**GAME 8-PUZZLE SỬ DỤNG CÁC THUẬT TOÁN LOCAL SEARCH**

**Lê Hoàng Nam 18110160**

**Trần Hữu Hào 18110103**

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

Tóm tắt:

Bài toán 8-puzzle (hay còn gọi là 8 số) là một bài toán quen thuộc với những người bắt đầu tiếp cận với môn Trí tuệ nhân tạo. Bài toán có nhiều phiên bản khác nhau dựa theo số ô, như 8-puzzle, 15-puzzle,… ở mức độ đơn giản nhất, chúng em xem xét dạng bài toán 8-puzzle. Bài toàn gồm một bảng ô vuông kích thước 3x3, có tám ô được đánh số từ 1 tới 8 và một ô trống. Trạng thái ban đầu, các ô được sắp xếp một cách ngẫu nhiên, nhiệm vụ của người chơi là tìm cách đưa chúng về đúng thứ tự như hình dưới:

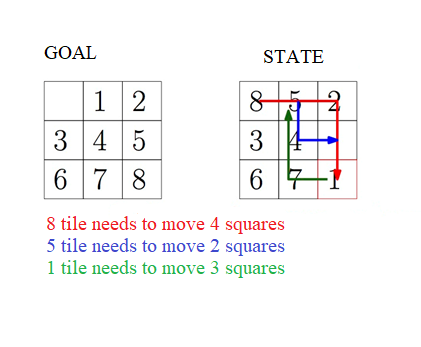
**Hình 1.** Trạng thái ban đầu và kết thúc

Trong phạm vi bài báo cáo, chúng tôi tìm hiểu về 4 phương pháp tìm kiếm lời giải cho bài toán 8-puzzle, đó là: hill climbing (steepest-ascent and first-choice variant), hill climbing with random restart, simulated annealing. Từ đó giúp ta có thể so sánh đánh giá được mức độ hiệu quả của các thuật toán cũng như thời gian hoàn thành các thuật toán.

Từ khóa: Trí tuệ nhân tạo, hill climbing, hill climbing with random restart, simulated annealing.

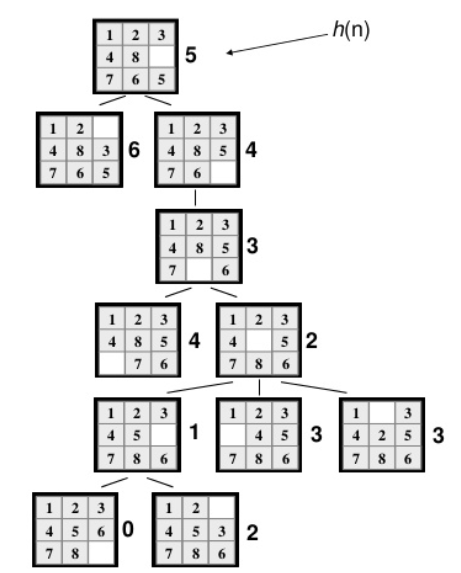
1. **ĐẶT VẤN ĐỀ**

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 là tính tổng số ô sai lệch của các ô số so với vị trí đúng của nó và nó có tên gọi là Manhattan. Theo đó, trạng thái nào có h min thì sẽ được chọn. Cách tính h như trong hình.



**Hình 2**. Minh họa cách tính h

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



**Hình 3.** Minh họa cây tìm kiếm game 8-puzzle

1. **PHƯƠNG PHÁP**
   1. Thuật toán Hill climbing

Giải thuật leo đồi chính là nền tảng cơ sở của các kỹ thuật tìm kiếm cục bộ. Mặc dù đây là giải thuật đơn giản nhưng nó lại rất mạnh và hiệu quả trong việc giải quyết các bài toán CSP lớn. Thuật ngữ “leo đồi” (hill climbing) xuất phát từ cơ chế “tu chỉnh lập”, ở mỗi bước của việc tìm kiếm chúng ta sẽ chọn một bước chuyển mà nó cải thiện giá trị hàm mục tiêu để thực hiện. Trong thuật toán leo đồi chỉ những bước cải thiện được hàm chi phí hoặc không làm cho hàm chi phí thay đổi mới được chọn vì vậy việc tìm kiếm sẽ liên tục bước lên vị trí cao hơn cho đến khi gặp điều kiện dừng[1]. Tuy nhiên phương pháp này không được cải thiện so với các phương pháp khác trong một số trường hợp sau:

* *Local maximum*: nút đang xét tốt hơn các nút lân cận, nhưng đó không phải là phương án tốt nhất trong toàn thể, ví vậy có thể phải quay lui về nút trước để đi theo hướng khác. Giải pháp này đòi hỏi ghi nhớ lại nhiều đường đi.
* *Flat* : Các giá trị của các phương án như nhau, không xác định được ngay hướng nào là tốt hơn trong vùng lân cận.

Để giải quyết được thuật toán hill climbing ta cần xây dựng một hàm tính cost theo công thức, bằng cách tính chi phí của các ô có giá trị khác với trạng thái đích.

Khoảng cách manhattan bằng tổng khoảng cách của từng ô từ trạng thái hiện tại P1(x1,y1) đến ô trạng thái đích P2(x2,y2) là: |x1-x2| + |y1-y2|

Sau khi tính h thì ta xây dụng hàm hill climbing như sau:

**Function** HILL-CLIMBING(*puzzle*) return a state that is a local maximum [2]

*current* <- *puzzle*

**loop do**

**if** *current* is goal **then** **return** *current*

*neighbor* <- lowest cost successor of *current*

**if** *neighbor*.cost > *current*.cost **then** **return** *current*

*current* <- *neighbor*

* 1. Thuật toán hill climbing with random restart

Thuật toán hill climbing with random restart là một biến thể của thuật toán hill climbing. Giống như thuật toán hill climbing nhưng khi sau khi mỗi vòng lặp mà vẫn chưa tìm ra kết quả thì ta sẽ tạo một trạng thái ngẫu nhiên và tiếp tục áp dụng thuật toán hill climbing cho trạng thái đó cho đến số lần lặp nhất định.

**Function** Random-Restart(*puzzle*) return a state that is a local maximum

*current* <- *puzzle*

**for** *i* = 1 **to** x **do** ;**x la so lan lap tu dinh nghia**

**loop do**

**if** *current* is goal **then** **return** *current*

*neighbor* <- lowest cost successor of *current*

**if** (*neighbor*.cost >= *current*.cost) **then** **break**

*current* <- *neighbor*

*current* <- random state

* 1. Thuật toán First-choice hill climbing

Thuật toán First-choice hill climbing thực hiện leo đồi ngẫu nhiên cho đến khi tạo ra được một trạng thái tốt hơn trạng thái hiện tại. Đây là một chiến lược tốt khi một trạng thái có nhiều successors.

**Function** First-Choice(*puzzle*) return a state that is a local maximum

*current* <- *puzzle*

**for** *i* = 1 **to** x **do** ;**x la so lan lap tu dinh nghia**

**if** *current* is goal **then** **return** *current*

**for** *t* = 1 **to** 20 **do**

*random* <- random successor of *current*

**if** *random*.cost < *current*.cost **then** **break**

**if** *t* == 20 **then return** *current*

*current* <- *random*

**return** *current*

* 1. Thuật toán Simulated annealing (SA)

Thuật toán simulated annealing để giải quyết các vấn đề tối ưu hóa không bị ràng buộc và ràng buộc. Phương pháp này mô hình hóa quá trình vật lý làm nóng vật liệu và sau đó từ từ hạ nhiệt độ để giảm khiếm khuyết, do đó giảm thiểu chi phí hệ thống. Ở mỗi lần lặp của thuật toán, một điểm mới được tạo ngẫu nhiên. Khoảng cách của điểm mới từ điểm hiện tại hoặc phạm vi tìm kiếm dựa trên phân phối xác suất với tỷ lệ thuận với nhiết độ. Thuật toán chấp nhận tất cả các điểm mới làm giảm chi phí, nhưng đồng thời, với một xác suất nhất định, các điểm nâng cao chi phí. Bằng cách chấp nhận tất cả các điểm nâng cao chi phí thực toán tránh bị mắc kẹt trong vùng local minimum và có nhiều giảm pháp khả thi hơn.[4]

**Function** simulated-annealing(*puzzle*) return a solution state

*current* <= *puzzle*

*TMax* <= 2.0

*t* <= -1.0

**loop do**

**if** *current* is goal **then return** *current*

*t* += 10-6

*T* <- *schedule*(*TMax,t*)

**if** T <= 0.0001 **then return** *current*

*next* <- random successor of *current*

*deltaE* <- *current*.cost - *next*.cost

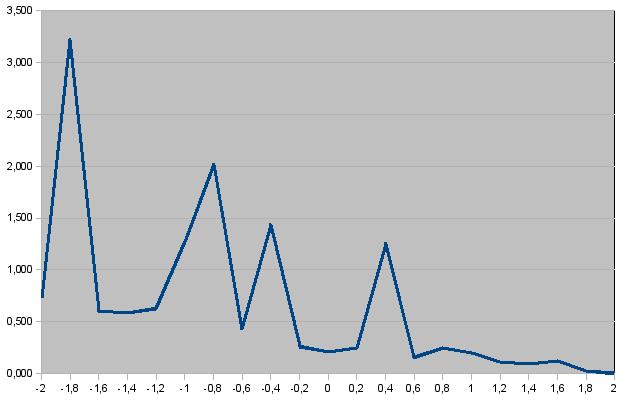
**if** *deltaE* > 0 **then** *current* <- *next*

**else** *current* <- *next* only with probability

Thuật toán Cooling dùng chiến lược random theo áp dụng công thức y = random(0..1) \* (TMax − x).

**Function** schedule(*TMax, t*) return temperature

**return** random(0..1) \* (*TMax − t*)



**Hình 4.** Lược đồ hiển thị nhiệt độ theo thời gian của thuật toán

1. **KẾT QUẢ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Success rate** | **Average steps** | **Total time** |
| Steepest-ascent hill climbing | 2/100 | 10 | 0.073 |
| First-choice hill climbing | 1/100 | 8 | 0.451 |
| Random-restart hill climbing | 28/100 | 9 | 6.139 |
| Simulated annealing | 91/100 | 1594 | 50.66 |

**Bảng 1.** Kết quả của các thuật toán

Bảng 1 thể hiện kết quả của 4 thuật toán về các phương diện: số trạng thái thành công, số bước trung bình và tổng thời gian hoàn thành của mỗi thực toán. Từ kết quả cho thấy thuật toán steepest-ascent hill climbing và first-choice hill climbing kém hiệu quả, thời gian nhất nhưng làm không chính xác. Thuật toán Random restart hill climbing chậm hơn so với hill climbing nhưng lại có kết quả tốt hơn. Cuối cùng là thuật toán Simulated annealing là thuật toán mạnh nhất cho kết quả và thời gian tối ưu nhất

1. **KẾT LUẬN**

Thông qua việc áp dụng các thuật toán local search vào giải quyết bài toán 8-puzzle giúp ta có cái nhìn toàn diện hơn về các thuật toán, giải đáp các khuất mắc về độ hiệu quả cũng như tốc độ tìm ra lời giải của 4 thuật toán: hill climbing, random restart hill climbing, first-choice hill climbing, simulated annealing. Từ đó ta nhận thấy rằng thuật toán simulated annealing là nổi trội hơn cả, với khả năng tìm ra lời giải nhanh và hiệu qua. Bài toán puzzle là 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 đã được nhiều người nghiên cứu giải quyết, nhưng đến nay vẫn chưa có cách giải quyết tối ưu cho tất cả không gian trạng thái trò chơi vì kích thước tăng không gian trạng thái sẽ tăng lên rất nhanh. Hy vọng những nghiên cứu đánh giá của chúng tôi sẽ góp phần bổ sung thêm một hướng giải quyết cho bài toán. Do thời gian có hạn nên đề tài không tránh khỏi những sai sót mong thầy góp ý đánh giá giúp chúng tôi hoàn thiện hơn.

**TÀI LIỆU THAM KHẢO**

[1] Informed search II, “[https://www.seas.upenn.edu/~cis391/Lectures/informed- search-II.pdf](https://www.seas.upenn.edu/~cis391/Lectures/informed-search-II.pdf)”

[2] Huỳnh Xuân Phụng,AI-Local search, <https://www.youtube.com/watch?time_continue=1536&v=5yiv4GcOe6w>

[3] MathWorks, “<https://www.mathworks.com/help/gads/what-is-simulated-annealing.html>”

[4] An approach to heuristics: 35-Puzzle with simulated annealing, “<https://kcir.pwr.edu.pl/~witold/aiarr/2009_projekty/35Puzzle/#Cooling_Functions>”