Razonamiento y Planificación Automática

César Augusto Guzmán Álvarez

Doctor en Inteligencia Artificial

Tema 5 : Búsqueda offline

Sesión 1 / 2

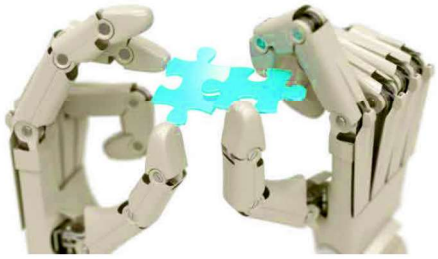
unir LA UNIVERSIDAD EN INTERNET

Universidad Internacional de La Rioja

Resumen – Tema anterior

Tema 4 : Razonamiento

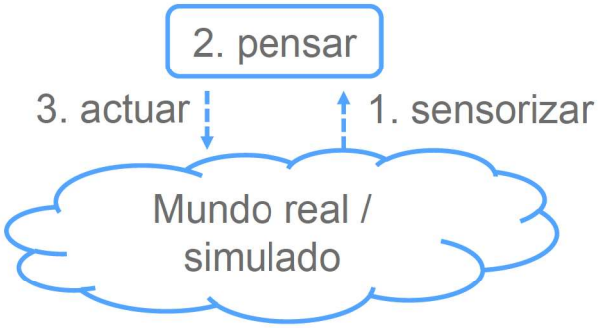
- Tipos de razonamiento
- Razonamiento lógico deductivo
- Razonamiento lógico inductivo
- Razonamiento lógico abductivo



Índice

Agentes basados en búsqueda

Definición de agente inteligente : Es cualquier sistema que de forma **autónoma** consigue una meta u objetivo por medio de un comportamiento **racional**.



Proceso de toma de decisiones

Agentes basados en búsqueda

¿Qué son los agentes basados en búsquedas?

- Mantienen un **modelo simbólico**
- **Modificar el estado del entorno**

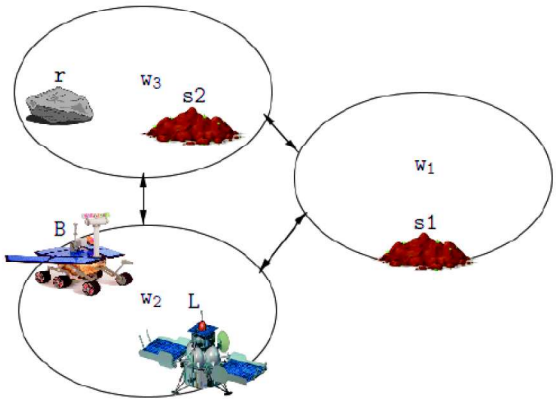


Figure 3.3: Initial state of the Mars domain single motivation scenario.

$$\Pi \begin{array}{l|l} a_1 & \text{(Navigate B } w_2\ w_1\text{)} \\ a_2 & \text{(Analyze B s1 } w_1\text{)} \\ a_3 & \text{(Communicate B s1 L } w_1\ w_2\text{)} \\ a_4 & \text{(Navigate B } w_1\ w_2\text{)} \end{array}$$

Goals

Comunicar análisis de s1
Localización de B en W2

Fuente : Gúzman Álvarez, C. A. (2019). *Reactive plan execution in multi-agent environments* (Doctoral dissertation).

Agentes basados en búsqueda

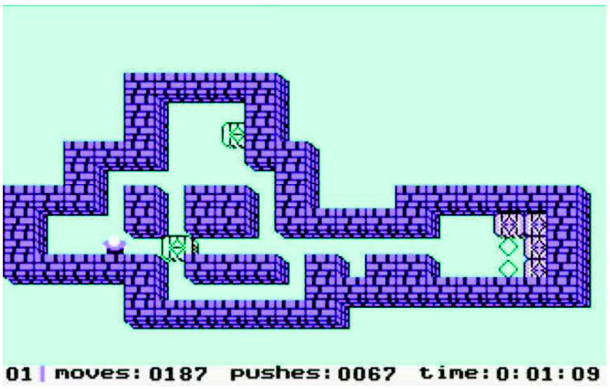
- Arquitectura Deliberativa :

- Arquitectura Reactivas :

Agentes basados en búsqueda

Mecanismos para resolver problemas

- Algoritmos específicos del problema
  - ✓ Método especifico resuelve el problema
  - ✓ Prever todos los escenarios posibles
  - ✓ Complejo para entornos reales.
- Algoritmos independientes del problema



As reported in [Junghanns and Schaeffer, 1998c], we implemented **IDA*** for Sokoban. We gave the algorithm a fixed node limit of 1 billion nodes for all experiments (varying from 1 to 3 hours of CPU time on a single 195 MHz processor of an SGI Origin 2000).

Fuente: Junghanns, A., & Schaeffer, J. (1999, July). Domain-dependent single-agent search enhancements. In *IJCAI* (pp. 570-577).

Agentes basados en búsqueda

Mecanismos para resolver problemas

- Algoritmos específicos del problema

  - ✓ Método especifico resuelve el problema
  - ✓ Prever todos los escenarios posibles
  - ✓ Complejo para entornos reales.

- **Algoritmos independientes del problema**

  - ✓ Algoritmo de búsqueda genérico
  - ✓ estado inicial
  - ✓ Operadores, que se instancian a acciones
  - ✓ Estado objetivo
  - ✓ Métrica o función objetivo (e. menos movimientos)

Agentes basados en búsqueda

Definir Problemas Búsqueda con estados

Espacio de estados

- Mundo
- Modelo simbólico o dominio
- Grafo

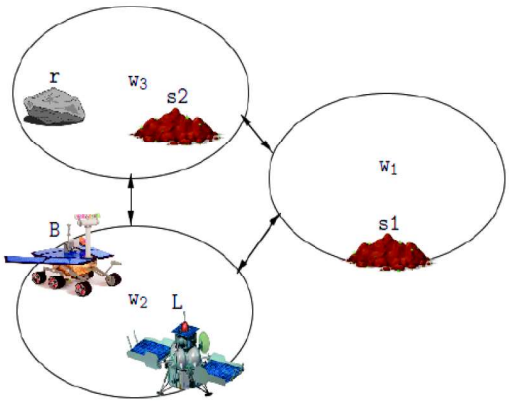


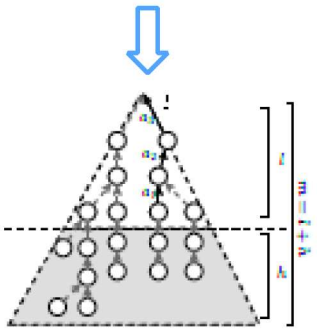
Figure 3.3: Initial state of the Mars domain single motivation scenario.

Problema del mundo real.

Fuente : Gúzman Álvarez, C. A. (2019). *Reactive plan execution in multi-agent environments* (Doctoral dissertation).

Estado:
Estado inicial
Estado final
Operadores (Move ?r ?w1 ?w2)
Posible plan

Dominio y problema



Grafo

Agentes basados en búsqueda

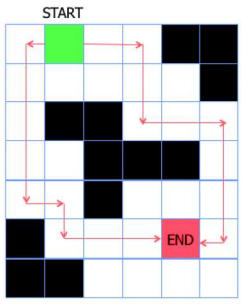
Definir Problemas Búsqueda con estados

Espacio de estados

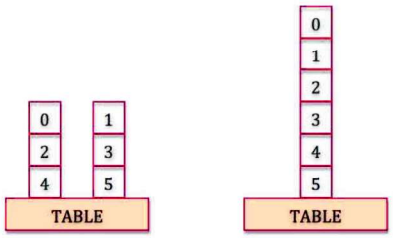
- Mundo
- Modelo simbólico o dominio
- Grafo

Búsqueda

- Independiente del problema
- Explorar espacio de estados
- Aplicando proceso de exploración



Short path planning problem. Source: C.J. Taylor, University of Pennsylvania



Blockworld problem.

Depots problem

Agentes basados en búsqueda

Definir Problemas Búsqueda con estados

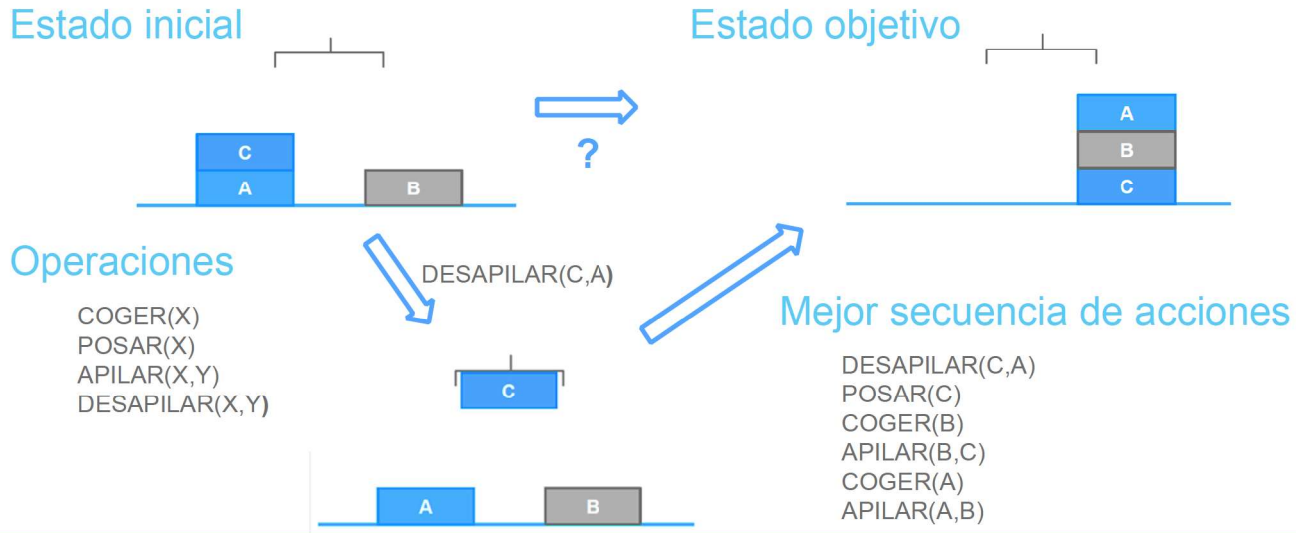
Espacio de estados	• Mundo • Modelo simbólico o dominio • Grafo
Búsqueda	• Independiente del problema • Explorar espacio de estados • Aplicando proceso de exploración
Objetivo	• Menor número de acciones • Menos tiempo

Agentes basados en búsqueda

BÚSQUEDA en INTELIGENCIA ARTIFICIAL

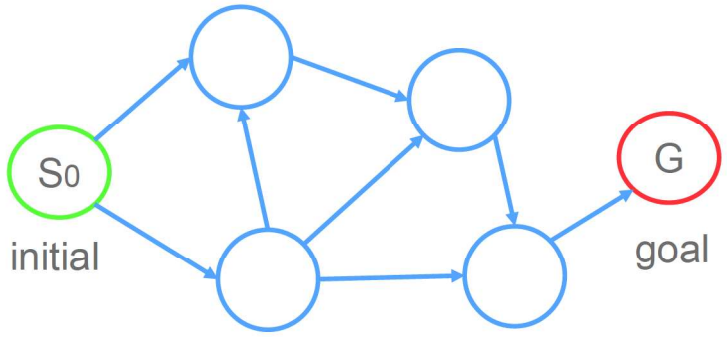
Aplicar una **estrategia de control** que permita encontrar un **camino desde el estado inicial al objetivo**, examinando las posibles secuencias de acciones y los estados que provocan, y seleccionando la mejor secuencia de acciones en base a un criterio.

- Ejemplo: Búsqueda en robótica «El mundo de los bloques»



Agentes basados en búsqueda

Definir Problemas Búsqueda con estados

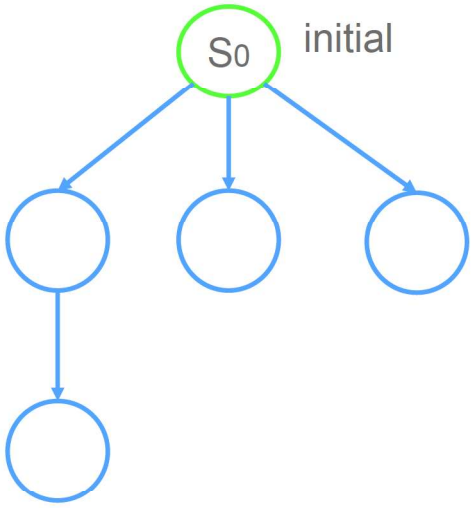


Conocimiento a priori del agente	
Estado inicial:	S_0
Estado objetivo:	G
Estados sucesores:	$\{S_1, S_2, \ldots, S_n\}$ $S_j = \text{result}(S_i, [a_i])$
Función de evaluación:	$G \subseteq S_i \mid S_i \in \{S1, S2, \ldots, Sn\}$
Coste de acción:	$\text{Coste}(\text{result}(S_i, [a_i]))$
Coste del plan:	$\text{Coste}(\text{result}(S_0, [a_i, \ldots, a_n,]))$

Agentes basados en búsqueda

Definir Problemas Búsqueda con estados

1. Elegir una hoja o nodo.
2. Verificar si es un estado objetivo.
3. Si no lo es, expandir este nodo
4. Ir al paso 1.



Agentes basados en búsqueda

Algoritmo general de búsqueda

Input: Estado inicial *S0*, Estado final *G*

```
1: colaAbierta ← {S0}
2: mientras colaAbierta ≠ ∅
3:     nodo ← extraer primero de colaAbierta
4:     si G ⊆ nodo entonces
5:         retornar camino a nodo
6:     fin si
7:     sucesores ← expandir(nodo)
8:     para cada sucesor ∈ sucesores hacer
9:         sucesor.padre ← nodo
10:        colaAbierta ← colaAbierta ∪ sucesor
11:    fin para
12: fin mientras
13: retorna plan vacío o problema sin solución
```

1. Elegir una hoja o nodo.
2. Verificar si es un estado objetivo.
3. Si no lo es, expandir este nodo
4. Ir al paso 1.

Agentes basados en búsqueda

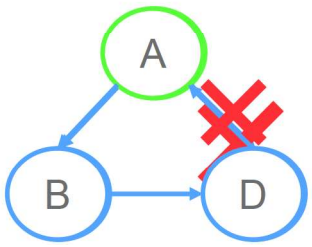
Algoritmo general de búsqueda

Input: Estado inicial *S0*, Estado final *G*

1: *colaAbierta* ← $\{S0\}$
2: mientras *colaAbierta* ≠ ∅
3: ***nodo*** ← extraer primero de *colaAbierta*
4: si *G* ⊆ ***nodo*** entonces
5: retornar camino a ***nodo***
6: fin si
7: *sucesores* ← expandir(***nodo***)
8: para cada sucesor ∈ sucesores hacer
9: *sucesor.padre* ← ***nodo***
10: *colaAbierta* ← *colaAbierta* ∪ *sucesor*
11: fin para
12: fin mientras
13: retorna plan vacío o problema sin solución

Nodos repetidos:

- Ignorarlo.
- Evitar ciclos simples.
- Evitar ciclos generales.
- Evitar todos los estados repetidos.



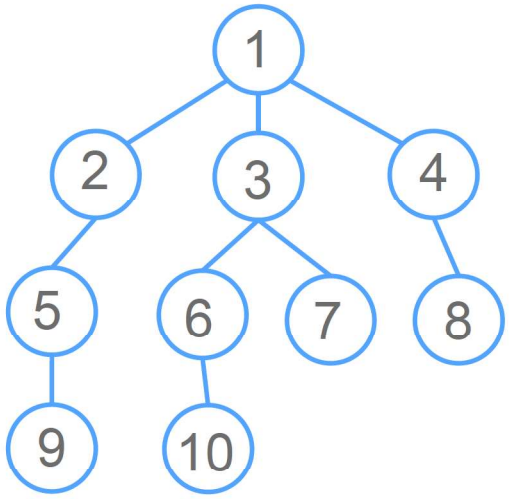
Agentes basados en búsqueda

Características de un algoritmo

Completitud	Optimalidad	Tiempo	Espacio
Encuentra solución si existe	Si hay varias soluciones encuentra la mejor	Tiempo en encontrar la solución	Memoria empleada para encontrar la solución

Búsqueda en amplitud o anchura

breadth first search (BFS)



- Algoritmo de búsqueda sin información
- Complejidad computacional y espacial:

Worst-case performance	$O(\|V\| + \|E\|) = O(b^d)$
Worst-case space complexity	$O(\|V\|) = O(b^d)$

- Completo y optimo.

Búsqueda en amplitud o anchura

breadth first search (BFS) - algoritmo

Input : Grafo, estado inicial S0, estado final G

```
1: definir colaAbierta como cola
2: marcar S0 como visitado
3: colaAbierta.add(S0)
4: mientras colaAbierta ≠ ∅
5:     nodo = colaAbierta.remover()
6:     si G ⊆ nodo entonces
7:         retornar camino a nodo
8:     fin si
9:     sucesores <- Grafo.hijos(nodo)
10:    para cada sucesor ∈ sucesores hacer
11:        si sucesor no esta marcado como visitado:
12:            marcar sucesor como visitado
13:            sucesor.padre = nodo
14:            colaAbierta.add(sucesor)
15:        fin si
16:    fin para
17: fin mientras
```

Características principales:

- No recursiva.
- Utiliza una cola (FIFO)
- Primero verifica si nodo es no procesado.
- Marcar como visitado :
  a) guardar en otro conjunto
  b) utilizando un atributo del nodo.
- Atributo padre es importante

Búsqueda en amplitud o anchura

breadth first search (BFS) - algoritmo

Input : Grafo, estado inicial S0, estado final G

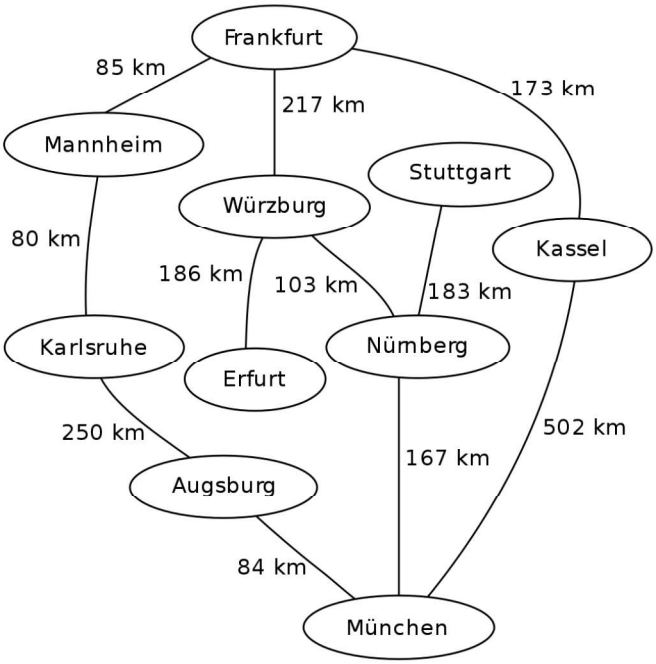
```
1: definir colaAbierta como cola
2: marcar S0 como visitado
3: colaAbierta.add(S0)
4: mientras colaAbierta ≠ ∅
5:     nodo = colaAbierta.remover()
6:     si G ⊆ nodo entonces
7:         retornar camino a nodo
8:     fin si
9:     sucesores <- Grafo.hijos(nodo)
10:    para cada sucesor ∈ sucesores hacer
11:        si sucesor no esta marcado como visitado:
12:            marcar sucesor como visitado
13:            sucesor.padre = nodo
14:            colaAbierta.add(sucesor)
15:        fin si
16:    fin para
17: fin mientras
```

Características principales:

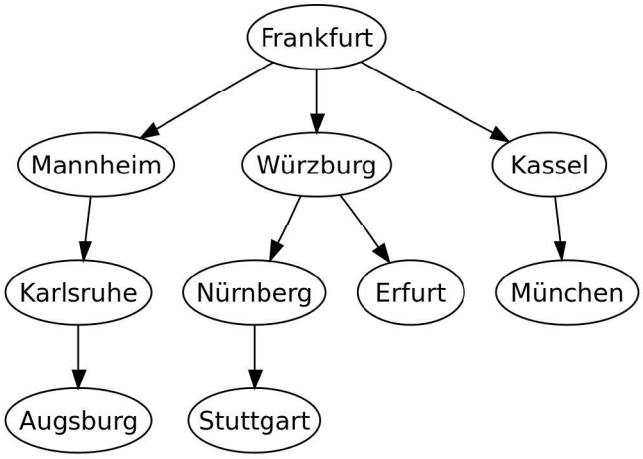
- No recursiva.
- Utiliza una cola (FIFO)
- Primero verifica si nodo es no procesado.
- Marcar como visitado :
  a) guardar en otro conjunto
  b) utilizando un atributo del nodo.
- Atributo padre es importante

Búsqueda en amplitud o anchura

breadth first search (BFS) – breadth first tree



An example map of Southern Germany with some connections between cities



The breadth-first tree obtained when running BFS on the given map and starting in Frankfurt

Fuente: https://en.wikipedia.org/wiki/Breadth-first_search

Gracias!