Inteligencia Artificial e Ingeniería del Conocimiento

- Revisión de la I.A clásica
  - ✓ **Representación y búsqueda**
  - ✓ Sistemas expertos
  - ✓ METOLOGIA KADS
- "Nuevos" enfoques de la I. A.
  - ✓ Agentes Inteligentes
  - ✓ Aprendizaje
  - ✓ Lógicas multivaluadas
  - ✓ Computación evolutiva

I. A. Clásica. Revisión

- Planteamiento Clásico
  - ✓ Representación del conocimiento
    - ✓ Planteamientos basados en la lógica
    - ✓ Planteamientos estructurados
      - Redes Semánticas
      - Marcos
  - ✓ **Búsqueda de solución**
    - ✓ Algoritmos de búsqueda

Problema de búsqueda de solución. Introducción

- Representación de problemas
  - ✓ Inteligencia Artificial vs. Computación clásica
  - ✓ Formulación de problemas
  - ✓ Sistemas de producción
    - ✓ Definición
    - ✓ Ejemplos
- Estrategias de búsqueda
  - ✓ Clasificación
    - ✓ Irrevocables
    - ✓ Tentativas
      - Backtracking
      - Exploración de grafos
  - ✓ Aplicaciones

Representación de problemas.

- Objetivo
  - ✓ Diferenciar los métodos computacionales clásicos y los de la I.A
    - ✓ APLICABILIDAD DE LOS MÉTODOS
  - ✓ Establecer una separación de los componentes de la computación
    - ✓ Datos
    - ✓ Operaciones
    - ✓ Control
- Definición de un formalismo computacional
  - ✓ SISTEMAS DE PRODUCCIÓN

Diferenciación de métodos computacionales I

- Métodos clásicos (directos)
  - Implementción de los razonamientos realizados por el analista en un formato preestablecido por el lenguaje de programación
  - El proceso de solución permanecerá rígido dentro del programa
  - Modificaciones en el enunciado son previos a la implementación
  - Ventajas
    - El programa generado es eficiente (velocidad de ejecución, espacio de almacenamiento)
    - La implementación es generalmente sencilla
  - Inconvenientes
    - Fragilidad, Necesidad de un enunciado exacto, Proceso de solución oscuro

Diferenciación de métodos computacionales II

- Métodos de la I.A
  - Implementación en un ordenador no sólo los resultados del razonamiento sino del propio proceso de razonar (Procedimientos de búsqueda de solución)
  - Permite variaciones en el enunciado, secuencia para encontrar la solución variable
  - Inconvenientes
    - Eficiencia disminuida
    - Tarea de análisis más compleja
  - Aplicabilidad
    - Procedimientos de solución no conocidos
    - Planteamientos variables de los problemas

Espacio de estados

- Representación de problemas
  - ✓ Pasos a realizar en la formulación de un problema mediante técnicas de I. A
    - ✓ Definición de un ambiente del problema
      - ESPACIO DE ESTADOS
        - Ejemplo: Conjuntos de FBD del lenguaje formal
    - ✓ Configuración (estado) inicial
      - Ejemplo: Axiomas
    - ✓ Estados objetivo
      - Ejemplo: Conjunto de FBD que contenga a la fórmula objetivo
    - ✓ Operaciones sobre el espacio de estados
      - Ejemplo: Método general de resolución
  - ✓ PROBLEMA: Exploxión combinatoria de posibilidades

Sistemas de producción

- Representación de problemas. SISTEMAS DE PRODUCCION
  - ✓ Definición
    - ✓ Formalismo de computación que establece una separación de los componentes en
      - Base de datos global
        - Es la estructura central de datos. En ella tendremos la descripción de forma única del estado del sistema objetivo del problema a través del espacio de estados
      - Reglas de producción
        - Operan sobre la base de datos global. Cada regla tiene una precondición. Si se cumple se tendrá como resultado la modificación de la base de datos global. Dos condiciones:
          - Todas las reglas pueden acceder a la base de datos global
          - Las reglas no pueden llamarse las unas a las otras.
      - Sistema de control. PROBLEMA DE BÚSQUEDA DE SOLUCION

Sistemas de producción. SISTEMA DE CONTROL

- Mecanismo de selección de aplicación de regla de producción en cada instante
- Objetivo
  - Buscar la "MEJOR" secuencia de aplicación de reglas que permite que la base de datos verifique la condición objetivo
- Implementa una estrategia de búsqueda de solución
  - Elemento diferenciador con los métodos directos de computación

```
PROCEDIMIENTO PRODUCCION
1       DATOS <-- Base de datos inicial
2       until  DATOS satisface la condición de
        terminación    do
3       begin
4           SELECT alguna regla R aplicable a DATOS
5           DATOS <-- El resultado de aplicar R a
            DATOS
6       end
```


Sistemas de producción. Ejemplo I (I)

- Problema de los baldes
  - Enunciado
    - "Supongamos dos baldes inicialmente vacíos de 6 y 8 litros respectivamente. Disponemos de un grifo pero los baldes no disponen de ninguna marca. El objetivo es llenar el balde de 8 litros exactamente por la mitad"
  - Espacio de estados
    - (x , y) de modo que: 0<=x<=6 , 0<=y<=8
  - Estado inicial
    - (0, 0)
  - Estado final
    - (_, 4)
  - Conjunto de reglas de producción

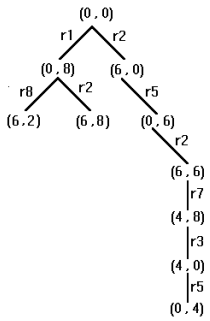
Sistemas de producción. Ejemplo I (II)

- Problema de los baldes
  - ✓ Base de datos global
    - ✓ Un elemento (x,y) del espacio de estados
  - ✓ Reglas de producción

R1.-	y<8	y=8	Llenar el segundo balde
R2.-	x<6	x=6	Llenar el primer balde
R3.-	0<y	y=0	Vaciar el segundo balde
R4.-	0<x	x=0	Vaciar el primer balde
R5.-	(8-y)>x ; x>0	y=y+x ; x=0	Descarga el primero en el segundo
R6.-	(6-x)>y ; y>0	x=x+y ; y= 0	Descarga el segundo en el primero
R7.-	(8-y)<x	y=8;x=x-(8-y)	Llenar el segundo con el primero
R8.-	(6-x)<y	x=6;y=y-(6-x)	Llenar el primero con el segundo

Sistemas de producción. Ejemplo I (III)

- Problema de los baldes
  - ✓ Resultado tras la utilización de la estrategia de búsqueda que constituye el sistema de control



Sistemas de producción. Ejemplo II (I)

- 8-puzzle
  - ✓ Enunciado
    - ✓ "Se trata del problema consituido por 8 fichas y un hueco distribuido en 9 posiciones, (cuadrado 3 x 3) de forma que sólo pueden moverse aquellas fichas (un máximo de cuatro) que están alrededor del espacio libre. Se trata de a partir de una posición dada llegar a otra final"
  - ✓ Espacio de estados
    - ✓ Matrices cuadradas (3 x 3) de números enteros entre 0 (casilla vacía) y 8

6	2	5	a(1,1)=6	a(1,2)=2	a(1,3)=5
3	0	4	a(2,1)=3	a(2,2)=0	a(2,3)=4
1	7	8	a(3,1)=1	a(3,2)=7	a(3,3)=8

Sistemas de producción. Ejemplo II (II)

- 8-puzzle
  - ✓ Reglas de producción

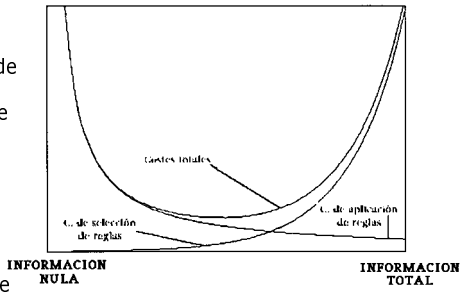
	PRECONDICION	RESULTADO	COMENTARIO
R1	a(i,j)=0 ,, i>1	a'(i-1,j)=0;a'(i,j)=a(i-1,j)	Hueco hacia arriba
R2	a(i,j)=0 ,, j<3	a'(i,j+1)=0;a'(i,j)=a(i,j+1)	Hueco a la derecha
R3	a(i,j)=0 ,, j>1	a'(i,j-1)=0;a'(i,j)=a(i,j-1)	Hueco a la izquierda
R4	a(i,j)=0 ,, i<3	a'(i+1,j)=0;a'(i,j)=a(i+1,j)	Hueco hacia abajo

Sistemas de producción. Clasifiación

- Clasificación
  - Sistemas de producción hacia adelante.
    - Se busca obtener la secuencia de aplicación de reglas de producción que aplicadas desde un estado inicial permiten llegar al estado final.
  - Sistemas de producción hacia atrás
    - Parten de la definición del estado objetivo buscando la secuencia de reglas que permita llegar hacia los estados iniciales.
  - Sistemas de producción conmutativos. Son aquellos en los cuales el proceso de solución no depende del orden de aplicación de las reglas

Sistemas de producción. El sistema de control

- Sistema de control. Utilización de información
  - Costes de computación
    - Costes de aplicación de reglas
      - Asociado a la evaluación de la precondición y a la modificación de la base de datos
    - Costes de selección de reglas
      - Evaluación de la información disponible sobre el problema (considerando la base de datos) para la selección de la reglas de producción.

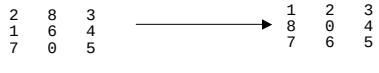


Estrategias de búsqueda. Clasificaciones

- Nivel de información
  - Informadas
  - No informadas
- Procedimiento de selección de reglas
  - Irrevocables
    - La regla seleccionada que se aplica y lo es sin ninguna reconsideración posterior
      - Existe seguridad de que la regla es la mejor
      - La aplicación de una regla no evita llegar a la solución
  - Tentativas
    - Una regla aunque sea utilizada se toman medidas con objeto de poder retornar a este punto del proceso y aplicar otra distinta. Se subdivide a su vez en dos clases
      - Estrategias retroactivas (Backtracking)
      - Estrategias de exploración de grafos

Estrategia irrevocable I

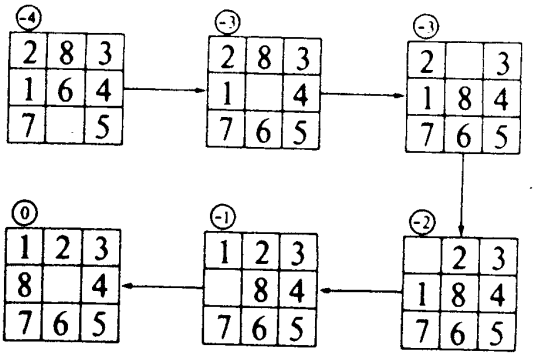
- Aplicación sin reconsideración posterior. Aplicabilidad
  - Siempre posible seleccionar la regla adecuada en cada caso
    - INFORMACION HEURISTICA
      - Utilización de una función de evaluación cuyo valor se intenta maximizar
      - No supone conocer la solución global del problema
  - La aplicación de una reglas no sea perjudicial para el proceso de solución del problema
- Ejemplo: 8-puzzle



f=-(Nº de casillas descolocadas)

Estrategia irrevocable II

- Ejemplo: 8 puzzle.



Estrategias tentativas I

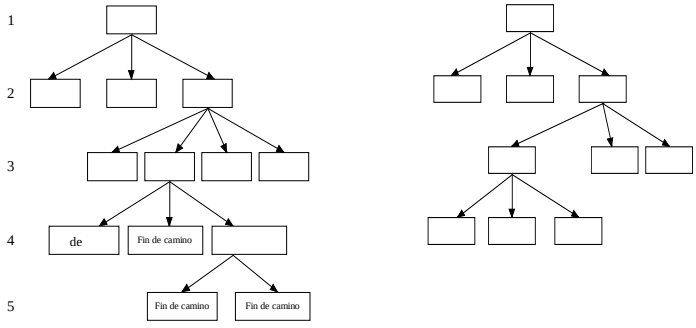
- La selección de una regla y su aplicación no excluye que cuando se crea necesario se pueda retornar al punto inicial con objeto de seleccionar otra regla
- Estrategia tentativa. BACKTRACKING
  - ✓ Cuando se aplica una regla y si no conduce a una solución, los pasos son **olvidados** y se selecciona otra regla de las aplicables
  - ✓ Permite manejar información **heurística**
  - ✓ Condiciones de backtrack
    - Estados repetidos
    - Estado sin salida (No existen reglas aplicables, ¿otras causas?
  - ✓ Variaciones
    - Backtrack múltiple
    - Longitud de solución elevada

Estrategias tentativas. BACKTRACKING I

1. if TERM(DATOS), return NADA; *TERM es un predicado verdadero para argumentos que satisfagan la condición de terminación. Tras la terminación con éxito, se devuelve la lista vacía NADA.*
2. if SINSALIDA(DATOS), return FALLO; *SINSALIDA es un predicado verdadero para argumentos de los que se sabe que no pueden conducir a una solución. En este caso, el procedimiento devuelve el símbolo FALLO.*
3. REGLAS<--APLIREGL(DATOS); *APLIREGL es una función que determina las reglas aplicables a su argumento y las ordena (ya sea arbitrariamente o de acuerdo con su mérito heurístico).*
4. CICLO: if NOHAY(REGLAS), return FALLO; *Si no hay mas reglas aplicables, el procedimiento falla.*
5. R<--PRIMER(REGLAS); *se selecciona la mejor de las reglas aplicables.*
6. REGLAS <--SUPR(REGLAS); *la lista de reglas aplicables se acorta suprimiendo en ella la regla seleccionada.*
7. RDATOS <--R(DATOS); *se produce una nueva base de datos aplicando la regla R.*
8. CAMINO<--BACKTRACK(RDATOS); *el procedimiento BACKTRACK es llamado recursivamente pasándole la nueva base de datos.*
9. if CAMINO = FALLO, go CICLO; *si la llamada recursiva falla, intentar la aplicación de otra regla.*
10. return CONS(R,CAMINO); *si no falló, pasar la lista de reglas que tuvieron éxito añadiendo R delante de ella.*

Estrategias tentativas. BACKTRACKING II

- Ejemplo de aplicación

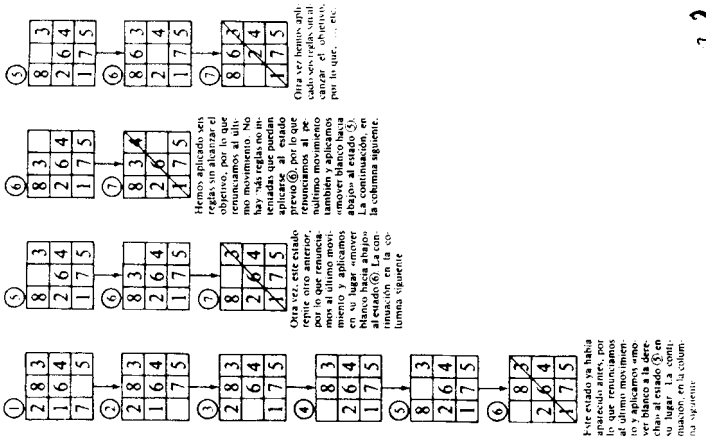


Estrategias tentativas. BACKTRACKING2 I

1. DATOS <--PRIMER(LISTABD); *LISTABD es una lista de bases de datos producidas, en un camino que lleva hacia atrás hasta la inicial; DATOS es la que se produjo más recientemente.*
2. if MIEMBRO(DATOS,SUPR(LISTABD)), return FALLO; *El procedimiento falla si DATOS había sido obtenida ya anteriormente en ese camino.*
3. if TERM(DATOS), return NADA
4. if SINSALIDA(DATOS), return FALLO
5. if LONGITUD(LISTABD) > LIMITE, return FALLO; *El procedimiento falla si se han aplicado demasiadas reglas. LIMITE es una variable global especificada antes de que sea llamado el procedimiento.*
6. REGLAS <--APLIREGL(DATOS)
7. CICLO: if NOHAY(REGLAS), return FALLO
8. R <--PRIMER(REGLAS)
9. REGLAS <--SUPR(REGLAS)
10. RDATOS <--R(DATOS)
11. RLISTABD <--CONS(RDATOS,LISTABD); *la lista de bases de datos obtenidas hasta ese punto se amplía añadiéndole RDATOS.*
12. CAMINO <--BACKTRACK1(RL/STABD)
13. if CAMINO = FALLO, go CICLO
14. return CONS(R,CAMINO)

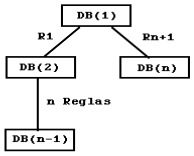
Estrategias tentativas. BACKTRACKING2 II

- Ejemplo 8-puzzle



Estrategias tentativas II

- Estrategias tentativas
  - ✓ Backtracking puede desestimar "caminos" (secuencia de aplicación de reglas) que contengan la solución



- Solución:
  - ✓ Manejar una estructura de datos en la que se mantengan todas las aplicaciones de reglas realizadas.
    - ✓ ALGORITMO DE EXPLORACIÓN DE GRAFOS

Estrategias tentativas. Exploración de grafos I

- Definiciones
  - ✓ Grafo: Conjunto de Nodos unidos por un conjunto de arcos
  - ✓ Arbol: Grafo donde el número máximo de antecesores es 1
- Aplicación en I.A
  - ✓ Constituye el sistema de control del sistema de producción
    - ✓ Nodo: Base de datos
    - ✓ Arco: Aplicación de regla de producción
    - ✓ Camino: Secuencia de aplicación de reglas
    - ✓ Arbol: Permite mantener un único camino entre dos nodos
      - Nodo padre: Nodo sin antecesor. Es la Base de Datos Inicial
      - Nodos Punta: Nodos sin antecesores. El objetivo es que este conjunto contenga a uno que verifique la condición de terminación

Estrategias tentativas. Exploración de grafos II

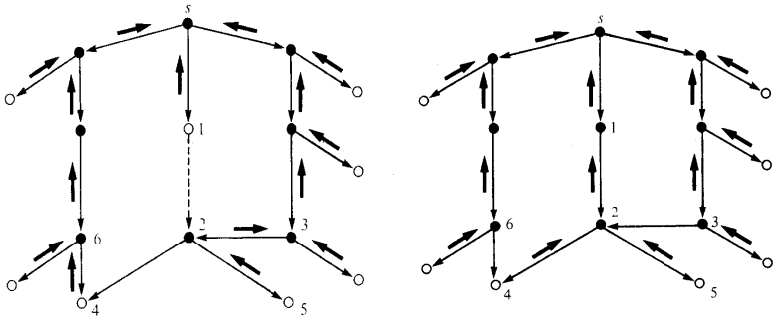
- Objetivo
  - ✓ Generar un grafo que contenga la secuencia de aplicación de reglas que constituyen la solución al problema
  - ✓ El grafo debe contener un árbol
    - ✓ Para cada nodo se selecciona su mejor antecesor
    - ✓ No contiene ciclos
    - ✓ La reconstrucción del camino de solución es inmediata
- Aplicabilidad
  - ✓ Espacios de búsqueda grandes
  - ✓ Mayor complejidad que el backtracking
    - ✓ Problemas con mínimos locales

Estrategias tentativas. Exploración de grafos III

1 Crear un *grafo de exploración G* que consista exclusivamente en el nodo inicial *s*. Iniciar con *s* una lista llamada *ABIERTOS*.
2 Crear una lista llamada *CERRADOS* que inicialmente estará vacía.
3 *CICLO*: **si** *ABIERTOS* está vacía, salida con **fallo**.
4 Seleccionar el primer nodo de la lista *ABIERTOS*, suprimirlo de ella e incluirlo en *CERRADOS*. Llamar *n* a este nodo.
5 **Si** *n* es un nodo objetivo, salida con **éxito**, dando la solución obtenida construyendo un camino, por medio de los apuntadores, desde *n* hasta *s* en *G*. (Los apuntadores se establecen en el paso 7).
6 Expandir el nodo *n*, generando el conjunto *M* de sus sucesores que no sean a la vez ascendientes de *n*. Incorporar estos miembros de *M*, como sucesores de *n*, en *G*.
7 Establecer un apuntador a *n* desde aquellos miembros de *M* que no estaban ya incluidos en *ABIERTOS* o *CERRADOS*. Añadir estos miembros de *M* a *ABIERTOS*. Para cada miembro de *M* que ya figurase en *ABIERTOS* o *CERRADOS*, decidir si se modifican o no sus apuntadores, dirigiéndolos a *n* (como se explica posteriormente). Para cada miembro de M que estuviese ya en *CERRADOS*, decidir, para cada uno de sus descendientes en *G*, si se modifican o no sus apuntadores (Ver texto).
8 Reordenar la lista *ABIERTOS* con arreglo a cualquier esquema arbitrariamente adoptado o de acuerdo con su mérito heurístico.
9 Ir a *CICLO*

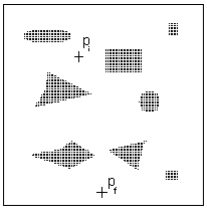
Estrategias tentativas. Exploración de grafos IV

- Generación de árboles

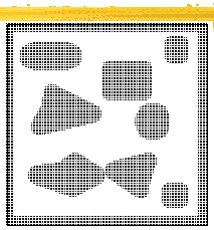


Est. búsqueda. Ejemplos I

- Robótica

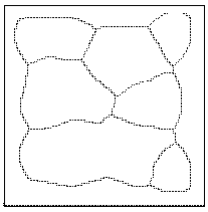


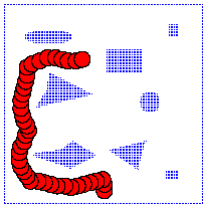
Espacio de trabajo



Espacio de las configuraciones

Espacio de estados



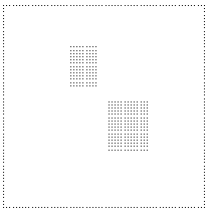


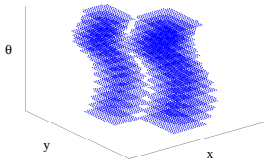
Solucion con backtracking

Est. búsqueda. Ejemplos II

- Robótica móvil. Traslación y giro

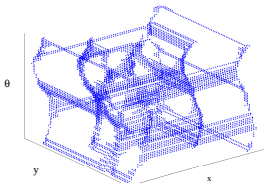
Espacio de trabajo

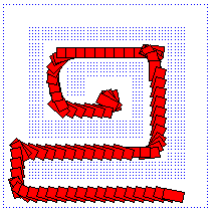




Espacio de las configuraciones

Espacio de estados





Solucion con exploración de grafos

Est. búsqueda. Utilización de información I

- Heurística
  - ✓ Información sobre un problema que se aporta a la estrategia de búsqueda del sistema de control para su resolución
  - ✓ Se implementa como una función $KB \xrightarrow{f} \Re$
  - ✓ Aplicación
    - ✓ Estrategias irrevocables
      - Se selecciona la regla que produce un mayor incremento
    - ✓ Estrategias tentativas
      - Backtracking
        - Ordenación del conjunto de reglas aplicables
        - Permite reducir el número de "backtrack´s"
      - Exploración de grafos
        - Ordenación del conjunto abiertos

Est. búsqueda. Utilización de información II

- Algoritmos de exploración de grafos
  - ✓ Estrategias no informadas
    - ✓ Búsqueda primero-profundidad
      - Se ordenan primero los nodos con mayor profundidad
        - Se pretende avanzar de forma rápida
        - Soluciones con longitud "no-optima" en el menor tiempo posible
        - Estrategia "ambiciosa"
    - ✓ Búsqueda primero-amplitud
      - Se ordenan primero los nodos con menor profundidad
        - Se pretenden soluciones de longitud optima
        - Estrategia "conservadora"
  - ✓ Estrategias informadas: Algoritmo A.

Est. búsqueda. Utilización de información III

- Algoritmo A
  - ✓ Algoritmo de exploración de grafos que utiliza para la ordenación del conjunto ABIERTOS una función de la forma

$$f(n) = g(n) + h(n) \quad estimación \quad de \quad f^*(n) = g^*(n) + h^*(n)$$

    - ✓ Para cada nodo $\mathbf{n}$, se define la función de evaluación $f(\mathbf{n})$ como la **estimación** de la suma del costo mínimo del nodo $\mathbf{s}$ hasta el nodo $\mathbf{n}$ más el costo mínimo del nodo $\mathbf{n}$ hasta un nodo objetivo $\mathbf{t}$. Se trata de una **estimación** del costo mínimo pasando por el nodo $\mathbf{n}$.
      - $k(\mathbf{n_i},\mathbf{n_j})$ es el costo del mínimo camino entre dos nodos $\mathbf{n_i}$ y $\mathbf{n_j}$. Esta función no estará definida para dos nodos entre los cuales no exista un camino.
      - $h^*(\mathbf{n})$ es el mínimo $k(\mathbf{n},\mathbf{t_i})$ para el conjunto de nodos objetivo $\{\mathbf{t_i}\}$
      - $g^*(\mathbf{n})$ es $k(\mathbf{s},\mathbf{n})$

Algoritmo A

- Estimaciones
  - $g(\mathbf{n})$ Suma de los costes de aplicación de reglas siguiendo los apuntadores de $\mathbf{s}$ a $\mathbf{n}$

    $g(\mathbf{n}) \geq g^*(\mathbf{n})$
  - $h(\mathbf{n})$ se utilizará la información heurística que se tiene sobre el dominio de cada problema concreto.
- Ejemplo
  - Búsqueda primero-amplitud

    $h(\mathbf{n})=0$ y $g(\mathbf{n})=d(\mathbf{n})$ siendo $d(\mathbf{n})$ la profundidad del nodo
- Algoritmo A*
  - Algoritmo A en el que $h(\mathbf{n}) \leq h^*(\mathbf{n})$ para todos los nodos $\mathbf{n}$
  - El algoritmo A* encontrará el camino solución óptimo, si existe éste
    - El algoritmo A* es admisible

Algoritmo A*. Admisibilidad I

- Demostración
  - 1º. El algoritmo finaliza para grafos finitos
    - Finaliza en paso 3 (ABIERTOS vacio) o 5 (Final)
    - En cada paso
      - 1 nodo de ABIERTOS se elimina y se añade un conjunto finito
    - Por ser finito
      - Finalización
      - Agotamiento de ABIERTOS
  - 2º. El algoritmo finaliza para grafos infinitos si existe un camino de solución
    - Se demuestra por reducción al absurdo
    - Contradicción: El valor de f(n) crece de forma ilimitada y siempre hay un nodo con valor acotado

Algoritmo A*. Admisibilidad II

- Demostración
  - ✓ 2º El algoritmo finaliza para grafos infinitos si existe un camino de solución
    - ✓ $d^*(\mathbf{n})$ Longitud del camino más corto de **s** a **n**
      - Entonces: $g^*(\mathbf{n}) \geq d^*(\mathbf{n})$e $\quad g(\mathbf{n}) \geq d^*(\mathbf{n})$e $\quad h(\mathbf{n}) \geq 0$
        - $f(\mathbf{n}) \geq d^*(\mathbf{n})$e
      - Aunque se seleccione el de menor valor el valor crecera cuando el algoritmo no finaliza (NO ACOTADO)
    - ✓ Existe en el conjunto ABIERTOS un nodo **n** que verifica
      - $f(\mathbf{n}) \leq f^*(\mathbf{s})$
      - Sea $(\mathbf{n_o},\mathbf{n_1},\ldots,\mathbf{n_k})$ donde $\mathbf{n_o}=\mathbf{s}$ y $\mathbf{n_k}$ es un nodo objetivo formando un camino óptimo
      - Existe n' en ABIERTOS del camino tal que
        - $f(\mathbf{n}')=g^*(\mathbf{n}')+h(\mathbf{n}') \leq g^*(\mathbf{n}')+h^*(\mathbf{n}')=f^*(\mathbf{n}')=f^*(\mathbf{s})$ (ACOTADO)

Algoritmo A*. Admisibilidad III

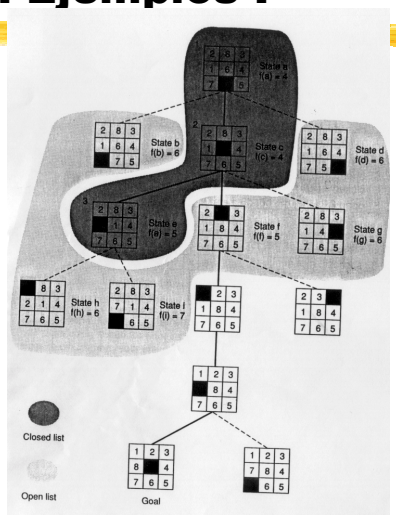
- Demostración
  - ✓ 3º. Todo nodo en ABIERTOS, verificando que $f(n) \leq f^*(s)$ es expandido por un algoritmo A*
    - ✓ Se supone que el algoritmo no expande un nodo **n'** con $f(\mathbf{n}') \leq f^*(\mathbf{s})$
      - A* finaliza en un nodo **t** y por tanto
        - $f(\mathbf{t}) \geq f^*(\mathbf{s}) \geq f(\mathbf{n}')$ (CONTRADICCION)
  - ✓ 4º El algoritmo A* es admisible
    - ✓ Está demostrado que A* encontrará un camino del nodo origen a un nodo objetivo **t**
    - ✓ Supongamos que encuentra un camino no óptimo $f(\mathbf{t})=g(\mathbf{t})>f^*(s)$
    - ✓ Existe en abiertos un nodo **n'** verificando que $f(\mathbf{n}') \leq f^*(\mathbf{s}) < f(\mathbf{t})$. En este punto el algoritmo A* habría escogido este nodo para su expansión en vez del nodo **t**, en contradicción con la hipótesis de que A* habría terminado

Algoritmo A*. Comparación

- Se tratará de comparar dos versiones de A*, denominadas A_1 y A_2 respectivamente.
  - Sean la siguientes sus funciones de evaluación:
    - $f_1(n)=g_1(\mathbf{n})+h_1(\mathbf{n})$ para A_1
    - $f_2(\mathbf{n})=g_2(\mathbf{n})+h_2(\mathbf{n})$ para A_2
    - $h_2(\mathbf{n})>h_1(\mathbf{n})$
  - Se demuestra que todos los nodos expandidos por A_2 es también expandido por A_1
  - Se dice que A2 está mejor informado que A1
    - Se puede llegar a perder la característica de admisibilidad

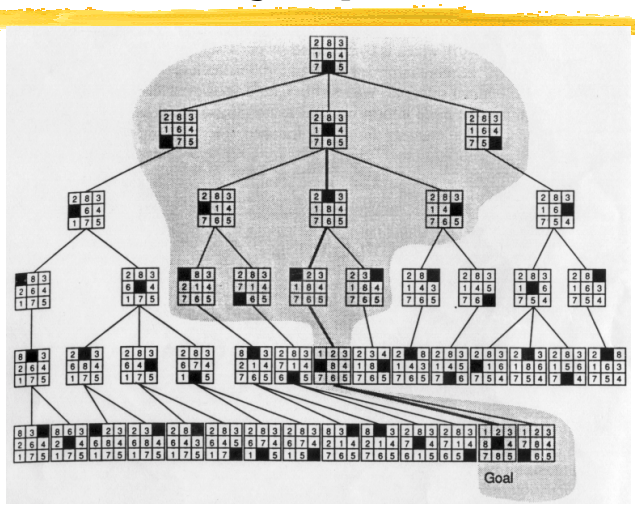
Algoritmo A*. Ejemplos I

- 8-puzzle. Proceso de solución



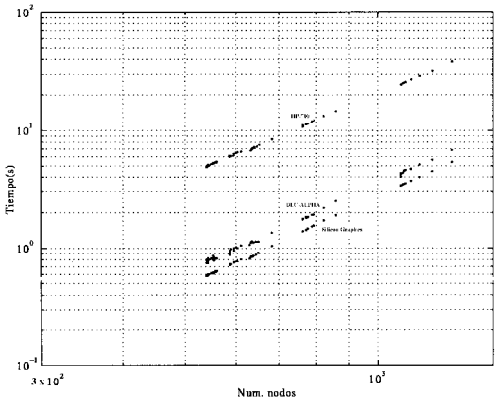
Algoritmo A*. Ejemplos II

- Comparativa de estrategias



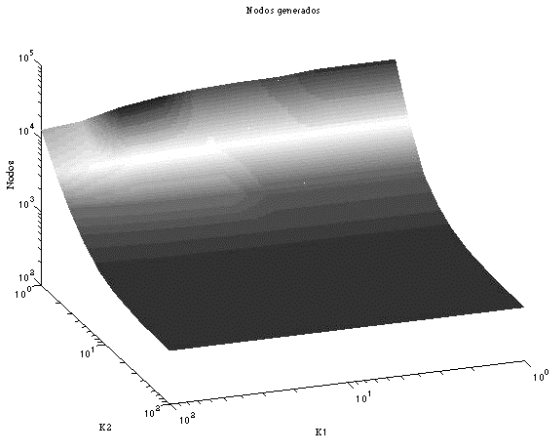
Algoritmo A*. Ejemplos III

- Robótica. Planificación de caminos.
  - ✓ Coste computacional. Aplicación de reglas



Algoritmo A*. Ejemplos IV

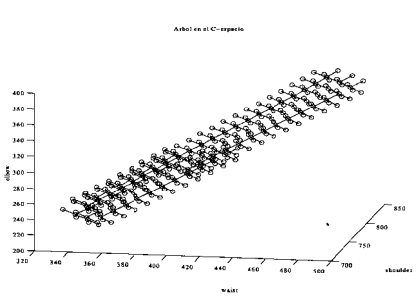
- Robótica. Planificación de caminos
  - ✓ Heurística I
    - ✓ $h_1(\mathbf{n}) = \mathbf{K_1}*\alpha$
    - ✓ $h_2(\mathbf{n}) = \mathbf{K_2}*d(\mathbf{n},\mathbf{t})$

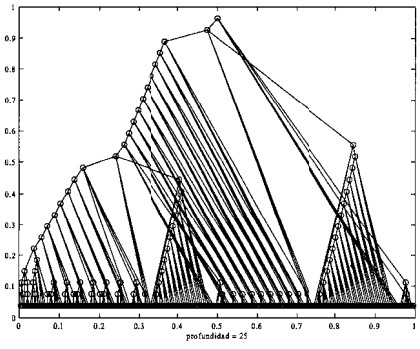


Algoritmo A*. Ejemplos V

- Robótica. Planificación de caminos
  - ✓ Heurística I. Comparación 90 expansiones

Heurística	K1	K2
A	1	1
B	1	1.668
C	1	100



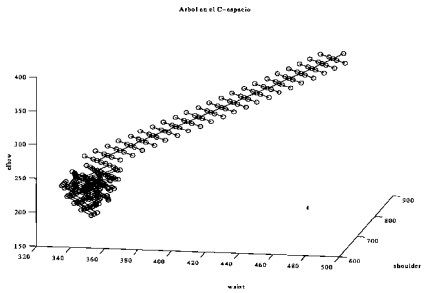


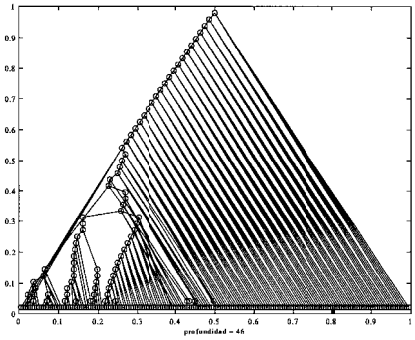
Heurística A

Algoritmo A*. Ejemplos VI

- Robótica. Planificación de caminos
  - ✓ Heurística I. Comparación 90 expansiones

Heurística	K1	K2
A	1	1
B	1	1.668
C	1	100

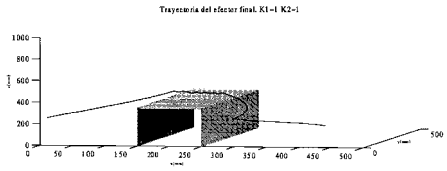


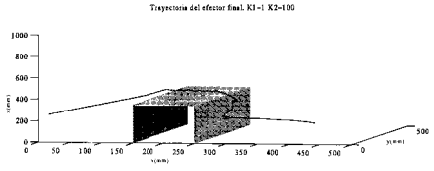


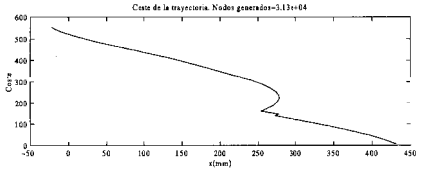
Heurística C

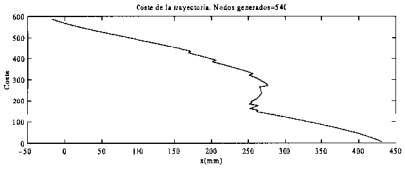
Est. búsqueda. Algoritmo A*. Ejemplos VII

- Robótica. Planificación de caminos
  - ✓ Heurística I. Trayectorias







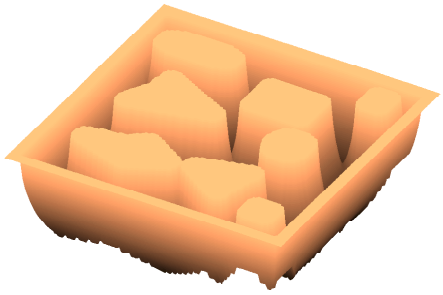


Algoritmo A*. Ejemplos VIII

- Robótica. Planificación de caminos
  - ✓ Heurística II

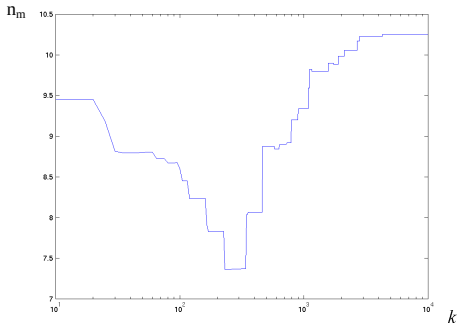
$$h = d_f + k * p$$

$$n_m = \frac{1}{N} \sum_{i=1}^{N} \frac{n_i}{l_i}$$



Algoritmo A*. Ejemplos IX

- Robótica. Planificación de caminos
  - ✓ Heurística II. Comparativa



Est. búsqueda. Algoritmos en Juegos

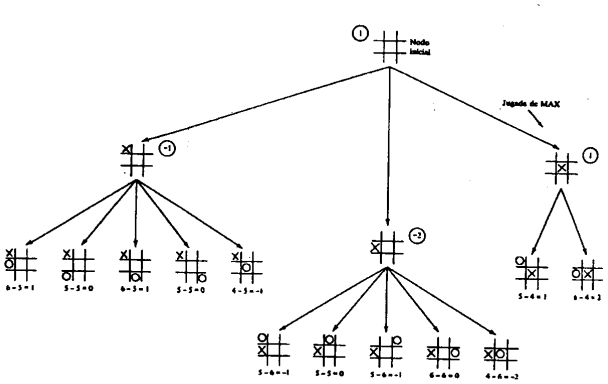
- Juegos
  - Utilizan actitudes inteligentes
    - 3 en raya
    - Ajedrez
  - Presentan un conjunto de reglas bien definidas
  - Se ha de hacer frente a modificaciones en la base de datos
    - Aplicación de reglas por el sistema de producción
    - Modificaciones externas (Jugador contrario)
  - Estrategias basadas en utilización de heurísticas
    - Función de evaluación
  - Tipos
    - Minimax
    - Alfabeta

Algoritmos en Juegos I

- Minimax
  - Planteamiento inicial
    - "Realizar la mejor jugada (Aplicar la mejor regla) suponiendo que el contrario hace lo mismo"
  - Dos Jugadores
    - MAX == Sistema de producción
    - MIN == Contrario
  - Criterior para la definición de la función de evaluación
    - Incremento => Beneficia a MAX
    - Decremento => Beneficia al MIN
  - Valor de evaluación
    - Minimo de la función de evaluación para todas las jugadas de MIN
  - Jugada elegida: La que **maximiza** el **valor de evaluación**

Algoritmos en Juegos II

- Minimax. Ejemplo de aplicación



Algoritmos en Juegos III

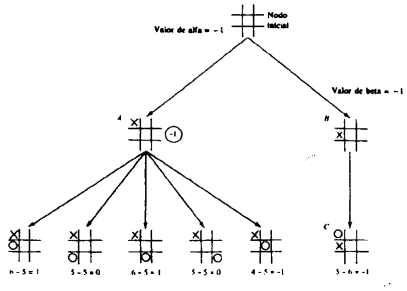
- Alfabeta
  - ✓ Basado en la estrategia Minimax
  - ✓ Planteamiento
    - ✓ Evitar la exploración de jugadas donde MIN tiene ventaja
  - ✓ Designación de valores de corte de exploración
    - ✓ α para cada nodo MAX
      - No decreciente
      - Se actualiza con los "valores de evaluación" de las jugadas de MIN
    - ✓ β para cada nodo MIN
      - No creciente
      - Se corresponde con el "valor de evaluación"

Algoritmos en Juegos IV

- Alfabeta. Procedimiento
  - ✓ Se inicia α con un valor $-\infty$
  - ✓ Para los nodos sucesores MIN
    - ✓ Se inicia β un valor
      - Para cada jugada si la f. de evaluación toma un valor menor que β se actualiza
  - ✓ Abandono de exploración
    - ✓ Si $\beta <= \alpha$.
      - Porque ya hay jugadas mejores para MAX
    - ✓ Si $\alpha >= \beta$ (antecesor)
      - Se ha producido una mejora (un aumento en la ventaja sobre el contrario MIN)

Algoritmos en Juegos V

- Alfabeta. Ejemplo



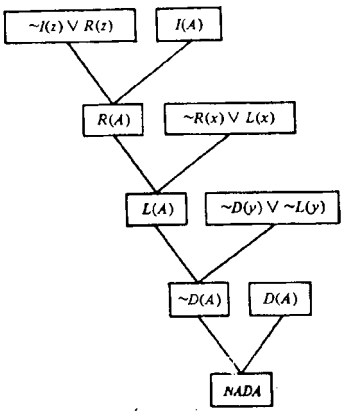
Est. búsqueda. Aplicaciones en lógica de P.

- Sistema de producción de refutación por resolución
  - ✓ Base de datos
    - ✓ Conjunto de fórmulas en FNC supuestas ciertas
    - ✓ Negación de la fórmula objetivo
  - ✓ Reglas de producción
    - ✓ Pares de clausulas unificables y resolubles
  - ✓ Sistema de control
    - ✓ Exploración primero amplitud
- Ejemplo
  - ✓ Considerar el conjunto de fórmulas $(\forall z)(I(z) \Rightarrow R(z))$ $(\forall y)(D(y) \Rightarrow \neg L(y))$ $(\forall x)(R(x) \Rightarrow L(x))$ $D(A)$
  - ✓ Fórmula objetivo

$$\neg I(A)$$

Aplicaciones en lógica de P. I

- Sistema de producción de refutación por resolución
  - ✓ El objetivo es alcanzar una contradicción
    - ✓ Obtención de la cláusula vacia dentro del conjunto de FBD's
  - ✓ Se puede realizar una representación como árbol binario
    - ✓ Exploración de grafos



Aplicaciones en lógica de P. II

- Sistema de producción de refutación por resolución