Local Search Algorithms and optimization problems

Optimization Problems

Global search: tìm kiếm trên toàn bộ không gian, tuy nhiên khi không gian quá lớn khiến việc tìm kiếm là bất khả thi nên không phải trường hợp nào cũng áp dụng được.
Local search: tìm kiếm trên một không gian cục bộ để giảm đi không gian tìm kiếm nhưng vẫn đảm bảo có được solution chấp nhận được (CĐại/CTiểu cục bộ).

> Tuy nhiên local search sẽ không thể đảm bảo được 2 tính chất completeness & optimality.

State-space landscape:

Sử dụng *objective function* để định nghĩa các *state*, mục tiêu đặt ra là tìm các điểm min/max trên space.
Đối với global search sẽ tìm được các điểm *maximum* & *mininum*, ngoài ra còn có các vị trí gọi là *shoulder* trước khi chạm đến peak (dễ nhầm lẫn với *flat*)
Khi sử dụng local search, cần chú ý các điểm min/max tìm được sẽ có thể chỉ là cục bộ. Ngoài ra còn có những điểm *flat* khiến không tìm được thêm điểm nào để đi (nếu nó quá rộng).

Hill-climbing Search

Tương tự như `Greedy best-first search`, expand theo cách lựa chọn successor (state --> goal: cheapest) tốt nhất.
Khi gặp `flat`, nên đặt ra một limit nhất định để tránh trường hợp `flat` đó quá rộng. Ước tính nếu giới hạn sideways 100 bước thì tỉ lệ success tăng từ 14% lên 94%.

Stochastic hill climbing

Thay đổi quy luật chọn successor, có thể là random để tránh trường hợp rule ban đầu đặt ra là xấu.

First-choice hill climbing

Trong trường hợp có quá nhiều successor, khôg thể tính toán hết để tìm ra cái tốt nhất thì cứ lấy đại cái đầu tiên tốt hơn cái hiện tại rồi move luôn để tránh mất thời gian.

Random-restart hill climbing

Khi nào bị `stuck` thì random chạy lại từ đầu. Giải quyết tốt được vấn đề khi `state space` có quá nhiều maxima.

Local Beam Search

- Thay gì như thuật toán trên là chọn ra cái tốt nhất thì `Local Beam` giữ lại `k states` tốt. (*k* là tùy theo các điều kiện đi kèm như vùng nhớ, thời gian xử lí,...).
- Từ một state ban đầu expand ra rồi giữ lại `k state` tốt nhất. `k state` này sau đó expand ra hết, trong tổng số tất cả các successor của `k state` expand ra lại chỉ giữ lại `k` successor.

Genetic Algorithm

> Lấy cảm hứng từ mô hình tiến hóa của các giống loài.

Thuật toán:

- Trong một quần thể ban đầu, ta sẽ sử dụng `selection` để giữ lại một số lượng (`k`) cá thể nhất định, bỏ những cá thể xấu ra.
- Sau đó đem `k` cá thể này đi `crossover` (lai ghép). Số lượng bắt cặp lai ghép là do tự quy định.
- Số cặp lai ghép này tiếp tục được xử lí `mutation` (đột biến) một cách ngẫu nhiên. Tất cả chúng sẽ được đi qua `fitness calculation` để xét xem tìm được kết quả chưa, nếu không thì đi tiếp.
- Cứ như vậy lặp lại đến khi nhận được kết quả có thể chấp nhận.

Đánh giá:

- Do việc `mutation` & `crossover` tạo ra những cá thể hoàn toàn mới khiến các state sẽ nhảy rất ngẫu nhiên chứ không còn đơn giản là expand ra các neighbor, không còn đơn giản là local mà mang tính global luôn.
- Do thuật toán không hề có định hướng nên thời gian hội tụ sẽ bị lâu hơn so với các thuật toán còn lại.