Tối ưu hóa (2)

TS. Đỗ Đức Đông
dongdoduc@gmail.com

Các cách tiếp cận

Tối ưu tổ hợp	Tối ưu liên tục
• Các phương pháp truyền thống Chứng minh hội tụ hoặc tỷ lệ tối ưu • Các phương pháp dựa trên thực nghiệm + Heuristic cấu trúc (constructive heuristic) + Tìm kiếm địa phương (local search) + Metaheurisics: - GA (Genetic Algorithms) - ACO (Ant Colony Optimization) - Memetic algorithm	• Quy hoạch tuyến tính • Quy hoạch phi tuyến + Tìm kiếm địa phương - Phương pháp gradient …. + Tối ưu toàn cục - Quy hoạch lồi, hiệu lồi - Tìm kiếm ngẫu nhiên (Monter-Carlo) - GA (Genetic Algorithms) …

Nội dung

- Lớp bài toán P, NP, NP-khó và NP-đầy đủ
- Giải pháp cho bài toán NP-Khó
  - Các phương pháp truyền thống:
    - Tỷ lệ tối ưu (ε-Xấp xỉ tuyệt đối)
  - Các phương pháp dựa trên thực nghiệm

    + Heuristic cấu trúc (constructive)

    + Tìm kiếm địa phương (local search)

    + Metaheurisics:

      - GA (Genetic Algorithms)

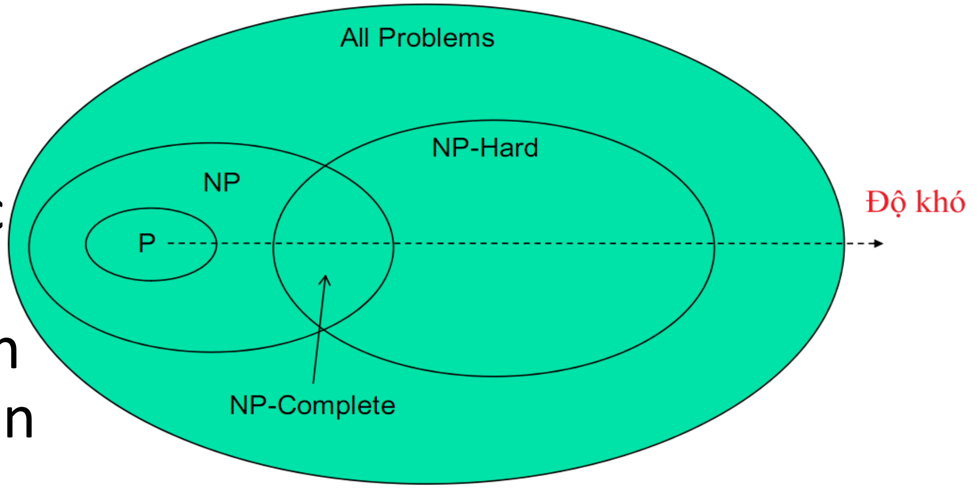
      - ACO (Ant Colony Optimization)

      - Memetic algorithm

      …

Lớp bài toán P, NP và NP-khó và NP-đầy đủ

- P là lớp các bài toán giải được bằng ***thuật toán đơn định*** trong ***thời gian đa thức***.
- NP là lớp tất cả các bài toán giải được bằng ***thuật toán không đơn định*** trong ***thời gian đa thức***. Theo cách khác, NP là lớp các bài toán có thể kiểm định (verifiable) được một xác thực (certificate) trong thời gian đa thức.
- $P = NP?$
- Khái niệm "dẫn về được ": Ta nói bài toán B dẫn về được bài toán A một cách đa thức, kí hiệu B $\prec$ A, nếu có thuật toán đơn định đa thức giải được A thì cũng có thuật toán đơn định đa thức để giải B (bài toán A "khó hơn" bài toán B, hay B "dễ hơn" A)
- A là bài toán NP-khó nếu $\forall L \in NP$: L $\prec$ A
- Bài toán A gọi là NP-đầy đủ nếu A là NP-khó và A $\in$NP



- Hamiltonian cycle problem: NP-đầy đủ
- Travelling salesman problem: NP-khó
- Graph isomorphism problem: ???

Giải pháp cho bài toán NP-khó

- Tìm nghiệm chính xác của bài toán bằng thuật toán đơn định thời gian hàm mũ
- Tìm nghiệm gần đúng của bài toán bằng thuật toán đơn định thời gian "đa thức".

Thuật toán ε-Xấp xỉ tuyệt đối

Định nghĩa: E là bài toán cực trị hóa, H là thủ tục tìm một nghiệm nào đó cho E. Kí hiệu OPT(I) là nghiệm tối ưu của bài toán E đối với thể hiện I. Kí hiệu H(I) là nghiệm gần đúng của E do H tìm ra.

Thủ tục H được gọi là thuật toán ε-Xấp xỉ tuyệt đối khi và chỉ khi: $|OPT(I)-H(I)| \leq \varepsilon$ cho mọi trường hợp I của bài toán E

Ví dụ: Bài toán lưu trữ tối đa số chương trình

- Dữ liệu vào: n chương trình với các dung lượng nhớ (độ dài) $d_1,\ldots,d_{n,}$ hai băng nhớ với dung lượng mỗi băng là L.
- Yêu cầu: Hãy ghi các chương trình lên hai băng nhớ với số lượng tối đa, mỗi chương trình chỉ được ghi liên tục trên một băng nhớ.
- Bài toán này đã được chứng minh là NP-đầy đủ.

Thuật toán 1-xấp xỉ tuyệt đối cho bài toán

Sắp xếp các chương trình theo thứ tự tăng dần, giả sử $d_1 \le d_2 \le \ldots \le d_n$

```
 i:=1;
for (j=1; j≤2; j++) {
        do_dai:=0;
        while ((do_dai+d_i  ≤ L) && (i≤n)) {
                Ghi chương trình i vào băng nhớ j;
                do_dai:=do_dai +d_i ;
                i:=i+1;
        }
}
```

Chứng minh được: $OPT(I) \le H(I)+1$

Chứng minh

- Ta đặt h=H(I) là nghiệm của thuật toán, như vậy h là số lượng chương trình được lưu trữ vào hai băng nhớ theo cách sắp đặt của thuật toán xấp xỉ trên theo thứ tự $d_1 \le d_2 \le \ldots \le d_n$.
- Gọi p là số chương trình được ghi lên một băng nhớ có độ dài 2L theo thứ tự $d_1 \le d_2 \le \ldots \le d_n$.
- $h \le OPT(I) \le p$ và $\sum_{i=1}^{p} d_i \le 2L$
- Ta chứng minh $OPT(I) \le h + 1$, thông qua việc chứng minh $p \le h + 1$
- Giả sử $p > h + 1$ hay $p \ge h + 2$ nên $\sum_{i=1}^{h+2} d_i \le \sum_{i=1}^{p} d_i \le 2L$ (1)
- Gọi m là số chương trình ghi trên băng 1 theo thuật toán xấp xỉ trên thì $\sum_{i=1}^{m} d_i + d_{m+1} > L$ và $\sum_{i=m+1}^{h} d_i + d_{h+1} > L$
  nên $\sum_{i=1}^{h} d_i + d_{h+1} + d_{h+2} > 2L$, trái với (1)

Heuristic cấu trúc (constructive heuristic)

- Bài toán TƯTH ứng với một bộ ba (S, f, Ω), trong đó S là tập hữu hạn trạng thái (lời giải tiềm năng hay phương án), f là hàm mục tiêu xác định trên S, còn Ω là tập các ràng buộc. Phương án $s = (s_1, \dots, s_n)$, trong đó $s_i \in C_i$
- Lời giải của bài toán TƯTH được xây dựng theo cách mở rộng tuần tự, mở rộng không quay lui. Từ thành phần khởi tạo trong tập C_1, bằng cách thêm vào các thành phần mới theo phương thức ngẫu nhiên hay tất định dựa trên các quy tắc heuristic đã chọn.
- Các quy tắc heuristic thường được xây dựng dựa trên các kết quả phân tích toán học hoặc kinh nghiệm.

Procedure Heuristic cấu trúc;

Begin

    s $\leftarrow$ **chọn** c_1 **trong** C_1**;**

    while (chưa xây dựng xong lời giải) do

        $\boxed{c_i \leftarrow \textbf{GreedyComponent(s,} C_i\textbf{);}}$

        s $\leftarrow$ **s** ^ c_i**;**

    end-while

    Đưa ra lời giải s**;**

End;

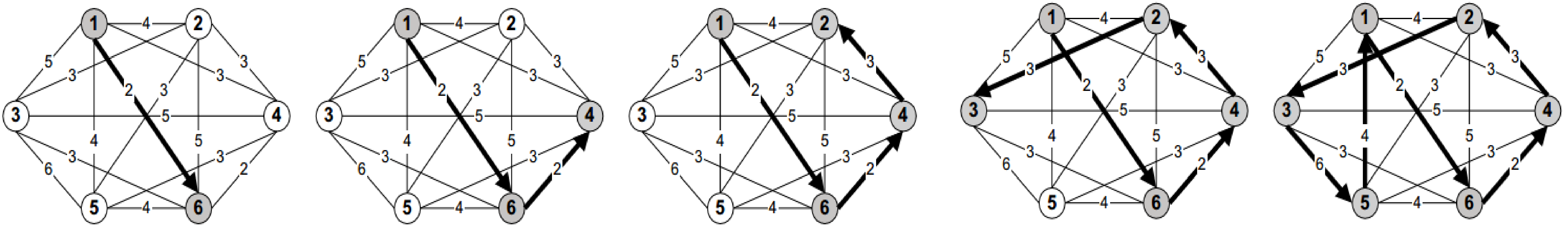
Ví dụ 1: Thuật toán Láng giềng gần nhất (nearest neighbor algorithm) giải bài toán người chào hàng (travelling salesman problem)

1. Khởi tạo, xuất phát tại một thành bất kỳ; $s = (s_1)$;

2. Lựa chọn, gọi $s = (s_1, s_2, .., s_k)$ là hành trình hiện tại $(k < n)$, tìm thành phố c tiếp theo mà gần s_k nhất;

3. Thêm thành phố $s_{k+1} = c$ vào hành trình, $s = (s_1, s_2, .., s_k, s_{k+1})$

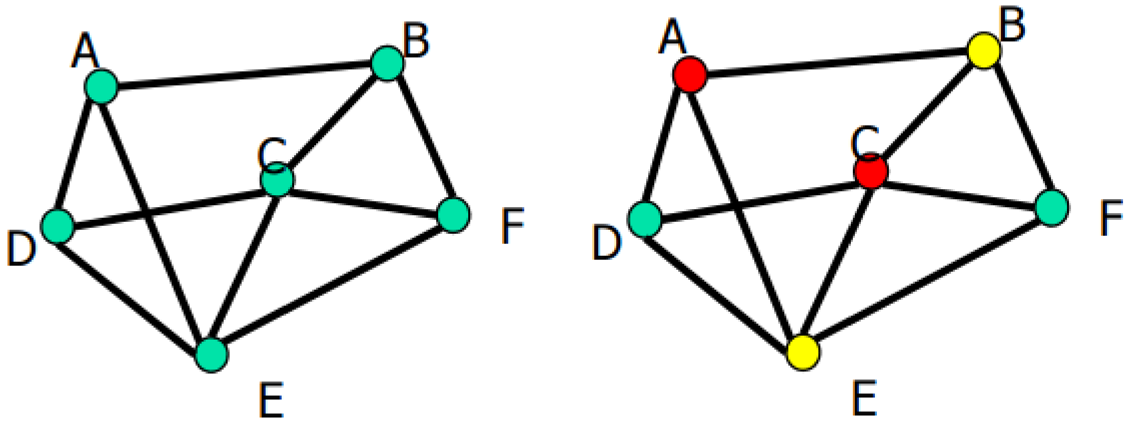
4. Nếu đủ n thành phần thì kết thúc, nếu không quay lại 2.



Ví dụ 2: Bài toán tô màu đồ thị

Quy tắc heuristic 1: Chọn đỉnh có bậc lớn nhất, tô bằng một nhỏ nhất thỏa mãn.

Quy tắc heuristic 2: Chọn đỉnh còn ít khả năng tô màu nhất, tô bằng một nhỏ nhất thỏa mãn.



Nhận xét về phương pháp Heuristic kiến trúc

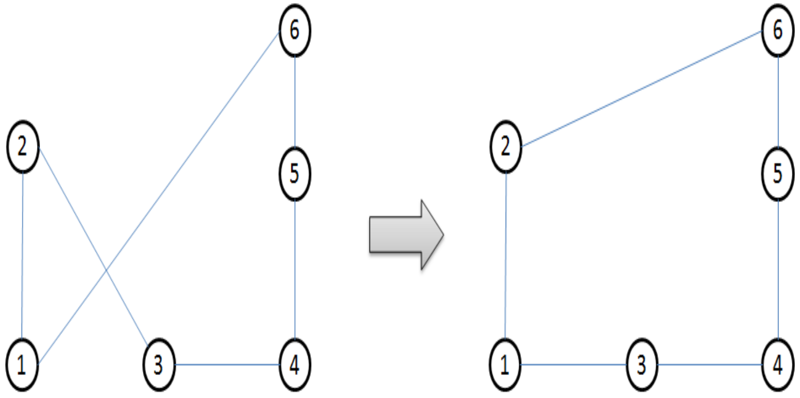
- Heuristic kiến trúc thông dụng, giải quyết cho nhiều bài toán khác nhau;
- Heuristic kiến trúc chạy nhanh, không ước lượng được độ tốt của lời giải như thuật toán ε-Xấp xỉ tuyệt đối.
- Quy tắc heuristic quyết định chất lượng lời giải;
- Heuristic kiến trúc dùng để xây dựng lời giải ban đầu cho các phương pháp khác.

Tìm kiếm địa phương (local search)

- Các tên gọi khác nhau: Tìm kiếm cục bộ, leo đồi.
- Bắt đầu từ một phương án chấp nhận được, lặp lại bước cải tiến lời giải nhờ các thay đổi cục bộ.
- Xác định được *cấu trúc lân cận* của mỗi phương án (lời giải) đang xét, tức là những phương án chấp nhận được, gần với nó nhất, nhờ thay đổi một số thành phần.
- Lân cận k-*thay đổi*, tức là *lân cận* bao gồm các phương án chấp nhận được khác với phương án đang xét nhờ thay đổi nhiều nhất k thành phần.

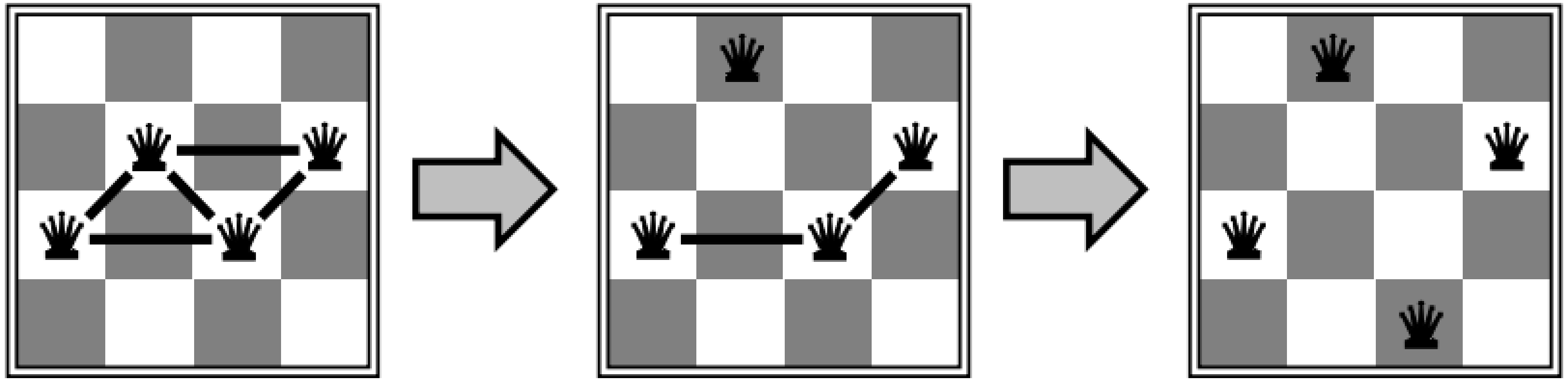
  Ví dụ: Lân cận *2-thay đổi* của một lời giải s trong bài toán TSP bao gồm tất cả các lời giải s' có thể nhận được từ s bằng cách đổi hai cạnh.

- Dựa theo hai chiến lược: Chiến lược *tốt nhất* và chiến lược *tốt hơn*.



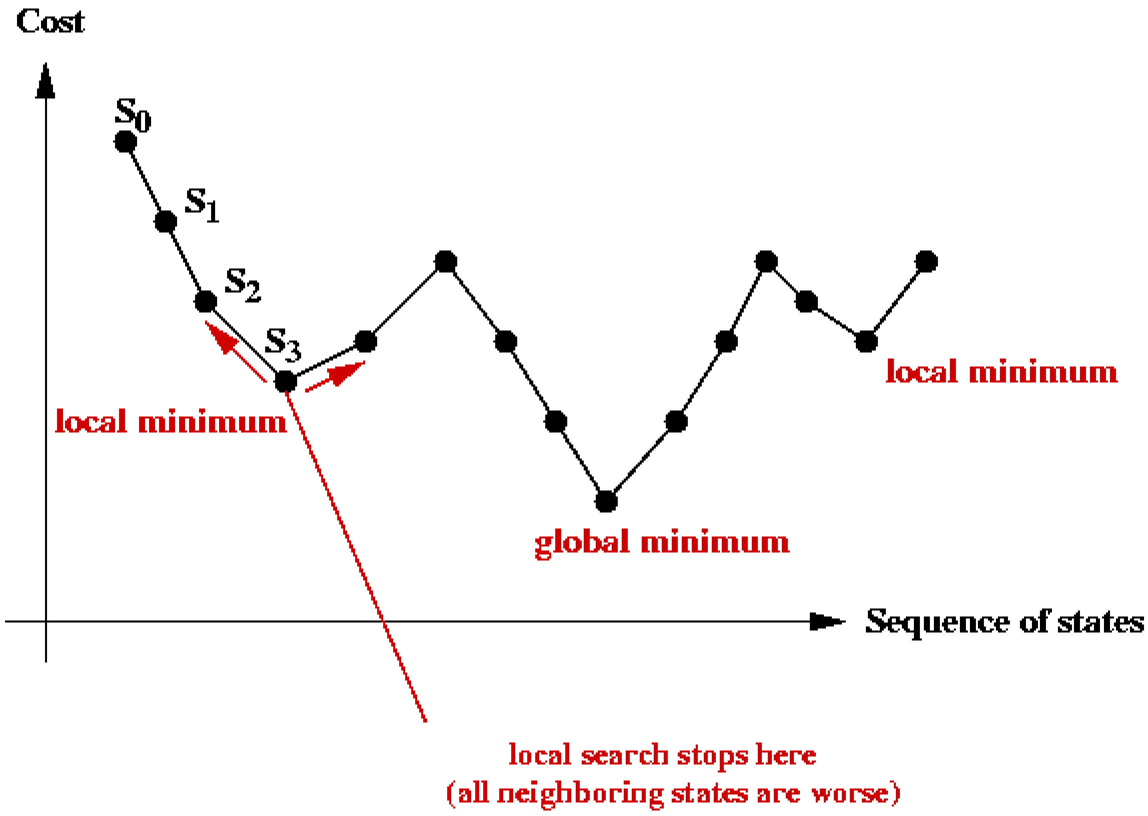
Ví dụ: Bài toán xếp hậu

Cần đặt n quân hậu lên bàn cờ kích thước $n \times n$ để không có 2 quân nào chiếu nhau.



Nhận xét phương pháp tìm kiếm địa phương

- Không ước lượng được độ tốt của lời giải như thuật toán ε-Xấp xỉ tuyệt đối.
- Lân cận *k-thay đổi*, nếu *k* lớn thì nhiều lân cận, dễ tìm được nghiệm tốt nhưng độ phưc tạp lớn. Định nghĩa lân cận quyết định chất lượng lời giải của phương pháp;
- Chiến lược *tốt nhất,* thực hiện chọn lời giải tốt nhất trong lân cận để làm lời giải cải tiến. Tuy nhiên, khi bài toán cỡ lớn có thể không tìm được lời giải tốt nhất do bị hạn chế về thời gian.
- Chiến lược *tốt hơn*, ta chọn phương án đầu tiên trong lân cận, cải thiện được hàm mục tiêu.



GRASP
(Greedy Randomized Adaptive Search Procedure)

```
While (chưa kết thúc) {
        Heuristic cấu trúc xây dựng lời giải S;
        Tìm kiếm cục bộ cải tiến lời giải S;
}
```

Đọc và trình bày

1) Comparison of Heuristic Algorithms for the N-Queen Problem
2) Greedy constructive heuristic and local search algorithm for solving Nurse Rostering Problems
3) Large Neighbourhood Search Algorithms for the Founder Sequences Reconstruction Problem