**ITEC**

**CS-300 Artificial Intelligence**

**Lab 2 - Local Search**

**Hill Climbing Algorithm and 8-queen problem**

***Dr. Nguyen Ngoc Thao***

***Msc. Nguyen Hai Dang – Msc. Do Trong Le – Nguyen Quang Thuc***

## Local Search (Tìm kiếm cục bộ):

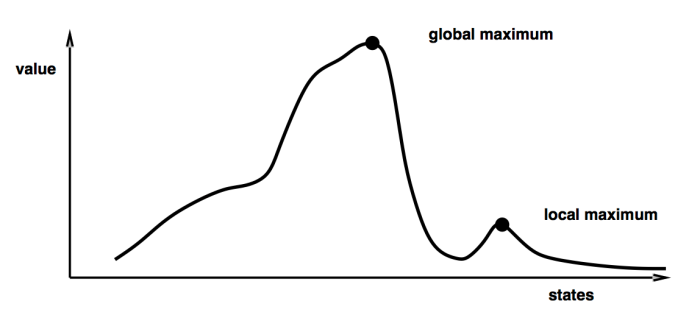
Những thuật toán tìm kiếm mà ta đã xem xét tới thời điểm hiện tại được thiết kế để duyệt trong không gian tìm kiếm một các có hệ thống. Tính hệ thống đạt được bằng cách giữ trong bộ nhớ một hoặc nhiều đường đi và bằng cách ghi nhận cái nào đã được xem xét trên đường đi và cái nào chưa. Khi đích được tìm thấy, đường đi tới đích này cũng chính là lời giải cho bài toán.

Trong một số bài toán, để tìm được lời giải tối ưu nhất (tối ưu toàn cục) cần tốn rất nhiều tài nguyên và chi phí tính toán, đặc biệt là các bài toán mà không gian tìm kiếm rất rộng, vô hạn hoặc gần như vô hạn. Do đó, việc tìm kiếm trên lời giải tối ưu nhất cho các bài toán này gần như là không thể, hoặc chỉ có thể tìm được do các giá trị khởi tạo gần với điểm tối ưu.

Đối với các bài toán như vậy, thuật toán **tìm kiếm cục bộ (Local Search)** là phương án thường được áp dụng. Thuật toán tìm kiếm cục bộ sử dụng một trạng thái hiện tại đơn lẻ (thay vì nhiều đường đi khác nhau) và chỉ di chuyển sang trạng thái lân cận. Thông thường các đường đi trong quá trình tìm kiếm sẽ không được giữ lại.

Thuật toán **tìm kiếm cục bộ (Local Search)** có hai ưu điểm nổi bật:

1. Sử dụng **rất ít bộ nhớ**, thông thường là một lượng cố định.
2. Thường tìm được **lời giải hợp lý** trong không gian trạng thái lớn hoặc vô hạn.



## Hill Climbing Algorithm (Thuật toán leo đồi):

**Thuật toán leo đồi** là một thuật toán trong nhóm các **thuật toán tìm kiếm cục bộ**. Thuật toán leo đồi được xây dựng dựa trên ý tưởng tham lam, liên tục di chuyển theo hướng tăng dần giá trị tại một trạng thái cho trước.

Thuật toán đơn thuần sử dụng một vòng lặp trong đó liên tục tìm kiếm các trạng thái lân cận có giá trị tối ưu hơn và di chuyển đến trạng thái đó. Thuật toán dừng khi tới được “trạng thái tối ưu” (local optimal state) mà các trạng thái gần lân cận không có giá trị nào cao hơn.

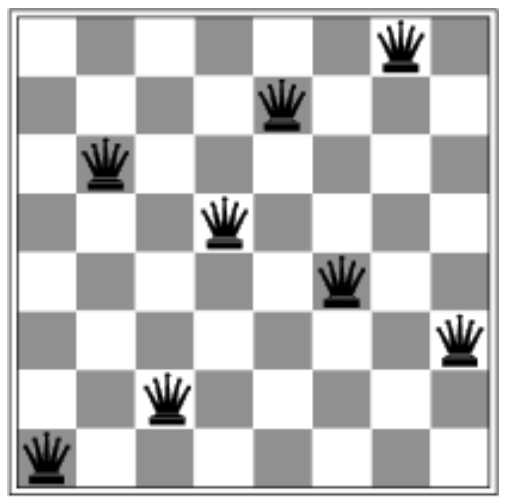
***Lưu ý****: “đỉnh tối ưu” đó có thể không phải là tối ưu nhất của bài toán.*

**Graphical user interface, text, application, email

Description automatically generated**

## Mô tả bài toán **N** hậu:

Cho trước một bàn cờ **N**x**N**, trên đó đặt sẵn **N** quân hậu ở **N** vị trí bất kỳ với ràng buộc mỗi cột có ít nhất một quân hậu. Ví dụ như hình bên dưới có 8 quân hậu.



Nhiệm vụ của bạn là tìm cách để thực hiện một số thao tác di chuyển quân hậu trên bàn cờ sao cho các quân hậu không tấn công lẫn nhau. Tại một thao tác di chuyển chỉ được phép di chuyển một quân hậu trên chính cột của quân hậu đó đang đứng sang hàng khác.

Hàm ***heuristic h(x)*** được định nghĩa là số lượng các quân hậu có thể tấn công lẫn nhau, trực tiếp hoặc gián tiếp. Ví dụ như trạng thái trên có h(x)=1.

**Dạng bài tập:** Cá nhân (Individual)

**Yêu cầu:**

Cho trước một trạng thái bàn cờ, xuất ra danh sách thứ tự các trạng thái để đi đến trạng thái tối ưu (có thể không phải là tối ưu nhất).

Mỗi trạng thái bàn cờ được mô tả bởi một danh sách N số, số thứ i cho biết số hàng quân hậu tại cột i đang đứng.

Các thuật toán cần cài đặt:

* **Hill-climbing**
* **First-choice hill climbing**
* **Random-restart hill climbing**

**Input và output mẫu:**

| input.txt | output.txt |
| --- | --- |
| 2 3 0 1 | 2 3 0 1  0 3 0 1  0 3 0 2  1 3 0 2 |

**Quy định nộp bài:**

Nộp bài trên moodle theo đúng cấu trúc tên sau:

* Bài nộp gồm đúng 01 file .zip chứa 01 file mã nguồn 8\_queens.py đã được hoàn thiện và có thể biên dịch (có thể chứa các file modules khác).
* File .zip được đặt tên theo cú pháp MSSV.zip

**Thời hạn: 2 tuần**

**Lưu ý:**

* Sai cú pháp nộp bài sẽ mất điểm của homework này.
* Chép code của nhau 0 điểm!
* Nếu các bạn tham khảo nguồn nào, bắt buộc phải ghi nguồn tham khảo vào trong file code dưới dạng chú thích (comment). Những bài không trích dẫn nguồn tham khảo sẽ bị 0 điểm.