**LOCAL SEARCH ALGORITHMS**

**1.Review 2 nhóm thuật toán tìm kiếm trước :**

Lưu không gian trạng thái 🡪 không hiệu quả khi không gian quá lớn, ví dụ DFS không hiệu quả, quá nhiều trạng thái để lưu

**2. Nhóm local search algorithms khắc phục được hiệu quả trên**

Sau khi chọn được nút 🡪 đánh giá với nút hiện tại xem nút nào tốt nhất thì chọn.

**3. Các loại Hill Climbing**

**Simple hill climbing ( leo đồi đơn giản )**

* Là cách đơn giản nhất để triển khai thuật toán leo đồi
* Chỉ kiểm tra từng trạng thái lân cận của trạng thái hiện tại.
* Nếu tìm thấy trạng thái tốt hơn trạng thái hiện tại thì di chuyển
* Hoạt động :

1. Kiểm tra xem trạng thái hiện tại có phải là đích không. Nếu đúng, dừng lại.
2. Tìm **tất cả** các trạng thái lân cận.
3. Chọn **trạng thái lân cận tốt nhất** (giá trị heuristic thấp nhất).
4. Nếu **không có trạng thái nào tốt hơn**, **mắc kẹt ở cực đại cục bộ** → Dừng lại.
5. Nếu tìm thấy trạng thái tốt hơn, **di chuyển đến trạng thái đó và quay lại bước 1**.

Mã giả :

Simple hill climbing( trangthai\_dau)

Trangthai\_hientai 🡨 khoitao( tranthai\_dau )

Kiểm tra( trangthai\_hientai) nếu là goal thì return trangthai\_hientai và f(trangthai\_hientai) , không thì tiếp tục.

While True:

Neighbours 🡨 get( trangthai\_hientai)

neighbours\_tamthoi = neighbours.copy()

trangthai\_totnhat = trangthai\_hientai

For trangthai in neighbours\_tamthoi :

Neighbours\_tamthoi bỏ neighbours\_tamthoi[i ]

Đánh giá trangthai với hàm f( trangthai), if f(trangthai) tốt hớn f(trangthai\_totnhat ), thì :

Trangthai\_totnhat = trangthai

If trangthai\_tonhat == trangthai\_hientai thì:

Return trangthai\_hientai, f(trangthai\_hientai ) #không tìm thấy trạng thái tốt hơn)

Trangthai\_hientai = trangthai\_totnhat #cập nhật trạng thái hiện tại

|  |  |
| --- | --- |
| Nút được xét | Nút lân cận được chọn |
| (0, A) | ( 2, B2 ), (3, C3), (1, D1) |
| (2, B2 ) | (3, E3 ), ( 4, L4 ) |
| ( 3, E3 ) | ( 5, I5) |
| (5, I5 ) | Đích |
|  |  |
|  |  |
|  |  |

**Steepest-Ascent Hill Climbing ( Leo đồi dốc nhất )**

Chi phí trạng thái sau phải tốt hơn trạng thái trước

ở thuật toán tìm kiếm của 2 nhóm trước : chỉ cần trạng thái có chi phí thấp tốt hớn chi phí trước đó là sẽ chọn

còn ở thuật toán leo đồi thì chi phí sau phải tốt hơn tất cả các chi phí trước thì mới chọn.

**Stochastic hill climbing ( Leo đồi ngẫu nhiên )**

**Lấy lân cận ngẫu nhiên, nếu không tốt hoặc không còn nút lân cận** 🡪 sẽ quay lại và chọn 1 lân cận khác của nút phía trước

**Simulated Annealing**

**Beam Search**

**Thuật toán di truyền :**

Bước 1 : Khởi tạo quần thể ( tương đương mở rộng trạng thái )

Bước 2 : Tính hàm thích nghi ( tương đương với tính hàm chi phí )

Bước 3 : Lựa chọn cá thể để lai ghép ( cá thể ở đây là trạng thái, chọn 2 trạng thái có chi phí tốt nhất để lai ghép )

Bước 4 : Lai ghép :

1. Chọn điểm lai ghép : khi lai ghép 2 trạng thái có thể sẽ bị trùng nên cần chọn điểm lai ghép sao cho khi lại kết quả không bị trùng. Lưu ý cần chọn vị trí

Bước 5 :