

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

**KHOA CÔNG NGHỆ THÔNG TIN**

------🙙🕮🙛-------



**BÁO CÁO THÍ NGHIỆM/THỰC NGHIỆM**

Học phần: **Trí tuệ nhân tạo**

**Đề tài**: **Thuật toán Minimax, Alpha-beta và ứng dụng trong trò chơi cờ Caro**

Giáo viên hướng dẫn : ThS. Nguyễn Lan Anh

Nhóm sinh viên thực hiện :

1. Nguyễn Văn Huấn - 2020600491

2. Nguyễn Văn Hưng - 2020605939

3. Vũ Khắc Huy Linh - 2020600294

Lớp: 20221IT6043007 Khóa: K15

Nhóm: 14

**Hà Nội - Năm 2022**

**MỤC LỤC**

[**LỜI NÓI ĐẦU** 4](#_Toc122272702)

[**CHƯƠNG 1: GIỚI THIỆU VỀ ĐỀ TÀI** 5](#_Toc122272703)

[**1.1** **Đặt vấn đề** 5](#_Toc122272704)

[**1.2 Mục tiêu của đề tài** 5](#_Toc122272705)

[**1.3 Phạm vi nghiên cứu** 5](#_Toc122272706)

[**1.4.Ý nghĩa thực tiễn của đề tài** 6](#_Toc122272707)

[**CHƯƠNG 2: TỔNG QUAN VỀ VẤN ĐỀ TÌM KIẾM** 7](#_Toc122272708)

[**2.1.Bài toán tìm kiếm và không gian trạng thái** 7](#_Toc122272709)

[**2.2.Cây trò chơi** 7](#_Toc122272710)

[**CHƯƠNG 3: THUẬT TOÁN TÌM KIẾM MINIMAX** 10](#_Toc122272711)

[**3.1 Giới thiệu** 10](#_Toc122272712)

[**3.1.1 Định lý Minimax** 10](#_Toc122272713)

[**3.2 Giải thuật Minimax** 11](#_Toc122272714)

[**3.2.1 Ý tưởng** 12](#_Toc122272715)

[**3.2.2 Áp dụng giải thuật Minimax đến độ sâu lớp cố định** 14](#_Toc122272716)

[**3.2.3 Thủ tục Minimax** 16](#_Toc122272717)

[**3.2.4 Đánh giá** 19](#_Toc122272718)

[**CHƯƠNG 4: THUẬT TOÁN CẢI TIẾN ALPHA-BETA** 21](#_Toc122272719)

[**4.1 Nhận định** 21](#_Toc122272720)

[**4.2 Ý tưởng** 22](#_Toc122272721)

[**4.3 Giải thuật** 23](#_Toc122272722)

[**4.4 Đánh giá** 25](#_Toc122272723)

[**4.5 So sánh giải thuật Minimax và giải thuật Alpha-beta.** 28](#_Toc122272724)

[CHƯƠNG 5: ỨNG DỤNG 30](#_Toc122272725)

[**5.1 Giới thiệu về trò chơi cờ caro** 30](#_Toc122272726)

[**5.2 Các quy định khi chơi cờ Caro** 30](#_Toc122272727)

[**5.2.1 Bàn cờ** 30](#_Toc122272728)

[**5.2.2 Quân cờ** 31](#_Toc122272729)

[**5.2.3 Di chuyển** 31](#_Toc122272730)

[**5.2.4. Kết thúc ván cờ** 31](#_Toc122272731)

[**5.3. Chế độ người với máy trong trò chơi cờ Caro** 31](#_Toc122272732)

[**5.4. Sơ đồ thuật toán** 31](#_Toc122272733)

[**5.5 Giao diện trò chơi** 32](#_Toc122272734)

[**KẾT LUẬN** 38](#_Toc122272735)

[**TÀI LIỆU THAM KHẢO** 39](#_Toc122272736)

**LỜI NÓI ĐẦU**

AI(Artificial Intelligent) hay còn gọi là trí tuệ nhân tạo đã mang lại rất nhiều lợi ích, tiện ích. Nghiên cứu, tạo lập cho các máy tính có khả năng “suy nghĩ”, thậm chí trong một số phạm vi hẹp nào đó, có thể cạnh tranh hoặc vượt quá khả năng bộ não của con người. Chơi trò chơi là một ví dụ điểm hình trong những lĩnh vực phát triển mạnh mẽ nhất. Đặc biệt là lý thuyết trò chơi đã thu hút được sự chú ý của các nhà khoa học máy tính do ứng dụng của nó trong Trí tuệ nhân tạo và Điều khiển khoa học.

Trí tuệ nhân tạo đã vận dụng lý thuyết trò chơi để nghiên cứu về các trò chơi đối kháng và thiết kế chương trình chơi cờ caro giữa Người và Máy tính. Do bùng nổ tổ hợp quá lớn của cây trò chơi mà cả người mà máy không thể có thể tìm kiếm vét cạn. Do đó phương pháp tìm kiếm duy nhất là chỉ tìm kiếm một độ sâu giới hạn nào đó và chọn nước đi dẫn đến một thế cờ có lợi nhất cho mình. Do phải tính cả khả năng chống trả của đối phương nên ta không dùng được các thuật toán tìm kiếm thông thường mà phải dùng thuật toán tìm kiếm riêng cho cây trò chơi, đó là thuật toán chiến lược Minimax và thuật toán Alpha-beta.

Lời đầu tiên , chúng em xin giành lời cảm ơn chân thành và lòng biết ơn sâu sắc nhất đến giảng viên Nguyễn Lan Anh, người đã giúp chúng em gợi mở ra những ý tưởng sâu sắc, hiệu quả về môn trí tuệ nhân tạo và cô đã tận tận tình tạo điều kiện giúp đỡ chúng em trong bộ môn trìu tượng này để có thể hoàn thành nhiệm vụ nghiên cứu và phát triển dự án này.

Bài báo cáo này được thực hiện trong thời gian ngắn, trong quá trình soạn thảo không thể tránh những sai sót, chúng em những người thực hiện dự án mong cô có thể đóng góp ý kiến để chúng em có thể thực hiện sửa chữa và hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

# **CHƯƠNG 1: GIỚI THIỆU VỀ ĐỀ TÀI**

* 1. **Đặt vấn đề**

Trong nhiều vấn đề tìm kiếm dựa trên tri thức ta thường đề cập đến các đối tượng khác nhau và mỗi đối tượng có cấu trúc bao gồm một số thuộc tính với những quan hệ nhất định. Một cách tổng quát, có thể hiểu là tìm một đối tượng thỏa mãn một số đòi hỏi nào đó, trong một tập hợp rộng lớn các đối tượng.

Ví dụ: Các trò chơi, chẳng hạn cờ vua, cờ caro có thể xem như vấn đề tìm kiếm. Trong số rất nhiều nước đi được phép thực hiện, ta phải tìm ra các nước đi dẫn tới tình thế kết cuộc mà ta là người thắng. Chứng minh định lý cũng có thể xem như vấn đề tìm kiếm. Cho một tập các tiên đề và các luật suy diễn, trong trường hợp này mục tiêu của ta là tìm ra một chứng minh (một dãy các luật suy diễn được áp dụng) để được đưa đến công thức mà ta cần chứng minh. Trong các lĩnh vực nghiên cứu của Biểu diễn tri thức và ứng dụng, em thường xuyên phải đối đầu với vấn đề tìm kiếm. Đặc biệt trong lập kế hoạch và học máy, tìm kiếm đóng vai trò quan trọng. Nghiên cứu máy tính chơi cờ đã xuất hiện rất sớm. Không lâu sau khi máy tính lập trình được ra đời vào năm 1950, Claude Shannon đã viết chương trình chơi cờ đầu tiên. Các nhà nghiên cứu Công nghệ và Biểu diễn tri thức đã nghiên cứu việc chơi cờ, vì rằng máy tính chơi cờ là một bằng chứng rõ ràng về khả năng máy tính có thể làm được các công việc đòi hỏi trí thông minh của con người.

## **1.2 Mục tiêu của đề tài**

Nghiên cứu các kỹ thuật suy luận và tìm kiếm trong gói giải bài tập trí tuệ nhân tạo sách Trí tuệ nhân tạo dành cho cử nhân ngành công nghệ thông tin. Từ đó có thể cập nhật thêm và cải tiến nhất định để có phương pháp tư duy, cải tiến trong các chiến lược:  
+ Chiến lược tìm kiếm nước đi Minimax.  
+ Phương pháp cắt tỉa α-β, một kỹ thuật để tăng hiệu quả của tìm kiếm Minimax.

## **1.3 Phạm vi nghiên cứu**

Về mặt kỹ thuật, đề tài chỉ tập trung nghiên cứu :  
- Các kỹ thuật tìm kiếm mù đó là xem xét theo một hệ thống nào đó tất cả các đối tượng để phát hiện ra đối tượng cần tìm.  
- Các kỹ thuật tìm kiếm kinh nghiệm (tìm kiếm heuristic) trong đó dựa vào kinh nghiệm và sự hiểu biết của về vấn đề cần giải quyết để xây dựng nên hàm đánh giá hướng dẫn sự tìm kiếm.  
- Các kỹ thuật tìm kiếm tối ưu.  
- Các phương pháp tìm kiếm có đối thủ, tức là các chiến lược tìm kiếm nước đi trong các trò chơi hai người, chẳng hạn cờ vua, cờ tướng, cờ caro.  
Về miền tri thức, đề tài chỉ tập trung chủ yếu vào các bài tập trí tuệ nhân tạo trong sách Trí tuệ nhân tạo dành cho cử nhân ngành công nghệ thông tin.

## **1.4.Ý nghĩa thực tiễn của đề tài**

Đối với giáo viên và sinh viên đề tài giúp hỗ trợ thiết thực hơn cho các giáo viên và các sinh viên thêm tri thức về các phương pháp tìm kiếm.  
Đối với bản thân: đề tài giúp người làm hiểu rõ hơn phương pháp tìm kiếm và cơ chế suy diễn của các chiến lược tìm kiếm.

**CHƯƠNG 2: TỔNG QUAN VỀ VẤN ĐỀ TÌM KIẾM**

**2.1.Bài toán tìm kiếm và không gian trạng thái**

Vấn đề chơi cờ có thể xem như vấn đề tìm kiếm không gian trạng thái. Mỗi trạng thái là một tình thế ( sự bố trí các quân trên bàn cờ ).

+ Trạng thái ban đầu là sự sắp xếp các quân cờ của hai bên lúc bắt đầu cuộc chơi.

+ Các bài toán tử là các bước đi hợp lệ.

+ Các trạng thái kết thúc là các trạng thái kết thúc là các tình thế cuộc chơi dừng, thường được xác định bởi một số điều kiện dừng nào đó.

+ Một hàm kết cuộc (payoff function) ứng mỗi trạng thái kết thúc với một giá trị nào đó. Chẳng hạn như cờ vua, mỗi trạng thái kết thúc là thắng, hoặc thua đối với( đối với Trắng) hoặc hòa. Do đó ta có thể , ta có thể xác định hàm kết cuộc là hàm nhận giá trị 1 tại các trạng thái kết thúc là thắng (đối với Trắng ), -1 là là các trạng thái kết thúc là thua (đối với trắng) và ) tại các trạng thái kết thúc hòa. Trong một trò chơi khác , chẳng hạn trò chơi tính điểm , hàm kết cuộc có thể nhận giá trị nguyên trong khoảng [-k,k] với k là số nguyên dương nào đó.

Như vậy vấn đề trắng là tìm một dãy nước đi sao cho xen kẽ với các nước đi của Đen tạo thành một đường đi từ trạng thái ban đầu trới trạng thái kết thúc là thắng cho Trắng.

Để thuận lợi cho việc nghiên cứu các chiến lược chọn nước đi, ta biểu diễn không gian trạng thái trên dưới trò chơi.

**2.2.Cây trò chơi**

Giới thiệu

Cây trò chơi được xây dựng như sau :

* Gốc của cây ứng dụng là trạng thái ban đầu, ví dụ trong một bàn cờ 2 quân Đen và Trắng, ta sẽ gọi đỉnh ứng với trạng thái mà Trắng (Đen) đưa ra nước đi là đỉnh Trắng (Đen). Nếu một đỉnh là Trắng (Đen) ứng với trạng thái u, thì các đỉnh con của nó là tất cả các đỉnh biểu diễn trạng thái v, v nhận được từ u do Trắng (Đen) thực hiện nước đi nào đó. Do đó, trên cùng một mức của cây các đỉnh đều là Trắng hoặc đều là Đen, các lá của cây ứng với các trạng thái kết thúc.

Ví dụ:

Chart

Description automatically generated

Hình1. Cây trò chơi Caro(Tic-tac-toe)

Trước hết muộn tạo ra một bàn cờ hoàn chỉnh dựa vào trí tuệ nhân tạo, ta phải hình thành nên một trạng thái nhất định cho bàn cờ có thể biểu diễn thành một cây tìm kiếm để biểu thị về vấn đề phức tạp của chúng.

Một cây trò chơi bao gồm tất cả các nước đi có thể của hai người chơi và mỗi nút của cây thể hiện một trạng thái bàn cờ sau khi nhận nước cờ đi của người chơi . Từ một nút (Trạng Thái ) có thể lựa chọn nước đi tiếp theo đó, số nước có thể chọn được gọi là hệ số phân nhánh. Độ sâu của cây trò chơi là số lần thay đổi lượt đi của hai người chơi. Như ví dụ trên đã minh họa cây trò chơi của trò chơi đối kháng Tic-Tac-Toe , trò chơi này cực kì đơn giản vì nó được chơi trên không gian 3x3 =9 ô, hai người chơi là X và O từ trạng thái ban đầu bàn cờ chưa có quân cờ nào do tính chất đối xứng của bàn cờ đầu tiên lượt đi dành cho người chơi O có thể đặt quân cờ đầu tiên vào 3 vị trí và hình thành nên ba nhánh .Lượt đi tiếp theo giành cho người chơi X, tại nhánh thứ nhất có 5 khả năng đặt quân cờ, nhánh thứ 2 có 2 khả năng đặt quân cờ, nhánh thứ 3 có 5 khả năng đặt quân cờ. Tương tự như các lượt đi tiếp theo , số lượt đi tối đa của cả hai người chơi là 9. Đối với trò chơi Tic-Tac-Toe , mỗi ô có tối đa ba trạng thái (O,X,trống), số ô của bàn cờ là 9, nên không gian trạng thái bàn cờ trò chơi là 39 = 19,683, lấy log10 = 4. Về số lượng có thể tính như sau 9 vị trí có thể đặt quân đi lượt đi đầu tiên nếu không quan tâm đến tính chất đối xứng của bàn cờ, 8 vị trí có thể đặt quân cho lượt đi cho thứ 2,7 vị trí có thể đặt quân cho lượt đi thứ 3, tương tự như vậy, số lượng cây là 9!=362,880 , lấy log10 =5. Đối với máy tính hiện đại thì thì những trò trơi có không gian tìm kiếm nhỏ như trò chơi cờ Caro thì máy tính có thể vét cạn, và lúc đó chương trình đánh cờ chỉ từ hòa đến thắng vì biết được cách đi tốt nhất theo cách đi của đối phương.

Bảng1. Độ phức tạp của trò chơi đối kháng Caro.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Trò chơi đối kháng | Kích thước | Số trạng thái (log10) | Số cây(log10) | Số nước đi | Khả năng đi trong 1 trạng thái |
| Caro(Tic-Tac-Toe) | 9 | 4 | 5 | 9 | 4 |

Vấn đề chơi cờ có thể xem như vấn đề tìm kiếm không gian trạng thái. Mỗi trạng thái là một tình thế ( sự bố trí các quân trên bàn cờ ).

**CHƯƠNG 3: THUẬT TOÁN TÌM KIẾM MINIMAX**

**3.1 Giới thiệu**

Minimax là giải thuật là một thuật toán đệ quy lựa chọn bước đi kế tiếp trong một trò chơi có hai người bằng cách định giá trị cho các Node trên cây trò chơi sau đó tìm Node có giá trị phù hợp để đi bước tiếp theo.

Một phiên bản của giải thuật áp dụng cho các trò chơi đặc biệt đối với trò chơi cờ caro, khi mà mỗi người chơi có thể thắng, thua, hoặc hòa. Nếu người chơi A có thể thắng trong một nước đi, thì “nước đi tốt nhất ” chính là nước đi để dẫn đến kết quả thắng đó. Nếu người B biết rằng có nước đi mà dẫn đến tình huống người A có thể thắng ngay ở nước đi tiếp theo, trong khi nước đi khác thì sẽ dẫn đến tình huống mà người chơi A chỉ có thể tốt nhất là hòa thì nước đi tốt nhất của người chơi B chính là nước đi sau.

Ta sẽ nắm rõ, thế nào là một nước đi “tốt nhất”. Giải thuật Minimax giúp tìm ra nước đi tốt nhất, bằng cách đi ngược từ cuối trò chơi về đầu. Tại mỗi bước , nó sẽ ước định rằng người A đang cố gắng tối đa hóa cơ hội thắng của người A khi đến phiên anh ta, còn ở nước đi kế tiếp thì người chơi B cố gắng để tối thiểu hóa cơ hội thắng của người A (nghĩa là tối ưu hóa cơ hội thắng của người B).

Lý thuyết trò chơi coi trò chơi là sự kết hợp hoặc trao đổi giữa hai hay nhiều đối thủ ở đó mỗi đối thủ cố gắng lựa chọn tối ưu hành động (hay nước đi) của mình nhằm đạt được lợi ích tối đa.

**3.1.1 Định lý Minimax**

\*Định lý:

Với trò chơi có tổng bằng không và có hai người chơi, một trong 3 điều kiện sau đây sẽ dẫn đến 2 điều kiện còn lại.

1. Tồn tại một cắp cân bằng.

2. **

3.Tồn tại một số thực v, một chiến lược hỗn hợp p\* và một chiến lược hỗn hợp q\* sao cho:

(a)  với c*j= 1, 2,…, n*

*(b)*  với *i= 1, 2,…, m*

Điều kiện (3a) nói rằng tổn thất trung bình cho người chơi 2 dùng chiến lược thuần túy bất kỳ nào không nhỏ hơn v. Tương tự điều kiện (3b) nói tổn thất trung bình của người chơi 1 dùng chiến thuật thuần túy bất kỳ thì không lớn hơn v.

Trong định lý Minimax, von Neumann (1928) chứng minh sự tồn tại tổng quát của các nghiệm Minimax trong chiến lược ngẫu nhiên hóa cho các trò chơi hữu hạn bước, hai người chơi và tổng bằng không. Với các trò chơi này , định lý Minimax tương đương về mặt logic với sự tồn tại của cân bằng Nash.

\* Định lý Minimax còn được phát biểu như sau :

- Với mọi trò chơi có đối kháng với hai người chơi thì luôn tồn tại chiến lược cân bằng.

-Từ đặc điểm của trò chơi có tổng bằng không với hai người chơi và từ định lý Minmax nên thuật toán Minimax thích hợp với loại trò chơi này. Và đảm bảo khi thuật toán ứng dụng cho các trò chơi này sẽ chắc chắn có lời giải.

**3.2 Giải thuật Minimax**

Xét một trò chơi đối kháng trong đó có hai người thay phiên nhau đi nước của mình như cờ vua, cờ tướng, cờ caro,... Trò chơi có một trạng thái bắt đầu và mỗi nước sẽ đi biến đổi trạng thái hiện hình thành một trạng thái mới. Trò chơi sẽ kết thúc theo một quy định nào đó, theo đó thì cuộc chơi sẽ dẫn đến một trạng thái phản ánh có một người thắng cuộc hoặc một trạng thái cả hai đấu thủ không thể phát triển được nước đi của mình, ta gọi nó là trạng thái hòa cờ. Ta tìm cách phân tích xem từ một trạng thái nào đó sẽ dẫn đến đấu thủ nào sẽ thắng với điều kiện cả hai đấu thủ đều có trình độ như nhau.

**3.2.1 Ý tưởng**

Hai đối thủ trong một trò chơi được gọi là MIN và MAX đại diện cho đối thủ quyết giành thắng lợi hay cố gắng tối đa hóa ưu thế của mình. Ngược lại MIN là đối thủ cố gắng tối thiểu hóa điểm số của MAX. Ta giả thiết MIN cũng dùng cùng những thông tin như MAX.

Một trò chơi như vậy có thể biểu diễn bởi một Cây trò chơi. Mỗi một nút của cây biểu diễn cho một trạng thái. Nút gốc biểu diễn cho trạng thái bắt đầu của cuộc chơi. Mỗi nút lá biểu diễn cho một trạng thái kết thúc của trò chơi (trạng thái thắng thua hoặc hòa). Nếu trạng thái x được biểu diễn bởi nút n thì các con của n biểu diễn cho tất cả các trạng thái kết quả của các nước đi có thể xuất phát từ trạng thái x. Do hái đấu thủ luân phiên nhau đi nước của mình nên các mức (lớp) trên cây trò chơi cũng luân phiên nhau là MAX và MIN. Cây trò chơi vì thế còn có tên là cây MIN-MAX trên cây trò chơi các nút ứng với trạng thái mà từ đó người chơi MAX chon nước đi sẽ thuộc lớp MAX, các nút ứng với trạng thái mà từ đó người chơi MIN chọn nước đi sẽ thuộc lớp MIN. Chiến lược minimax thể hiện qua quy tắc định trị cho các nút trên cây trò chơi như sau:

* Nếu nút là nút gán cho nút đó một giá trị để phản ánh trạng thái thắng hay thua hay hòa của các đấu thủ.
* Sử dụng giá trị của các nút lá để xác định giá trị các nút ở các mức trên trong cây trò chơi theo quy tắc:

+ Nút thuộc lớp MAX thì gán cho nó giá trị lớn nhất của các nút con của nút đó.

+ Nút thuộc lớp MIN thì gán cho nó giá trị nhỏ nhất của các nút con của nút đó.

Giá trị được gán cho từng trạng thái theo quy tắc trên chỉ rõ ra giá trị của trạng thái tốt nhất mà mỗi đối thủ có thể hy vọng đạt được. Người chơi sẽ sử dụng các giá trị này để lựa chọn các nước đi cho mình. Đối với người chơi MAX khi đến lượt đi, người chơi này sẽ chọn nước đi ứng với trạng thái có giá trị cao nhất trong các trạng thái con, còn người chơi MIN khi đến lượt sẽ chon nước đi ứng với trạng thái có giá trị nhỏ nhất trong các thái con.

Diagram

Description automatically generated

Hình 2: Minh họa cây trò chơi

Trong cây trò chơi trên, các nút lá được tô nền và viền khung đôi để dễ phân biệt với các nút khác. Ta có thể gán cho mỗi nút lá một giá trị để phản ánh trạng thái thắng thua hay hòa của các đấu thủ. Chẳng hạn ta gán cho nút lá các giá trị như sau :

1 Nếu tại đó người đi X đã thắng

-1 nếu tại đó người đi X đã thua

0 nếu tại đó người đi X hòa nhau

Như vậy từ một trạng thái bất kì, đến lượt mình, người đi X sẽ chọn cho mình một nước đi sao cho dẫn đến trạng thái có giá trị lớn nhất (trong trường hợp này là 1). Ta nói X chọn nước đi MAX, nút mà từ đó X chọn nước đi của mình được gọi là nút MAX. Người đi O đến lượt mình sẽ chọn một nước đi sao cho dẫn đến trạng thái giá trị nhỏ nhất (trong trường hợp này là -1, khi đó X sẽ thua và do đó O sẽ thắng ). Ta nói O chọn nước đi MIN, nút mà từ đó O chọn nước đi của mình được gọi là nút MIN. Áp dụng chiến lược Minimax cho một nhánh trong cây trò chơi của trò chơi Tic-tac-toe ta có giá trị (phái trên mỗi nút) của các nút có thể hiện trong hình trên.

**3.2.2 Áp dụng giải thuật Minimax đến độ sâu lớp cố định**

Khi áp dụng Minimax cho các trò chơi phức tạp, hiếm khi có khả năng mở rộng đồ thị không gian trạng thái đến các nút lá. Thay vào đó không gian trạng thái này chỉ có thể được triển khai đến một số mức xác định phụ thuộc tiềm năng về thời gian và bộ nhớ chẳng hạn. Chiến lược này được gọi là tính trước n nước đi (n-move lookahead). Vì giá trị các nút trong đồ thị con này không phải là trạng thái kết thúc của trò chơi nên chúng không phản ánh giá trị thắng hay thua cuộc. Chúng chit có thể gán một giá trị phù hợp với một hàm đánh giá heuristic nào đó. Giá trị được truyền ngược về nút gốc không cung cấp thông tin thắng cuộc hay thua cuộc mà chỉ là giá trị heuristic của trạng thái tốt nhất có thể tiếp cận sau n nước đi kể từ nút xuất phát. Việc tính trước này sẽ làm tăng hiệu quả của heuristic vì nó được áp dụng vào một phạm vi lớn hơn trong không gian trạng thái. Minimax sẽ hợp nhất tất cả các giá trị của các nút con của một trạng thái thành một giá trị duy nhất cho trạng thái đó.

Trong các cây trò chơi được tìm kiếm bằng mức hay lớp(ply), MAX và MIN luân phiên nhau chọn các nước đi. Mỗi nước đi của một đối thủ sẽ các định một lớp mới trên cây. Các chương trình trò chơi nói chung đều dự tính trước một độ sâu lớp cố định (thường được xác định bằng các giới hạn về không gian hoặc thời gian của máy tính). Các trạng thái trên mức đó được đánh giá theo các heuristic và các giá trị này sẽ đuợc truyền ngược lên bằng thủ tục Minimax, sau đó thuật toán tìm kiếm sẽ dùng các giá trị vừa nhận được để lựa chọn một nước trong số các nước đi kết tiếp. Bằng cách tối đa hóa cho các cha mẹ MAX và tối thiểu hóa cho các cha mẹ MIN, những giá trị này đi lùi theo đồ thị đến con của trạng thái hiện hành. Sau đó trạng thái hiện hành dùng chúng để tiến hành lựa chọn trong các con của nó. Dưới hình 2.3 :

Chart, line chart

Description automatically generated

Diagram

Description automatically generatedỞ đây sử dụng một heuristic phức tạp hơn, nó cố đo mức độ tranh chấp trong trò chơi. Heuristic chọn một trạng thái cần đo, tính tất cả các đường thắng mở ra cho MAX, rồi trừ đi tổng số các đường thắng mở ra cho MIN. Giải thuật tìm kiếm sẽ cố gắng tối đa hóa sự chênh lệch (hiệu số) đó. Nếu có một trạng thái bắt buộc thắng cuộc cho MAX, nó sẽ được đánh giá là +∞,còn với trạng thái bắt buộc thắng cuộc cho MIN thì được đánh giá là -∞.

**3.2.3 Thủ tục Minimax**

Giả sử chúng ta có một bộ phận phân tích thế cờ có thể áp dụng tất cả các luật, các phương pháp đánh cờ khác nhau vào từng thế cờ và chuyển đổi chúng thành một con số đại diện (cho điểm thế cờ ). Mặt khác, ta giả sử con số đó là dương khi áp dụng cho thế cờ của một đấu thủ (được gọi là người chơi cực đại) và là âm khi áp dụng cho đấu thủ bên kia (được gọi là người chơi cực tiểu). Quá trình tính toán cho điểm thế cờ được gọi là lượng giá tĩnh (static evaluation). Hàm thực hiện việc tính toán được gọi là một bộ lượng giá tĩnh và giá trị nhận được gọi là điểm lượng giá tĩnh.

Cả hai đấu thủ đều cố gắng đi như thế nào đó để đạt được điểm tuyệt đối lớn nhất. Người chơi cực đại (MAX) sẽ tìm những nước đi dẫn đến điểm của mình trở nên lớn hơn (hay cao nhất có thể được) hay điểm của đối bớt âm hơn (nhỏ hơn về giá trị tuyệt đối). Còn đấu thủ của anh ta , người chơi cực tiểu, lại ra sức phản kháng lại, để dẫn tới điểm âm của anh ta âm hơn hay dương của đối thủ nhỏ đi (hình 2.3).

Độ sâu xem xét

Đường tìm kiếm- chuỗi nước đi dẫn đến thế cờ tốt nhất có thể đạt được

**…**

Các bàn cờ trung gian, phải biến đổi qua chúng trong quá trình đạt đến các bàn cờ đích, không cần so sánh chúng.

Nước chọn đi

Các bàn cờ đích, ta phải so sánh chúng với nhau để cân nhắc hơn thiệt do nước đi mang lại.

Hình 2.5: Minh họa chiến lược chơi cờ người lẫn máy.

Các nút trắng là các thế cờ trung gian phải trải qua để đến được các thế cờ đích. Không cần phải xét đến độ tốt xấu của các thế cờ trung gian. Các nút đen là các thế cờ đích và phải xem xét độ tốt xấu để so sánh chúng với nhau. Từ đó sẽ tìm ra đường đi để đến thế cờ tốt nhất có thể được( chú ý không là thế cờ tốt nhất trong số đó do phải xem xét các cách đi chống trả của đối phương).

\*Từ ý tưởng ta có thể suy ra các bước của thuật toán Minimax như sau:

* Nếu như đạt đến giới hạn tìm kiếm (đến tầng dưới cùng của cây tìm kiếm tức nút lá), tính giá trị tĩnh của thế cờ hiện tại ứng với người chơi ở đó. Ghi nhớ kết quả .
* Nếu như mức đang xét là người chơi cực tiểu( nút MIN), áp dụng thủ tục Minimax này cho các con của nó. Ghi nhớ kết quả nhỏ nhất.
* Nếu như mức đang xét là người chơi cực đại (nút MAX), áp dụng thủ tục này cho các con của nó. Ghi nhớ kết quả lớn nhất.

\*Từ ý tưởng phân tích trên ta có thể xây dựng tủ tục Minimax sau:

* Hàm Minimax nhận vào một thế cờ pos và trả về giá trị của thế cờ đó.
* Nếu cờ pos tương ứng với nút lá trong cây trò chơi thì trả về giá trị được gắn cho nút lá. Ngược lại pos một giá trị tạm value là -∞ hoặc ∞ tùy thuộc pos là nút MAX hay MIN và xét các thế cờ con của pos. Sau khi một con của pos có giá trị V thì đặt lại value = max( value, V) nếu n là nút MAX và value =min(value, V) nếu n là nút MIN. Khi tất cả các con của n đã được xét thì giá trị tạm value của pos trở thành giá trị của nó.

*Ta có giả mã cho giải thuật Minimax như sau :*

Text, letter

Description automatically generated

* Trong cài đặt , bần cờ được biểu diễn bằng các biến toàn cục. Do đó thay cho truyền tham số là một bàn cờ mới pos vào thủ tục Minimax thì người ta biến đổi luôn biến toàn cục này nhờ thực hiện nước đi “thử”( nước đi dẫn đến bàn cờ mới pos). Sau khi Minimax thực hiện việc tính toán dựa vào bàn cờ lưu ở biến toàn cục thì thuật toán sẽ dùng một số thủ tục để loại bỏ nước đi này.

*Như vậy Minmax bỏ các tham số pos như sau:*

Text, letter

Description automatically generated

Minimax := best;   
end;   
end;

### **3.2.4 Đánh giá**

Thuật toán Minimax thăm toàn bộ cây trò chơi bằng việc dùng chiến lược tìm kiếm theo chiều chiều sâu. Nên độ phức tạp của thuật toán này tương ứng trực tiếp với kích thước không gian tìm kiếm *bd* , trong đó b là hệ số phân nhánh của cây hay chính là nước đi hợp pháp tại mỗi đêm, d là độ sâu tối đa của cây. Thuật toán sẽ thăm tất cả các nút không chỉ là nút lá vì vậy số lượng các nút được thăm sẽ là *b(bd-1)/(b-1)*. Nhưng hàm lượng giá sẽ là phương thức chi phối hầu hết thời gian và chỉ làm việc trên các nút lá, vì vậy việc đòi hỏi không gian bộ nhớ của nó chỉ tuyến tính với d và b. Vì thế độ phức tạo không gian là O(bd).

Nếu hệ số nhánh trung bình của cây là b=40, và tìm kiếm đến độ sâu d=4 (các con số thường gặp trong trò chơi cờ) thì số nút phải lượng giá là 404 = 2560000 ( trên 2 triệu rưỡi nút ). Còn với b=40, d=5 thì số nút phải lượng giá sẽ tăng 40 lần nữa thành 405 = 102400000 (trên 102 triệu nút ), đây là con số tương đối lớn.

Có thể tiết kiệm được nhiều thời gian bằng việc dùng các thuật toán tìm kiếm thông minh hơn như thuật toán Alpha-beta, thuật toán này không thăm tất cả các nút lá mà vẫn cho kết quả đúng với thuật toán Minimax. Trong phần tiếp theo ta sẽ xét thuật toán cải tiến này.

# **CHƯƠNG 4: THUẬT TOÁN CẢI TIẾN ALPHA-BETA**

## **4.1 Nhận định**

Thuật toán Minimax yêu cầu phải có sự phân tích qua hai bước đối với không gian tìm kiếm : bước đầu truyền xuống đến độ sâu lớp áp dụng heuristic và bước sau để truyền ngược vào các giá trị trên cây. Minimax lần theo tất cả các nhánh trong không gian bao gồm cả những nhánh mà một thuật toán thông minh hơn có thể bỏ qua hay tỉa bớt. Các nhà nghiên cứu trong lĩnh vực chơi game xây dựng một kỹ thuật tìm kiếm gọi là cắt tỉa Alpha-beta nhằm nâng cao hiệu quả tìm kiếm trong các bài toán trò chơi hai đối thủ.

Bộ đánh giá tĩnh trong thủ tục Minimax cần được thực hiện đối với tất cả các nút tại mức cuối của cây trò chơi ( nút lá). Ta có thể giảm bớt số tính toán tốn kém này bằng cách giảm số nhánh cây cần tạo ra và số các đánh giá tĩnh cần tính ra. Do vậy một giải pháp như đã dùng trong thủ tục nhánh và biên là không tiếp tục đi theo các đường không tốt.

Thuật toán Alpha-beta là một cải tiến của thuật toán Minimax nhằm tỉa bớt nhánh của cây trò chơi, làm giảm số lượng nút phải sinh lượng giá, do đó có thể tăng độ sâu của cây tìm kiếm. Giả sử hình sau là một thế cờ mà hai nút đầu tiên đã được lượng giá. Nếu thực hiện thủ tục Minimax đối với các nút đó sẽ cho thấy người chơi cực đại đã được đảm bảo nếu đi nước bên trái sẽ được ít nhất là 2 điểm dù là các lượng giá của các nút khác cho kết quả như thế nào đi nữa.

Diagram

Description automatically generated

Bây giờ, ta giả sử nút tiếp theo được lượng giá và cho kết quả là 1. Nếu đi vào nhánh này thì đối phương sẽ đảm bảo làm điểm của người chơi cực đại không thể vượt quá giá trị 1 dù là các lượng giá các nút khác cho kết quả như thế nào đi nữa. Do đó đến đây, nước đi tốt nhất là chọn nước đi bên trái với đảm bảo là ít nhất được 2 điểm. Và do đó, hoàn toàn không cần thiết phải lượng giá nút còn lại.

## **4.2 Ý tưởng**

Ý tưởng của tìm kiếm Alpha-beta rất đơn giản. Thay vì nếu như tìm kiếm toàn bộ không gian đến một độ sâu lớp cố định, tìm kiếm Alpha-beta thực hiện theo kiểu tìm kiếm sâu. Có hai giá trị, gọi là alpha và beta được tạo ra trong quá trình tìm kiếm:

* Giá trị alpha liên quan với các nút MAX và có khuynh hướng không bao giờ giảm
* Ngược lại giá trị beta liên quan đến các nút MIN và có khuynh hướng không bao giờ tăng.

Giả sử có giá trị alpha của một nút MAX là 6, MAX không cần phải xem xét giá trị truyền ngược nào nhỏ hơn hoặc bằng 6 có liên quan với một nút MIN nào đó bên dưới. Giá trị alpha là giá trị thấp nhất mà MAX có thể nhận được sau khi cho rằng MIN cũng sẽ nhận giá trị tốt nhất cho nó. Thương tự nếu MIN có giá trị beta là 6 nó cũng không cần xem xét các nút nằm dưới nó có giá trị lớn hơn 6.

Để bắt đầu thuật toán tìm kiếm Alpha-beta, ta di xuống hết độ sâu lớp theo kiểu tìm kiếm sâu, đồng thời áp dụng, đánh giá heuristic cho một trạng thái và tất cả các trạng thái anh em của nó. Giả thuyết tất cả đều là nút MIN, giá trị tối đa của các nút MIN này sẽ được truyền ngược lên cho nút cha mẹ (là nút MAX). Sau đó giá trị này được gán cho ông bà của các nút cha mẹ MIN như là một giá trị beta kết thúc tốt nhất. Tiếp theo thuật toán này sẽ đi xuống các nút cháu khác và kết thúc việc tìm kiếm đối với nút cha mẹ của chúng nếu gặp bất kì một giá trị nào lớn hơn hoặc bằng giá trị beta này. Quá trình này gọi là cắt tỉa beta ( cut). Cách làm tương tự cũng được thực hiện cho việc cắt tỉa Alpha ( cut) đối với các nút cháu của một nút MAX.

* Hai luật cắt tỉa dựa trên các giá trị alpha và beta là :

1. Quá trình tìm kiếm có thể kết thúc bên dưới một nút MIN nào có giá trị beta nhỏ hơn hoặc bằng giá trị alpha của một nút MAX bất kỳ của nó.
2. Quá trình tìm kiếm có thể kết thúc bên dưới một nút MAX nào có giá trị alpha lớn hơn hoặc bằng giá trị beta của một nút cha MIN bất kì của nó.

Việc cắt tỉa Alpha -beta như vậy thể hiện quan hệ giữa các nút ở lớp n và các nút ở lớp n+2 và do quan hệ đó toàn bộ các cây con bắt nguồn ở lớp n+1 đều có thể loại khỏi việc xem xét.

\*Chú ý rằng giá trị truyền ngược thu được hoàn toàn giống như kết quả Minimax đồng thời tiết kiệm được các bước tìm kiếm một cách đáng kể.

*Nguyên tắc alpha-beta:*

* Nếu biết điều đó thật sự tồi thì đừng mất thời gian tìm hiểu nó sẽ tồi tệ đến đâu.
* Ý tưởng này được gọi là nguyên tắc Alpha-beta do nó dùng trong thủ tục Alpha-beta (ta sẽ xét dưới đây). Hai tham số của thủ tục này được gọi là alpha và beta được dùng để theo dõi các triển vọng- chúng cho biết các giá trị nằm ngoài khoảng [alpha, beta] là các điểm “thật sự tồi” và không cần phải xem xét nữa.
* Khoảng[alpha, beta] còn được gọi là cửa sổ alpha, beta. Trong ngữ cảnh cách trò chơi, nguyên tắc Alpha-beta nói rằng, mỗi khi xem xét một nút bất kỳ, nên kiểm tra các thông tin đã biết về các nút cha, ông của nó. Có thể do có đủ thông tin từ cha, ông nên không phải làm bất kì việc gì nữa cho nút này. Do đó, nguyên tắc này cũng giúp chỉnh sửa hoặc xác định chính xác giá trị tại nút cha, ông nó. Như trên nói, một cách để tiện theo dõi quá trình tính toán là dùng các tham số alpha và beta để ghi lại các thông tin theo dõi cần thiết. Thủ tục Alpha-beta được bắt đầu tại nút gốc với giá trị của alpha là -∞ và beta là +∞. Thủ tục sẽ tự đệ quy chính nó với khoảng cách giữa các giá trị alpha và beta ngày càng hẹp hơn.

## **4.3 Giải thuật**

*Thuật toán Alpha-Beta:*

* Nếu mức đang xét là đỉnh (gốc cây) , đặt giá trị alpha là -∞ và beta là +∞.
* Nếu như đạt đến giới hạn tìm kiếm (đến tầng dưới cùng của cây tìm kiếm, nút lá), tính giá trị tĩnh của thế cờ hiện tại ứng với người chơi ở đó. Ghi lại kết quả.
* Nếu như mức đang xét là của người chơi cực tiểu (MIN), thực hiện các công việc sao cho đến khi tất cả các con của nó đã được xét với thủ tục Alpha-beta hoặc cho đến khi alpha là bằng hoặc lớn hơn beta.

Áp dụng thủ tục Alpha-beta với giá trị alpha và beta hiện tại cho một con. Ghi nhớ lại kết quả.

So sánh giá trị ghi nhớ với giá trị beta, nếu giá trị đó nhỏ hơn thì đặt beta bằng giá trị mới này. Ghi nhớ lại beta ( thu hẹp khoảng[alpha, beta] bằng cách giảm giá trị beta).

* Nếu như mức đang xét là của người chơi cực đại (MAX), thực hiện các công việc sau cho đến khi tất cả con của nó đã được xét với thủ tục Alpha-beta hoặc cho đến khi alpha là bằng hoặc lớn hơn beta.

Áp dụng thủ tục Alpha-beta với giá trị alpha và beta hiện tại cho con. Ghi nhớ kết quả.

So sánh giá trị ghi nhớ với giá trị alpha, nếu giá trị đó lớn hơn thì đặt Alpha bằng giá trị mới này. Ghi nhớ lại alpha ( thu hẹp khoảng [alpha,beta] bằng cách tăng giá trị alpha).

Chart, radar chart

Description automatically generated

Hình 2.7 Minh hoạc giải thuật Alpha-beta

A có β=3 ( Giá trị nút A sẽ không lớn hơn 3). B bị cắt tỉa β, vì 5>3 có α=3 ( Giá trị nút C sẽ không nhỏ hơn 3). D bị cắt tỉa α, E bị cắt tỉa α, vì 0<3, E bị cắt tỉa α, vì 2<3. Giá trị nút C là 3.

Từ ý tưởng trên ta sẽ cây dựng hàm Alpha-Beta bằng ngôn ngữ Pascal. Hàm này sẽ có dạng khai báo như dưới đây, trong dod depth là độ sâu tìm kiếm, INFINITY là giá trị vô cùng, thuật toán tính toán dựa trên thế cờ hiện tại pos là các biến toàn cục.

Text, letter

Description automatically generated

Ví dụ lời gọi thủ tục Alpha-Beta đầu tiên với độ sâu tìm kiếm 4 và thế cờ hiện tại pos có dạng như sau:

Alpha-Beta (-INFINITY, +INFINITY,4);

So với thuật toán Minimax thì trong thuật toán Alpha-Beta đã đưa thêm hai biến alpha, beta làm hai mực ngưỡng. Ta thấy cứ mỗi khi best >= beta thì thuật toán không thực hiện tiếp vòng lặp, có nghĩa là nó không chịu mở rộng tiếp những nhánh còn lại nữa. Các nhánh đó đã bị cắt bỏ và do đó ta sẽ tiết kiệm được thời gian. Việc cắt bỏ này hoàn toàn an toàn với những lý do ta đã xét ở trên. Ta thấy rằng mỗi lần hàm này được gọi thì chỉ có tham số beta được dùng để so sánh cắt bỏ, còn tham số alpha không được dùng. Tuy nhiên khi áp dụng cùng thuật toán cho cây con thì ta đã hoán vị hai giá trị alpha, beta cho nhau (và đảo cả dấu), do đó alpha sẽ có tác dụng trong độ sâu sau, rồi độ sâu sau nữa lại đến lượt beta... Nói cách khác, một giá trị chỉ luôn ảnh hưởng đến người chơi cực đại, còn giá trị kia lại luôn ảnh hưởng đến người chơi cực tiểu. Chúng là các ngưỡng giữa các nước đi được chấp nhận và không chấp nhận. Những nước đi cần quan tâm phải nằm lọt giữa hai giá trị này. Dần dần khoảng cách giữa hai giá trị alpha và beta ngày càng thu hẹp và dẫn đến các nhánh cây có giá trị nằm ngoài khoảng này nhanh chóng bị cắt bỏ

*alpha*a

*beta*

Miền xem xét

miền “tồi tệ” (cẳt bỏ)

miền “tồi tệ” (cẳt bỏ)

Hình 2.8 : alpha, beta giống nhau như lưỡi dao dùng để cắt bỏ lớp vỏ dày. Hai lưỡi dao này sẽ được thu hẹp dần dần. Lượng cắt bỏ phụ thuộc vào tốc độ thu hẹp, thu hẹp càng nhanh cắt bỏ càng nhiều.

## **4.4 Đánh giá**

Hiệu quả của việc cắt nhánh Alpha-beta phụ thuộc nhiều vào thứ tự các nước đi kế tiếp được thực hiện. Nếu các nước đi kế tiếp được thực hiện có thứ tự ngẫu nhiên thì tổng số nút được thực hiện sẽ khoảng O(b3d/4). Trong đó b là độ rộng của cây (hệ số phân nhánh trung bình của các con), d là độ sâu của cây.

Trong điều kiện lý tưởng, thuật toán Alpha-beta chỉ phải xét số nút theo công thức:

2-1 với d chẵn

+-1 với *d* lẻ

Như vậy trong trường hợp tốt nhất thuật toán Alpha-beta chỉ cần thực hiện khoảng O(bd/2 ) nút để chọn ra nước đi tốt nhất, thay vì O(bd). Hệ số phân nhánh hiệu quả trở thành thay vì b như trong thuật toán Minimax.

Thuật toán Alpha-beta nói chung giúp chúng ta tiết kiệm nhiều thời gian so với Minimax mà vẫn đảm bảo kết quả tìm kiếm chính xác. Tuy nhiên lượng tiết kiệm này không ổn định- phụ thuộc vào số nút mà nó cắt bỏ. Trong trường hợp xấu nhất thuật toán không cắt được một nhánh nào và phải xem số nút đúng bằng thuật toán Minimax. Ta cần đẩy mạnh việc cắt bỏ nhờ đẩy nhanh sự thu hẹp của cửa sổ tìm kiếm Alpha-beta. Cửa sổ này được thu hẹp một bước khi gặp một giá trị mới tốt hơn giá trị cũ. Khi gặp giá trị tốt nhất thì cửa sổ này thu hẹp nhất. Do đó nếu càng sớm gặp giá trị tốt nhất thì cửa sổ cang chings thu hẹp. Như vậy phải làm sao cho các nút ở lá được sắp xếp theo trật tự từ cao xuống thấp. Trật tự này càng tốt bao nhiêu thì thuật toán chạy càng nhanh bấy nhiêu ( các công thức về số nút phải lượng giá trong điều kiện lý tưởng ở trên tính được với trật tự là tốt nhất).

Ví dụ : Ta xét xem thuật toán Alpha-beta hoạt động như thế nào đối với cây trò chơi sau :

1

3

4

9

11

7

13

17

19

21

24

26

28

34

32

36

5

2

=8

8

9

38

29

14

12

10

30

23

25

27

22

20

18

37

35

33

15

6

39

31

16

2

4

= 4

1

3

= 5

1

2

=3

3

9

6

5

= 5

3

8

= 4

4

5

= 5

Kết quả tìm kiếm nước đi

8 7 2 9 1 6 2 4 1 1 3 5 3 9 2 6 5 2 1 2 3 9 7 2 16 6 4

Hình 2.9 :Một cây trò chơi có độ sâu 3 và hệ số nhánh 3. Số trong các hình chữ nhật cho biết thứ tự đưa ra các kết luận. Chú ý rằng chỉ có tất cả là 16 đánh giá tĩnh được thực hiện (các nút đen) chứ không phải là 27 như khi không dùng thuật toán Alpha-beta. Nhánh đi tốt nhất cho người chơi cực đại là nhánh giữa.

[1-2] tìm kiếm đi xuống dưới theo nhánh trái cho đến lá. ở đây giá trị tĩnh thu được là 8. Giá trị đầu tiên này do người chơi cực đại được cho phép chọn trong ba giá trị ở nhánh này đã đảm bảo rằng kết quả thu được sẽ ít nhất là 8. Điều lưu ý này được bước 2 ghi lại.

[3-5]Để chắc nhắn không còn điểm nào cao hơn 8, người chơi cực đại phải xét cả hai thế cờ còn lại và thu được các giá trị 7 và 2. Do đó đến đây đã kết luận chính xác điểm cao nhất có thể đạt được cây con là đúng bằng 8.

[6]Leo lên một tầng cây. Đây là nước đi của người chơi cực tiểu. Ta không hi vọng anh ta cho người chơi cực đại được nhiều điểm nên có thể tạm kết luận ở mức này sẽ là đạt được nhiều nhất là 8 điểm.

[7-8]Để xem người chơi cực tiểu còn lựa chọn nào tốt hơn (và tồi tệ hơn cho người chơi cực đại) ta phải xem xét cả hai nước đi còn lại. Nước đi còn lại đầu tiên dẫn đến giá trị lượng giá tĩnh là 9>8. Như vậy nhánh giữa là tệ hơn cho người chơi cực tiểu. Đến đây việc cắt bỏ được thực hiện do đó người chơi cực đại không thể với tới được điểm đó khi đã có sẵn lựa chọn thấp hơn cho anh ta (là 8). Điều này cũng dẫn đến không cần thiết phải xem hai nút còn lại-đằng nào nhánh giữa cũng đủ tồi tệ rồi và người chơi cực tiểu sẽ không chọn nó để đi.

[9-14] Người chơi cực tiểu cần phải khảo sát tiếp lựa chọn cuối cùng. Cách làm tương tự như phần trên. Ở đây phải lượng giá cả ba nút cây và kết luận cưối cùng được đưa ra là người chơi cực đại đi giỏi lắm thì chỉ đạt 4 điểm.

[15] Như vậy nhờ việc khảo sát nhánh cây bên phải người chơi cực tiểu thấy rằng nếu chọn đi theo nhánh này thì người chơi cực đại chỉ được có 4 điểm thay cho 8.

[16] Bây giờ ta có thể kết luận ở mức trên cùng. Mức này là của người chơi cực đại. Anh ta thấy rằng nếu chọn đi theo nhánh trái thì được 4 điểm. Như vậy anh ta đã chắc nhắn điểm của mình sẽ ít nhất là 4. Để xem liệu có thể đạt được điểm cao hơn nữa hay không cần phải xét hai nhánh còn lại.

[17-30] Tương tự như phần trên, ta kết luận nhánh giữa sẽ mang lại cho người chơi cực đại 5 điểm.

[31] Cũng tương tự như kết luận 16, ở đây ta kết luận khả quan hơn là người chơi cực đại đã cầm chắc 5 điểm và có thể còn cao hơn.

[32-38] Ta kết luận được rất nhanh là cây con bên phải chỉ cho “thu hoạch” nhiều nhất là 3 điểm-một điểm số quá kém do đó thuật toán không xem xét các trường hợp còn lại nữa. Do đó đã tiết kiệm được 6 nút không cần phải lượng giá và cũng không phải sinh nước đi cho hai trường hợp.

[39] Kết luận cuối cùng là điểm cao nhất mà người chơi cực đại có thể thu được là 5 điểm nhờ chọn đi theo nhánh giữa.

## **4.5 So sánh giải thuật Minimax và giải thuật Alpha-beta.**

Dưới đây là bẳng so sánh số nút phải xét giữa hai giải thuật Minimax và Alpha-beta.

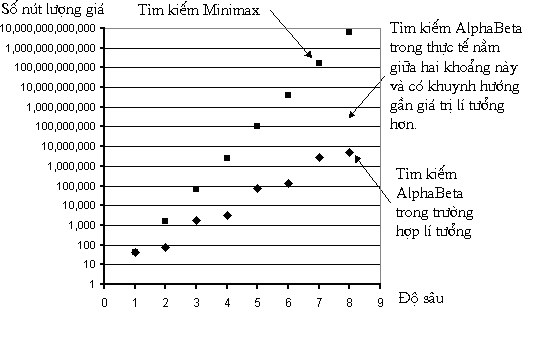
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Độ sâu | Minimax | | AlphaBeta | | Tỉ lệ số nút Minimax/Alpha-beta |
| Số nút | Số lần tăng | Số nút | Số lần tăng |
| 1 | 40 |  | 40 | 1 |  |
| 2 | 1600 | 40 | 79 | 1.9 | 20 |
| 3 | 64000 | 40 | 1852 | 23.2 | 34 |
| 4 | 2560000 | 40 | 3199 | 1.7 | 800 |
| 5 | 102400000 | 40 | 74118 | 23.2 | 1381 |
| 6 | 4096000000 | 40 | 127999 | 1.7 | 32000 |
| 7 | 163840000000 | 40 | 2964770 | 23.2 | 55262 |
| 8 | 6553600000000 | 40 | 5120000 | 1.7 | 1280000 |

Với b=40 và d=4 ta có số nút phải xét là 2x402 - 1 = 3199. Như vậy trong điều kiện lí tưởng thì số nút phải xét nhờ Alpha-beta ( chỉ khoảng 3 nghìn nút) ít hơn thuật toán Minimax (hơn 2.5 triệu nút) là 2560000/3199 khoảng 800 lần. Còn với b=40vaf d=5 ta có số nút phải xét là 403 + 40(5/2) - 1 = 64000+10119-1 = 74118. Số nút phải xét nhờ Alpha-beta ít hơn thuật toán Minimax ( hơn 102 triệu nút) là 102400000/74118 = 1382 lần.

Ta có thể nhận xét như sau:

* Số lần tăng số nút khi tăng độ sâu của Minimax luôn là hệ số phân nhánh b, trong trường hợp này là 40. Số lần tăng của Alpha-beta ít hơn nhiều: chỉ cỡ 1.7 lần khi tăng từ d lẻ sang d chẵn và 32.2 lần khi từ d chẵn và d lẻ, trung bình chỉ tăng khoảng 6 lần khi tăng d.
* Số nút của Alpha-beta tăng chậm hơn rất nhiều so với Minimax. Tỉ số nút phải xét giữa hai giải thuật bày càng cao khi d càng lớn.

Công thức tính số nút cho thấy số nút phải xét khi dùng Alpha-beta ít hơn nhiều so với Minimax nhưng vẫn là hàm số mũ và vẫn dẫn tới bùng nổ tổ hợp. Thuật toán Alpha-beta hoàn toàn không chống được bùng nổ tổ hợp mà chỉ làm giảm tốc độ bùng nổ tổ hợp. Tuy trong thực tế số nút phải xét (lương giá) thường nhiều hơn trong điều kiện lý tưởng nhưng nó vẫn đủ để tiết kiệm khá nhiều thời gian. Trong cùng một khoảng thời gian, thuật toán Alpha-beta có thể tìm đến độ sâu gấp hai lần độ sâu tìm kiếm bằng Minimax. Hình sau đây là đồ thị so sánh giữa hai thuật toán này.

**

Hình 2.10: Khảo sát sự bùng nổ tổ hợp. Thuật toán Alpha-beta chỉ làm giảm sự bùng nổ tổ hợp chứ không chống được nó. Hệ số phân nhánh trong các dồ thị trên là 40.

Tóm lại: Do bùng nổ tổ hợp quá lớn của cây trò chơi mà cả hai người chơi không thể ( và không bao giờ) có thể tìm vét cạn ( hết mọi khả năng). Do đó phương pháp tìm kiếm duy nhất là chỉ tìm kiếm đến một độ sâu giới hạn nào đó và chọn nước đi dẫn đến một thế cờ có lợi nhất cho mình. Do phải tính cả khả năng chống trả của đối phương nên ta không dùng được các thuật toán tìm kiếm thông thường. Phải dùng một thuật toán tìm kiếm riêng cho cho cây trò chơi. Đó là thuật toán Minimax và cải tiến của nó là thuật toán Alpha-beta. Tuy cả hai thuật toán đều không tránh được bùng nổ tổ hợp nhưng thuật toán Alpha-beta làm chậm bùng nổ tổ hợp hơn nên được dùng nhiều trong các trò chơi cờ.

# CHƯƠNG 5: ỨNG DỤNG

## **5.1 Giới thiệu về trò chơi cờ caro**

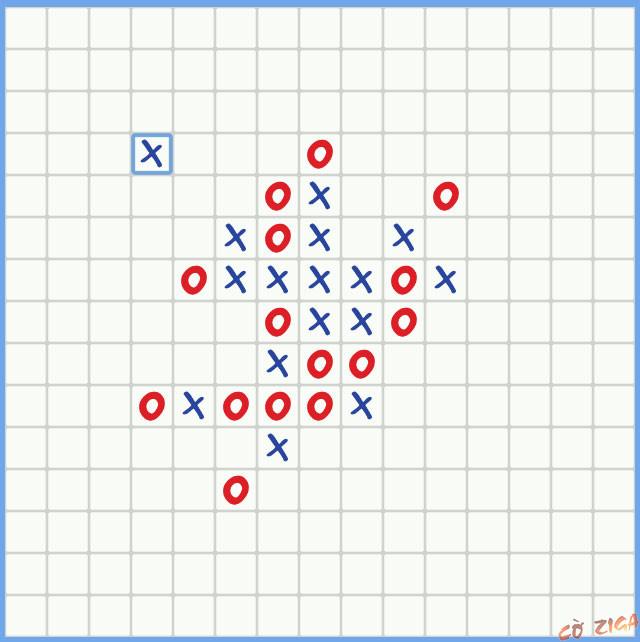
Cờ Caro ( hay cờ ca-rô) là một trò chơi dân gian phổ biến của Việt Nam. Ban đầu loại cờ này được chơi bằng các quân cờ vây (quân cờ màu trắng và đen) trên một bàn cờ vây (19x19). Ở Việt Nam, cờ này thường chơi trên giấy tập học sinh (đã có sẵn các ô ca-rô), dùng bút đánh dấu hình tròn (O) và chữ X để đại diện cho 2 quân cờ. Trò chơi không cần yêu cầu khắt khe về luật và phương tiện chơi, vì vậy trò này được mọi người ưa chuộng và phù hợp với mọi lứa tuổi.

Ngày nay do sự phát triển của công nghệ, trò chơi cờ Caro đã được phát triển trên các nền tảng online, các ứng dụng, trang web được tích hợp trên thiết bị di động, máy tính,…

## **5.2 Các quy định khi chơi cờ Caro**

### **5.2.1 Bàn cờ**

Bạn cần có 1 bàn cờ gồm nhiều ô vuông, tối thiểu là 19\*19. Bàn cờ càng to càng tốt, chơi lâu và cực kỳ gay cấn.



Hình 2.11. game cờ Caro

### **5.2.2 Quân cờ**

Có 2 quân cờ cho bạn lựa chọn được ký hiệu là:

* O
* X

### **5.2.3 Di chuyển**

Không giống như cờ vây, các quân cờ caro bạn phải đi vào trong ô vuông chứ không phải các giao điểm đường kẻ.

* Hai người chơi quyết định quân đi của mình: một bên là dấu hình tròn (O) và chữ X và quyết định người đi trước người đi sau.
* Người chơi đầu tiên đánh dấu vị trí đặt quân của mình lên trên vị trí ô trống đã định trước. Tiếp đến là lượt chơi của người thứ hai với cách đi tương tự như vậy. Quân cờ có thể đánh ở mọi ô vuông trống trên bàn cờ, nhưng không được phép đánh dấu vào ô đã có đánh dấu của mình hoặc đối phương trước đó.

### **5.2.4. Kết thúc ván cờ**

* Ván cờ sẽ kết thúc khi 1 bên nối được 5 quân liên tiếp nhau hoặc kết thúc thời gian ván cờ mà chưa có ai thắng.
* Trong trường hợp hết thời gian hoặc ô vuông để đánh mà không phân thắng bạn thì ván cờ sẽ xử hòa

## **5.3. Chế độ người với máy trong trò chơi cờ Caro**

Khác với trò chơi truyền thống trên giấy, game cờ Caro được lập trình trên máy tính bằng các thuật toán ứng dụng của trí tuệ nhân tạo giúp con người có thể độc lập chơi với máy mà không cần người thứ hai

Game cờ Caro có thể được lập trình viên lựa chọn xây dựng bằng nhiều ngôn ngữ khác nhau như : C#, Java, Python,…và sử dụng nhiều thuật toán khác nhau. Tuy nhiên trong bài tập lớn lần này, nhóm lựa chọn phương pháp triển khai là ngôn ngữ C# và ứng dụng thuật toán Minimax và Alpha-beta để xử lý AI trong game.

## **5.4. Sơ đồ thuật toán**

**Diagram

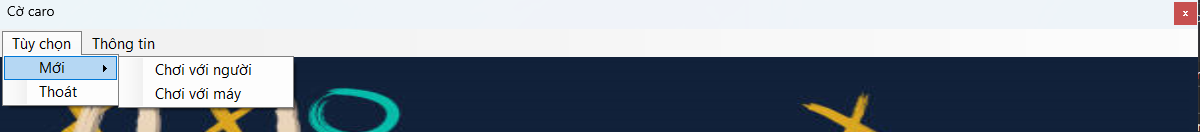
Description automatically generated**

## **5.5 Giao diện trò chơi**



Hình 2.12.Giao diện ban đầu

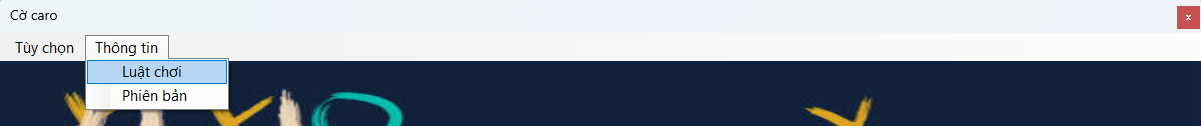
* Thanh menu
  + Chọn chế độ chơi

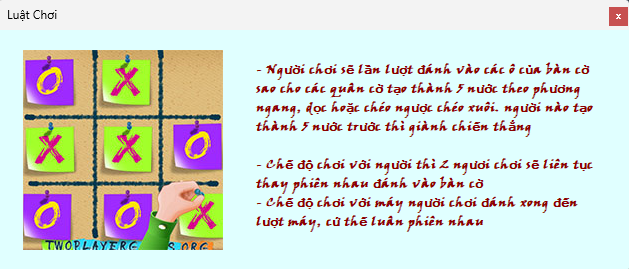
****

* Thoát trò chơi

****

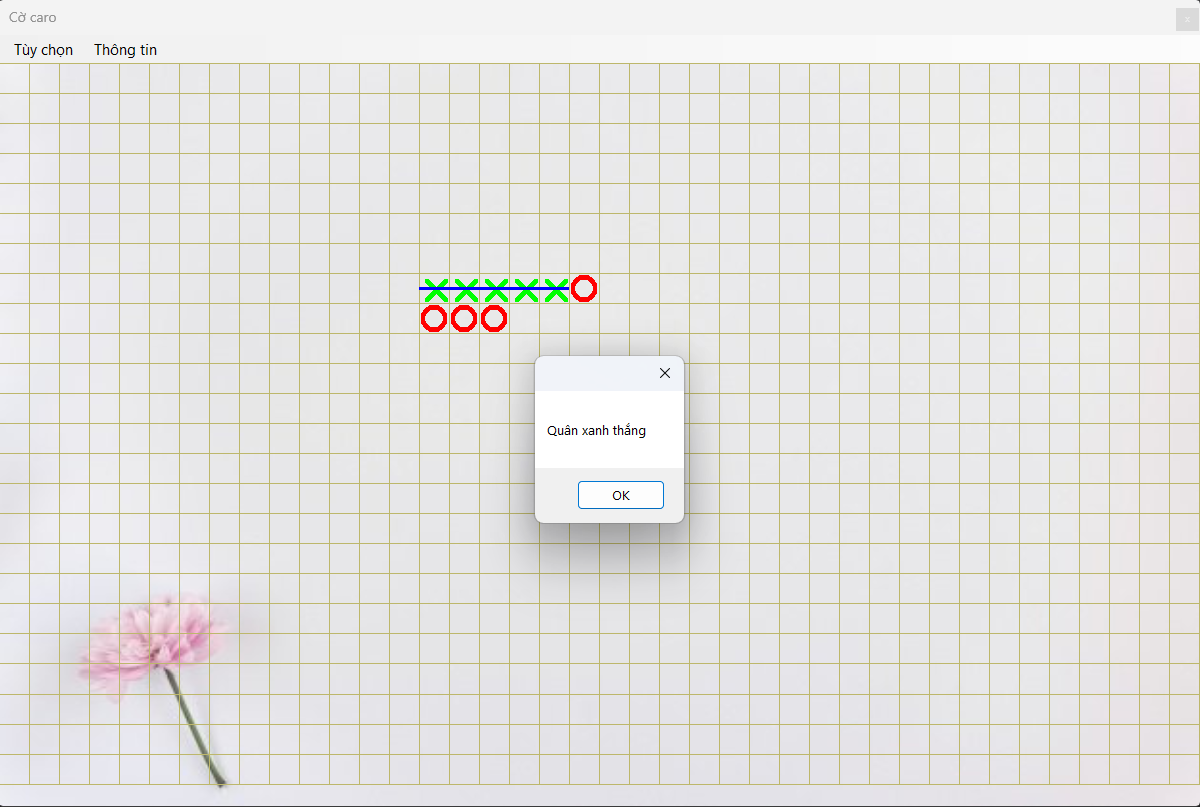
* Xem thông tin trò chơi



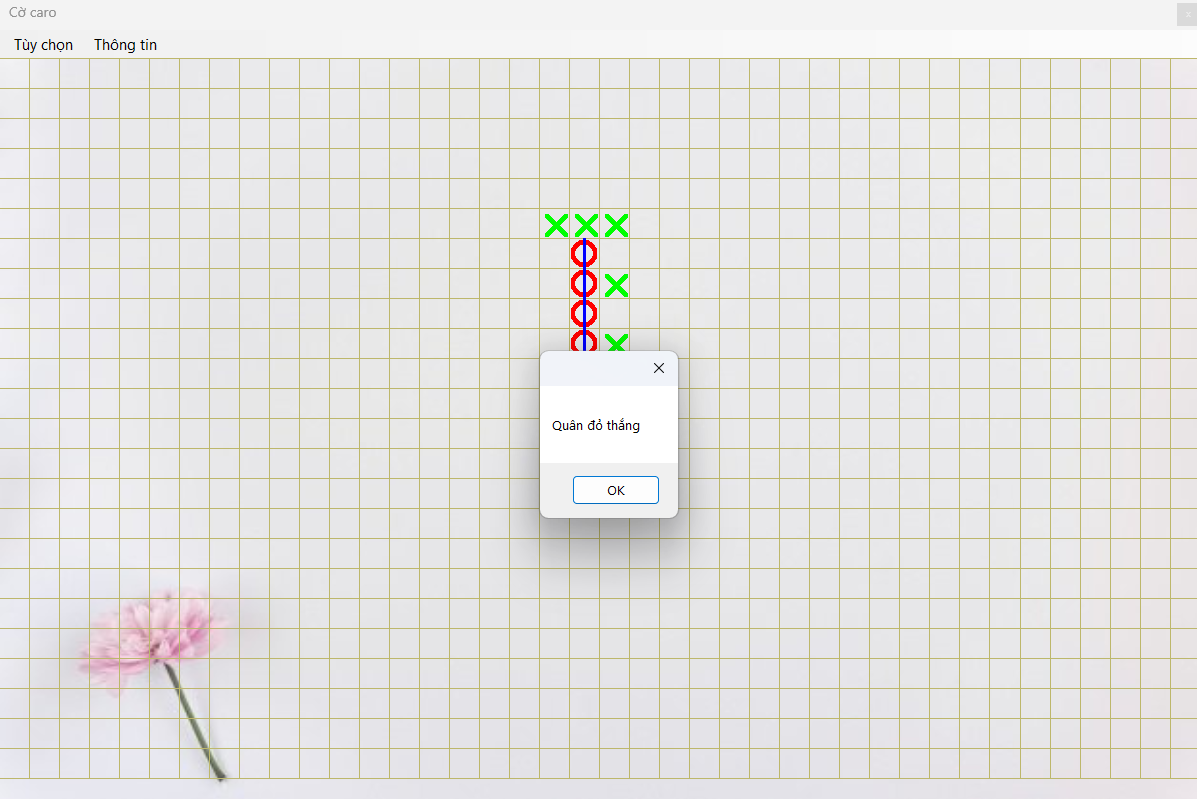
****

Hình 2.13.Luật chơi

**Giao diện trong chế độ Chơi với người**

****

H2.14 Giao diện thông báo khi người chơi 1 thắng

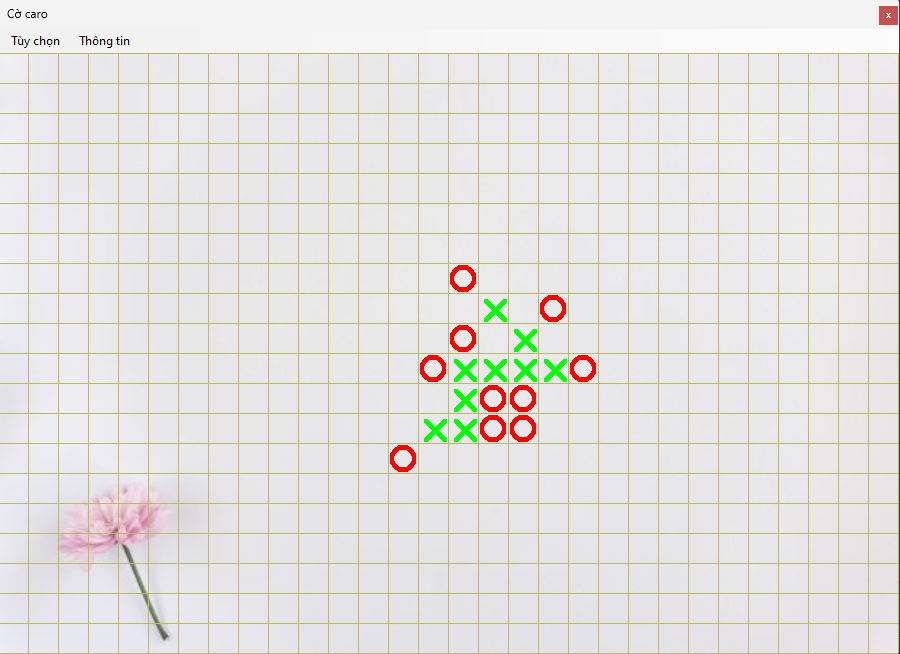
****

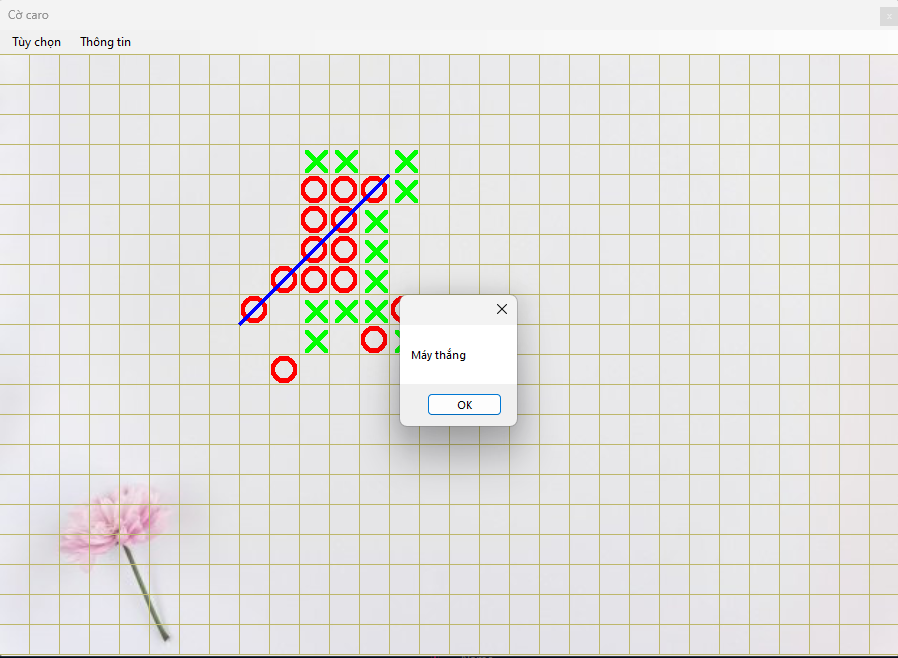
H2.15.Giao diện thông báo khi người chơi 2 thắng

**Giao diện trong chế độ Chơi với máy**

Diagram

Description automatically generated





H2.16, H2.17, H2.18. Giao diện khi chơi với máy

# **KẾT LUẬN**

Thuật toán AlphaBeta nói chung giúp chúng em tiết kiệm nhiều thời gian so  
với Minimax mà vẫn đảm bảo kết quả tìm kiếm chính xác. Tuy nhiên lượng  
tiết kiệm này không ổn định - phụ thuộc vào số nút mà nó cắt bỏ. Trong  
trường hợp xấu nhất thuật toán không cắt được một nhánh nào và phải xét số  
nút đúng bằng Minimax. Ta cần đẩy mạnh việc cắt bỏ nhờ đẩy nhanh sự thu  
hẹp của cửa sổ tìm kiếm alpha - beta. Cửa sổ này được thu hẹp một bước khi  
gặp một giá trị mới tốt hơn giá trị cũ. Khi gặp giá trị tốt nhất thì cửa sổ này  
thu hẹp nhất. Do đó nếu càng sớm gặp giá trị tốt nhất thì cửa sổ càng chóng  
thu hẹp. Như vậy phải làm sao cho các nút ở lá được sắp xếp theo trật tự từ  
cao xuống thấp. Trật tự này càng tốt bao nhiêu thì thuật toán chạy càng nhanh  
bấy nhiêu (các công thức về số nút phải lượng giá trong điều kiện lí tưởng ở  
trên tính được với trật tự là tốt nhất).

Qua quá trình học môn trí tuệ nhân tạo và qua việc thực hiện đề tài này, chúng em đã hiểu thêm hơn về việc ứng dụng trí tuệ nhân tạo trong việc giải quyết các vấn đề trong thực tế. Game cờ caro là một game ứng dụng rất tốt thuật toán Minimax và cắt cụt alpha-beta. Tuy nhiên nhóm em chưa thể tối ưu được thuật toán, hàm tính điểm cũng chưa thật sự được tốt như mong đợi, cũng như trong quá trình thực hiện không thể tránh khỏi sai sót. Chúng em rất mong nhận được sự góp ý của cô để game được thật sự hoàn thiện.

Chúng em xin chân thành cảm ơn!

# **TÀI LIỆU THAM KHẢO**

[1] Nguyễn Phương Nga, Trần Hùng Cường, Giáo trình trí tuệ nhân tạo, 2021

[2] <https://vi.wikipedia.org/wiki/Minimax>

[3] <https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/>

[4] <https://stackoverflow.com/questions/32425162/minimax-algorithm-for-tic-tac-toe>

[5] <https://viblo.asia/p/thuat-toan-minimax-ai-trong-game-APqzeaVVzVe>