**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

---------------o0o---------------

**Logo, company name

Description automatically generated**

**BÁO CÁO MÔN HỌC**

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

**XÂY DỰNG AI CHƠI CỜ VUA ĐƠN GIẢN**

**GVHD: PGS TS Thân Quang Khoát**

**Nhóm: 07**

**SVTH:** **Nguyễn Minh An 20200004**

**Trần Xuân Bách 20204634**

**Lê Duy Anh 20204629**

**Hoàng Đức Thành 20204689**

***LỜI CẢM ƠN***

***Bài tập môn học này là kết quả được đúc kết từ quá trình học tập môn học Nhập môn Trí tuệ nhân tạo và cùng nhau nghiên cứu bổ sung của các thành viên nhóm 07. Cảm ơn thầy và các bạn chú ý theo dõi. Chúc thầy và các bạn luôn hạnh phúc và thành công trên con đường sự nghiệp!***

*Tp. Hà Nội, tháng 2 năm 2023.*

**Sinh viên**

**Nhóm 07**

MỤC LỤC

[**GIỚI THIỆU BÀI TOÁN** 4](#_Toc126616824)

[**1.** **Trò chơi cờ vua:** 4](#_Toc126616825)

[**2.** **Bài toán xây dựng một AI chơi cờ vua:** 4](#_Toc126616826)

[**GIẢI THUẬT ĐỀ XUẤT** 8](#_Toc126616827)

[**THỰC NGHIỆM** 15](#_Toc126616828)

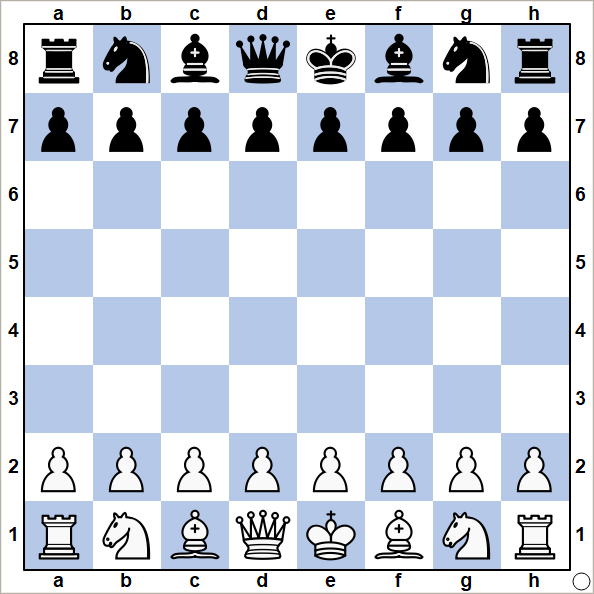
[**KẾT LUẬN** 19](#_Toc126616829)

[**TÀI LIỆU THAM KHẢO** 19](#_Toc126616830)

# **GIỚI THIỆU BÀI TOÁN**

## **Trò chơi cờ vua:**

Cờ vua là trò chơi có lịch sử lâu đời trên thế giới và là môn thể thao trí tuệ dành cho 2 người chơi. Ngày này, cờ vua là trò chơi phổ biến nhất trên thế giới với hàng triệu người chơi tại nhà riêng, trong các câu lạc bộ, trực tuyến, từ xa hay trong các giải đấu. Cờ vua được chơi bằng cách sử dụng một bàn cờ với 8 hàng và 8 cột, và mỗi người chơi sử dụng 16 quân cờ để cố gắng chiếm đất địa của đối thủ và đánh bại quân vua của họ.



*Hình 1. Một bàn cờ vua tiêu chuẩn*

## **Bài toán xây dựng một AI chơi cờ vua:**

Xét một ván cờ vua, bài toán xây dựng AI chơi cờ vua là bài toán yêu cầu một AI có thể di chuyển các nước đi trên bàn cờ để có thể tương tác với người chơi, đồng thời có thể xác định các nước đi thông minh để tối ưu và tìm ra cách chiến thắng ván cờ.

Trong đề tài này, nhóm sẽ nghiên cứu để ứng dụng các giải thuật tìm kiếm cơ bản và tìm kiếm bổ sung tri thức, đồng thời cải tiến thuật toán để áp dụng vào AI.

**Thiết kế giao diện trò chơi**

**1. Giao diện bàn cờ:**

+ Bàn cờ được thiết kế với kiểu dữ liệu list, mỗi phần tử sẽ gồm 2 kí tự để biểu diễn với kí tự chứa b đại diện cho quân đen, chứa w đại diện cho quân trắng:

- Quân tốt: bp, wp

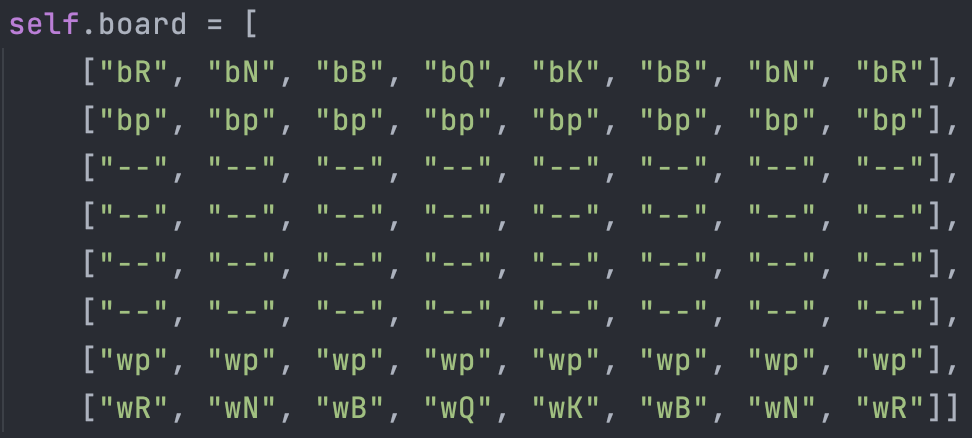
- Quân xe: bR, wR

- Quân mã: bN, wN

- Quân tượng: bB, wB

- Quân hậu: bQ, wQ

- Quân vua: bK, wK



+ Các quân cờ này sẽ được load phần ảnh thông qua hàm loadImages().

+ Bàn cờ được thiết kế bằng hàm drawBoard() : Gồm vẽ bàn cờ, các quân cờ, thanh lợi thế, …

**2. Thiết kế các nước đi cho từng quân cờ:**

+ Các quân cờ trên bàn cờ sẽ có các hàm riêng biệt để thực hiện các nước đi của mỗi loại như: getPawnMoves() dành cho quân tốt, getRookMoves() cho quân xe,...

+ Bàn cờ được thiết kế có đầy đủ các chức năng như: Nhập thành, Bắt tốt qua đường, Tốt phong cấp, Tính năng Undo move, Restart game,...

# 

# **GIẢI THUẬT ĐỀ XUẤT**

Phần này sẽ trình bày về giới thiệu về các kiến thức liên quan và đề xuất của nhóm để giải quyết bài toán.

**1. Xây dựng hàm đánh giá lợi thế:**

+ Ở đây, mỗi quân cờ sẽ mang một trọng số giá riêng (tham khảo theo chuẩn quốc tế). Qua nhiều lần kiểm thử với các trọng số khác nhau. Ví dụ: quân tốt mang các giá trị 0.5, 1.0, 1.5,... ; quân mã mang các giá trị 1.0, 2.0, 3.0, 4.0,... Ta rút ra trọng số của các quân cờ gần sát với chuẩn quốc tế.

:1.0 , :3.0 , Ảnh có chứa cây kéo, chìa khóa

Mô tả được tạo tự động :3.0 , Ảnh có chứa văn bản, bàn

Mô tả được tạo tự động:5.0 , Ảnh có chứa văn bản

Mô tả được tạo tự động: 9.0

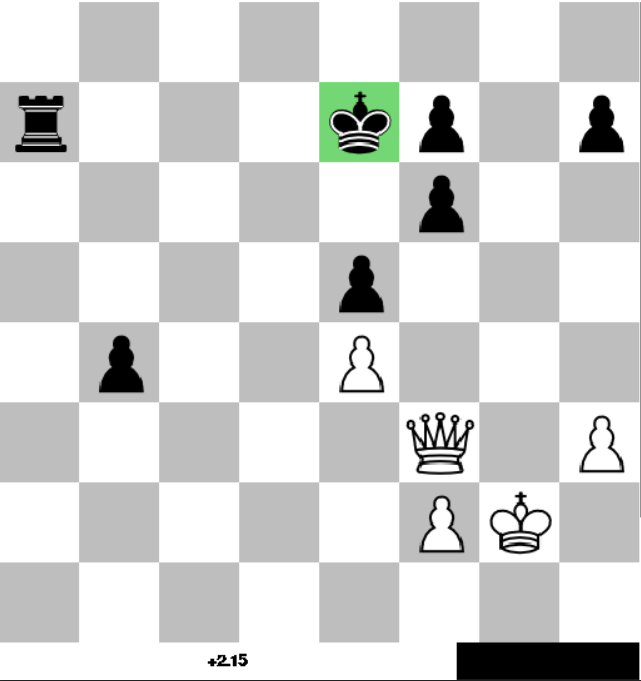


+ Tiếp theo, chúng ta xây dựng trọng số các quân cờ ứng với từng vị trí của nó trên bàn cờ: với mỗi quân cờ, tùy thuộc vào đặc tính của từng quân thì các vị trí sẽ có các lợi thế khác nhau.

Ví dụ: Quân mã ở trung tâm sẽ kiểm soát được 8 ô xung quanh (rất mạnh), ngược lại nếu nó ở biên hoặc ở góc bàn cờ thì sẽ rất yếu. Hoặc quân xe ở hàng ngang số 8 sẽ rất mạnh vì sẽ khóa vua đối phương ở hàng ngang số 8.

+Từ đó, chúng ta sẽ xây dựng hàm đánh giá lợi thế:

Ví dụ :



+ Nếu Evaluation > 0 thì bên trắng đang có lợi thế, ngược lại nếu Evaluation < 0 thì bên đen đang có lợi thế. Nếu hàm này bằng 0 thì lợi thế hai bên đang ngang nhau.

Ở đây phần giao diện sẽ có thanh hiển thị trực quan lợi thế của các bên.



**2. Xây dựng thuật toán AI:**

+ Xây dựng hàm getValidMoves: sinh ra tất cả các nước đi

Cụ thể, chúng ta sẽ duyệt 1 lượt tất cả các quân cờ đang có trên bàn cờ, sau đó xem xét các điều kiện và sinh ra được các nước đi khả thi.

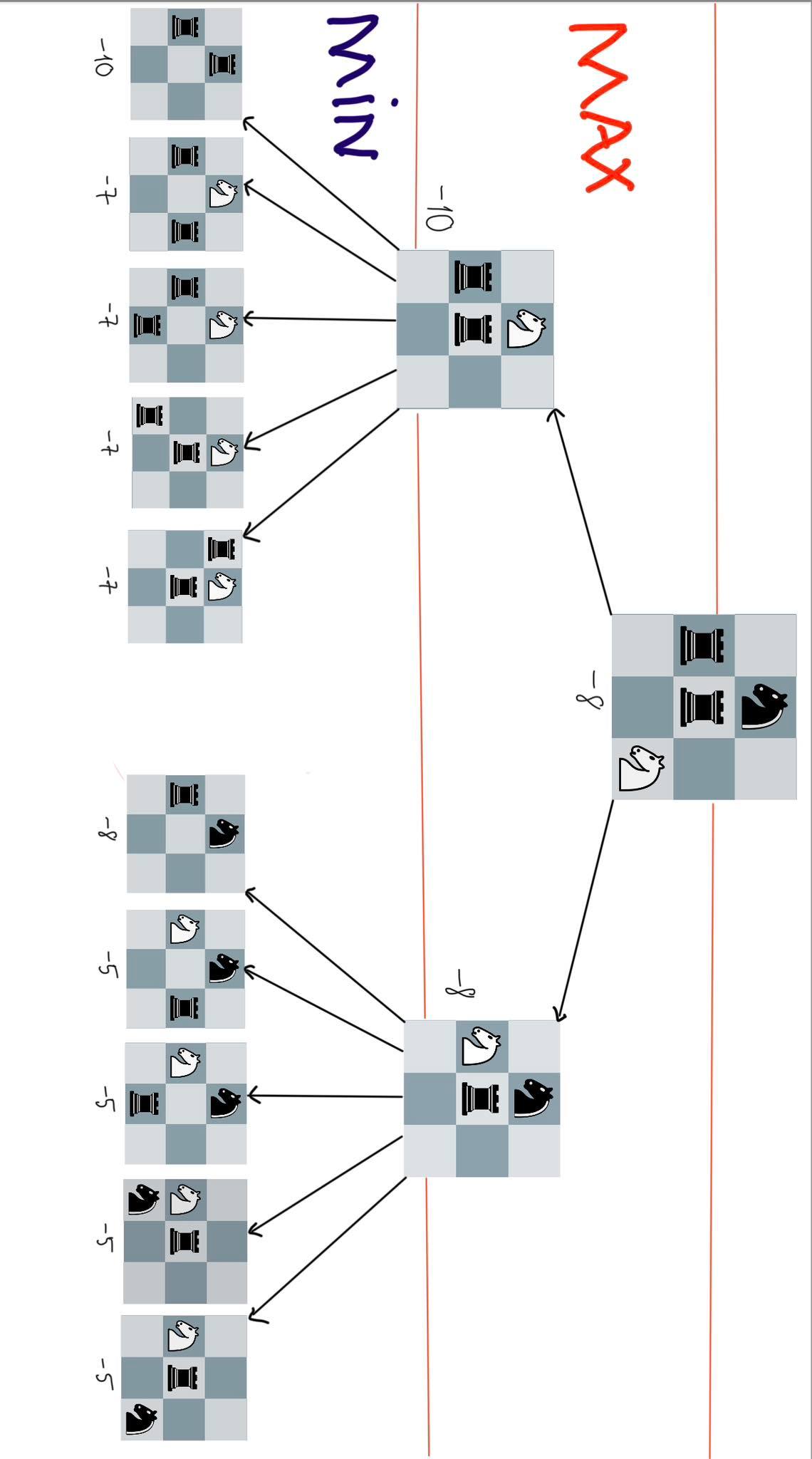
+ Tiếp theo, chúng ta xây dựng thuật toán Minimax: sinh ra cây tìm kiếm tất cả các nước có thể đi được với độ sâu tùy chỉnh, từ đó chọn nước đi có lợi thế lớn nhất.

Ví dụ:

+ Ở tầng dưới cùng (MIN), quân đen sẽ cố gắng làm giảm Evaluation.

+ Ở tầng trên (MAX) , quân trắng sẽ cố gắng làm tăng Evaluation.

Vậy nước tối ưu sẽ là Mã trắng ăn Xe đen, Xe đen ăn Mã Trắng.



Code: (MiniMax)

|  |
| --- |
| def MiniMax(game\_state, valid\_moves, depth, turn\_multiplier):  global next\_moveMiniMax  global number\_of\_nodes  if depth == 0:  return turn\_multiplier \* scoreBoard(game\_state)  max\_score = -CHECKMATE  for move in valid\_moves:  game\_state.makeMove(move)  next\_moves = game\_state.getValidMoves()  score = -MiniMax(game\_state, next\_moves, depth -1, -turn\_multiplier)  number\_of\_nodes += 1  if score > max\_score:  max\_score = score  if depth == DEPTH:  next\_moveMiniMax = move  game\_state.undoMove()  return max\_score |

+Tuy nhiên, nếu trung bình một node có B nodes con và thực hiện MiniMax với độ sâu D thì độ phức tạp thuật toán lên tới B^D là rất rất lớn.

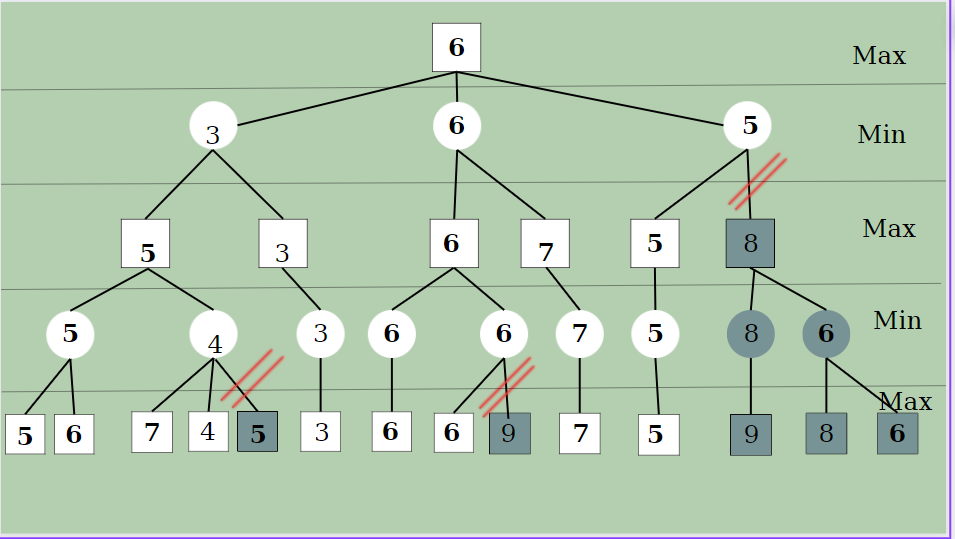
+Vì vậy để giảm bớt các nhánh dư thừa, chúng ta sẽ cải tiến bằng phương pháp cắt tỉa Alpha-Beta bằng cách loại bỏ các nhánh thừa trên cây tìm kiếm.

Alpha là lựa chọn tốt nhất cho đến hiện tại trên đường về gốc của Maximizer ( Quân trắng)

Beta là lựa chọn tốt nhất cho đến hiện tại trên đường về gốc của Minimizer (Quân đen)

Ví dụ: Với cây tìm kiếm dưới đây:

Chúng ta đã cắt tỉa được 3 nhánh dư thừa, với không gian tìm kiếm lớn hơn thì thuật toán sẽ càng cắt tỉa được nhiều hơn.



Code : (Cắt tỉa alpha-beta)

|  |
| --- |
| def AlphaBeta(game\_state, valid\_moves, depth, alpha, beta, turn\_multiplier):  global next\_move  global number\_of\_nodes  if depth == 0:  return turn\_multiplier \* scoreBoard(game\_state)  max\_score = -CHECKMATE  for move in valid\_moves:  game\_state.makeMove(move)  next\_moves = game\_state.getValidMoves()  score = -AlphaBeta(game\_state, next\_moves, depth - 1, -beta, -alpha, -turn\_multiplier)  number\_of\_nodes += 1  if score > max\_score:  max\_score = score  if depth == DEPTH:  next\_move = move  game\_state.undoMove()  if max\_score > alpha:  alpha = max\_score  if alpha >= beta:  break  return max\_score |

+ Để tiếp tục cải tiến Thuật toán Alpha-Beta, ta sử dụng Move Ordering.: Để nhanh chóng tìm được nước đi tối ưu, chúng ta ưu tiên các nước như: nước ăn quân, nước chiếu vua, nước nhập thành,... Đây là những nước đi có nước có tỉ lệ cao mang lại lợi thế hơn so với cắt tỉa Alpha-Beta.

Code : (Move Ordering)

Tương tự như cắt tỉa Alpha-Beta nhưng xét các node theo thứ tự trên bằng hàm move\_ordering:

|  |
| --- |
| def move\_ordering(valid\_moves):  move\_orders = []  for move in valid\_moves:  if move.is\_castle\_move:  move\_orders = [move] + move\_orders  elif move.is\_capture:  move\_orders = [move] + move\_orders  else:  move\_orders.append(move)  return move\_orders |

# **THỰC NGHIỆM**

**1. Thực nghiệm hàm đánh giá:**

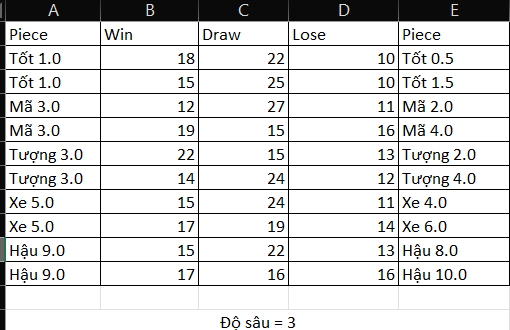
+ Đánh giá việc sử dụng bảng trọng số và khi không sử dụng:

Waterfall chart

Description automatically generated with low confidence

Vậy việc sử dụng bảng sẽ hiệu quả hơn.

+ Đánh giá khi thay đổi giá trị trọng số của từng quân cờ: (mỗi quân cờ được thay bằng 1 giá trị mới và cho thi đấu với giá trị chuẩn 50 ván và rút ra được kết quả sau)



|  |  |  |
| --- | --- | --- |
| Quân | Trọng số nhỏ (%) | Trọng số lớn (%) |
| Quân tốt (0.5, 1.5) | 35 | 41 |
| Quân mã (2.0, 4.0) | 40 | 48 |
| Quân tượng (2.0, 4.0) | 37 | 47 |
| Quân xe (4.0, 6.0) | 43 | 46 |
| Quân hậu (8.0, 10.0) | 47 | 48 |

Kết luận: Trọng số chuẩn có kết quả lợi thế hơn so với khi thay đổi.

**-> Dùng trọng số chuẩn để thực hiện thuật toán.**

**2. Thực nghiệm thuật toán:**

+ Các giải thuật tìm kiếm được thử nghiệm để kiểm tra hiệu quả.

**Các thế cờ được thử nghiệm:**

**A picture containing diagram

Description automatically generated**

**Thế cờ số 1**

**(Đen đi, nước đi tối ưu là Tốt b4 ăn Mã c3)**

**Square

Description automatically generated with low confidence**

**Thế cờ số 2**

**(Đen đi, nước đi tối ưu là Tốt e5 đi e6 chiếu Vua bắt Hậu)**

**Bảng 1: So sánh không gian tìm kiếm (số node phải duyệt qua) của 3 thuật toán với thế cờ số 1:**

Table

Description automatically generatedChart, line chart

Description automatically generated

**Thế cờ số 1**

**Bảng 2: So sánh không gian tìm kiếm (số node phải duyệt qua) của 3 thuật toán với thế cờ số 2:**

**Graphical user interface

Description automatically generated with medium confidence**

# **KẾT LUẬN**

Trong đề tài này, nhóm đã thực hiện tìm hiểu và nghiên cứu các giải thuật tìm kiếm, từ đó ứng dụng vào việc giải quyết bài toán AI chơi cờ vua đơn giản. Ngoài ra, nhóm cũng tìm hiểu và ứng dụng Pygame cũng như ngôn ngữ lập trình Python để xây dựng ứng dụng và triển khai các thuật toán.

**Bảng phân công công việc**

|  |  |
| --- | --- |
| Thành viên | Nhiệm vụ |
| : Nguyễn Minh An | Xây dựng giao diện bàn cờ  Viết báo cáo |
| Ng Lê Duy Anh | Xây dựng các cải tiến thuật toán  Thuyết trình |
| Trần Xuân Bách | Xây dựng các nước đi  Tìm hiểu và xây dựng thuật toán Minimax |
| Hoàng Đức Thành | Làm PowerPoint  Kiểm thử chương trình |

# 

# **TÀI LIỆU THAM KHẢO**

[1] PGS.TS Thân Quang Khoát, slide bài giảng Nhập môn Trí tuệ nhân tạo.

[2] Youtuber: Eddie Sharick

**-THE END-**