**GIẢI BÀI TOÁN 8-PUZZLE KHI HIỂN THỊ MỘT GIÁ TRỊ**

**Hồ Quốc Đạt1,Võ Thành Văn1, Huỳnh Trần Thảo Nhi1**

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

**TÓM TẮT**

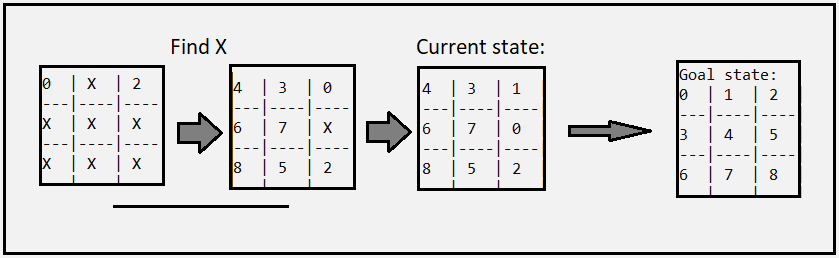
Hiện nay, việc sử dụng Trí tuệ nhân tạo để tìm kiếm lời giải cho những bài toán tìm đường đi là vấn đề thu hút được khá nhiều sự quan tâm của các cá nhân, tổ chức trên thế giới. Sau quá trình học tập và tìm hiểu, nhóm thực hiện đã quyết định áp dụng trí tuệ nhân tạo để giải bài toán 8-puzzle trong trường hợp bảng bị che lại, chỉ có một ô để trống để có thể nhìn thấy giá trị của các ô khi đi qua. Đề tài sử dụng phương pháp tìm đường đi tối ưu, cụ thể là thuật toán Breadth First Search, và được viết bằng ngôn ngữ C++. Sau quá trình hiện thực hoá đề tài, nhóm thực hiện đã hoàn thành chương trình giải bài toán 8-puzzle trong trường hợp bảng bị che lại, chỉ có một ô để trống để có thể nhìn thấy giá trị của các ô khi đi qua, nhưng trong một số trường hợp vẫn không tìm thấy kết quả. Thông qua việc nghiên cứu đề tài, nhóm đã hiểu rõ hơn về cách thức hoạt động và lợi ích của thuật toán Breadth First Search trong bài toán tìm kiếm, qua đó nhận thấy rõ tầm quan trọng của Trí tuệ nhân tạo trong việc giải quyết các vấn đề phức tạp trong cuộc sống hiện nay.

**Từ khoá:** trí tuệ nhân tạo; đường đi tối ưu; 8-puzzle; thuật toán Breadth First Search.

1. **GIỚI THIỆU ĐỀ TÀI**

Game 8-puzzle có lẽ là một game không còn xa lạ gì với chúng ta nữa. Với sự phổ biến của game thì mức độ chơi của game cũng ngày càng đa dạng. Từ một bảng số 3x3 gồm 8 ô số và 1 ô trống, để tăng độ khó cho game thì nhóm đã che 8 ô số lại và để lại 1 ô trống. Ở trạng thái bắt đầu thì các ô được random ở một vị trí bất kỳ và được random từ trạng thái đích để đảm bảo bài toán luôn có lời giải. Nhiệm vụ của bài toán là phải chọn một ô bất kỳ để nhìn thấy giá trị, nhờ vào ô trống để di chuyển các ô bị ẩn qua vị trí thấy để có thể xem giá trị. Sau khi các giá trị được mở thì di chuyển các ô về đúng trạng thái đích để giải được game.

Để tìm trạng thái đích, nhóm đã sử dụng thuật toán Breadth First Search (BFS). Thuật toán BFS là thuật toán xét (duyệt) hoặc tìm kiếm trên cây và đồ thị, có chiến lược tìm kiếm mù (tìm kiếm không có định hướng, không chú ý đến thông tin, giá trị được duyệt). Từ một đỉnh (nút) gốc ban đầu là đỉnh đang xét, xác định và lần lượt duyệt các đỉnh kề xung quanh đỉnh gốc vừa xét. Tiếp tục quá trình duyệt qua các đỉnh kề đỉnh vừa xét cho đến khi đạt được kết quả cần tìm hoặc duyệt qua tất cả các đỉnh.[[1]](#endnote-1)[1]

1. **PHƯƠNG PHÁP THỰC HIỆN**
2. **Khởi tạo trạng thái ban đầu**

Nhóm thực hiện sử dụng hàm random8Puzzle để tạo một trạng thái ban đầu với vị trí các giá trị được đảo lộn từ trạng thái đích. Mã giả của hàm random8Puzzle:

**function** random8Puzzle(root) **return** a node

**Loop do**

Ktra = true

**loop**

Mangrandom[j] = -1

Vitri A = Timkiemrong(child)

**Loop do**

**If** isLegalAction(child,action) **then**

Mangrandom[action] = action

**While** ktra

**If** Mangrandom[ran] == left **and** test\_action != right **then**

Swap(child[index\_i][index\_j-1],child([index\_i][index\_j])

Ktra = false

Test\_action = left

**Else if** Mangrandom[ran] = right **and** test\_action != left **then**

Swap(child[index\_i][index\_j+1],child([index\_i][index\_j])

Ktra = false

Test\_action = right

**Else if** Mangrandom[ran] = up **and** test\_action != down **then**

Swap(child[index\_i-1][index\_j],child([index\_i][index\_j])

Ktra = false

Test\_action = up

**Else if** Mangrandom[ran] = down **and** test\_action != up **then**

Swap(child[index\_i+1][index\_j],child([index\_i][index\_j])

Ktra = false

Test\_action = down

1. **Hiển thị các giá trị bị che**

Nhóm thực hiện đã sử dụng hàm FindX để hiển thị lần lượt các giá trị bị che với mã giả như sau:

Function **FindX** (node, x, y) return node

**If** problem. !CheckExistedInList(list, node.state[x][y]) **then return** list 🡨 node.state[x][y]

Array[24] = {1,3,0,2,1,3,0,2,1,1,3,3,0,0,2,2,1,1,3,3,0,0,2,2}

**Loop do** !Exit

**Loop** (each i in 0… 23)

**If** problem. Array[i] = Action::Left and previous step don’t Action::right **then return** swap(node.state[x][y-1], node.state[x][y]). Solution node

**Check** Exit 🡪 break

**Else If** problem. Array[i] = Action::Right and previous step don’t Action::left **then return** swap(node.state[x][y+1], node.state[x][y]). Solution node

**Check** Exit 🡪 break

**Else If** problem. Array[i] = Action::Up and previous step don’t Action::Down **then return** swap(node.state[x-1][y], node.state[x][y]). Solution node

**Check** Exit 🡪 break

**Else If** problem. Array[i] = Action::Down and previous step don’t Action::Up **then return** swap(node.state[x+1][y], node.state[x][y]). Solution node

**Check** Exit 🡪 break

1. **Thuật toán Breadth First Search**

Nhóm thực hiện đã tiến hành giải quyết vấn đề tìm kiếm trạng thái đích bằng thuật toán BFS với mã giả như sau:[[2]](#endnote-2)[2]

**fuction** BreadthFirstSearch(Start, Goal) **returns** a solution

node ‹– Start.State

**if** Goal\_test(node, goal) **then** **return** Solution(node)

frontier ‹– FIFO\_add(node)

explorer ‹– an empty set

**loop** **do**

node ‹– pop(frontier)

explorer ‹– FIFO\_add(node)

**for each** action **in** Actions(node) **do**

child ‹– Child\_node(node, action)

**if** child.state is not in explorer or frontier **then**

**if** Goal\_test(child, goal) **then** **return** Solution(child)

frontier ‹– FIFO\_add(frontier, child)

1. **KẾT LUẬN**
2. **Kết quả**

* Nhìn chung, chương trình đã có thể giải quyết được hầu hết yêu cầu của đề tài.
* Chương trình đã khởi tạo thành công trạng thái ban đầu của game 8puzzle với thứ tự các ô bị đảo lộn ngẫu nhiên.
* Chương trình hiển thị được toàn bộ các giá trị bị che.
* Chương trình đã tìm được đường đi từ trạng thái ban đầu đến trạng thái đích với xác suất khoảng 70%.

1. **Hạn chế**

* Chương trình không để giải hết các trường hợp.
* Code chưa hoàn toàn được tối ưu, rút gọn.
* Không áp dụng thêm được các giải thuật khác.

1. **Phát triển trong tương lai**

* Tối ưu code để chương trình có thể giải quyết bài toán nhanh chóng hơn.
* Áp dụng các giải thuật tìm kiếm khác như Depth First Search và A\* để chương trình giải quyết bài toán theo nhiều cách và tối ưu hơn.

**TÀI LIỆU THAM KHẢO**

1. [1] Trần Minh Thắng, *Thuật toán Breadth First Search,* 2015*,* <https://www.stdio.vn/articles/thuat-toan-breadth-first-search-285#8-khuyet-diem> [↑](#endnote-ref-1)
2. [2] Dùng thuật toán breadth first search (bfs) giải bài toán 8 puzzle, 30-01-2015, <http://diendan.congdongcviet.com/threads/t230037::dung-thuat-toan-breadth-first-search-bfs-giai-bai-toan-8-puzzle.cpp> [↑](#endnote-ref-2)