Trí tuệ nhân tạo (Artificial Intelligence)

Giải quyết bài toán bằng tìm kiếm

By Hoàng Hữu Việt
Email: viethh@vinhuni.edu.vn
Viện Kỹ thuật và Công nghệ, Đại học Vinh

Vinh, 3/2019

Tài liệu

- Tài liệu chính

  [1] Stuart Russell, Peter Norvig. Artificial Intelligence. A modern approach. 3rd ed. Prentice Hall, 2009.

- Tài liệu khác

  [2] Milos Hauskrecht. Artificial Intelligence, 2013. people.cs.pitt.edu/~milos/courses/cs1571-Fall2013.

  [3] Các tài liệu từ internet

Nội dung

- Tác tử giải quyết bài toán
- Các bài toán ví dụ
- Tìm kiếm nghiệm
- Bài tập

Tác tử giải quyết bài toán

- Giả thiết:
  - Môi trường có thể quan sát được (observable), do vậy tác tử luôn biết trạng thái hiện thời.
  - Môi trường là rời rạc (discrete), do vậy ở bất kỳ trạng thái nào, chỉ có một số hành động hữu hạn để chọn.
  - Môi trương là được biết (known), do vậy tác tử biết trạng thái nào đạt được sau mỗi hành động.
  - Môi trường là xác định (deterministic), do vậy mỗi hành động có chính xác một kết quả.
- Nghiệm: một chuỗi cố định các hành động.
- Tìm kiếm: quá trình xác định một chuỗi các hành động để đi đến mục tiêu.

Tác tử giải quyết bài toán

- Một thiết kế đơn giản "*formulate, search, execute*" cho tác tử:
  - Xác định mục tiêu bài toán và phát biểu bài toán
  - Tìm kiếm một chuỗi các hành động để giải quyết vấn đề.
  - Thực hiện từng hành động một.
- Chương này tập trung mô tả quy trình phát biểu/định nghĩa bài toán và các thuật toán tìm kiếm.

Tác tử giải quyết bài toán

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  persistent: seq, an action sequence, initially empty
              state, some description of the current world state
              goal, a goal, initially null
              problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
      if seq = failure then return a null action
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

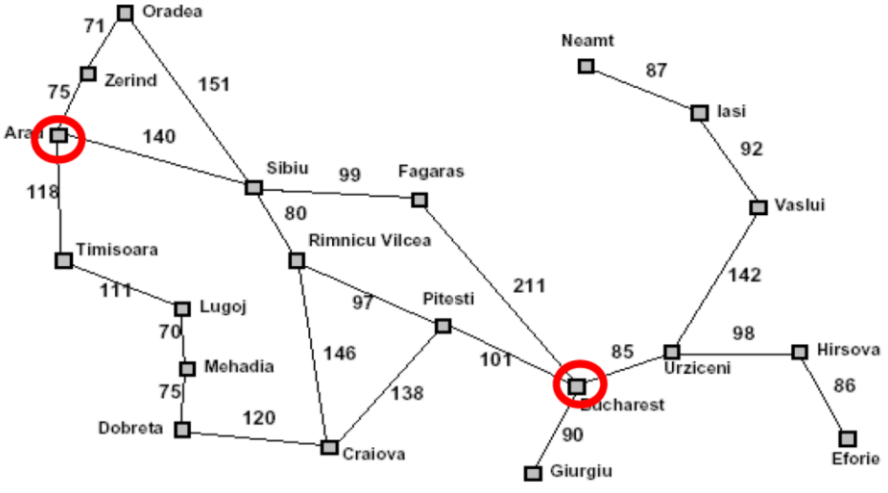
Figure 3.1 A simple problem-solving agent. It first formulates a goal and a problem, searches for a sequence of actions that would solve the problem, and then executes the actions one at a time. When this is complete, it formulates another goal and starts over.

Định nghĩa bài toán

- Một bài toán tìm kiếm có thể được định nghĩa bởi 5 thành phần:
  - Trạng thái khởi đầu (*initial state*)
  - Các hành động (*actions*)
    - Cho một trạng thái *s*, *ACTIONS*(*s*) trả về tập hành động mà được thực hiện ở trạng thái *s*.
  - Mô hình chuyển trạng thái (*transition model*)
    - Ví dụ *RESULT*(*s*,*a*) trả về trạng thái khi tác tử ở trạng thái *s* thực hiện hành động *a*.
  - Kiểm tra trạng thái đích (*goal test*)
    - Kiểm tra một trạng thái có phải là trạng thái đích hay không?
  - Chi phí đường đi (*path cost*)

Định nghĩa bài toán

- Bài toán tìm đường đi từ *Arad* đến *Bucharest*



Định nghĩa bài toán

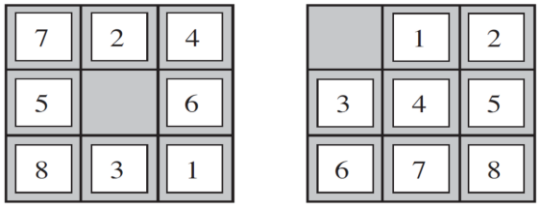
- Bài toán tìm đường đi từ *Arad* đến *Bucharest*
  - Trạng thái khởi đầu: *Arad*.
  - Các hành động (*actions*): từ một thành phố chỉ đi đến các thành phố lân cận
    - Ví dụ: *ACTIONS*(*Arad*) = {*Go*(*Sibiu*), *Go*(*Timisoara*), *Go*(*Zerind*)}
  - Mô hình chuyển trạng thái
    - Ví dụ: *RESULT*(*In*(*Arad*),*Go*(*Zerind*)) = *In*(*Zerind*).
  - Kiểm tra trạng thái đích: trạng thái {*In*(*Bucharest*)}.
  - Chi phí đường đi: chiều dài đường đi từ *Arad* đến *Bucharest*.

Các loại bài toán

- Các bài toán trò chơi (*toy problems*)
  - Nhằm mục đích minh họa hoặc luyện tập các phương pháp giải quyết bài toán khác nhau.
  - Thường là các mô tả ngắn gọn, chính xác và do đó các nhà nghiên cứu có thể sử dụng để so sánh hiệu quả của các thuật toán.
- Các bài toán thực tế (*real-world problems*)
  - Một bài toán thực tế là một bài toán trong thế giới thực cần giải quyết.
  - Thường không có mô tả duy nhất, nhưng có thể đưa được ra các đặc tính tổng quát.

Bài toán trò chơi

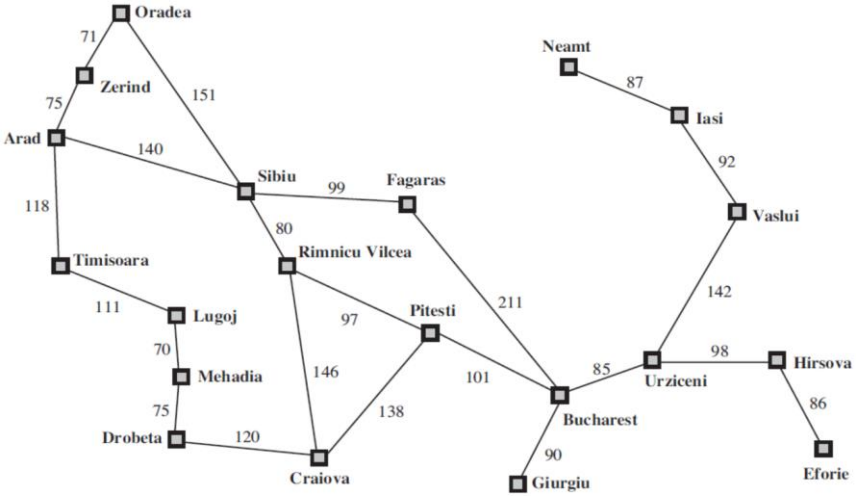
- Ví dụ bài toán 8 số (8-puzzle)
  - Các trạng thái (states): một trạng thái là vị trí của mỗi của 8 ô (tile) và vị trí ô trắng (có 9!/2 trạng thái, 15-puzzle có 1,3 triệu trạng thái).
  - Trạng thái khởi đầu (initial state): có thể tại mọi trạng thái.
  - Tập hành động (actions): chuyển ô trắng sang trái, phải, lên, xuống.
  - Mô hình chuyển trạng thái (transition model): trả về trạng thái của một trạng thái và một hành động.
  - Kiểm tra đích (goal test): các ô nằm đúng trạng thái đích.
  - Chi phí đường đi (path cost): tổng số bước di chuyển, mỗi bước có chi phí là 1.



Start State Goal State

Bài toán thực tế

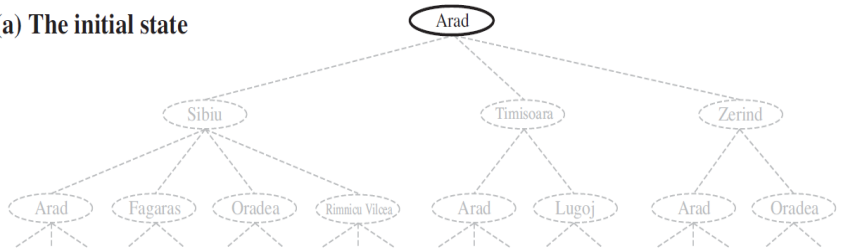
- Ví dụ bài toán người du lịch - traveling salesperson problem (TSP)



Tìm kiếm nghiệm

- Một nghiệm là một chuỗi các hành động
  - Các thuật toán tìm kiếm làm việc bằng cách xem xét các chuỗi hành động.
  - Biểu diễn theo cấu trúc cây tìm kiếm (search tree):
    - Gốc của cây tìm kiếm là trạng thái khởi đầu.
    - Các nhánh là các hành động.
    - Các nút tương ứng với các trạng thái trong không gian trạng thái.

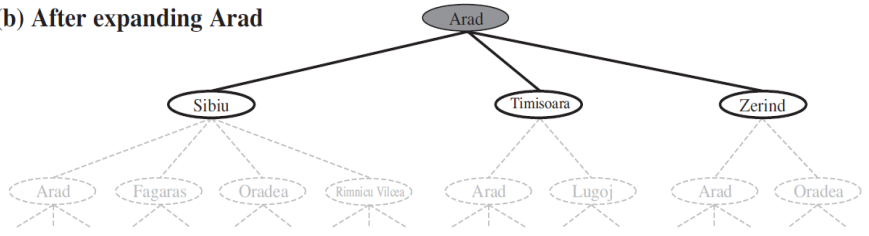
(a) The initial state



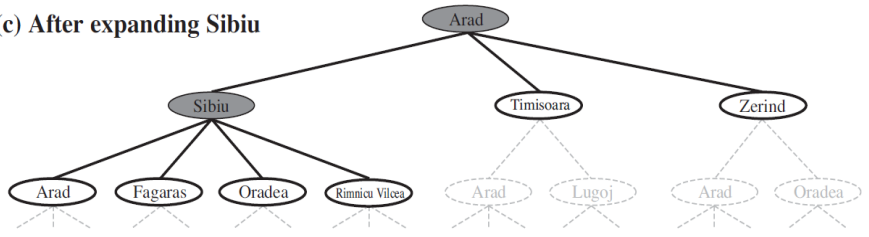
Tìm kiếm nghiệm

- Biểu diễn theo cấu trúc cây tìm kiếm (search tree):

(b) After expanding Arad



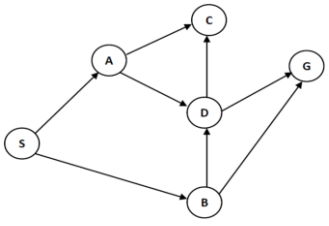
(c) After expanding Sibiu



Thuật toán tìm kiếm tổng quát

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

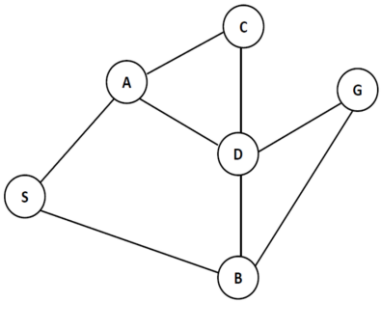
- Ví dụ tìm đường đi từ đỉnh S → G
  - Trường hợp 1 (FIFO):
    - Lấy nút đầu tiên của *frontier*
    - Bổ sung các nút mở rộng vào cuối *frontier*
  - Trường hợp 2 (LIFO):
    - Lấy nút đầu tiên của *frontier*
    - Bổ sung các nút mở rộng vào đầu *frontier*



Thuật toán tìm kiếm tổng quát

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

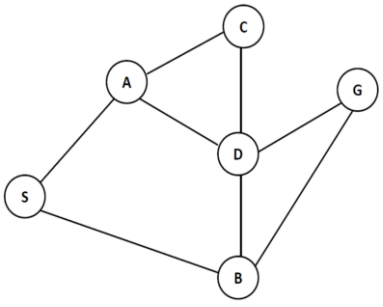
- Tìm đường đi từ đỉnh S → G



Thuật toán tìm kiếm tổng quát

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

- Tìm đường đi từ đỉnh S → G
  - Thuật toán tạo ra các đường lặp, tức là tìm kiếm lặp vô hạn.
  - Để tránh các đường lặp, cần nhớ các nút đã mở rộng.
  - Đưa thêm một cấu trúc dữ liệu để lưu các nút đã mở rộng.



Thuật toán tìm kiếm tổng quát

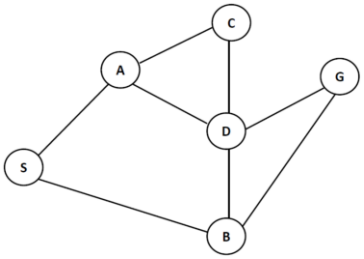
function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

- Các dòng nghiêng trong GRAPH-SEARCH được thêm vào so với TREE-SEARCH
- Các thuật toán tiếp theo được phát triển dựa trên thuật toán này!

Thuật toán tìm kiếm tổng quát

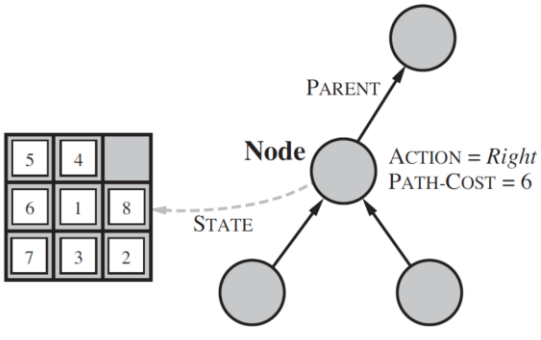
function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
initialize the frontier using the initial state of *problem*
initialize the explored set to be empty
loop do
if the frontier is empty **then return** failure
choose a leaf node and remove it from the frontier
if the node contains a goal state **then return** the corresponding solution
add the node to the explored set
expand the chosen node, adding the resulting nodes to the frontier
only if not in the frontier or explored set

- Tìm đường đi từ S→G:
  (1) frontier là FIFO
  (2) frontier là LIFO



Cấu trúc dữ liệu cài đặt thuật toán

- Yêu cầu một cấu trúc dữ liệu để lưu vết cây tìm kiếm.
- Với mỗi nút *n* của cây, cần cấu trúc với 4 thành phần:
  - *n.STATE*: trạng thái trong không gian trạng thái ứng với nút *n*.
  - *n.PARENT*: nút trong cây tìm kiếm sinh ra nút *n*.
  - *n.ACTION*: hành động đã được thực hiện bởi nút cha sinh ra nút *n*.
  - *n.PATH – COST*: chi phí, ký hiệu *g*(*n*), của đường từ trạng thái ban đầu đến nút *n*.



Các hàm cài đặt của thuật toán

- Cấu trúc lưu trữ dữ liệu lưu trữ "*frontier*" là một "*queue*"
- Các thao tác trên "*queue*":
  - *EMPTY?(queue):* trả về true nếu không có phần tử trong queue.
  - *POP(queue):* lấy phần tử đầu tiên của queue và xóa nó trong queue.
  - *INSERT(element,queue):* chèn một phần tử element vào queue và trả về queue.
- Cấu trúc queue được đặc trưng bởi thứ tự lưu trữ các nút được chèn vào → CHIẾN LƯỢC TÌM KIẾM:
  - FIFO queue - lấy phần tử lâu nhất của queue.
  - LIFO queue - lấy phần tử mới nhất.
  - PRIORITY queue - lấy phần tử ưu tiên cao nhất theo một hàm sắp xếp nào đó.

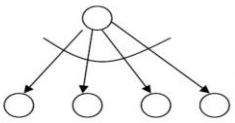
Đánh giá hiệu quả của thuật toán

- Hoàn chỉnh (completeness)
  - Phương pháp có tìm ra được nghiệm không nếu nghiệm tồn tại?
- Tối ưu (optimality)
  - Nghiệm tìm được của phương pháp tìm kiếm có phải là nghiệm tối ưu không?
- Độ phức tạp tính toán thời gian (time complexity)
  - Thời gian để tìm ra nghiệm?
- Độ phức tạp tính toán không gian (space complexity)
  - Bộ nhớ cần dùng để tìm ra nghiệm?

Các tham số đo độ phức tạp tính toán

- Độ phức tạp tính toán được xác định theo 3 tham số của cây tìm kiếm:
  - *b*: số nhánh tối đa (maximum branching factor)
  - *d*: độ sâu tìm được nghiệm
  - *m*: độ sâu tối đa của không gian trạng thái
- Thời gian được tính theo dạng số nút được sinh ra trong quá trình tìm kiếm.
- Không gian được tính theo dạng số lớn nhất của các nút được lưu trữ trong bộ nhớ.

Branching factor



The number of applicable operators

Bài tập

- Sử dụng thuật toán GRAPH-SEARCH tìm đường đi cho một robot trên lưới sau:
  - Điểm bắt đầu *S* và điểm đích *G*
  - Mỗi vị trí robot có thể đi đến các ô lân cận theo 4 hướng: trái, phải, lên, xuống nhưng không đi được vào các ô màu xám (vật cản).