**Evaluating the effects of the evaluation functions on the Go games**

Võ Quốc Huy

Nghiêm Văn Tiến

# Abstract

Bài báo cáo này tập trung đánh giá sự khác biệt về hiệu năng và kết quả khi sử dụng 2 hàm đánh giá (stone count và liberty count) trong thuật toán minimax với cắt tỉa alpha- beta để giải quyết bài toán cờ GO.

**Index Terms—** One, two, three, four, five

# Introduction

* 1. **Giới thiệu bài toán**

Cờ GO là một trò chơi chiến thuật cổ xưa có nguồn gốc từ Trung Quốc, hiện được chơi phổ biến trên toàn thế giới. Trò chơi diễn ra trên một bàn cờ gồm lưới vuông 19x19 (có thể nhỏ hơn như 9x9 hoặc 13x13 cho người mới bắt đầu), trong đó hai người chơi lần lượt đặt các quân cờ đen và trắng lên các giao điểm với mục tiêu kiểm soát nhiều lãnh thổ hơn đối thủ. Mỗi khi một quân cờ hoặc nhóm quân cờ của đối thủ bị bao vây hoàn toàn (tức không còn “khí” – các điểm tự do xung quanh), chúng sẽ bị bắt và loại khỏi bàn cờ. Trò chơi kết thúc khi cả hai người chơi đều đồng ý không tiếp tục đặt thêm quân, và người chiến thắng là người kiểm soát được nhiều lãnh thổ nhất sau khi trừ đi số quân bị bắt.

Tuy luật chơi đơn giản, nhưng không gian trạng thái khổng lồ của trò chơi khiến việc giải bài toán cờ GO bằng máy tính trở thành một thách thức lớn. Một trong những chiến lược phổ biến để giải bài toán này là sử dụng thuật toán minimax với cắt tỉa alpha-beta, kết hợp với các hàm đánh giá trạng thái như số quân (stone count) hoặc số điểm tự do (liberty count). Vấn đề đặt ra là: *liệu việc lựa chọn hàm đánh giá có ảnh hưởng đáng kể đến hiệu quả và kết quả của thuật toán minimax trong việc ra quyết định hay không?*

* 1. **Tại sai chúng ta cần giải quyết bài toán?**

Việc lựa chọn hàm đánh giá trong thuật toán minimax đóng vai trò then chốt trong hiệu quả của quá trình tìm kiếm lời giải cho các trò chơi chiến lược như cờ GO. Một hàm đánh giá tốt không chỉ giúp tiết kiệm tài nguyên tính toán mà còn nâng cao chất lượng quyết định, đặc biệt trong những tình huống mà không thể xét hết toàn bộ không gian trạng thái do giới hạn độ sâu tìm kiếm.

Trong bối cảnh này, việc so sánh hai hàm đánh giá phổ biến – **stone count** (đếm số quân) và **liberty count** (đếm số điểm tự do quanh quân cờ) – có ý nghĩa thực tiễn và học thuật rõ ràng. Stone count thể hiện mức độ kiểm soát vật lý của người chơi trên bàn cờ, trong khi liberty count phản ánh tiềm năng sống sót và linh hoạt trong chiến lược. Việc phân tích hiệu quả của hai hàm này giúp hiểu rõ hơn cách máy tính “nhìn nhận” một trạng thái cờ, từ đó hướng tới xây dựng các chiến lược đánh giá phức tạp hơn hoặc thậm chí học được từ dữ liệu.

Trong lĩnh vực trí tuệ nhân tạo và khoa học máy tính, đây là một ví dụ điển hình về vấn đề đánh đổi giữa **đơn giản và hiệu quả** trong thiết kế hàm heuristic. Kết quả nghiên cứu không chỉ giúp cải thiện các thuật toán chơi cờ, mà còn đóng góp cho các ứng dụng rộng hơn của minimax và các thuật toán tìm kiếm heuristic trong các bài toán ra quyết định có không gian trạng thái lớn như lập kế hoạch, robot di chuyển, hoặc phân tích chiến lược trong hệ thống đa tác tử.

* 1. **Những việc nhóm đã làm**

Trong khuôn khổ báo cáo này, chúng tôi đã xây dựng một thuật toán minimax kết hợp với cắt tỉa alpha-beta dựa trên các kiến thức đã được học trong môn học tí tuệ nhân tạo, nhằm giải bài toán ra quyết định trong trò chơi cờ GO. Thuật toán được hiện thực hóa bằng ngôn ngữ lập trình Python, trong đó chúng tôi triển khai hai phiên bản khác nhau dựa trên hai lớp đánh giá có tên lần lượt là **StoneCountEvaluator** và **LibertyCountEvaluator**. StoneCountEvaluator đánh giá trạng thái dựa trên số lượng quân cờ mà mỗi bên đang kiểm soát, trong khi LibertyCountEvaluator dựa trên số điểm tự do (liberties) quanh các quân cờ – phản ánh khả năng sống sót và tính linh hoạt chiến thuật.

Sau khi xây dựng hoàn chỉnh, chúng tôi đã tiến hành các bài kiểm thử (benchmark) để đánh giá và so sánh hiệu quả hoạt động của hai hàm đánh giá trong thực tế. Các chỉ số được đo lường bao gồm: **tỉ lệ thắng** trong các ván đấu thử nghiệm, **thời gian chạy trung bình mỗi bước**, **bộ nhớ sử dụng ở mỗi bước**, và **tổng bộ nhớ tiêu thụ** trong toàn bộ ván đấu. Kết quả thu được giúp chúng tôi phân tích ưu nhược điểm của từng hàm đánh giá trong bối cảnh thuật toán minimax giới hạn độ sâu, đồng thời cung cấp cái nhìn rõ ràng hơn về ảnh hưởng của hàm heuristic đến hiệu suất và chất lượng của hệ thống ra quyết định trong các trò chơi có không gian trạng thái lớn như cờ GO.

# 2. METHOD

Trong nghiên cứu này, chúng tôi sử dụng thuật toán **Minimax kết hợp cắt tỉa Alpha-Beta** để tối ưu hóa quá trình ra quyết định trong trò chơi cờ GO. Minimax là một thuật toán ra quyết định cổ điển trong các trò chơi hai người, nơi người chơi này cố gắng **tối đa hóa điểm số** trong khi giả định rằng đối phương sẽ **tối thiểu hóa** điểm số của họ.

Tuy nhiên, do không gian trạng thái của cờ GO rất lớn, việc duyệt toàn bộ cây trạng thái là bất khả thi về mặt thời gian và bộ nhớ. Do đó, chúng tôi tích hợp **cắt tỉa Alpha-Beta** vào thuật toán minimax để loại bỏ những nhánh không cần thiết, giúp giảm đáng kể số lượng trạng thái cần đánh giá mà không ảnh hưởng đến kết quả cuối cùng.

**Nguyên lý hoạt động:**

* **Alpha**: là giá trị tốt nhất hiện tại mà người chơi MAX có thể đảm bảo.
* **Beta**: là giá trị tốt nhất hiện tại mà người chơi MIN có thể đảm bảo.
* Trong quá trình duyệt cây, nếu tại một nút MIN, giá trị nhỏ hơn Alpha xuất hiện, thì MIN sẽ không chọn nhánh này → **cắt tỉa**.
* Tương tự, tại nút MAX, nếu xuất hiện giá trị lớn hơn Beta → cũng bị cắt.

Pseudocode thuật toán Minimax Alpha-Beta:

*def alphabeta(node, depth, alpha, beta, maximizingPlayer):*

*if depth == 0 or node is terminal:*

*return evaluate(node)*

*if maximizingPlayer:*

*value = -infinity*

*for child in node.children:*

*value = max(value, alphabeta(child, depth-1, alpha, beta, False))*

*alpha = max(alpha, value)*

*if alpha >= beta:*

*break # Beta cut-off*

*return value*

*else:*

*value = +infinity*

*for child in node.children:*

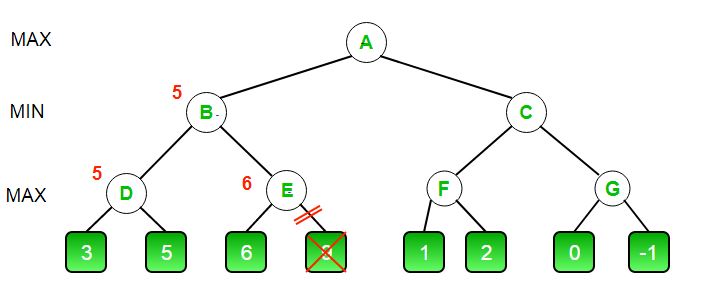
*value = min(value, alphabeta(child, depth-1, alpha, beta, True))*

*beta = min(beta, value)*

*if beta <= alpha:*

*break # Alpha cut-off*

*return value*



**Định giá trạng thái và vai trò của hàm ước lượng (Evaluation Function)**

Trong các trò chơi có không gian trạng thái lớn như cờ GO, việc duyệt đến trạng thái kết thúc là không khả thi ở mỗi bước do giới hạn thời gian và tài nguyên. Do đó, thay vì đợi đến khi trò chơi kết thúc để đánh giá thắng-thua, chúng tôi cần một **hàm ước lượng (evaluation function)** để định giá tạm thời cho mỗi trạng thái khi duyệt đến một độ sâu nhất định trong cây tìm kiếm.

Giá trị mà thuật toán minimax sử dụng tại các nút lá hoặc tại độ sâu cắt là **giá trị do hàm ước lượng trả về**. Với người chơi MAX, thuật toán sẽ cố gắng chọn nước đi mang lại giá trị lớn nhất; ngược lại, với người chơi MIN, thuật toán chọn nước đi có giá trị nhỏ nhất.

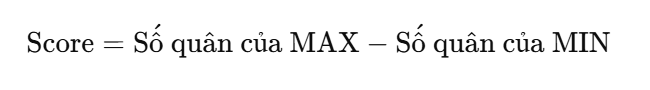
Việc thiết kế một hàm ước lượng hiệu quả đóng vai trò then chốt trong chất lượng của quyết định được đưa ra. Một hàm tốt sẽ phản ánh chính xác lợi thế chiến lược tại thời điểm hiện tại và dẫn đến những nước đi khôn ngoan hơn.

**Hai hàm ước lượng được sử dụng trong nghiên cứu**

Trong nghiên cứu này, chúng tôi thiết kế và triển khai hai hàm ước lượng đơn giản nhưng có ý nghĩa rõ ràng trong ngữ cảnh của trò chơi cờ GO:

**StoneCountEvaluator**  
Hàm này ước lượng giá trị của một trạng thái dựa trên **sự chênh lệch số quân cờ trên bàn** giữa hai người chơi:

Hàm này phản ánh mức độ kiểm soát vật lý của người chơi đối với bàn cờ.



**LibertyCountEvaluator**  
Hàm này đánh giá dựa trên tổng số **điểm tự do (liberties)** mà các quân cờ của mỗi bên đang sở hữu:

Liberty là các ô trống xung quanh một quân cờ hoặc nhóm quân cờ, quyết định việc quân đó còn “sống” hay có nguy cơ bị bắt. Do đó, hàm này đánh giá tính “sống còn” và tiềm năng phát triển thế trận trong tương lai.



**Tổng kết vai trò**

Cả hai hàm đều hướng tới cung cấp một ước lượng tạm thời tại mỗi trạng thái để thuật toán minimax đưa ra quyết định phù hợp. Trong khi **StoneCountEvaluator** đơn giản và nhanh, thì **LibertyCountEvaluator** phản ánh chiến lược sâu sắc hơn. Việc so sánh hiệu quả của hai hàm này là trọng tâm chính của nghiên cứu này.

# 3. Experimental results and analysis

This section shows:

* Dataset(s),
* Experimental setup,
* Evaluation methods,
* Experimental results,
* Analysis/discussion of the results.

# 4. Conclusion

This section concludes what has been done in this report. The advantages, disadvantage, and the future direction also are shown in this section.

# 5. References

[1] A.B. Smith, C.D. Jones, and E.F. Roberts, “Article Title,” *Journal*, vol. <volume>, no. <issue number>, pp. <page range>, Month Year.

[2] C.D. Jones, A.B. Smith, and E.F. Roberts, “Paper Title,” in *Proceedings Title*, Publisher, Location, Year, vol. <volume>, pp. <page range>.

Đàasdd

Dsfa

Ads

f

sdf

á

f

à

s

f

sd

f

á

df

sfasf

sà

sd

f

sa

df

d

d

d

d

d

d

d

d

d