Day 5
Searching algorithms for problem solvings

Lecturer: Msc. Minh Tan Le

Contents

I. Goal-based agents & Utility-based agents

II. Environment abstraction

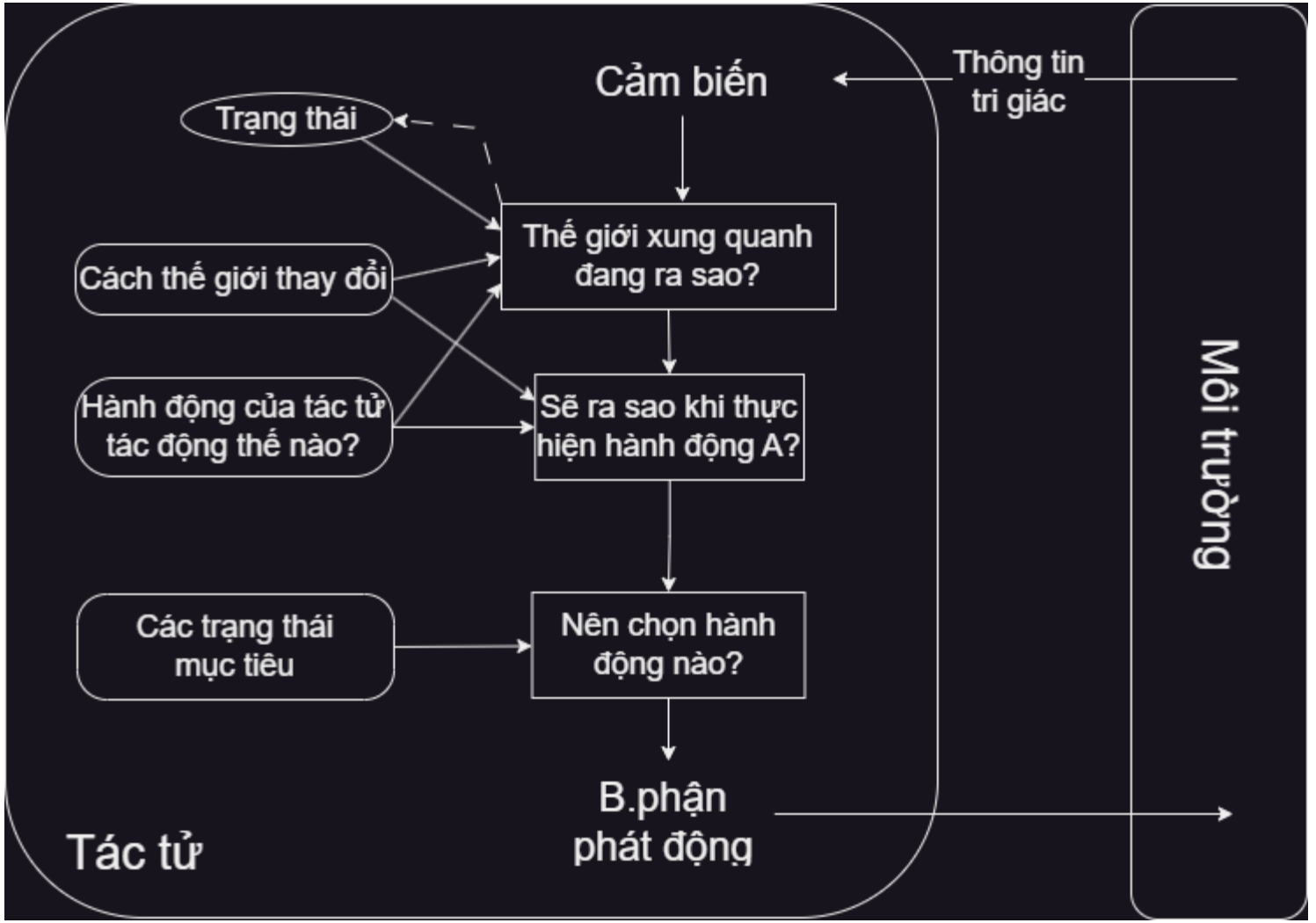
III. Searching for goals

    1. Best first search (BFS)

    2. A* search

Goal-based agent

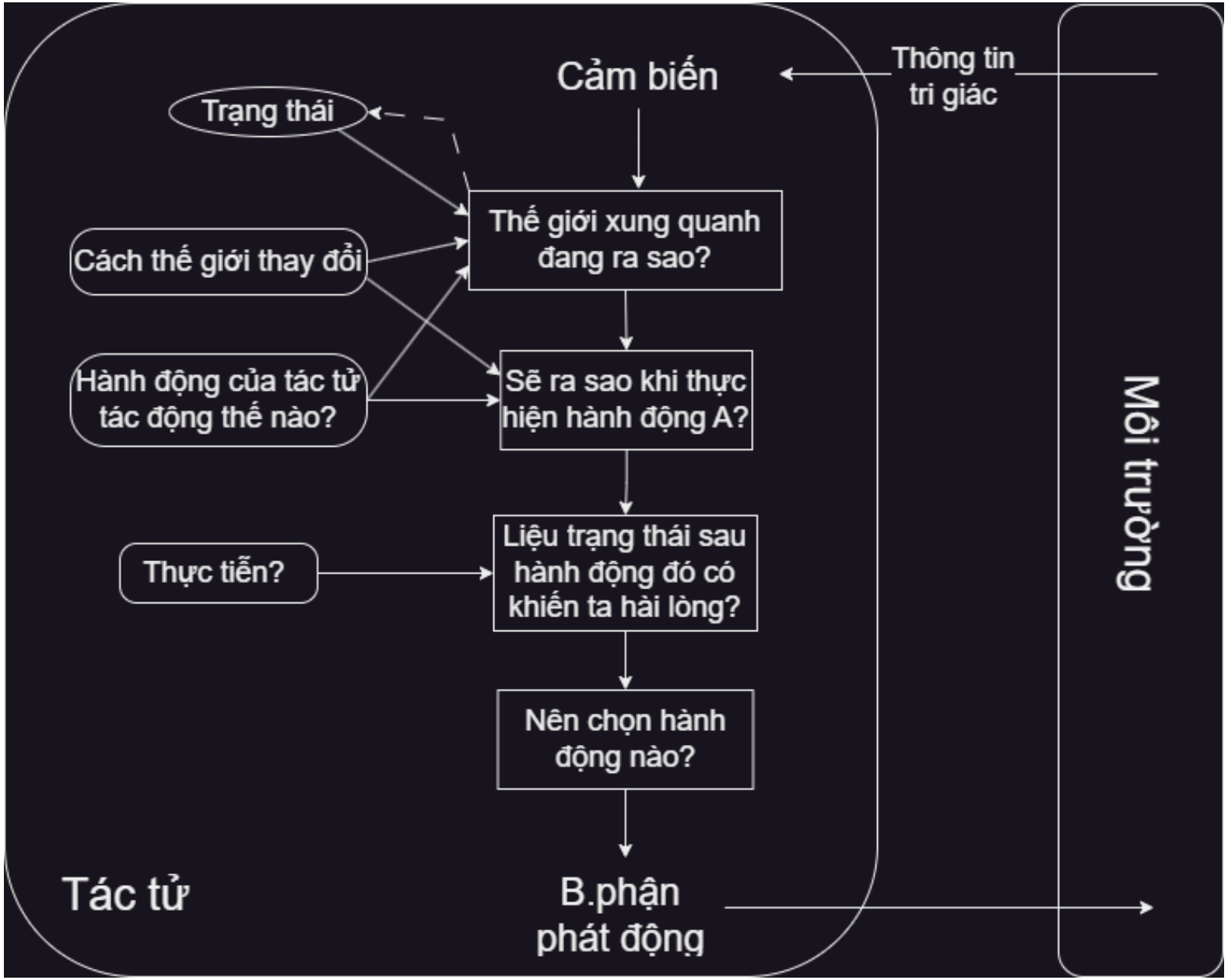
- The agent learn the goal and find a way to meet.

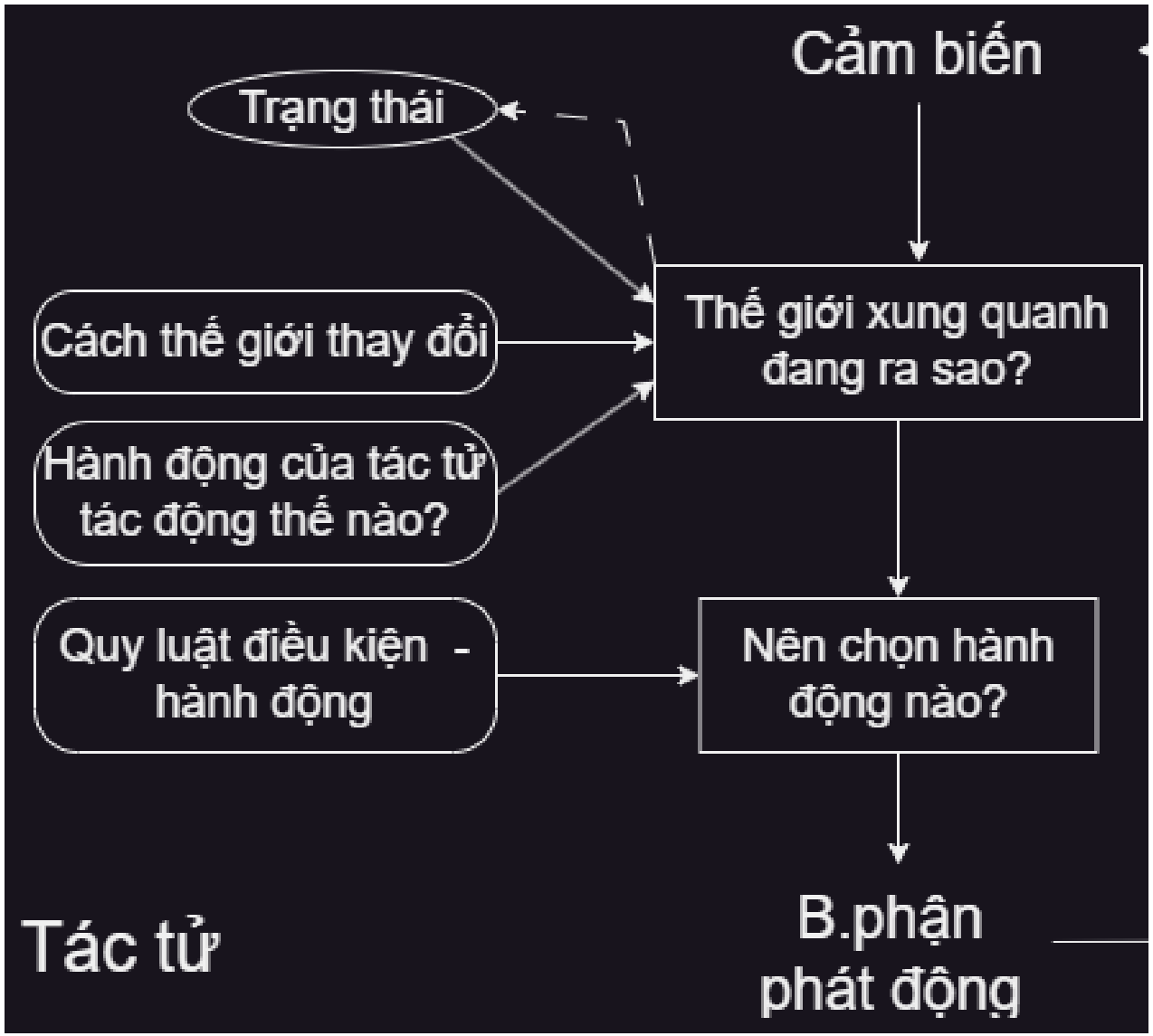


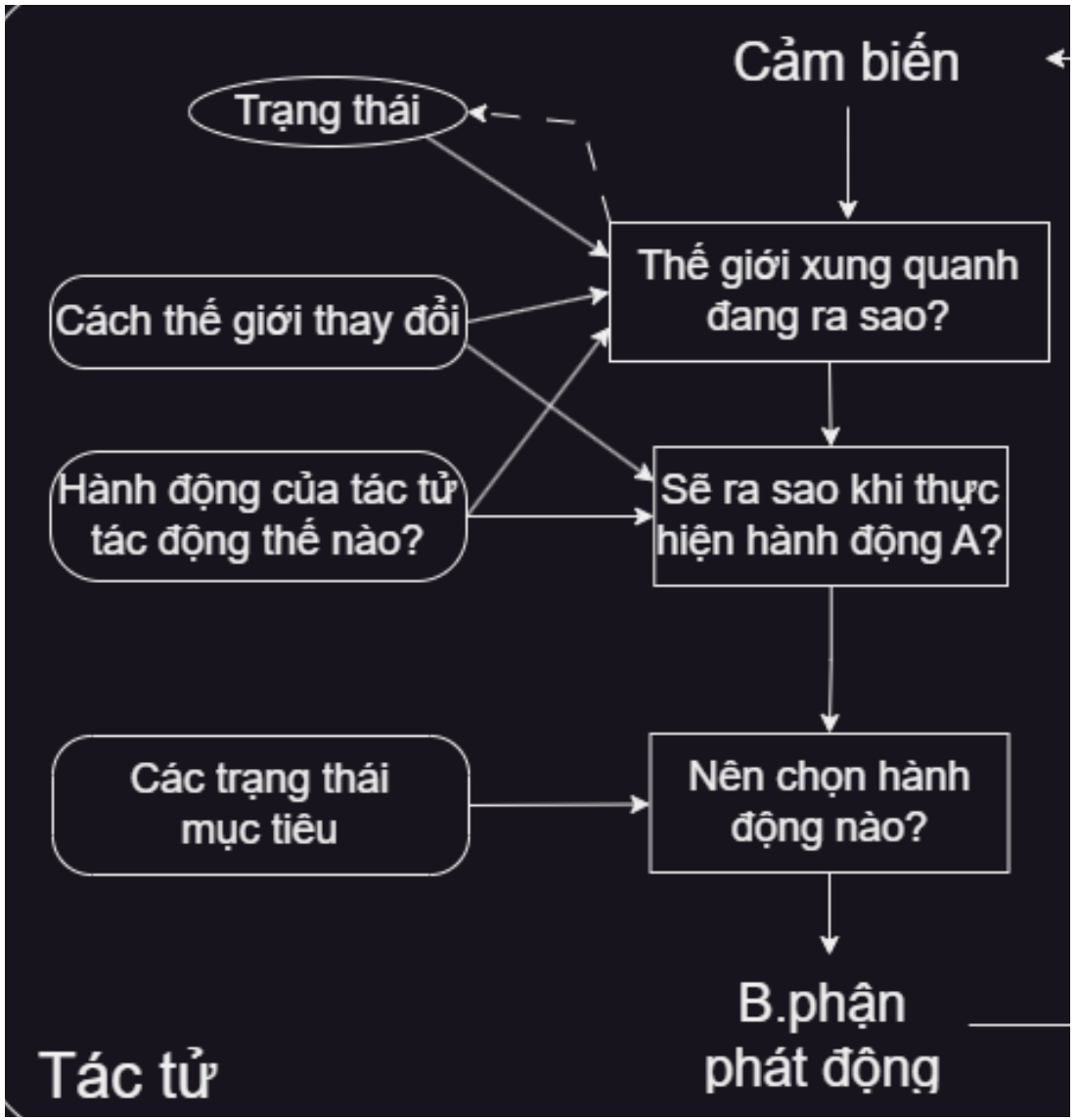
Utility-based agent

Find a useful & satisfying (best) solution.

Ex.: Root-finding problem




Cảm biến
Trạng thái
Thế giới xung quanh đang ra sao?
Cách thế giới thay đổi
Hành động của tác tử tác động thế nào?
Quy luật điều kiện - hành động
Nên chọn hành động nào?
Tác tử
B.phận phát động


Cảm biến
Trạng thái
Thế giới xung quanh đang ra sao?
Cách thế giới thay đổi
Hành động của tác tử tác động thế nào?
Sẽ ra sao khi thực hiện hành động A?
Các trạng thái mục tiêu
Nên chọn hành động nào?
Tác tử
B.phận phát động

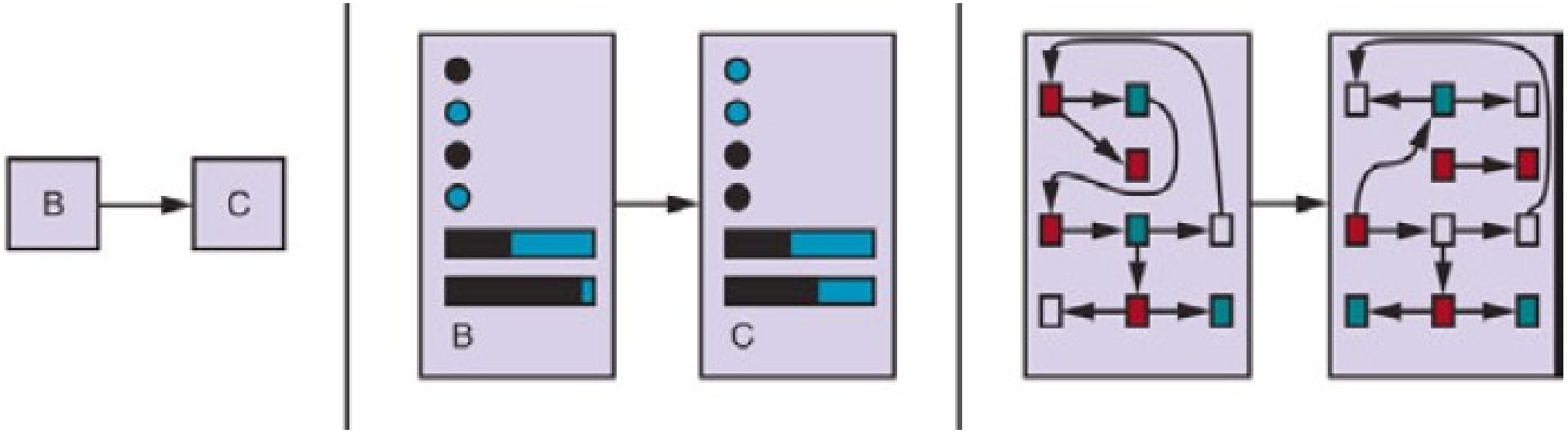
Model-free agent

- An opposite of model-based one.
- Does not estimate the transition.
- Possible solution: Trial-and-error (greedy).

Learning agents

- The preferred type nowadays.
- The idea popped up by Turing in 1950:
  - A model that can be taught time by time.
- Many agents of other types can be upgraded to be learnable.

II. Environment abstraction



Atomic
List/Array
Tuple
Str
Num...

Factored
Class
Structured

Structured
Relationship

Structured environment

A cow is drinking milk stored in a bottle.

III. Searching for goals

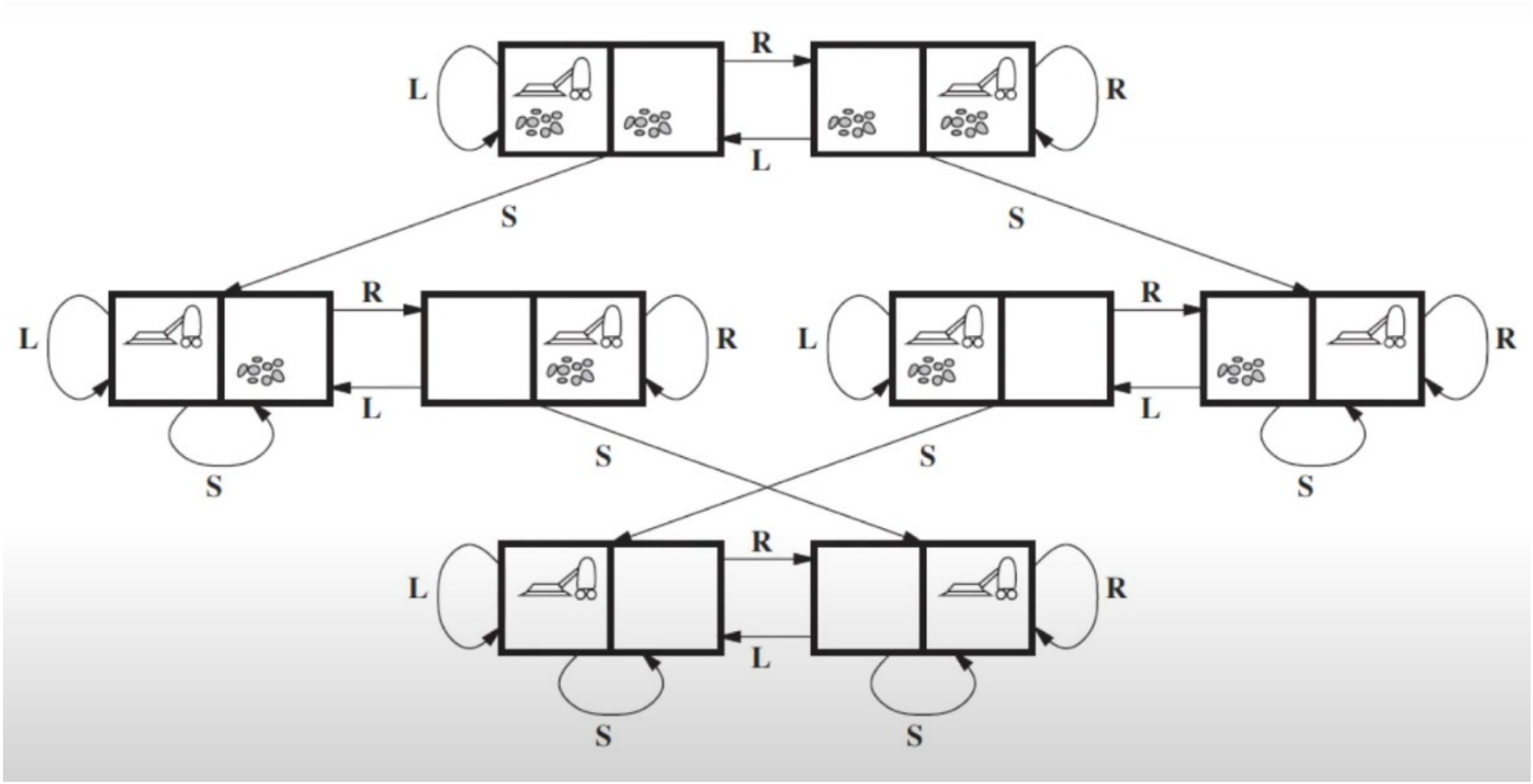
- **Searching:** An action to find the **goals.**
- **Searching** is for **problem-solving agents.**
  - The states must be **atomic.**
- For **factored or structured** ones, we build **planning agents.**
- **Tree** searching >< **Graph** searching.

4 steps:

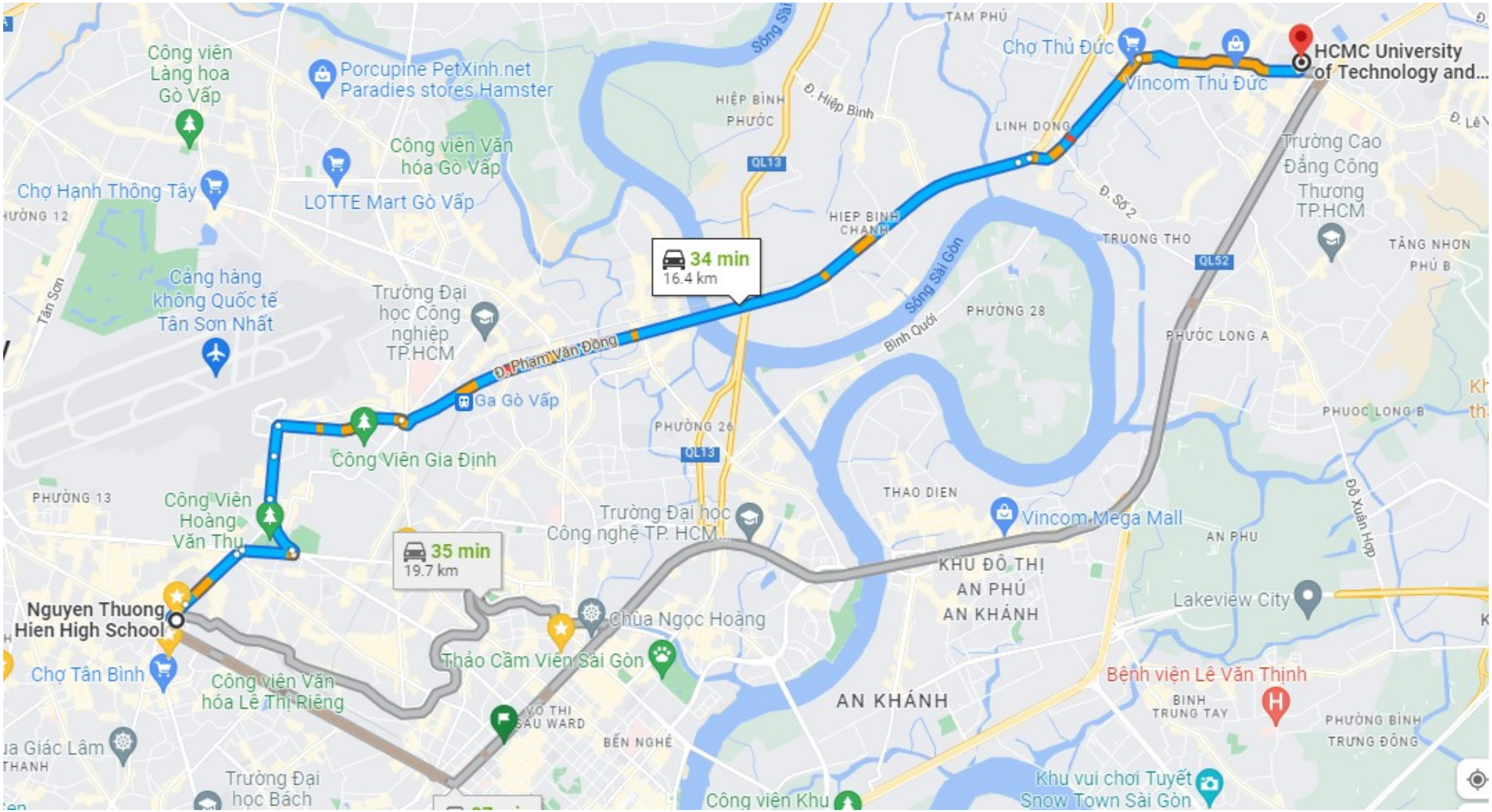
1. Formulize the goal(s): The states that must be achieved.
2. Formulize the environment.
3. Searching.
4. Execute.

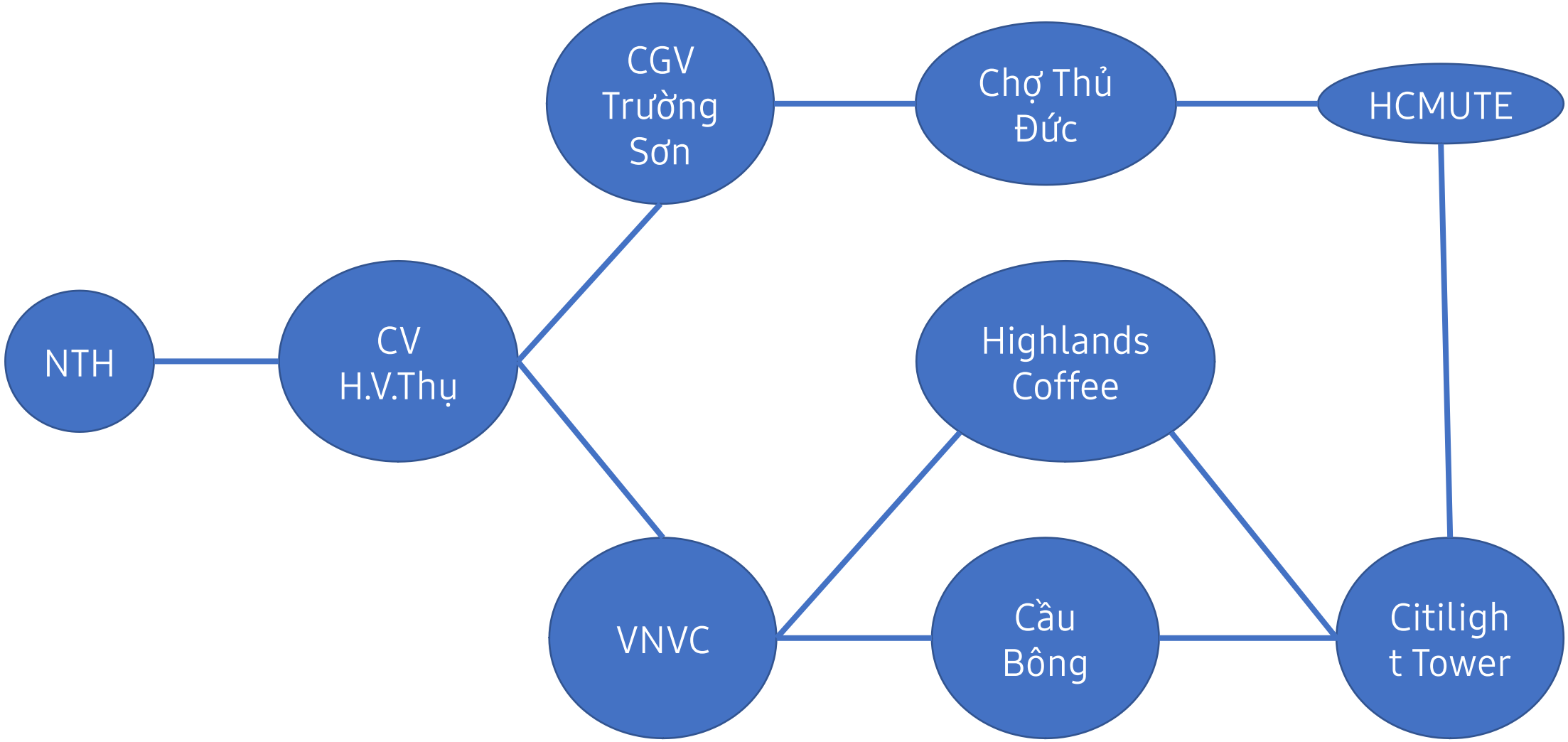
- State space: ...
- Initial state: ...
- Actions: ...
- Transition model: ...
- Cost: ...

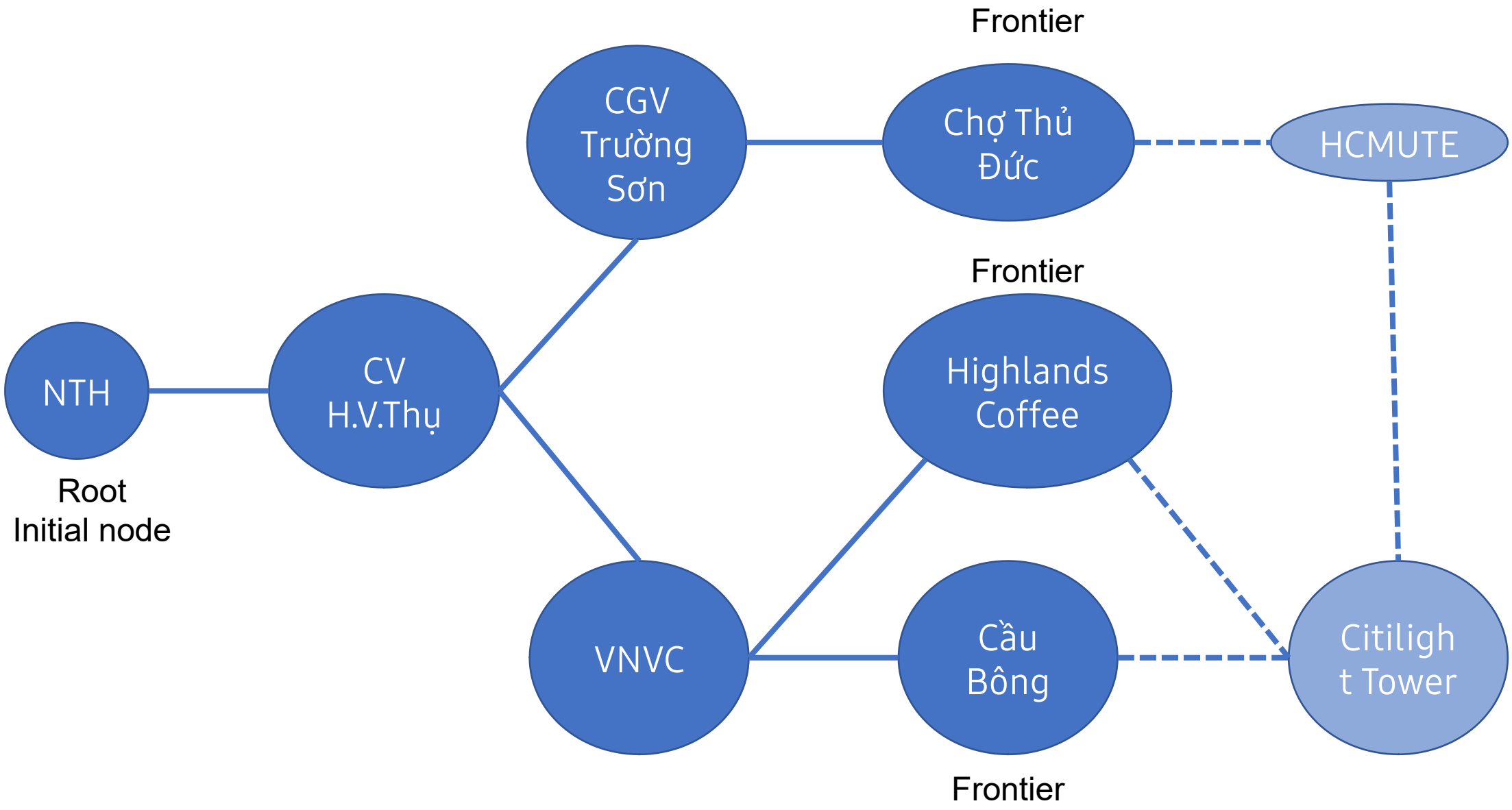
- State space: Position of tiles and blank.
- Initial state: Any.
- Actions: Move the blank.
- Transition model:
- Cost: 1 step == 1 unit of cost.

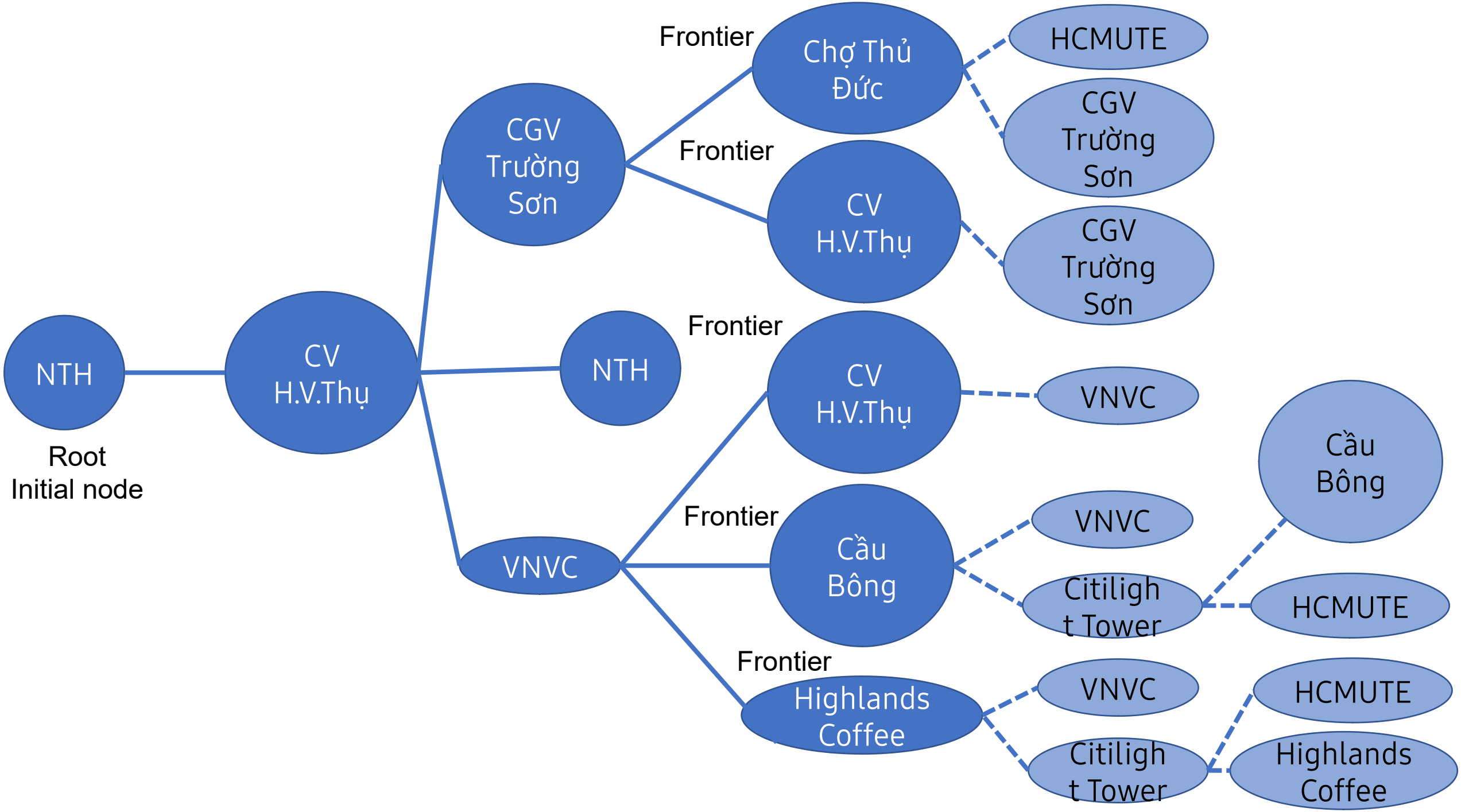


Cleaner problem


Công viên Làng hoa Gò Vấp
Porcupine PetXinh.net Paradies stores Hamster
Công viên Văn hóa Gò Vấp
LOTTE Mart Gò Vấp
Chợ Hạnh Thông Tây
Cảng hàng không Quốc tế Tân Sơn Nhất
Trường Đại học Công nghiệp TP.HCM
Đ. Phạm Văn Đồng
Ga Gò Vấp
Công Viên Gia Định
34 min
16.4 km
TAM PHÚ
Chợ Thủ Đức
Vincom Thủ Đức
HCMC University of Technology and...
HIỆP BÌNH PHƯỚC
Đ. Hiệp Bình
LINH DONG
QL13
HIEP BINH CHANH
Đ. Số 2
TRUONG THO
Trường Cao Đẳng Công Thương TP.HCM
TĂNG NHƠN PHÚ B
QL52
Sông Sài Gòn
Bình Quới
PHƯỜNG 28
PHƯỚC LONG A
PHUOC LONG B
PHƯỜNG 26
QL13
THAO DIEN
Vincom Mega Mall
AN PHU
Đỗ Xuân Hợp
KHU ĐÔ THỊ AN PHÚ AN KHÁNH
Lakeview City
PHƯỜNG 13
Công Viên Hoàng Văn Thụ
35 min
19.7 km
Trường Đại học Công nghệ TP. HCM...
Nguyen Thuong Hien High School
Chùa Ngọc Hoàng
Thảo Cầm Viên Sài Gòn
Chợ Tân Bình
Công viên Văn hóa Lê Thị Riêng
VÕ THỊ SÁU WARD
BẾN NGHÉ
AN KHÁNH
Bệnh viện Lê Văn Thịnh
BÌNH TRỊ TÂY
PHƯỜNG BÌNH TRƯNG ĐÔNG
Trường Đại học Bách
Khu vui chơi Tuyết Snow Town Sài Gòn
Công viên Khu

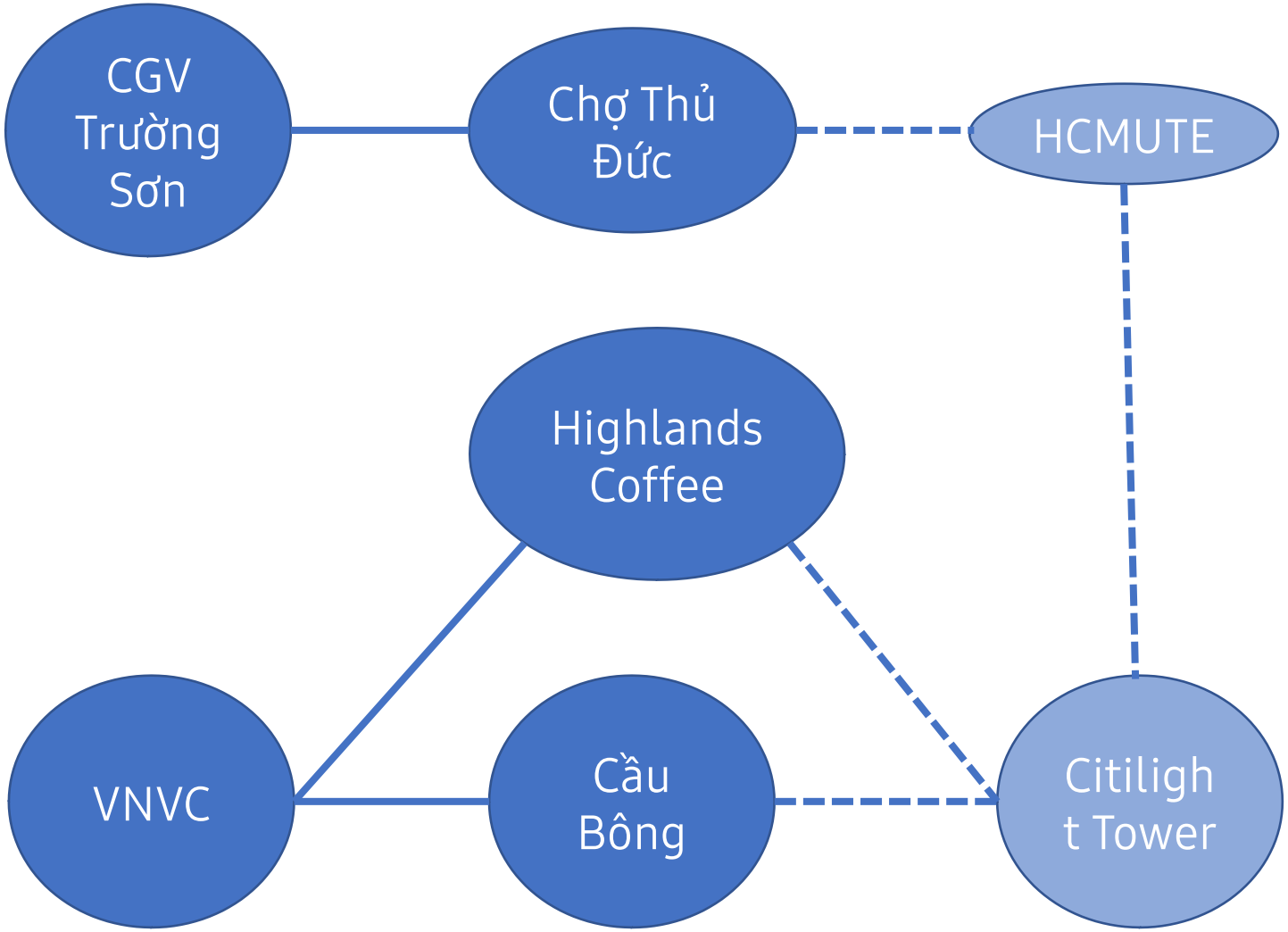

CGV Trường Sơn
Chợ Thủ Đức
HCMUTE
NTH
CV H.V.Thụ
Highlands Coffee
VNVC
Cầu Bông
Citiligh t Tower


Frontier
CGV Trường Sơn
Chợ Thủ Đức
HCMUTE
Frontier
NTH
CV H.V.Thụ
Highlands Coffee
Root
Initial node
VNVC
Cầu Bông
Citilight Tower
Frontier


NTH
Root
Initial node
CV H.V.Thụ
CGV Trường Sơn
NTH
VNVC
Frontier
Chợ Thủ Đức
Frontier
CV H.V.Thụ
HCMUTE
CGV Trường Sơn
CGV Trường Sơn
Frontier
CV H.V.Thụ
VNVC
Frontier
Cầu Bông
VNVC
Cầu Bông
Citilight Tower
HCMUTE
Frontier
Highlands Coffee
VNVC
HCMUTE
Citilight Tower
Highlands Coffee

- Evaluation function: Return values to evaluate for comparing, frontier selection.
- The function presents the strategies.
- Types of func: Cost, performance (metric), loss, score,...
- Other names: Objective func, heuristic evaluation func.

1. Create an empty list L of frontiers.
2. Add the root to L.
3. Loop:
    1. If there is no frontiers, return no result.
    2. Get the target T, which is the best frontier judged by **evaluation function**.
    3. If T is goal, return the action(s).
    4. Else, expand T and add new frontiers to L.



Best-first search

Figure 3.7

```
function BEST-FIRST-SEARCH(problem, f) returns a solution node or failure
  node ← NODE(STATE=problem.INITIAL)
  frontier ← a priority queue ordered by f, with node as an element
  reached ← a lookup table, with one entry with key problem.INITIAL and value node
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
      s ← child.STATE
      if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
        reached[s] ← child
        add child to frontier
  return failure

function EXPAND(problem, node) yields nodes
  s ← node.STATE
  for each action in problem.ACTIONS(s) do
    s' ← problem.RESULT(s, action)
    cost ← node.PATH-COST + problem.ACTION-COST(s, action, s')
    yield NODE(STATE=s', PARENT=node, ACTION=action, PATH-COST=cost)
```

- 2 group of algorithms:
  - Informed: Pick the better way by evaluating.
    - **Greedy best-first search**
    - A* search
  - Uninformed: Only based on goals and states.
    - Breadth-first search
    - depth-first search

Tree search

+ Memory saving.
+ Natively inf. Loop avoidance.

Graph search

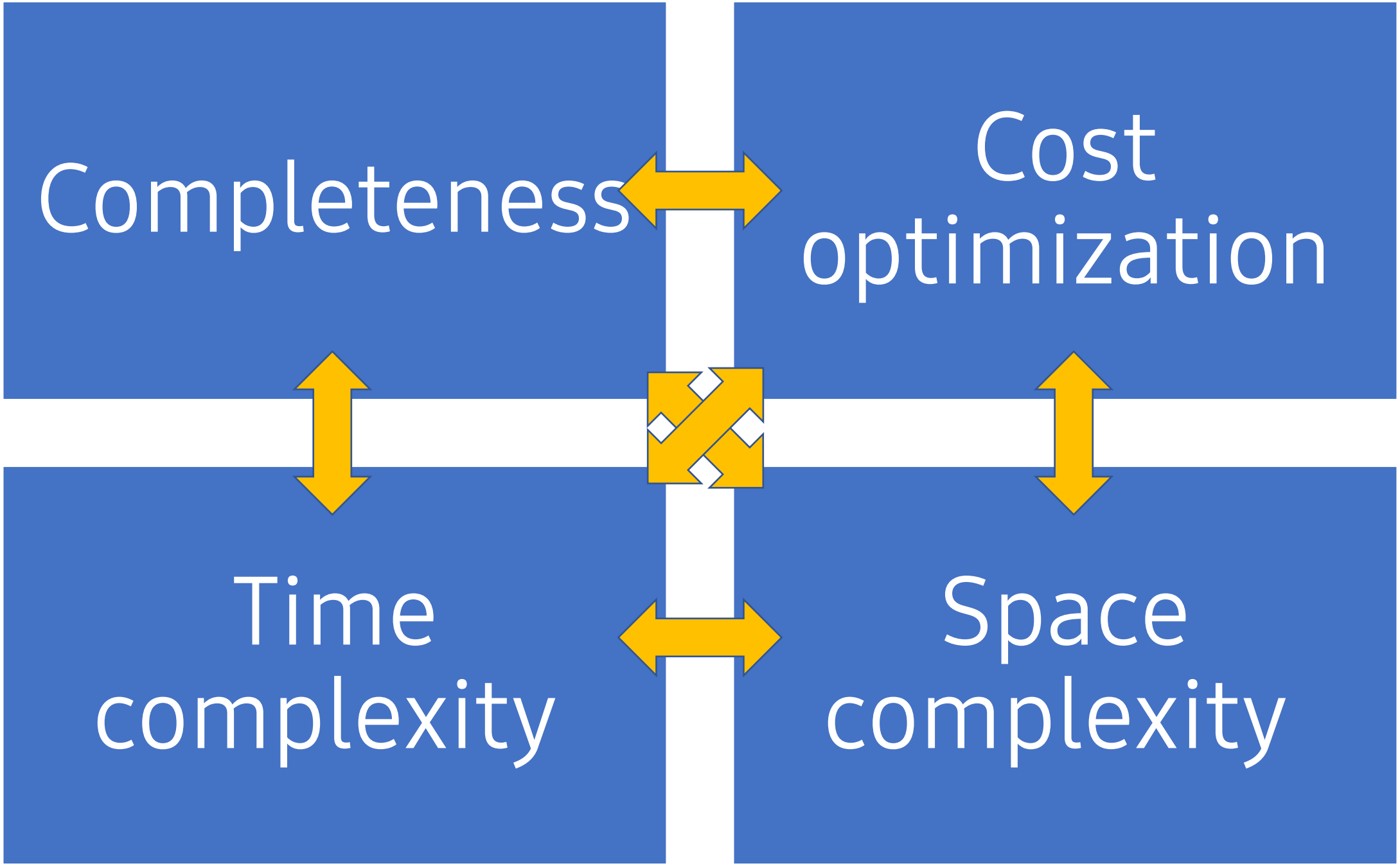
- Can cause inf. Loop.
- Use more memory.

Completeness

Cost optimization

Time complexity

Space complexity

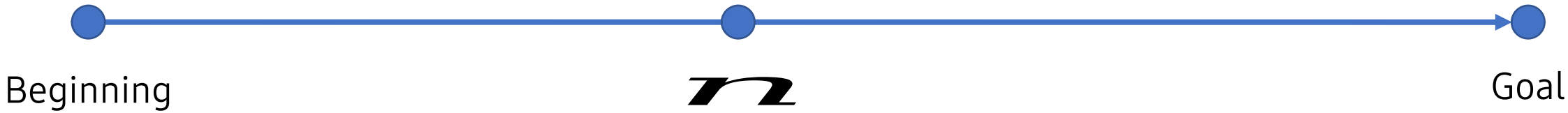

Completeness
Cost optimization
Time complexity
Space complexity

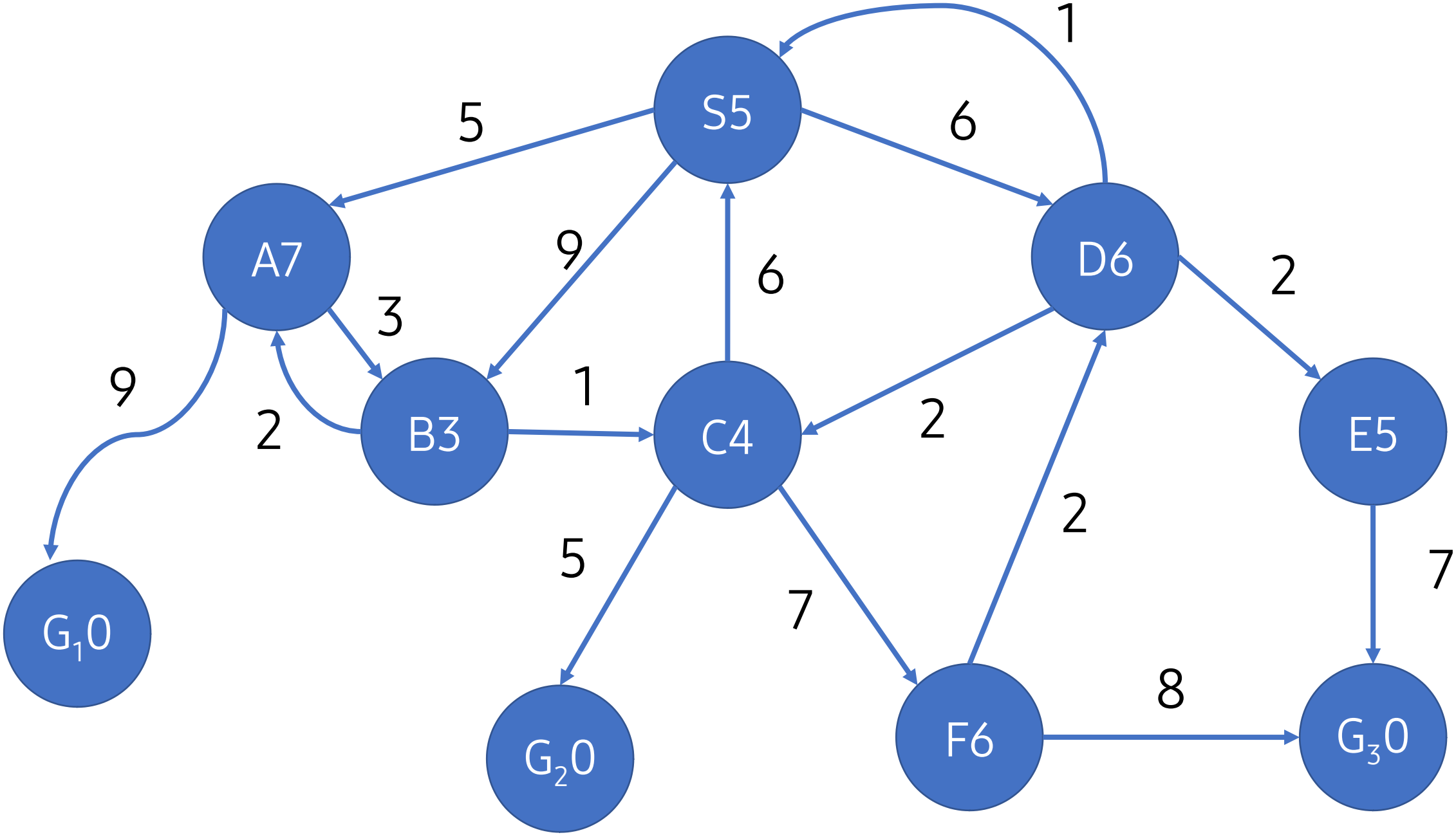
III. A* search

- Using the evaluating function:

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

- In which:
  - is the cost determinator from the beginning to .
  - is the lowest cost estimator from to goal.




S5
A7
D6
B3
C4
E5
G1 0
G2 0
F6
G3 0
1
5
6
9
6
2
3
9
2
1
2
2
5
7
7
8