TRÍ TUỆ NHÂN TẠO

Bài 5: Tìm kiếm có định hướng

Nội dung

1. Tìm kiếm mù vs Tìm kiếm có định hướng
2. Tìm kiếm theo tốt nhất (best-first search)
3. Tìm kiếm tham lam (greedy best-first search)
4. Thuật toán A*

Phần 1

Tìm kiếm mù vs Tìm kiếm có định hướng

Tìm kiếm mù vs Tìm kiếm có định hướng

- Nhắc lại hoạt động của thuật toán tìm kiếm mù:
  - Duy trì một tập các trạng thái đang xem xét (tập S), ban đầu chỉ chứa điểm xuất phát
  - Chọn ngẫu nhiên trạng thái N trong S, mở rộng tập S kết nạp các trạng thái con của N
  - Dừng nếu đến điểm đích (goal) hoặc hết trạng thái xem xét
- Hoạt động của tìm kiếm mù bản chất là mở rộng không gian tìm kiếm ngày càng lan xa dần điểm xuất phát, nếu không gian tìm kiếm là hữu hạn, thuật toán sẽ đến đích vào một thời điểm nào đó hoặc xét hết các trạng thái
- **Vấn đề**: không gian trạng thái mở rộng một cách ngẫu nhiên, bùng nổ số trạng thái

Tìm kiếm mù vs Tìm kiếm có định hướng

- Làm thế nào để tránh việc mở rộng một cách ngẫu nhiên: sử dụng thông tin bổ sung trong quá trình mở rộng
- Thay vì mở rộng ngẫu nhiên, có thể xây dựng cơ chế nào đó ưu tiên các trạng thái (mà theo kinh nghiệm thì) có nhiều cơ hội đến đích hơn
- Bài toán di chuyển trên bản đồ thì nên ưu tiên theo hướng đi: Đi từ Hà Nội vào Đà Nẵng thì tới Nam Định tốt hơn là tới Yên Bái hoặc Thái Nguyên

Tìm kiếm mù vs Tìm kiếm có định hướng

- Trong tình huống khác, ta có thể ưu tiên các dấu hiệu: đi biển nếu đi càng xa bờ thì nước càng xanh thẫm hơn
- Chơi cờ vua: nên chọn nước ăn quân hậu hơn là nước ăn quân tốt hoặc quân tượng
- Chơi trò 8-mảnh: ưu tiên những trạng thái có nhiều mảnh đặt đúng vị trí cuối cùng của nó
- Tất cả những cơ chế này đều có tính kinh nghiệm, không phải lúc nào cũng đúng, thậm chí nhiều lúc còn là lựa chọn tệ nhất

Phần 2

Tìm kiếm theo tốt nhất (best-first search)

Tìm kiếm theo tốt nhất

- Thay vì lấy ngẫu nhiên một trạng thái khỏi tập quan sát S, ta chọn trạng thái mà ta đánh giá là tốt nhất
  - Xây dựng hàm f(node) để đánh giá cơ hội của trạng thái node
  - Hàm f(node) có thể có nhiều phương pháp khác nhau tùy vào từng loại bài toán
  - Quy định f(node) càng nhỏ thì cơ hội của node càng cao (đây chỉ là quy ước, có thể quy định ngược lại mà không làm mất tính tổng quát của lời giải)
  - Xây dựng lại cấu trúc của S: làm việc theo nguyên tắc lấy ra phần tử có f tương ứng là nhỏ nhất (có thể sử dụng một cấu trúc heap nào đó thích hợp)

Tìm kiếm theo tốt nhất

```
function BEST-FIRST-SEARCH(START)
                                    return solution/failure {
  S = {START}
  loop {
    if S is EMPTY then return failure
    node = REMOVE-BEST-ONE(S)
    if node is GOAL then return SOLUTION(node)
    S = S + EXPAND(node)
  }
}
```

S: ***Tập các hình trạng đang được xem xét***

REMOVE-BEST-ONE(S): ***Lấy phần tử có f nhỏ nhất khỏi tập S***

EXPAND(node): ***Tập hình trạng liên quan đến node***

Phần 3

Tìm kiếm tham lam (greedy best-first search)

Tìm kiếm tham lam

- Tìm kiếm tham lam là một trường hợp cụ thể của chiến lược tìm kiếm theo tốt nhất
- Hàm f(node) được xây dựng dựa trên ước lượng từ trạng thái node đến trạng thái goal, ví dụ:
  - Với bài toán di chuyển trên bản đồ: f(node) có thể là độ dài đường chim bay từ node đến goal
  - Với bài toán 8-mảnh: f(node) là số mảnh bị lệch khỏi vị trí chuẩn trong trạng thái goal
  - Với bài toán chơi cờ: ăn quân nào càng có giá trị càng tốt, chấp nhận thiệt quân càng nhỏ càng tốt
- *Nhận xét: thuật toán tìm kiếm tham lam sẽ ưu tiên những trạng thái có vẻ gần đích hơn*

Phần 4

Thuật toán A*

Thuật toán A*

- Thuật toán A* là một dạng tìm kiếm theo tốt nhất
- Hàm f(node) = g(node) + h(node), trong đó:
  - Hàm g(node) là chi phí để đi đến trạng thái node
  - Hàm h(node) là chi phí ước lượng để đi từ node đến goal
  - Như vậy f(node) là ước lượng chi phí tổng thể để từ trạng thái start đến trạng thái goal nhưng đi qua trạng thái node
- Trong tình huống của A*, giá trị hàm g(node) xác định, giá trị hàm h(node) là ước đoán, có thể sử dụng các kĩ thuật tương tự như thuật toán tìm kiếm tham lam
- *Nhận xét: thuật toán A* ưu tiên những trạng thái có chi phí ước lượng tổng thể là thấp nhất*

Thuật toán A*

- u_0 = đỉnh xuất phát.
- goal = đỉnh kết thúc.
- close = tập các đỉnh đã được tính toán chính xác đường đi ngắn nhất
- open = tập các đỉnh còn lại
- a[i,j] = trọng số của cung (i,j)
- g[i] = khoảng cách min từ u_0 đến i
- h[i] = khoảng cách min ước lượng từ i đến goal

Thuật toán A*

```
g[i] = +∝ ∀ i ∈ [1..n]
close = [u0]
open = [1..n] - [u0]
k = u0
while (goal ∈ open) {
    // sửa đổi ước lượng nhỏ nhất
    ∀ i ∈ open
        g[i] = min {g[i], g[k]+a[k,i]}
    // chọn trạng thái mở rộng
    chọn k ∈ open để ∀ i ∈ open
        có (g[k]+h[k]) ≤ (g[i]+h[i])
    open = open - [k]
    close = close + [k]
}
```

Thuật toán A*: ước lượng chấp nhận được

- Ước lượng h(node) gọi là ước lượng chấp nhận được nếu với mọi node thì $0 \leq h(node) \leq h^*(node)$
  - h*(node) là chi phí tối ưu thực tế để đi từ node đến goal
- Thuật toán A* được chứng minh là tìm được kết quả tối ưu nếu có thể xây dựng được ước lượng h(node) chấp nhận được
- Có nhiều cách xây dựng h(node), và phương pháp nào càng gần h*(node) thì càng hữu ích
  - Bài toán 8-mảnh, h1(node): số mảnh lệch với goal
  - Bài toán 8-mảnh, h2(node): tổng số bước dịch chuyển của từng mảnh về vị trí tương ứng trên goal
  - Độ sâu 12: A*h1 hơn 200 trạng thái, A*h2 dưới 100 trạng thái

Ví dụ so sánh

So sánh thuật toán tìm kiếm mù, tìm kiếm tham lam và A* trong bài toán di chuyển trên mê cung:

- Di chuyển từ ô đỏ (3, 5) đến ô xanh (0, 7)
- Có thể di chuyển đến một trong 8 ô xung quanh (chung cạnh/đỉnh)
- Hàm ước lượng khoảng cách: tối đa độ lệch dòng hoặc cột
  $h(n) = \max\{|r_n - r_{goal}|, |c_n - c_{goal}|\}$