Inteligencia Artificial

II – Resolución de problemas mediante búsquedas

1. Introducción

Dr. Edgard Iván Benítez Guerrero

cursofei@gmail.com

1. Introducción

- Agentes solucionadores de problemas
- Problemas y soluciones
- Ejemplos de problemas
- Búsqueda de soluciones

Agentes solucionadores de problemas

- Agentes basados en objetivos que deciden qué hacer para encontrar secuencias de acciones que conduzcan a estados deseables; i.e. encontrar soluciones a problemas
- Un agente de este tipo elije (o se le instruye) un objetivo (conjunto de estados del mundo) y trata de satisfacerlo
- Formulación del problema: dado un objetivo, es el proceso de decidir qué acciones y estados deben ser considerados en la búsqueda de los estados deseables
- Algoritmo de búsqueda
  - Entrada: un problema
  - Salida: secuencia de acciones

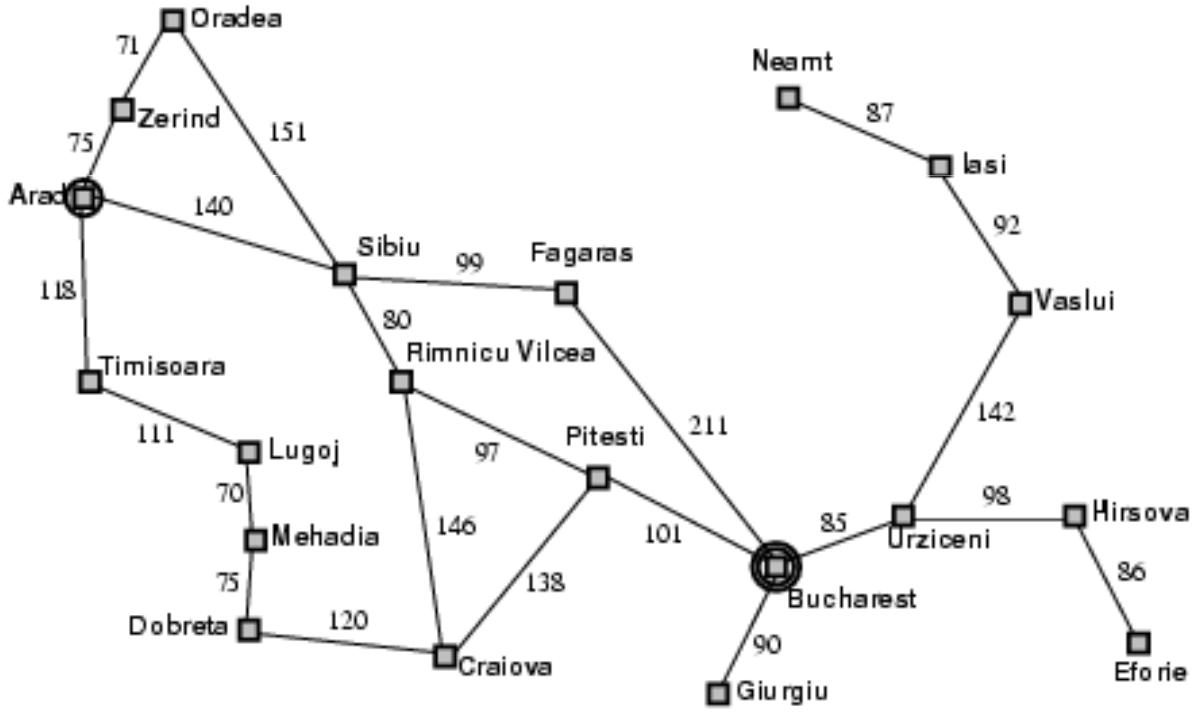
Agentes solucionadores de problemas

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
    static: 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 do
        goal ← FORMULATE-GOAL(state)
        problem ← FORMULATE-PROBLEM(state, goal)
        seq ← SEARCH(problem)
    action ← FIRST(seq)
    seq ← REST(seq)
    return action
```

Ejemplo: viaje en Rumania

- De vacaciones; actualmente en Arad. El vuelo de regreso sale de Bucarest
- Objetivo: llegar a Bucarest



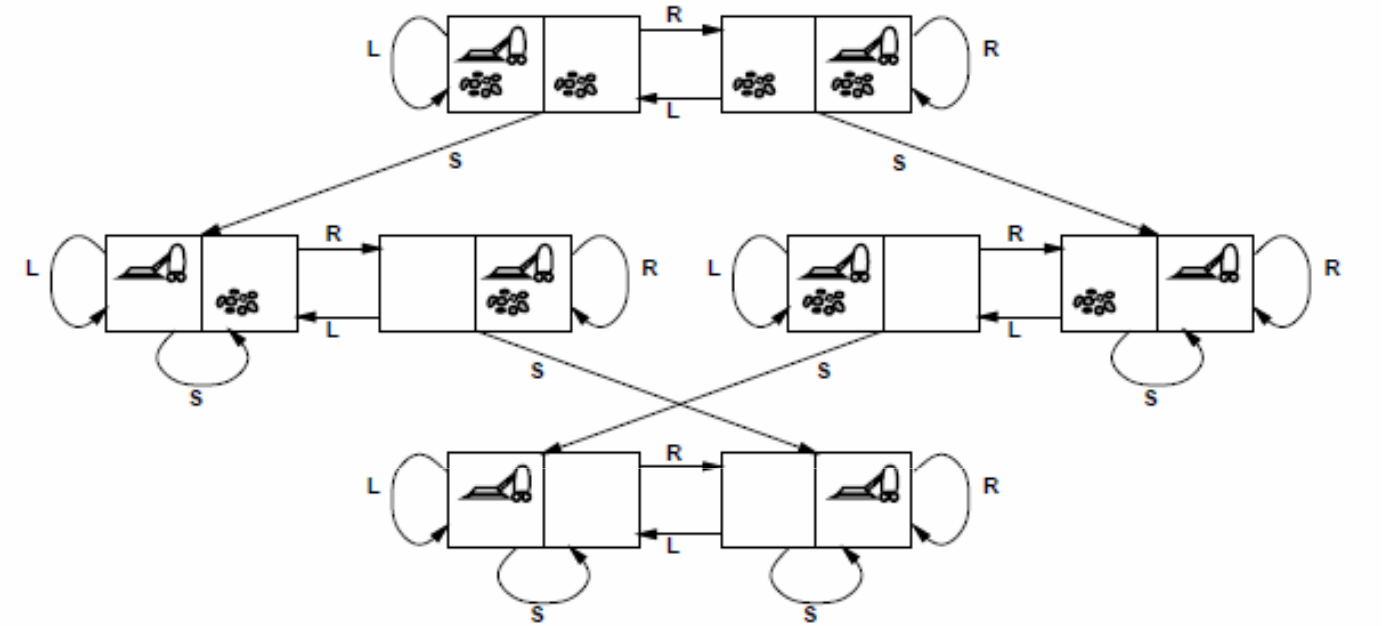
Ejemplo: viaje en Rumania

- Formulación del problema
  - estados: varias ciudades
  - Acciones: ir de una ciudad a otra
- Solución: encontrar una secuencia de ciudades para llegar a Bucarest a partir de Arad. Por ej.: Arad, Sibiu, Fagaras, Bucharest

Problemas y soluciones

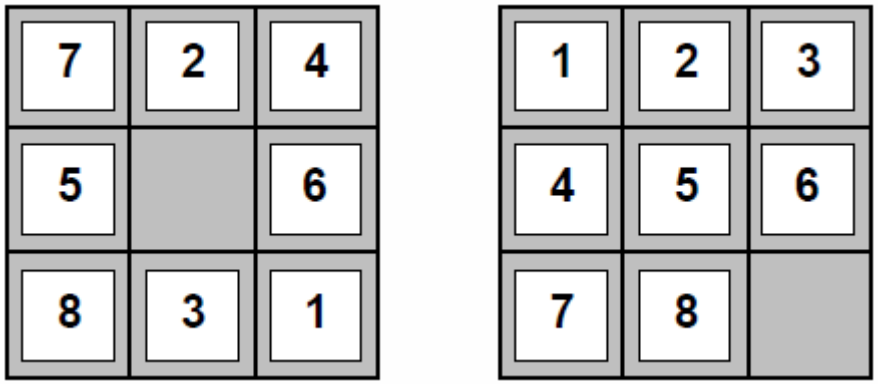
- Un problema consta de cuatro componentes
  - Estado inicial: estado en el que comienza el agente; p.ej: “en Arad“
  - Descripción de las posibles acciones disponibles por el agente: función sucesor *S(x)* = conjunto de pares acción–estado; p.ej. *S(Arad)* = {<*Arad* → *Zerind, Zerind*>, … }
  - Test objetivo: el cual determina si un estado es un estado objetivo; p.ej. “en Bucarest?”
  - Función de costo del camino: asigna un costo numérico a cada camino; usualmente descrita como la suma de los costos de las acciones individuales a lo largo del camino; p.ej. suma de distancias
- Una solución de un problema es una camino desde el estado inicial a un estado objetivo
  - Su calidad se mide por la función de costo del camino
  - La solución óptima tiene el costo más pequeño

Ejemplos de problemas: Mundo de la aspiradora



- Estados: ubicaciones del agente y del polvo = 8 estados posibles
- Estado inicial: cualquier estado puede ser designado como inicial
- Función sucesor: genera estados legales al intentar las tres acciones Izquierda, Derecha o Aspirar
- Test objetivo: comprueba si todos los cuartos están limpios
- Costo del camino: número de pasos que lo componen

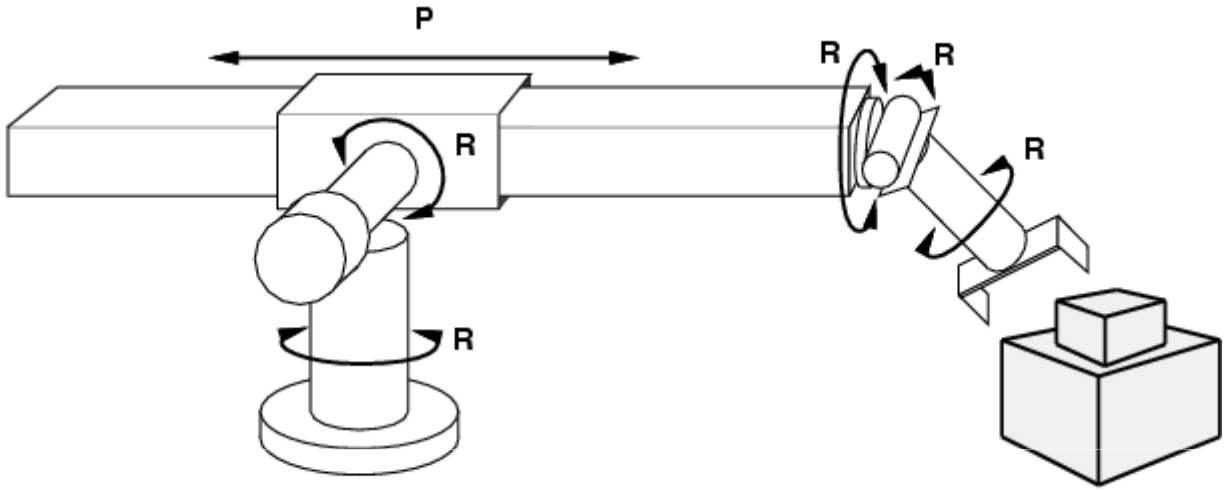
Ejemplos de problemas: Rompecabezas de 8 piezas



Estado inicial Estado final

- Estados: ubicaciones de las piezas
- Estado inicial: el que muestra la figura de la izquierda
- Función sucesor: genera estados legales al intentar mover el espacio vacío hacia a la izquierda, a la derecha, arriba o abajo
- Test objetivo: ¿se llegó al estado mostrado en la figura de la derecha?
- Costo del camino: 1 por movimiento

Ejemplos de problemas: ensamblado robótico



- Estados: coordenadas de las partes del brazo robótico más ubicación de las piezas del objeto a ser ensamblado
- Acciones: movimiento continuo de las piezas del brazo
- Test objetivo: ¿se terminó de ensamblar el objeto?
- Costo del camino: tiempo de ejecución

Ejemplos de problemas: Búsqueda de una ruta en una línea aérea

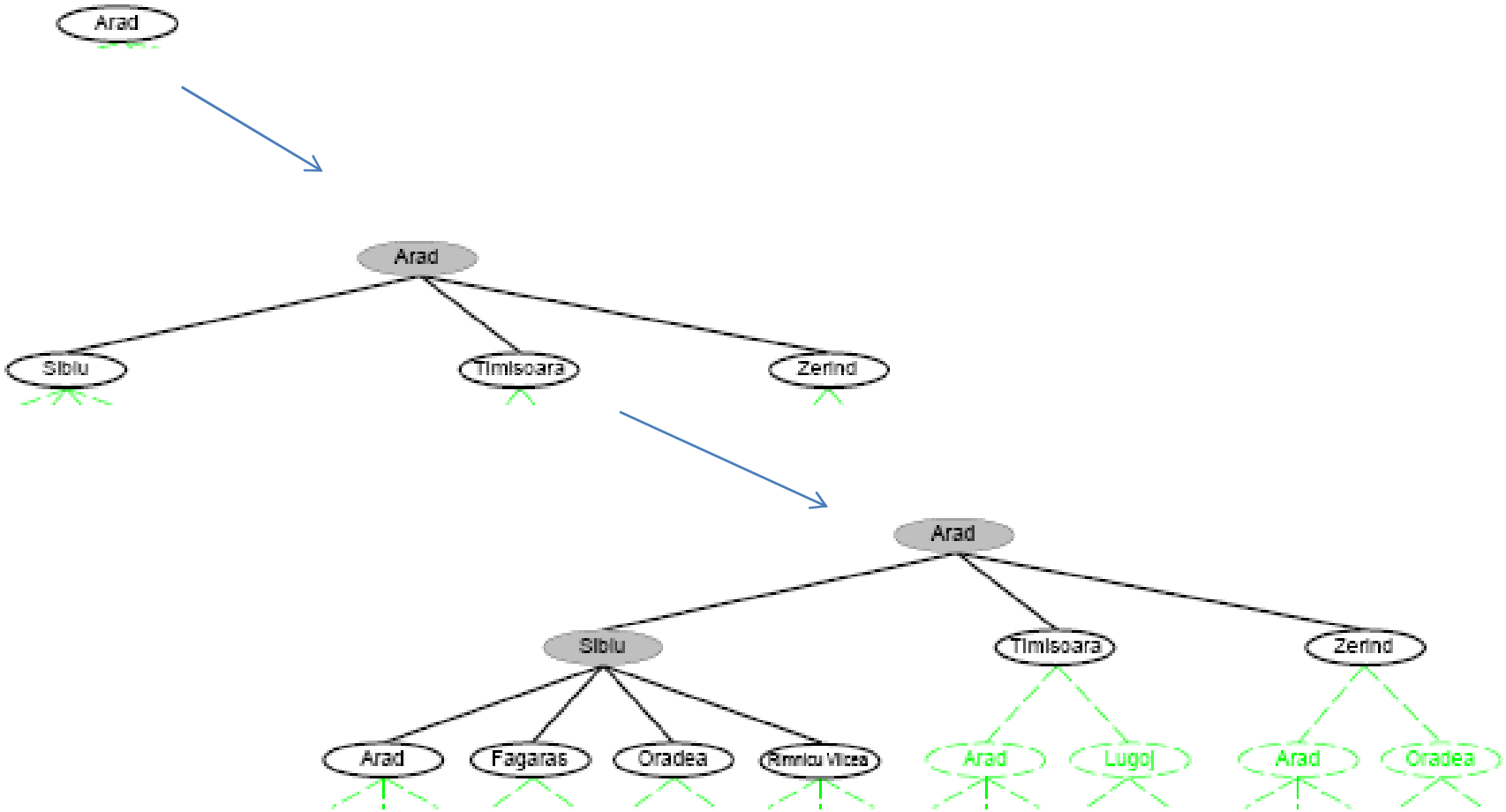
- Estados: cada estado está representado por una ubicación (p.ej. aeropuerto) y la hora actual
- Estado inicial: especificado por el problema
- Función sucesor: devuelve los estados que resultan de tomar cualquier vuelo programado desde la ubicación actual a otra, que salgan a la hora actual más el tiempo de tránsito en el aeropuerto
- Test objetivo: ¿se alcanza el destino a una cierta hora especificada?
- Costo del camino: costo en dinero, tiempo de espera, tiempo del vuelo, procedimientos de inmigración, etc.

Búsqueda de soluciones

- La resolución de problemas se hace mediante búsqueda a través del espacio de estados
- Las técnicas básicas utilizan un árbol de búsqueda explícito generado a partir del estado inicial y la función sucesor para generar nuevos estados (i.e. para expandir) el actual formando nuevos nodos

```
function TREE-SEARCH(problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

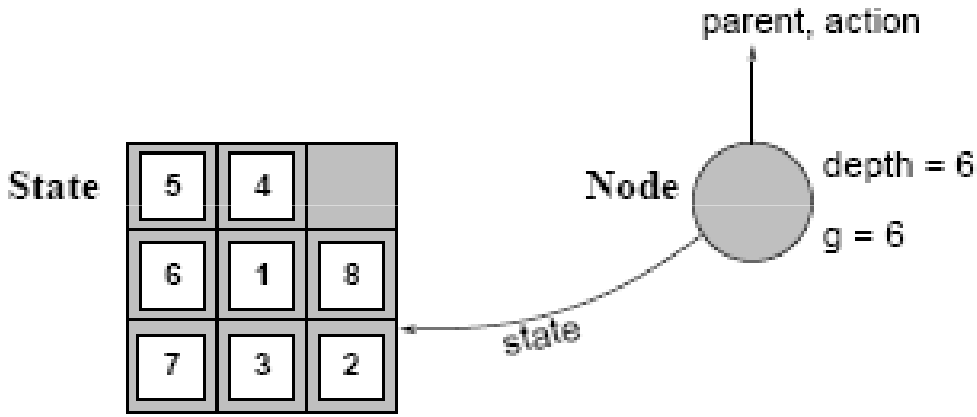
Ejemplo


Arad
Arad
Sibiu
Timisoara
Zerind
Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea

Estructura general de los nodos de un árbol de búsqueda

Formado por:

- Estado: el estado que corresponde con el nodo
- Nodo Padre: el nodo del árbol de donde se generó este nodo
- Acción: la acción aplicada al padre para generar el nodo
- Costo del camino: el costo desde el nodo conteniendo al estado inicial hasta el nodo actual
- Profundidad: el número de pasos a lo largo del camino desde el nodo inicial



Búsqueda general en árboles

```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE(node)) then return node
        fringe ← INSERTALL(EXPAND(node, problem), fringe)
```

```
function EXPAND( node, problem) returns a set of nodes
    successors ← the empty set
    for each action, result in SUCCESSOR-FN(problem, STATE[node]) do
        s ← a new NODE
        PARENT-NODE[s] ← node;  ACTION[s] ← action;  STATE[s] ← result
        PATH-COST[s] ← PATH-COST[node] + STEP-COST(node, action, s)
        DEPTH[s] ← DEPTH[node] + 1
        add s to successors
    return successors
```

Estrategia de búsqueda

- Una estrategia se define eligiendo el orden de expansión de nodos
- Las estrategias se evalúan de acuerdo a su
  - Completitud: ¿se encuentra la solución si esta existe?
  - Complejidad temporal: número de nodos generados/expandidos
  - Complejidad espacial: máximo número de nodos en memoria
  - Optimalidad: ¿siempre se encuentra la solución de más bajo costo?

Tipos de estrategias de búsqueda

- No Informadas ó a Ciegas:
  - Busca la primer solución sin importar que tan óptima sea
  - No detecta si se esta aproximando o alejando de la solución.
  - No es capaz de encontrar una solución aceptable en caso de que no exista o sea demasiado costoso encontrar la solución óptima
- Informadas ó Heurísticas:
  - Busca soluciones aceptables
  - Reduce el espacio de búsqueda y es capaz de determinar su proximidad a una solución y la calidad de la misma utilizando conocimiento a priori.

Tipos de estrategias de búsqueda

- Búsqueda ciega o no informada
  - Primero en anchura
  - Primero en profundidad
- Búsqueda informada
  - Primero el mejor
  - A*
  - Algoritmos de búsqueda local