**

**1, Chức năng của hệ thống**

* Chơi game với máy, số lượng Ghost nằm trong khoảng từ 1 đến 3.
* Thực hiện cắt tỉa α – β với độ sâu tùy ý, số lượng Ghost nằm trong khoảng 1 đến 3.
* Thực hiện Expectiminimax với độ sâu tùy ý, số lượng Ghost nằm trong khoảng từ 1 đến 3.
* Lựa chọn hàm lượng giá của Ghost .
* Thống kê điểm trung bình, thời gian chạy tối đa và thời gian chạy trung bình của Pacman chạy cắt tỉa α – β và Exectiminimax.

**2, Hướng dẫn sử dụng**

* Chơi game với 3 Ghost bằng cách nhấn trực tiếp vào file ProjectAI.jar
* Sử dụng dòng lệnh cmd như sau

java –jar path/ProjectAI.jar [-option value]

path là đường dẫn tới thư mục chứa file ProjectAI.jar

Các option:

usage: Pacman [-albe <depth>] [-eval <evaluate function 0 or 1>] [-expect

<depth>] [-numGhost <number of ghost>] [-repeat <number of

iteration>]

-albe <depth> Pacman cat tia alpha - beta

-eval <evaluate function 0 or 1> Ham danh gia cua Ghost

-expect <depth> Pacman expectiminimax

-numGhost <number of ghost> So luong ghost

-repeat <number of iteration> So lan lap lai

* Ví dụ:

java –jar ProjectAI.jar –albe 3 –numGhost 3 – repeat 4

Sẽ chạy tự động với Pacman cắt tỉa α – β với độ sâu 3, số lượng Ghost là 3, lặp lại 4 lần. Sau đó sẽ in ra điểm, thời gian trung bình của 3 lần.

**III, Các gói phần mềm và tài liệu tham khảo**

* Java common CLI, Apache Common.
* Artificial Intelligence: A Modern Approach, Stuart J. Russell and Peter Norvig, 3rd edition.
* UC Berkeley**,** CS188 Pacman Project.
* Slide Trí tuệ nhân tạo.

**IV, Khó khăn và cách giải quyết**

* Các thành viên đa phần mới tiếp xúc với một đồ án có khối lượng công việc tương đối lớn nên kĩ năng lập trình và triển khai còn yếu, và mất nhiều thời gian để khắc phục lỗi.
* Việc lựa chọn đề tài gặp phải khó khăn do tất cả các thành viên đều có ý tưởng riêng và nhất quyết bảo vệ quan điểm của mình.
* Chưa quy định chặt chẽ nguyên tắc làm việc của nhóm nên phân công công việc không đồng đều.
* Không có source code Java chuẩn để kế thừa nên cần phải xây dựng tất cả từ đầu.
* Việc xây dựng hàm đánh giá trạng thái rất mất công sức do phải xác định và điều chỉnh các trọng số dựa trên kinh nghiệm.

**V, Kết quả đạt được và phát triển trong tương lai**

**1, Kết quả thu được**

* Đã xây dựng được một game đạt mức độ chuyên gia.
* Tìm hiểu và hiểu rõ về giải thuật A\* và tìm kiếm có đối thủ với cắt tỉa α – β và Expectiminimax.
* Thu được nhiều kinh nghiệm thiết kế, lập trình với Java và triển khai hệ thống tương đối phức tạp.
* Rèn luyện được tinh thần và kỹ năng làm việc tập thể.

**2, Phát triển trong tương lai**

Do không gian trạng thái tìm kiếm là lớn và việc xây dựng hàm đánh giá trạng thái là phức tạp nên chưa thể khẳng định ngay các tác tử được xây dựng đã là tốt nhất. Reinforcement Learning là ứng cử viên tiềm năng để giải quyết các thách thức trên. Cụ thể mô hình được áp dụng là Approximate Q – learning Feature – based. Ta không cần phải sử dụng cây tìm kiếm mà chỉ cần xây dựng một mô hình đánh giá ban đầu. Mô hình được xây dựng có thể là tuyến tính hoặc áp dụng mạng nơ ron. Việc tiếp theo là mô tả luật chơi và xác định trạng thái thắng thua và lợi ích của các hành động. Tác tử sẽ tự học qua những lần thắng và thua của bản thân và cải thiện trọng số của các mô hình đánh giá. Sau khi huấn luyện đủ số lần, mô hình đánh giá của tác tử sẽ rất chính xác. Kể từ đó, việc xác định chiến lược đi như thế nào chỉ cần áp dụng mô hình đã huấn luyện vào. Việc này đảm bảo đưa ra kết quả rất tốt trong một thời gian ngắn.

1. Giovanni Viglietta, Gaming is a hard job, but someone has to do it! [↑](#footnote-ref-1)