State space size in search problems

A black and white picture of a chess board and a checkerboard

AI-generated content may be incorrect.

Trong các bài toán tìm kiếm, **state space** được hiểu là tập hợp tất cả các trạng thái khả dĩ mà hệ thống có thể rơi vào trong quá trình tìm lời giải. Kích thước không gian trạng thái (**state space size**) phản ánh độ khó của bài toán: càng nhiều trạng thái thì tìm kiếm càng phức tạp. Dưới đây là phân tích một số ví dụ minh họa.

* Maze (Bài toán mê cung)

**Trạng thái**: vị trí hiện tại của tác tử (agent) trong mê cung. **Kích thước không gian trạng thái**: phụ thuộc vào số ô của mê cung.

Với mê cung kích thước m×n, số trạng thái tối đa là m×n.

Đặc điểm: tăng tuyến tính theo kích thước bản đồ, nhưng độ khó đến từ số lượng đường đi khả dĩ.

* 8-Queens Problem (Bài toán 8 quân hậu)

**Trạng thái**: cách sắp xếp 8 quân hậu trên bàn cờ 8×8

Kích thước không gian trạng thái≈4.4×10^9 (chọn 8 vị trí trong số 64 ô để đặt quân hậu).

Nhưng thường ta ràng buộc: mỗi hàng chỉ đặt 1 hậu (để tránh việc hai hậu đứng cùng hàng), khi đó ở mỗi hàng có 8 cột để chọn → số trạng thái là:8^8=16,777,216

Mặc dù nhỏ hơn nhiều so với không gian trạng thái của cờ vua, đây vẫn là một bài toán có độ phức tạp đáng kể và đòi hỏi chiến lược tìm kiếm hoặc heuristic.

* 8-Puzzle Problem (Bài toán 8-puzzle)

**Trạng thái**: cách sắp xếp 8 số và 1 ô trống trên bảng 3×3

**Số trạng thái có thể**: 9!= 362,880

**Số trạng thái hợp lệ (giải được)**: chỉ một nửa số cấu hình có thể giải được, tức là:

9!/2 = 181,440. Vì các phép di chuyển hợp lệ của trò này chỉ sinh ra một nhóm hoán vị **nhỏ hơn** toàn bộ các hoán vị của 9 phần tử (8 miếng + ô trống). Cụ thể, các phép di chuyển cho phép ta thực hiện các **3-cycle** (chu kỳ 3 phần tử) trên các viên số (bằng cách di chuyển ô trống quanh 2×2), và một **3-cycle** là tích của **hai transposition** ⇒ nó là một hoán vị **chẵn**. Tập các hoán vị sinh ra bởi các 3-cycle chính là **nhóm chẵn (alternating group)**, có kích thước bằng **1/2** của toàn bộ nhóm hoán vị. Vì vậy chỉ có **một nửa** cấu hình (hoặc hoán vị) có thể đạt được bằng các phép di chuyển hợp lệ.

Đây là bài toán kinh điển dùng để minh họa tìm kiếm có heuristic (A\*), độ khó đến từ số bước di chuyển lớn dù không gian trạng thái không quá lớn.

* Tic-tac-toe

Trạng thái là tình huống bàn cờ với 9 ô, mỗi ô có thể là X, O hoặc trống;

Không gian trạng thái thô là: 3^9= 19,683

Nhưng trạng thái hợp lệ chỉ khoảng 5,478 do ràng buộc luật chơi

Đây là một trò chơi có không gian nhỏ, có thể giải exhaustively, dẫn đến kết quả cân bằng (perfect play = hoà)

Search Complexity Analysis

A diagram of a state

AI-generated content may be incorrect.

Bài toán yêu cầu phân tích độ phức tạp tìm kiếm (search complexity) trong một không gian trạng thái (state space) có mô hình chuyển trạng thái như hình minh họa. Các tham số quan trọng cần xác định gồm:

**b**: hệ số phân nhánh tối đa (branching factor).

**m**: số hành động tối đa trên bất kỳ đường đi nào (không có lặp lại).

**d**: độ sâu của lời giải tối ưu (optimal solution depth).

Các giá trị này giúp ta ước lượng chi phí thời gian và bộ nhớ khi áp dụng các thuật toán tìm kiếm (BFS, DFS, UCS, v.v.).

**2. Xác định tham số**

**a. Hệ số phân nhánh tối đa (b)**

Hệ số phân nhánh là số hành động tối đa có thể thực hiện tại một trạng thái.

Trong sơ đồ, mỗi trạng thái có 3 hành động khả dĩ: **L** (Left), **R** (Right), và **S** (Stay/Skip).

Do đó: b=3

**b. Số hành động tối đa trong một đường đi không lặp (m)**

Đây là độ dài lớn nhất của một chuỗi hành động hợp lệ để đi từ trạng thái gốc đến trạng thái bất kỳ (không quay vòng).

Quan sát hình, từ trạng thái khởi đầu đến một trạng thái lá sâu nhất có tối đa 4 bước.

Do đó: m=4

**c. Độ sâu của lời giải tối ưu (d)**

Độ sâu (depth) là số bước di chuyển ngắn nhất từ trạng thái khởi đầu đến trạng thái đích (goal).

Trong mô hình, trạng thái mục tiêu (Goal state) nằm ở tầng thứ 3 (sau 3 hành động từ Initial state).

Do đó:d=3

**3. Phân tích độ phức tạp tìm kiếm**

Với các tham số b=3,m=4,d=3, ta có:

* **Độ phức tạp thời gian (time complexity)**:

Với Breadth-First Search (BFS), số trạng thái tối đa được mở rộng là:

O(b^d)=O(3^3)=O(27)

**Độ phức tạp không gian (space complexity)**:

BFS cần lưu toàn bộ các node ở một tầng → cũng là O(b^d)=O(27)

DFS chỉ cần lưu đường đi → O(bm)=O(3×4)=O(12)

Heuristics from Relaxed Problems

A maze with a maze and a maze with a maze and a maze with a maze and a maze with a maze and a maze with a maze and a maze with a maze and a maze with

AI-generated content may be incorrect.

Trong tìm kiếm có heuristic (ví dụ như **A\***), việc xây dựng hàm heuristic tốt đóng vai trò then chốt trong hiệu quả tìm kiếm. Một cách phổ biến để xây dựng heuristic là **relaxation (nới lỏng)** bài toán gốc, tức là bỏ qua một số ràng buộc để có thể ước lượng chi phí một cách đơn giản hơn.

Trong ví dụ này, ta phân tích hai loại heuristic thường gặp trong bài toán mê cung:

**Euclidean distance (khoảng cách Euclid)**

**Manhattan distance (khoảng cách Manhattan)**

1. Heuristic Euclidean Distance

**Định nghĩa**:

heuclidean(n) = sqrt(pow(xcur-xgoal, 2)+ pow(ycur-ygoal, 2))

**Ý nghĩa**:  
Đây là **khoảng cách ngắn nhất có thể có** nếu agent được phép đi thẳng tới đích theo đường chéo, không gặp bất kỳ rào cản nào.

* **Ưu điểm**:

Rất **tự nhiên** và sát nghĩa với "khoảng cách thực tế" trong không gian 2D.

Phù hợp cho môi trường liên tục hoặc khi di chuyển đường chéo được cho phép.

* **Nhược điểm**:

Trong **mê cung dạng grid chỉ cho 4 hướng**, Euclidean thường **quá lạc quan**. Ví dụ: nếu goal ở ngay góc chéo nhưng có tường chắn, chi phí thực có thể gấp đôi hoặc gấp ba Euclidean.

Điều này dẫn đến heuristic ít thông tin, khiến A\* mở rộng nhiều nút dư thừa.

**Ví dụ minh họa**:  
Giả sử agent ở (0,0) và goal ở (3,4). Euclidean: sqrt(3^2+4^2) ​= 5.

* Trong mê cung 4 hướng, chi phí thực **tối thiểu** là 7 bước (3 ngang + 4 dọc).  
  → Heuristic Euclidean đánh giá thấp (5 thay vì 7).

1. Heuristic Manhattan Distance

**Định nghĩa**:

hmanhattan(n)=∣xcur−xgoal∣+∣ycur−ygoal∣

**Ý nghĩa**: Khoảng cách ngắn nhất trong một **grid chỉ cho phép đi ngang hoặc dọc** (giống cách đi trong một thành phố Manhattan – lên phố, rẽ trái, rẽ phải).

* **Ưu điểm**:

**Sát thực tế** hơn Euclidean khi chỉ có 4 hướng di chuyển.

Chấp nhận được vì không bao giờ vượt quá chi phí thật.

Giúp A\* thu hẹp đáng kể không gian tìm kiếm.

* **Nhược điểm**:

Nếu cho phép di chuyển chéo hoặc trong không gian liên tục, Manhattan lại trở nên **bảo thủ quá mức**, đánh giá chi phí cao hơn cần thiết.

**Ví dụ minh họa**:  
Với agent ở (0,0) và goal ở (3,4) : Manhattan: ∣3∣+∣4∣=7.

* Chi phí thực khi chỉ đi ngang/dọc đúng là 7.  
  → Heuristic Manhattan chính xác hơn Euclidean trong trường hợp này.