**CHƯƠNG TRÌNH ĐÁNH GIÁ CÁC THUẬT TOÁN LOCAL SEARCH CHO GAME 8-PUZZLE**

Sinh viên thực hiện

**Đặng Minh Khang 18110130**

**Thiều Văn Hoàng 18110114**

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

**TÓM TẮT**

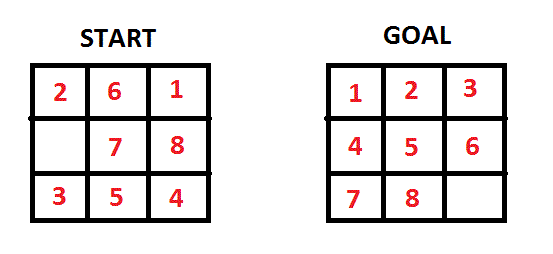
**Metaheuristic** là cách gọi chung cho các giải thuật heuristic trong việc giải quyết các bài toán tổ hợp khó. Metaheuristic bao gồm những chiến lược khác nhau trong việc khám phá không gian tìm kiếm bằng cách sử dụng những phương thức khác nhau và đạt được sự cân bằng giữa tính đa dạng và chuyên sâu của không gian tìm kiếm.Những ví dụ của Mettaheuristic bao gồm giải thuật Hill Climbing, giải thuật di truyền (GA), Simulated Annealing (SA),… Sau quá trình nghiên cứu và tìm hiểu, nhóm chúng em muốn thiết kế ra một tựa game giải trí tuy hại và đặc biệt vừa có tính đối kháng vừa thử thách vận may, đó là tựa game 8-puzzle được áp dụng các thuật toán Metaheuristic để đánh giá nhưng game cũng giúp chúng ta thư giãn sau những giờ học tập, làm việc căng thẳng. Mục đích thứ hai cũng là muốn hoàn thành yêu cầu của project cuối kì của môn học nhằm áp dụng được kiến thức đã học trên lớp cũng như các kiến thức khác. Bằng cách tạo ra một số lượng lớn trạng thái ban đầu của game và một trạng thái đích để giải, áp dụng thuật toán Hill Climbing, Random Restart Hill Climbing và Simulated Annealing để giải quyết vấn đề và xuất ra chi phí, số trường hợp giải được để từ đó so sánh cả 3 thuật toán này. Sau khi so sánh thì nhóm em rút ra kết luận về mức độ hiệu quả của thuật toán đối với game 8-puzzle.

Từ khóa**:** 8-puzzle local search; 8-puzzle Algorithm; 8-puzzle AI using heuristic; Heuristic Algorithm

**I. ĐẶT VẤN ĐỀ**

Bài toán (hay game) n-puzzle có lẽ rất quen thuộc với chúng ta cũng như những người mới bắt đầu tiếp cận với môn trí tuệ nhân tạo. Nó được biết đến với nhiều phiên bản và tên gọi khác nhau như 8-puzzle, 15-puzzle, Gem puzzle, Boss puzzle, Game of Fifteen, Mystic Square,… Ở mức độ đơn giản tôi xin nói về 8-puzzle.

Có n2-1 số mang các giá trị từ 1 tới n2-1 được sắp xếp vào một lưới các ô vuông kích thước n x n (Ở bài toán này là 3x3). Mỗi số đó được gọi là một quân cờ và lưới ô đó được gọi là bàn cờ. Có một vị trí của bàn cờ bỏ trống. Mỗi lần di chuyển quân, người chơi được phép chuyển một quân ở vị trí ô tiếp giáp cạnh với ô trống vào ô trống.Từ một trạng thái ban đầu (sự sắp xếp ban đầu của các quân trên bàn cờ), thực hiện các nước đi hợp lệ để thu được trạng thái kết thúc (trạng thái đích cần đạt được).



*Hình 1. 8-puzzle*

Chức năng đánh giá dF tính toán tổng của các di chuyển cần thiết cho mỗi ngăn xếp để đạt đến trạng thái cuối cùng của nó. Kể từ khi gạch 1 đến 7 đã ở vị trí chính xác của nó, họ không cần phải được di chuyển. Tuy nhiên, gạch 8 là 1 di chuyển ra khỏi vị trí cuối cùng của nó.

dF(8) = m(1)+m(2)+m(3)+m(4)+m(5)+m(6)+m(7)+m(8) = 1

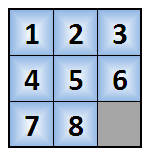
Uớc lượng các di chuyển tiếp theo có thể và chọn một trong đó có khoảng cách ít nhất. Nó cũng kiểm tra nếu nhà nước mới sau khi di chuyển đã được quan sát. Nếu đúng, sau đó nó bỏ qua di chuyển và chọn di chuyển tốt nhất tiếp theo. Khi gạch bỏ trống chỉ có thể được lấp đầy bởi những người hàng xóm xung quanh của mình. Ý tưởng là đi qua một con đường cho một số bước xác định (chiều sâu) để xác nhận rằng đó là di chuyển tốt nhất. Quan sát thấy chiều sâu đầu tiên cách tiếp cận cải thiện hiệu quả tổng thể đạt đến trạng thái cuối cùng. Tuy nhiên, bộ nhớ của nó chuyên sâu, tỷ lệ thuận với giá trị độ sâu được sử dụng. Điều này là do hệ thống có để theo dõi các quốc kỳ tương lai theo chiều sâu được sử dụng. Vậy bằng cách sử dụng hàm heuristic để tính chi phí thấp nhất (đồng thời là bước di chuyển tốt nhất) để giải quyết vấn đề bài toán game 8-puzzle bằng 3 thuật toán khác nhau để đưa ra kết luận cuối cùng cho bài toán thì nhóm em xin trình bày ở phía dưới.

**II. PHƯƠNG PHÁP**

**2.1. Dữ liệu**

8-puzzle sử dụng dữ liệu đầu vào với không gian trạng thái là (3x3)! = 362880 với một chuỗi các số từ 0-8 gồm 4 bước đi là sang trái, sang phải, đi lên và đi xuống bằng cách di chuyển số 0. Có những trạng thái ban đầu người chơi sử dụng không thể đi đến trạng thái đích. Điều kiện để trạng thái ban đầu đi đến trạng thái đích là với mỗi ô trong 8-puzzle, xét theo chiều từ trái sang phải và từ trên xuống dưới, tổng số ô có giá trị nhỏ hơn nó theo sau phải là số chẵn.

Ví dụ với trạng thái đích là {1, 2, 3, 4, 5, 6, 6, 7, 8, 0}



Hình 2. 8-puzzle trạng thái đích

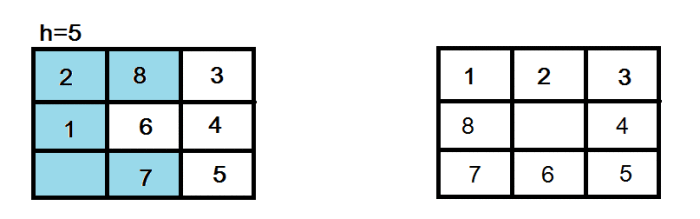
Bằng cách tạo và sử dụng hàm RandomMang để thực hiện Random ra nhiều trạng thái ban đầu để giải quyết và đánh giá chuẩn xác mức độ hiểu quả của từng thuật toán đối với bài toán 8-puzzle.

**2.2. Thực thi từng thuật toán**

**1. Hill Climbing (steepest-ascent and first-choice variant)**

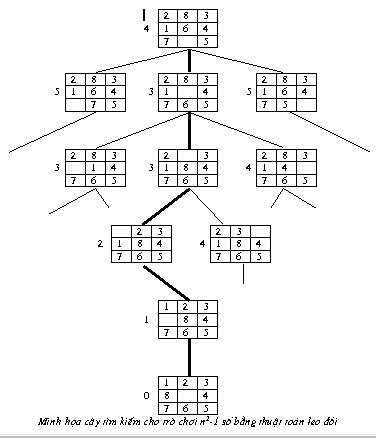
Từ trạng thái ban đầu, ta tìm ra các trạng thái con bằng cách đi tất cả các nước đi có thể. Để chọn trạng thái tiếp theo, ta sử dụng một hàm đánh giá cho từng trạng thái. Ở đây, ta sẽ dùng hàm h tính bằng số ô khác với trạng thái đích. Theo đó, trạng thái nào có h min thì sẽ được chọn.

Các tính h như trong hình. Các ô tô màu xanh là những ô có giá trị khác với trạng thái đích. Ô ở giữa ở trạng thái đích không có giá trị nên ta sẽ không so sánh với nó.



*Hình 3.Tính giá trị heuristic*

Quá trình giải diễn ra như hình bê dưới. Đường đi được tìm thấy là đường được bôi đậm.



*Hình 4. Giải quyết vấn đề bằng Hill Climbing*

## *Mã giả:*

begin

Gọi hàm kiểm tra mảng goal (checkGoal)

Nếu giống goal trả về 8

Nếu không chưa đạt goal trả về <8

Chạy vòng lặp while (checkGoal<8)

Gọi lại những hàm con sau khi mảng đã thay đổi để tính toán lại.

Nếu costT lớn nhất then current 🡸 x // swap lên top

Nếu costL lớn nhất then current 🡸 x // swap qua trái

Nếu costR lớn nhất then current 🡸 x // swap qua phải

Nếu costD lớn nhất then current 🡸 x // swap xuống dưới

Ngược lại không thỏa thì sẽ thoát khỏi vòng lặp

end.

1. **Random Restart Hill Climbing**

## *Mã giả:*

begin

Khởi tạo trạng thái ban đầu: k=0;

Tạo mảng 2 chiều chứa mảng ban đầu.

Vòng lặp while(k>=0 && k<=4)

Gọi lại những hàm con sau khi mảng đã thay đổi để tính toán lại.

Nếu k =0 Gọi lại hàm hillclimbing giá trị trả về sẽ là mảng 2 chiều và giá trị check bước đầu tiên mà nó di chuyển với giá trị 1,2,3 hoặc 4. Gọi hàm checkGoal kiểm tra có phải đích chưa nếu chưa thì quay lại mảng ban đầu.

Nếu k=1 và check khác 1 then Current 🡸 top

Gọi lại hàm hillclimbing giá trị trả về sẽ là mảng 2 chiều và giá trị. Gọi hàm checkGoal kiểm tra có phải đích chưa nếu chưa thì quay lại mảng ban đầu.

Nếu k=2 và check khác 2 then Current 🡸 left

Gọi lại hàm hillclimbing giá trị trả về sẽ là mảng 2 chiều và giá trị. Gọi hàm checkGoal kiểm tra có phải đích chưa nếu chưa thì quay lại mảng ban đầu.

Nếu k=3 và check khác 3 then Current 🡸 right

Gọi lại hàm hillclimbing giá trị trả về sẽ là mảng 2 chiều và giá trị. Gọi hàm checkGoal kiểm tra có phải đích chưa nếu chưa thì quay lại mảng ban đầu.

Nếu k=4 và check khác 4 then Current 🡸 down

Gọi lại hàm hillclimbing giá trị trả về sẽ là mảng 2 chiều và giá trị. Gọi hàm checkGoal kiểm tra có phải đích chưa nếu chưa thì quay lại mảng ban đầu.

Giá trị k tăng 1 sau mỗi lần lặp

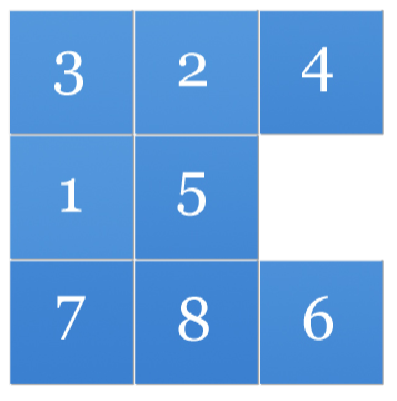
end.

Bản chất của thuật toán này là Hill Climbing nhưng nó đã được cải tiến bằng việc nếu thuật toán Hill Climbing đơn giản không giải quyết được vấn đề của bài toán thì nó sẽ khởi động lại ngẫu nhiên một trạng thái địa phương gần với trạng thái ban đầu để tiếp tục giải quyết theo một hướng khác và cứ thế tiếp túc nếu gặp vấn đề tương tự. Ở thuật toán này nhóm em đã đặt ra một giới hạn duyệt độ sâu để ngắt thuật toán nhằm giải quyết vòng lặp vô hạn khi thực thi.

1. **Simulated Annealing**

Là một [kỹ thuật xác suất](https://en.wikipedia.org/wiki/Probabilistic_algorithm) để xấp xỉ mức [tối ưu toàn cầu](https://en.wikipedia.org/wiki/Global_optimum) của một [hàm](https://en.wikipedia.org/wiki/Function_(mathematics)) nhất định . Cụ thể, đó là một [siêu dữ liệu](https://en.wikipedia.org/wiki/Metaheuristic) để xấp xỉ [tối ưu hóa toàn cầu](https://en.wikipedia.org/wiki/Global_optimization) trong một [không gian tìm kiếm](https://en.wikipedia.org/wiki/Solution_space) lớn cho một [vấn đề tối ưu hóa](https://en.wikipedia.org/wiki/Optimization_problem) . Nó thường được sử dụng khi không gian tìm kiếm rời rạc.

Giả sử ta có trạng thái 8-puzzle như thế này :



*Hình 5. Trạng thái ban đầu của 8-puzzle*

 Điều duy nhất có vẻ khó khăn là tìm ra cách mô tả chi phí giải quyết (khoảng cách của trạng thái từ trạng thái đã giải quyết bằng hàm Manhattan). Bạn phải có trạng thái được giải có chi phí thấp nhất trong hệ thống, và thật tuyệt khi có trạng thái gần như được giải quyết có chi phí thấp. Một chi phí hợp lý sẽ là tổng khoảng cách của tất cả các ô từ vị trí đã giải quyết của chúng. Nói cách khác, số lần di chuyển sẽ cần để giải câu đố nếu chúng ta không bị giới hạn chỉ một ô trên mỗi ô. Một chi phí hợp lý khác sẽ là số lượng gạch không đúng chỗ.

## *Mã giả:*

begin

Khởi tạo trạnh thái ban đầu: T=100, t=0, p;

Chon ngẫu nhiên y

T giảm 1 trong mỗi lần di chuyển

Cho chạy vòng lập (t < 100)

Tính denta(y,x) = Costy – costx

Nếu denta(y,x) <0 then p=1

Ngược lại p= exp(-denta(x,y))/T

Kiểm tra vị trí hiện tại là tốt nhất

Nếu P trong khoản được nhận và Costx > Costx\* then x<-y

Ngược lại Costx < Costx\* then x🡸x

return x

end.

**2.3. Xử lý dữ liệu đầu vào**

Với trạng thái đầu và trạng thái đích, thì các số nhập vào phải thõa mãn từ 0 đến 8 và không có số trùng nhau. Với việc chạy các thuật toán khác nhau nên phần lớn sẽ không xuất ra giống nhau và các thuật toán có khả năng xử lý trạng thái ban đầu đến trạng thái đích tùy vào điều kiện xử lí chi phí.

**3. KẾT QUẢ**

Nhóm chúng em đã cho chương trình tiến hành chạy độc lập các thuật toán để tìm đường đi tới đích , nhờ vào những bước đi tốt nhấ bằng chi phi heuristic thấp nhất để di chuyển. Chương trình hoàn thành đa số các trạng thái đầu vào, tuy nhiên ở một vài trạng thái, chương trình sẽ không thể tìm thấy lời giải do chi phí quá lớn hoặc không tìm được chi phí đích h(n) = 0.

Xuất ra các trạng thái ban đầu và số bước đi (chi phí) và số trạng thái mà từng thuật toán giải được -> So sánh số lượng trạng thái giải được và thời gian thực thi để đánh giá từng loại thuật toán

|  |  |  |
| --- | --- | --- |
| Họ và tên | Công việc | Mức độ thực hiện |
| Đặng Minh Khang | - Lên ý tưởng cho bài toán  - Thuật toán Hill climbing và mã giả  - Thuật toán Restart Random và mã giả  - Bổ sung báo cáo | 50% |
| Thiều Văn Hoàng | - Tìm tại liệu tham khảo  - Sườn source code và random mảng  - Thuật toán Simulated Annaling và mã giả  - Viết báo cáo | 50% |

*Hình 6. Bảng phân công nhiệm vụ*

**4. KẾT LUẬN**

**4.1. Đánh giá:**

- Hill Climbing:

+ Thuật toán chỉ la phương pháp giải quyết cục bộ, hiệu quả, đơn giản, thời gian chạy nhanh.

+ Tuy nhiên thuật toán không tối ưu với bài toán 8-puzzle vì trong khi thực hiện có nhiều vấn đề nảy sinh như: Gặp trạng thái bằng phẳng, đỉnh chóp đa số là trạng thái tối ưu cục bộ,…

- Random Restart Hill Climbing:

+ Thời gian chạy có phần lâu hơn so với Hill Climbing đơn giản.

+ Xác suất có thể tìm ra được trạng thái tối ưu toàn cục cao nếu độ sâu tìm kiếm rộng bằng cách chọn ngẫu nhiên một trạng thái gần với trạng thái tối ưu cục bộ để tiếp tục giải.

+ Có khả năng không tìm thấy trạng thái tốt hơn trạng thái ban đầu sau khi đặt ngẫu nhiên trạng thái khác vì không thể lưu tất cả các trạng thái.

- Simulated Annaling:

+ Có thể áp dụng nhiều bài toán, hệ thống khác nhau tùy vào hàm tính chi phí.

+ Code tương đối dễ với các vấn đề phức tạp.

+ Đối với bài toán 8-puzzle này có thể giải được khá nhiều trường hợp.

+ Lặp đi lặp lại với một lịch trình là rất chậm, đặc biệt là nếu hàm chi phí đắt tiền để tính toán, nhưng với bài toán này thì phù hợp giải quyết nhanh.

|  |  |  |  |
| --- | --- | --- | --- |
| Thuật toán  Sắp xếp | Hill climbing | Random Restart | Simulated Annaling |
| Thời gian (ms) | 1 | 3 | 2 |
| Số trường hợp giải được | 3 | 2 | 1 |
| Mức độ tối ưu với bài toán | 2 | 3 | 1 |
| Độ lớn không gian tìm kiếm | 3 | 1 | 2 |

*Hình 7. So sánh 3 thuật toán*

🡺 Đối với bài toán 8-puzzle, nếu trạng thái ban đầu có ít cực tiểu cục bộ hoặc chi phí trơn tru thì phương pháp Hill Climbing khá phù hợp để giải quyết với tốc độ nhanh. Ngược lại thì 2 phương pháp còn sẽ phù hợp hơn nhưng cá nhân nhóm em thì thấy phương pháp Simulated Annaling có phần tối ưu hơn so với Random Restart 🡺 Từ những đánh giá so sánh 3 thuật toán trong việc thực thi giải quyết bài toán 8-puzzle thì nhóm em cho rằng phương pháp Simulated Annaling là phương pháp tối ưu nhất để áp dụng.

**4.2. Hướng phát triển:**

Để phát triển chương trình, nhóm sẽ cố gắng khắc phục nhược điểm lớn nhất của chương trình là giới hạn về độ sâu để chương trình có thể thực thi được toàn bộ các trạng thái. Chương trình được thực hiện trên Console nên giao diện có phần hạn chế, do đó trong tương lai nhóm chúng em sẽ cố gắng cải thiện phần giao diện để thu hơn với người chơi và không gây cảm giác nhàm chán bất tiện. Ngoài ra để tăng thêm độ khó, chương trình sẽ có thêm phần tạo ra AI để chơi với người và độ khó là các thuật toán khác nhau và độ sâu tìm kiếm khác nhau. Từ đó tạo ra một chương trình AI cho máy tính hoàn thiện nhất.

**TÀI LIỆU THAM KHẢO**

[1] Labs.Septenti-technology.jp/machine-learning

[2] github.com/search?q=8puzzleSolver

[3] Wikipedia; *Heuristic*

[4] [Collamar@gmail.com](mailto:redblobgames@gmail.com); 8-puzzle *Use of the Heuristic; 2016*

[5] <https://stackoverflow.com/questions/23032305/how-can-i-formulate-8-puzzle-for-sa-algorithm>