Local Search
(ARTIFICIAL INTELLIGENCE)

(slides adapted and revised from

Dan Klein, Pieter Abbeel, Anca Dragan, et al)

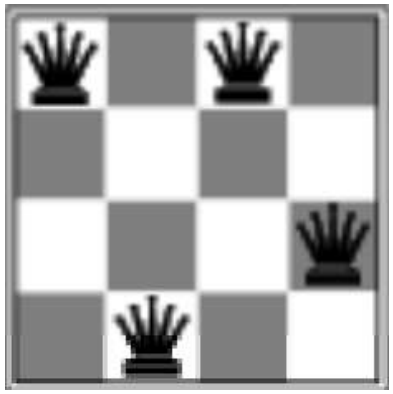
Dr. Rebecca Hutchinson, et al, Introduction to Artificial Intelligence

Local Search

Local Search

Example: 4-queens problem

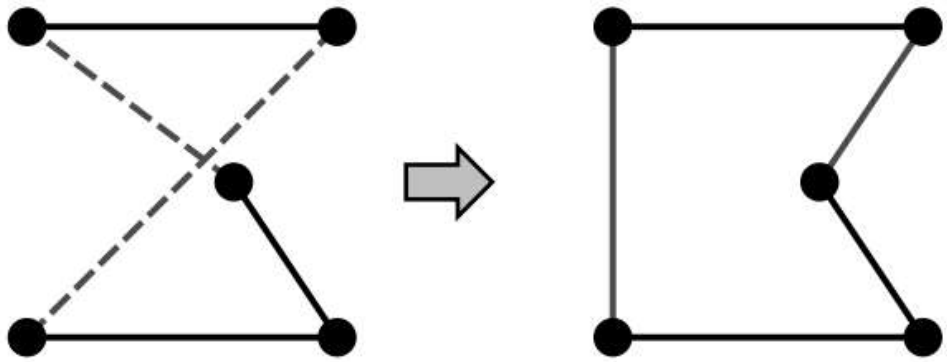
- h = number of pairs of queens that are attacking each other, either directly or indirectly



Local Search

Example: Travelling Salesperson Problem

- *Bắt đầu với bất kỳ chuyến tham quan hoàn chỉnh nào, thực hiện trao đổi từng cặp*

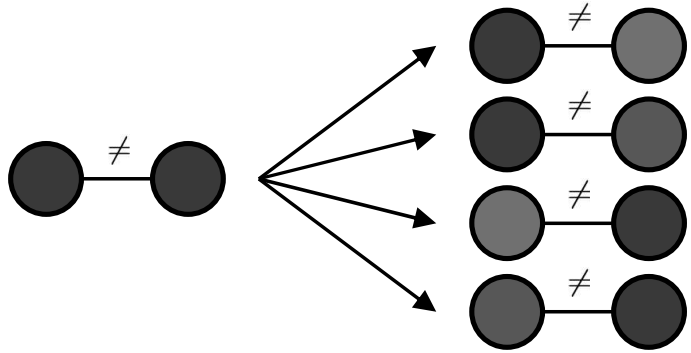


Local Search

- Những vấn đề này có điểm gì chung?
  - Những vấn đề này không giống với các vấn đề tìm kiếm từ lớp trước:
    - Đường dẫn đến mục tiêu không liên quan -- tất cả những gì bạn quan tâm là cấu hình cuối cùng
    - Đây thường là các vấn đề tối ưu hóa trong đó bạn tìm trạng thái tốt nhất theo một hàm mục tiêu
- Những vấn đề này là ví dụ về các vấn đề tìm kiếm cục bộ

Local Search

- Tìm kiếm cây giữ các lựa chọn thay thế chưa được khám phá ở biên (đảm bảo tính hoàn chỉnh)
- Tìm kiếm cục bộ: cải thiện một tùy chọn duy nhất cho đến khi bạn không thể làm cho nó tốt hơn (không có rìa!)
- Hàm kế thừa mới: thay đổi cục bộ



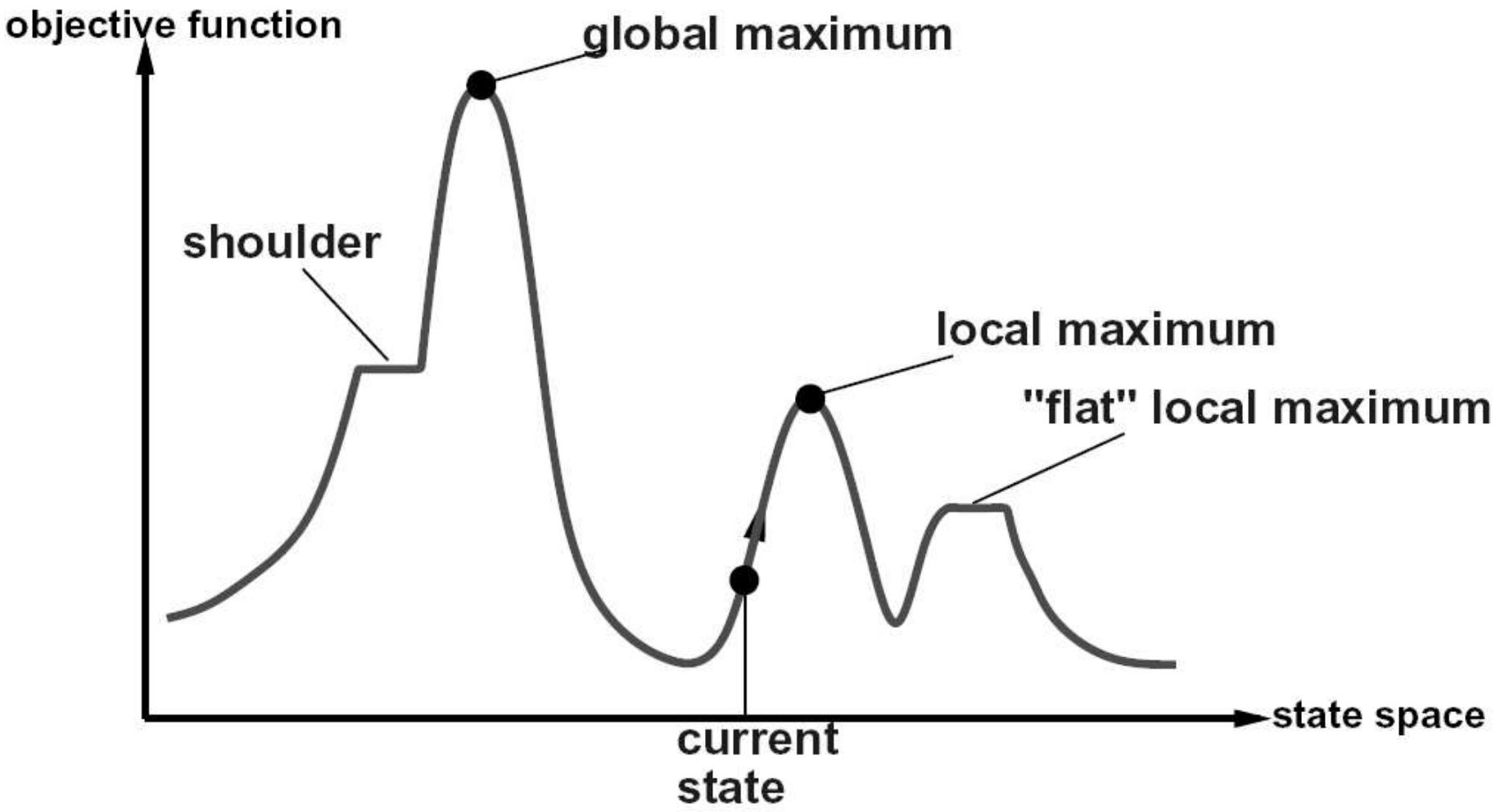
- Nói chung là nhanh hơn nhiều và hiệu quả hơn về bộ nhớ (nhưng không đầy đủ và không tối ưu)

Local Search Problems

- Given a set of states S = {X1 , …, Xm}
- And an objective function h(Xi) that returns the “goodness” of a state
- Find the state X* that maximizes the objective function

Lưu ý: Đôi khi h(Xi) là hàm chi phí thay vì hàm mục tiêu. Trong trường hợp này, chúng ta muốn tìm trạng thái X* tối thiểu hóa hàm chi phí. Chúng ta sẽ giải quyết các hàm mục tiêu trong các slide này nhưng dễ dàng chỉ cần đảo ngược dấu và nghĩ đến các hàm chi phí.

Local Search Problems



Local Search Problems

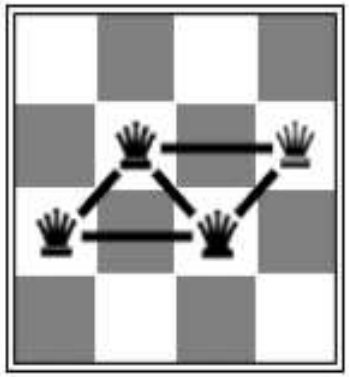
- Tại sao điều này khó?
  - Nhiều trạng thái (Thỉnh thoảng là số vô cùng)
  - Hầu hết những vấn đề gặp phải là NP-đầy đủ
  - Hàm mục tiêu có thể đắt

Nhưng

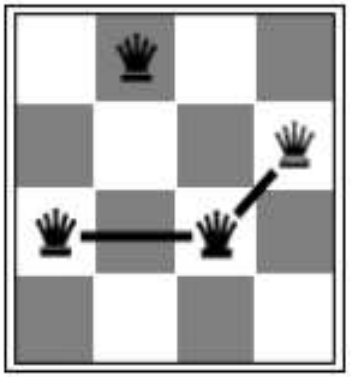
- Sử dụng rất ít bộ nhớ (Thường là hằng số)
- Tìm kiếm được giải pháp hợp lý (nhưng thỉnh thoảng không tối ưu) trong những không gian trạng thái lớn hoặc vô cùng

Local Search Problems

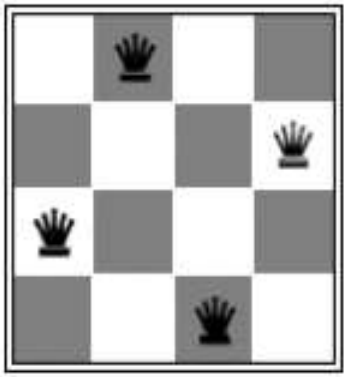
- Ví dụ: n-queens



h(X)=5



h(X)=2



h(X)=1

– Trạng thái X là vị trí của những quân hậu trên bàn cờ
– h(X) là số cặp các con hậu tấn công nhau
– Điều quan trọng là làm cho h(X) nhỏ nhất

Local Search Algorithm Recipe

- 1. Bắt đầu với trạng thái ban đầu X
- 2. Đánh giá các hàng xóm của nó tức là tập hợp tất cả các trạng thái có thể đạt được trong một lần di chuyển từ X
- 3. Chọn một trong các hàng xóm của nó là X*
- 4. Di chuyển đến X* và lặp lại cho đến khi cấu hình hiện tại đạt yêu cầu

Cách định nghĩa vùng lân cận rất quan trọng đối với hiệu suất củathuật toán.

Chọn vùng lân cận nào cũng rất quan trọng

Tiêu chí dừng chung:
• Chạy trong số lần lặp lại được chỉ định
• Chạy trong thời gian được chỉ định
• Chạy cho đến khi không thể di chuyển lên dốc

Các Thuật toán Local Search

1. Hill-climbing
2. Simulated Annealing
3. Beam Search
4. Genetic Algorithms
5. Gradient Descent

Hill Climbing

- Ý tưởng đơn giản:
  - Bắt đầu từ bất cứ trạng thái nào
  - Lặp lại: Di chuyển đến trạng thái tốt hơn (Trạng thái có giá trị của hàm mục tiêu tốt hơn trạng thái bắt đầu)
  - Nếu không có trạng thái nào tốt hơn trạng thái hiện tại thì dừng

- Cách tiếp cận này có gì tốt?

- Cách tiếp cận này có gì xấu?

Hill-climbing search

X ← Initial state

Iterate:

    E ← h(X)

    N ← Neighbors(X)

    For each Xi in N

        Ei ← h(Xi)

    E* ← Highest Ei

    X* ← Xi with highest Ei

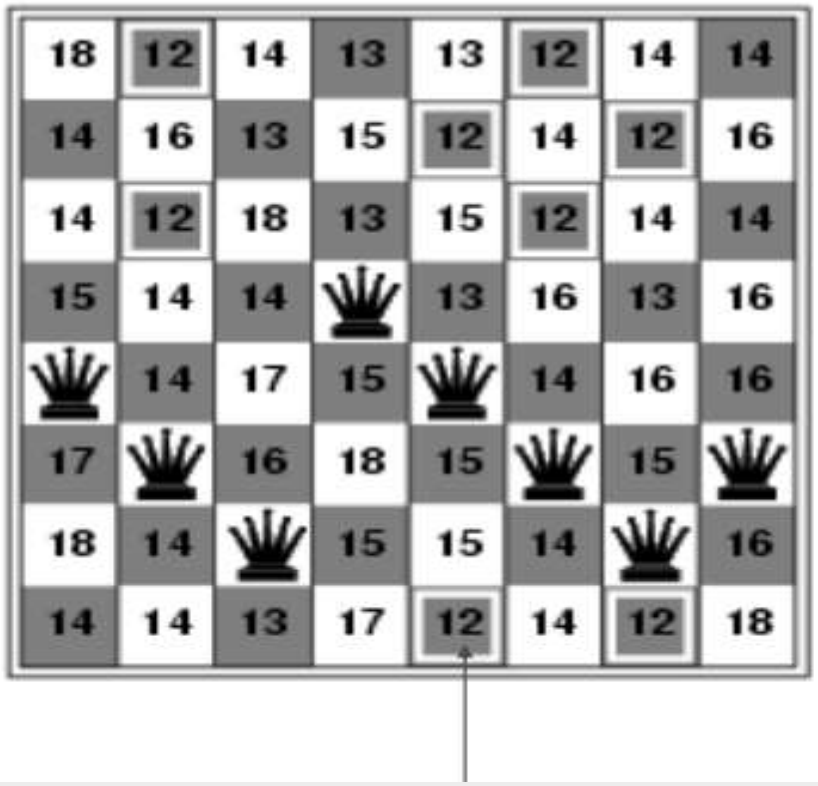
    If E* > E

        X ← X*

    Else

        Return X

Hill-climbing search



- Trạng thái: một bàn cờ có 8 quân hậu
- Mở rộng trạng thái: di chuyển một quân hậu đến bất kỳ ô nào khác trong cùng một cột
- Hàm chi phí heuristic(h): # cặp quân hậu tấn côngLẫn nhau

Những con số này là giá trị của hàm chi phí heuristic nếu bạn di chuyển quân hậu trong cột đó đến ô vuông đó

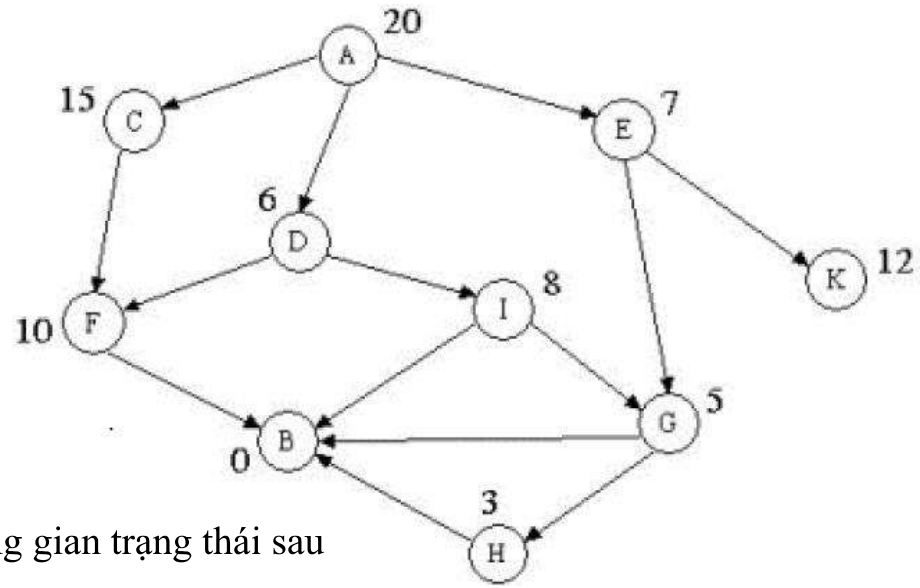
Hill Climbing

Cho đồ thị

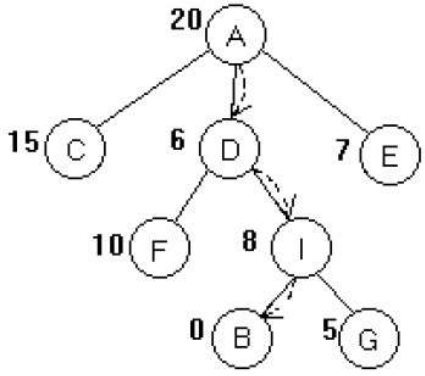
Đầu vào:
Trạng thái đầu: A,
Trạng thái kết thúc: B.

Quá trình thực hiện:

- A được xét → C, D, E.
- Chọn D, vì h(D) = 6 (min), sinh ra F, I.
- Trong số các đỉnh con của D, chọn I, vì h(I) = 8.
- Với I được chọn, sinh ra B và G.
- B là trạng thái kết thúc

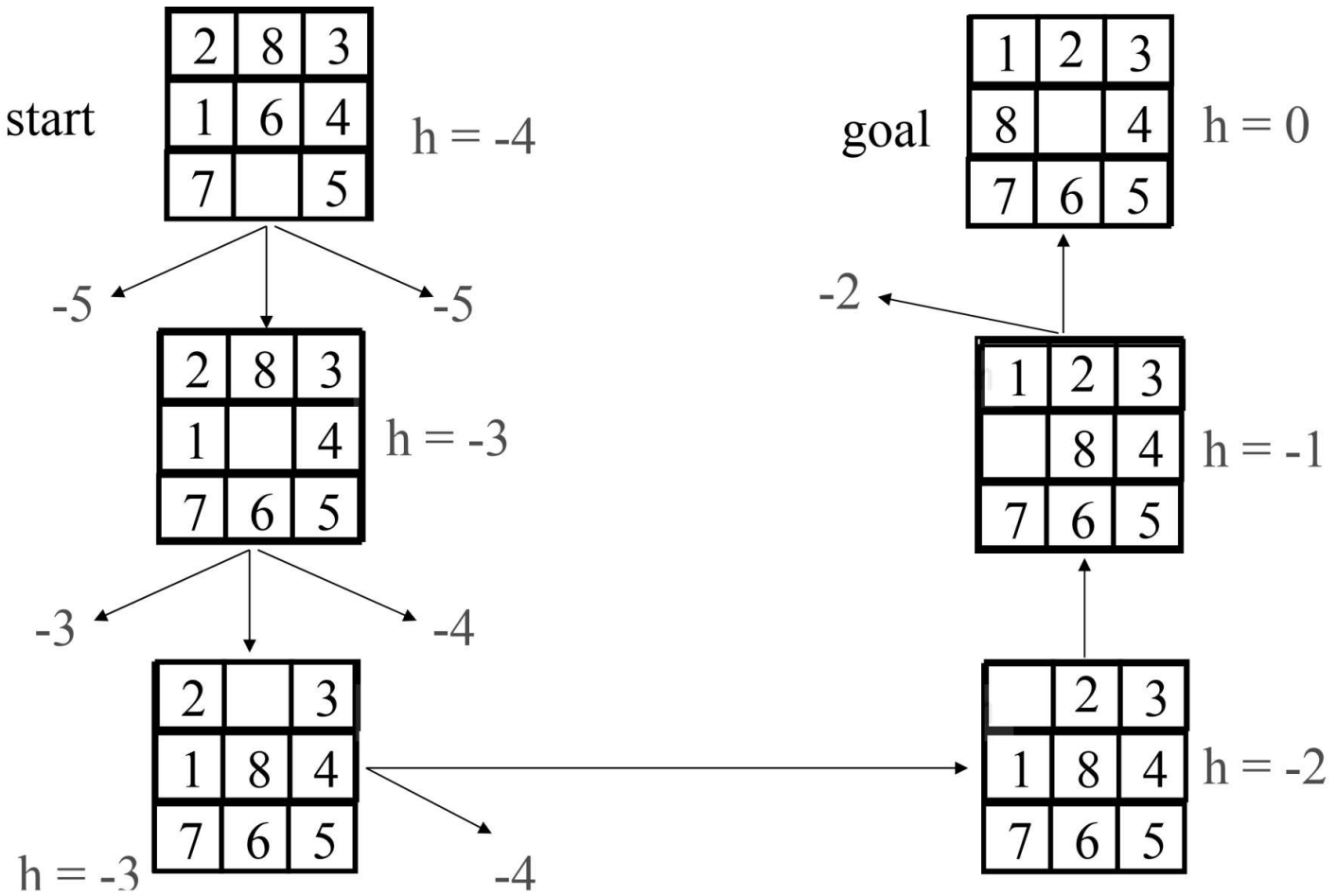


Xét không gian trạng thái sau



Hill Climbing

- Example: 8puzzle



Example

- Example: 8puzzle

Trạng thái đầu **(S)**

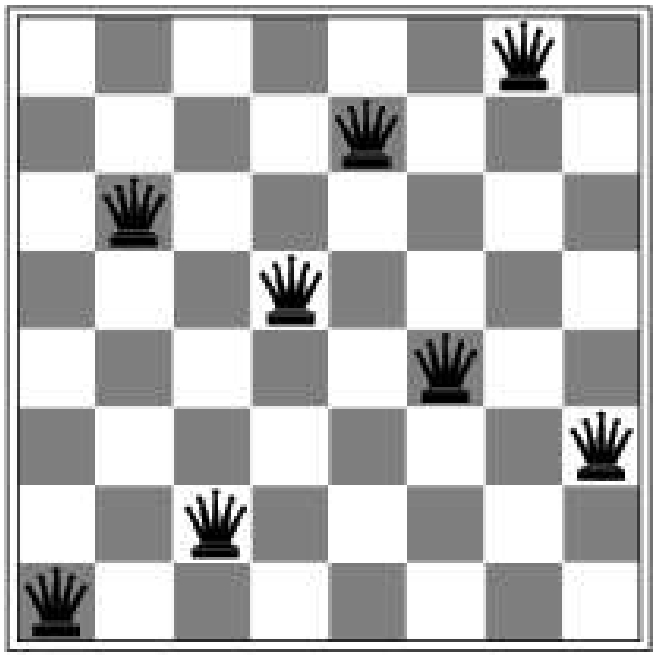
3	5	6
	7	8
4	2	1

Trạng thái đích **(G)**

3	5	6
1	2	4
8		7

Hill-climbing search

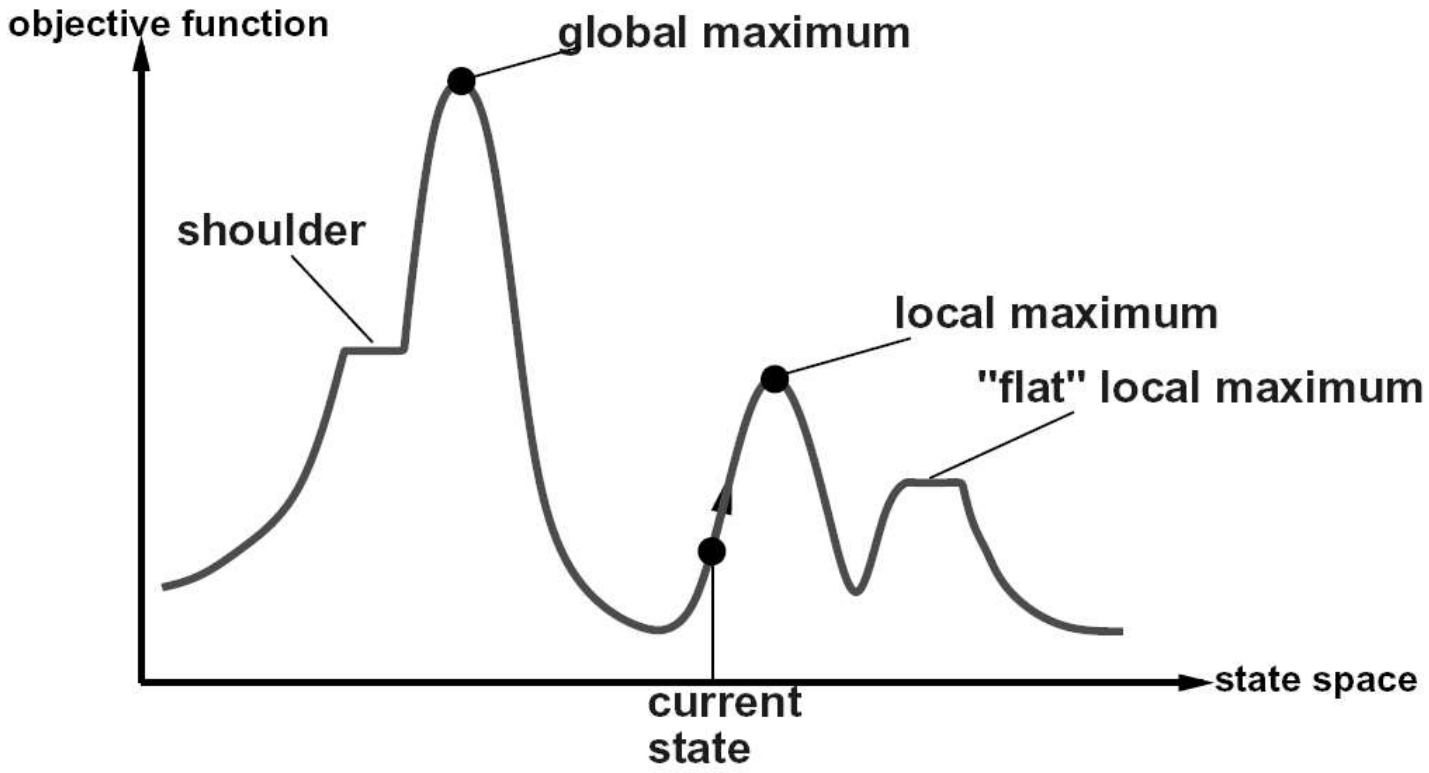
Example: 8-queens problem



- A local minimum with *h = 1*

Hill Climbing Diagram

- Vấn đề: tùy thuộc vào trạng thái ban đầu:
  - có thể bị kẹt ở cực đại cục bộ
  - cực đại cục bộ phẳng:



Random Restart Hill climbing

- Khá rõ ràng đây là gì….
- Tạo trạng thái bắt đầu ngẫu nhiên
- Chạy leo đồi và lưu trữ câu trả lời
- Lặp lại, giữ nguyên câu trả lời tốt nhất hiện tại khi bạn thực hiện
- Dừng lại… khi nào?

Random Restart Hill climbing

- Các vùng lân cận
  - Hãy nhớ rằng chúng tôi đã nói rằng việc xác định đúng vùng lân cận là rất quan trọng đối với hiệu suất của thuật toán
  - Vùng lân cận lớn: Nhiều đánh giá cần thực hiện tại mỗi trạng thái nhưng có nhiều cơ hội hơn để tìm ra một giá trị tối đa tốt
  - Vùng lân cận nhỏ: Ít đánh giá hơn tại mỗi trạng thái nhưng có khả năng bị kẹt trong nhiều giá trị tối đa cục bộ hơn

Simulated Annealing

- Leo đồi không bao giờ thực hiện động tác xuống dốc
- Sẽ thế nào nếu chúng ta thêm một số động tác ngẫu nhiên vào leo đồi để giúp nó thoát khỏi cực đại cục bộ?
- Đây là động lực cho quá trình simulated annealing

Simulated Annealing

Mô tả tìm kiếm Simulated Annealing (mô phỏng luyện kim)

+ Giả sử đang ở trạng thái **u** nào đó. Chọn ngẫu nhiên trạng thái **v** trong lân cận **u**.

- Nếu **v** tốt hơn **u**: sang **v**.
- Nếu **v** không tốt hơn **u**: chỉ sang v với xác suất nào đó.

+ Xác suất này giảm theo hàm mũ của "độ tồi" của trạng thái **v**. Xác suất phụ thuộc vào tham số **T** (nhiệt độ), **T** càng lớn khả năng sang trạng thái tồi càng cao.

+ Xác suất được xác định: $e^{\frac{\Delta}{T}}$

$$với\ \Delta = cost(v) - cost(u)$$

Simulated Annealing

X ← Initial configuration

Iterate:

E ← h(X)

X' ← Randomly selected neighbor of X

E' ← h(X')

If E' ≥ E

X ← X'

E ← E'

Else with probability p

X ← X'

E ← E'

Simulated Annealing

X ← Initial configuration

Iterate:

E ← h(X)

X' ← Randomly selected neighbor of X

E' ← h(X')

If E' ≥ E

X ← X'

E ← E'

Else with probability p

X ← X'

E ← E'

This part is different from hill-climbing

Simulated Annealing

X ← Initial configuration

Iterate:

E ← h(X)

X' ← Randomly selected neighbor of X

E' ← h(X')

If E' ≥ E

X ← X'

E ← E'

Else with probability p

X ← X'

E ← E'

Làm sao để thiết lập p?????

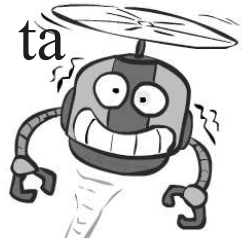
Thiết lập p

- Nếu p quá thấp thì sao?
- Nếu p quá cao thì sao?
- P có nên là hằng số không?

Thiết lập p

- Nếu p quá thấp thì sao?

– Chúng ta không thực hiện nhiều động thái xuống dốc và chúng ta có thể không thoát khỏi nhiều cực đại cục bộ



- Nếu p quá cao thì sao?

– Chúng ta có thể thực hiện quá nhiều động thái không tối ưu

- P có nên là hằng số không?

– Chúng ta có thể thực hiện quá nhiều động thái ngẫu nhiên khi chúng ta ở gần cực đại toàn cục

Thiết lập p

- Giảm p khi các lần lặp tiến triển

  – Chấp nhận nhiều động thái xuống dốc hơn khi bắt đầu, chấp nhận ít hơn khi tìm kiếm tiếp tục

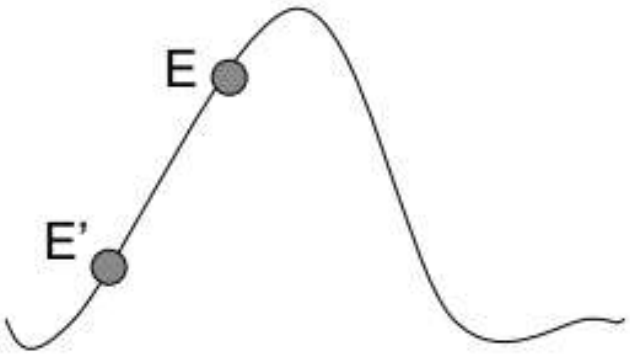
  – Trực giác: khi tìm kiếm tiến triển, chúng ta đang tiến tới các khu vực triển vọng hơn và rất có thể hướng tới mức tối đa toàn cầu

- Giảm p khi E-E’ tăng

  – Chấp nhận ít động thái xuống dốc hơn nếu độ dốc cao

  – Xem trang trình bày tiếp theo để biết trực giác

Giảm p khi E-E’ tăng



E-E’ lớn: chúng ta có thể đang di chuyển về phía cực đại đột ngột nên đừng di chuyển xuống quá nhiều

E-E’ nhỏ: chúng ta có thể đang tiến tới một cực đại trơn tru nên chúng ta muốn thoát khỏi cực đại cục bộ này

Thiết lập p

Cần tham số nhiệt độ T

• Nếu E’ ≤ E, chấp nhận di chuyển xuống dốc với xác suất p= e-(E-E’)/T

• Bắt đầu với nhiệt độ cao T, (cho phép nhiều di chuyển xuống dốc hơn khi bắt đầu)

• Giảm T dần dần khi số lần lặp tăng lên (cho phép ít di chuyển xuống dốc hơn)

• Annealing schedule mô tả cách T giảm ở mỗi bước

Simulated Annealing

X ← Initial configuration

Iterate:

Do K times:

E ← h(X)

X' ← Randomly selected neighbor of X

E' ← h(X')

If E' ≥ E

X ← X'

E ← E'

Else with probability $p = e^{-(E-E')/T}$

X ← X'

E ← E'

$T = \alpha T$

Simulated Annealing

X ← Initial configuration

Iterate:

Do K times:

E ← Eval(X)

X' ← Randomly selected neighbor of X

E' ← Eval(X')

If E' ≥ E

X ← X'

E ← E'

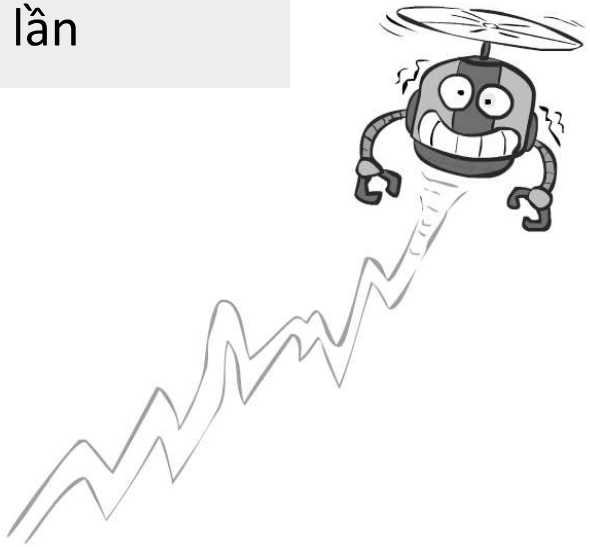
Else with probability $p = e^{-(E-E')/T}$

X ← X'

E ← E'

$T = \alpha T$

Giữ nhiệt độ cố định
và lặp lại K lần



Lịch trình làm mát theo cấp số nhân
$T(n) = \alpha\, T(n-1)$ với $0 < \alpha < 1$

Những điều cần lưu ý

- Thiết kế neighborhood là rất quan trọng
- Nhiều thông số cần điều chỉnh ví dụ như α, K, nhiệt độ ban đầu
- Simulated annealing thường tốt hơn việc leo đồi nếu bạn có thể tìm thấy các thông số phù hợp
- Nếu lịch trình hạ T đủ chậm, thuật toán sẽ tìm ra một tối ưu toàn cục với xác suất gần bằng 1
- Trong thực tế, việc đạt được tối ưu toàn cục có thể mất một số lượng lớn các lần lặp lại