Uninformed Search

LAURA CORTÉS-RICO (www.cortes-rico.com)

Multimedia Engineering

Artificial Intelligence

Problem-solving

What to do to find the **sequence of actions** that allow reaching the **goal (desirable state)**?

Uninformed Search

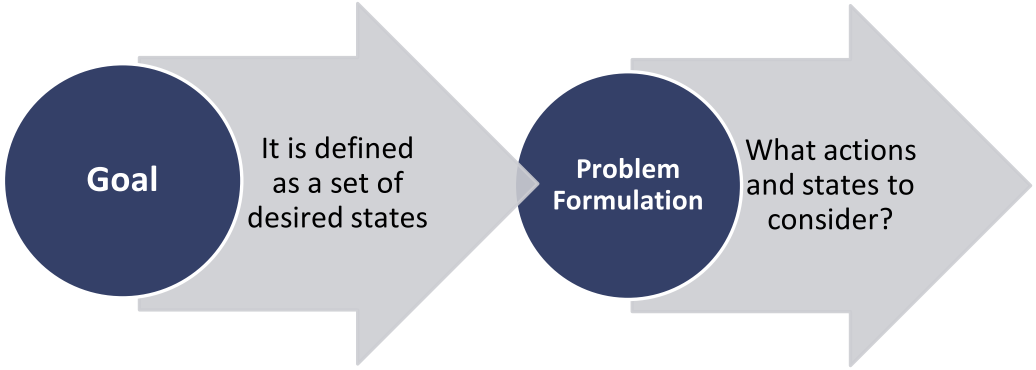
1. Search the sequence of actions.
2. Uninformed? The only available information is the definition of the problem.

Problem solving

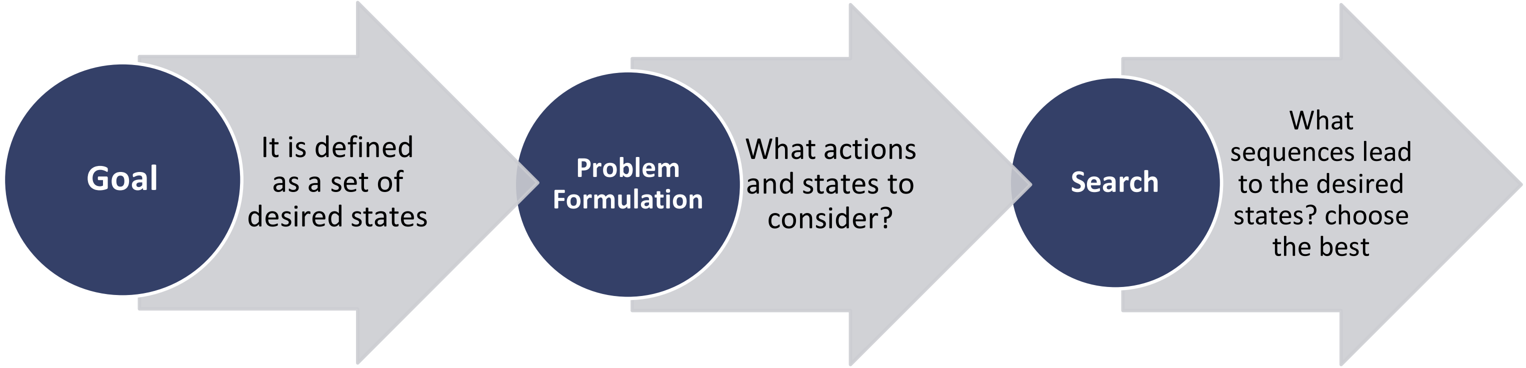
Goal

It is defined as a set of desired states

Problem solving



Problem solving



Problem solving

- **Goal** — It is defined as a set of desired states
- **Problem Formulation** — What actions and states to consider?
- **Search** — What sequences lead to the desired states? choose the best
- **Solution** — It takes a problem as input and returns the solution as a sequence.

Problem solving

- **Goal**: It is defined as a set of desired states
- **Problem Formulation**: What actions and states to consider?
- **Search**: What sequences lead to the desired states? choose the best
- **Solution**: It takes a problem as input and returns the solution as a sequence.
- **Execution**: Execute the recommended actions in the solution

Problem Solver Agent Algorithm

```
function solve returns action
        inputs: perception
        static: seq: Sequence of actions, initially empty.
                  state: Current state
                  goal: Initially null
                  problem: Problem formulation, initially null
        state <- UPDATE-STATE (state, perception)
        if (seq is empty) then
                 goal <- GOAL_FORMULATION(state)
                 problem <- PROBLEM_FORMULATION(state, goal)
                seq <- SEARCH (problem)
        end_if
        action <- FIRST(seq)
        seq <- LEFT (seq)
        returns action
end_function
```

Environment

1. Static
2. Completely observable
3. Discret
4. Sequence
5. Deterministic
6. Mono-agent

PROBLEM_FORMULATION

Problem Formulation

How is the agent initially?

Initial State

Problem Formulation

How is the agent initially?

Initial State

successor function: given a state, returns a set of ordered pairs <action, successor>

Actions

Problem Formulation

How is the agent initially?

Initial State

successor function: given a state, returns a set of ordered pairs <action, successor>

Actions

Is the current the **goal state**?

Goal test

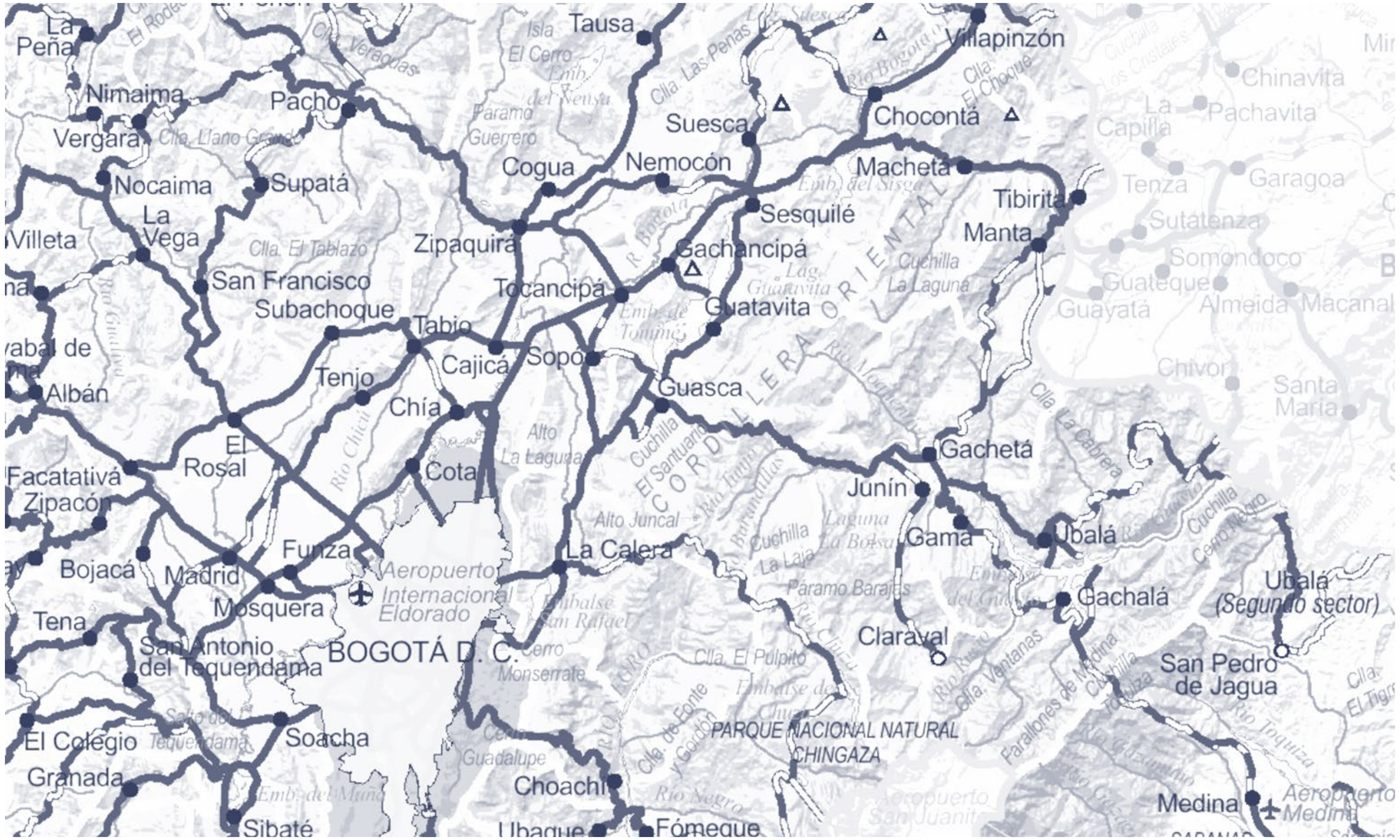
Problem Formulation

How is the agent initially?	**successor** function: given a state, returns a set of ordered pairs <action, successor>	Is the current the **goal state**?	Numerical cost of each path. A path is a **sequence of states**.
Initial State	**Actions**	**Goal test**	**Path cost**

GLOSSARY

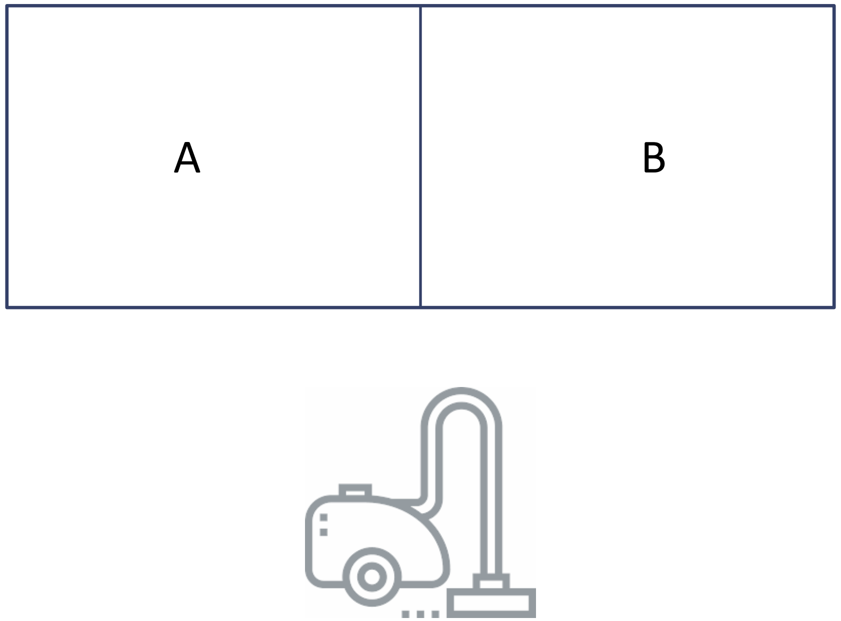
1. **Initial State:** State in which the agent **starts**
2. **Actions:** Concrete **execution**. E.g. Move to left.
3. **Successor Function:** Function that, **given a state**, returns a set of **ordered pairs** where each pair defines a possible action from the parameter state and the consequent state after executing the action.
4. **States space:** Set of all the possible **states**.
5. **Path:** State sequence connected by **actions**.
6. **Goal Test:** Function that determines if a given state is the **goal state**.
7. **Path cost:** Numeric cost associated with a path.
8. **Individual cost:** Associated to a given action.
9. **Abstract:** Representing eliminating details.

Example


Tausa
Villapinzón
Nimaima
Pacho
Vergara
Suesca
Chocontá
Cogua
Nemocón
Machetá
Nocaima
Supatá
Tibiritá
Sesquilé
Manta
Villeta
Zipaquirá
Gachancipá
San Francisco
Subachoque
Tocancipá
Guatavita
Tabio
Cajicá
Sopó
Tenjo
Guasca
Albán
Chía
El Rosal
Cota
Gachetá
Facatativá
Zipacón
Junín
Funza
La Calera
Gama
Ubalá
Bojacá
Madrid
Mosquera
Aeropuerto Internacional Eldorado
Gachalá
Tena
San Antonio del Tequendama
BOGOTÁ D. C.
Claraval
San Pedro de Jagua
El Colegio
Soacha
Granada
Choachí
Sibaté
Fómeque
Medina
PARQUE NACIONAL NATURAL CHINGAZA
Chinavita
Pachavita
Garagoa
Tenza
Sutatenza
Somondoco
Guateque
Guayatá
Almeida
Macanal
Chivor
Santa María

Example

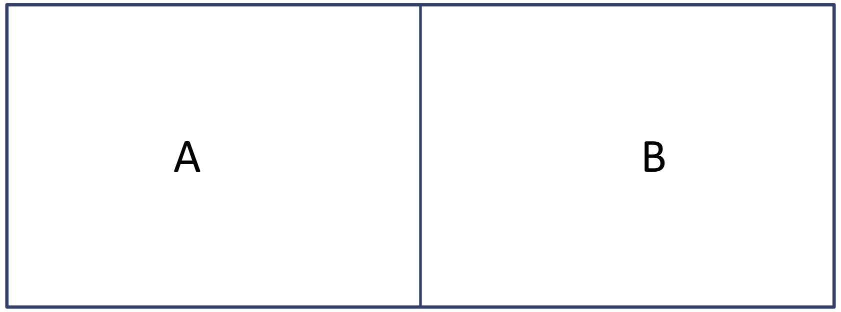
"Toy problem" (Russell, 2008)

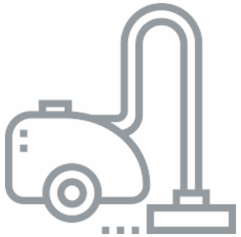


- States:
- Initial State:
- Successor Function:
- Goal test:
- Cost:

Example

"Toy problem" (Russell, 2008)

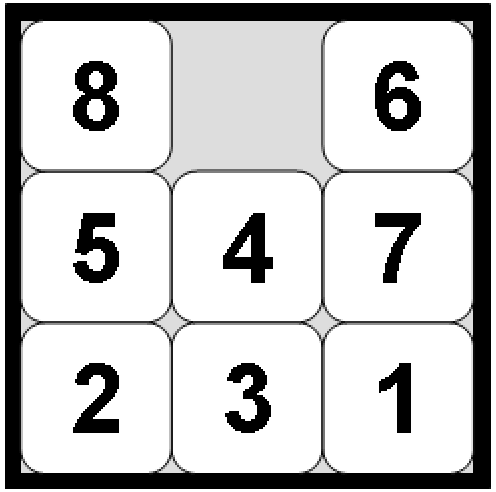


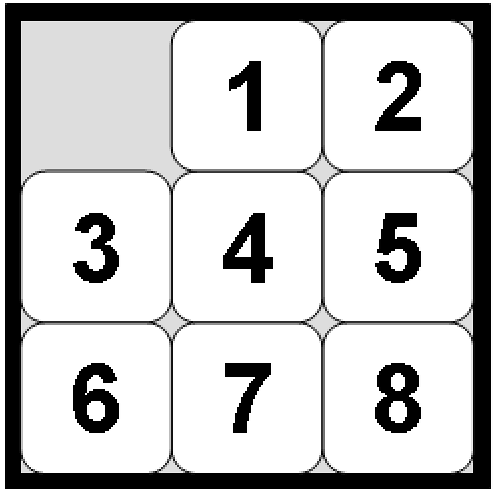


- **States:** 2 x 2^2: (A,B,CleA), (A,B, CleA), (A,B, CleA), (A,B, CleA), (A,B, CleB), (A,B, CleB), (A,B, CleB), (A,B, CleB)
- **Initial State:** Any of the 8 possible.
- **Successor function:** <action,successor> donde acción Left, Right, Clean.
- Goal test: Is the state (A,B,CleA) or (A,B, CleB)?
- Cost: Each action costs 1.

Example

“Toy problem”: 8-Puzzle (Russell, 2008)

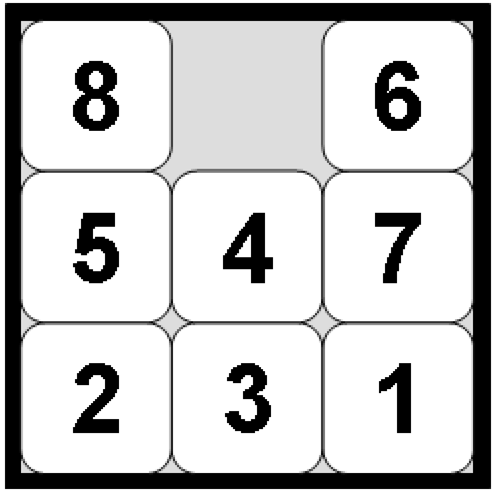


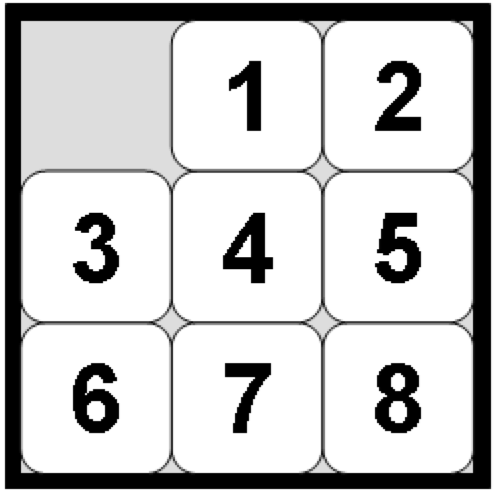


- States:
- Initial State:
- Successor Function:
- Goal test:
- Cost:

Example

“Problema de juguete”: 8-Puzzle (Russell, 2008)

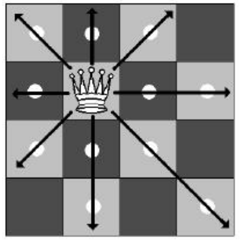


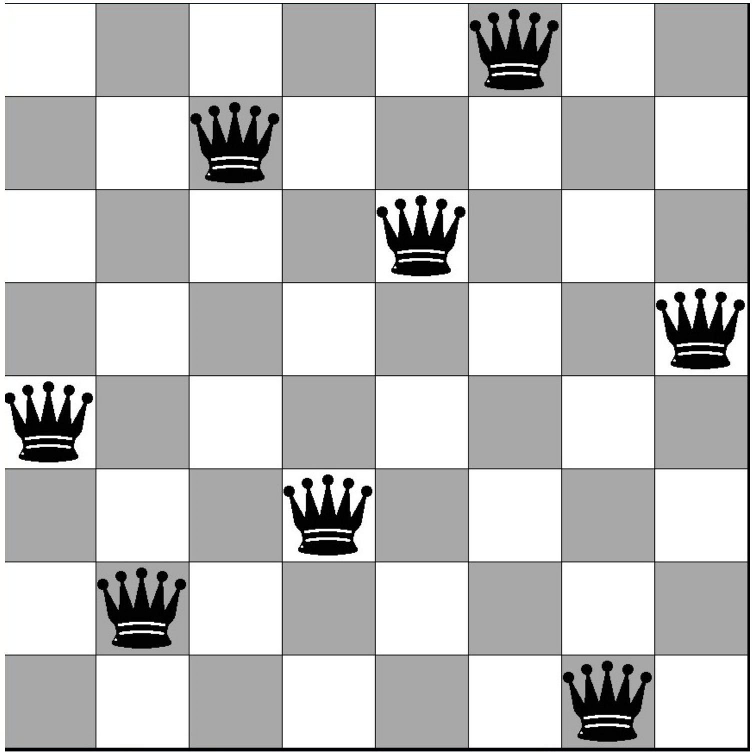


- **States:** All possible configurations: 9!
- **Initial State:** Any of the 9!
- **Function successor:<**action,successor>
  Possible actions: move the hole up, down, left, right.
- **Goal test:** Is the board ordered? (The desirable state may change)
- **Cost:** Each movement costs 1.

Example

"Toy problems": 8 Queens (Russell, 2008)





- **States:** 64x63x62x61x60x59x58x57 different configurations of the board with 8 Queens.
  Initial State: Any of the previous.
- **Function successor:** <action,successor>
  Possible actions: Move queen N to the position (x,y).
- **Goal test:** State with 8 Queens on the board, with none attacking other. (92 options)
- **Cost:** Each movement costs 1.

Real world problems

Route finding (Russell, 2008)

- **States**: Each state obviously includes a location (e.g., an airport) and the current time. Furthermore, because the cost of an action (a flight segment) may depend on previous segments, their fare bases, and their status as domestic or international, the state must record extra information about these "historical" aspects.
- **Initial state**: This is specified by the user's query.
- **Actions**: Take any flight from the current location, in any seat class, leaving after the current time, leaving enough time for within-airport transfer if needed.
- **Transition model**: The state resulting from taking a flight will have the flight's destination as the current location and the flight's arrival time as the current time.
- **Goal test**: Are we at the final destination specified by the user?
- **Path cost**: This depends on monetary cost, waiting time, flight time, customs and immigration procedures, seat quality, time of day, type of airplane, frequent-flyer mileage awards, and so on.

Real world problems

Turist

Travel guide

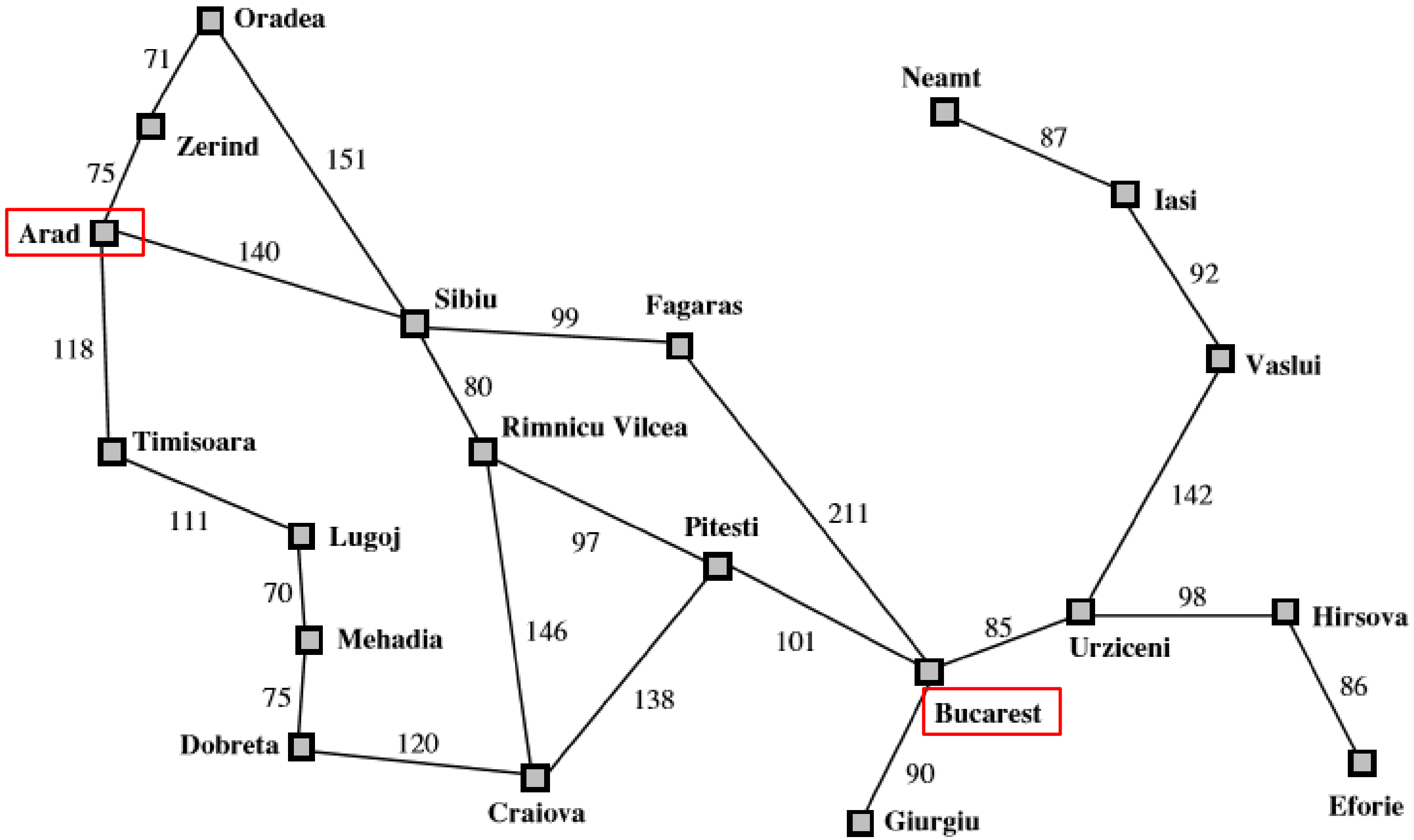
VLSI

Robotic navigation

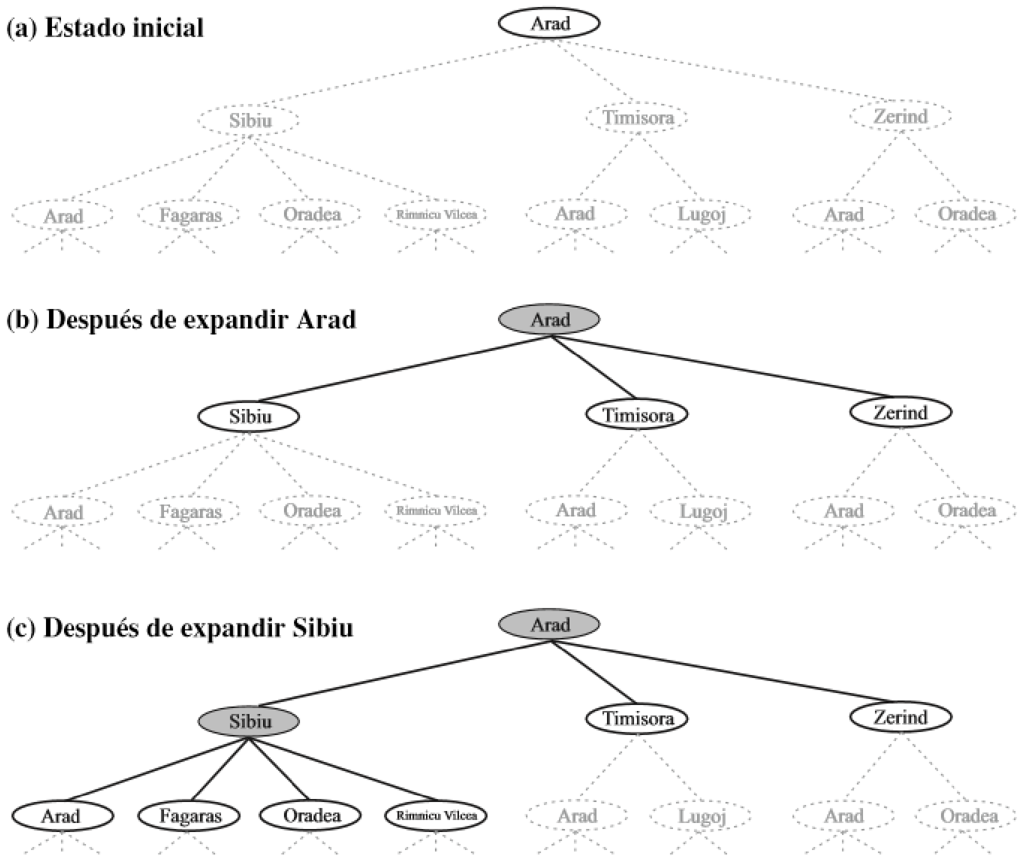
SEARCH

Search tree

(Russell, 2008)



Search tree



(Russell, 2008)

Search Strategy

What is a **node**?

Data **structure**:

1. STATE
2. PARENT
3. ACTION
4. PATH COST
5. DEPTH

What is a **node**?

Data **structure**:

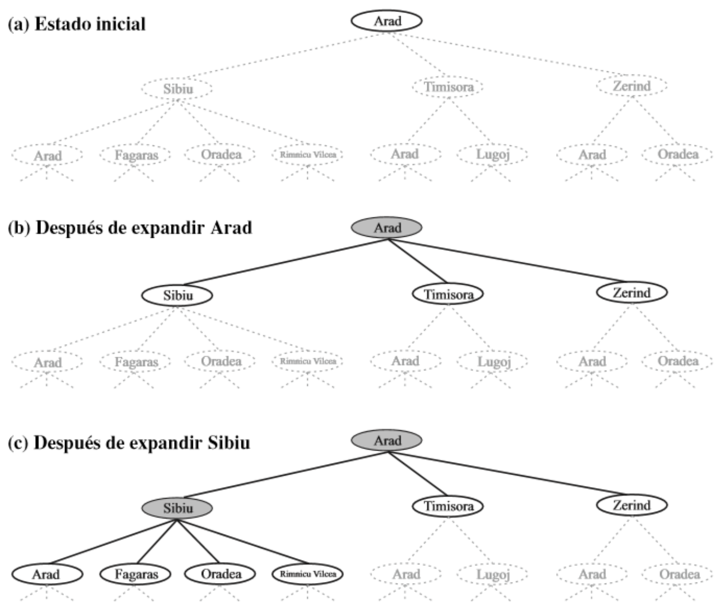
1. STATE: That of the set of states that corresponds to the node.
2. PARENT: **Node** that generates this node**.**
3. ACTION: The **concrete execution** that the father carried on to generate this node.
4. PATH COST: Cost from the initial state to the node.
5. DEPTH: Number of steps from the initial state to the node.

Search algorithm

```
function search returns solution or fail
        starts tree with the initial state
        do
                if there are no candidates to expand then
                        returns fail
                end_if
                choose the candidate to expand, according to the strategy
                if node contains the goal then
                        return solution
                else
                        expand node and complete the tree with the expanded nodes.
                end_if
        end_do
end_function
```

Glossary

1. **Frontier:** Generated but not expanded nodes.
2. **Leaf node:** Node without successors (childs).
3. **QUEUE:** FIFO.
4. **Performance:** Completeness. If there is a solutions, the algorithm finds it.
5. **Performance:** Optimality, It finds the optimal solution
6. **Performance:** Time complexity, time to find the solution.
7. **Performance:** Space complexity, required memory to find the solution



Research algorithm in trees

```
function searchTree (problem, frontier) return solution or fail
        frontier <-INSERT(DO-NODE(INITIAL-STATE[problem]),frontier)
        do
                if EMPTY?(frontier) then
                        return fail
                end_if
                node<-POP(frontier)
                if GOAL_TEST[problem] applied to STATE[node] then
                        return SOLUTION(node)
                end_if
                frontier<-INSERT-ALL(EXPAND(node,problem),frontier)
        fin_hacer
fin_función
```

Algorithm EXPAND

```
function EXPAND(node, problem) return set of nodes
        succesors<-empty set
        for each(action, result) in SUCCESSOR-FN[problem](STATE[node]) do
                s<- new NODE
                STATE[s]<-result
                NODE-PARENT[s]<-node
                ACTION[s]<-action
                PATH-COST[s]<-PATH-COST[node]+INDIVIDUAL-COST(node,action,s)
                DEPTH[s]<-DEPTH[node]+1
                add s to succesors
        end_for
        return succesors
End_function
```

Assignment

1. Breadth-first search (*Búsqueda primero en anchura*)
2. Uniform-cost search (*Búsqueda de costo uniforme*)
3. Depth-first search (*Búsqueda primero en profundidad (y hacia atrás)*)
4. Depth-limited search (*Búsqueda de profundidad limitada*)
5. Iterative deepening depth-first search (*Búsqueda primero en profundidad con profundidad iterativa*)
6. Bidirectional search (*Búsqueda bidireccional*)

- Explanation
- Example
- Advantages
- Disadvantages

References

González, E. (2015) *Agentes Racionales y SMA, Notas de clase*. Pontificia Universidad Javeriana.

González, J. (n.d.) *Inteligencia Artificial. Tema 1: Introducción*. Recuperada de: https://ccc.inaoep.mx/~jagonzalez/AI/Sesion1_Introduccion.pdf Julio 2017.

Russell, S. & Norvig P. (2008) *Inteligencia Artificial: Un Enfoque Moderno*. Pearson, Prentice Hall.

Winston, P. (1992) Artificial Intelligence, 3ra edición. Addison-Wesley Publishing Company. Recuperado de: http://courses.csail.mit.edu/6.034f/ai3/rest.pdf Julio 2017.