



DECSAI

Departamento de Ciencias de la Computación e I.A.

Universidad de Granada



Búsqueda en Inteligencia Artificial

© Fernando Berzal, berzal@acm.org

Búsqueda en I.A.



- Introducción
- Espacios de búsqueda
- Agentes de búsqueda
- Uso de información en el proceso de búsqueda
 - Búsqueda sin información (a ciegas)
 - Búsqueda con información (heurística)
- Estrategias de control
 - Estrategias irrevocables
 - Estrategias tentativas
- Características de los problemas de búsqueda
 - Problemas "ignorables" (monótonos y reversibles)
 - Problemas "no ignorables" o irrecuperables



Introducción



Problemas NP-difíciles

Definición informal:

Problemas que, para resolverlos de forma exacta, **requieren** la realización de una búsqueda en un espacio que es de tamaño exponencial.

- Nadie sabe como evitar esa búsqueda.
- No se espera que nadie lo consiga nunca...



Introducción



Problemas NP-difíciles

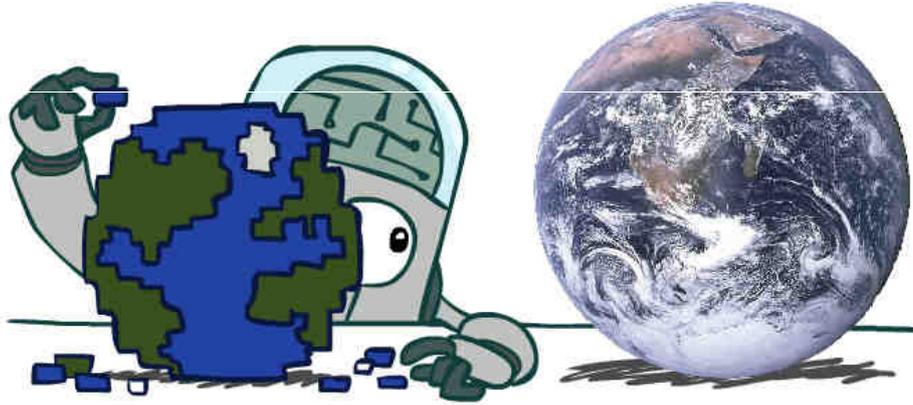
- Todos los problemas de los que se ocupa la I.A. son NP-difíciles (si existe un algoritmo rápido para un problema, difícilmente consideraremos que el problema requiere inteligencia).
- Si un problema es NP-difícil y tenemos un algoritmo que encuentra la solución de forma rápida y casi siempre correcta, podemos considerar que el algoritmo es "inteligente".
- La I.A. implica que la búsqueda está sujeta a errores.



Introducción



Para poder resolver el problema, debemos construir un modelo para basar nuestras decisiones en las consecuencias (hipotéticas) de nuestras acciones:



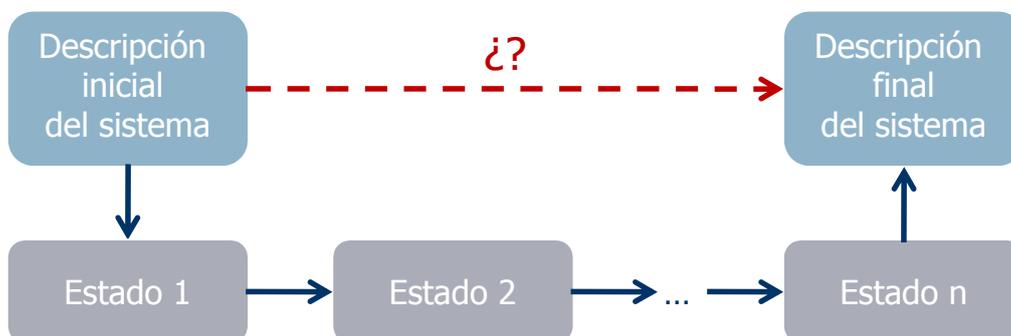
Berkeley CS188



Introducción



Problema típico en I.A.



→ representa la aplicación de algún operador o transformación del sistema.

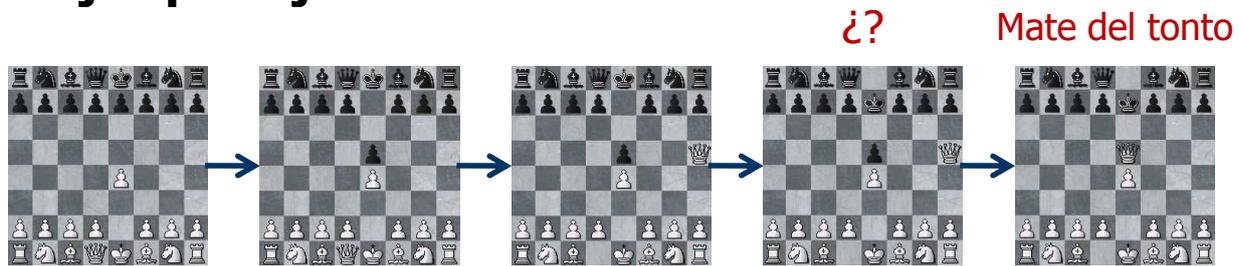
¿Cómo se selecciona, en cada momento, el operador que se debería aplicar?



Introducción



Ejemplo: Ajedrez



- Se dispone de varios operadores (movimientos legales).
- Se ha de elegir cuál es el más adecuado en cada momento (¿qué pieza movemos y a dónde?).
- No tiene sentido aplicar (ejecutar) el primer operador que sea válido, sino que se debe realizar un proceso previo de **búsqueda** del mejor operador, tras lo cual se procederá a su aplicación.

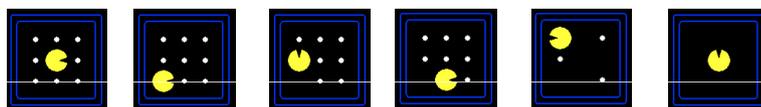


Introducción



Un problema de búsqueda en I.A. consta de:

- Un **espacio de estados**.



- Un conjunto de **operadores** (acciones, con costes).



- Un **estado inicial** (punto de partida de la búsqueda).
- Una **función objetivo** (comprueba si el estado actual corresponde a una solución del problema).



Espacios de búsqueda



- La búsqueda la realiza un programa (o agente).
- El espacio de búsqueda será un grafo dirigido en el que cada nodo representa un posible estado del sistema.

NOTA: Dependiendo del problema, cada nodo incluirá una descripción completa del sistema, o bien sólo las modificaciones necesarias para pasar de un nodo padre a su hijo.



Espacios de búsqueda



Búsqueda en un espacio de estados

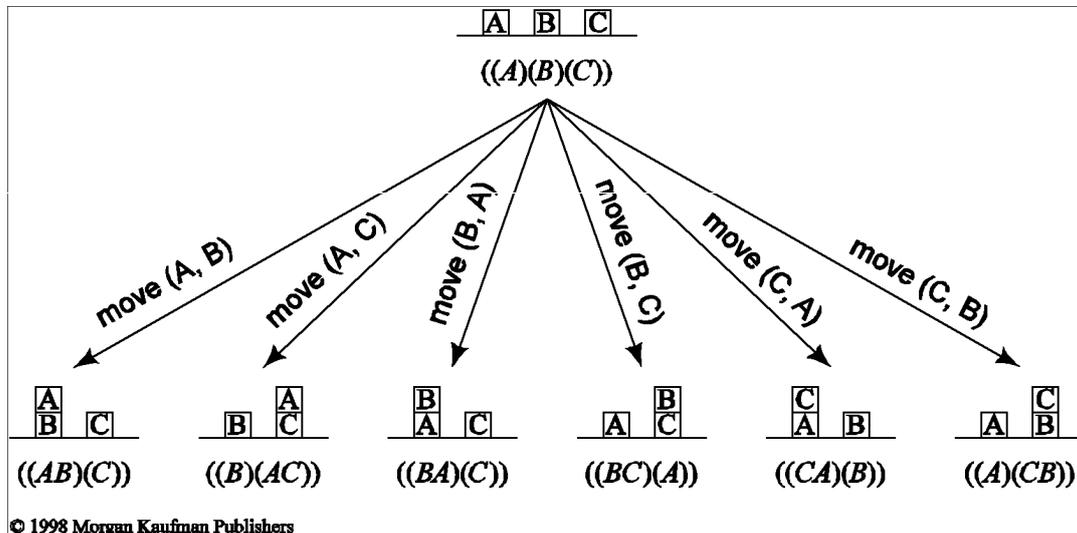
- **Espacio de estados:**
Representación del problema a través de las (posibles) acciones del agente.
- **Búsqueda en el espacio de estados:**
Resolución del problema mediante la proyección de las distintas acciones del agente.



Espacios de búsqueda



Posibles acciones del agente (operadores)



Espacios de búsqueda



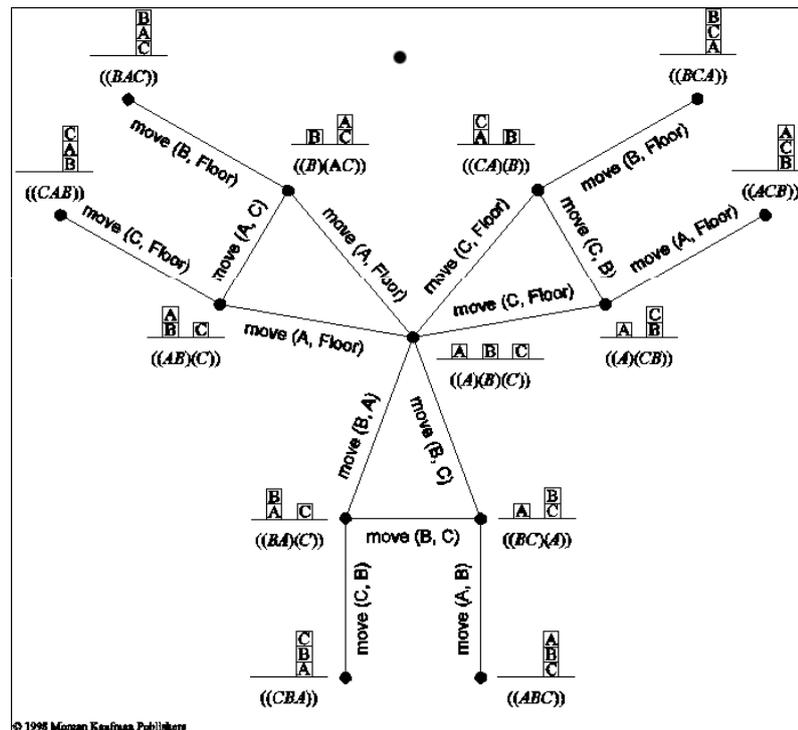
Búsqueda en un espacio de estados

Grafo del espacio de estados / Grafo implícito

- Representación matemática de un problema de búsqueda (nodos: estados; arcos: operadores): Grafo teórico que representa todas las posibles transformaciones del sistema aplicando todos los operadores posibles recursivamente.
- Debido a su complejidad exponencial, que requeriría una cantidad inviable de memoria y tiempo, el grafo del espacio de estados no puede generarse por completo.



Espacios de búsqueda



Espacios de búsqueda

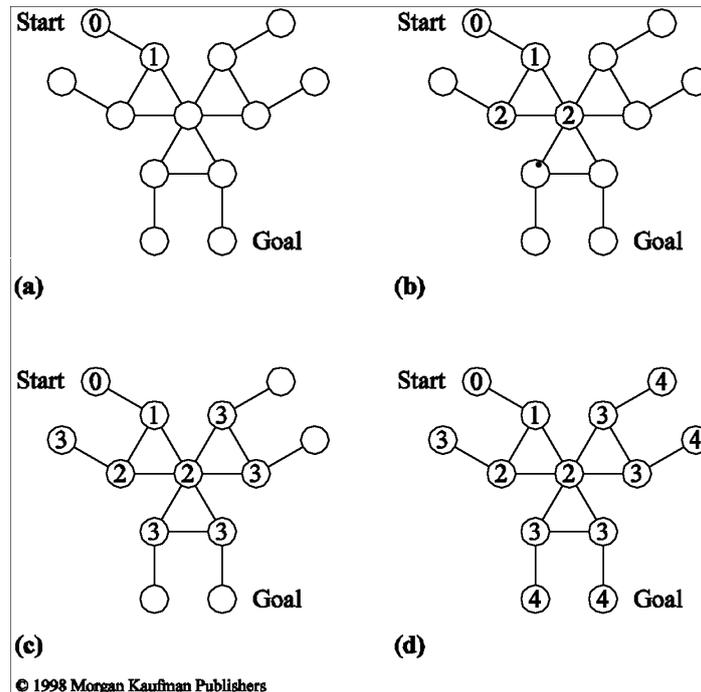


Búsqueda en un espacio de estados Árbol de búsqueda / Grafo explícito

- Debido a la complejidad exponencial del grafo implícito, se irá generando, paso a paso, una porción del grafo conforme avance el proceso de búsqueda.
- El **grafo explícito** es el subgrafo del grafo implícito que se va generando durante el proceso de búsqueda de una secuencia de operadores que resuelva nuestro problema (camino solución).
- Usualmente, en forma de **árbol**, de ahí su nombre.



Espacios de búsqueda



Espacios de búsqueda



Búsqueda en un espacio de estados Árbol de búsqueda / Grafo explícito

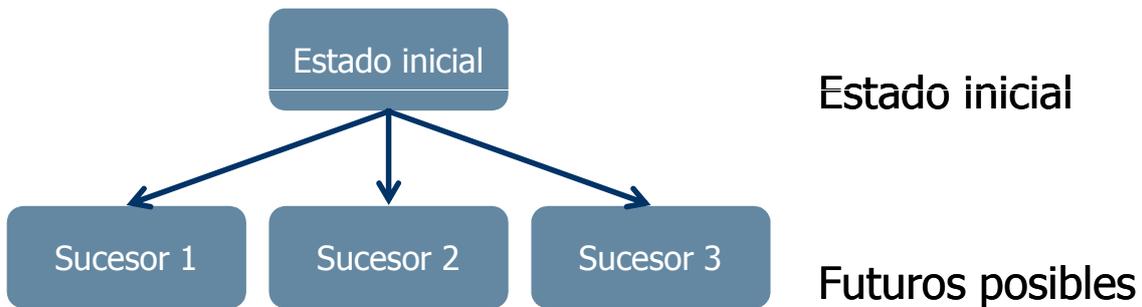
- Nodo raíz: Estado inicial.
- Hijos de un nodo: Posibles sucesores (nodos correspondientes a estados resultantes de la aplicación de un operador al nodo padre).
- Los nodos del árbol representan estados, pero corresponden a PLANES mediante los cuales se alcanzan dichos estados.



Espacios de búsqueda



Búsqueda en un espacio de estados **Árbol de búsqueda / Grafo explícito**



Espacios de búsqueda



Búsqueda en un espacio de estados **Árbol de búsqueda / Grafo explícito**

Condiciones de parada

- Se ha encontrado la solución.
- Se ha acabado el tiempo disponible.
- Se ha llegado a un nivel de profundidad determinado.



Espacios de búsqueda



Ejemplo: 8-puzzle



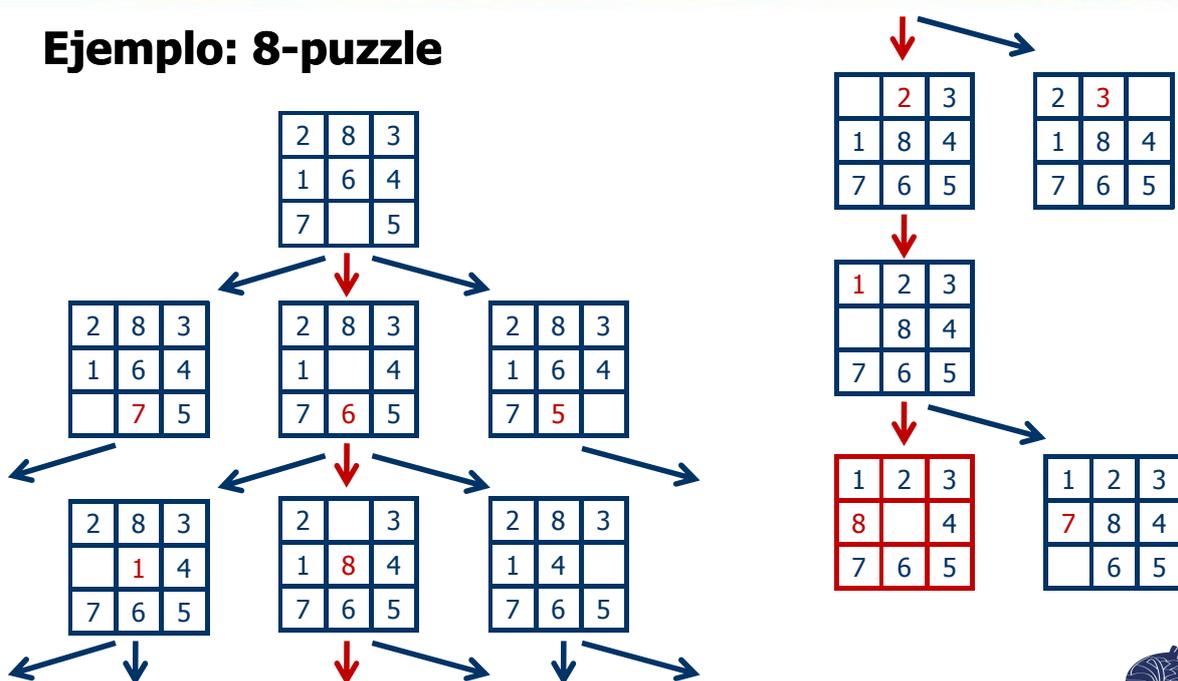
En el problema del 8-puzzle hay que encontrar un camino desde la configuración inicial hasta una determinada configuración final.



Espacios de búsqueda



Ejemplo: 8-puzzle



Espacios de búsqueda



Ejercicio: Problema del mono y los plátanos

Un mono está en la puerta de una habitación. En el centro de la habitación hay un plátano colgado del techo, pero no puede alcanzarlo desde el suelo. En la ventana de la habitación hay una caja, que el mono puede mover y a la que puede encaramarse para alcanzar el plátano. El mono puede realizar las siguientes acciones: desplazarse de la puerta al centro, del centro a la ventana y viceversa; empujar la caja a la vez que se desplaza; subirse y bajarse de la caja; coger el plátano.

El problema consiste en encontrar una secuencia de acciones que permita al mono coger el plátano.



Agentes de búsqueda



Según el tipo de problema, nos podemos encontrar con:

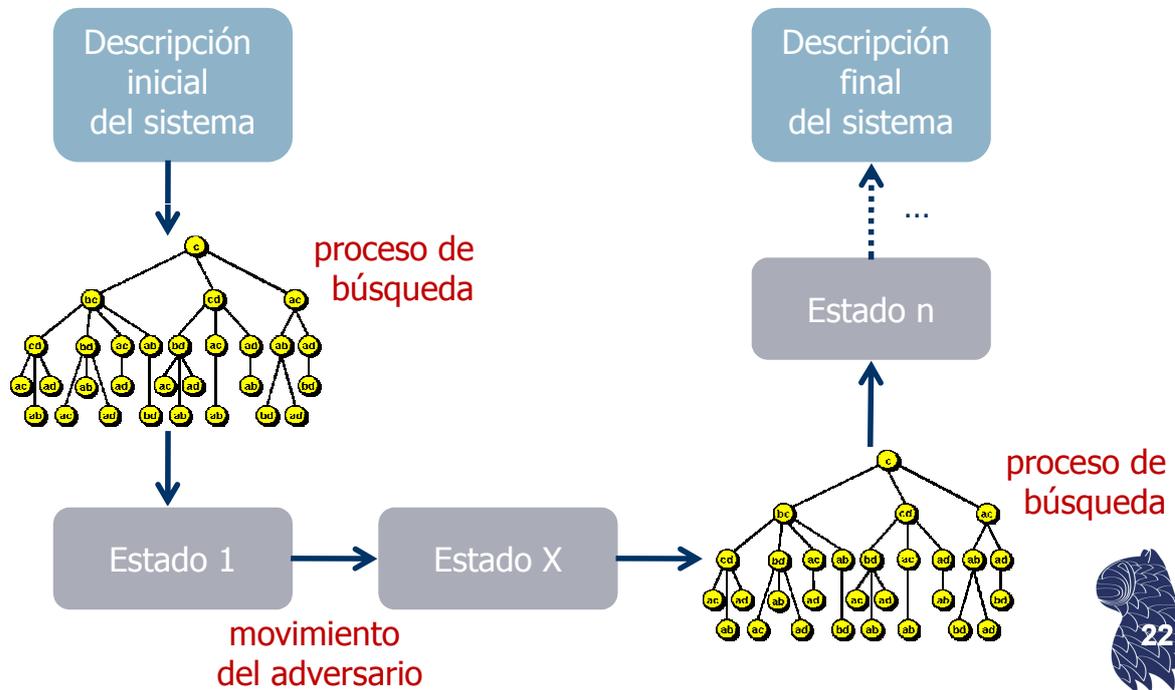
- **Agentes de búsqueda que devuelven un único operador**
vg. Juegos con adversario (como el ajedrez)
- **Agentes de búsqueda que devuelven una secuencia de operadores**
vg. Juegos sin adversario (como el 8-puzzle)
Sistemas de planificación
Sistemas expertos (con encadenamiento hacia adelante)



Agentes de búsqueda



... que devuelven un único operador, p.ej. ajedrez

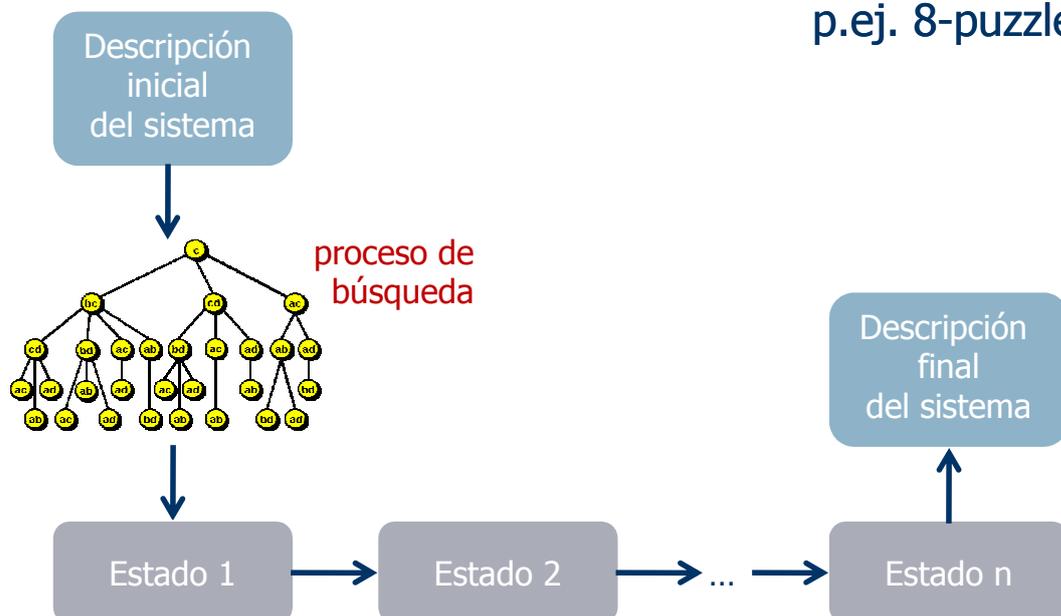


22

Agentes de búsqueda



... que devuelven una secuencia de operadores
p.ej. 8-puzzle



23

Uso de información



- Un problema puede tener varias soluciones, pero debido a la extensión del grafo implícito, no podemos explorarlo por completo, por lo que tampoco podemos buscar la mejor solución al problema (suponiendo algún criterio de optimalidad).
- Por eso, en muchos problemas de I.A. nos conformamos con buscar **soluciones aceptables** (cualquier camino a una solución lo suficientemente buena) y no soluciones óptimas (la mejor solución posible).

NOTA: Existen técnicas matemáticas/algorítmicas para resolver determinados tipos de problemas de optimización, p.ej. programación dinámica.



24

Uso de información



Heurísticas

Las heurísticas son criterios, métodos o principios para decidir cuál de entre varias acciones promete ser la mejor para alcanzar una determinada meta.

- En la generación del árbol de búsqueda, nos podemos guiar con heurísticas que nos den una indicación acerca de cómo de bueno o prometedor es un determinado estado u operador.

p.ej.

¿Qué pieza deberíamos mover en una partida de ajedrez?

¿Qué regla aplicamos en primer lugar al hacer un diagnóstico?



25

Uso de información



Heurísticas

- El uso de heurísticas nos permite convertir nuestra búsqueda de una solución en un **proceso guiado** de ensayo y error.

NOTA: Compárese el uso heurístico de información para guiar el proceso de búsqueda en Inteligencia Artificial con formalismos como la teoría de la decisión o la teoría de juegos, que en ocasiones nos permitirán determinar la decisión óptima si somos capaces de identificar todos los factores relevantes para el problema en cuestión (y las incertidumbres asociadas a ellos !!).

http://en.wikipedia.org/wiki/Decision_theory

http://en.wikipedia.org/wiki/Game_theory



Búsqueda sin información



- Sólo realizaremos una **búsqueda a ciegas** [blind search] cuando no exista información específica sobre el problema que nos ayude a determinar cuál es el mejor operador que se debería aplicar en cada momento o el mejor nodo por el que continuar la búsqueda.
- Se pueden utilizar distintos criterios para explorar el espacio de búsqueda, p.ej. LIFO (en profundidad) o FIFO (en anchura).



Búsqueda con información



- En problemas medianamente complejos, no obstante, tendremos que utilizar algún tipo de información para guiar nuestra búsqueda.

p.ej. Para generar el grafo completo del juego del ajedrez (10^{47} estados), generando 3 billones de nodos por segundo y sin restricciones de memoria, tardaríamos unos **10^{30} años** en resolver el problema, ¡ 10^{20} veces la edad estimada del universo!

<http://www.wolframalpha.com/input/?i=universe+age>



Búsqueda con información



Se pueden distinguir dos casos básicos:

- Información incluida en la descripción del propio conocimiento que tenemos del problema.
p.ej. Uso de prioridades en los S.E.B.R.
- Información especificada aparte de la descripción del conocimiento.

p.ej. Uso de una función heurística que evalúa la bondad de un estado del sistema:

$f(\text{estado}) \in \mathbf{R}$



Búsqueda con información



Ejemplo: 8-puzzle

Ya que conocemos el estado final deseado, podemos utilizar la siguiente función heurística:

$$f(\text{tablero}) = - \text{número de piezas mal colocadas con respecto al objetivo}$$

2	8	3
1	6	4
7		5

$$f(E) = -4$$



1	2	3
8		4
7	6	5

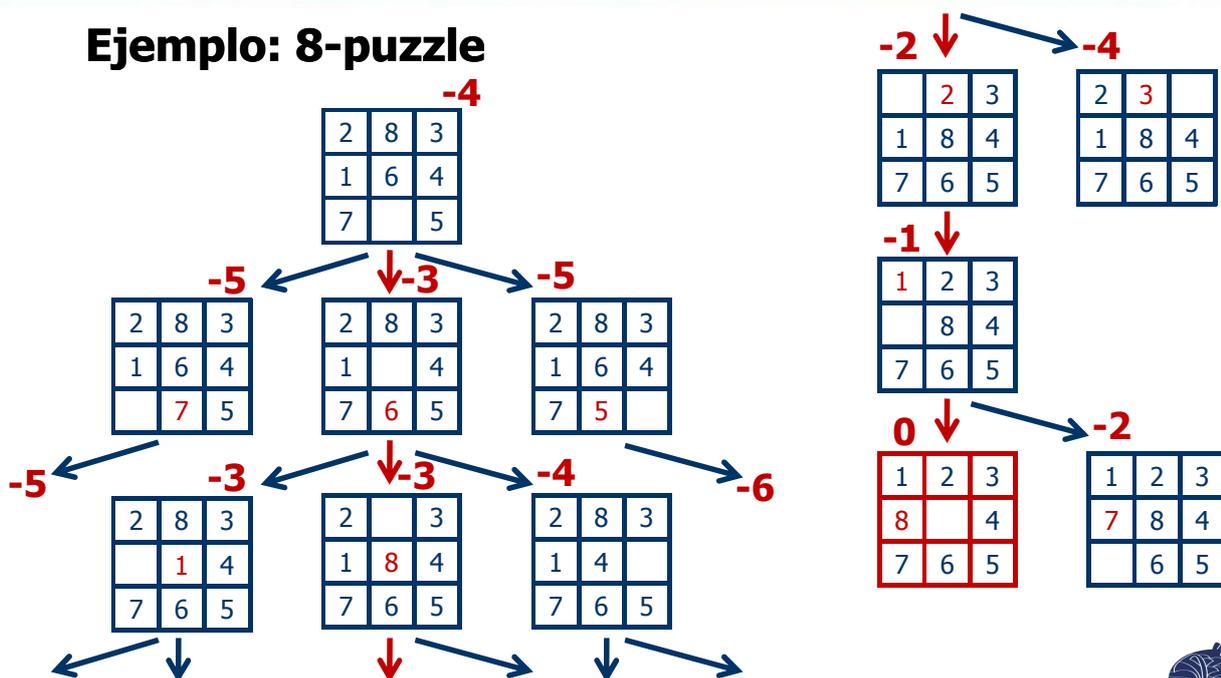
$$f(\text{OBJ}) = 0$$



Búsqueda con información



Ejemplo: 8-puzzle



Estrategias de control



El tipo de estrategia de control que utilice nuestro agente de búsqueda (irrevocable o tentativa) determinará la respuesta a las siguientes preguntas:

- ¿Cómo vamos generando el árbol de búsqueda en un proceso de búsqueda?
- ¿Guardamos todos los nodos generados?
- ¿Guardamos sólo el camino explorado hasta el último nodo generado?
- ¿Guardamos exclusivamente el último nodo?



Estrategias de control



Estrategias irrevocables

En cada momento, sólo mantenemos en memoria un único nodo, que incluye la descripción completa del sistema en ese momento:

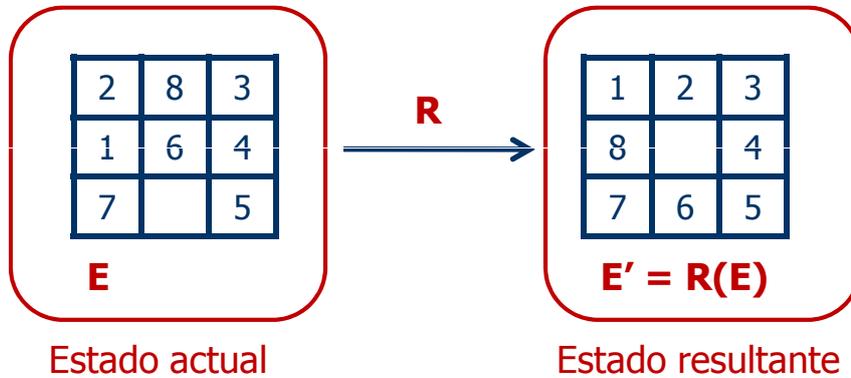
- Se selecciona un operador R .
- Se aplica sobre el estado del sistema E , para obtener un nuevo estado $E' = R(E)$.
- Se borra de memoria E y se sustituye por E' .



Estrategias de control



Estrategias irrevocables



Estrategias de control



Estrategias irrevocables

■ Búsqueda sin información

vg. CLIPS (estrategias greedy FIFO, LIFO o LEX)

■ Búsqueda con información

vg. CLIPS (conocimiento: prioridades de las reglas)
Ascensión de colinas (función heurística)



Estrategias de control



Estrategias tentativas de control

En memoria se guardan todos los estados o nodos generados hasta el momento, de forma que la búsqueda puede proseguir por cualquiera de ellos:

1. Seleccionar un estado E del grafo.
2. Seleccionar un operador R aplicable sobre E .
3. Aplicar R , para obtener un nuevo nodo $R(E)$.
4. Añadir el arco $E \rightarrow R(E)$ al grafo
5. Repetir el proceso.



Estrategias de control



Estrategias tentativas sin información

- Al no disponer de información que nos guíe en el proceso de búsqueda, seguimos criterios generales

vg. Exploración del grafo en profundidad
Exploración del grafo en anchura

- Podemos definir criterios de parada (p.ej. detener la generación cuando se haya llegado a un nivel de profundidad determinado o se haya consumido el tiempo disponible).



Estrategias de control



Estrategias tentativas con información Exploración primero el mejor

Se explora exhaustivamente el grafo utilizando una función heurística para determinar el orden en que se exploran los nodos:

1. Seleccionar el nodo E del grafo que tenga mayor valor para la función heurística.
2. Seleccionar un operador R aplicable sobre E .
3. Aplicar R , para obtener un nuevo nodo $R(E)$.
4. Añadir el arco $E \rightarrow R(E)$ al grafo
5. Repetir el proceso.



Características del problema



¿Cuáles son las características de un problema que obligan a utilizar una determinada estrategia?

- Problemas "ignorables".
 - Problemas "ignorables" monótonos.
 - Problemas "ignorables" reversibles.
- Problemas "no ignorables" o irrecuperables.



Características del problema



Problemas "ignorables"

Un problema es ignorable cuando, si se puede alcanzar el objetivo desde un estado E , también puede alcanzarse desde cualquier otro estado $R(E)$, donde R es un operador arbitrario.

En definitiva, no corremos ningún riesgo en la aplicación de cualquier operador.



Características del problema



Problemas "ignorables" monótonos

Cuando la aplicación de cualquier operador se limita a añadir más hechos, sin modificar los ya existentes.

Este tipo de problemas son los tratados en la lógica clásica (que es monótona).

EJEMPLO

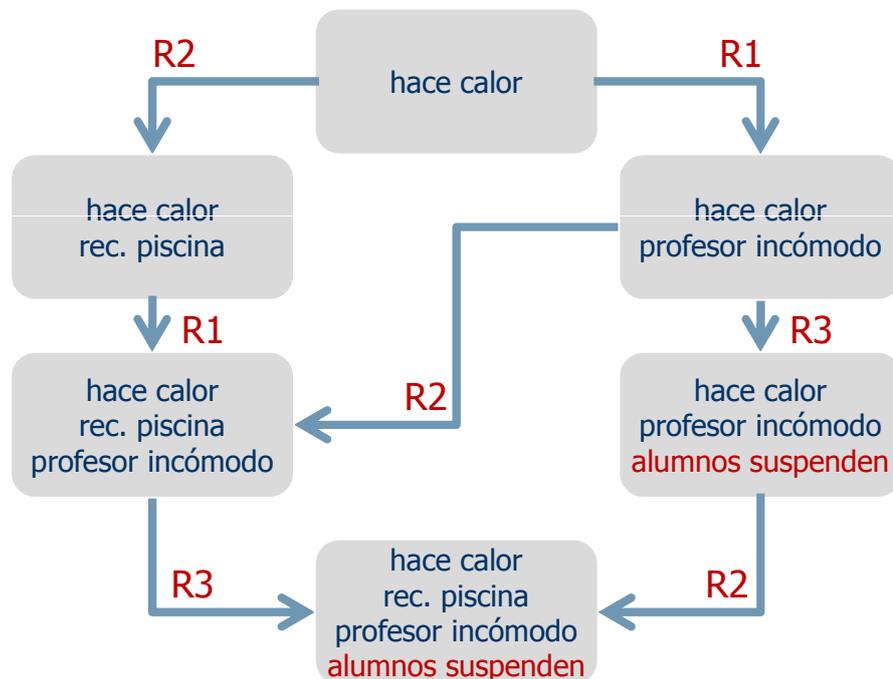
- R1. Si hace calor, entonces el profesor no está cómodo
- R2. Si hace calor, entonces recomendar piscina
- R3. Si el profesor no está cómodo, entonces los alumnos suspenden



Características del problema



Problemas "ignorables" monótonos



Características del problema



Problemas "ignorables" monótonos

- El ejemplo anterior ilustra la "ignorabilidad" de un problema: siempre que sea posible, se llegará a la solución (aunque tal vez sea necesario aplicar más reglas de las estrictamente necesarias).
- Al ser el problema ignorable, el grafo de búsqueda puede explorarse utilizando una **estrategia irrevocable** (así lo haremos, sobre todo, si el coste de representación de los estados en memoria es elevado).

NOTA: Obviamente, siempre se puede explorar con una estrategia de control retroactiva.



Características del problema



Problemas "ignorables" reversibles

Cuando las acciones de los operadores son reversibles;
es decir, \forall estado E , \forall operador R , \exists operador R^{-1} tal que

$$R^{-1}(R(E))=E$$

EJEMPLOS

- 8-puzzle.
- Planificación de movimientos de un robot con STRIPS.



Características del problema



Problemas "ignorables" reversibles

- Se les puede aplicar una **estrategia irrevocable**:

Si se decide no proseguir la búsqueda por un estado E' ,
generado a partir de otro estado E aplicando $R(E)$,
siempre se puede aplicar R^{-1} a E' para obtener de
nuevo E y buscar otro operador aplicable a E .

- Si el coste computacional de la aplicación de los
operadores es alto y no hay muchas restricciones de
memoria, entonces es mejor utilizar una **estrategia
retroactiva**.



Características del problema



Problemas "ignorables" reversibles



Operación reversible



Operador inverso (o vuelta atrás)



Operación alternativa



Características del problema



Problemas "no ignorables" o irrecuperables

Cuando la aplicación de un operador puede generar un estado desde el cual no sea posible llegar a un estado objetivo.

EJEMPLOS

- Control de la adición de sustancias químicas en el proceso realizado por una planta industrial.
- Juegos con adversario, como el ajedrez.



Características del problema



Problemas "no ignorables" o irrecuperables

EJEMPLO: Sustituciones simbólicas en una cadena

R1: C → DL

R2: C → BM

R3: B → MM

R4: Z → BBM

Estado inicial: $E_0 = \{C B Z\}$

Estado final: Cadena sólo con emes

- Si aplicamos R1 es imposible llegar a un estado final, por lo que el problema es irrecuperable.
- Si quitásemos R1, entonces el problema sí sería "ignorable".

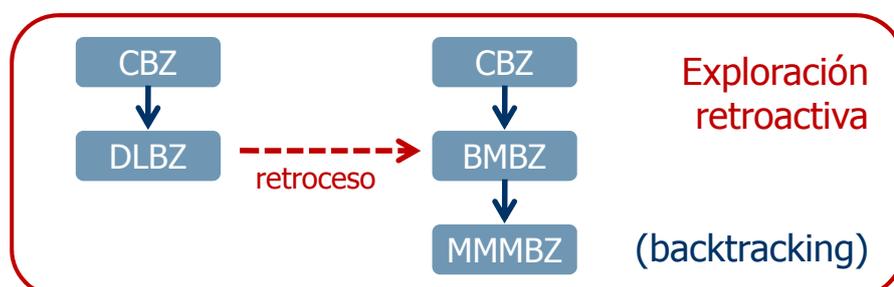
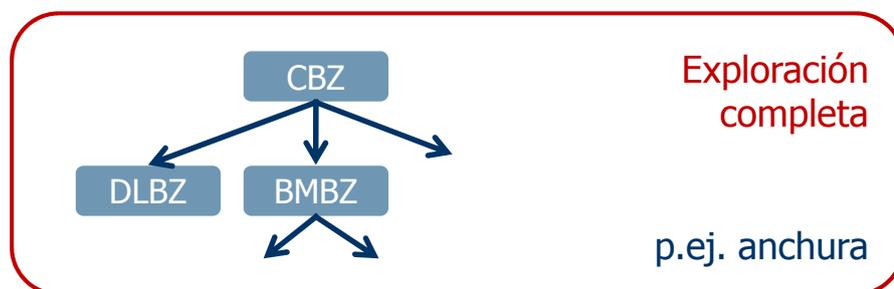


Características del problema



Problemas "no ignorables" o irrecuperables

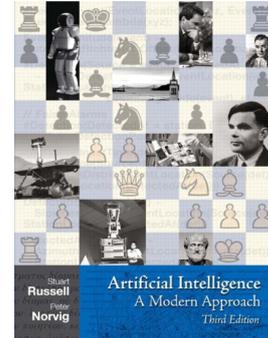
Lo usual será utilizar una estrategia tentativa:



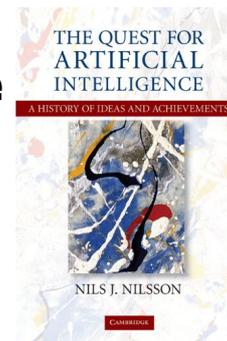
Bibliografía



- Stuart Russell & Peter Norvig:
**Artificial Intelligence:
A Modern Approach**
Prentice-Hall, 3rd edition, 2009
ISBN 0136042597



- Nils J. Nilsson
The Quest for Artificial Intelligence
Cambridge University Press, 2009
ISBN 0521122937



Bibliografía



Bibliografía complementaria

- Elaine Rich & Kevin Knight:
Artificial Intelligence.
McGraw-Hill, 1991.
- Patrick Henry Winston:
Artificial Intelligence.
Addison-Wesley, 1992.
- Nils J. Nilsson:
Principles of Artificial Intelligence.
Morgan Kaufmann, 1986.

