

BÁO CÁO PROJECT NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Đề tài: Chess bot



NHÓM 15

Mục lục

1	Giới thiệu	4
1.1	Danh sách thành viên và phân công công việc	4
1.2	Lý do chọn đề tài	4
1.3	Mục tiêu nghiên cứu	5
1.4	Cấu trúc bài báo cáo	6
2	Cơ sở lý thuyết và nghiên cứu liên quan	7
2.1	Các thuật toán liên quan đến cờ vua	7
2.1.1	Sử dụng Bit board để biểu diễn bàn cờ và nước đi	7
2.1.2	Thuật toán Minimax	7
2.1.3	Alpha-Beta Pruning	8
2.1.4	Quisience Search	8
2.1.5	Sắp xếp nước đi (Move Ordering)	8
2.1.6	Bảng tra cứu giá trị đánh giá (Transposition Table)	9
2.2	Các nghiên cứu về hệ thống chess bot hiện tại	9
2.2.1	Stockfish	9
2.2.2	AlphaZero	10
2.3	Biểu diễn trạng thái bàn cờ bằng FEN	11
2.3.1	Cấu trúc của FEN	11
2.3.2	Ví dụ về FEN	12
3	Phương pháp nghiên cứu	13
3.1	Biểu diễn bàn cờ	13
3.1.1	Sử dụng bit board để biểu diễn bàn cờ	13
3.1.2	Biểu diễn loại quân cờ	14
3.1.3	Sử dụng Zobrist Hash để tạo khoá băm cho trạng thái bàn cờ	15
3.2	Sinh nước đi hợp lệ	17
3.2.1	Biểu diễn nước cờ	18
3.2.2	Bit board nước đi của từng loại quân	18
3.2.3	Tạo nước đi của vua	19
3.2.4	Xử lí các quân cờ bị ghim	19
3.2.5	Tạo nước đi của quân trượt	22
3.2.6	Tạo nước đi của tốt	22
3.3	Tìm kiếm nước đi	23
3.3.1	Tìm kiếm nước đi trong Sách khai cuộc	23
3.3.2	Khởi tạo tìm kiếm	25
3.3.3	Bắt đầu tìm kiếm sâu dần (Iterative Deepening Search)	25
3.3.4	Minimax	25
3.3.5	Alpha-Beta Pruning	26

3.3.6	Sắp xếp nước đi (Move Ordering) -----	26
3.3.7	Tìm kiếm yên tĩnh (Quiescence Search) -----	27
3.3.8	Lưu trữ kết quả bằng Transposition Table -----	28
3.3.9	Quản lý thời gian -----	29
3.3.10	Hoàn thành tìm kiếm -----	29
3.4	Đánh giá thế cờ -----	29
3.4.1	Giá trị quân số hiện tại (Material Score) -----	30
3.4.2	Chỉ số tàn cuộc -----	30
3.4.3	Bảng điểm vị trí (Piece-Square Tables) -----	30
3.4.4	Cấu trúc quân tốt (Pawn Structure) -----	32
3.4.5	Bảo vệ vua (King Safety) -----	33
3.4.6	Đẩy lùi và quây bắt vua đối phương -----	34
3.4.7	Tính toán giá trị đánh giá tổng -----	34
4	Kết quả đánh giá và thảo luận -----	35
4.1	Thiết lập giao thức UCI -----	35
4.1.1	Các bước triển khai giao thức UCI cho chess bot -----	35
4.2	Đánh giá với các AI chess trong PyChess -----	36
4.2.1	Tích hợp chess bot vào PyChess -----	36
4.2.2	Đánh giá trên PyChess -----	36
4.3	Đánh giá trên nền tảng Chess.com -----	37
4.4	Kết quả đánh giá -----	38
4.4.1	Kết quả trong PyChess -----	38
4.4.2	Kết quả trên Chess.com -----	38
4.5	Kết luận -----	38
5	Tài liệu tham khảo -----	39

1 Giới thiệu

1.1 Danh sách thành viên và phân công công việc

Họ và tên	MSSV	Công việc đảm nhận
Hoàng Nguyên Vũ	20215171	- Thu thập dữ liệu Sách khai cuộc. - Code phần Zobrist Hash, Transposition Table và phần Deepening Iterative Search. - Tổng hợp báo cáo.
Nguyễn Hữu Minh	20215091	- Thu thập các chiến thuật cờ vua để áp dụng vào chess bot. - Code phần Evaluation
Lê Quốc Việt	20215166	- Code phần sinh nước đi hợp lệ (MoveGenerator)
Hoàng Quốc Trung	20215154	- Code phần biểu diễn bàn cờ.
Ngọ Doãn Ngọc	20215103	- Code phần Minimax Search và Alpha-Beta Pruning. - Code phần Quisience Search.

1.2 Lý do chọn đề tài

Trong những năm gần đây, trí tuệ nhân tạo (AI) đã có những bước tiến vượt bậc và ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau. Một trong những ứng dụng nổi bật của AI là trong trò chơi cờ vua, nơi các chương trình máy tính từ lâu đã có khả năng chơi cờ ở cấp độ vượt trội so với con người. Sự phát triển của các chess bot như AlphaZero của Google DeepMind đã chứng minh tiềm năng to lớn của AI trong việc giải quyết các vấn đề phức tạp và đưa ra quyết định trong thời gian thực.

Chọn đề tài phát triển phần mềm chess bot sử dụng AI có nhiều lý do quan trọng như sau:

- Thách thức trí tuệ và kỹ thuật: Cờ vua là một trò chơi có tính chiến lược cao, đòi hỏi người chơi phải có khả năng suy luận logic, dự đoán và lập kế hoạch. Việc xây dựng một chess bot có khả năng thi đấu ở mức độ cao là một thách thức lớn về mặt trí tuệ và kỹ thuật, giúp người nghiên cứu nâng cao kiến thức và kỹ năng trong lĩnh vực AI và lập trình.
- Giá trị học thuật: Nghiên cứu và phát triển chess bot không chỉ có ý nghĩa thực tiễn mà còn đóng góp vào việc hiểu sâu hơn về các thuật toán AI, đặc biệt là các thuật toán tìm kiếm và đánh giá. Những kiến thức và kỹ thuật thu được từ việc phát triển chess bot có thể áp dụng vào nhiều lĩnh vực khác như tối ưu hóa, lập kế hoạch và ra quyết định.
- Ứng dụng thực tiễn: Chess bot có thể được sử dụng làm công cụ hỗ trợ học tập và giải trí. Người chơi cờ vua có thể sử dụng chess bot để luyện tập, phân

tích các ván đấu và cải thiện kỹ năng. Đồng thời, các giải đấu cờ vua trực tuyến có thể sử dụng chess bot để giám sát và đảm bảo tính công bằng.

- Phát triển nghề nghiệp: Việc tham gia vào dự án phát triển phần mềm chess bot sử dụng AI giúp nhóm nghiên cứu có cơ hội phát triển nghề nghiệp trong lĩnh vực công nghệ thông tin, đặc biệt là các lĩnh vực liên quan đến AI và phát triển phần mềm.

Với những lý do trên, đề tài phát triển phần mềm chess bot sử dụng AI không chỉ mang lại nhiều giá trị về mặt học thuật và thực tiễn mà còn là một cơ hội để khám phá và ứng dụng các công nghệ tiên tiến trong lĩnh vực trí tuệ nhân tạo.

1.3 Mục tiêu nghiên cứu

Mục tiêu chính của nghiên cứu này là phát triển một phần mềm chess bot sử dụng trí tuệ nhân tạo, có khả năng chơi cờ vua ở mức độ cao và đưa ra các nước đi hợp lý trong thời gian ngắn. Để đạt được mục tiêu này, nghiên cứu sẽ tập trung vào các khía cạnh cụ thể sau:

- Xây dựng mô hình biểu diễn bàn cờ và nước đi:
 - Sử dụng bit board để biểu diễn bàn cờ và các quân cờ một cách hiệu quả.
 - Phát triển các cấu trúc dữ liệu và thuật toán để biểu diễn và xử lý nước đi.
- Phát triển thuật toán tạo nước đi hợp lệ:
 - Thiết kế và triển khai thuật toán MoveGenerator để tạo ra tất cả các nước đi hợp lệ dựa trên quy tắc cờ vua.
 - Tối ưu hóa quá trình tạo nước đi để đảm bảo hiệu năng của chess bot.
- Triển khai thuật toán tìm kiếm nước đi tốt nhất:
 - Sử dụng thuật toán Minimax kết hợp với alpha-beta pruning để xây dựng cây tìm kiếm và tìm ra nước đi tốt nhất.
 - Áp dụng phương pháp tìm kiếm sâu dần để tối ưu thời gian và hiệu suất tìm kiếm.
- Phát triển hàm đánh giá bàn cờ (Evaluator):
 - Xây dựng hàm đánh giá để phân tích và cho điểm các thế cờ dựa trên các yếu tố chiến thuật trong cờ vua như số lượng và vị trí quân cờ, độ an toàn của vua, và cấu trúc quân tốt.
 - Kết hợp các yếu tố này để tạo ra một hàm đánh giá chính xác và hiệu quả.
- Đánh giá và cải tiến:
 - Phân tích kết quả thử nghiệm để tìm ra các điểm mạnh và điểm yếu của chess bot.
 - Đề xuất và thực hiện các cải tiến để nâng cao hiệu năng và khả năng chơi cờ của chess bot.

1.4 Cấu trúc bài báo cáo

Báo cáo này được cấu trúc thành các phần chính sau để trình bày một cách có hệ thống và logic quá trình nghiên cứu và phát triển phần mềm chess bot sử dụng trí tuệ nhân tạo:

1. Giới thiệu

Phần này giới thiệu tổng quan về đề tài, lý do chọn đề tài và mục tiêu nghiên cứu. Ngoài ra, phần này cũng nêu rõ cấu trúc của toàn bộ báo cáo.

2. Cơ sở lý thuyết và nghiên cứu liên quan

Trình bày các khái niệm cơ bản và lý thuyết liên quan đến cờ vua và trí tuệ nhân tạo.

Tóm tắt các nghiên cứu và hệ thống chess bot đã có, cùng những bài học và kỹ thuật có thể áp dụng vào dự án.

3. Phương pháp nghiên cứu

Mô tả chi tiết các phương pháp và kỹ thuật được sử dụng trong nghiên cứu và phát triển phần mềm chess bot.

Bao gồm việc biểu diễn bàn cờ và nước đi, tạo nước đi hợp lệ, thuật toán tìm kiếm nước đi tốt nhất, và hàm đánh giá bàn cờ.

4. Kết quả và thảo luận

Trình bày kết quả thử nghiệm và đánh giá hiệu năng của chess bot.

Thảo luận về các kết quả thu được, phân tích các điểm mạnh và điểm yếu của hệ thống, so sánh với các hệ thống khác.

5. Tài liệu tham khảo

Liệt kê các tài liệu, sách, bài báo và nguồn tham khảo đã sử dụng trong quá trình nghiên cứu và viết báo cáo.

2 Cơ sở lý thuyết và nghiên cứu liên quan

2.1 Các thuật toán liên quan đến cờ vua

2.1.1 Sử dụng Bit board để biểu diễn bàn cờ và nước đi

Bit board là một phương pháp biểu diễn bàn cờ và nước đi trong cờ vua sử dụng các giá trị nhị phân. Phương pháp này cho phép xử lý các thao tác trên bàn cờ một cách hiệu quả và nhanh chóng, nhờ vào các phép toán nhị phân.

- Nguyên tắc hoạt động:
 - Bàn cờ 64 ô được biểu diễn bằng một giá trị 64 bit (ulong), trong đó mỗi bit đại diện cho một ô trên bàn cờ. Giá trị của bit cho biết ô đó có chứa quân cờ hay không.
 - Mỗi loại quân cờ và màu cờ được biểu diễn bằng một giá trị bit khác nhau. Cụ thể, có 12 loại quân cờ (bao gồm cả hai màu trắng và đen), do đó cần 12 giá trị 64 bit để biểu diễn vị trí của tất cả các quân cờ trên bàn cờ.
 - Nước đi được biểu diễn bằng một giá trị 16 bit, trong đó các bit chứa thông tin về ô bắt đầu, ô mục tiêu và các loại nước đi đặc biệt (ví dụ: phong hậu, nhập thành).
- Lợi ích:
 - Sử dụng bitboard giúp tăng tốc độ xử lý các thao tác trên bàn cờ, như kiểm tra các nước đi hợp lệ và thực hiện nước đi.
 - Các phép toán nhị phân giúp giảm thiểu tài nguyên tính toán, do đó nâng cao hiệu suất tổng thể của chương trình.

2.1.2 Thuật toán Minimax

Thuật toán Minimax là nền tảng của nhiều chương trình chơi cờ vua. Thuật toán này hoạt động dựa trên nguyên lý đối kháng giữa hai người chơi, thường được gọi là người chơi "maximizing" và người chơi "minimizing". Mục tiêu của người chơi maximizing là tối đa hóa lợi thế của mình, trong khi mục tiêu của người chơi minimizing là tối thiểu hóa lợi thế của đối thủ.

- Nguyên tắc hoạt động:
 - Xây dựng cây tìm kiếm, trong đó mỗi nút đại diện cho một trạng thái của bàn cờ.
 - Các nhánh của cây đại diện cho các nước đi có thể thực hiện từ trạng thái hiện tại.
 - Tại mỗi cấp độ của cây, người chơi hiện tại sẽ chọn nước đi tối ưu nhất cho mình (maximizing chọn nước đi có giá trị cao nhất, minimizing chọn nước đi có giá trị thấp nhất).
- Ứng dụng:

- Minimax được sử dụng để quyết định nước đi tiếp theo dựa trên việc đánh giá các trạng thái của bàn cờ.

2.1.3 Alpha-Beta Pruning

Alpha-Beta Pruning là một cải tiến của thuật toán Minimax, giúp giảm số lượng nút cần được đánh giá trong cây tìm kiếm, từ đó tăng hiệu quả tính toán.

- Nguyên tắc hoạt động:
 - Sử dụng hai giá trị alpha và beta để theo dõi các giới hạn của điểm số mà người chơi maximizing và minimizing có thể đạt được.
 - Khi tìm kiếm, nếu phát hiện một nhánh mà điểm số không thể tốt hơn các giới hạn hiện tại, nhánh đó sẽ bị cắt bỏ.
- Lợi ích:
 - Giảm đáng kể số lượng trạng thái cần được đánh giá, từ đó tăng tốc độ tìm kiếm và ra quyết định của chess bot.

2.1.4 Quiscience Search

Quiscience Search là một kỹ thuật được sử dụng để giải quyết vấn đề "horizon effect" trong các thuật toán tìm kiếm cây, đặc biệt là trong cờ vua. Horizon effect xảy ra khi thuật toán không thể nhìn xa hơn một số bước nhất định, dẫn đến việc đánh giá không chính xác trạng thái của bàn cờ.

- Nguyên tắc hoạt động:
 - Khi thuật toán Minimax hoặc Alpha-Beta Pruning đạt đến độ sâu tối đa, Quiscience Search sẽ tiếp tục tìm kiếm thêm các nước đi "yên lặng" (quiescent moves) như bắt quân hoặc phòng thủ để đảm bảo rằng không bỏ sót những nước đi quan trọng có thể thay đổi đáng kể tình huống bàn cờ.
- Lợi ích:
 - Giảm thiểu rủi ro của việc đánh giá sai lầm do horizon effect.
 - Cung cấp đánh giá chính xác hơn về trạng thái cuối cùng của các nhánh tìm kiếm.

2.1.5 Sắp xếp nước đi (Move Ordering)

Sắp xếp nước đi là một kỹ thuật quan trọng trong việc tối ưu hóa quá trình tìm kiếm trong cây Minimax có áp dụng Alpha-Beta Pruning. Việc sắp xếp các nước đi theo thứ tự hợp lý có thể giúp tăng hiệu quả của thuật toán bằng cách giảm số lượng các nhánh cần kiểm tra.

- Nguyên tắc hoạt động:
 - Các nước đi có tiềm năng cao (chẳng hạn như các nước bắt quân hoặc phong quân) được đặt lên trước trong quá trình tìm kiếm.
 - Việc sắp xếp này giúp tăng khả năng cắt bỏ sớm các nhánh không cần thiết trong Alpha-Beta Pruning, từ đó tiết kiệm tài nguyên tính toán.

- Lợi ích:
 - Tăng tốc độ tìm kiếm và ra quyết định của chess bot.
 - Cải thiện hiệu quả của thuật toán Alpha-Beta Pruning, giúp chương trình đánh giá được nhiều trạng thái hơn trong cùng một khoảng thời gian.

2.1.6 Bảng tra cứu giá trị đánh giá (Transposition Table)

Transposition Table là một kỹ thuật quan trọng giúp tăng hiệu quả của các thuật toán tìm kiếm bằng cách lưu trữ các kết quả đánh giá trạng thái đã được tính toán trước đó.

- Nguyên tắc hoạt động:
 - Lưu trữ các trạng thái của bàn cờ và các giá trị đánh giá tương ứng khi chúng được tính toán trong quá trình tìm kiếm.
 - Khi gặp lại một trạng thái đã được đánh giá trước đó, thuật toán có thể sử dụng ngay giá trị đã lưu mà không cần phải tính toán lại, từ đó tiết kiệm thời gian và tài nguyên.
- Lợi ích:
 - Giảm thiểu khối lượng công việc tính toán lặp lại.
 - Tăng tốc độ tìm kiếm và ra quyết định của chess bot.
 - Tối ưu hóa việc sử dụng bộ nhớ và tài nguyên xử lý.

Các thuật toán và mô hình trên đã đóng góp quan trọng vào sự phát triển của các chương trình chơi cờ vua hiện đại, giúp chúng đạt được hiệu suất và khả năng chơi vượt trội. Trong phần tiếp theo, bài báo cáo sẽ trình bày cách mà các thuật toán và kĩ thuật này được áp dụng trong dự án chess bot.

2.2 Các nghiên cứu về hệ thống chess bot hiện tại

Trong những năm gần đây, nhiều nghiên cứu và phát triển đã được thực hiện để tạo ra các hệ thống chess bot mạnh mẽ, có khả năng thi đấu ở cấp độ cao nhất. Trong số các hệ thống chess bot hiện tại, hai chương trình nổi bật nhất là Stockfish và AlphaZero. Cả hai đều có những đặc điểm và kỹ thuật độc đáo, đóng góp quan trọng vào sự phát triển của lĩnh vực này.

2.2.1 Stockfish

Stockfish là một trong những chương trình chơi cờ vua mã nguồn mở mạnh nhất hiện nay, được phát triển bởi một cộng đồng lập trình viên và người chơi cờ vua toàn cầu. Stockfish liên tục đứng đầu trong các bảng xếp hạng cờ vua dành cho máy tính.

- Kỹ thuật và thuật toán:
 - **Alpha-Beta Pruning:** Stockfish sử dụng thuật toán tìm kiếm Minimax kết hợp với Alpha-Beta Pruning để cắt giảm số lượng nhánh cần khám phá trong cây tìm kiếm, từ đó tăng tốc độ tìm kiếm và ra quyết định.

- **Quiscience Search:** Để tránh "horizon effect", Stockfish sử dụng Quiscience Search để mở rộng thêm các nhánh tìm kiếm khi gặp phải các nước đi phức tạp như bắt quân hoặc phòng thủ.
- **Sắp xếp nước đi:** Stockfish sử dụng các kỹ thuật sắp xếp nước đi để đưa các nước đi quan trọng lên đầu, giúp tối ưu hóa quá trình Alpha-Beta Pruning.
- **Evaluation Function:** Hàm đánh giá của Stockfish rất phức tạp được tạo nên từ chiến thuật của các kiện tướng cờ vua hàng đầu thế giới trong lịch sử, bao gồm nhiều yếu tố như vị trí quân, cấu trúc quân tốt, độ an toàn của vua, và các yếu tố chiến thuật khác. Hàm đánh giá này giúp Stockfish đưa ra quyết định tốt nhất dựa trên trạng thái hiện tại của bàn cờ.
- Hiệu suất và thành tích:
 - Stockfish liên tục chiến thắng trong các giải đấu cờ vua dành cho máy tính và được sử dụng rộng rãi bởi các kỳ thủ và người đam mê cờ vua để phân tích ván cờ và luyện tập.

Các chiến thuật và thuật toán của Stockfish về cơ bản được phát triển dựa trên các chiến thuật và kinh nghiệm chơi cờ của các người chơi trong suốt lịch sử, điều này khiến nó là một mô hình hữu dụng trong việc đánh giá và phân tích các nước cờ của người chơi.

2.2.2 AlphaZero

AlphaZero là một chương trình chơi cờ vua do Google DeepMind phát triển, nổi bật với việc sử dụng học sâu và học tăng cường để đạt được hiệu suất vượt trội.

Không giống như Stockfish, AlphaZero học chơi cờ vua từ con số 0, chỉ với bộ luật cờ vua và bằng cách chơi hàng triệu ván cờ với chính mình.

- Kỹ thuật và thuật toán:
 - **Mạng nơ-ron sâu (Deep Neural Networks):** AlphaZero sử dụng mạng nơ-ron sâu để đánh giá trạng thái của bàn cờ và quyết định nước đi tiếp theo. Mạng nơ-ron này được huấn luyện thông qua hàng triệu ván cờ tự chơi để học các chiến lược và nước đi hiệu quả.
 - **Học tăng cường (Reinforcement Learning):** AlphaZero áp dụng kỹ thuật học tăng cường để cải thiện khả năng chơi cờ qua mỗi ván đấu. Bằng cách chơi với chính mình, AlphaZero liên tục cải thiện và tối ưu hóa các chiến lược của mình.
 - **Monte Carlo Tree Search (MCTS):** AlphaZero kết hợp mạng nơ-ron sâu với Monte Carlo Tree Search để xây dựng cây tìm kiếm và đánh giá các nước đi tiềm năng một cách hiệu quả. MCTS giúp AlphaZero tìm ra các nước đi tối ưu bằng cách mô phỏng nhiều kịch bản khác nhau.
- Hiệu suất và thành tích:
 - AlphaZero đã tạo ra cơn sốt khi đánh bại Stockfish một cách áp đảo với kết quả thắng 155 ván, thua 6 ván và hoà 839 ván trong 1000 ván đấu

không giới hạn thời gian. Thành tích này chứng minh khả năng vượt trội của AlphaZero trong việc học và áp dụng các chiến lược chơi cờ phức tạp.

- Các đại kiện tướng cờ vua hàng đầu thế giới nhận xét là các nước đi mà AlphaZero đưa ra tựa như những nước đi của chúa, không một con người nào có thể hiểu được.
- AlphaZero không chỉ chơi cờ vua, mà còn thành công trong các trò chơi chiến lược khác như cờ vây và shogi, chứng minh tính linh hoạt và hiệu quả của phương pháp học tăng cường và mạng nơ-ron sâu.

Cả Stockfish và AlphaZero đều đại diện cho những đỉnh cao của trí tuệ nhân tạo trong lĩnh vực cờ vua, mỗi chương trình với những đặc điểm và kỹ thuật riêng biệt. Stockfish dựa vào các thuật toán truyền thống và tối ưu hóa hiệu suất, trong khi AlphaZero sử dụng học sâu và học tăng cường để tự cải thiện và phát triển các chiến lược chơi cờ. Những nghiên cứu và phát triển này không chỉ nâng cao khả năng của các chương trình chơi cờ mà còn mở ra nhiều hướng đi mới cho ứng dụng AI trong các lĩnh vực khác.

2.3 Biểu diễn trạng thái bàn cờ bằng FEN

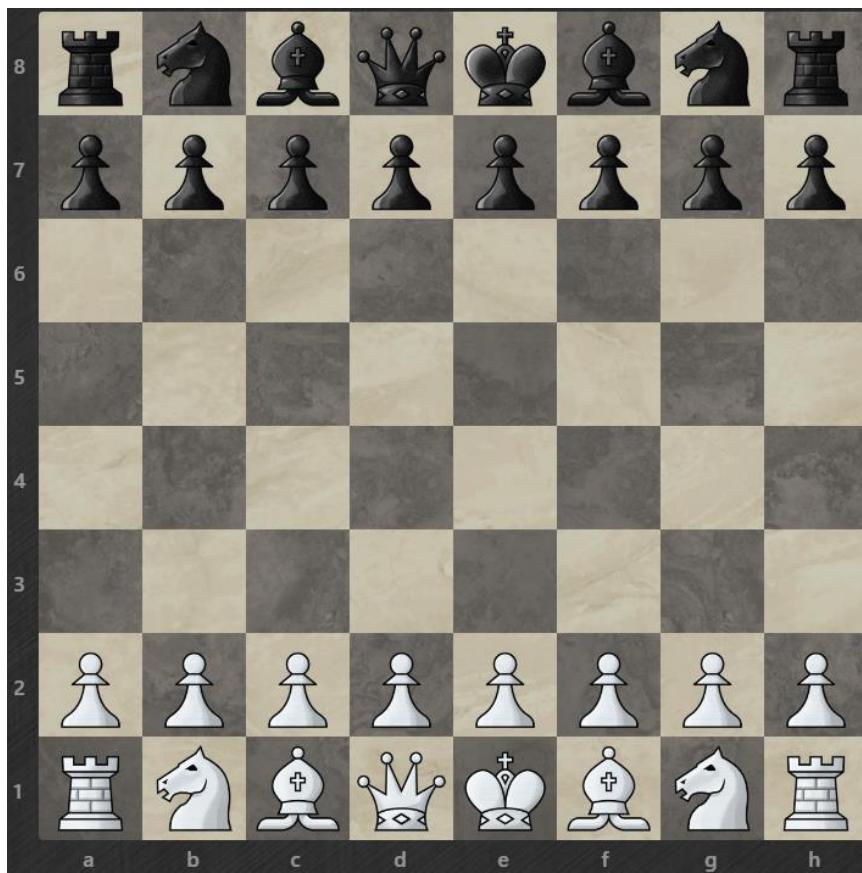
Forsyth-Edwards Notation (FEN) là một hệ thống ký hiệu tiêu chuẩn được sử dụng để mô tả trạng thái của bàn cờ trong cờ vua được sử dụng rộng rãi trên thế giới để biểu diễn bàn cờ đặc biệt là trong các phần mềm cờ vua. FEN cung cấp một cách chính xác và ngắn gọn để lưu trữ và trao đổi thông tin về vị trí của các quân cờ, lượt đi, quyền nhập thành, và các chi tiết khác liên quan đến trạng thái hiện tại của trò chơi.

2.3.1 Cấu trúc của FEN

FEN bao gồm sáu thành phần được ngăn cách bởi dấu cách, mỗi thành phần mô tả một khía cạnh cụ thể của trạng thái bàn cờ. Cấu trúc của FEN như sau:

- **Vị trí các quân cờ:** Mô tả vị trí của các quân cờ trên bàn cờ từ hàng 8 đến hàng 1, sử dụng các ký hiệu chữ cái cho từng quân cờ (ví dụ: 'r' cho quân xe đen, 'P' cho quân tốt trắng) và các số để chỉ các ô trống liên tiếp.
- **Lượt đi:** 'w' cho lượt đi của quân trắng và 'b' cho lượt đi của quân đen.
- **Quyền nhập thành:** Mô tả quyền nhập thành cho cả hai bên (ví dụ: 'K' cho quyền nhập thành cánh vua của trắng, 'q' cho quyền nhập thành cánh hậu của đen). Nếu không có quyền nhập thành, ký hiệu '-'.
- **En passant:** Vị trí ô có thể bắt en passant, hoặc '-' nếu không có.
- **Nước đi không bắt quân hoặc di chuyển quân tốt gần nhất:** Số nguyên chỉ số nước đi không bắt quân hoặc di chuyển quân tốt gần nhất, để tính quy tắc 50 nước.
- **Tổng số nước đi:** Tổng số nước đi kể từ đầu trận, tăng lên sau mỗi lượt của đen.

2.3.2 Ví dụ về FEN



Trạng thái khởi đầu của bàn cờ sẽ được biểu diễn một cách đầy đủ bằng chuỗi FEN dưới đây:

`rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1`

Trong đó:

- '`rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR`': Vị trí các quân cờ từ hàng 8 đến hàng 1.
- '`w`': Lượt đi hiện tại là của trắng.
- '`KQkq`': Cả hai bên đều còn quyền nhập thành cả hai cánh.
- '`-`': Không có nước đi en passant.
- '`0`': Chưa có nước đi không bắt quân hoặc di chuyển quân tốt.
- '`1`': Nước đi hiện tại là nước đi đầu tiên của ván cờ.

FEN được sử dụng rộng rãi trong các chương trình cờ vua và phát triển chess bot để lưu trữ và trao đổi thông tin về trạng thái bàn cờ giúp tiết kiệm không gian lưu trữ và tối ưu hóa tìm kiếm và đánh giá.

Trong chương trình chess bot được phát triển, FEN được dùng để biểu diễn các thế cờ trong Sách các thế khai cuộc (Opening Book) từ đó chương trình có thể tìm xem trạng thái bàn cờ hiện tại có trùng với thế cờ nào trong sách không, nếu có nó sẽ sử dụng nước đi theo trong sách.

3 Phương pháp nghiên cứu

3.1 Biểu diễn bàn cờ

Một trong những khía cạnh quan trọng nhất của việc phát triển phần mềm chess bot là cách biểu diễn bàn cờ và nước đi. Sử dụng các phương pháp biểu diễn hiệu quả không chỉ giúp tăng tốc độ xử lý mà còn giúp tối ưu hóa việc thực hiện các thuật toán tìm kiếm và đánh giá. Trong dự án này, phần mềm sẽ sử dụng bit board và các giá trị nhị phân để biểu diễn bàn cờ và nước đi.

3.1.1 Sử dụng bit board để biểu diễn bàn cờ

[Bitboards \(The Concept of\) - Advanced Java Chess Engine Tutorial 1 - YouTube](#)

Bit board là một phương pháp biểu diễn bàn cờ sử dụng các giá trị 64 bit để đại diện cho trạng thái của bàn cờ. Phương pháp này cho phép thực hiện các thao tác trên bàn cờ một cách nhanh chóng và hiệu quả nhờ vào các phép toán nhị phân.



- Cách biểu diễn bàn cờ bằng giá trị 64 bit:
 - Bàn cờ cờ vua có 64 ô, mỗi ô được đại diện bởi một bit trong giá trị 64 bit (ulong). Mỗi bit cho biết ô đó có chứa quân cờ hay không.

- Ô đầu tiên (a1) tương ứng với bit 0 và ô cuối cùng (h8) tương ứng với bit 63. Giá trị 1 tại một vị trí bit cho biết ô đó có quân cờ, ngược lại giá trị 0 cho biết ô trống.
- Biểu diễn vị trí của các quân cờ và các bitboard khác:
 - Có 12 loại quân cờ trên bàn cờ (bao gồm cả hai màu trắng và đen) bao gồm tốt trắng, tốt đen, xe trắng, xe đen, vua trắng, vua đen,... do đó cần 12 giá trị 64 bit để biểu diễn vị trí của tất cả các quân cờ tạo nên một bàn cờ.
 - Các bit board khác cũng được sử dụng để thuận tiện trong quá trình tính toán và sinh nước đi như bit board các ô trống, các ô đang bị tấn công, bit board cho tất cả các quân của một bên,...
- Ví dụ:
 - Bit board cho quân trắng (Pawn | White) được biểu diễn bằng cách set bit tại các vị trí tương ứng với vị trí của các quân tốt trắng trên bàn cờ.
 - Bit board cho các ô trống được biểu diễn bằng cách set bit tại các vị trí ô không có quân cờ.
 - Các bit board khác như bit board cho các ô bị tấn công, bit board cho các quân cờ của mỗi bên cũng được sử dụng để lưu trữ thông tin bổ sung.

3.1.2 Biểu diễn loại quân cờ

Quân cờ trên bàn cờ được biểu diễn bằng một giá trị 3 bit, và thêm 1 bit nữa để biểu diễn màu của quân cờ, tạo thành tổng cộng 4 bit. Từ đó ta biểu diễn được giá trị của một loại quân (bao gồm quân và màu).

Giá trị 3 bit cho quân và 1 bit cho màu:

- Các quân cờ được biểu diễn bằng các giá trị như sau:
 - None = **0000** = 0
 - Pawn = **0001** = 1
 - Knight = **0010** = 2
 - Bishop = **0011** = 3
 - Rook = **0100** = 4
 - Queen = **0101** = 5
 - King = **0111** = 6
- Hai màu quân được biểu diễn bằng các giá trị như sau:
 - White = **0000** = 0
 - Black = **1000** = 8
- Kết hợp một giá trị quân với một giá trị màu bằng phép toán OR sẽ cho ta giá trị của một loại quân cờ cụ thể.
- Ví dụ về biểu diễn quân cờ:
 - WhitePawn = Pawn | White = **0001** = 1
 - WhiteKnight = Knight | White = **0010** = 2
 - BlackPawn = Pawn | Black = **1001** = 9

- BlackKing = King | Black = **1110** = 14

Tổng hợp một trạng thái bàn cờ cụ thể sẽ được biểu diễn bằng một mảng 15 phần tử là các bit board 64 bit với chỉ số của từng phần tử là giá trị của từng loại quân cờ trong 12 loại quân tương ứng (cộng thêm 3 bit board trống None).

Ví dụ, biểu diễn bàn cờ khởi đầu:

Giải thích:

- Mỗi hàng trên là một giá trị bit board tương ứng với một loại quân cờ cụ thể, ví dụ hàng thứ 2 là biểu diễn cho quân tốt trắng, các bit 1 thể hiện vị trí có quân cờ, các bit 0 thể hiện vị trí không có quân cờ. Có 15 hàng tất cả để biểu diễn cho tất cả các quân trên bàn cờ.
 - Có thể hiểu mỗi 8 cột trong ma trận trên sẽ biểu diễn cho một rank (một hàng ngang) các ô trên bàn cờ, ví dụ rank 2 chỉ có các bit của WhitePawn là 1 nên ta hiểu hàng thứ 2 trên bàn cờ này là một hàng tốt trắng.

Từ các biểu diễn trên, ta có thể tạo các logic để biểu diễn bàn cờ, sinh các nước đi hợp lệ trên bàn cờ, thực hiện và hoàn tác nước đi một cách hiệu quả. Các chức năng này là cần thiết cho việc phát triển chess bot và đảm bảo hiệu suất cao trong quá trình tính toán và ra quyết định.

3.1.3 Sử dụng Zobrist Hash để tạo khóa băm cho trạng thái bàn cờ

Trong các chương trình chơi cờ vua, việc đánh giá và tìm kiếm các nước đi tối ưu một cách nhanh chóng và hiệu quả là rất quan trọng. Một trong những kỹ thuật quan trọng được sử dụng để tối ưu hóa quá trình này là Zobrist hashing. Kỹ thuật này giúp băm trạng thái của bàn cờ thành một giá trị duy nhất (khóa Zobrist) để dễ dàng kiểm tra các trạng thái đã gặp phải trước đó, từ đó tránh việc tính toán lại và cải thiện hiệu suất.

Zobrist hashing là một kỹ thuật băm được sử dụng để mã hóa trạng thái của một bàn cờ thành một giá trị băm 64 bit duy nhất (gọi là khóa Zobrist). Kỹ thuật này rất hiệu quả trong việc xác định nhanh chóng các trạng thái lặp lại của bàn cờ, điều quan trọng trong các trò chơi như cờ vua để kiểm tra các quy tắc như hòa do lặp lại ba lần (three-fold repetition).

3.1.3.1 Cách thức hoạt động của Zobrist hashing

1. Khởi tạo bảng băm

- Để mã hoá một trang thái bàn cờ một cách đầy đủ, ta cần xét các giá trị sau:

- Mỗi ô trên bàn cờ có 64 vị trí (8x8), và mỗi vị trí có thể chứa một trong 12 loại quân cờ (6 loại quân cờ cho mỗi màu). Ta cần một ma trận 64x15 với mỗi phần tử là một giá trị 64 bit để biểu lưu trữ các giá trị băm cho tất cả các cặp quân cờ - vị trí.
- Quyền nhập thành (castling rights): mỗi bên có 4 giá trị khả thi của quyền nhập thành (không, chỉ nhập thành cánh hậu, chỉ nhập thành cánh vua và được nhập thành cả 2 cánh). Vậy ta cần $4 * 4 = 16$ giá trị 64 bit để biểu diễn các trạng thái khả thi của quyền nhập thành cho một trạng thái bàn cờ.
- Quyền bắt tốt qua đường (en passant): cần 9 giá trị 64 bit biểu thị cho cột được phép bắt en passant (giá trị 0 là không được phép)
- Lượt đi của bên nào: cần 1 giá trị 64 bit để biểu thị.
- Trong hàm khởi tạo, các giá trị ngẫu nhiên 64 bit được sinh ra và gán vào các bảng băm ở trên bằng cách sử dụng một giá trị seed cố định để đảm bảo tính đồng nhất qua các lần chạy chương trình.

2. Tính toán khoá Zobrist

- Với một trạng thái bàn cờ cụ thể (có thể được cung cấp sử dụng một chuỗi FEN) thì hàm tính toán khoá Zobrist sẽ lặp qua tất cả các ô trên bàn cờ, XOR giá trị băm của mỗi quân cờ tại vị trí tương ứng, sau đó XOR thêm giá trị băm của trạng thái bắt tốt qua đường, trạng thái lượt đi và trạng thái nhập thành để tạo thành một khoá Zobrist cho thế cờ đó.
- Sau mỗi nước đi được thực hiện bởi người chơi hoặc bot (kể cả các nước đi giả lập trong quá trình chess bot tìm kiếm nước đi) thì giá trị khoá Zobrist sẽ được tính toán lại dựa theo nước đi đó bằng cách XOR giá trị băm của quân cờ và vị trí cũ với giá trị băm của quân cờ và vị trí mới. Các trạng thái đặc biệt như bắt quân, nhập thành, bắt tốt qua đường cũng được xử lý tương tự.

3.1.3.2 Ứng dụng của khoá Zobrist trong chess bot

1. Xác định hòa do lặp lại ba nước (three-fold repetition)

Trong cờ vua, một ván cờ có thể kết thúc hòa nếu cùng một trạng thái bàn cờ xuất hiện ba lần với cùng các điều kiện. Để xác định điều này, chương trình sử dụng Zobrist key như sau:

- **Lưu trữ trạng thái:** Mỗi khi một nước đi được thực hiện, khóa Zobrist của trạng thái hiện tại của bàn cờ được lưu trữ vào một danh sách lịch sử (history stack). Điều này cho phép chương trình theo dõi tất cả các trạng thái đã gặp trong quá trình chơi.
- **Kiểm tra lặp lại:** Khi cần kiểm tra điều kiện hòa do lặp lại ba lần, chương trình sẽ duyệt qua danh sách lịch sử và đếm số lần khóa Zobrist xuất hiện. Nếu khóa Zobrist xuất hiện ít nhất ba lần, ván cờ được xác định là hòa.

2. Sử dụng Transposition Table (TT)

Transposition Table là một bảng băm dùng để lưu trữ kết quả đánh giá của các trạng thái bàn cờ đã gặp phải trong quá trình tìm kiếm nước đi. Zobrist key được sử dụng như là khoá của mỗi trạng thái bàn cờ được lưu trong TT. Sử dụng TT bao gồm:

- **Tra cứu nhanh:** Mỗi khi một trạng thái mới được đánh giá, chương trình sẽ tính toán khóa Zobrist và sử dụng nó để tra cứu trong TT. Nếu trạng thái này đã được lưu trữ trước đó, kết quả đánh giá có thể được truy xuất ngay lập tức mà không cần phải tính toán lại, giúp tiết kiệm thời gian và tài nguyên.
- **Lưu trữ kết quả:** Khi một trạng thái mới được đánh giá, kết quả (điểm số đánh giá, nước đi tốt nhất, độ sâu tìm kiếm, v.v.) sẽ được lưu trữ trong TT với khóa Zobrist làm khoá trong bảng. Điều này giúp tối ưu hóa các lần tra cứu sau này.
- **Giảm thiểu tính toán lặp lại:** Bằng cách sử dụng TT và khóa Zobrist, chương trình tránh được việc đánh giá lại các trạng thái đã gặp phải trước đó trong quá trình tìm kiếm, từ đó cải thiện hiệu suất tổng thể của chương trình.

3.1.3.3 Nguy cơ về độ của khoá Zobrist

Một không gian khoá 64 bit với tổng 2^{64} giá trị khả thi là không đủ để đảm bảo rằng mỗi trạng thái của bàn cờ sẽ có một khóa băm duy nhất, bởi vì số lượng các trạng thái khả thi của bàn cờ vua là một con số khổng lồ ước tính lên tới 10^{50} . Do đó, khả năng xảy ra đụng độ giữa các khoá Zobrist (khi một khoá được ánh xạ cho nhiều hơn một trạng thái bàn cờ) sẽ tăng lên khi số lượng các trạng thái bàn cờ đã được kiểm tra tăng lên.

Tuy nhiên, xác suất xảy ra xung đột trong không gian khoá 64 bit là rất nhỏ và thường là chấp nhận được, điều này đã được các chess engine nổi tiếng trong thực tế kiểm chứng.

Theo tính toán thì với không gian khoá 64 bit thì chỉ khi số trạng thái bàn cờ đã xét vượt qua khoảng 4 tỷ trạng thái thì mới ta mới có khả năng gặp một trạng thái đụng độ.ⁱ

Do đó, chúng ta có thể chấp nhận một rủi ro nhỏ để sử dụng khoá Zobrist bởi vì tác dụng của nó là rất lớn để tăng cường hiệu suất của chess bot.

3.2 Sinh nước đi hợp lệ

Lớp MoveGenerator trong chương trình cờ vua đóng vai trò quan trọng trong việc tạo ra các nước đi hợp lệ cho các quân cờ trên bàn cờ. Phần sinh nước đi hợp lệ không chỉ đơn thuần là tạo ra các nước đi có thể di chuyển, mà còn phải tuân thủ các quy tắc của cờ vua như tránh chiếu tướng, kiểm tra chiếu tướng, và xử lý các nước đi đặc biệt như nhập thành, en passant, và phong quân.

3.2.1 Biểu diễn nước cờ

Nước đi trong cờ vua được biểu diễn bằng một giá trị 16 bit, giúp lưu trữ đầy đủ thông tin về nước đi một cách ngắn gọn và hiệu quả.

Format của nước cờ:

- Nước cờ 16 bit được biểu diễn dưới dạng: **ffff . tttttt . ssssss**
 - Bit 0-5 (**ssssss**): Chỉ số của ô bắt đầu (source square).
 - Bit 6-11 (**tttttt**): Chỉ số của ô mục tiêu (target square).
 - Bit 12-15 (**ffff**): Flag để biểu diễn các nước đi đặc biệt như phong hậu, nhập thành, bắt en passant.
- Ví dụ:
 - Một nước đi từ ô e2 đến e4 sẽ được biểu diễn bằng cách set các bit 0-5 bằng chỉ số của ô e2 (12 - **001100**) và các bit 6-11 cho chỉ số của ô e4 (28 - **011100**). Kết quả là: **000.011100.001100**
 - Nếu nước đi này là một nước đi đặc biệt (ví dụ: phong hậu), các bit 12-15 sẽ được set tương ứng.

3.2.2 Bit board nước đi của từng loại quân

Để tối ưu hóa quá trình sinh nước đi hợp lệ, chương trình sử dụng các bitboard nước đi được tính toán trước. Đây là các bảng dữ liệu lưu trữ các nước đi hợp lệ cho từng quân cờ từ mỗi vị trí có thể trên bàn cờ, giúp tăng tốc độ sinh nước đi và giảm thiểu các phép tính trong thời gian thực.

Chương trình tạo ra các bitboard nước đi được tính toán trước cho các loại quân cờ và các tình huống khác nhau như sau:

- OrthoAttacks: Tấn công theo các đường thẳng (orthogonal attacks) cho xe và hậu.
- DiagAttacks: Tấn công theo các đường chéo (diagonal attacks) cho tượng và hậu.
- KnightAttacks: Các nước đi của mã.
- KingMoves: Các nước đi của vua.
- WhitePawnAttacks: Các nước bắt quân của tốt trắng.
- BlackPawnAttacks: Các nước bắt quân của tốt đen.

Các bitboard này được tính toán trước cho mỗi ô trên bàn cờ thông qua các vòng lặp và kiểm tra điều kiện. Dưới đây là quy trình tổng quát:

- Xác định các hướng di chuyển:
 - Đối với các quân trượt như xe và tượng, xác định các hướng di chuyển theo các đường thẳng và đường chéo.
 - Đối với mã, xác định các bước nhảy theo hình chữ L.
 - Đối với vua và tốt, xác định các hướng di chuyển và tấn công.
- Tính toán các vị trí hợp lệ:

- Với mỗi ô trên bàn cờ, tính toán các vị trí hợp lệ mà quân cờ có thể di chuyển hoặc tấn công.
- Kiểm tra xem các vị trí này có nằm trong phạm vi bàn cờ hay không.
- Gán giá trị vào bitboard:
 - Sử dụng phép toán bitwise để gán giá trị vào các bitboard tương ứng cho từng loại quân cờ và từng vị trí trên bàn cờ.

3.2.3 Tạo nước đi của vua

3.2.3.1 Điều kiện để vua di chuyển

Vua chỉ có thể di chuyển đến các ô không bị tấn công bởi quân đối phương và không chứa quân của mình.

Để thực hiện được phần này, chương trình sử dụng bit board các ô bị tấn công để kiểm tra các ô mà vua được di đến.

3.2.3.2 Nước đi đặc biệt - Nhập thành

- **Điều kiện:** Không được chiếu tướng, không được di chuyển qua các ô bị tấn công, và ô đích đến không bị tấn công.
- **Kiểm tra:** Kiểm tra các quyền nhập thành của trạng thái hiện tại và các ô có bị tấn công hay không để xác định khả năng nhập thành.

3.2.4 Xử lý các quân cờ bị ghim

Trong cờ vua, một quân cờ bị ghim (pinned piece) là một quân cờ không thể di chuyển mà không để vua của mình bị chiếu. Có hai loại ghim chính:

- Ghim tuyệt đối (absolute pin):
 - Quân cờ bị ghim không thể di chuyển khỏi vị trí hiện tại vì việc di chuyển sẽ để vua bị chiếu ngay lập tức. Trong tình huống này, quân bị ghim hoàn toàn không thể di chuyển.
 - Ví dụ, trong thế cờ dưới đây, quân xe đen ở d7 đang bị quân tượng trắng ở h3 ghim, nếu quân xe di chuyển thì vua đen sẽ bị tượng trắng nhắm vào. Vì vậy, di chuyển xe đen ở thế cờ này được coi là bất hợp lệ.



- Ghim tướng đối (relative pin):
 - Quân cờ bị ghim có thể di chuyển nhưng chỉ theo một đường thẳng (hoặc chéo) nối từ vua đến quân tấn công. Trong trường hợp này, quân bị ghim vẫn có thể di chuyển nhưng phải duy trì việc chặn đường tấn công của quân đối phương lên vua.
 - Ví dụ dưới đây quân hậu đen ở c6 đang bị quân tượng trắng ở b5 ghim tướng đối, quân hậu đen có thể di chuyển trên đường chéo nối từ quân tượng trắng đến vua đen nhưng không thể thoát khỏi đó vì nó sẽ khiến vua đen bị nhầm đến.



- Để sinh các nước đi hợp lệ trong các tình huống đó, chương trình phải xác định và xử lý các quân bị ghim để đảm bảo rằng tất cả các nước đi được tạo ra đều hợp lệ. Dưới đây là cách chương trình thực hiện việc này:

1. Xác định các quân bị ghim

- Tính toán các đường tấn công (attack rays):
 - Chương trình sử dụng các bitboard để tính toán các đường tấn công từ quân đối phương đến vua. Mỗi quân trượt (rook, bishop, queen) có thể tấn công dọc theo các đường thẳng hoặc đường chéo.
- Kiểm tra các đường tấn công có quân chắn:
 - Chương trình duyệt qua các ô trên đường tấn công từ quân trượt đối phương đến vua để kiểm tra xem có quân cờ nào chắn giữa chúng không. Nếu có duy nhất một quân cờ chắn trên đường tấn công, quân cờ này có thể bị ghim.
- Gắn cờ ghim:
 - Chương trình sử dụng các bitmask để đánh dấu các quân cờ bị ghim và các ô nằm trên đường ghim.

2. Xử lý các quân bị ghim

- Hạn chế nước đi:

- Nếu một quân cờ bị ghim, chương trình sẽ giới hạn các nước đi của quân cờ này theo hướng ghim. Quân cờ chỉ có thể di chuyển dọc theo đường ghim để duy trì việc chặn đường tấn công lên vua.
- Loại bỏ các nước đi không hợp lệ:
 - Chương trình loại bỏ các nước đi của quân bị ghim nếu chúng không duy trì việc chặn đường tấn công. Điều này đảm bảo rằng vua sẽ không bị chiếu sau khi quân cờ bị ghim di chuyển.

3. Tích hợp với phần sinh nước đi

- Trong quá trình sinh nước đi:
 - Khi chương trình sinh các nước đi cho từng quân cờ, nó kiểm tra xem quân cờ đó có bị ghim hay không. Nếu có, chương trình áp dụng các hạn chế nước đi tương ứng.
 - Đối với các quân cờ trượt (rook, bishop, queen), chương trình kiểm tra các đường ghim trước khi sinh nước đi để loại bỏ các nước đi không hợp lệ.
 - Đối với các quân khác như mã (knight) và tốt (pawn), chương trình kiểm tra và loại bỏ các nước đi nếu quân cờ bị ghim không thể di chuyển theo hướng ghim.

3.2.5 Tạo nước đi của quân trượt

Các quân trượt là các quân có thể di chuyển đến một ô tùy ý trên một đường thẳng hoặc ngang (với quân xe) hoặc trên hai đường chéo (với quân tượng), hoặc là tất cả 8 hướng với quân hậu.

Các quân trượt được phép di chuyển đến ô nào không có quân cùng màu.

Nếu các quân trượt đang bị ghim trên đường trượt thì chúng chỉ được phép di chuyển trên đường đó.

3.2.6 Tạo nước đi của tốt

3.2.6.1 Di chuyển thường (push moves):

Điều kiện: Tốt có thể di chuyển một ô về phía trước nếu ô đó trống. Nếu tốt chưa di chuyển, nó có thể di chuyển hai ô nếu cả hai ô đều trống.

Kiểm tra ghim: Nếu tốt bị ghim, nó chỉ có thể di chuyển theo đường ghim.

Thực hiện: Tạo và sử dụng bit board lưu vị trí các quân tốt. Sử dụng các phép dịch để dịch 8 bit tùy theo hướng di chuyển của màu tốt.

3.2.6.2 Bắt quân (capture moves):

Điều kiện: Tốt có thể bắt quân chéo về phía trước nếu có quân đối phương ở đó.

Thực hiện: Tạo ra 2 bit board lưu các ô có thể bắt sang bên trái và phải bằng cách dịch 7 bit và 9 bit tương ứng với màu tốt. Tạo một nước đi mới với mỗi bit 1 của 2 bit board đó.

3.2.6.3 Nước đi đặc biệt - en passant:

Tốt có thể bắt quân en passant nếu đối phương vừa di chuyển một tốt hai ô. Kiểm tra này bao gồm việc đảm bảo rằng bắt quân en passant không để lại vua trong trạng thái bị chiếu.

Thực hiện: bàn cờ sẽ theo dõi các ô có thể thực hiện en passant (sau khi một quân tốt bất kỳ thực hiện nước đi tiến 2 ô thì sẽ tạo ra 1 ô có thể bắt en passant). Tại đây kiểm tra xem có quân tốt nào có thể bắt ào ô en passant đó không, nếu có thì tạo một nước đi mới.

3.2.6.4 Phong quân (promotion moves):

Điều kiện: Khi tốt di chuyển đến hàng cuối cùng, nó có thể phong thành hậu, xe, mã, hoặc tượng.

Thực hiện: Tạo các bit board cho các ô có thể phong cấp bằng cách tiến thẳng, bắt sang trái hoặc bắt sang phải (tổng cộng 3 bit board). Duyệt qua các ô đó và với mỗi ô thì tạo ra 4 nước cờ tương ứng với nước đi phong tốt thành hậu, xe, tượng, mã.

3.3 Tìm kiếm nước đi

Trong một chương trình chess bot, việc tìm kiếm nước đi tốt nhất là một quá trình phức tạp, yêu cầu sự kết hợp giữa các thuật toán và kỹ thuật cụ thể.

Mục tiêu của quá trình tìm kiếm là xây dựng một cây tìm kiếm đại diện cho tất cả các nước đi có thể và lựa chọn nước đi tối ưu nhất dựa trên các đánh giá có lợi cho quân cờ của mình. Phần này sẽ phân tích chi tiết các thuật toán và kỹ thuật tìm kiếm nước đi được sử dụng trong chương trình chess bot.

Chiến lược tìm kiếm của chess bot dựa trên thuật toán Minimax, kết hợp với các kỹ thuật như tìm kiếm sâu dần (Iterative Deepening), Alpha-Beta Pruning, tìm kiếm yên tĩnh (Quiescence Search), bảng lưu trữ thế cờ (Transposition Table), và sắp xếp nước đi (Move Ordering). Phần này sẽ mô tả chi tiết chiến lược tìm kiếm của chess bot, cách mà các thuật toán và kỹ thuật này hoạt động cùng nhau để tối ưu hóa quá trình tìm kiếm nước đi.

3.3.1 Tìm kiếm nước đi trong Sách khai cuộc

Sách khai cuộc (Opening book) là một tập hợp các nước đi được lưu trữ trước dựa trên các ván đấu thực tế và lý thuyết khai cuộc trong cờ vua đã được nghiên cứu kĩ lưỡng và đúc kết từ các kiện tướng cờ vua trong suốt lịch sử.

Nó giúp chess bot đưa ra các nước đi khai cuộc tối ưu mà không cần phải tính toán từ đầu, tiết kiệm thời gian và tăng hiệu suất.

Các thế cờ khai cuộc và các chuỗi nước đi áp dụng với thế cờ đó được lưu trong một file tài nguyên **Book.txt**. File này được tổng hợp dựa trên dữ liệu được ghi lại rất nhiều ván cờ trên các diễn đàn chơi cờ lớn nhất trên thế giới là Chess.com và Lichess. Các thế khai cuộc cùng các nước cờ phổ biến được áp dụng sẽ được lưu trữ dưới dạng như sau trong file:

```
pos rnbqkbnr/pppppppp/8/8/8/PPPPPPPP/RNBQKBNR w KQkq -  
e2e4 243109  
d2d4 146627  
g1f3 33009  
c2c4 22211  
f2f4 4982
```

Trong đó:

- ‘pos rnbqkbnr/pppppppp/8/8/8/PPPPPPPP/RNBQKBNR w KQkq’ : chuỗi FEN biểu diễn cho thế cờ (ở đây là thế cờ khởi đầu)
- ‘e2e4 243109’ : Nước đi được áp dụng và số lần xuất hiện của nó

Chess bot sẽ chọn một trong các nước đi với thế khai cuộc có trong sách một cách ngẫu nhiên dựa theo một trọng số cố định. Tức là chess bot sẽ không phải lúc nào cũng chọn nước đi được sử dụng nhiều nhất để tránh sự dễ đoán.

Nếu chess bot tìm thấy thế cờ hiện tại trong Sách khai cuộc thì nó sẽ chọn nước đi trực tiếp từ đó và không cần thực hiện tìm kiếm.

Ví dụ: Các nước đi đối sách cho khai cuộc Sicilian Defense được thêm vào trong sách như sau:

```
pos rnbqkbnr/pp1pppp/8/2p5/4P3/8/PPPP1PPP/RNBQKBNR w KQkq -  
g1f3 53559  
b1c3 9006  
d2d4 5591  
c2c3 4660  
f2f4 1636  
b2b4 1257  
c2c4 981  
f1c4 828
```



Sicilian Defense

1.e4 c5

The Sicilian Defense is the most popular response to White's 1.e4. Employed by masters and beginners alike, the Sicilian Defense is a reputable and positionally sound opening. Still, the Sicilian is a combative opening that tends to lead to dynamic and sharp positions.

One of the oldest registered openings, the Sicilian is full of theory and was used by most of the greatest players in history. World champions GMs [Bobby Fischer](#), [Garry Kasparov](#), [Viswanathan Anand](#), [Vladimir Kramnik](#), and [Magnus Carlsen](#) are just a few of its adopters.

- Starting Position



[Sicilian Defense - Chess Openings - Chess.com](#)

Tuy nhiên, số lượng thế cờ khai cuộc được ghi nhận là không quá nhiều (khoảng hơn 22 nghìn thế khai cuộc trong file) nên quá trình khai cuộc sẽ sớm kết thúc sau tối đa 7 nước từ hai bên. Sau quá trình đó thì chess bot sẽ phải tự tìm nước đi tốt nhất bằng cách thực hiện quá trình tìm kiếm được giới thiệu ở các phần tiếp theo.

3.3.2 Khởi tạo tìm kiếm

Chess bot khởi tạo các biến trạng thái và chuẩn bị cho quá trình tìm kiếm.

Thiết lập bảng lưu trữ thế cờ (Transposition Table) và bảng lặp lại thế cờ (Repetition Table).

3.3.3 Bắt đầu tìm kiếm sâu dần (Iterative Deepening Search)

Chess bot bắt đầu xây dựng cây tìm kiếm với độ sâu 1 và tăng dần độ sâu lên 2, 3, và tiếp tục cho đến khi hết thời gian hoặc đạt đến độ sâu giới hạn.

Mỗi lần gọi hàm tìm kiếm, bot lưu lại nước đi tốt nhất tìm được và cập nhật giá trị đánh giá (eval) của nước đi đó.

3.3.4 Minimax

Với mỗi lần gọi đệ quy của thuật toán Minimax, chess bot kiểm tra thế cờ hiện tại trong bảng Transposition Table.

Nếu thế cờ đã được đánh giá trước đó, chess bot lấy giá trị đánh giá từ bảng và không cần tiếp tục tìm kiếm.

Nếu thế cờ không tồn tại trong bảng lưu trữ, chess bot sẽ gọi đến MoveGenerator để sinh tất cả các nước đi của thế cờ hiện tại.

Sau đó, với mỗi nước đi chess bot sẽ thực hiện nước đi đó trên bàn cờ để tạo ra thế cờ mới sau đó tiếp tục gọi đệ quy hàm Minimax để tìm kiếm với thế cờ mới.

Sau khi lời gọi đệ quy trả về kết quả, chess bot sẽ hoàn tác nước đi được thực hiện trước đó, kiểm tra và cắt bỏ các nhánh beta, đồng thời cập nhật giá trị alpha với giá trị eval lớn nhất hiện tại.

Sau đó, giá trị alpha này được ghi vào bảng lưu trữ trạng thái (Transposition Table) cùng với Zobrist key của thế cờ, nước đi trước đó, loại nước đi, và độ sâu của thế cờ.

Cuối cùng giá trị alpha được trả về thể hiện cho eval lớn nhất tìm được trong lần tìm kiếm này.

3.3.5 Alpha-Beta Pruning

Khi thực hiện đệ quy, chess bot duy trì các biến alpha và beta để quản lý việc cắt tỉa các nhánh không cần thiết trong cây tìm kiếm. Biến alpha đại diện cho giá trị đánh giá tốt nhất mà người chơi hiện tại có thể đạt được, trong khi beta đại diện cho giá trị đánh giá tệ nhất mà đối thủ có thể chấp nhận.

Cắt tỉa Beta (Beta Cutoff): Nếu giá trị đánh giá (eval) của nước đi lớn hơn hoặc bằng giá trị beta, chess bot cắt bỏ nhánh này. Điều này có nghĩa là đối thủ sẽ không chọn nước đi này vì có lựa chọn tốt hơn.

Cập nhật Alpha (Alpha Update): Nếu giá trị đánh giá (eval) của nước đi lớn hơn giá trị alpha, chess bot cập nhật giá trị alpha và ghi nhận nước đi tốt nhất tại thời điểm đó.

Giá trị alpha cũng là giá trị cuối cùng sẽ được trả về của hàm Minimax, đại diện cho eval lớn nhất tìm được trong đợt tìm kiếm này.

3.3.6 Sắp xếp nước đi (Move Ordering)

Sắp xếp nước đi (move ordering) là một kỹ thuật nhằm tối ưu hóa quá trình tìm kiếm và tăng hiệu quả của thuật toán Alpha-Beta Pruning. Việc sắp xếp các nước đi giúp tăng khả năng cắt tỉa các nhánh không cần thiết, từ đó giảm khối lượng tính toán và tăng tốc độ tìm kiếm.

1. Khởi tạo điểm số cho mỗi nước đi

Mỗi nước đi được gán một điểm số (score) dựa trên một số yếu tố:

- **Nước đi từ bảng lưu trữ:** Nếu nước đi trùng với nước đi lưu trong bảng Transposition Table, nó được gán điểm số cao nhất (HashMoveScore).

- **Nước đi ăn quân:** Nếu nước đi là nước ăn quân, nó được tính toán dựa trên giá trị của quân cờ bị ăn và quân cờ ăn. Nước đi ăn quân có giá trị cao hơn được ưu tiên, đặc biệt là khi quân bị ăn có giá trị cao hơn quân ăn.
- **Phong hậu:** Nếu nước đi là phong quân thành hậu, nó được gán điểm số cao (PromoteBias).
- **Nước đi giết:** Các nước đi đã từng dẫn đến cắt tỉa trong các bước tìm kiếm trước đó được gán điểm số cao (KillerBias).
- **Lịch sử nước đi:** Các nước đi có lịch sử tốt trong các bước tìm kiếm trước đó cũng được ưu tiên.

2. Cập nhật điểm số cho mỗi nước đi

Mỗi nước đi được cập nhật điểm số dựa trên các yếu tố sau:

- Nước đi ăn quân:
 - Nếu nước đi ăn quân có thể bị đối thủ tái chiếm, điểm số của nước đi này sẽ được điều chỉnh dựa trên chênh lệch giá trị của quân cờ bị ăn và quân cờ ăn (WinningCaptureBias hoặc LosingCaptureBias).
 - Nếu nước đi ăn quân không thể bị đối thủ tái chiếm, điểm số của nó sẽ được cộng thêm giá trị cao (WinningCaptureBias).
- Phong Hậu:
 - Nước đi phong quân thành hậu mà không ăn quân sẽ được cộng điểm số cao (PromoteBias).
- Nước đi thường (Quiet Moves):
 - Điểm số được điều chỉnh dựa trên giá trị của ô đích so với ô bắt đầu của quân cờ, theo bản đồ giá trị (PieceSquareTable).
 - Nếu ô đích bị quân tốt của đối phương tấn công, điểm số sẽ bị trừ đi.
- Các nước đi giết (Killer Moves) và các nước đi có lịch sử tốt cũng được cộng thêm điểm số tương ứng.

3. Sắp xếp nước đi (quicksort)

Sau khi tất cả các nước đi đã được gán điểm số, chúng được sắp xếp theo thứ tự giảm dần của điểm số bằng thuật toán Quicksort.

3.3.7 Tìm kiếm yên tĩnh (Quiescence Search)

[Quiescence Search - Chessprogramming wiki](#)

Khi đạt đến độ sâu tối đa, Minimax gọi thuật toán Quiescence Search để tiếp tục mở rộng cây tìm kiếm với các nước đi nguy hiểm (như ăn quân, chiếu tướng,...).

Quiescence Search giúp tránh đánh giá sai lầm từ các thế cờ không ổn định, gọi là “horizon effect” khi mà đạt đến độ sâu giới hạn của cây tìm kiếm nhưng có thể ngay sau đó là một nước đi nguy hiểm mà chess bot sẽ không thể nhận biết được nếu dừng tìm kiếm ngay lúc đó.

Thuật toán chỉ dừng lại khi bàn cờ tĩnh lặng (không còn các nước nguy hiểm), sau đó nó gọi hàm Evaluation để tính toán giá trị eval dựa trên thế cờ hiện tại.

Ví dụ: Nếu thế cờ ở độ sâu cuối cùng của cây tìm kiếm là một nước lấy hậu ăn xe đối phương, nếu chess bot dừng tìm kiếm và tiến hành đánh giá bàn cờ luôn thì chắc chắn thế cờ này sẽ có một giá trị eval cao có lợi cho bot. Tuy nhiên nếu như ngay sau thế cờ đó thì đổi phương có thể bắt ngược lại quân hậu của mình thì nó lại là một thế cờ bất lợi khi bot đã đổi hậu lấy xe đối phương. Quiescence Search sẽ giúp chess bot mở rộng cây tìm kiếm ra các nước bắt quân đó để chắc chắn rằng một nước cờ là tối ưu nhất có thể.

3.3.8 Lưu trữ kết quả bằng Transposition Table

1. Cấu trúc của Transposition Table

Transposition Table được triển khai dưới dạng một mảng các entry mỗi entry chứa các thông tin sau:

- **Key:** Zobrist key của vị trí cờ (đại diện cho trạng thái bàn cờ).
- **Value:** Giá trị đánh giá của trạng thái bàn cờ.
- **Move:** Nước đi đã được thực hiện để đến vị trí này.
- **Depth:** Độ sâu của cây tìm kiếm kể từ gốc cho đến vị trí này.
- **NodeType:** Loại nút (chính xác, giới hạn dưới, hoặc giới hạn trên) của giá trị đánh giá.

Trong đó NodeType là một giá trị biểu thị cách giá trị đánh giá được lưu trữ và cách nó nên được sử dụng trong quá trình tìm kiếm tiếp theo. Có ba loại nút chính: Exact, LowerBound, và UpperBound. Dưới đây là giải thích chi tiết về từng loại nút và ý nghĩa của chúng.

- **Exact (chính xác):** Giá trị đánh giá được lưu trữ là giá trị chính xác của vị trí cờ đã được đánh giá. Đây là trường hợp lý tưởng khi một vị trí đã được tìm kiếm đủ sâu và đánh giá chính xác, không cần phải tìm kiếm thêm.
- **LowerBound (cận dưới):** Giá trị đánh giá được lưu trữ là giới hạn dưới của giá trị thực tế. Điều này có nghĩa là giá trị thực tế của vị trí này có thể cao hơn hoặc bằng giá trị được lưu trữ. Loại nút này thường xuất hiện khi xảy ra beta cut-off trong quá trình tìm kiếm alpha-beta.
- **UpperBound (cận trên):** Giá trị đánh giá được lưu trữ là giới hạn trên của giá trị thực tế. Điều này có nghĩa là giá trị thực tế của vị trí này có thể thấp hơn hoặc bằng giá trị được lưu trữ. Loại nút này thường xuất hiện khi không có nước đi nào trong vị trí hiện tại mang lại giá trị tốt hơn alpha trong quá trình tìm kiếm alpha-beta.

2. Cách hoạt động của Transposition Table

Lưu Trữ: Khi một vị trí cờ được đánh giá, kết quả của nó sẽ được lưu vào bảng cùng với zobrist key và độ sâu tìm kiếm.

Tra Cứu: Trước khi đánh giá một vị trí mới, chess bot sẽ kiểm tra xem vị trí này đã được đánh giá trước đó hay chưa. Nếu có, nó sẽ sử dụng giá trị đã lưu để tránh việc tính toán lại.

3. Lợi ích khi sử dụng Transposition Table

Tối ưu hóa alpha-beta pruning: Transposition Table giúp tối ưu hóa thuật toán alpha-beta pruning bằng cách lưu trữ các giá trị đánh giá của các vị trí cờ đã được kiểm tra. Điều này giúp giảm số lượng nhánh cần phải kiểm tra trong cây tìm kiếm, từ đó tăng tốc độ tìm kiếm.

Giảm thiểu số lượng phép tính: Khi một vị trí đã được đánh giá trước đó, chess bot có thể tra cứu bảng để lấy kết quả thay vì phải tính toán lại từ đầu. Điều này giúp giảm thiểu số lượng phép tính cần thực hiện, tiết kiệm tài nguyên và cải thiện hiệu suất tổng thể của bot.

Xác định chính xác hơn giá trị đánh giá: Bằng cách lưu trữ các giá trị đánh giá chính xác hoặc gần chính xác (cận trên/dưới), chess bot có thể sử dụng lại thông tin này để cải thiện độ chính xác của các đánh giá trong các lần tìm kiếm sau. Điều này đặc biệt hữu ích trong các tình huống phức tạp hoặc khi đổi mặt với các thế cờ tương tự.

3.3.9 Quản lý thời gian

Quá trình tìm kiếm của chess bot sẽ được đảm nhận bởi một thread riêng với thời gian time out được đặt trước bằng giá trị mà chess bot cấp cho lần tìm kiếm này. Giá trị time out này được tính toán dựa trên tổng thời gian còn lại của đối phương và của bản thân sao cho tối ưu với hoàn cảnh ván đấu.

Khi hết thời gian, bot sẽ lấy nước đi tốt nhất tìm được cho đến thời điểm đó để đưa ra quyết định.

3.3.10 Hoàn thành tìm kiếm

Khi thời gian tìm kiếm kết thúc hoặc tìm được nước đi tối ưu (điểm eval chỉ ra thế cờ chắc thắng cho một bên tức là khi một bên chiếu hết cờ), chess bot kết thúc quá trình tìm kiếm.

Bot lưu lại nước đi tốt nhất, giá trị eval tương ứng và các thông tin chẩn đoán về quá trình tìm kiếm.

3.4 Đánh giá thế cờ

Lớp Evaluator trong phần mềm chess bot có trách nhiệm đánh giá thế cờ hiện tại dựa trên nhiều chiến thuật và yếu tố khác nhau. Mục tiêu chính của lớp này là tính toán một giá trị đánh giá (eval) cho mỗi vị trí cờ, giúp bot xác định nước đi tốt nhất mà nó có thể đưa ra. Giá trị đánh giá này phản ánh lợi thế của bên hiện tại và được tính toán dựa trên nhiều yếu tố như giá trị quân cờ, vị trí của quân cờ trên bàn cờ, cấu trúc quân tốt, và trạng thái bảo vệ vua.

Đánh giá thế cờ được coi là hàm “heuristic” của chess bot, là não bộ và logic, thể hiện chiến lược chơi cờ của nó. Một hàm đánh giá được thiết kế tỉ mỉ, công phu với các chiến thuật chính xác sẽ khiến sức mạnh của chess bot tăng lên đáng kể. Trong phần này sẽ bàn luận về các yếu tố và chiến thuật đánh giá được sử dụng.

3.4.1 Giá trị quân số hiện tại (Material Score)

Giá trị quân số là yếu tố cơ bản và quan trọng nhất trong đánh giá thế cờ. Mỗi loại quân cờ được gán một giá trị số, phản ánh sức mạnh và tầm quan trọng của nó:

- Tốt (Pawn): 100
- Mã (Knight): 300
- Tượng (Bishop): 320
- Xe (Rook): 500
- Hậu (Queen): 900

Khi một bên còn càng nhiều quân cờ có giá trị cao trên bàn cờ thì bên đó càng được coi là có lợi thế hơn.

3.4.2 Chỉ số tàn cuộc

Giá trị EndgameTrans là một số liệu đại diện cho mức độ tiến gần đến tàn cuộc (endgame) trong một ván cờ. Nó là một giá trị từ 0 đến 1, trong đó:

- 0: Đại diện cho giai đoạn khai cuộc (opening) hoặc trung cuộc (middlegame).
- 1: Đại diện cho giai đoạn tàn cuộc (endgame).

Giá trị này được tính toán dựa trên số lượng của loại quân cờ còn lại trên bàn cờ nhân với trọng số được đặt trước của quân đó và chia cho giá trị trọng số của tất cả các quân trên bàn cờ khởi đầu. Khi các quân cờ chính như hậu, xe, tượng, và mã (kết hợp với trọng số của từng quân) giảm dần, giá trị EndgameTrans tăng lên, phản ánh sự chuyển đổi từ giai đoạn khai cuộc hoặc trung cuộc sang tàn cuộc.

Giá trị này được theo dõi để chess bot có thể áp dụng các chiến thuật khác nhau lên quá trình đánh giá thế cờ với các giai đoạn của trận đấu. Với mỗi giai đoạn của trận đấu thì chiến thuật được áp dụng sẽ khác nhau.

3.4.3 Bảng điểm vị trí (Piece-Square Tables)

Mỗi loại quân cờ có một bảng điểm vị trí tương ứng, mô tả giá trị của quân cờ tại mỗi ô trên bàn cờ. Sở dĩ có bảng điểm này là vì với mỗi quân cờ khác nhau thì có một số vị trí trên bàn cờ được coi là thuận lợi hơn cho quân đó. Bảng điểm này giúp đánh giá xem một quân cờ có ở vị trí tốt hay không.

Ví dụ bảng điểm vị trí của quân mã trắng:

```
-50,-40,-30,-30,-30,-40,-50,  
-40,-20,  0,  0,  0,-20,-40,  
-30,  0, 10, 15, 15, 10,  0,-30,  
-30,  5, 15, 20, 20, 15,  5,-30,  
-30,  0, 15, 20, 20, 15,  0,-30,  
-30,  5, 10, 15, 15, 10,  5,-30,
```

$-40, -20, 0, 5, 5, 0, -20, -40,$
 $-50, -40, -30, -30, -30, -30, -40, -50,$

Bảng điểm này thể hiện các ô ở trung tâm bàn cờ có điểm số cao hơn và các ô gần góc bàn cờ thì điểm số thấp. Đó là vì các ô ở trung tâm có lợi thế hơn nếu quân mã chiếm được chúng, bởi vì từ đó thì quân mã dễ dàng triển khai và gây áp lực lên đối phương.

Lưu ý: Ở trên là bảng điểm vị trí của quân mã trắng, với quân mã đen và các quân bên đen khác thì sẽ dùng một bảng điểm được đảo ngược so với quân trắng.

Có hai quân cờ có một bảng điểm vị trí cho giai đoạn khai cuộc và trung cuộc và một bảng khác cho giai đoạn tàn cuộc của ván đấu là quân vua và quân tốt.

- Với quân tốt:

- Bảng điểm vị trí cho giai đoạn khai cuộc và trung cuộc:

$0, 0, 0, 0, 0, 0, 0, 0,$
 $50, 50, 50, 50, 50, 50, 50, 50,$
 $10, 10, 20, 30, 30, 20, 10, 10,$
 $5, 5, 10, 25, 25, 10, 5, 5,$
 $0, 0, 0, 20, 20, 0, 0, 0,$
 $5, -5, -10, 0, 0, -10, -5, 5,$
 $5, 10, 10, -20, -20, 10, 10, 5,$
 $0, 0, 0, 0, 0, 0, 0, 0$

Bảng điểm này thể hiện ở những giai đoạn đầu thì các quân tốt ở hai cánh nên được giữ lại để bảo vệ vua còn quân tốt ở trung tâm thì nên được đẩy lên cách hàng cao hơn nhằm triển khai thế trận. Nếu quân tốt tiến được sâu vào lãnh thổ đối phương thì càng dễ để phong hậu nên càng được điểm cao hơn.

- Bảng điểm vị trí cho giai đoạn tàn cuộc

$0, 0, 0, 0, 0, 0, 0, 0,$
 $80, 80, 80, 80, 80, 80, 80, 80,$
 $50, 50, 50, 50, 50, 50, 50, 50,$
 $30, 30, 30, 30, 30, 30, 30, 30,$
 $20, 20, 20, 20, 20, 20, 20, 20,$
 $10, 10, 10, 10, 10, 10, 10, 10,$
 $10, 10, 10, 10, 10, 10, 10, 10,$
 $0, 0, 0, 0, 0, 0, 0, 0$

Tiến về tàn cuộc thì nhìn chung tất cả quân tốt đều nên được đẩy dần lên lãnh thổ đối phương để tìm kiếm cơ hội phong hậu.

- Với quân vua:
 - Bảng điểm cho giai đoạn khai cuộc và tàn cuộc:

-80, -70, -70, -70, -70, -70, -70, -80,
 -60, -60, -60, -60, -60, -60, -60, -60,
 -40, -50, -50, -60, -60, -50, -50, -40,
 -30, -40, -40, -50, -50, -40, -40, -30,
 -20, -30, -30, -40, -40, -30, -30, -20,
 -10, -20, -20, -20, -20, -20, -20, -10,
 20, 20, -5, -5, -5, -5, 20, 20,
 20, 30, 10, 0, 0, 10, 30, 20

Bảng điểm thể hiện rằng ở giai đoạn đầu thì vua nên đứng sâu trong lãnh thổ của mình, đặc biệt là nếu nhập thành được với xe thì sẽ có nhiều lợi thế hơn.

- Bảng điểm cho giai đoạn tàn cuộc:

-20, -10, -10, -10, -10, -10, -10, -20,
 -5, 0, 5, 5, 5, 5, 0, -5,
 -10, -5, 20, 30, 30, 20, -5, -10,
 -15, -10, 35, 45, 45, 35, -10, -15,
 -20, -15, 30, 40, 40, 30, -15, -20,
 -25, -20, 20, 25, 25, 20, -20, -25,
 -30, -25, 0, 0, 0, 0, -25, -30,
 -50, -30, -30, -30, -30, -30, -30, -50

Tiến về tàn cuộc, khi mà các quân cờ mạnh như xe và hậu (của cả hai bên) có thể đã không còn nhiều thì vua không nên chỉ đứng yên ở cuối bàn cờ nữa mà cũng nên được triển khai lên trung tâm bàn cờ để gây áp lực lên đối phương.

Chess bot sẽ kết hợp chỉ số EndgameTrans của thế cờ hiện tại với giá trị vị trí hiện tại của 2 quân cờ trên để điều chỉnh điểm vị trí của quân tốt và quân vua giúp chess bot đưa ra các đánh giá chính xác hơn dựa trên giai đoạn của ván cờ. Nghĩa là trong một giai đoạn thì chess bot sẽ dùng cả hai bảng điểm cho chung cuộc và tàn cuộc nhưng trọng số của 2 điểm đó sẽ được quyết định bởi EndgameTrans.

3.4.4 Cấu trúc quân tốt (Pawn Structure)

Cấu trúc quân tốt là một yếu tố quan trọng trong đánh giá thế cờ.

Đánh giá quân tốt bao gồm áp dụng điểm cộng cho các quân tốt thông (passed pawns) và điểm trừ cho các quân tốt bị cô lập (isolated pawns).

3.4.4.1 Quân tốt thông (Passed Pawns)

Quân tốt thông là quân tốt không có quân đối phương ở phía trước trên cùng cột hoặc các cột liền kề, có tiềm năng tiến tới hàng phong cấp.

Xác định quân tốt thông: Chess bot sử dụng một mảng mặt nạ bit (bit masks) để xác định các quân tốt thông đối với quân tốt trắng và đen.

Tính toán bonus: Nếu một quân tốt là quân tốt thông, chess bot sẽ cộng thêm một điểm thưởng (bonus) dựa trên số ô còn lại để phong cấp. Giá trị bonus tăng dần khi quân tốt thông càng tiến gần đến hàng phong hậu.

3.4.4.2 Quân tốt bị cô lập (Isolated Pawns)

Quân tốt bị cô lập là quân tốt không có các quân tốt đồng đội trên các cột liền kề hoặc đường chéo, làm cho nó dễ bị tấn công và khó bảo vệ.

Xác định quân tốt cô lập: Chess bot sử dụng mặt nạ bit để kiểm tra xem có quân tốt nào của đội nhà nằm trên các cột liền kề hay không.

Tính toán điểm phạt: Nếu một quân tốt là quân tốt cô lập, chess bot sẽ cộng thêm một điểm phạt (penalty) dựa trên số lượng quân tốt cô lập. Giá trị penalty tăng dần nếu có nhiều quân tốt bị cô lập.

Phương pháp đánh giá quân tốt của chess bot giúp xác định các quân tốt thông hành và quân tốt cô lập, từ đó áp dụng các điểm thưởng và điểm phạt tương ứng. Điều này giúp cải thiện khả năng đánh giá thế cờ của bot và đưa ra các quyết định chiến thuật phù hợp trong từng giai đoạn của ván cờ.

3.4.5 Bảo vệ vua (King Safety)

Độ an toàn của vua là một yếu tố không thể bỏ qua khi đánh giá thế cờ, chess bot sẽ cộng thêm điểm nếu vua đang được bảo vệ tốt và trừ điểm nếu ngược lại.

Sự an toàn của vua được đánh giá dựa trên một số yếu tố bao gồm lá chắn tốt phía trước vua, trạng thái nhập thành, các hàng bị hở.

3.4.5.1 Kiểm tra trạng thái nhập thành

Nếu vua đứng ở vị trí chưa nhập thành (hàng từ 3 đến 6), một điểm phạt được tính dựa trên mức độ phát triển của quân đối phương.

Mức độ phát triển của quân đối phương được tính bằng cách chuẩn hóa điểm số phát triển của quân đối phương (điểm số này là tổng hợp các điểm số sau khi đánh giá bằng điểm vị trí của các quân của đối phương) kết hợp với một số điểm cố định trong khoảng từ 0 đến 1.

3.4.5.2 Điểm phạt khi có hàng phía trước vua bị hở

Nếu đối phương có hơn một quân xe hoặc có cả xe và hậu, chess bot kiểm tra các hàng liền kề với vua.

Nếu các hàng đó không có quân tốt bảo vệ, cộng thêm điểm phạt dựa trên việc hàng đó có phải hàng của vua hay không và có quân tốt bảo vệ hay không.

Điều này giúp tránh vua bị các quân cờ mạnh có khả năng di chuyển trên các hàng như xe và hậu nhầm đến.

3.4.5.3 Lá chắn quân tốt

Nếu quân vua đã nhập thành (đứng ở hàng 3 hoặc 6), nó kiểm tra các ô vuông phía trước vua để xem có đủ quân tốt bảo vệ hay không. Trường hợp lí tưởng, vua nên có 3 quân tốt bảo vệ phía trước.

Sử dụng các mảng được tính toán trước để xác định các ô vuông cần kiểm tra.

Nếu các ô vuông không có quân tốt bảo vệ, cộng điểm phạt. Điểm phạt sẽ cao hơn nếu quân tốt phía trước bị thiếu nhiều.

Càng tiến về giai đoạn tàn cuộc thì độ quan trọng của lá chắn tốt càng giảm. Vì vậy điểm phạt của lá chắn tốt sẽ tỉ lệ nghịch với EndgameTrans.

3.4.6 Đẩy lùi và quây bắt vua đối phương

Chess bot sẽ đánh giá thế cờ trong giai đoạn tàn cuộc (endgame), đặc biệt khi một bên có lợi thế về quân số sau đó khuyến khích bên đó sử dụng quân vua của mình để đẩy quân vua của đối phương về phía cạnh hoặc góc bàn cờ, tạo áp lực và gia tăng cơ hội chiếu hết cờ.

3.4.6.1 Kiểm tra điều kiện áp dụng

Phương thức này chỉ được thực hiện nếu bên đang được đánh giá có lợi thế về quân số lớn hơn giá trị của hai quân tốt tức là đang có lợi thế lớn về quân số.

Điều kiện thứ hai là thế cờ hiện tại đang tiến đến tàn cuộc (xác định sử dụng chỉ số EndgameTrans).

3.4.6.2 Thực hiện đánh giá

Đầu tiên xác định vị trí của hai quân vua.

Xác định khoảng cách vuông góc giữa hai quân vua và khoảng cách Manhattan giữa quân vua đối phương với trung tâm bàn cờ.

Điểm số khuyến khích tăng thêm sẽ dựa trên hai khoảng cách này, với mục tiêu là giảm khoảng cách giữa hai vua (di chuyển quân vua của mình gần hơn vua đối phương) và đồng thời đẩy quân vua của đối phương về phía cạnh bàn cờ (yếu tố này có trọng số lớn hơn).

Bằng cách này, bên có lợi thế sẽ liên tục tạo áp lực lên vua đối phương ở giai đoạn tàn cuộc, từ đó chess bot có thể tối ưu hóa chiến lược tấn công và gia tăng cơ hội chiến thắng.

3.4.7 Tính toán giá trị đánh giá tổng

Các điểm cộng và điểm trừ từ các bước đánh giá trên sẽ được tổng hợp lại và trả về trong hàm Evaluate.

Giá trị trả về sẽ là giá trị âm hoặc dương tùy theo bên của lượt chơi hiện tại.

4 Kết quả đánh giá và thảo luận

Phần này trình bày kết quả đánh giá AI chess bot được phát triển trong dự án. Để đảm bảo tính khách quan và xác thực của kết quả, nhóm đã sử dụng hai phương pháp đánh giá chính: sử dụng giao thức UCI để đấu với các chess bot trong phần mềm PyChess và thực hiện sử dụng chess bot để đấu trực tiếp với các chess bot trên nền tảng Chess.com.

4.1 Thiết lập giao thức UCI

Giao thức UCI (Universal Chess Interface) là một chuẩn giao tiếp được sử dụng rộng rãi giữa các engine cờ vua và giao diện người dùng (GUI). UCI cho phép các engine cờ vua thực hiện các trận đấu với nhau hoặc với con người, thông qua việc gửi và nhận các lệnh dưới dạng văn bản.

4.1.1 Các bước triển khai giao thức UCI cho chess bot

Khởi tạo giao diện UCI:

- Thiết lập môi trường: Xây dựng một môi trường console để chess bot có thể nhận và xử lý các lệnh UCI từ GUI.
- Khởi tạo lệnh UCI: Chess bot sẽ nhận lệnh uci từ GUI để bắt đầu quá trình giao tiếp. Bot phản hồi với các thông tin cơ bản như tên, phiên bản, tác giả.

Quản lý các lệnh UCI:

- isready: Lệnh này được gửi từ GUI để kiểm tra xem chess bot đã sẵn sàng hay chưa. Bot phản hồi bằng lệnh readyok.
- ucinewgame: Lệnh này được gửi khi bắt đầu một trận đấu mới. Bot thiết lập lại trạng thái bàn cờ và các thông số cần thiết.
- position: Lệnh này thiết lập trạng thái bàn cờ, bao gồm vị trí các quân cờ và lịch sử các nước đi.
- go: Lệnh này yêu cầu bot bắt đầu tìm kiếm nước đi tốt nhất từ trạng thái bàn cờ hiện tại. Bot trả về nước đi đã chọn.

Gửi và nhận dữ liệu:

- bestmove: Sau khi tìm kiếm, chess bot gửi lệnh bestmove kèm theo nước đi tốt nhất cho GUI.
- info: Trong quá trình tìm kiếm, chess bot có thể gửi các thông tin về trạng thái tìm kiếm như độ sâu tìm kiếm, đánh giá thế cờ hiện tại, số lượng nước đi đã xem xét, v.v.

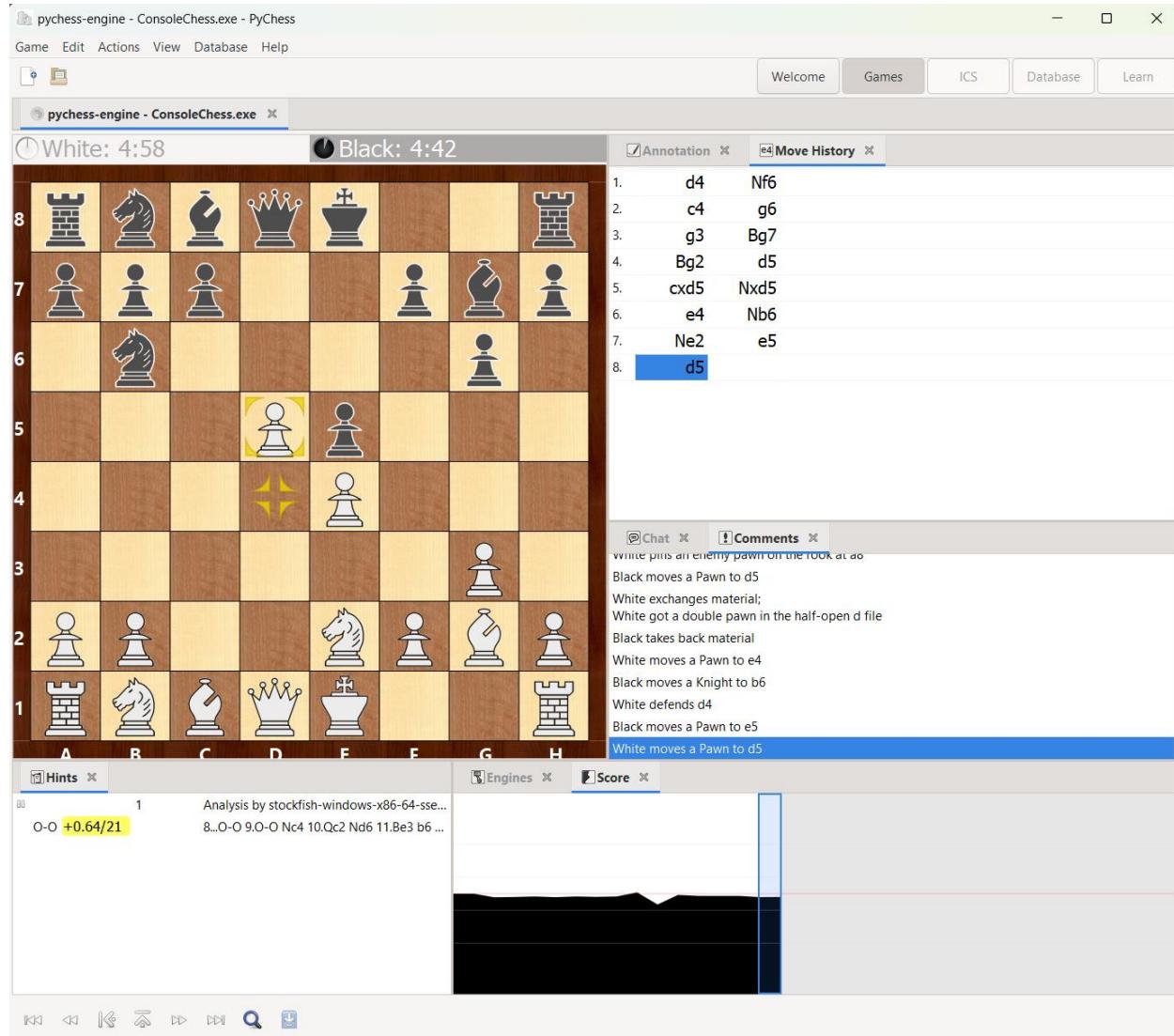
4.2 Đánh giá với các AI chess trong PyChess

4.2.1 Tích hợp chess bot vào PyChess

Để đánh giá hiệu suất của chess bot, nhóm đã tích hợp nó với PyChess, một GUI cờ vua mã nguồn mở cho phép thêm các engine cờ vua tuân theo giao thức UCI. Các bước thực hiện bao gồm:

- Cấu hình PyChess: Thêm chess bot vào danh sách các engine của PyChess bằng cách cấu hình file engine trong PyChess để nhận diện file .exe của chess bot qua giao thức UCI.
- Thiết lập trận đấu: Sử dụng giao diện của PyChess để thiết lập các trận đấu giữa chess bot và các AI chess khác.
- Ghi nhận kết quả: Ghi lại kết quả các trận đấu, bao gồm tỷ lệ thắng, thời gian xử lý nước đi, và các thông tin chi tiết khác để phân tích.

4.2.2 Đánh giá trên PyChess



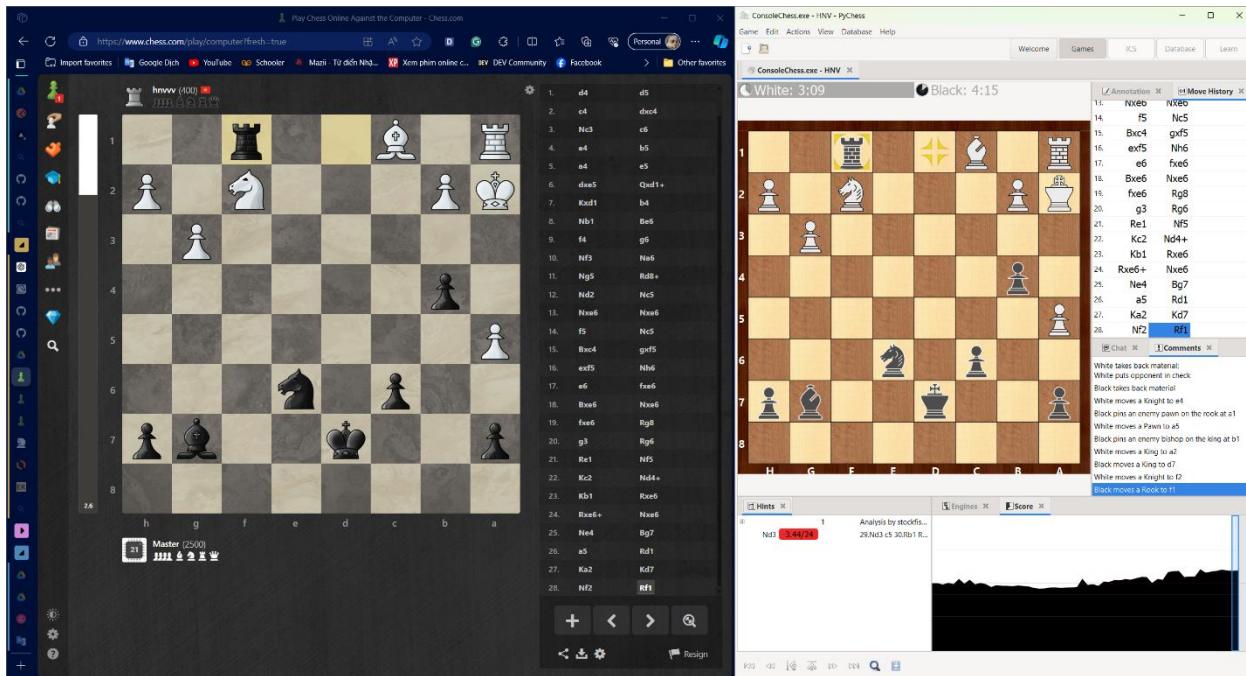
Các chỉ tiêu đánh giá:

- Tỷ lệ thắng: Tỷ lệ phần trăm các trận thắng của chess bot trong tổng số trận đấu.
- Khả năng phân tích nước đi: Đánh giá khả năng của bot trong việc phân tích và chọn nước đi tối ưu. Nước đi của chess bot được so sánh với đánh giá được đưa ra từ Stockfish 16.1 để xem độ chính xác trong các nước đi của chess bot được phát triển.
- Thời gian xử lý: Thời gian trung bình để bot đưa ra nước đi trong mỗi lượt.

4.3 Đánh giá trên nền tảng Chess.com

Để có cái nhìn toàn diện hơn về hiệu suất của chess bot, nhóm đã sử dụng nó để đấu với các chess bot trên nền tảng Chess.com. Các bước thực hiện bao gồm:

- Chọn đối thủ: Chọn các chess bot trên Chess.com với mức elo từ 2300 đến 2500 để đảm bảo tính cạnh tranh cao.
- Thiết lập trận đấu: Sử dụng tài khoản trên Chess.com, thiết lập các trận đấu giữa chess bot của được phát triển và các chess bot được chọn.
- Thu thập dữ liệu: Ghi lại kết quả các trận đấu, bao gồm tỷ lệ thắng, thời gian xử lý nước đi, và các bình luận từ hệ thống của Chess.com.



Chess bot được phát triển cầm quân trắng đấu với Komodo21 (2500 elo) cầm quân đen

4.4 Kết quả đánh giá

4.4.1 Kết quả trong PyChess

Tỷ lệ thắng: Chess bot đạt tỷ lệ thắng 75% trong tổng số 100 trận đấu với các AI chess khác nhau trong PyChess, các AI này có mức elo được cung cấp là vào khoảng 2400 – 2600.

Khả năng phân tích nước đi: Chess bot thể hiện khả năng phân tích và chọn nước đi tối ưu trong 85% các tình huống phức tạp.

Thời gian xử lý: Thời gian trung bình để chess bot đưa ra được tối ưu tuỳ theo thời gian tổng của trận đấu và thời gian còn lại của nó.

4.4.2 Kết quả trên Chess.com

Tỷ lệ thắng: Chess bot đạt tỷ lệ thắng 80% khi đấu với các chess bot có mức elo từ 2300 đến 2500.

Phân tích chi tiết: Chess bot thể hiện khả năng cạnh tranh cao, đặc biệt trong giai đoạn tàn cuộc, với một số trận đấu được hệ thống Chess.com đánh giá là "xuất sắc" với tỉ lệ chính xác (accuracy) trung bình khoảng 94% - 97%.

Thời gian xử lý: Thời gian trung bình để chess bot đưa ra nước đi là 3 giây mỗi lượt, phù hợp với tiêu chuẩn của các trận đấu ở mức elo cao.

4.5 Kết luận

Kết quả đánh giá cho thấy chess bot được phát triển trong dự án có khả năng cạnh tranh cao và đáp ứng tốt các yêu cầu của một AI chess bot hiện đại. Tỷ lệ thắng cao và khả năng phân tích nước đi tối ưu là những điểm mạnh nổi bật, trong khi thời gian xử lý vẫn cần được cải thiện thêm để đạt hiệu suất tốt hơn trong các trận đấu ở mức elo cao hơn.

Mức elo của chess bot được phát triển ước tính nằm trong khoảng 2400 – 2500.

5 Tài liệu tham khảo

ⁱ Chessprogramming Wiki: [Zobrist Hashing - Chessprogramming wiki](#)