

# BÀI TẬP LÝ THUYẾT TUẦN 4

## Nhập môn Trí tuệ Nhân tạo

Nguyễn Hồng Yến – MSSV: 23280099

GVHD: PGS.TS Nguyễn Thanh Bình

### Phân tích tính tối ưu của giải thuật $A^*$ và các điều kiện của hàm heuristic

**Câu 1.** Giả sử  $h(n)$  là admissible, chứng minh rằng  $A^*$  sử dụng Tree Search sẽ cho lời giải tối ưu.

Nhắc lại kiến thức nền:

- Trong giải thuật  $A^*$ , mỗi node  $n$  được đánh giá bởi hàm:

$$f(n) = g(n) + h(n)$$

trong đó  $g(n)$  là chi phí thật từ trạng thái xuất phát đến  $n$ , và  $h(n)$  là chi phí ước lượng nhỏ nhất từ  $n$  đến đích (goal).

- Hàm  $h(n)$  được gọi là **admissible** nếu:

$$h(n) \leq h^*(n)$$

với  $h^*(n)$  là chi phí thật tối thiểu để đi từ  $n$  đến đích. Khi đó,  $h(n)$  không bao giờ đánh giá quá cao chi phí thật.

#### 1. Trực giác của chứng minh

Giả sử  $C^*$  là chi phí của đường đi tối ưu. Với mọi node  $n$  nằm trên đường tối ưu, ta có:

$$f(n) = g(n) + h(n) \leq g(n) + h^*(n) = C^*$$

Điều này có nghĩa là **mọi node trên đường tối ưu đều có giá trị  $f(n)$  không vượt quá  $C^*$** .

Ngược lại, nếu có một goal  $G_2$  không tối ưu với chi phí  $C > C^*$ , thì:

$$f(G_2) = g(G_2) + h(G_2) = C + 0 = C > C^*$$

$A^*$  luôn chọn mở rộng node có  $f(n)$  nhỏ nhất trong hàng đợi. Do đó, trước khi  $G_2$  (với  $f > C^*$ ) được chọn để mở rộng, tất cả các node có  $f \leq C^*$  — tức là các node nằm trên đường tối ưu — sẽ được mở rộng trước.

Kết luận:  $A^*$  sẽ luôn mở rộng (và tìm thấy) goal có chi phí  $C^*$  trước bất kỳ goal nào có chi phí lớn hơn, nên  **$A^*$  Tree Search với  $h$  admissible luôn cho lời giải tối ưu.**

*Trực giác:* Đường đẳng mức  $f = C^*$  bao quanh toàn bộ đường đi tối ưu, còn mọi lời giải kém tối ưu đều có  $f > C^*$ .  $A^*$  quét dần theo  $f$  tăng, nên gặp lời giải tốt nhất trước.

## 2. Chứng minh hình thức

**Bổ đề 1.** (Giới hạn trên của  $f$  trên đường tối ưu) Với mọi node  $n$  trên đường tối ưu, nếu  $h$  admissible thì:

$$f(n) = g(n) + h(n) \leq g(n) + h^*(n) = C^*.$$

*Chứng minh:* Do  $h(n) \leq h^*(n)$  theo định nghĩa admissible. Thay vào biểu thức  $f(n)$ , suy ra  $f(n) \leq C^*$ .

**Bổ đề 2.** (Giới hạn dưới của  $f$  tại goal kém tối ưu) Với mọi goal  $G_2$  có chi phí  $C > C^*$ :

$$f(G_2) = g(G_2) = C > C^*.$$

**Định lý.** (Tính tối ưu của  $A^*$  với  $h$  admissible trong Tree Search) Giả sử phản chứng rằng  $A^*$  Tree Search trả về goal  $G_2$  không tối ưu với  $C > C^*$ . Tại thời điểm  $G_2$  được lấy ra khỏi hàng đợi (để kiểm tra goal), ta có:

$$f(G_2) = g(G_2) = C > C^*.$$

Theo Bổ đề 1, trên đường tối ưu vẫn còn ít nhất một node  $n$  có  $f(n) \leq C^* < C$ . Do  $A^*$  luôn mở rộng node có  $f$  nhỏ nhất,  $G_2$  (với  $f > C^*$ ) không thể được chọn trước node  $n$  (với  $f \leq C^*$ ).

Mâu thuẫn. Do đó giả định sai. Suy ra  $A^*$  Tree Search với  $h(n)$  admissible **luôn trả về lời giải tối ưu.**

$A^* \text{ Tree Search} + \text{admissible } h(n) \Rightarrow \text{tối ưu.}$
--

**Câu 2.** Giả sử  $h(n)$  là consistent, chứng minh rằng  $A^*$  sử dụng Graph Search sẽ cho lời giải tối ưu.

Nhắc lại kiến thức (Consistency/Monotonicity).

- Heuristic  $h$  được gọi là **consistent (monotonic)** nếu với mọi cạnh  $(n \xrightarrow{a} n')$  có chi phí  $c(n, a, n') > 0$ ,

$$h(n) \leq c(n, a, n') + h(n').$$

- Khi đó, với  $f(n) = g(n) + h(n)$  ta có tính đơn điệu:

$$f(n') = g(n') + h(n') = g(n) + c(n, a, n') + h(n') \geq g(n) + h(n) = f(n).$$

Tức là  $f$  không giảm dọc theo bất kỳ đường đi nào.

- Trong **Graph Search**, ta dùng tập **CLOSED** (đã mở rộng) để tránh lặp lại trạng thái. Kết quả quan trọng: nếu  $h$  consistent, một khi một node được lấy ra để mở rộng,  $g$ -cost của nó là tối ưu  $\Rightarrow$  không cần “reopen” node.

## 1. Trực giác của chứng minh

Do  $f$  không giảm dọc theo đường đi, nên  $A^*$  sẽ mở rộng các node theo thứ tự  $f$  không giảm. Khi một node  $n$  được lấy ra để mở rộng, **mọi đường đi khác đến  $n$  nếu còn trên OPEN đều có  $f$  không nhỏ hơn** (vì muốn tới  $n$  phải đi qua các node trung gian có  $f$  không nhỏ hơn  $f$  hiện tại). Vì vậy, **đường đi hiện có đến  $n$  là rẻ nhất  $\Rightarrow g(n)$  đã tối ưu tại thời điểm mở rộng  $n$ .**

Áp dụng cho node đích  $G$ : khi  $G$  được lấy ra để mở rộng, ta đã có  $g(G) = g^*(G) = C^*$  (chi phí tối ưu). Do đó  $A^*$  (Graph Search) trả về **lời giải tối ưu**.

*Trực giác:  $f$  không giảm dọc đường đi  $\Rightarrow A^*$  “khóa” dần những  $g$ -cost tối ưu khi mở rộng. Đến lượt goal, chi phí đã tối ưu.*

## 2. Chứng minh hình thức

**Bổ đề 1 (Đơn điệu của  $f$ ).** Nếu  $h$  consistent, với mọi cạnh  $(n \xrightarrow{a} n')$  ta có

$$f(n') \geq f(n).$$

*Chứng minh.* Theo định nghĩa consistency:  $h(n) \leq c(n, a, n') + h(n')$ . Suy ra

$$f(n') = g(n') + h(n') \geq g(n) + c(n, a, n') + h(n') \geq g(n) + h(n) = f(n).$$

**Bổ đề 2 (Tối ưu hoá  $g$  khi mở rộng).** Giả sử  $h$  consistent. Trong  $A^*$  Graph Search, khi một node  $n$  được lấy ra khỏi **OPEN** để mở rộng, thì  $g(n) = g^*(n)$  (chi phí từ start tới  $n$  là tối ưu). *Chứng minh phản chứng.* Giả sử  $n$  được mở rộng với  $g(n) > g^*(n)$ . Xét một đường đi tối ưu từ start tới  $n$ :

$$s \rightarrow \dots \rightarrow p \xrightarrow{a} n.$$

Tại thời điểm  $n$  được chọn để mở rộng, hãy xét *node đầu tiên  $x$  trên đường tối ưu* mà chưa nằm trong CLOSED. Gọi  $y$  là *tiền nhiệm* của  $x$  trên đường tối ưu; khi đó  $y \in \text{CLOSED}$  (vì  $x$  là node đầu tiên chưa đóng).

Do  $y$  đã được mở rộng trước đó,  $x$  hẳn đã từng được đưa vào OPEN (từ  $y$ ). Mặt khác, vì  $h$  consistent, theo Bổ đề 1,  $f(x) \geq f(y)$  và dọc theo đoạn tối ưu  $s \rightarrow \dots \rightarrow y \rightarrow x \rightarrow \dots \rightarrow n$  ta có dãy  $f$  không giảm. Suy ra

$$f(x) \leq g^*(n) + h(n).$$

Đặc biệt, vì  $h(n) \geq 0$  và  $g(n) > g^*(n)$ , ta được

$$f(x) < g(n) + h(n) = f(n).$$

Nhưng  $A^*$  luôn ưu tiên node có  $f$  nhỏ hơn. Vậy  $x$  phải được lấy ra để mở rộng *trước*  $n$ , mâu thuẫn với giả thiết  $n$  đang được mở rộng còn  $x$  chưa. Suy ra giả định  $g(n) > g^*(n)$  là sai. Vậy  $g(n) = g^*(n)$ .

**Định lý (Tối ưu của  $A^*$  Graph Search khi  $h$  consistent).** Cho  $h$  consistent. Khi  $A^*$  Graph Search lấy goal  $G$  ra để mở rộng, theo Bổ đề 2 ta có  $g(G) = g^*(G) = C^*$ . Do đó lời giải trả về là tối ưu.

$A^*$ Graph Search + consistent $h(n) \Rightarrow$ tối ưu (không cần reopen).
---

### Câu 3. Chỉ ra trường hợp $A^*$ sử dụng Graph Search không cho lời giải tối ưu khi $h(n)$ admissible

**Mục tiêu:** Xây dựng một phản-ví dụ trong đó heuristic  $h$  *admissible* (không vượt quá chi phí thật đến goal) nhưng *không consistent*, khiến  $A^*$  Graph Search (không “reopen” node đã đóng) trả về lời giải không tối ưu.

**Đồ thị phản-ví dụ.** Các đỉnh:  $S$  (start),  $A$ ,  $B$ ,  $G$  (goal). Chi phí cạnh dương:

$$S \rightarrow A : 3, \quad S \rightarrow B : 1, \quad B \rightarrow A : 1, \quad A \rightarrow G : 1, \quad B \rightarrow G : 100.$$

Heuristic (đặt  $h(G) = 0$  như thường lệ):

$$h(S) = 2, \quad h(A) = 0, \quad h(B) = 2, \quad h(G) = 0.$$

**Kiểm tra admissible.** Chi phí thật tối ưu từ mỗi node đến  $G$  (ký hiệu  $h$ ):

$$h(A)=1, \quad h(B)=2 \text{ (đi } B \rightarrow A \rightarrow G), \quad h(S)=3 \text{ (đi } S \rightarrow B \rightarrow A \rightarrow G).$$

Ta có  $h(A) = 0 \leq 1$ ,  $h(B) = 2 \leq 2$ ,  $h(S) = 2 \leq 3$ . Vậy  $h$  là **admissible**.

**Kiểm tra (không) consistent.** Điều kiện consistent yêu cầu  $h(B) \leq c(B, A) + h(A)$  trên cạnh  $B \rightarrow A$ :

$$h(B) = 2 \not\leq c(B, A) + h(A) = 1 + 0 = 1.$$

Vì  $2 \leq 1$  sai nên  $h$  **không consistent**.

**Quan sát lời giải tối ưu thật.** Đường đi tối ưu:  $S \rightarrow B \rightarrow A \rightarrow G$  với tổng chi phí

$$C=1+1+1=3.$$

**A\* Graph Search (không reopen) chạy như sau.** Khởi tạo:  $g(S) = 0$ ,  $f(S) = g(S) + h(S) = 2$ . Mở rộng  $S$  sinh  $A, B$ :

$$g(A) = 3, f(A) = 3 + 0 = 3; \quad g(B) = 1, f(B) = 1 + 2 = 3.$$

Hai node có  $f$  bằng nhau ( $f = 3$ ). Giả sử tie-break chọn  $A$  trước  $B$  (một lựa chọn hợp lệ, phổ biến).

- **Bước 1:** POP  $A$  (đưa  $A$  vào CLOSED). Sinh  $G$  với  $g(G) = 3 + 1 = 4$ ,  $f(G) = 4$ .
- **Bước 2:** POP  $B$  (đang có  $f(B) = 3 < f(G) = 4$ ). Từ  $B$  sinh lại  $A$  với  $g'(A) = g(B) + 1 = 2$ .
- **Graph Search không reopen:** Do  $A$  đã nằm trong CLOSED, nhánh tốt hơn tới  $A$  bị bỏ qua (không cập nhật  $g(A)$  từ 3 xuống 2).
- **Bước 3:** POP  $G$  với  $g(G) = 4$  và trả về lời giải có chi phí 4.

**Kết luận.** A\* Graph Search (không reopen) với heuristic *admissible nhưng không consistent* đã trả về **lời giải không tối ưu** ( $4 > C=3$ ). Nguyên nhân: do *không consistent*,  $f$  có thể giảm dọc đường đi; khi một node (ở đây là  $A$ ) bị đóng với  $g$ -cost chưa tối ưu, đường đi tốt hơn đến cùng node xuất hiện sau đó *không được xét* nếu thuật toán không cho phép “reopen”.

*Lưu ý.* Nếu:

- dùng heuristic **consistent**, hoặc
- vẫn dùng heuristic này nhưng **cho phép reopen** node khi tìm thấy đường đi tốt hơn,

thì A\* Graph Search sẽ khôi phục được tính tối ưu trong ví dụ này.

Admissible nhưng không consistent  $\nRightarrow$  A\* Graph Search tối ưu (nếu không reopen).

### Câu 4. Thuộc tính của giải thuật A\* Search

Bảng dưới đây tóm tắt các đặc tính quan trọng của giải thuật A\* dựa trên phần lý thuyết và các chứng minh ở các câu trước:

Thuộc tính	Ý nghĩa	Kết quả	Ghi chú
Complete?	Có luôn tìm được lời giải nếu tồn tại	<b>Có</b> (nếu chi phí cạnh $> 0$ và số node hữu hạn)	A* mở rộng theo thứ tự $f(n)$ tăng dần nên chắc chắn gặp goal tối ưu nếu tồn tại.
Time	Độ phức tạp thời gian	Thường <b>exponential</b> theo độ sâu $d$ (xấp xỉ $O(b^d)$ )	Phụ thuộc chất lượng heuristic: heuristic càng tốt thì thời gian càng giảm.
Space	Độ phức tạp bộ nhớ	<b>Exponential</b> $O(b^d)$	A* lưu toàn bộ OPEN và CLOSED để duy trì tính tối ưu.
Optimal?	Có đảm bảo lời giải tối ưu?	<b>Có</b> , nếu $h$ admissible (Tree) hoặc consistent (Graph)	Đây là điều đã được chứng minh trong Câu 1 và Câu 2.

A\* là giải thuật hoàn chỉnh và tối ưu, nhưng tốn thời gian và bộ nhớ.