

Curso Extraordinario  
INTELIGENCIA ARTIFICIAL Y SISTEMAS  
EXPERTOS



Contenidos del Curso

- ☛ Introducción a la I.A.
- ☛ ¿Cómo razonamos?. Algunas experiencias con el razonamiento automático
- ☛ Procedimientos de solución automática de problemas
  - Los sistemas de producción
  - Los juegos
  - Los sistemas expertos
- ☛ Redes Neuronales artificiales. ¿Cómo reproducir el "funcionamiento" del cerebro con un ordenador"
- ☛ La IA en el nuevo milenio. Sistemas autónomos



## 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

Resolución de problemas

Vidal Moreno. USAL. Junio 2005

3



## Representación de problemas.

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

Resolución de problemas

Vidal Moreno. USAL. Junio 2005

4



## Diferenciación de métodos computacionales I

### ■ Métodos clásicos (directos)

- Implementación 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: Explosión combinatoria de posibilidades

Resolución de problemas

Vidal Moreno, USAL, Junio 2005

7



## 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

Resolución de problemas

Vidal Moreno, USAL, Junio 2005

8



## 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 Ra
6       DATOS
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:  $0 \leq x \leq 6$ ,  $0 \leq y \leq 8$
- 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

Resolución de problemas

Vidal Moreno. USAL. Junio 2005

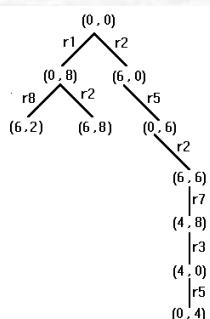
11



## 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



Resolución de problemas

Vidal Moreno. USAL. Junio 2005

12



## Sistemas de producción. Ejemplo II (I)

### 8-puzzle

- Enunciado

- “Se trata del problema constituido 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

$$\begin{array}{ccc} 6 & 2 & 5 \\ 3 & 0 & 4 \\ 1 & 7 & 8 \end{array}
 \quad
 \begin{array}{lll} a(1,1)=6 & a(1,2)=2 & a(1,3)=5 \\ a(2,1)=3 & a(2,2)=0 & a(2,3)=4 \\ a(3,1)=1 & a(3,2)=7 & a(3,3)=8 \end{array}$$

Resolución de problemas

Vidal Moreno, USAL, Junio 2005

13



## 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

Resolución de problemas

Vidal Moreno, USAL, Junio 2005

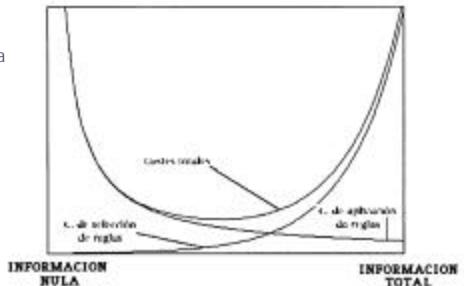
14



## 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 las reglas de producción.



Resolución de problemas

Vidal Moreno. USAL. Junio 2005

15



## 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

Resolución de problemas

Vidal Moreno. USAL. Junio 2005

16



## 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

2	8	3		1	2	3
1	6	4		8	0	4
7	0	5		7	6	5

$f = -(\text{Nº de casillas descolocadas})$

Resolución de problemas

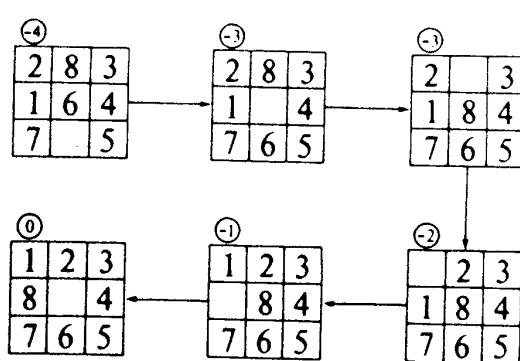
Vidal Moreno, USAL, Junio 2005

17



## Estrategia irrevocable II

Ejemplo: 8 puzzle.



Resolución de problemas

Vidal Moreno, USAL, Junio 2005

18



## 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

Resolución de problemas

Vidal Moreno, USAL. Junio 2005

19



## 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.*

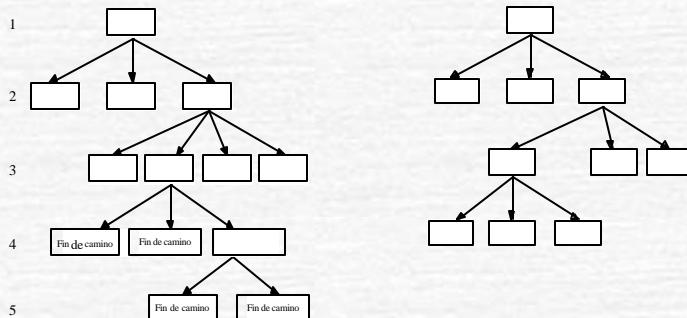
Resolución de problemas

Vidal Moreno, USAL. Junio 2005

20

## Estrategias tentativas. BACKTRACKING II

### Ejemplo de aplicación



Resolución de problemas

Vidal Moreno. USAL. Junio 2005

21

## 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<-APIREGL(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(RLSTABD)
13. if CAMINO = FALLO, go CICLO
14. return CONS(R,CAMINO)

Resolución de problemas

Vidal Moreno. USAL. Junio 2005

22

### Ejemplo 8-puzzle



Resolución de problemas

23

Diagrama de 8-puzzle:

1	2	8	3					
	1	6	4					
7		5						
2	8	3						
1	6	4						
	7	5						

Diagrama de 8-puzzle:

5	8	3						
	2	6	4					
1	7	5						
2	8	3						
1	6	4						
	7	5						

Diagrama de 8-puzzle:

6	8	3						
	2	6	4					
1	7	5						
2	8	3						
1	6	4						
	7	5						

Diagrama de 8-puzzle:

3	8	3						
	2	6	4					
1	7	5						
2	8	3						
1	6	4						
	7	5						

Resumen de pasos:

- 1. Se aplica la regla de mover el blanco a la derecha.
- 2. Se aplica la regla de mover el blanco a la derecha.
- 3. Se aplica la regla de mover el blanco a la derecha.
- 4. Se aplica la regla de mover el blanco a la derecha.
- 5. Se aplica la regla de mover el blanco a la derecha.
- 6. Se aplica la regla de mover el blanco a la derecha.
- 7. Se aplica la regla de mover el blanco a la derecha.

Otra vez, este estado no es el objetivo, por lo que se aplica la regla de mover el blanco a la derecha. No hay más reglas ni estrategias que puedan aplicarse al estado previo (6), por lo que se aplica el mismo movimiento ramal y aplicamos «mover blanco hacia abajo» al estado (5). La continuación es la columna siguiente.

Otra vez hemos aplicado las reglas socializadas al objetivo, por lo que, ..., etc.

Este estado ya había aparecido antes, por lo que resolvemos al último movimiento y aplicamos «mover blanco a la derecha» al estado (5) en su lugar. La continuación es la columna siguiente.

## Estrategias tentativas II



### Estrategias tentativas

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

Diagrama de datos:

```
graph TD; DB1[DB(1)] -- R1 --> DB2[DB(2)]; DB1 -- Rn+1 --> DBn[DB(n)]; DB2 --- nReglas[n Reglas]; DB2 --- DBn1[DB(n-1)];
```

### Solución:

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

Resolución de problemas

Vidal Moreno, USAL. Junio 2005

24



## Estrategias tentativas. Exploración de grafos I

### Definiciones

- Grafo: Conjunto de Nodos unidos por un conjunto de arcos
- Árbol: 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
  - Árbol: 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

Resolución de problemas

Vidal Moreno, USAL, Junio 2005

25



## 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

Resolución de problemas

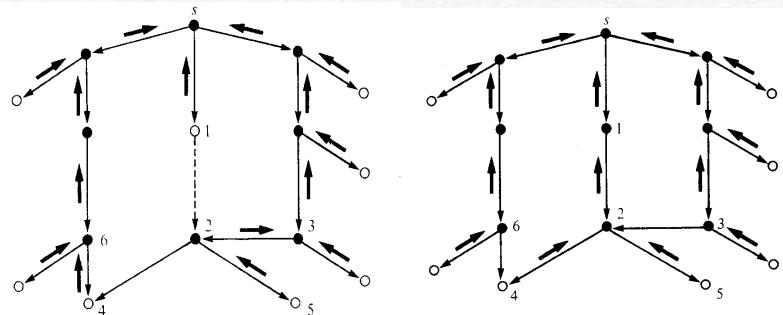
Vidal Moreno, USAL, Junio 2005

26



## Estrategias tentativas. Exploración de grafos III

■ Generación de árboles



Resolución de problemas

Vidal Moreno, USAL, Junio 2005

27

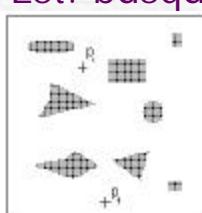


## Est. búsqueda. Ejemplos I

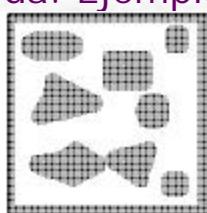
■ Robótica



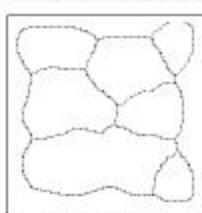
Espacio de estados



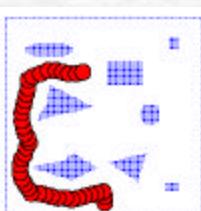
Espacio de trabajo



Espacio de las configuraciones



Solución con backtracking



Resolución de problemas

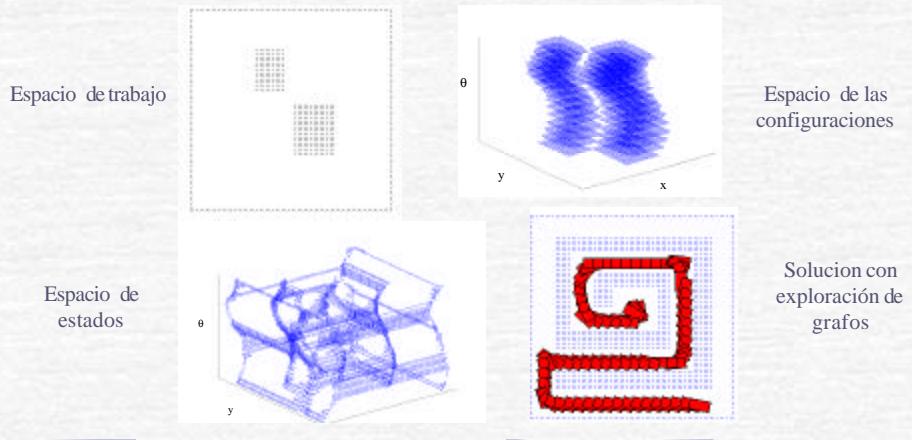
Vidal Moreno, USAL, Junio 2005

28



## Est. búsqueda. Ejemplos II

### Robótica móvil. Traslación y giro



Resolución de problemas

Vidal Moreno, USAL, Junio 2005

29



## 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  $KB \xrightarrow{f} \mathfrak{R}$
- 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

Resolución de problemas

Vidal Moreno, USAL, Junio 2005

30



## 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

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

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



## 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

$$h(\mathbf{n}) = 0 \text{ y } g(\mathbf{n}) = d(\mathbf{n}) \quad \text{siendo } d(\mathbf{n}) \text{ la profundidad del nodo}$$

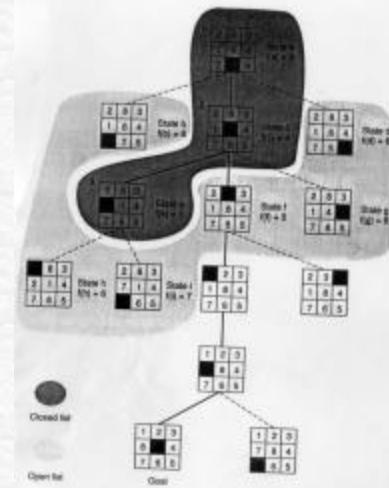
### Algoritmo A\*

- 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

Resolución de problemas

Vidal Moreno, USAL, Junio 2005

33



- ### 8-puzzle. Proceso de solución

Resolución de problemas

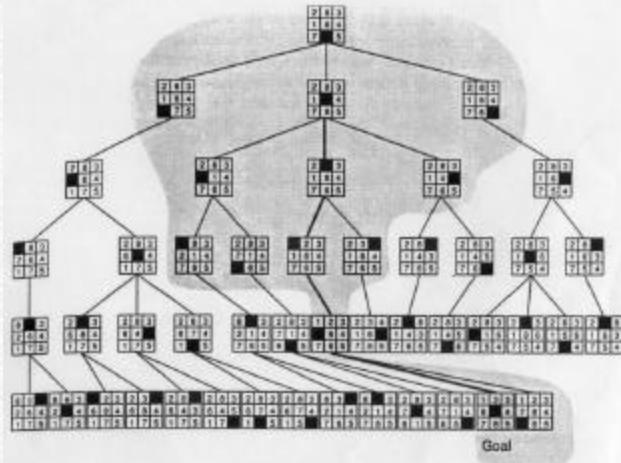
Vidal Moreno, USAL, Junio 2005

34



## Algoritmo A\*. Ejemplos II

- Comparativa de estrategias



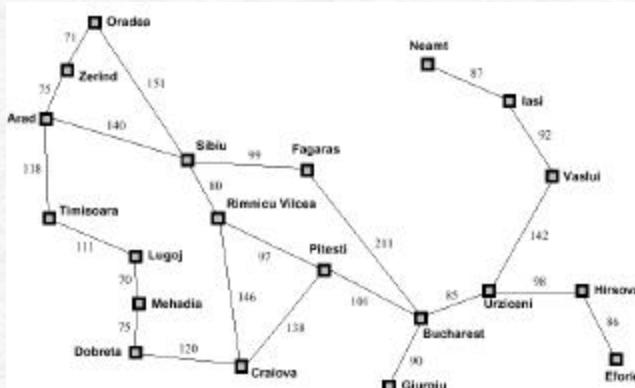
Resolución de problemas

Vidal Moreno, USAL, Junio 2005

35



## Algoritmo A\*. Ejemplos III Sistema de Navegación



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	103
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

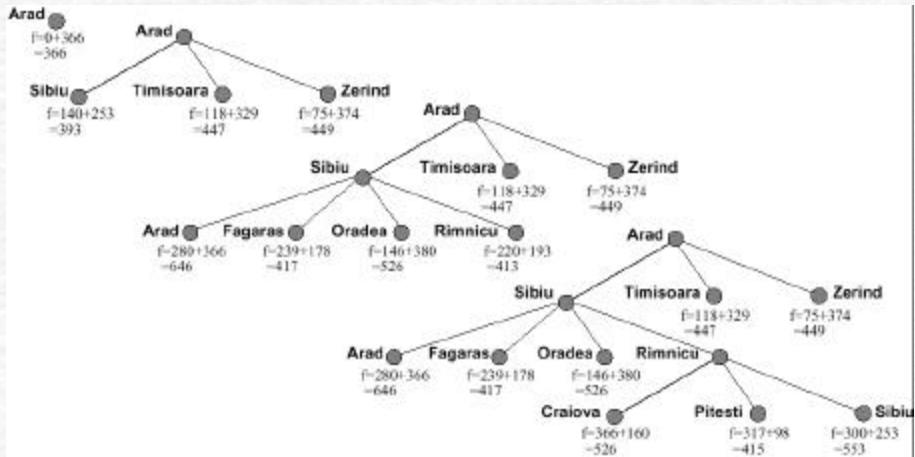
Resolución de problemas

Vidal Moreno, USAL, Junio 2005

36



## Algoritmo A\*. Ejemplos IV



Resolución de problemas

Vidal Moreno, USAL, Junio 2005

37



## 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
  - Alfabeto

Resolución de problemas

Vidal Moreno, USAL, Junio 2005

38



## 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
- Criterio para la definición de la función de evaluación
  - Incremento => Beneficia a MAX
  - Decremento => Beneficia al MIN
- Valor de evaluación
  - Mínimo de la función de evaluación para todas las jugadas de MIN
- Jugada elegida: La que **maximiza el valor de evaluación**

Resolución de problemas

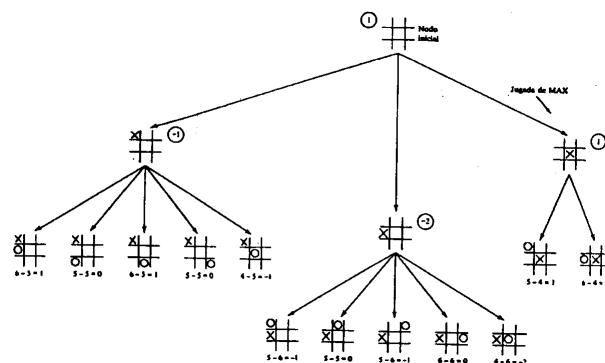
Vidal Moreno, USAL, Junio 2005

39



## Algoritmos en Juegos II

### Minimax. Ejemplo de aplicación



Resolución de problemas

Vidal Moreno, USAL, Junio 2005

40



## Algoritmos en Juegos III

### Alfabeto

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

Resolución de problemas

Vidal Moreno. USAL. Junio 2005

41



## Algoritmos en Juegos IV

### Alfabeto. Procedimiento

- Se inicia  $\alpha$  con un valor  $-\infty$
- Para los nodos sucesores MIN
  - Se inicia  $\beta$  un valor
    - Para cada jugada si la f. de evaluación toma un valor menor que  $\beta$  se actualiza
- Abandono de exploración
  - Si  $\beta \leq \alpha$ .
    - 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)

Resolución de problemas

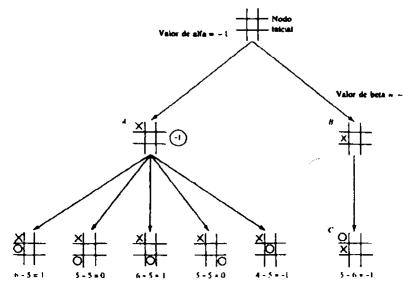
Vidal Moreno. USAL. Junio 2005

42



## Algoritmos en Juegos V

### Alfabeto. Ejemplo



Resolución de problemas

Vidal Moreno. USAL. Junio 2005

43



## Prolog. Generación de un sistema de producción

### Ejemplo

- Un pastor ha atravesar un río en una balsa de dos plazas como máximo. Va acompañado de un lobo, una oveja y una coliflor. Dadas las predicciones gastronómicas de los participantes, se trata de resolver el problema con un sistema de producción

### Separación entre base de datos, reglas y sistema de control

- Base de datos: `e(P,L,O,B)` donde
  - P, L, O, B son las posiciones (Oeste o Este) de cada uno de los miembros
  - Implementación: Utilizando functores (Funciones de LPPO)

```
domains
    est=estado(symbol,symbol,symbol,symbol)
    lista_estados=est*
```

Resolución de problemas

Vidal Moreno. USAL. Junio 2005

44



## Prolog. Generación de un sistema de producción

- ☞ Implementación de las reglas
  - 4 movimientos de cruzar orilla (PL,PO,PB,P)
  - Necesita definir un predicado opuesto:
    - Opuesto(e,o).
    - Opuesto(o,e)
- ☞ Se debe incluir la verificación de la precondición
  - Definición del predicado seguro
    - Ejemplo: Inseguro si el pastor esta en una orilla distinta a la del lobo y la oveja
- ☞ Se debe garantizar que no existen estados repetidos
  - Definición del predicado miembro

Resolución de problemas

Vidal Moreno. USAL. Junio 2005

45



## Prolog. Generación de un sistema de producción

- ☞ Implementación de las reglas

```

lobos2.pro
55:30      Insert      Indent
mueve(estado(I,F,O,B),estado(P,F,O,B),Lista) :- 
    opuesto(I,F),
    not(inseguro(estado(P,F,O,B))),
    not(miembro(estado(P,F,O,B),Lista)),
    write("\nLleva al lobo. Estado: ",F,".",F,".",O,".",B).
mueve(estado(I,L,I,B),estado(P,L,F,B),Lista) :- 
    opuesto(I,F),
    not(inseguro(estado(P,L,F,B))),
    not(miembro(estado(P,L,F,B),Lista)),
    write("\nLleva a la oveja. Estado: ",F,".",L,".",F,".",B).
mueve(estado(I,L,O,I),estado(P,L,O,F),Lista) :- 
    opuesto(I,F),
    not(inseguro(estado(P,L,O,F))),
    not(miembro(estado(P,L,O,F),Lista)),
    write("\nLleva a la berza. Estado: ",F,".",L,".",O,".",F).

```

Resolución de problemas

Vidal Moreno. USAL. Junio 2005

46

## Prolog. Generación de un sistema de producción

- Sistema de control (Proporcionado por el lenguaje)
  - Backtracking sin información

```

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 aplicado demasiadas reglas. LIMITE es una variable global esp 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(RLISTABD)
13. if CAMINO = FALLO,go CICLO
14. return CONS(R,CAMINO)

```

Resolución de problemas Vidal Moreno. USAL. Junio 2005 47

## Prolog. Generación de un sistema de producción

- Definición del objetivo: Incluir en goal principal

```

/*Genera la salida del problema*/
labos2() :-principal, !.

principal:-
    ruta([estado(o,o,o,o),estado(e,e,e,e),[estado(o,o,o,o)]) .

```

Resultado de la ejecución

Resolución de problemas Vidal Moreno. USAL. Junio 2005 48



## Prolog. Depuración

🕒 Herramienta separada

`llobos2.exe - Visual Prolog Debugger For Windows`

60:17 Insert Indent ReadOnly

```

mueve(estado[I, I, O, B], estado[F, F, O, B], Lista) :-  

    opuesto(I, F),  

    not(inseguro(estado(F, F, O, B))),  

    not(miembro(estado(F, F, O, B), Lista)),  

    write("!\nLleva al lobo. Estado: " ,F, " ", F, " ", O, " ", B).  

mueve(estado[I, L, I, B], estado(F, L, F, B), Lista) :-  

    opuesto(I, F),  

    not(inseguro(estado(F, L, F, B))),  

    not(miembro(estado(F, L, F, B), Lista)),  

    write("!\nLleva a la oveja. Estado: " ,F, " ", L, " ", F, " ", B).  

mueve(estado[I, L, O, I], estado(F, L, O, F), Lista) :-  

    opuesto(I, F),
    
```

Variables For Current Class

- I = `"o"`
- O = `"o"`
- B = `"o"`
- F = `-`
- Lista = [estado("o", "o", "o", "o")]

Modules

- llobos2
- llobos2exe
- + llobos2.pro

Backtrace Prints

```

1  mueve( estu, mat, lista_estados ) - (I,o,I)

```

Resolución de problemas Vidal Moreno. USAL. Junio 2005 49



## Prolog

🕒 Ejercicios

- Realizar los programas en Prolog que resuelven los siguientes problemas
  - “Dados dos baldes de 8 y 6 litros (initialmente vacíos) encontrar la secuencia de operaciones que hay que realizar para que el de 8 litros tenga la mitad de su contenido”
  - Generar un sistema de navegación
- Realizar una implementación de sistema de producción que incorpore la limitación en profundidad de la búsqueda

Resolución de problemas Vidal Moreno. USAL. Junio 2005 50