**CHƯƠNG TRÌNH AI CHO MÁY TÍNH CHƠI GAME 8-PUZZLE VỚI NGƯỜI**

**Nguyễn Ngọc Minh Thư**

**Cao Thị Thùy Linh**

**TS Huỳnh Xuân Phụng**

*Trường đại học Sư phạm Kỹ thuật TP.HCM*

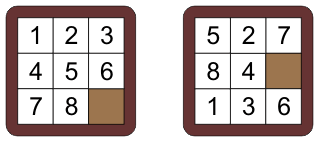
**TÓM TẮT**

*Qua bao thế hệ, con người vẫn bị hấp dẫn bởi những trò chơi mang tính đối kháng và vận may, hiện nay những dòng game với cấu hình khủng phục vụ nhu cầu giải trí thỏa mãn những tiêu chí trên hàng loạt ra mắt (game pvp, game cá cược,…) và cùng với đó là sự phát sinh hàng loạt mặt tối của trò chơi điện tử (nghiện game, vỡ nợ do thua cược,…). Sau quá trình nghiên cứu và tìm hiểu, nhóm chúng em muốn thiết kế ra một tựa game giải trí tuy đơn giản nhưng giúp chúng ta thư giãn sau những giờ học tập, làm việc căng thẳng mà không gây hại và đặc biệt vừa có tính đối kháng vừa thử thách vận may, đó là tựa game 8-puzzle đã được cải tiến: người chơi với máy, tung đồng xu để quyết định lượt đi. 8-puzzle là một loại hình game giải đố (3x3) quen thuộc, người chơi sẽ di chuyển ô trống để xếp các hình hoặc con số về trạng thái hoàn chỉnh đã được cho trước. Đây là trò chơi cần sử dụng trí tuệ và sẽ gây nhàm chán khi chơi một mình. Vì vậy chúng em đã cải tiến để tăng tính hấp dẫn và sự lôi cuốn cho người chơi bằng lối chơi đối kháng người với máy, máy sẽ có trí thông minh vượt bậc bằng thuật toán A\* với hàm đánh giá heuristic cho phép máy tính tìm ra giải pháp hợp lí để chiến thắng, thêm vào đó là màn thử thách vận may mỗi lượt sẽ “tung đồng xu”, tung ra mặt ngửa thì được đi, mặt sấp thì đối phương tung đồng xu. Kết quả sau những lần thử nghiệm với người chơi cho thấy máy tính nhờ thuật toán A\* trở nên đủ thông minh và tạo ra thử thách lớn để người chơi chiến thắng nhưng nhờ việc tung đồng xu mà xác suất chiến thắng là không thể dự đoán được làm cho người chơi hứng thú muốn tiếp tục “thử thách”.*

***Từ khóa:*** *8-puzzle; A\*; heuristic; đối kháng; vận may*

1. **GIỚI THIỆU TRÒ CHƠI 8-PUZZLE CẢI TIẾN**
2. **Giới thiệu về trò chơi 8-puzzle**

Trò chơi gồm một bảng 3x3 với các ô số được đánh từ 1->8, ô đánh số 0 sẽ là ô trống dùng để di chuyển. Ở trạng thái bắt đầu, các ô được sắp đặt ngẫu nhiên và nhiệm vụ của người chơi là tìm cách di chuyển các ô sao cho các con số về đúng thứ tự. Tại một trạng thái thì chỉ có tối đa 4 cách để di chuyển sang trạng thái khác (trái, phải, lên, xuống). Điều đầu tiên cần phải quan tâm để chiến thắng trò chơi này đó là xác định trạng thái đích, trạng thái đích được xác định dựa trên trạng thái đầu.



1. **Giới thiệu thuật toán A\***

 Trong khoa học máy tính, **A\*** (đọc là *A sao*) là một [thuật toán tìm kiếm trong đồ thị](https://vi.wikipedia.org/wiki/Duy%E1%BB%87t_c%C3%A2y). Thuật toán này tìm một đường đi từ một [nút](https://vi.wikipedia.org/w/index.php?title=N%C3%BAt_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)&action=edit&redlink=1) khởi đầu tới một nút đích cho trước (hoặc tới một nút thỏa mãn một điều kiện đích). Thuật toán này sử dụng một "đánh giá heuristic" để xếp loại từng nút theo ước lượng về tuyến đường tốt nhất đi qua nút đó. Thuật toán này duyệt các nút theo thứ tự của đánh giá heuristic này. Do đó, thuật toán A\* là một ví dụ của [tìm kiếm theo lựa chọn tốt nhất](https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_l%E1%BB%B1a_ch%E1%BB%8Dn_t%E1%BB%91t_nh%E1%BA%A5t) (*best-first search*).

Thuật toán A\* được mô tả lần đầu vào năm 1968 bởi [Peter Hart](https://vi.wikipedia.org/w/index.php?title=Peter_Hart&action=edit&redlink=1), [Nils Nilsson](https://vi.wikipedia.org/w/index.php?title=Nils_Nilsson&action=edit&redlink=1), và [Bertram Raphael](https://vi.wikipedia.org/w/index.php?title=Bertram_Raphael&action=edit&redlink=1). Trong bài báo của họ, thuật toán được gọi là thuật toán A; khi sử dụng thuật toán này với một đánh giá heuristic thích hợp sẽ thu được hoạt động tối ưu, do đó mà có tên A\*.

Năm 1964, Nils Nilsson phát minh ra một phương pháp tiếp cận dựa trên khám phá để tăng tốc độ của thuật toán Dijkstra. Thuật toán này được gọi là A1. Năm 1967 Bertram Raphael đã cải thiện đáng kể thuật toán này, nhưng không thể hiển thị tối ưu. Ông gọi thuật toán này là A2. Sau đó, trong năm 1968 Peter E. Hart đã giới thiệu một đối số chứng minh A2 là tối ưu khi sử dụng thuật toán này với một đánh giá heuristic thích hợp sẽ thu được hoạt động tối ưu. Chứng minh của ông về thuật toán cũng bao gồm một phần cho thấy rằng các thuật toán A2 mới là thuật toán tốt nhất có thể được đưa ra các điều kiện. Do đó ông đặt tên cho thuật toán mới là A \*(A sao, A-star).[1]

Sử dụng thuật toán A\* bởi vì nó sẽ luôn luôn tìm thấy một lời giải nếu bài toán có lời giải.

- Mã giả:

Begin

Khởi tạo danh sách frontier chỉ chứa trạng thái đầu;

For (int action=0; action <4; action++)

{

If hành động được chọn là đúng {

if node vừa di chuyển không nằm trong frontier

{

Thêm node vừa kiểm tra vào frontier

if Path\_cost lớn hơn chi phí của node vừa kiểm tra // Path\_cost >chi phí hiện tại

then chi phí của node vừa kiểm tra = Path\_cost và

lưu hành động vừa di chuyển

}

}

print bước di chuyển tối ưu

return node đã lưu

}

1. **Giới thiệu lối chơi cải tiến**

Chương trình máy tính chơi với người bằng cách tung đồng xu để xác định người hay máy được đi:

* Cách chơi: Mỗi lần sẽ tung đồng xu để chọn, nếu mặt ngửa thì(1) được đi, mặt sấp(0) thì đối phương tung xu.
* Điểm mới lạ hấp dẫn của lối chơi này là: bằng cách tung xu một cách ngẫu nhiên sẽ không biết trước người hay máy tính sẽ giành chiến thắng. Có những trường hợp đặc biệt như máy sẽ luôn tung được mặt ngửa từ đó sẽ chỉ có máy đi và giành chiến thắng.

1. **Hàm heuristic**

Ðó là các hàm đánh giá thô, giá trị của hàm phụ thuộc vào trạng thái hiện tại của bài toán tại mỗi bước giải. Nhờ

giá trị này, ta có thể chọn được cách hành động tương đối hợp lý trong từng bước của thuật giải. [2]

1. **PHƯƠNG PHÁP**

Để áp dụng thuật toán A\* giải bài toán này, bạn cần một hàm Heuristic h để ước lượng giá trị của mỗi giá trị của mỗi trạng thái của bảng số. Trong bài này tôi sử dụng cách là tính tổng số ô sai lệch của các ô đúng vị trí của nó.

1. **Ý tưởng:**

Thuật toán A\*:

* Gọi g là số bước đã di chuyển (trong bài này không cần thiết)
* h là hàm heuristic, ước tích số hao tổn để tới trạng thái tốt hơn, tính bằng tổng của các ô ở sai vị trí để tới trạng thái tốt hơn.

f(n) = g(n) + h(n)

Có danh sách frontier chứa các trạng thái đã xét.

Ban đầu ta thêm trạng thái Start vào frontier, sau đó di chuyển ô trống sang vị trí khác. Nếu di chuyển được thì cho các trạng thái mới di chuyển được vào frontier và chọn trạng thái có f=g+h nhỏ nhất. Trạng thái đó chính là trạng thái máy di chuyển. Khi máy di chuyển lần tiếp theo thì danh sách frontier sẽ được làm mới.

1. **Cách tính tổng số ô sai lệch của các ô đúng vị trí của nó:**

For (int i=0; i<3; i++)

{

For (int j=0; j<3; j++)

{

If giá trị hiện tại ở ô đang xét khác 0 và khác giá trị đúng của vị ô đang xét

tăng biến đếm một đơn vị

}

}

Retrun biến đếm

1. **KẾT LUẬN**

Trong những trường hợp khác nhau thì thời gian máy di chuyển sẽ khác nhau tùy thuộc vào trường hợp đó phức tạp hay đơn giản. Thời gian trung bình mỗi lần máy di chuyển là khoảng 0.005s.

Thông qua việc tìm hiểu và nghiên cứu đề tài giúp chúng em có cái nhìn toàn diện hơn trong việc ứng dụng trí tuệ nhân tạo vào giải quyết vấn đề thực tế. Đây là trò chơi mang phong cách bài toán cổ điển trong trí tuệ nhân tạo cho các thuật toán mô hình hóa liên quan đến tìm kiếm có tri thức bổ sung. Đề tài có phần cải tiến giúp tăng tính sáng tạo và cải thiện lối chơi bớt phần nhàm chán, tăng tính tối ưu cho trò chơi tuy nhiên vẫn còn nhiều hạn chế trong thuật toán vì chưa có cách giải quyết tối ưu cho tất cả các không gian trạng thái. Hy vọng những nghiên cứu đánh giá của chúng em sẽ góp phần cải thiện và tăng tính tiếp cận cho trò chơi. Do thời gian có hạn nên đề tai không tránh khỏi những sai sót, mong thầy góp ý, đánh giá giúp chúng em hoàn thiện đề tài.

**TÀI LIỆU THAM KHẢO**

[1][2] <https://vi.wikipedia.org/wiki/Heuristic>