BÀI TẬP LÝ THUYẾT TRÍ TUỆ NHÂN TẠO TUẦN 2

- Câu 1: Consistent Heuristic
- Câu 2: A-star SEARCH
  - 1. Complete
  - 2. Time
  - 3. Space
  - 4. Optimal
- Câu 3: Chứng minh mọi công thức trên logic mệnh đề đều có thể đưa về dạng chuẩn tắc.

Câu 1: Consistent Heuristic

Một heuristic được gọi là consistent nếu với mọi nút n và mọi nút kế thừa n' của n thông qua hành động a, bất đẳng thức sau luôn đúng: $h(n) \leq c(n, a, n') + h(n')$

Trong đó:

- $c(n, a, n')$ là chi phí để chuyển từ n tới n' thông qua hành động a.

Nếu $h(n)$ là consitent thì ta có
$f(n') = g(n') + h(n') = g(n) + c(n, a, n') + h(n') \geq g(n) + h(n) = f(n)$. Với $f(n)$ không giảm dọc theo đường đi bất kỳ.

Chứng minh rằng: Nếu heuristic $h(n)$ là nhất quán, thì thuật toán A^* sử dụng phương pháp Graph Search sẽ tìm được đường đi tối ưu đến mục tiêu.

Đầu tiên ta chứng minh rằng tính không giảm dọc theo đường đi bất kỳ của hàm $f(n)$ dọc theo bất kỳ đường đi nào. Với consistent heuristic ta có:

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n, a, n') + h(n') \\ &\geq g(n) + h(n) \quad (\text{do } h(n) \leq c(n, a, n') + h(n')) \\ &= f(n) \end{aligned}$$

Điều này có nghĩa là hàm $f(n)$ không giảm dọc theo bất kỳ đường đi nào.

Ở trong Graph Search ta duy trì một tập hợp nút đã được mở rộng (explored set) để tránh việc mở rộng lại các nút đã thăm trước đó. Với Heuristic consistent và tính không giảm của $f(n)$, ta có:

1. Khi một nút được mở rộng, nghĩa là đã tìm được đường đi ngắn nhất tới nó, bởi vì giả sử có một đường đi tốt hơn đến n sau khi n đã được mở rộng, vậy nó vi phạm $f(n)$ không giảm.

2. Mọi nút trong hàng đợi ưu tiên fringe có giá trị $f(n)$ lớn hơn hoặc bằng giá trị f của các nút đã được mở rộng. Vì $f'(n) \geq f(n)$ và chúng ta luôn mở rộng các nút mà có $f(n)$ nhỏ nhất, từ đó các nút trong fringe không có giá trị nhỏ hơn các nút đã được mở rộng.

Từ đó ta sẽ chứng minh thuật toán A^* tìm được đường đi tối ưu tới mục tiêu.

Giả sử thuật toán A^* không tìm được đường tối ưu tới mục tiêu khi sử dụng heuristic consitstent.

Gọi S_{opt} là đường đi tối ưu tới mục tiêu với chi phí là C^*. Giả sử rằng A^* đã tìm được một đường đi S' với chi phí C' sao cho $C' > C^*$. Tại thời điểm thuật toán kết thúc, mục tiêu đã được mở rộng thông qua S'. Xét một nút trên đường đi tối ưu S_{opt} mà chưa được mở rộng tại thời điểm này, do heuristic consistent và $f(n)$ không giảm, giá trị $f(n)$ của nút trên S_{opt} sẽ không lớn hơn C^*. Nhưng bởi vì $C' > C^*$, nên giá trị của nút trong fringe nhỏ hơn giá trị của các nút đã được mở rộng. Điều này mâu thuẫn với việc A^* luôn mở rộng các nút có $f(n)$ nhỏ nhất.

Vậy do đó, A^* Graph Search với Heuristic Consistent và $f(n)$ không giảm sẽ tìm được đường đi tối ưu tới mục tiêu.

Câu 2: A-star SEARCH

1. Complete

Thuật toán A^* Search là **hoàn chỉnh** khi sử dụng một hàm heuristic mà admissible và consistent, và tất cả các bước đi có chi phí tối thiểu dương (tức là không có bước đi nào có chi phí bằng 0 hoặc âm). Trong đó:

- Heuristic admissible: thỏa mãn $h(n) \leq h^*(n)$ với $h^*(n)$ là chi phí thực sự từ n tới đích.
- Heuristic consistent: thỏa mãn $h(n) \leq c(n, a, n') + h(n')$ với $c(n, a, n')$ là chi phí từ n đến n' qua hành động a.

2. Time

Độ phức tạp thời gian của A^* Search trong trường hợp xấu nhất là **exponential** theo độ sâu của giải pháp d, hay là $\mathcal{O}(b^d)$. Có thể là do số lượng nút có thể mở rộng trong vùng "goal contour" là vùng mà $f(n) \leq C^*$, với C^* là chi phí của giải pháp tối ưu.

Nó phụ thuộc rất nhiều vào chất lượng của hàm heuristic $h(n)$. Nếu hàm heuristic hoàn hảo $(h(n) = h^*(n))$, A^* sẽ tìm ra giải pháp trong thời gian tuyến tính. Nếu heuristic không thông tin $h(n) = 0$, A^* sẽ trở thành Uniform-Cost Search với độ phức tạp là exponential.

Sai số của Heuristic cũng ảnh hưởng tới độ phức tạp:

- Sai số tuyệt đối (Δ): $\Delta = h^*(n) - h(n)$. Độ phức tạp là $\mathcal{O}(b^{\Delta})$
- Sai số tương đối (ε): $\varepsilon = (h^*(n) - h(n))/h^*(n)$. Độ phức tạp là $\mathcal{O}(b^{\varepsilon d})$

3. Space

A^* Search yêu cầu lưu trữ tất cả các nút đã sinh ra trong bộ nhớ để đảm bảo tính tối ưu, do đó độ phức tạp không gian cũng là **exponential**

4. Optimal

Thuật toán A^* là **optimal** khi sử dụng một hàm heuristic admissible. Vì $h(n)$ không bao giờ đánh giá quá cao chi phí thực sự, A^* sẽ không bỏ qua bất kỳ giải pháp tối ưu nào. Trong phiên bản Graph Search, để đảm báo tính tối ưu, $h(n)$ cần phải consistent. Điều này giúp tránh việc đánh giá lại các nút và đảm bảo rằng khi một nút đã được mở rộng, đường đi tới nút đó là tối ưu.

Câu 3: Chứng minh mọi công thức trên logic mệnh đề đều có thể đưa về dạng chuẩn tắc.

Một công thức được gọi là chuẩn tắc nếu nó là hội của các câu phức tạp (clause). Một câu phức tạp có dạng $A_1 \vee A_2 \vee \cdots \vee A_n$, trong đó A_i là các câu đơn (literal).

Để chứng minh điều này, chúng ta sẽ sử dụng một quy trình biến đổi tuần tự các công thức logic mệnh đề về dạng chuẩn tắc. Quy trình này bao gồm các bước sau:

Bước 1: Loại bỏ kéo theo $(\rightarrow)$ và tương đương $(\leftrightarrow)$

- Loại bỏ kéo theo: Các biểu thức kéo theo có dạng $P \rightarrow Q$ có thể được biến đổi thành $\neg P \vee Q$.
- Loại bỏ tương đương: Các biểu thức tương đương có dạng $P \leftrightarrow Q$ có thể được biến đổi thành $(P \rightarrow Q) \wedge (Q \rightarrow P)$ sau đó áp dụng quy tắc với phép kéo theo để biến đổi tiếp.

Bước 2: Đưa phủ định vào trong hoặc loại bỏ phủ định kép bằng định luật De Morgan. Theo định luật De Morgan:

- $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
- $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$

Phủ định kép: $\neg\neg P \equiv P$

Bước 3: Sử dụng các luật phân phối để đưa công thức về dạng chuẩn:
$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$. Áp dụng luật sao cho mọi $\vee$ nằm bên trong và mọi $\wedge$ nằm bên ngoài.

Như vậy chỉ cần áp dụng quy trình này thì mọi công thức trên logic mệnh đề đều có thể đưa về dạng chuẩn tắc.