LAB 2 REPORT – ADVERSARIAL SEARCH

1. Thông tin sinh viên

MSSV: 20127062
Họ tên: Nguyễn Khải Phú

2. Thuật toán: Minimax alpha-beta pruning (heuristic function cho 5x5)

Minimax là thuật toán được áp dụng trong Trí tuệ nhân tạo (AI) hay lý thuyết trò chơi, tập trung vào việc giảm đến mức tối thiểu những tổn thất có thể nhận lấy sau mỗi bước đi. Giả sử hai người chơi luôn sử dụng chiến thuật tối ưu nhất để giành chiến thắng, thuật toán minimax có thể tính toán được giá trị thấp nhất/bước đi tệ nhất mà đối phương có thể đi ở những bước tiếp theo, từ đó giải ra được những bước đi phù hợp để ép đối phương lựa chọn các bước đi tệ nhất đó. Trong thực tế, thuật toán này rất khó được áp dụng một cách triệt để trong các trò chơi phức tạp như cờ vua, cờ vây, tic-tac-toe,… vì ta không thể tính toán tất cả các trạng thái có thể có trên bàn cờ. Chính vì vậy, cần sử dụng thuật toán này kèm theo một vài phương pháp khác như giải thuật alpha-beta pruning, hàm heuristic, tìm kiếm sâu dần (IDS),… Các phương pháp này rút ngắn thời gian tính toán một cách đáng kể và được áp dụng phổ biến trong các AI chuyên về chơi cờ.
Trong phương pháp giải được trình bày trong báo cáo này, giải thuật alpha-beta pruning được áp dụng cho cả 2 kích cỡ 3x3 và 5x5. Giải thuật alpha-beta pruning giảm bớt đáng kể số trường hợp cần phải xét theo như lý thuyết thuật toán minimax bằng cách loại các bước đi chắc chắn không phải tối ưu, chỉ giữ lại các bước đi tối ưu của mỗi bên để tính toán. Ngoài alpha-beta pruning, phương pháp này còn sử dụng hàm heuristic cho riêng bàn cờ 5x5.

2.1. 3x3

Tạo class Game() để quản lý tất cả thuộc tính của trò chơi (bao gồm bàn cờ, thứ tự chơi và hàm kiểm tra tính hợp lệ của mỗi nước đi). Class này cũng chứa hai hàm đệ quy tương hỗ *maxab* và *minab*, dựng lên theo thuật toán minimax.

```
function MAX-VALUE(state,α,β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← -∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a),α,β))
    if v ≥ β then return v
    α ← MAX(α, v)
return v
```

```
function MIN-VALUE(state,α,β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a) ,α,β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

Mã giả của hai hàm maxab và minab

Ý tưởng cơ bản của hai hàm này có thể mô tả như sau: 2 người chơi được gọi là MAX và MIN. Người chơi MAX hướng đến giá trị lớn nhất, còn người chơi MIN đối địch thì hướng đến giá trị lớn nhất, và hai người này liên tục chọn các bước đi tối ưu dành cho mình. Hai hàm maxab và minab sẽ lần lượt xét các giá trị tối ưu của hàm còn lại, để chọn ra con đường ép đối phương đi bước tệ nhất có thể, đồng nghĩa với nước đi tốt nhất dành cho bản thân. Mỗi hàm có một giá trị v lưu lại giá trị tốt nhất có thể đạt được. Trong trò chơi tic-tac-toe 3x3, có thể gán giá trị 1 cho nước đi giúp người chơi MAX thắng, -1 cho nước đi giúp người chơi MIN thắng, và 0 cho nước đi tạo ra kết quả hoà.
Giải thuật alpha-beta pruning bao gồm hai giá trị alpha và beta lưu giá trị tệ nhất mà người chơi này có thể ép người chơi còn lại phải nhận. Nếu trong quá trình xem xét, ta tìm ra bước đi tốt hơn bước đi tệ nhất, thì không cần phải xem xét kĩ hơn bước đi đấy, vì đó chắc chắn không phải là bước đi mà ta lựa chọn.
Đặt m là chiều sâu tối đa của cây trò chơi (9 đối với bàn cờ 3x3, 25 với bàn cờ 5x5), b là hệ số phân nhánh trung bình của cây (lần lượt là 4 và 12 cho 3x3 và 5x5) thì độ phức tạp về thời gian của minimax alpha-beta pruning là $O(b^m)$, độ phức tạp về không gian là O(m).

2.2. 5x5

Nếu chỉ áp dụng tương tự thuật toán minimax alpha-beta pruning như ở bàn cờ 3x3, thì độ phức tạp về thời gian của bàn cờ 5x5 rất lớn, bởi vì bàn cờ 5x5 có đến 25! trạng thái tổng cộng (so với 9! của bàn cờ 3x3). Chính vì vậy, cần giới hạn chiều sâu của thuật toán, kèm theo việc sử dụng hàm heuristic để hỗ trợ đánh giá các nước đi. Hàm heuristic sẽ đánh giá và "chấm điểm" cho trạng thái của bàn cờ sau mỗi nước đi. Các số điểm này sẽ được sử dụng trong thuật toán minimax, thay cho các giá trị 1, 0 và -1.
Về cách chấm điểm, hàm heuristic sẽ cộng 10, 100 và 1000 điểm cho các nước đi có thể tạo ra lần lượt 1, 2 và 3 điểm thẳng hàng cho người chơi MAX. Đặc biệt, nước đi giành chiến thắng, tức tạo ra 4 điểm thẳng hàng, sẽ có số điểm cách biệt 100000 để khiến thuật toán ưu tiên nước đi chiến thắng hơn. Ngược lại, hàm heuristic sẽ trừ 10, 100, 1000 và 100000 điểm cho các nước đi tạo ra lần lượt 1, 2, 3 và 4 điểm thẳng hàng cho nguòi chơi MIN. Cách đánh giá này xem nước đi để 2 người chơi chiến thắng có giá trị tuyệt đối tương đương nhau, vì vậy, cần phải giới hạn chiều sâu theo số chẵn (2, 4, 6, …) để AI ưu tiên nước thắng hơn so với chặn đường thắng của đối phương.

3. Nguồn tham khảo

[1] https://viblo.asia/p/thuat-toan-minimax-ai-trong-game-APqzeaVVzVe - viblo
[2] https://en.wikipedia.org/wiki/Minimax - Wikipedia
[3] https://github.com/deerishi/Tic-Tac-Toe-Using-Alpha-Beta-Minimax-Search
[4] https://stackabuse.com/minimax-and-alpha-beta-pruning-in-python/ - stackabuse.com
[5] https://www.pygame.org/docs/ref/pygame.html - pygame.org

[6] https://www.geeksforgeeks.org/tic-tac-toe-gui-in-python-using-pygame/ - GeeksforGeeks
[7] https://pythonguides.com/create-a-game-using-python-pygame/ - PythonGuides

4. Video

https://www.youtube.com/watch?v=YPokvW9GtZM