**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỀ TÀI:**

**TÌM HIỂU VÀ CÀI ĐẶT THUẬT TOÁN TÌM KIẾM HILL CLIMBING**

Giáo viên hướng dẫn: **MAI CƯỜNG THỌ**

Sinh viên thực hiện: **BÙI VĂN VIỆT**

**NGUYỄN MINH THẢO**

**TRẦN TIẾN**

Lớp**: 60.CNTT-3**

**Năm học: 2020 - 2021**

[**LỜI MỞ ĐẦU** 3](#_Toc62074600)

[**I.** **LÝ THUYẾT VỀ HEURISTIC:** 3](#_Toc62074601)

[**1.** **Tổng quan về Heuristic search:** 3](#_Toc62074602)

[**2.** **Chức năng của Heuristic:** 3](#_Toc62074603)

[**3. Phương pháp xây dựng thuật giải Heuristic:** 3](#_Toc62074604)

[**4. Ưu điểm của thuật giải Heuristic:** 4](#_Toc62074605)

[**5. Ứng dụng kinh nghiệm cho bài toán cụ thể:** 4](#_Toc62074606)

[**II.** **TÌM HIỂU VỀ GIẢI THUẬT LEO ĐỒI HILL CLIMBING** 4](#_Toc62074607)

[**GIỚI THIỆU** 5](#_Toc62074608)

[**NGUỒN GỐC** 5](#_Toc62074609)

[**MÔ TẢ THUẬT TOÁN – MÃ GIẢ** 5](#_Toc62074610)

[**CÁC VẤN ĐỀ CỦA THUẬT TOÁN** 6](#_Toc62074611)

[III. **ÁP DỤNG THUẬT TOÁN HILL-CLIMBING ĐỂ GIẢI BÀI TOÁN 8-QUEEN** 6](#_Toc62074612)

[**1.** **Giới thiệu về bài toán 8-Queen** 6](#_Toc62074613)

[a. Giới thiệu: 6](#_Toc62074614)

[b. Đề bài: 6](#_Toc62074615)

[**2.** **Thuật toán Hill-Climbing để giải bài toán 8-Queen:** 7](#_Toc62074616)

[a. Thuật toán leo đồi: 7](#_Toc62074617)

[b. Lời giải cho thuật toán (áp dụng Hill Climbing): 8](#_Toc62074618)

[c. Mã giả: 9](#_Toc62074619)

[**IV.** **KẾT LUẬN** 14](#_Toc62074620)

[1. Kết quả đạt được: 14](#_Toc62074621)

[2. Nhận xét: 14](#_Toc62074622)

[3. Kết luận 14](#_Toc62074623)

[TÀI LIỆU THAM KHẢO 15](#_Toc62074624)

ĐỀ BÀI: TÌM HIỂU VÀ CÀI ĐẶT THUẬT TOÁN HILL CLIMBING

1. Mục đích:

Vận dụng kiếm thức lập trình C, toán rời rạc, lập trình hướng đối tượng. Nâng cao khả năng tự giải quyết vấn đề, khả năng làm việc nhóm, và quản lý mã nguồn trong làm việc nhóm.

1. Nội dung:
2. Tìm hiểu tổng quan bài toán tìm kiếm dự trên kinh nghiệm (Heuristic Search),và cụ thể cho thuật toán tìm kiếm Hill Climbing và phân tích cài đặt.
3. Tổ chức dữ liệu và cài đặt thuật toán ứng dụng cho một bài toán cụ thể.

# **LỜI MỞ ĐẦU**

HEURISTIC SEARCH là các kỹ thuật dựa trên kinh nghiệm để giải quyết vấn đề, học hỏi hay khám phá nhằm đưa ra một giải pháp mà không được đảm bảo là tối ưu. Với việc nghiên cứu khảo sát không có tính thực tế, các phương pháp heuristic được dùng nhằm tăng nhanh quá trình tìm kiếm với các giải pháp hợp lý thông qua các suy nghĩ rút gọn để giảm bớt việc nhận thức vấn đề khi đưa ra quyết định.

1. **LÝ THUYẾT VỀ HEURISTIC:**
2. **Tổng quan về Heuristic search:**

Heuristic: là các kỹ thuật dựa trên kinh nghiệm để giải quyết vấn đề, nhằm đưa ra một giải pháp mà không được đảm bảo là tối ưu (theo Wiki)

Heuristic function: Hàm đánh giá dựa trên kinh nghiệm, dựa vào đó để xếp hạng thứ tự tìm kiếm, cách chọn hàm đánh giá quyết định nhiều đến kết quả tìm kiếm.

1. **Chức năng của Heuristic:**

Các chương trình giải quyết những vấn đề trí tuệ nhân tạo sử dụng Heuristic cơ bản theo 2 dạng:

* Vấn đề có thể không có giải pháp chính xác vì những điều không rõ ràng trong diễn đạt vấn đề hoặc trong các dữ liệu có sẵn.
* Vấn đề có thể có giải pháp chính xác, nhưng chi phí tính toán để tìm ra nó không cho phép.

## **3. Phương pháp xây dựng thuật giải Heuristic:**

Thuật giải Heuristic gồm 2 phần: Hàm đánh giá Heuristic và thuật toán để sử dụng nó trong tìm kiếm không gian trạng thái.

Có nhiều các để xây dựng một thuật giải Heuristic, trong đó người ta thường dựa và một số nguyên lý cơ bản như sau:

- Nguyên lý vét cạn thông minh: Trong một bài toán tìm kiếm nào đó, khi không gian tìm kiếm lớn, ta thường tìm cách giới hạn lại không gian tìm kiếm hoặc thực hiện một kiểu dò tìm đặc biệt dựa vào đặc thù của bài toán để nhanh chóng tìm ra mục tiêu

- Nguyên lý tham lam (Greedy): lấy tiêu chuẩn tối ưu (Trên phạm vi toàn cục) của bài toán để làm tiêu chuẩn chọn lựa hành động cho phạm vi cục bộ của từng bước (Hay từng giai đoạn) trong quá trình tìm kiếm lời giải

- Nguyên lý thứ tự: thực hiện hành động dựa trên một cấu trúc thứ tự hợp lý của không gian khảo sát nhằm nhanh chóng đạt được một lời giải tốt.

## **4. Ưu điểm của thuật giải Heuristic:**

Thuật giải Heuristic thể hiện cách giải bài toán với các đặc tính sau:

- Thường tìm được lời giải tốt (Nhưng không chắc là lời giải tốt nhất).

- Giải bài toán theo thuật giải Heuristic thường dễ dàng và nhanh chóng đưa ra kết quả hơn so với giải thuật tối ưu, vì vậy chi phí thấp hơn.

- Thuật giải Heuristic thường thể hiện khá tự nhiên, gần gũi với cách suy nghĩ và hành động con người

## **5. Ứng dụng kinh nghiệm cho bài toán cụ thể:**

Bài toán tìm đường đi ngắn nhất.

1. **TÌM HIỂU VỀ GIẢI THUẬT LEO ĐỒI HILL CLIMBING**

Diagram

Description automatically generated

Trạng thái

Giá trị

**GIỚI THIỆU**

Trong khoa học máy tính, giải thuật Hill Climbing là một kĩ thuật tối ưu toán học thuộc họ tìm kiếm cục bộ. Nó thực hiện tìm một trạng thái tốt hơn trạng thái hiện tại để mở rộng.

Hill Climbing dễ dàng tìm thấy một giải pháp tốt cục bộ (local optimum) nhưng khó tìm thấy giải pháp tốt nhất (global optinum) trong tất cả các giải pháp được đưa ra (search space).

Tính đơn giản của giải thuật khiến nó trở thành lựa chọn đầu tiền trong số các giải thuật tối ưu. Nó được sử dụng rất nhiều trong trí tuệ nhân tạo, dùng cho mục đích đi đến trạng thái đích từ một node bắt đầu. Việc chọn node tiếp theo và node bắt đầu có thể thay đổi nhiều giải thuật khác nhau.

**NGUỒN GỐC**

Hill Climbing là một kỹ thuật **tối ưu toán học** (mathematical optimization).

Trong toán học, khoa học máy tính (computer science) và vận động học (operation research), **tối ưu toán học** (mathematical optimization, cũng gọi là optimization hay mathematical programing) **là sự lựa chọn một thành phần tốt nhất (liên quan đến một vài tiêu chuẩn) từ tập hợp các lựa chọn có sẵn.**

Các giải thuật thuộc loại tìm kiếm tối ưu như là: Hill Climbing, Stimulate Annealing, Generic Algorithm.

Hill climbing thuộc họ **tìm kiếm cục bộ** (local search).

Trong khoa học máy tính, **tìm kiếm cục bộ** là một phương pháp heuristic để giải quyết các bài toán tối ưu khó. Tìm kiếm cục bộ có thể sử dụng trong các bài toán mà có thể được tính bằng cách tìm một giải pháp tối đa hóa một tiêu chí nào đó trong số các giải pháp được đưa ra. Giải thuật tìm kiếm cục bộ chuyển từ giải pháp này đến giải pháp khác trong không gian các giải pháp được đưa ra (không gian tìm kiếm) bằng cách áp dụng những thay đổi cục bộ cho đến khi một giải pháp được coi là tối ưu được tìm thấy hoặc thời gian giới hạn trôi qua.

**MÔ TẢ THUẬT TOÁN – MÃ GIẢ**

B1: Xét trạng thái đầu: Nếu là đích => dừng

Ngược lại, thiết lập trạng thái bắt đầu = trạng thái hiện tại.

B2: Lựa một luật để áp dụng vào trạng thái hiện tại để sinh ra một trạng thái mới.

B3: Xem xét trạng thái mới này:

Nếu là đích => dừng.

Nếu không phải là đích nhưng tốt hơn trạng thái hiện tại thì thiết lập trạng thái hiệu t là trạng thái mới.

Nếu không tốt hơn thì đến trạng thái mới tiếp theo

Lặp đến khi: gặp đích hoặt không còn luật nào nữa chưa được áp dụng vào trạng thái hiện tại.

**CÁC VẤN ĐỀ CỦA THUẬT TOÁN**

* Tối ưu toàn cục (global optimum).

Vấn đề lớn nhất mà giải thuật Hill Climbing gặp phải là nó dễ rơi vào vùng tối ưu cục bộ, đó là lúc chúng ta leo lên một đỉnh mà chúng ta không thể tìm láng giềng nào tốt hơn được nữa nhưng đỉnh này lại không phải là đỉnh cao nhất.

Để giải quyết vấn đề này, khi leo đến một đỉnh tổi ưu cục bộ, để tìrn được lời giải tốt hơn nữa ta có thể lặp lại thuật toán leo đổi vớí nhiều điểm xuất phát khác nhau được chọn ngẫu nhiên và lưu lại kết quả tốt nhất ở mỗi lần lặp. Nếu số lần lặp đủ lớn thì ta có thể tìm được đỉnh tối ưu toàn cục, tuy nhiên Tới những bài toán có không  gian tìm kiểm khổng lổ (chẳng bạn như bài toán xếp lịch) ta không thể đưa ra số lần lặp đủ lớn để đảm báo tìm được lời giải tổi ưu. Cho nên đây là phương pháp giải quyết không mang lại nhiều hiệu quả trong trường hợp này.

Như vậy hiệu quả của bài toán phụ thuộc rất nhiều vào “bề mặt” của không gian tìm kiếm. Nếu bài toán chỉ có và đỉnh tối ưu cục bộ thì giải thuật sẽ tìm ra lời giải rất nhanh. Tuy nhiên, trong trường hợp không gian tìm kiếm quá lồi lõm, giải thuật sẽ bị luẩn quẩn trong vùng tối ưu cục bộ và có thể không tìm ra lời giải tối ưu của bài toán.

* Chóp nhọn (Ridges and Alleys).
* Cao nguyên (Plateau).

1. **ÁP DỤNG THUẬT TOÁN HILL-CLIMBING ĐỂ GIẢI BÀI TOÁN 8-QUEEN**
2. **Giới thiệu về bài toán 8-Queen**
3. Giới thiệu:

Bài toán tám quân hậu là bài toán đặt tám quân hậu trên bàn cờ vua kích thước 8×8 sao cho không có quân hậu nào có thể “ăn” được quân hậu khác, hay nói khác đi không quân hậu nào có để di chuyển theo quy tắc cờ vua. Màu của các quân hậu không có ý nghĩa trong bài toán này. Như vậy, lời giải của bài toán là một cách xếp tám quân hậu trên bàn cờ sao cho không có hai quân nào đứng trên cùng hàng, hoặc cùng cột hoặc cùng đường chéo.

1. Đề bài:

Cho một bàn cờ kích thước 8\*8. Chúng ta cần đặt n (n>=4) quân hậu vào bàn cờ sao cho chúng không tấn công được nhau, tức là không có cặp con hậu nào nằm cùng hàng, cùng cột, cùng đường chéo

**A picture containing text, crossword puzzle

Description automatically generated**.

1. **Thuật toán Hill-Climbing để giải bài toán 8-Queen:**
2. Thuật toán leo đồi:

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 của việc tìm kiếm, chúng ta sẽ chọn một bước chuyển mà nó cải thiện giá trị hàm mục tiêu để thực hiện. Giải thuật leo đồi được mô tả như sau:

**Procedure** Hill\_Climbing\_Search;

**Begin**

Khởi tạo danh sách L chỉ chứa trạng thái đầu;

**Loop do**

**If** L rỗng **then** {thông báo thất bại; **stop**;}

Loại trạng thái u đầu danh sách L;

**If** u là trạng thái kết thúc của **then** {thông báo thành công; **stop**;}

**For** mỗi trạng thái v kề u đặt v vào L sao cho các phần tử được đưa vào đầu danh sách L có đánh giá giảm dần;

End;

Trong giải thuật leo đồi, chỉ những bước chuyển 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 nó gặp điều kiện dừng. Khuyết điểm của giải thuật leo đồi:

Vấn đề lớn nhất của giải thuật leo đồi mắc phải là nó sẽ bị rơi vào vùng tối ưu cục bộ, đó là lúc chúng ta leo lên một đỉnh mà chúng ta không thể tìmđược láng giềng nào tốt hơn được nữa nhưng đỉnh này lại không phải đỉnh cao nhất.

Khi leo đến một đỉnh tối ưu cục bộ, để tìm được lời giải tốt hơn nữa ta có ta có thể lặp lại thuật toán leo đồi với nhiều điểm xuất phát khác nhau được chọn ngẫu nhiên và lưu lại kết quả tốt nhất ở mỗi lần lặp. Nếu số lần lặp đủ lớn thì ta có thể tìm được đỉnh tối ưu toàn cục, tuy nhiên với những bài toán có không gian tìm kiếm khổng lồ (chẳng hạn như bài toán xếp lịch) ta không thể đưa ra số lần lặp đủ lớn để đảm bảo tìm được lời giải tối ưu. Cho nên đây là phương pháp giải quyết không mang lại nhiều hiệu quả, chúng ta sẽ khảo sát các giải thuật cải tiến khác khắc phục được vấn đề này ở phần sau.

Nhận xét: hiệu quả của giải thuật leo đồi phụ thuộc rất nhiều vào “bề mặt” của không gian tìm kiếm bài toán. Nếu bài toán chỉ có vài đỉnh tối ưu cục bộ thì giải thuật sẽ không tìm ra lời giải tối ưu rất nhanh, tuy nhiên trong trường hợp bề mặt không gian tìm kiếm lại quá lồi lõm (rất nhiều đỉnh tối ưu cục bộ), giải thuật sẽ bị luẩn quẩn trong vùng tối ưu cục bộ và có thể không tìm ra được lời giải tối ưu của bài toán.

1. Lời giải cho thuật toán (áp dụng Hill Climbing):

Ý tưởng thuật toán: Xếp tất cả N con Hậu lên bàn cờ (mỗi con hậu chiếm 1 ô cờ). Mỗi cách xếp như vậy được gọi là 1 State (trạng thái).Ta cần một trạng thái ban đầu, có thể là ngẫu nhiên. Trạng thái đích là trạng thái mà các quân hậu không thể tấn công nhau. Từ trạng thái ban đầu, ta thực hiện thay đổi vị trí quân cờ để tạo ra trạng thái tiếp theo. Hàm chi phí tính bằng số quân hậu có thể bị tấn công. Ta sẽ chọn trạng thái nào có giá trị hàm nhỏ nhất. Và để cải thiện hàm chi phí, chúng ta sẽ chọn quân hậu nào có khả năng bị tấn công nhiều nhất dịch chuyển tạo ra trạng thái mới.

B1: Khởi tạo bàn cờ ngẫu nhiên các vị trí của quân hậu. Ta gọi đây là trạng thái ban đầu.

B2: Chọn quân cờ có khả năng bị tấn công nhiều nhất. Thực hiện dịch chuyển các nước đi phù hợp để tạo ra các trạng thái con.

B3: Từ các trạng thái con này chọn ra trạng thái có hàm chi phí nhỏ nhất. Chọn nó là thái đầu tiên và lưu chúng vào danh sách trạng thái được chọn.

B4 Lặp lại B2 cho đến khi không còn quân cờ nào trong tình trạng bị tấn công.

1. Mã giả:

***//khởi tạo hàm cung cấp điểm bắt đầu cho thuật toán***

void configureRandomly(int board[][N],

int\* state)

{

***//lặp lại các chỉ số cột***

for (int i = 0; i < N; i++) {

***// lấy chỉ mục hàng ngẫu nhiên***

state[i] = rand() % N;

***//Đặt quân hậu vào vị trí vừa lấy được trong bàn cờ***

board[state[i]][i] = 1;

}

}

Diagram, engineering drawing, line chart

Description automatically generated***// hàm tính toán giá trị mục tiêu của trạng thái(quân hậu tấn công nhau)***

int calculateObjective(int board[][N], int\* state)

{

***//với mỗi quân hậu trong cột, ta kiểm tra các quân hậu khác có nằm trong hàng của quân hậu này***

***//nếu tìm thấy thì tăng biến số tấn công***

***// số lượng các quân hậu tấn công nhau***

***//ban đầu = 0***

int attacking = 0;

***// tạo biến để định nghĩa hàng và cột***

int row, col;

for (int i = 0; i < N; i++) {

***// tại mỗi cột 'i', quân hậu được đặt ở hàng state[i]***

***// bên trái cùng 1 hàng(hàng không đổi và cột giảm)***

row = state[i], col = i - 1;

while (col >= 0 && board[row][col] != 1) {

col--;

}

if (col >= 0 && board[row][col] == 1) {

attacking++;

}

***// bên phải cùng 1 hàng(hàng không đổi và cột tăng lên)***

row = state[i], col = i + 1;

while (col < N && board[row][col] != 1) {

col++;

}

if (col < N && board[row][col] == 1) {

attacking++;

}

***// Theo đường chéo sang trái lên(hàng và cột giảm dần)***

row = state[i] - 1, col = i - 1;

while (col >= 0 && row >= 0 && board[row][col] != 1) {

col--;

row--;

}

if (col >= 0 && row >= 0 && board[row][col] == 1) {

attacking++;

}

***// Theo đường chéo sang phải xuống(hàng và cột tăng đồng thời)***

row = state[i] + 1, col = i + 1;

while (col < N && row < N && board[row][col] != 1) {

col++;

row++;

}

if (col < N && row < N && board[row][col] == 1) {

attacking++;

}

***// Theo đường chéo từ trái xuống(cột giảm và hàng tăng)***

row = state[i] + 1, col = i - 1;

while (col >= 0 && row < N && board[row][col] != 1) {

col--;

row++;

}

if (col >= 0 && row < N && board[row][col] == 1) {

attacking++;

}

***// Theo đường chéo sang phải lên(cột tăng và hàng giảm)***

row = state[i] - 1, col = i + 1;

while (col < N && row >= 0 && board[row][col] != 1) {

col++;

row--;

}

if (col < N && row >= 0 && board[row][col] == 1) {

attacking++;

}

}

***// trả về các cặp***

return (int)(attacking / 2);

}

***//hàm nhận trạng thái hiện tại có giá trị nhỏ nhất trong số các trạng thái lân cận***

void getNeighbour(int board[][N], int\* state)

{

***//khai báo và khởi tạo bảng và trạng thái tối ưu(optimal)***

***//với bảng hiện tại và trạng thái là điểm bắt***

int opBoard[N][N];

int opState[N];

copyState(opState, state);

generateBoard(opBoard, opState);

***//khởi tạo giá trị mục tiêu tối ưu***

int opObjective = calculateObjective(opBoard, opState);

***//khai báo và khởi tạo bảng và trạng thái tạm thời cho mục đích tính toán***

int NeighbourBoard[N][N];

int NeighbourState[N];

copyState(NeighbourState, state);

generateBoard(NeighbourBoard, NeighbourState);

***//lặp lại thông qua tất cả các trạng thái lân cận khả thi của bảng***

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

***//điều kiện để bỏ qua trạng thái hiện tại***

if (j != state[i]) {

***//khởi tạo trạng thái lân cận tạm thời với trạng thái lân cận hiện tại***

NeighbourState[i] = j;

NeighbourBoard[NeighbourState[i]][i] = 1;

NeighbourBoard[state[i]][i] = 0;

***//tính giá trị khả thi của trạng thái lân cận***

int temp = calculateObjective(NeighbourBoard, NeighbourState);

***//so sánh các mục tiêu tạm thời và tối ưu lân cận***

***//nếu tạm thời nhỏ hơn tối ưu thì cập nhật cho phù hợp***

if (temp <= opObjective)

{

opObjective = temp;

copyState(opState, NeighbourState);

generateBoard(opBoard, opState);

}

***//cấu hình ban đầu cho lần lặp tiếp theo***

NeighbourBoard[NeighbourState[i]][i] = 0;

NeighbourState[i] = state[i];

NeighbourBoard[state[i]][i] = 1;

}

}

}

***//sao chép bảng và trạng thái tối ưu được tìm thấy đến bảng và trạng thái hiện tại***

copyState(state, opState);

fill(board, 0);

generateBoard(board, state);

}

***//hàm hill climbing***

void hillClimbing(int board[][N], int\* state)

{

***//khai báo và khởi tạo bảng và trạng thái lận cận với bảng và trạng thái hiện tại là điểm bắt đầu***

int neighbourBoard[N][N] = {};

int neighbourState[N];

copyState(neighbourState, state);

generateBoard(neighbourBoard, neighbourState);

do {

***//sao chép bảng và trạng thái lân cận sang bảng và trạng thái hiện tại, từ lân cận trở thành hiện tại sau bước này***

copyState(state, neighbourState);

generateBoard(board, state);

***//lấy trạng thái lân cận tối ưu***

getNeighbour(neighbourBoard, neighbourState);

if (compareStates(state, neighbourState)) {

***//nếu lân cận và hiện tại bằng nhau thì không tồn tại lân cận tối ưu và do đó xuất ra kết quả(ngắt vòng lặp)***

printBoard(board);

break;

}

else if (calculateObjective(board,state) == calculateObjective(neighbourBoard, neighbourState))

{

***//nếu lận cận và hiện tại không bằng nhau nhưng mục tiêu của chúng bằng nhau thì chúng ta đang tiến gần đến mức tối ưu cục bộ***

***//trong mọi trường hợp, nhảy tới một trạng thái lân cận bất kì để thoát khỏi nó***

***//trạng thái lân cận ngẫu nhiên***

neighbourState[rand() % N] = rand() % N;

generateBoard(neighbourBoard, neighbourState);

}

} while (true);

}

1. **KẾT LUẬN**
2. Kết quả đạt được:

![Text

Description automatically generated]()

1. Nhận xét:

* Ưu điểm:

Dễ cài đặt, có thể đưa ra tất cả lời giải của bài toán

* Khuyết điểm:

Thời gian chạy lâu, tốn nhiều tài nguyên hệ thống

1. Kết luận

Bài toán 8 quân hậu là bài toán phổ biến để đánh giá thuật toán. Trong trường hợp này, thuật toán leo đồi vẫn phát huy được tính đơn giản và hiệu quả.

Thuật toán leo đồi được áp dụng trong bài toán 8 quân hậu là phiên bản cơ bản. Chỉ có một bước cải tiến là cải thiện hàm chi phí bằng cách chọn quân cờ có khả năng bị tấn công nhiều nhất.

TÀI LIỆU THAM KHẢO

[1] <https://vi.wikipedia.org/wiki/B%C3%A0i_to%C3%A1n_t%C3%A1m_qu%C3%A2n_h%E1%BA%ADu>

[2] <https://voer.edu.vn/m/bai-toan-8-quan-hau/a5a1e2a6>

[3] <http://text.123doc.org/document/1245563-bao-cao-tri-tue-nhan-tao-ap-dung-thuat-toan-hill-climbing-de-giai-bai-toan-n-queen.htm>