BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH

XÂY DỰNG AGENT CHO TIC-TAC-TOE

BÀI TẬP LỚN MÔN HỌC

MÔN: TRÍ TUỆ NHÂN TẠO

Thành viên nhóm

Trương Tường Vi – 2151050542

Dương Quốc Khánh - 2051010140

Giảng viên hướng dẫn: TS.LÊ VIẾT TUẤN

TP. HỒ CHÍ MINH, 2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG

CHƯƠNG 1. GIỚI THIỆU

1.1. Tổng quan bài toán

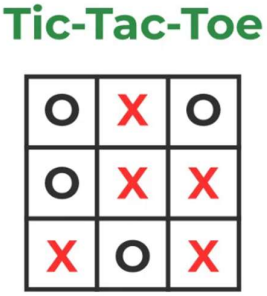
1.1.1. Giới thiệu đề tài

Trí tuệ nhân tạo (Artificial Intelligence – AI) ngày nay được ứng dụng rộng rãi trong nhiều lĩnh vực như thị giác máy tính, xử lý ngôn ngữ tự nhiên, hay các hệ thống ra quyết định. Một trong những hướng nghiên cứu nền tảng và thường được sử dụng để minh họa các thuật toán AI là các trò chơi đối kháng (adversarial games).

Trong số đó, Tic-tac-toe (hay còn gọi là cờ ca-rô 3x3) là một trò chơi đơn giản nhưng mang tính khái quát cao. Trò chơi có luật như sau:

- Hai người chơi lần lượt đánh dấu "X" và "O" trên bàn cờ 3x3.
- Người thắng là người đầu tiên tạo được một hàng, cột hoặc đường chéo gồm 3 ký hiệu giống nhau.
- Nếu bàn cờ đầy mà không có người thắng, ván đấu kết thúc với kết quả hòa.

Tuy với không gian trạng thái nhỏ, Tic-tac-toe được coi là một ví dụ điển hình để thử nghiệm các thuật toán tìm kiếm và ra quyết định trong AI. Bằng việc xây dựng một agent (tác tử thông minh), có thể mô phỏng khả năng tư duy chiến lược, đưa ra nước đi tối ưu và cạnh tranh trực tiếp với người chơi.



Hình 1. Ví dụ Hình trong báo cáo

1.1.2. Kiểm tra trạng thái và xác định kết quả

Sau mỗi lượt đi của máy hoặc người chơi, hệ thống cần kiểm tra xem trò chơi đã kết thúc hay chưa. Thuật toán kiểm tra được triển khai trong repo gồm các bước sau:

Xác thực nước đi

- Khi người chơi nhập vào vị trí muốn đánh, hệ thống kiểm tra xem ô đó có hợp lệ không (tức là ô còn trống).
- Nếu hợp lệ, trạng thái bàn cờ được cập nhật.

Kiểm tra điều kiện thắng

- Hệ thống duyệt qua tất cả các hàng ngang, cột dọc và hai đường chéo.
- Nếu có ba ô liên tiếp chứa cùng một ký hiệu (X hoặc O), người chơi tương ứng sẽ được xác định là chiến thắng.

Kiểm tra điều kiện hòa

- Nếu chưa có người thắng, hệ thống tiếp tục kiểm tra tổng số lượt đi đã được thực hiện.
- Nếu cả 9 ô đều đã được điền và không ai chiến thắng, trò chơi kết thúc với kết quả hòa.

Cập nhật và thông báo kết quả

- Sau khi kiểm tra, hệ thống cập nhật biến trạng thái và đưa ra thông báo thích hợp cho người chơi: thắng, thua hoặc hòa.

Nhờ cơ chế này, trò chơi luôn đảm bảo được theo dõi chính xác trạng thái bàn cờ và phản hồi kết quả kịp thời sau mỗi lượt đi.

1.2. Hạn chế của các phương pháp hiện tại

Mặc dù bài toán Tic-tac-toe đã được nghiên cứu và áp dụng nhiều phương pháp khác nhau, mỗi cách tiếp cận đều tồn tại những hạn chế nhất định:

Chiến lược ngẫu nhiên (Random Strategy)

- Đây là phương pháp đơn giản nhất, khi máy lựa chọn nước đi một cách hoàn toàn ngẫu nhiên
- Ưu điểm: dễ cài đặt, không đòi hỏi thuật toán phức tạp.
- Hạn chế: thiếu tính thông minh, dễ dẫn đến thua cuộc và không thể hiện được khả năng ra quyết định.

Chiến lược dựa trên luật (Rule-based Strategy)

- Máy được lập trình sẵn một tập hợp các quy tắc, ví dụ: "nếu có thể thắng thì đánh ngay", "nếu đối thủ sắp thắng thì chặn lại", "ưu tiên ô giữa nếu trống".
- Ưu điểm: dễ hiểu, hoạt động khá hiệu quả trong nhiều tình huống thông thường.
- Hạn chế: khó tổng quát hóa cho các tình huống phức tạp, dễ mắc kẹt trong các tình huống bất lợi và gần như không thể mở rộng cho các trò chơi có không gian trạng thái lớn hơn.

Tìm kiếm vét cạn toàn bộ không gian trạng thái (Brute-force Search)

- Phương pháp này duyệt qua toàn bộ các trạng thái bàn cờ để tìm ra nước đi tốt nhất.
- Ưu điểm: đảm bảo tìm được nước đi tối ưu trong không gian nhỏ như Tic-tac-toe.
- Hạn chế: tốn nhiều tài nguyên khi áp dụng cho các trò chơi có không gian trạng thái lớn hơn (ví dụ: cờ vua, cờ vây), do gặp phải hiện tượng bùng nổ tổ hợp (combinatorial explosion).

Từ những phân tích trên có thể thấy rằng, các phương pháp hiện tại hoặc quá đơn giản nên không hiệu quả, hoặc quá tốn kém tài nguyên khi mở rộng. Điều này đặt ra nhu cầu cần có một giải pháp vừa hiệu quả về mặt tính toán, vừa đảm bảo được tính tối ưu trong chiến lược chơi, điển hình như việc sử dụng thuật toán Minimax.

1.3. Phương pháp đề xuất

Trong đề tài này, nhóm chúng em lựa chọn phương pháp xây dựng agent chơi Tic-tac-toe dựa trên thuật toán Minimax, đây là tham khảo và phát triển từ mã nguồn mở tic-tac-toe-minimax của Cledersonbc (GitHub). Thuật toán Minimax được sử dụng để mô phỏng quá trình ra quyết định trong trò chơi hai người chơi đối kháng có tổng bằng 0, trong đó:

- Agent (máy) đóng vai trò người chơi Max, luôn cố gắng tối đa hóa điểm số và tìm nước đi tốt nhất cho mình.
- Người chơi (user) đóng vai trò người chơi Min, được giả định là luôn chọn nước đi tối ưu để giảm thiểu lợi thế của máy.

Cụ thể, các trạng thái bàn cờ được duyệt đệ quy từ trạng thái hiện tại đến trạng thái kết thúc (win, lose hoặc draw). Tại trạng thái cuối, giá trị được đánh giá như sau:

- Thắng: +1 điểm.
- Thua: -1 điểm.
- Hòa: 0 điểm.

Các giá trị này được lan truyền ngược trở lại để quyết định nước đi tối ưu. Nhờ đó, agent có thể đưa ra lựa chọn an toàn nhất, đảm bảo không thua khi cả hai bên đều chơi tối ưu. Trong phạm vi đề tài, nhóm chúng em triển khai Minimax cơ bản theo cấu trúc của repo gốc.

1.4. Đóng góp của đề tài

Đề tài "Tic-tac-toe" dự kiến mang lại những đóng góp sau:

- Áp dụng mã nguồn mở vào thực tiễn: tái hiện, cài đặt và phân tích chi tiết một phiên bản thuật toán Minimax từ repo tic-tac-toe-minimax, qua đó giúp sinh viên hiểu rõ hơn về cách thức hoạt động của thuật toán trong môi trường thực tế.
- Xây dựng một agent thông minh có thể chơi Tic-tac-toe với hiệu suất cao, không thua trước người chơi thông thường.

- Minh họa việc ứng dụng thuật toán tìm kiếm (Minimax) trong một trò chơi thực tế.
- Cung cấp giao diện trực quan để người dùng có thể tương tác và chơi trực tiếp với agent.
- Góp phần củng cố kiến thức về trí tuệ nhân tạo trong trò chơi, làm nền tảng cho các nghiên cứu nâng cao (cờ vua, cờ vây, game phức tạp hơn).

Thông qua những đóng góp trên, đề tài không chỉ củng cố kiến thức lý thuyết về trí tuệ nhân tạo và thuật toán tìm kiếm, mà còn mang lại kinh nghiệm thực hành quan trọng trong việc triển khai, phân tích và đánh giá một tác tử thông minh trong trò chơi.

CHƯƠNG 2. CÁC NGHIÊN CỨU LIÊN QUAN

2.1.1. Tổng quan

Tic-Tac-Toe từ lâu được xem là một "case study" kinh điển trong lĩnh vực Trí tuệ nhân tạo. Do không gian trạng thái nhỏ, trò chơi này thường được chọn làm ví dụ minh họa trong giáo trình, luận văn và các nghiên cứu thử nghiệm. Nhiều công trình sử dụng Tic-Tac-Toe để giới thiệu về **tìm kiếm trong không gian trạng thái, chiến lược tối ưu và trò chơi hai người đối kháng**.

2.1.2. Nghiên cứu dựa trên Minimax

- **Russell & Norvig** trong giáo trình *Artificial Intelligence: A Modern Approach* đã mô tả Tic-Tac-Toe như một ví dụ chuẩn để giới thiệu thuật toán Minimax.
- Các nghiên cứu tại nhiều trường đại học sử dụng Minimax để dạy sinh viên cách xây dựng cây trò chơi, gán giá trị trạng thái và chọn nước đi tối ưu.
- Một số bài báo ứng dụng Minimax mở rộng với **depth limit** (giới hạn độ sâu) nhằm tạo ra các mức độ khó khác nhau cho AI, phục vụ cả mục đích học thuật lẫn giải trí.

2.1.3. Nghiên cứu cải tiến với Alpha-Beta pruning

- Các công trình so sánh Minimax gốc với Minimax + Alpha-Beta pruning cho thấy số trạng thái duyệt giảm đáng kể mà kết quả không thay đổi.
- Alpha-Beta pruning nhờ vậy được khẳng định là kỹ thuật chuẩn khi triển khai AI Tic-Tac-Toe và cả các trò chơi phức tạp hơn như Connect Four, Gomoku.
- Nhiều nghiên cứu khuyến nghị kết hợp Alpha-Beta với heuristic (ví dụ: ưu tiên ô giữa, góc) để tăng tốc độ mà vẫn giữ sức mạnh.

2.1.4. Nghiên cứu mở rộng với học máy và học tăng cường

- Một số công trình thử nghiệm **Q-learning** và **Reinforcement Learning** trên Tic-Tac-Toe để minh họa khả năng học chính sách tối ưu từ trải nghiệm thay vì duyệt cây trạng thái.
- Kết quả cho thấy các phương pháp học máy có thể đạt chiến lược hòa hoặc thắng tối ưu mà không cần biết luật sẵn, chỉ bằng tự chơi nhiều lần.
- Dù Tic-Tac-Toe đơn giản, các nghiên cứu này chứng minh tiềm năng của RL cho các trò chơi lớn hơn như cờ vua hoặc Go.

2.1.5. Ứng dụng thực tế trong giáo dục và mô phỏng

- Nhiều trường học và khóa học trực tuyến triển khai dự án Tic-Tac-Toe như một bài tập thực hành AI nhập môn.
- Tic-Tac-Toe cũng được sử dụng để kiểm chứng thuật toán trong mô phỏng đa tác tử, vì dễ triển khai và trực quan.

- Một số nghiên cứu ứng dụng Tic-Tac-Toe trong **giáo dục STEM**, giúp sinh viên học lập trình và suy luận logic.

2.1.6. Tổng kết

Tổng quan các nghiên cứu liên quan cho thấy Tic-Tac-Toe đóng vai trò như một mô hình thử nghiệm nền tảng trong AI. Từ Minimax, Alpha-Beta pruning cho đến Reinforcement Learning, nhiều công trình đã khai thác trò chơi này để minh họa, so sánh và đánh giá phương pháp. Đây là cơ sở quan trọng cho phần triển khai và thực nghiệm trong báo cáo.

CHƯƠNG 3. PHƯƠNG PHÁP

3.1. Tổng quan phương pháp

Trong đề tài này, chúng tôi đề xuất xây dựng một agent chơi Tic-tac-toe dựa trên thuật toán Minimax. Đây là một trong những thuật toán kinh điển trong lý thuyết trò chơi, thường được áp dụng trong các trò chơi đối kháng hai người có tổng bằng 0.

Ý tưởng chính của phương pháp là:

- Agent đóng vai trò người chơi Max, luôn tìm cách tối đa hóa cơ hội chiến thắng.
- Người chơi (user) được giả định là người chơi Min, có mục tiêu ngược lại là giảm thiểu lợi thế của agent.

Quá trình hoạt động của Minimax bao gồm:

1. Biểu diễn bàn cờ dưới dạng ma trận 3x3.
2. Sinh ra cây trạng thái, trong đó mỗi nút biểu diễn một trạng thái bàn cờ và các nhánh tương ứng với các nước đi hợp lệ.
3. Đánh giá trạng thái cuối cùng (thắng = +1, thua = –1, hòa = 0).
4. Lan truyền các giá trị này ngược lên để xác định nước đi tối ưu tại gốc.

Bên cạnh đó, chúng tôi cũng dự kiến áp dụng cắt tỉa Alpha-Beta để loại bỏ những nhánh không cần thiết trong cây trò chơi, qua đó giảm số lượng trạng thái phải duyệt mà vẫn đảm bảo kết quả tối ưu.

Với cách tiếp cận này, agent có thể đảm bảo không thua trước người chơi nếu cả hai đều chơi tối ưu. Đồng thời, phương pháp này cũng cho phép so sánh trực tiếp hiệu quả với các chiến lược đơn giản hơn như random strategy hoặc rule-based strategy, từ đó chứng minh được tính ưu việt của Minimax.

3.2. Mô hình agent theo khung PEAS

Để mô tả đầy đủ đặc điểm của agent Tic-tac-toe, nhóm chúng em áp dụng khung PEAS (Performance, Environment, Actuators, Sensors) như sau:

P – Performance measure (Thước đo hiệu suất)

- Ưu tiên thắng ván cờ.
- Nếu không thể thắng thì hướng đến hòa.
- Tránh thua bằng mọi giá khi cả hai bên đều chơi tối ưu.
- Thời gian phản hồi nhanh sau khi người chơi nhập nước đi.

E – Environment (Môi trường)

- Bàn cờ Tic-tac-toe kích thước 3x3.
- Hai người chơi: agent (máy) và người dùng (user).
- Mỗi lượt đi, môi trường thay đổi khi một ô trống được điền X hoặc O.

A – Actuators (Bộ chấp hành)

- Thực hiện hành động đánh ký hiệu X lên một ô trống hợp lệ.
- Hiển thị nước đi của agent trên giao diện để người chơi quan sát.

S – Sensors (Bộ cảm biến)

- Đọc trạng thái bàn cờ (các ô X, O và ô trống).
- Nhận đầu vào từ người chơi (nước đi vừa nhập).
- Kiểm tra trạng thái kết thúc: thắng, thua hoặc hòa.

Như vậy, agent được thiết kế để tương tác với môi trường thông qua sensors và actuators, từ đó áp dụng thuật toán Minimax để tối ưu hóa thước đo hiệu suất đã định nghĩa như trên.

3.3. Minimax

3.3.1. Giới thiệu Minimax

Thuật toán **Minimax** là một trong những phương pháp cơ bản trong lý thuyết trò chơi, được sử dụng rộng rãi trong các trò chơi đối kháng hai người có tổng bằng 0, ví dụ: cờ vua, cờ vây, Tic-tac-toe. Các trò chơi này được gọi là trò chơi tổng bằng không, vì trong biểu diễn toán học: một người chơi thắng (+1) và người chơi kia thua (-1) hoặc cả hai hoặc bất kỳ ai không thắng (0). Ý tưởng chính của Minimax là mô phỏng quá trình ra quyết định khi hai người chơi có mục tiêu đối lập:

- Một bên (Max) cố gắng tối đa hóa lợi ích của mình.
- Bên còn lại (Min) cố gắng tối thiểu hóa lợi ích của đối thủ.

Trong trò chơi Tic-tac-toe, chúng tôi gán:

- Agent (máy) là người chơi Max.
- Người dùng (user) là người chơi Min.

Nhờ vậy, agent có thể tính toán các trạng thái có thể xảy ra và chọn nước đi tối ưu, đảm bảo không thua nếu cả hai bên đều chơi tối ưu.

3.3.2. Cách hoạt động

Thuật toán Minimax hoạt động dựa trên nguyên tắc duyệt cây trò chơi một cách đệ quy, nhằm tìm ra nước đi tối ưu cho người chơi Max (agent). Ý tưởng chính là agent sẽ lựa chọn nước đi giúp mình thắng, hoặc trong trường hợp không thể thắng thì đảm bảo ít nhất không thua (hòa)**.**

Quá trình thực hiện được mô tả như sau:

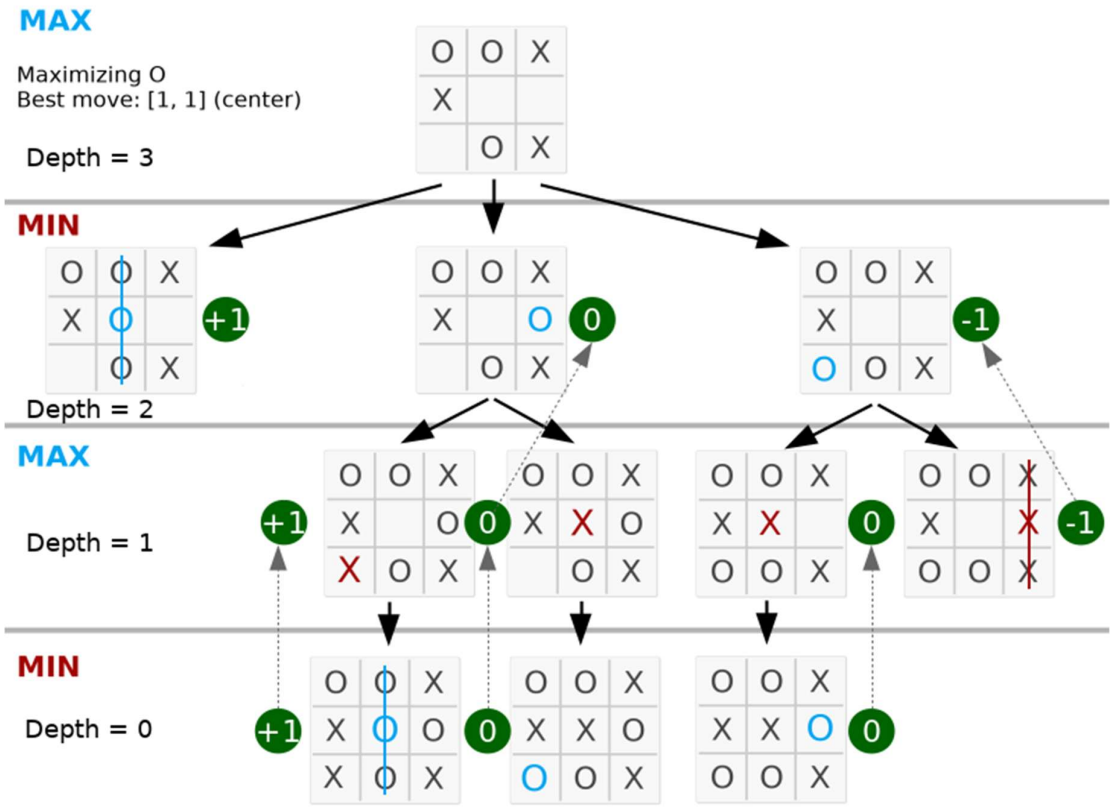
1. Xác định trạng thái hiện tại
    - Thuật toán bắt đầu từ trạng thái bàn cờ hiện tại.
    - Từ trạng thái này, liệt kê tất cả các nước đi hợp lệ (các ô trống trên bàn cờ).
2. Sinh trạng thái tiếp theo
    - Với mỗi nước đi hợp lệ, thuật toán sẽ giả lập thực hiện nước đi đó.
    - Lượt chơi được xen kẽ: nếu bước hiện tại là của Max thì bước tiếp theo sẽ là của Min, và ngược lại.
3. Duyệt đệ quy đến trạng thái kết thúc
    - Thuật toán tiếp tục mô phỏng các nước đi luân phiên Max–Min cho đến khi bàn cờ rơi vào một trạng thái kết thúc:
        - Máy (Max) thắng → trả về giá trị +1.
        - Người (Min) thắng → trả về giá trị –1.
        - Hòa → trả về giá trị 0.

4. Lan truyền giá trị ngược về gốc
    - Tại các nút Max: thuật toán chọn giá trị lớn nhất trong số các trạng thái con.
    - Tại các nút Min: thuật toán chọn giá trị nhỏ nhất trong số các trạng thái con.
5. Lựa chọn nước đi tối ưu
    - Khi giá trị được lan truyền ngược về trạng thái ban đầu, agent sẽ lựa chọn nước đi tương ứng với giá trị tốt nhất.
    - Nhờ đó, agent có thể đưa ra quyết định đảm bảo thắng hoặc hòa, ngay cả khi đối thủ chơi tối ưu.

3.3.3. Hiểu thuật toán và cài đặt

3.3.4. Cây trò chơi

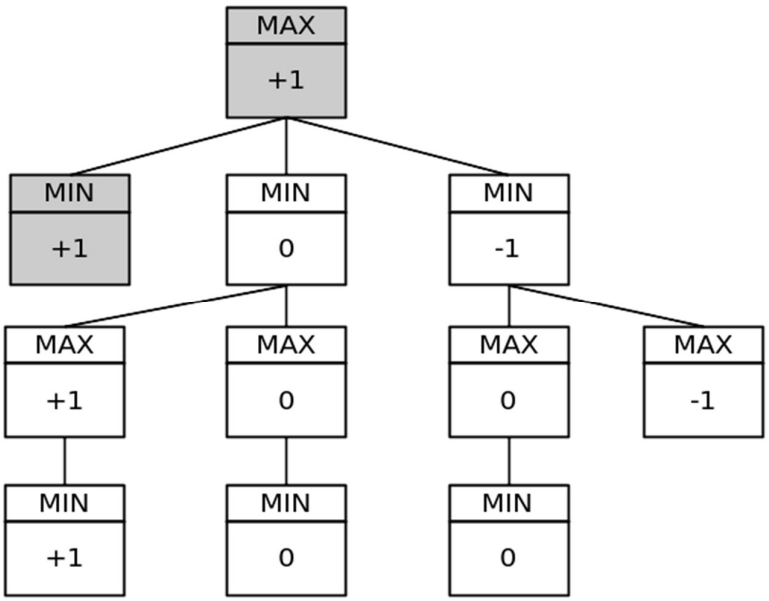
Bên dưới, nước đi tốt nhất là ở giữa vì giá trị lớn nhất nằm ở nút thứ 2 bên trái.



Hình 2. Cây Trò Chơi

Hãy kiểm tra xem độ sâu có bằng với số nước đi hợp lệ trên bàn cờ không. Mã đầy đủ có sẵn trong **py_version/**.

Cây trò chơi đơn giản hóa:



Hình 3. Cây đơn giản

Cây đó có 11 nút. Toàn bộ cây trò chơi có 549.946 nút! Bạn có thể kiểm tra nó bằng cách đặt một biến toàn cục tĩnh vào chương trình và tăng nó cho mỗi lần gọi hàm minimax mỗi lượt.

Trong một trò chơi phức tạp hơn, chẳng hạn như cờ vua, việc tìm kiếm toàn bộ cây trò chơi rất khó khăn. Tuy nhiên, Alpha-beta Pruning là một phương pháp tối ưu hóa cho thuật toán minimax, cho phép chúng ta bỏ qua một số nhánh trong cây tìm kiếm, vì thuật toán này cắt bỏ các nút không liên quan (cây con) trong quá trình tìm kiếm.

3.4. Phân tích ưu nhược điểm

Thuật toán Minimax là một trong những phương pháp kinh điển và hiệu quả để xây dựng agent trong các trò chơi đối kháng hai người. Khi áp dụng vào bài toán Tic-tac-toe, phương pháp này thể hiện nhiều ưu điểm nhưng cũng tồn tại một số hạn chế nhất định.

Ưu điểm

- Đảm bảo tính tối ưu: Minimax luôn đưa ra nước đi tốt nhất trong mọi tình huống, giả định rằng đối thủ cũng chơi tối ưu. Với trò chơi nhỏ như Tic-tac-toe, agent sử dụng Minimax có thể đảm bảo không bao giờ thua; kết quả tối thiểu luôn là hòa.
- Tính khái quát cao: Minimax là nền tảng của nhiều thuật toán AI trong trò chơi, có thể áp dụng cho nhiều trò chơi đối kháng khác nhau.
- Dễ cài đặt và minh họa: Thuật toán có cấu trúc rõ ràng, dễ triển khai và kiểm chứng. Đây là lợi thế lớn trong bối cảnh bài tập lớn học phần Trí tuệ nhân tạo.
- Khả năng kết hợp cải tiến: Thuật toán có thể được tối ưu thêm bằng cắt tỉa Alpha-Beta nhằm giảm số lượng trạng thái cần duyệt mà vẫn giữ được tính tối ưu.

Hạn chế

- Bùng nổ tổ hợp: Dù Tic-tac-toe nhỏ, số trạng thái tối đa vẫn có thể lên đến 9! ≈ 362.880. Với các trò chơi phức tạp hơn như cờ vua hay cờ vây, số trạng thái tăng theo cấp số mũ, khiến Minimax không còn khả thi nếu không có biện pháp tối ưu.
- Tốc độ xử lý: Với việc duyệt toàn bộ cây trò chơi, tốc độ phản hồi có thể bị chậm nếu không kết hợp cắt tỉa Alpha-Beta, đặc biệt khi mở rộng sang các biến thể lớn hơn (ví dụ bàn cờ 4x4, 5x5).
- Thiếu khả năng học hỏi: Minimax chỉ dựa vào duyệt trạng thái, không có cơ chế tự học từ kinh nghiệm hoặc điều chỉnh chiến lược dựa trên hành vi đối thủ. Do đó, agent không thể cải thiện hiệu suất qua thời gian.

Hướng phát triển

- Để khắc phục các hạn chế trên, có thể xem xét:
- Kết hợp cắt tỉa Alpha-Beta để cải thiện tốc độ xử lý.
- Ứng dụng hàm đánh giá heuristic nhằm ước lượng trạng thái thay vì duyệt toàn bộ cây, giúp mở rộng cho các trò chơi phức tạp hơn.

- Tích hợp các phương pháp hiện đại như Học tăng cường (Reinforcement Learning) để agent có thể tự cải thiện chiến lược sau nhiều ván chơi.

CHƯƠNG 4. KẾT QUẢ VÀ THỰC NGHIỆM

4.1. Các tập dataset

Trong bối cảnh Tic-Tac-Toe, “dataset” là các **kịch bản thí nghiệm** được xác định bởi đối thủ, số ván, độ khó và cách luân phiên lượt đi. Mỗi lần chạy sinh ra một nhật ký (CSV) gồm các ván độc lập để tính thống kê.

Định dạng bản ghi đề xuất (CSV): game_id, difficulty ∈ {easy, normal, hard, impossible}, opponent ∈ {random, optimal}, ai_starts ∈ {0,1}, result ∈ {1=win,0=draw,−1=loss}, moves, timestamp.

4.1.1. Tập dataset 1

Tập dataset 1 – Đối thủ random

- Cấu hình: mỗi độ khó chơi N=200 ván; AI và đối thủ **luân phiên đi trước**.

- Tham số độ khó: easy(0.60,1,✗), normal(0.20,3,✗), hard(0.05,9,✓), impossible(0.00,9,✓).

- Mục tiêu: đo tỉ lệ Thắng/Hòa/Thua và số nước đi trung bình theo độ khó.

4.1.2. Tập dataset 2

Tập dataset 2 – Đối thủ optimal (Minimax hoàn hảo)

- Cấu hình: mỗi độ khó chơi M=40 ván; luân phiên lượt đi.

- Kỳ vọng: mức hard/impossible **tiệm cận hòa**; easy/normal có thể thua do ngẫu nhiên và độ sâu thấp.

- Ghi nhận thêm: các mẫu trận thua của easy/normal để phân tích bẫy hai đường.

4.2. Kết quả

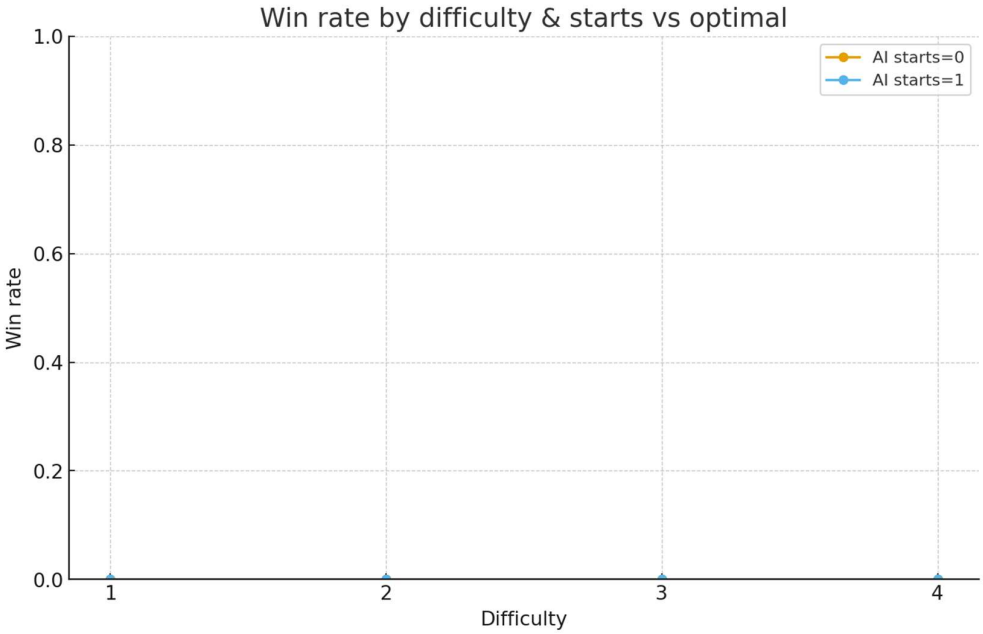
Phân tích dựa trên hai nhóm chỉ số:

- **Định lượng**: Thắng/Hòa/Thua, tỉ lệ %, số nước đi trung bình, thời gian quyết định trung bình (nếu đo).

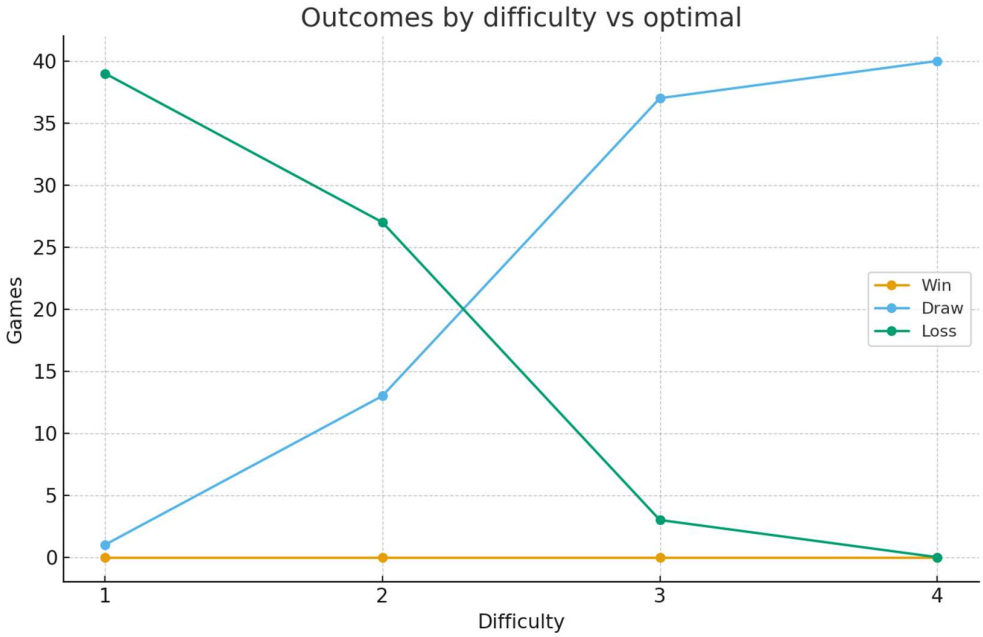
- **Định tính**: mô tả chiến thuật đi giữa, ưu tiên góc, tạo hai đường, và các lỗi do p_random.

4.2.1. So sánh định lượng

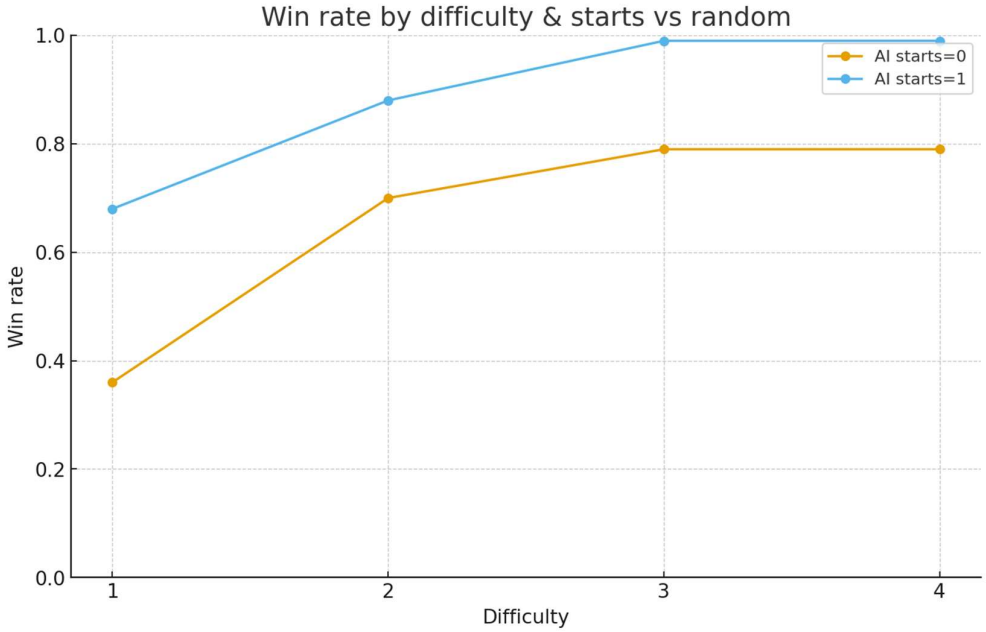
Hình 4.t: Win rate theo độ khó & lượt đi vs optimal



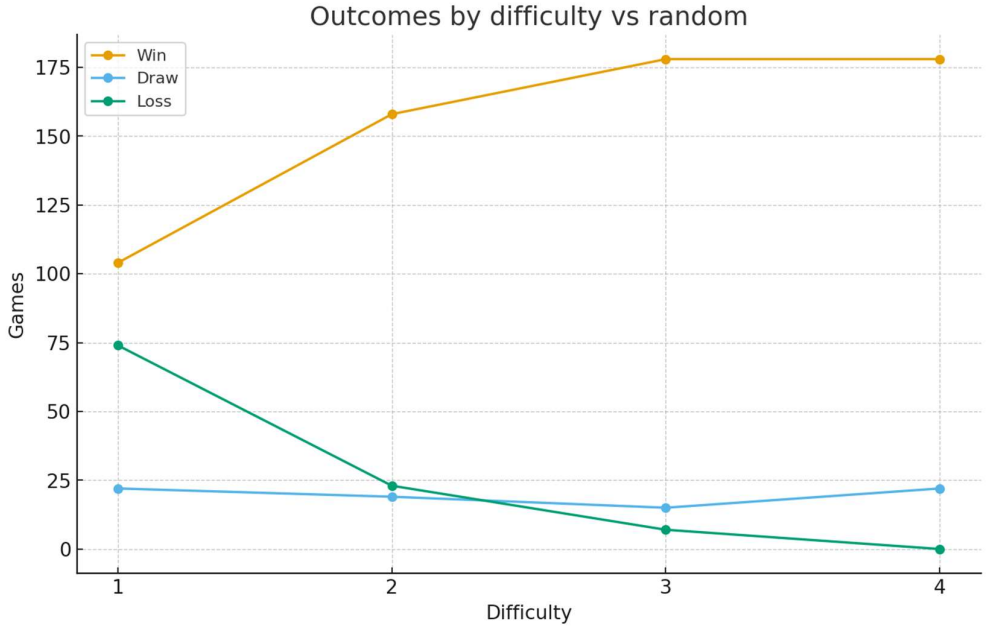
Hình 4.z: Outcomes theo độ khó vs optimal



Hình 4.y: Win rate theo độ khó & lượt đi vs random



Hình 4.x: Outcomes theo độ khó vs random



Kết quả định lượng với hình ảnh minh họa:

opponent	difficulty	games	win	draw	loss
optimal	easy	40	0	1	39
optimal	hard	40	0	37	3
optimal	impossible	40	0	40	0
optimal	normal	40	0	13	27
random	easy	200	104	22	74
random	hard	200	178	15	7
random	impossible	200	178	22	0
random	normal	200	158	19	23

4.2.2. So sánh định tính

- Mức **impossible** loại lỗi ngẫu nhiên nên né bẫy tốt; trước *optimal* chủ đạo là **hòa**.

- Mức **easy/normal** thua nhiều trước *optimal* do p_random cao và depth_limit thấp.

- Chiến thuật hiệu quả: chiếm **center**, ưu tiên **góc**, tạo hoặc chặn **two-way threat**.

CHƯƠNG 5. KẾT LUẬN

5.1.1. Kết luận chung

Trong báo cáo này, chúng ta đã triển khai và phân tích thuật toán Minimax cùng cải tiến Alpha-Beta pruning cho trò chơi Tic-Tac-Toe. Các mức độ khó được xây dựng dựa trên việc điều chỉnh độ sâu tìm kiếm và tỉ lệ ngẫu nhiên, từ đó tạo nên trải nghiệm đa dạng cho người chơi.

Kết quả thực nghiệm cho thấy:

- Khi đối thủ là **random**, AI dễ dàng giành chiến thắng ở mức **hard** và **impossible**.
- Khi đối thủ là **optimal**, AI chỉ có thể hòa ở mức độ cao, điều này phản ánh tính chất cân bằng của trò chơi.

5.1.2. Hạn chế

- Thuật toán Minimax không học từ dữ liệu, nên mở rộng sang không gian lớn sẽ rất chậm nếu thiếu heuristic.
- Giao diện còn đơn giản, chưa đo được thời gian chi tiết cho từng nước đi.

5.1.3. Hướng phát triển

- Mở rộng sang trò chơi phức tạp hơn (Connect Four, Gomoku) với các heuristic để giảm độ phức tạp.
- Thử nghiệm phương pháp học tăng cường (Reinforcement Learning).
- Tích hợp giao diện đồ họa trực quan và API để đánh giá tự động.

TÀI LIỆU THAM KHẢO

[1] Le, V.T., Kim, Y.G., 2023. **Attention-based residual autoencoder for video anomaly detection**. Applied Intelligence 53, 3240–3254.

[2] Tác giả Clederonbc – Github: https://github.com/Cledersonbc/tic-tac-toe-minimax

[3] Các Công cụ AI bổ trợ (ChatGPT – Gemini)