ĐẠI HỌC ĐẠI NAM

BÁO CÁO BÀI TẬP LỚN

Môn: Trí tuệ nhân tạo

Thuật toán AI trong cờ vua

Giáo viên hướng dẫn: Lê Trung Hiếu
Nhóm sinh viên thực hiện:
Nguyễn Viết Tâm - 1771020611
Hoàng Khắc Tùng - 1771020726

Hà Nội, 2025

Chương 1

Thuật toán Negamax với Cắt tỉa Alpha-Beta

1.1 Giới thiệu về thuật toán Negamax

Thuật toán Negamax là một biến thể của Minimax, được sử dụng trong các trò chơi hai người có tổng bằng không như cờ vua. Negamax hoạt động dựa trên nguyên tắc rằng nếu một nước đi có lợi cho một người chơi thì sẽ có bất lợi tương ứng cho đối thủ.

1.2 Cách thức hoạt động của thuật toán Negamax

Negamax đánh giá các nước đi có thể của người chơi, sau đó đảo ngược điểm số để tìm ra nước đi tốt nhất. Công thức cơ bản của thuật toán như sau:

$$score = -Negamax(trangthai\_moi, do\_sau - 1, -beta, -alpha)$$

Điều này giúp thuật toán đơn giản hơn so với Minimax vì không cần xử lý riêng biệt hai lượt đi của hai người chơi.

1.3 Tích hợp thuật toán Negamax vào chương trình

Trong chương trình AI cờ vua, Negamax được sử dụng để tính toán nước đi tốt nhất bằng cách duyệt cây trạng thái của trò chơi.

```
def negamax(game_state, depth, alpha, beta, turn_multiplier):
    if depth == 0:
        return turn_multiplier * evaluate_board(game_state)

    max_score = float('-inf')
    for move in game_state.getValidMoves():
        game_state.makeMove(move)
        score = -negamax(game_state, depth-1, -beta, -alpha, -
   turn_multiplier)
        game_state.undoMove()
        max_score = max(max_score, score)
        alpha = max(alpha, score)
```

```
        if alpha >= beta:
            break  #  C t   t a  Alpha-Beta
    return max_score
```

Listing 1.1: Triển khai thuật toán Negamax với Alpha-Beta

1.4 Giải thích từng phần trong thuật toán

1. **Kiểm tra độ sâu của thuật toán**: Khi đạt đến độ sâu quy định, thuật toán gọi hàm ‘evaluate$_b oard(game_s tate)$‘$dedanhgiatrangthaibanco2. * *duyetquatatcanuocdihople * * : ‘game_s tate.getV alidMoves()‘travedanhsachtatcanuocdicothethuchienohientai.3. * *thuchiennuocdiØequy** : ‘game_s tate.makeMove(move)‘AIkiemthunuocØisaudogoi‘negamax‘Øe
*capnhatgiatrialpha−beta** : giupogiamsoluongtrangthaicankiemtrabangcachloaibonhungnhanh
*hoantacnuocdisaukhidanhgia** : ‘game_s tate.undoMove()‘giupquaylaitrangthaicusaukhidiemdan$

1.5 Ví dụ về thuật toán Negamax trong thực tế

Dưới đây là một ví dụ cụ thể minh họa cách thuật toán Negamax đưa ra quyết định nước đi:

```
# nguoi choi di: tot trang tu e2 den e4
game_state.makeMove(Move((6, 4), (4, 4), game_state.board))

# AI tinh toan nuoc di tot nhatnhat
best_move = None
best_score = float('-inf')
for move in game_state.getValidMoves():
    game_state.makeMove(move)
    score = -negamax(game_state, 3, float('-inf'), float('inf'), -1)
    game_state.undoMove()
    if score > best_score:
        best_score = score
        best_move = move

# ai thuc hien nuoc di tot nhat
game_state.makeMove(best_move)
```

Listing 1.2: Ví dụ cụ thể về nước đi của AI

1.6 Kết quả mong đợi

1. Người chơi đi quân tốt từ e2 đến e4. 2. AI sử dụng thuật toán Negamax để tìm nước đi tốt nhất trong vòng 3 lượt tiếp theo. 3. AI có thể phản ứng bằng cách di chuyển tốt từ e7 đến e5 để kiểm soát trung tâm. 4. Trò chơi tiếp tục với nước đi mới từ AI.

Kết hợp với thuật toán Negamax, AI có thể đưa ra quyết định chính xác hơn dựa trên việc đánh giá các trạng thái có lợi và bất lợi trên bàn cờ.

Chương 2

Tổng quan về trò chơi cờ vua và các thuật toán AI

2.1 Giới thiệu về trò chơi cờ vua

Cờ vua là một trò chơi chiến thuật dành cho hai người chơi trên bàn cờ kích thước 8x8. Mỗi người chơi điều khiển một tập hợp các quân cờ gồm: vua, hậu, xe, tượng, mã và tốt. Mục tiêu chính của trò chơi là chiếu hết (checkmate) đối thủ, tức là đặt vua của đối thủ vào tình thế không thể di chuyển mà không bị bắt.

2.1.1 Lịch sử của cờ vua

Cờ vua có nguồn gốc từ Ấn Độ vào thế kỷ thứ 6 với tên gọi "Chaturanga". Trò chơi sau đó được truyền sang Ba Tư, châu Âu và phát triển thành hình thức hiện đại vào khoảng thế kỷ 15.

2.1.2 Luật chơi cơ bản

1. **Di chuyển quân cờ**: - Vua đi một ô theo mọi hướng. - Hậu có thể di chuyển ngang, dọc hoặc chéo bất kỳ số ô nào. - Xe di chuyển ngang hoặc dọc. - Tượng di chuyển theo đường chéo. - Mã đi theo hình chữ "L"(2 ô theo một hướng, sau đó 1 ô vuông góc). - Tốt di chuyển về phía trước một ô, có thể ăn quân chéo.

2. **Nhập thành**: Cho phép vua và xe đổi chỗ để bảo vệ vua.

3. **Phong cấp**: Khi tốt đi đến hàng cuối của đối thủ, nó có thể biến thành hậu, xe, tượng hoặc mã.

4. **Bắt tốt qua đường**: Khi một tốt di chuyển hai ô từ vị trí ban đầu, nó có thể bị bắt bởi một tốt đối phương nếu đứng ở vị trí chéo.

5. **Chiếu và chiếu hết**: Nếu vua bị tấn công mà không có nước đi hợp lệ nào để thoát, ván cờ kết thúc với chiến thắng của bên chiếu hết.

2.2 Thuật toán AI trong cờ vua

Để xây dựng một AI có thể chơi cờ vua, chúng ta cần sử dụng các thuật toán tìm kiếm và đánh giá nước đi.

2.2.1 Thuật toán Minimax

Minimax là thuật toán tìm kiếm trạng thái tối ưu trong các trò chơi có tổng bằng 0. Nó hoạt động bằng cách giả định rằng cả hai bên đều chơi hoàn hảo và cố gắng tối đa hóa lợi thế của mình trong khi tối thiểu hóa lợi thế của đối thủ.

2.2.2 Thuật toán Negamax

Negamax là một biến thể của Minimax giúp đơn giản hóa việc tính toán điểm số bằng cách sử dụng công thức:

$$score = -Negamax(trangThai\_moi, do\_sau - 1, -beta, -alpha)$$

2.2.3 Thuật toán Alpha-Beta Pruning

Cắt tỉa Alpha-Beta giúp tối ưu hóa Minimax bằng cách loại bỏ những nhánh tìm kiếm không cần thiết, giảm số trạng thái cần kiểm tra.

2.2.4 Hàm đánh giá bàn cờ

Hàm đánh giá bàn cờ giúp AI xác định trạng thái có lợi hay bất lợi. Một số tiêu chí đánh giá bao gồm: - Giá trị quân cờ (hậu = 9, xe = 5, tượng/mã = 3, tốt = 1). - Vị trí quân cờ trên bàn. - Khả năng kiểm soát trung tâm. - Số lượng nước đi có thể thực hiện.

2.3 Ví dụ về đánh giá bàn cờ trong Python

```python
def evaluate_board(game_state):
    score = 0
    piece_values = {"K": 0, "Q": 9, "R": 5, "B": 3, "N": 3, "p": 1}

    for row in game_state.board:
        for piece in row:
            if piece != "--":
                value = piece_values.get(piece[1], 0)
                score += value if piece[0] == 'w' else -value

    return score
```

Listing 2.1: Hàm đánh giá bàn cờ

2.4 Ví dụ thực tế về thuật toán AI đưa ra quyết định

Giả sử người chơi đi quân tốt từ e2 đến e4. AI sẽ tính toán nước đi tiếp theo dựa trên thuật toán Negamax với cắt tỉa Alpha-Beta:

```
# Nguoi choi ddi tot tu e2 den e4
game_state.makeMove(Move((6, 4), (4, 4), game_state.board))

# AI tinh toann nuoc di tot nhat
best_move = None
best_score = float('-inf')
for move in game_state.getValidMoves():
    game_state.makeMove(move)
    score = -negamax(game_state, 3, float('-inf'), float('inf'), -1)
    game_state.undoMove()
    if score > best_score:
        best_score = score
        best_move = move

# AI thuc hien nuoc di totot nhat
game_state.makeMove(best_move)
```

Listing 2.2: AI phản ứng với nước đi của người chơi

2.5 Kết luận chương 2

Chương này đã trình bày tổng quan về trò chơi cờ vua và các thuật toán AI phổ biến. Việc áp dụng thuật toán Minimax, Negamax kết hợp với cắt tỉa Alpha-Beta giúp AI có thể đưa ra quyết định tối ưu trong trò chơi.

Chương 3

Cài đặt và triển khai hệ thống AI cờ vua

3.1 Tổng quan về kiến trúc hệ thống

Hệ thống AI cờ vua được triển khai với các thành phần chính: - **Giao diện đồ họa**: Hiển thị bàn cờ, quân cờ và cho phép người chơi tương tác. - **Logic trò chơi**: Quản lý trạng thái bàn cờ, luật chơi, và kiểm tra điều kiện thắng/thua. - **Thuật toán AI**: Đưa ra quyết định nước đi dựa trên thuật toán Negamax với cắt tỉa Alpha-Beta.

3.2 Quản lý giao diện đồ họa

Chương trình sử dụng thư viện Pygame để hiển thị bàn cờ và các quân cờ.

```
import pygame as p
BOARD_WIDTH = BOARD_HEIGHT = 512
DIMENSION = 8
SQUARE_SIZE = BOARD_HEIGHT // DIMENSION
IMAGES = {}

def loadImages():
    pieces = ['wp', 'wR', 'wN', 'wB', 'wK', 'wQ', 'bp', 'bR', 'bN', '
   bB', 'bK', 'bQ']
    for piece in pieces:
        IMAGES[piece] = p.transform.scale(p.image.load("images/" +
   piece + ".png"), (SQUARE_SIZE, SQUARE_SIZE))
```

Listing 3.1: Giao diện đồ họa với Pygame

3.3 Quản lý logic cốt lõi

Hệ thống quản lý trạng thái bàn cờ và xử lý nước đi:

```
class GameState:
    def __init__(self):
        self.board = [
            ["bR", "bN", "bB", "bQ", "bK", "bB", "bN", "bR"],
```

```
            ["bp", "bp", "bp", "bp", "bp", "bp", "bp", "bp"],
            ["--", "--", "--", "--", "--", "--", "--", "--"],
            ["--", "--", "--", "--", "--", "--", "--", "--"],
            ["--", "--", "--", "--", "--", "--", "--", "--"],
            ["--", "--", "--", "--", "--", "--", "--", "--"],
            ["wp", "wp", "wp", "wp", "wp", "wp", "wp", "wp"],
            ["wR", "wN", "wB", "wQ", "wK", "wB", "wN", "wR"]
        ]
```

Listing 3.2: Quản lý trạng thái bàn cờ

3.4 Cài đặt thuật toán AI

Hệ thống sử dụng thuật toán Negamax với cắt tỉa Alpha-Beta để tính toán nước đi tối ưu.

```
def negamax(game_state, depth, alpha, beta, turn_multiplier):
    if depth == 0:
        return turn_multiplier * evaluate_board(game_state)

    max_score = float('-inf')
    for move in game_state.getValidMoves():
        game_state.makeMove(move)
        score = -negamax(game_state, depth-1, -beta, -alpha, -
   turn_multiplier)
        game_state.undoMove()
        max_score = max(max_score, score)
        alpha = max(alpha, score)
        if alpha >= beta:
            break  # Cat tia alpha - beta
    return max_score
```

Listing 3.3: Thuật toán Negamax với Alpha-Beta

3.5 Ví dụ thực tế

Dưới đây là một ví dụ minh họa về quá trình AI phản ứng với nước đi của người chơi:

```
# Nguoi choi di tot tu e2 den e4
game_state.makeMove(Move((6, 4), (4, 4), game_state.board))

# AI tinh toan nuoc di tot nhat
best_move = None
best_score = float('-inf')
for move in game_state.getValidMoves():
    game_state.makeMove(move)
    score = -negamax(game_state, 3, float('-inf'), float('inf'), -1)
    game_state.undoMove()
    if score > best_score:
        best_score = score
        best_move = move

# AI thuc hien nuoc di tot nhatnhat
```

```
16 game_state.makeMove(best_move)
```

Listing 3.4: AI phản ứng với nước đi của người chơi

3.6 Kết luận chương 3

Chương này đã trình bày chi tiết cách cài đặt và triển khai hệ thống AI cờ vua, bao gồm quản lý giao diện, logic trò chơi và thuật toán AI. Hệ thống hoạt động dựa trên việc tìm kiếm nước đi tối ưu bằng thuật toán Negamax với cắt tỉa Alpha-Beta.

Chương 4

Kiểm thử và đánh giá hệ thống AI cờ vua

4.1 Giới thiệu về kiểm thử hệ thống

Kiểm thử là một phần quan trọng trong quá trình phát triển phần mềm nhằm đảm bảo hệ thống hoạt động đúng như mong đợi. Đối với AI cờ vua, kiểm thử giúp đánh giá tính chính xác của thuật toán, tốc độ xử lý và khả năng phản ứng trước các nước đi của đối thủ.

4.2 Mục tiêu kiểm thử

Các mục tiêu kiểm thử chính của hệ thống bao gồm: - **Độ chính xác của AI**: Đánh giá xem AI có thể thực hiện nước đi hợp lệ và chiến lược tốt hay không. - **Tốc độ xử lý**: Xác định thời gian AI cần để tính toán nước đi. - **Khả năng phản ứng**: Kiểm tra xem AI có thể phản ứng đúng với các nước đi khác nhau của người chơi. - **Kiểm thử tính ổn định**: Đảm bảo hệ thống không bị lỗi hoặc treo khi xử lý các tình huống phức tạp.

4.3 Phương pháp kiểm thử

Hệ thống AI cờ vua được kiểm thử theo các phương pháp sau:

4.3.1 Kiểm thử chức năng từng thành phần

- Kiểm tra từng module như giao diện đồ họa, logic trò chơi và thuật toán AI. - Đảm bảo các quân cờ di chuyển theo đúng luật. - Xác minh rằng thuật toán Negamax với Alpha-Beta Pruning hoạt động chính xác.

4.3.2 Kiểm thử đối kháng giữa AI và người chơi

- Chơi thử nhiều ván cờ để xem cách AI phản ứng với các chiến thuật khác nhau. - Đánh giá AI trong các tình huống cụ thể như chiếu hết, bắt quân, nhập thành, phong

cấp. - So sánh kết quả với các engine cờ vua mạnh hơn để đánh giá mức độ thông minh của AI.

4.3.3 Kiểm thử hiệu suất

- Đo thời gian AI tính toán nước đi ở các độ sâu khác nhau. - Kiểm tra số lượng trạng thái mà AI duyệt qua trong mỗi lần tìm kiếm. - Tối ưu hóa thuật toán để đảm bảo tốc độ xử lý nhanh.

4.4 Ví dụ kiểm thử thực tế

Dưới đây là một ví dụ kiểm thử AI với nước đi cụ thể:

```
# Nguoi choi di tot tu e2 den e4
game_state.makeMove(Move((6, 4), (4, 4), game_state.board))

# AI tinh toan nuoc  i  tot nhat
best_move = None
best_score = float('-inf')
for move in game_state.getValidMoves():
    game_state.makeMove(move)
    score = -negamax(game_state, 3, float('-inf'), float('inf'), -1)
    game_state.undoMove()
    if score > best_score:
        best_score = score
        best_move = move

# AI thuc hien nuoc di tot nhat
game_state.makeMove(best_move)

displayBoard(game_state.board)

def displayBoard(board):
    for row in board:
        print(" ".join(row))
```

Listing 4.1: Kiểm thử nước đi của AI

Kết quả mong đợi: - AI tính toán và đưa ra nước đi tốt nhất trong vòng 3 lượt. - Nếu người chơi đi e2-e4, AI có thể phản ứng bằng e7-e5 để kiểm soát trung tâm. - Hiển thị trạng thái bàn cờ sau khi AI thực hiện nước đi.

4.5 Đánh giá kết quả kiểm thử

Sau khi tiến hành kiểm thử, hệ thống AI cờ vua đạt được các kết quả sau: - AI có thể thực hiện các nước đi hợp lệ và tuân theo chiến lược cơ bản. - Thời gian xử lý trung bình cho mỗi nước đi ở độ sâu 3 là khoảng 1-2 giây. - AI có thể phản ứng chính xác trước các nước đi phổ biến của người chơi. - Không có lỗi nghiêm trọng xảy ra trong quá trình kiểm thử.

4.6 Kết luận chương 4

Kiểm thử cho thấy AI hoạt động ổn định, tính toán nhanh và có thể đối đầu với người chơi ở mức trung bình. Tuy nhiên, AI cần được cải thiện thêm trong chiến lược khai cuộc và tàn cuộc để mạnh hơn.