

Dipn. de Ciència de la Computació i Intel·ligència Artificial
Dipn. de Ciència de la Computación e Inteligencia Artificial

Sistemas Inteligentes



Bloque 1: Inteligencia Artificial. Búsqueda. Heurística.

Tema 1: Inteligencia Artificial y Sistemas Inteligentes. Objetivos.



T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

1

Dipn. de Ciència de la Computació i Intel·ligència Artificial
Dipn. de Ciència de la Computación e Inteligencia Artificial

Sistemas Inteligentes



Tema 1: IA y Sistemas Inteligentes. Objetivos. Índice

- ¿Qué es la Inteligencia?
- Tipos de Inteligencia según Howard Gardner
- ¿Qué es IA?
- ¿Puede ser una máquina inteligente?
- Historia de la I.A
- Áreas de Aplicación
- Futuro de la IA
- Bibliografía Básica.



T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

2

Dipn. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Universitat d'Alacant
Universidad de Alicante

¿Qué es la Inteligencia?

- Todos somos inteligentes.
- No es patrimonio exclusivo de los genios.
- No hay una única inteligencia.
- ¿Característica que distingue al ser humano de las demás especies?.
- Efecto Flynn
- Aspectos de la Inteligencia
 - La memoria
 - El pensamiento abstracto y el razonamiento
 - El lenguaje y la comunicación
 - El aprendizaje
 - La resolución del problemas
 - La creatividad

T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

3

Dipn. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Universitat d'Alacant
Universidad de Alicante

¿Qué es la Inteligencia?

- La inteligencia natural no tiene una fácil definición (distintas acepciones), pero en general:
 - potencia intelectual: facultad de conocer, de entender o comprender.
 - conjunto de habilidades desarrolladas por el ser humano para recibir información, analizarla, comprenderla, almacenarla y saberla aplicar en el futuro para la resolución de problemas.
- Hofstadter (1987): Inteligencia es la habilidad para:
 - responder flexiblemente a diferentes situaciones,
 - saber aprovechar circunstancias fortuitas,
 - dar sentido a mensajes ambiguos o contradictorios,
 - encontrar similitudes entre situaciones diferentes, y
 - generar nuevos conceptos e ideas innovadoras.

T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

4

Sistemas Inteligentes

Típos de Inteligencia según Howard Gardner (I)

Inteligencia lingüística, la que tienen los escritores, los poetas, los buenos redactores.

Inteligencia lógica-matemática, la que utilizamos para resolver problemas de lógica y matemáticas. Es la inteligencia que tienen los científicos. Se corresponde con el modo de pensamiento del hemisferio lógico.

Inteligencia espacial, consiste en formar un modelo mental del mundo en tres dimensiones, es la inteligencia que tienen los marineros, los ingenieros, los cirujanos, los escultores, los arquitectos, o los decoradores.

Inteligencia musical, es naturalmente la de los cantantes, compositores, músicos, bailarines.

Inteligencia corporal-kinestésica, o la capacidad de utilizar el propio cuerpo para realizar actividades o resolver problemas. Es la inteligencia de los deportistas, los artesanos, los cirujanos y los bailarines



Universitat d'Alicant
Universidad de Alicante

T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

5

Sistemas Inteligentes

Típos de Inteligencia según Howard Gardner (y II)

Inteligencia Intrapersonal, es la que nos permite entendernos a nosotros mismos. No está asociada a ninguna actividad concreta.

Inteligencia Interpersonal, la que nos permite entender a los demás, y la solemos encontrar en los buenos vendedores, políticos, profesores o terapeutas.

Inteligencia emocional es formada por la inteligencia intrapersonal y la interpersonal y juntas determinan nuestra capacidad de dirigir nuestra propia vida de manera satisfactoria.

Inteligencia Naturalista, la que utilizamos cuando observamos y estudiamos la naturaleza. Es la que demuestran los biólogos o los herbolarios.

Inteligencia Cibernética, la que desarrollan las personas estudiando y aprovechando la ciencia que se ocupa de los sistemas de control y telecomunicaciones.



Universitat d'Alicant
Universidad de Alicante

T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

6

Sistemas Inteligentes

¿Qué es Inteligencia Artificial (IA)?

Sistemas que piensan como humanos

- “El nuevo y excitante esfuerzo de hacer que los computadores piensen... máquinas con mentes, en el más amplio sentido literal”. (Haugeland, 1985)
- “La automatización de actividades que vinculamos con procesos de pensamiento humano, actividades como la toma de decisiones, resolución de problemas, aprendizaje...”. (Bellman, 1978)

Sistemas que piensan razonadamente

- “El estudio de las facultades mentales mediante el uso de modelos computacionales”. (Charniak y McDermott, 1985)
- “El estudio de los cálculos que hacen posible percibir, razonar y actuar”. (Winston, 1992)

Sistemas que actúan como humanos

- “El arte de construir máquinas capaces de hacer cosas que requerirían inteligencia si las hicieran los seres humanos”. (Minsky, 1986)
- “El estudio de cómo lograr que los computadores realicen tareas que, por el momento, los humanos hacen mejor”. (Rich y Knight, 1991)

Sistemas que actúan razonadamente

- “Estudio del diseño de agentes inteligentes”. (Poole et al., 1998)
- “IA...está relacionada con conductas inteligentes en artefactos”. (Nilsson, 1998)

T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

7

Sistemas Inteligentes

¿Qué es IA?

IA fuerte

Proporcionando a un programa de computador suficiente capacidad de procesamiento y dándole la suficiente inteligencia, se puede crear un ordenador que pueda pensar y ser consciente de la misma forma que lo hacen los seres humanos.

IA débil

El comportamiento inteligente puede ser modelado y usado por un computador para resolver problemas complejos.

Solo porque un computador se comporte de manera inteligente, no significa que sea realmente inteligente de la misma manera en la que lo es un ser humano.

T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

8

Dipòt. de Ciència de la Computació i Intel·ligència Artificial
Dipòt. de Ciència de la Computació i Intel·ligència Artificial

Sistemes Inteligentes

¿Puede ser una máquina inteligente?

Test de Alan Turing (1950)

A. HARDING

¿en qué habitación está la máquina?

T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

9

Dipòt. de Ciència de la Computació i Intel·ligència Artificial
Dipòt. de Ciència de la Computació i Intel·ligència Artificial

Sistemas Inteligentes

¿Puede ser una máquina inteligente?

La sala china: Searle en 1980 propuso un contraejemplo al test de Turing.

HOME

¿Qué es HOME?

T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

10

Dipn. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Universitat d'Alacant
Universidad de Alicante

Sistemas Inteligentes

Historia de la IA

Historia

Bases de la I.A. Moderna

- Definición del Campo: Conferencia de Dartmouth (1956) y los Años Dorados (1956-63)
- Las Conquistas de los Micro-Mundos: 1963-70
- Años de Crítica y Madurez: Los Difíciles Años 70.
- Etapa de Expansión: Los Años 80

Estado actual

- Se abordan problemas concretos.
- Modelos de representación simbólica.
- Modelos conexionistas.
- Modelos evolutivos.
- Robots: AIBO (robot-perro), HONDA (robot humanoide)
- ¿Modelo más adecuado en IA?
- DEEP LEARNING**

T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

11

Dipn. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Universitat d'Alacant
Universidad de Alicante

Sistemas Inteligentes

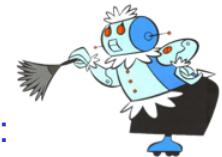
Aréas de Aplicación

- Problemas de percepción: visión y habla.
- Planificación, estrategias inteligentes.
- Robótica.
- Predicción financiera.
- Aprendizaje.
- Minería de datos.
- Juegos.
- Mundos virtuales.
- Internet (google, amazon, ebay, etc.)
- Sistemas expertos en campos de la medicina, geología, aeronáutica..



HAL ¿Fantasía o realidad?

- Ajedrez
- Reconocimiento del habla
- Visión
- Emociones



Algunos éxitos:

- Logic Theorist (1956)
- Prospector
- Deep Blue (1997)

T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

12

Sistemas Inteligentes

Futuro de la IA

Orientado a abordar aquellas tareas que, ya sea por lo incomodo, peligroso o complicado, conviene apoyarlas o delegarlas en sistemas inteligentes artificiales.



Universitat d'Alicant
Universidad de Alicante

T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

13

Sistemas Inteligentes

Futuro de la IA

Aunque la IA ya ha sido capaz de producir algunos sistemas prácticos muy útiles, alcanzar una inteligencia artificial fuerte está aún muy distante... pero se ha iniciado el camino ..



Universitat d'Alicant
Universidad de Alicante

T1. Inteligencia Artificial y Sistemas Inteligentes. Objetivos

14

Bibliografía Básica.

Inteligencia Artificial. Un enfoque Moderno. Stuart Russell, Peter Noving. Ed Prentice Hall.



Tema 2. Estrategias de búsqueda



Tema 2. Estrategias de búsqueda

Objetivos

- Conocer la importancia de la búsqueda en IA
- Conocer los sistemas de producción en búsqueda
- Ser capaz de especificar un problema mediante un Sistema de Producción (SP)
- Aprender problemas clásicos de IA
- Entender las distintas estrategias de búsqueda

Tema 2. Estrategias de búsqueda

Contenidos

1. Introducción
2. Especificación de problemas
3. Caracterización del problema
4. Problemas clásicos
5. Estrategias de búsqueda básicas
 1. Estrategia irrevocable
 2. Estrategia tentativa
 1. No informadas
 2. Informadas
6. Esquema de tipos básicos de búsqueda.
7. Estrategias híbridas.
8. Búsqueda heurística
 1. Conceptos básicos.
 2. Búsqueda A*. Búsqueda óptima.
 3. Nivel de información heurístico.
 4. Ejemplos de heurísticas.
 5. Inconvenientes de mantener la admisibilidad
 6. Relajación de la restricción de optimalidad
9. Bibliografía

1. Introducción

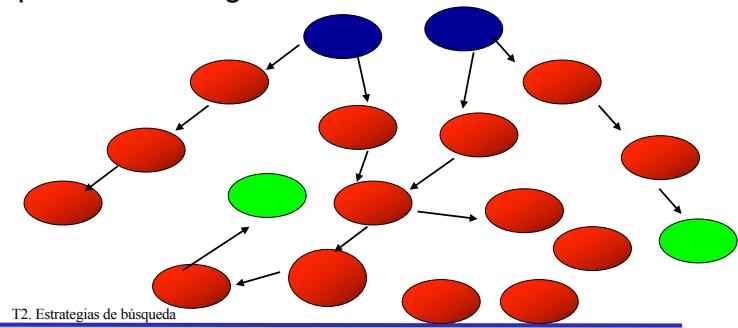
¿Qué podemos resolver mediante búsqueda?

- Algunos problemas que se resuelven con técnicas de búsqueda:
 - Problemas de búsqueda en rutas
 - Planificación líneas aéreas
 - Rutas en redes de computadores
 - Problemas turísticos
 - El viajante de comercio
 - La Distribución VLSI
 - Navegación de un robot
 - Secuenciación para el ensamblaje automático
 - Diseño de proteínas
 - Búsqueda en Internet
 - ...

2. Especificación de problemas

¿Cómo resolver un problema? (I)

- Definición del problema como una búsqueda en un espacio de estados
 - Definir un espacio de estados.
Definición por extensión que evite enumerar todos los estados que contiene
 - Especificar el/los estados iniciales
 - Especificar los estados meta
 - Especificar las reglas de transformación

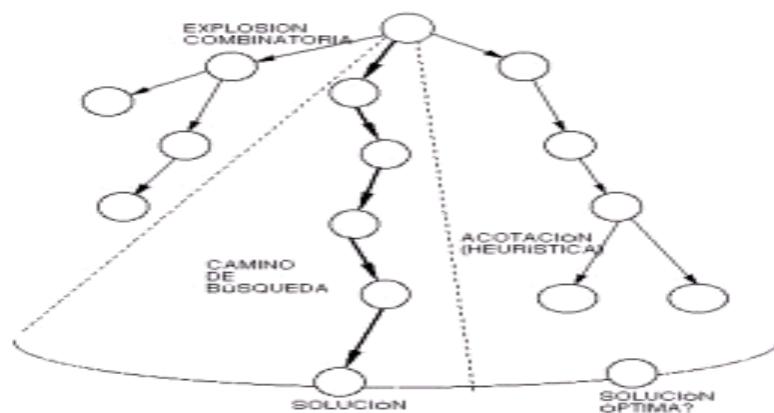


5

2. Especificación de problemas

¿Cómo resolver un problema? (II)

El proceso de búsqueda se puede realizar explorando un árbol (**árbol de búsqueda**) o en general un grafo (eliminando repeticiones de estados)



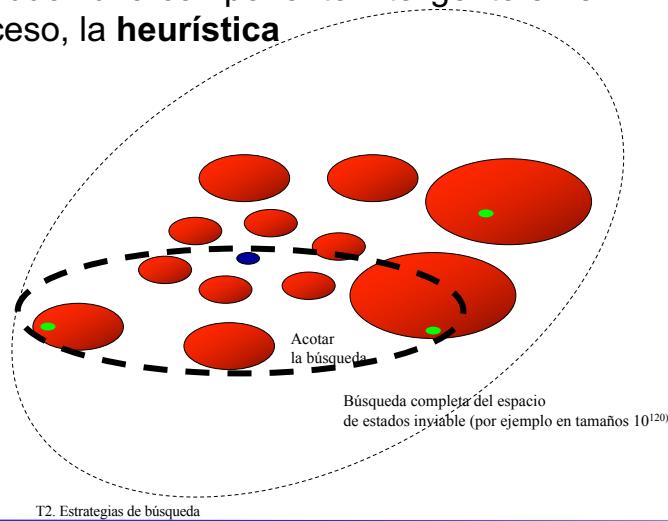
T2. Estrategias de búsqueda

6

2. Especificación de problemas

¿Cómo resolver un problema? (III)

- ¿Cuál es el principal problema de la búsqueda de estados?
 - Introducir una componente inteligente en el proceso, la **heurística**



T2. Estrategias de búsqueda

7

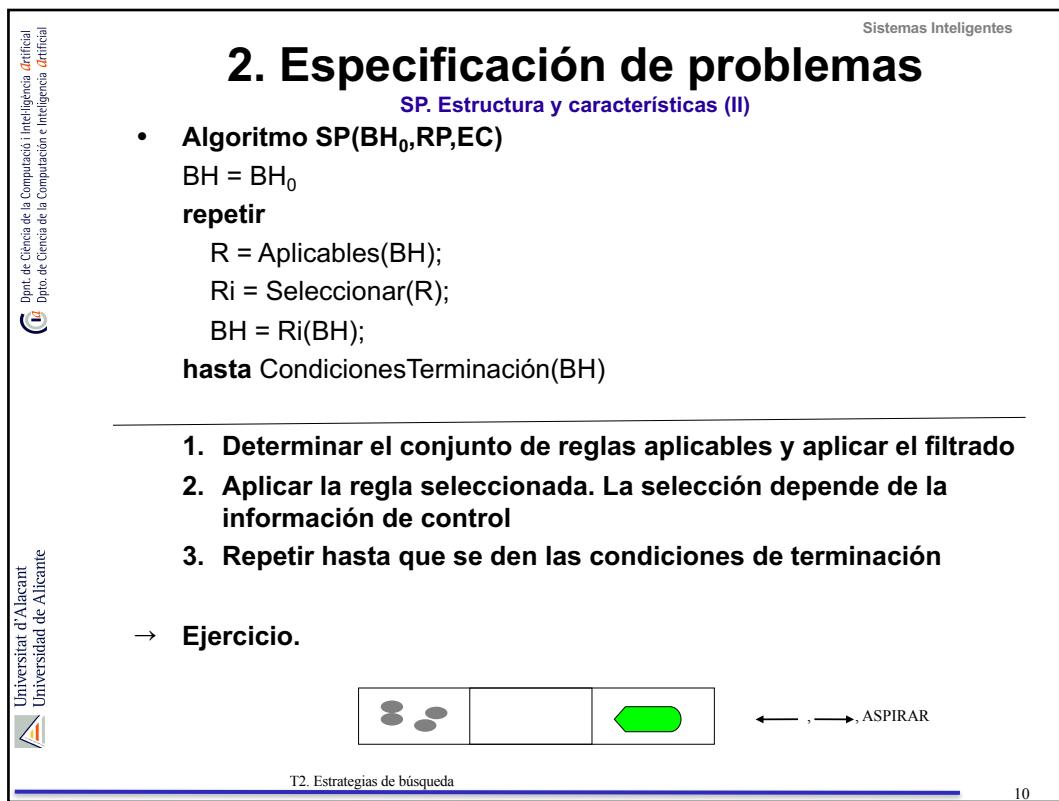
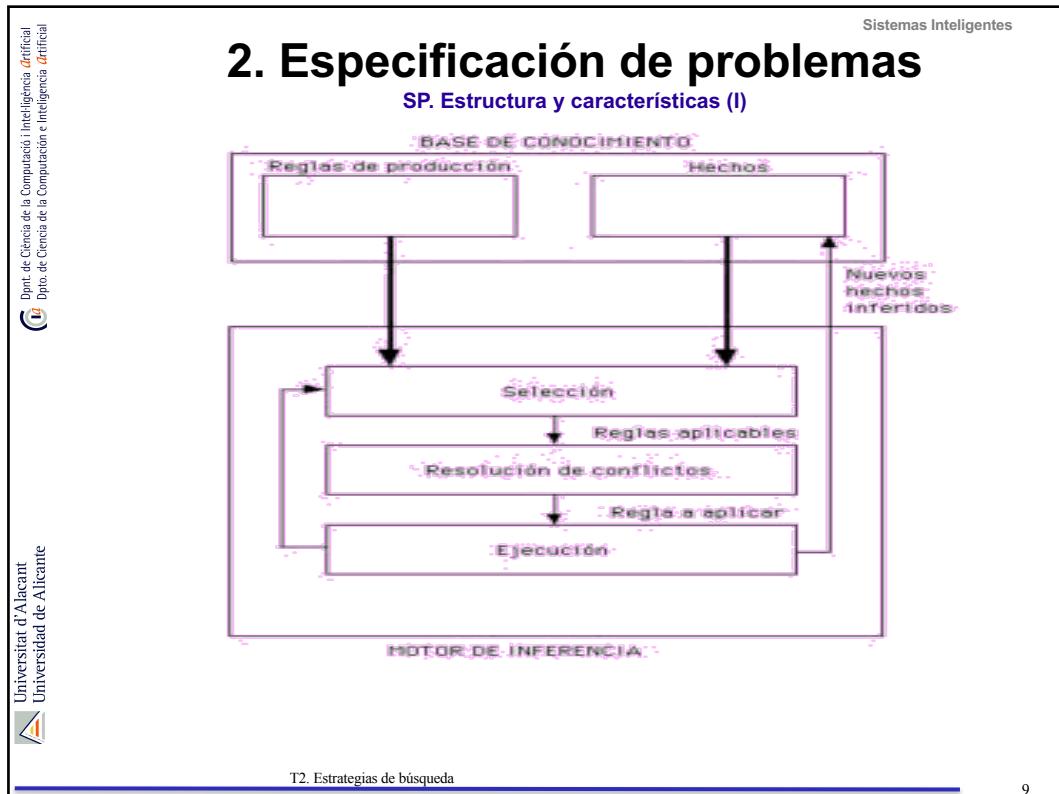
2. Especificación de problemas

Sistemas de producción (I)

- Sistemas de producción
 - Formalizan los problemas de búsqueda de estados
 - Propuesto por POST en 1943
 - Un sistema de producción es una terna **(BH, RP, EC)** donde:
 - **BH (Base de Hechos).**
Conjunto de representaciones de uno o más estados por los que atraviesa el problema. Constituye la estructura de datos global
 - **RP (Reglas de Producción).**
Conjunto de operadores para la transformación de los estados del problema, es decir, de la base de hechos. Cada regla tiene dos partes:
 - Precondiciones
 - Postcondiciones
 - **EC (Estrategia de control).**
Determina el conjunto de reglas aplicables mediante un proceso de pattern-matching y resuelve conflictos entre varias reglas a aplicar mediante el filtrado

T2. Estrategias de búsqueda

8



3. Caracterización del problema (I)

¿Cómo elegir la EC?

- ¿Se puede **descomponer** el problema en subproblemas independientes?
- ¿Pueden ignorarse o al menos deshacerse pasos hacia la solución si se constata que son erróneos?
 - **Ignorables** -> Demostración de teoremas
 - **Recuperables** -> Podemos retroceder
 - **Irrecuperables** -> No se puede retroceder
- ¿Es el universo de discurso **predecible**?
- La **bondad** de una solución.
- ¿Es absoluta o relativa?
- La solución,
- ¿es un **estado** o es un **camino**?
- ¿Qué papel desempeña **el conocimiento**?

4. Problemas clásicos (I)

- **El problema de las jarras de agua:** Se tienen dos jarras de agua
 - Una de tres litros y otra de cuatro.
 - No disponen de marcas de medición.
 - Las jarras se pueden llenar y también vaciar.

¿Cómo se puede lograr tener exactamente dos litros de agua en la jarra de cuatro?

El espacio de estados: (x, y)

- » $x = 0, 1, 2, 3 \text{ ó } 4$ e $y = 0, 1, 2 \text{ ó } 3$.
- » x representa el número de litros de agua en la jarra de cuatro litros
- » y ídem en la jarra de tres litros.

El estado inicial corresponderá al estado $(0, 0)$ y el estado final (objetivo) al $(2, n)$ donde n significa cualquier valor en la jarra de tres litros.

Ejercicio: Plantear las reglas (precondiciones y postcondiciones) y aplicarlas para solucionar el problema

4. Problemas clásicos (II)

- Las reglas de producción

1	$(x, y) \text{ si } x < 4$	$\rightarrow (4, y)$
2	$(x, y) \text{ si } y < 3$	$\rightarrow (x, 3)$
3	$(x, y) \text{ si } x > 0$	$\rightarrow (x-d, y)$
4	$(x, y) \text{ si } y > 0$	$\rightarrow (x, y-d)$
5	$(x, y) \text{ si } x > 0$	$\rightarrow (0, y)$
6	$(x, y) \text{ si } y > 0$	$\rightarrow (x, 0)$
7	$(x, y) \text{ si } x+y \geq 4 \text{ e } y > 0$	$\rightarrow (4, y-(4-x))$
8	$(x, y) \text{ si } x+y \geq 3 \text{ e } x > 0$	$\rightarrow (x-(3-y), 3)$
9	$(x, y) \text{ si } x+y \leq 4 \text{ e } y > 0$	$\rightarrow (x+y, 0)$
	$(x, y) \text{ si } x+y \leq 3 \text{ e } x > 0$	$\rightarrow (0, x+y)$

Jarra 1 Jarra 2 Regla a aplicar

0	0	2
0	3	9
3	0	2
3	3	7
4	2	5
0	2	9
2	0	8

4. Problemas clásicos (III)

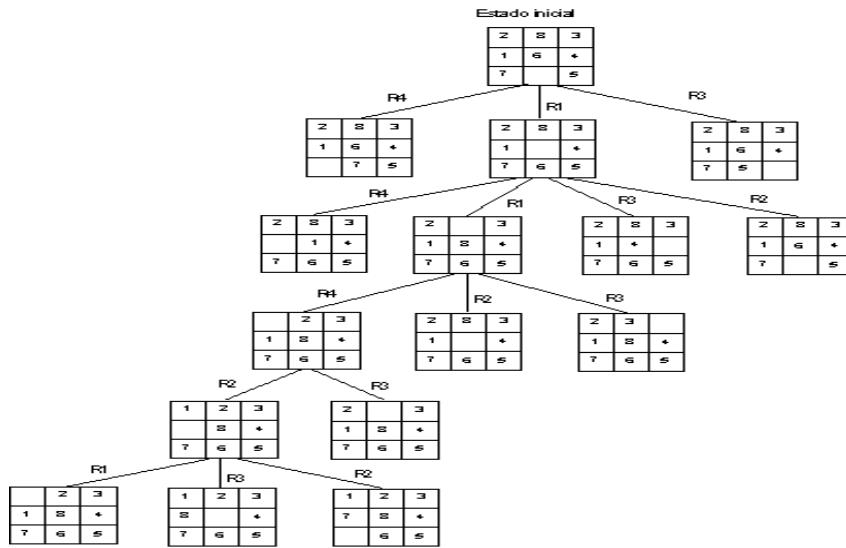
- Problema del 8-puzzle



- Espacio de estados (e)
- Consta de todas las configuraciones de las 8 fichas y el cuadro vacío ($\text{Card}(e) = 9!$) (en general $N!$, siendo $N-1$ el número de fichas).
- Estados iniciales (ei)
- La búsqueda puede arrancar desde cualquier combinación que no cumpla la condición de estado meta.
- Estados finales (ef)
 - Configuración que cumpla la condición de estado meta
- Reglas de producción (R1 Vacia arriba, R2 v. abajo , R3 v. a la derecha, R4 v. a la zqda)

4. Problemas clásicos (IV)

- La solución consiste en presentar las reglas aplicadas. En este caso: R1-R1-R4-R2-R3.

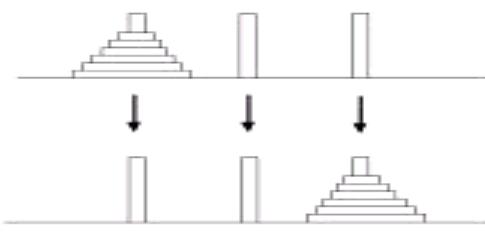


T2. Estrategias de búsqueda

15

4. Problemas clásicos (V)

- El problema del viajante de comercio
- El dilema del granjero
- El dilema de los misioneros
- Las torres de Hanoi

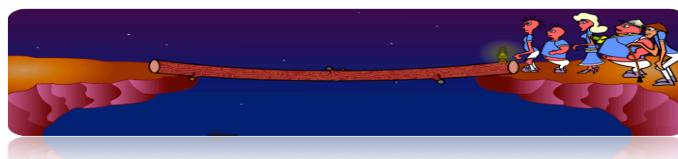


T2. Estrategias de búsqueda

16

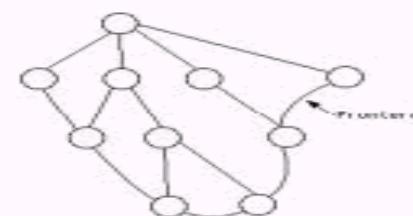
Ejercicio

- Modela el siguiente enunciado como un sistema de producción y resuélvelo:
 - Ayuda a la familia a cruzar el puente. Ten en cuenta que es de noche y necesitan la linterna para cruzar.
 - Cada miembro cruza a una velocidad distinta (1 seg., 3 seg., 6 seg., 8 seg y 12 seg).
 - El puente sólo resiste un máximo de 2 personas.
 - Un par debe cruzar a la velocidad del miembro más lento.
 - La linterna sólo dura 30 segundos.



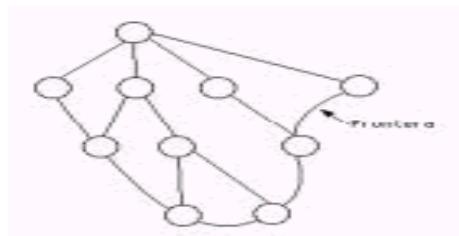
5. Estrategias de búsqueda básica (I)

- Ciclo de control básico** dentro de una estrategia de control:
 - E1 Exploración de la frontera.
 - E2 Cálculo de reglas aplicables
 - E3 Resolución de conflictos.
 - E4 Aplicación de regla y memorización de estado.
 - En el paso E1 debemos seleccionar el **mejor estado** candidato entre aquellos que no hayan sido todavía seleccionados. Hablamos pues de mantener una frontera de búsqueda, compuesta por los estados que no han sido todavía expandidos



5. Estrategias de búsqueda básica(II)

- En el siguiente paso E2 debemos de obtener la aplicabilidad de las reglas sobre el estado seleccionado, es decir, qué reglas entre todas las posibles son aplicables.
- En el paso E3 se elige la regla a aplicar definitivamente. En este paso podemos **incorporar el conocimiento para decidir qué regla nos acercará más a la solución**.
- Por último, en el paso E4 aplicamos la regla seleccionada y el resultado lo almacenamos dentro del árbol de búsqueda, actualizando éste.



5. Estrategias de búsqueda básica(III)

- Las estrategias a considerar las podemos subdividir en:
 - **Irrevocables**
 - Presenta la característica de que no se permite la vuelta atrás. Mantenemos una frontera unitaria.
 - **Tentativas**
 - La búsqueda es **multi o mono camino**. Se mantienen estados de vuelta atrás por si el estado actual no llega a buen fin.
- Los requerimientos exigibles a las estrategias empleadas son:
 - En todo momento se debe producir un avance y este debe ser dirigido.
 - Este avance debe ser metódico

5.1 Estrategia Irrevocable

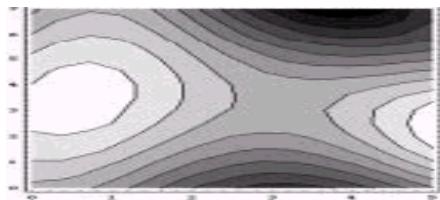
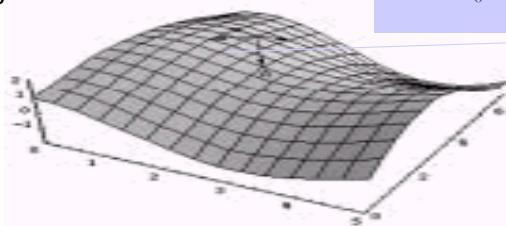
- Búsqueda irrevocable. Descenso por gradiente
 - Supuestos de partida:
 - Disponemos del suficiente conocimiento local.
 - Las equivocaciones sólo alargan la búsqueda
 - Pretendemos buscar optimalidad global a partir de la local.
 - Debemos especificar una función de evaluación $f()$ que nos proporcione un mínimo (máximo) en el estado final. En la literatura esta estrategia aparece como búsqueda por gradiente o ascenso (descenso).
 - En el ciclo de control, concretamente en el paso E3 se debe elegir aquella regla que optimiza localmente $f()$
 - En el problema del 8-puzzle, podemos definir una función de evaluación sumando el número de fichas descolocadas. En el estado final, el resultado de la función es cero, por lo que tenemos el mínimo buscado.

5.1 Estrategia Irrevocable

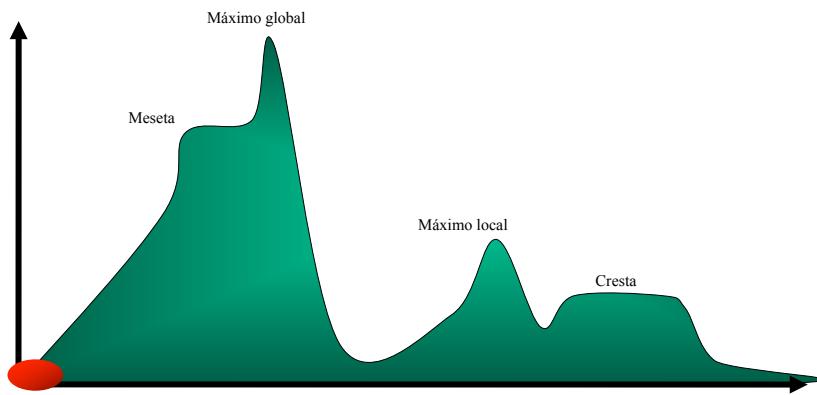
- Gráfico de la función de evaluación del problema del 8-puzzle.

Des

Valor de $f()$ en el estado actual



5.1 Problemas

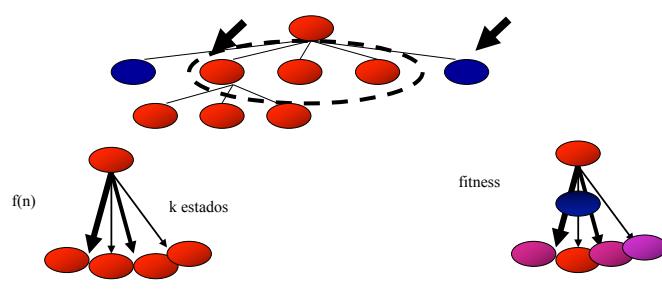


Problemas de las búsquedas irrevocables

- Mesetas
- Máximos locales
- Crestas

5.1 Otras estrategias

- Otras soluciones, para aplicarlas es necesario que no importe el camino al objetivo:
 - Ascenso por gradiente estocástico
 - Ascenso por gradiente de primera opción
 - Ascenso por gradiente de reinicio aleatório
 - Temple simulado (simulated annealing)
 - Búsqueda por haz local
 - Algoritmos genéticos



5.2.1 Estrategia Tentativa

Tentativas No informadas

- Estrategias desinformadas (No informadas)
 - “son ciegas en el sentido de que el orden en el cual la búsqueda progresá **no depende de la naturaleza de la solución que buscamos**”
- Búsqueda en profundidad
 - También conocida como primero el mejor, es una variación del conocido **backtracking**.
 - El siguiente estado a desarrollar es el de mayor profundidad en el grafo.
- Búsqueda en anchura
 - Asigna **mayor prioridad** a aquellos nodos que se encuentran a **menor profundidad** en el grafo. De esta manera nos estamos asegurando que la búsqueda se realiza por todo el grafo.
- Coste uniforme
 - Esta estrategia selecciona aquel nodo tal que la **suma de los costes de aplicación** de las reglas en el camino desde el nodo inicial sea mínima. Esta estrategia es similar al procedimiento en anchura cuando el coste de aplicación de cada regla es unitario.
- **Pregunta:** En grafos finitos ¿Se asegura obtener la solución óptima con todas las estrategias anteriores?

5.2.2 Estrategia Tentativa

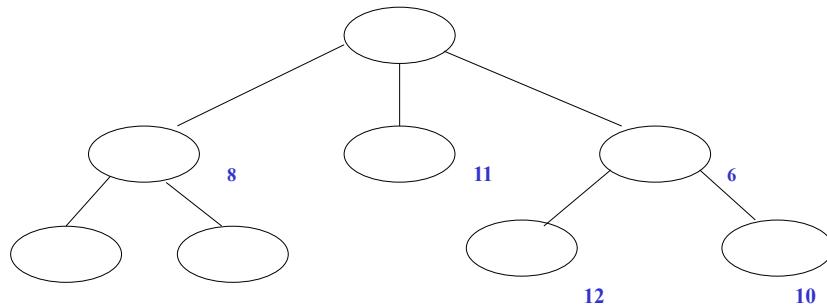
Tentativas Informadas

- Al contrario de las “ciegas” las informadas sí que van a disponer de información de lo **prometedor que es un nodo** para llegar desde él a la solución.
- Estimación de lo que nos va a costar **llegar a la solución óptima**. Esta función la vamos a denominar **heurística** $h(n)$.
- En general vamos a disponer de una función $f(n)$ que va a estimar el coste del camino de coste mínimo desde el nodo inicial B_0 hasta un nodo objetivo, condicionado este camino a pasar por n .
- En esta estrategia el criterio de selección de un nodo de la lista de frontera es el de menor valor de $f()$.
- En la literatura aparecen estas estrategias como **Best-first** (primero el mejor).

5.2.2 Estrategia Tentativa

- Algoritmo de búsqueda **tentativa informada**

El nodo a expandir es aquel que proporciona el coste más pequeño para ir desde el nodo inicial hasta un nodo objetivo



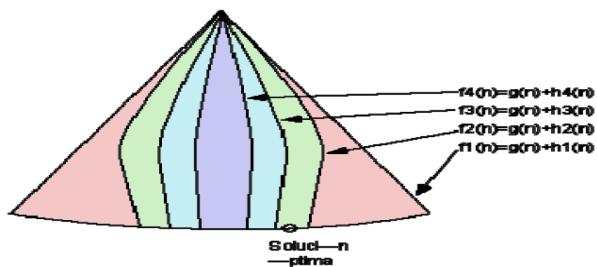
5.2.2 Estrategias Informadas. Funciones de Evaluación

- Algoritmos A (de aditivos).
 - Presentan una función de evaluación de la forma: $f(n) = g(n) + h(n)$
 - dónde:
 - $g(n)$ Estimación del coste del camino de **coste mínimo** desde el estado inicial hasta el nodo n .
 - $h(n)$ Estimación del **coste del camino** de coste mínimo desde n hasta algún nodo objetivo o meta. Esta función incluye el conocimiento heurístico sobre el problema a resolver.
- Nuestro objetivo
 - Diseñar la heurística
 - Elección compleja
 - Podemos llegar a dejar fuera el óptimo
 - Podemos explorar demasiados nodos

5.2.2 Estrategias Informadas. Funciones de Evaluación (II)

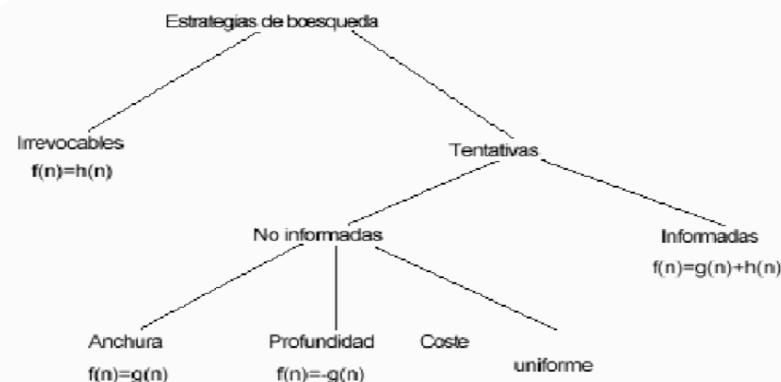
En la figura:

- $h_1(n) = 0$,
- $h_2 < h_3 < h_4$.



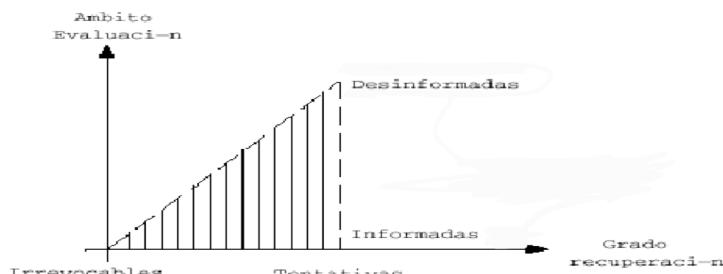
- ¿Podemos diseñar una función heurística de tal manera que el óptimo se queda fuera?
- ¿Cuáles son las condiciones que debe cumplir dicha función para que se garantice la optimalidad?

6. Esquema tipos básicos de búsqueda.



6. Comparativa tipos de búsqueda

- Comparación de las estrategias irrevocables, desinformadas e informadas
 - Los ejes de la gráfica mostrada en la figura tienen el siguiente significado:
 - Alcance en la recuperación:** Grado en el que una estrategia permite recuperación de alternativas suspendidas previamente.
 - Ambito de evaluación:** Número de alternativas consideradas en cada decisión.



- Ejercicio - Situar en la gráfica anterior las siguientes estrategias de búsqueda:
Irrevocables, Tentativas Informadas, Tentativas Desinformadas

7. Estrategias híbridas

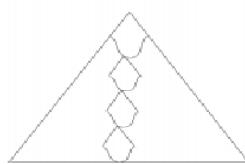
- Combinación informada + backtracking



Otra posible combinación de estas dos estrategias puede ser realizar una búsqueda informada local dentro de una búsqueda backtracking global

- Combinación informada + irrevocable:

Búsqueda informada pero se eliminan aquellos nodos de la lista frontera que son menos prometedores



8.1 Búsqueda heurística. Conceptos básicos

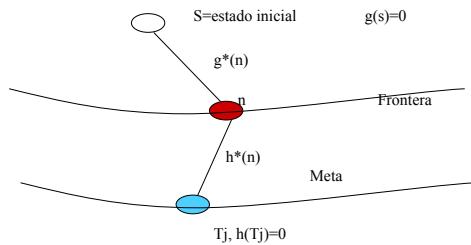
- **Compleitud** encuentra una solución si ésta existe
- **Admisibilidad** encuentra la solución óptima
- **Dominación**
 - Un algoritmo A_1 es *dominante* sobre A_2 si cada nodo expandido por A_1 es también expandido por A_2
- **Optimalidad**
 - Un algoritmo es el *óptimo* de un conjunto de algoritmos si es el dominante sobre todos los algoritmos del conjunto (es el que menos nodos expande)
- **La solución del problema vendrá dada por el camino de menor coste entre el estado inicial (s) y cualquier estado objetivo (t_j)**

8.2 Búsqueda A*. Búsqueda óptima

$$A^* : f^*(n) = g^*(n) + h^*(n)$$

- $g^*(n) = c(s, n)$
 - Coste del camino de **coste mínimo** desde el nodo inicial s al nodo n .
 - Estimada por $g(n)$
- $h^*(n)$
 - Coste del camino de **coste mínimo** de todos los caminos desde el nodo n a cualquier estado solución t_j
 - Estimada por $h(n)$
- $f(n)$
 - Coste del camino de coste mínimo desde el nodo inicial hasta un nodo solución condicionado a pasar por n
 - $f(n) = g^*(n) + h^*(n)$
 - Estimada por $f(n)$
- C^*
 - Coste del camino de coste mínimo desde el nodo inicial a un nodo solución.

8.2 Búsqueda A*. Búsqueda óptima



- $g(n) \geq g^*(n); g(n_j) = g(n_i) + c(n_i, n_j)$ n_j es sucesor de n_i
- Si tenemos una función $h(n) = 0$ y el coste de cada regla es unitario,
- ¿Qué tipo de exploración realizaremos?
- $f^*(s) = h^*(s) = g^*(t_j) = f^*(t_j) = C^*$ $\forall t_j \in \Gamma$
- ¿ $f^*(n) = C^*$?

8.2 Búsqueda A*. Búsqueda óptima

- Una función heurística $h(n)$ se dice que es **admissible** (garantiza la obtención de un camino de coste mínimo hasta un objetivo) cuando se cumple:

$$h(n) \leq h^*(n) \quad \forall n$$

8.2 Pseudocódigo A*

```

Alg A*
    listaInterior = vacío
    listaFrontera = inicio

    mientras listaFrontera no esté vacía
        n = obtener nodo de listaFrontera con menor f(n) = g(n) + h(n)
        listaFrontera.del(n)
        listaInterior.add(n)

        si n es meta
            devolver
            reconstruir camino desde la meta al inicio siguiendo los punteros
        fsi

        para cada hijo m de n que no esté en listaInterior
            g'(m) = n.g + c(n, m) //g del nodo a explorar m

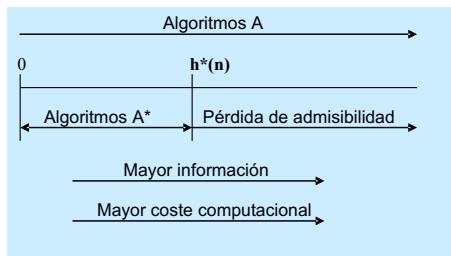
            si m no está en listaFrontera
                almacenar la f, g y h del nodo en (m.f, m.g, m.h)
                m.padre = n
                listaFrontera.add(m)
            sino si g'(m) es mejor que m.g //Verificamos si el nuevo camino es mejor
                m.padre = n
                recalcular f y g del nodo m
            fsi

            fpara
        fmientras
        devolver no hay solución
    falg

```

8.3 Nivel de información heurístico

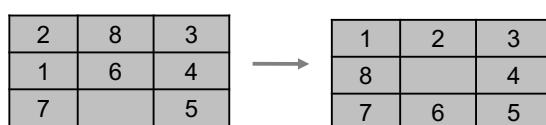
- En general el **nivel de información** de las heurísticas permite **encontrar antes la solución**, pero tiene la desventaja de requerir un **mayor coste computacional** para su cálculo. La figura muestra los **límites de la admisibilidad** en los algoritmos tipo A:



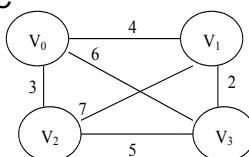
- $h(n) = 0$
- $h(n) \leq h^*(n)$
- $h(n) > h^*(n)$

8.4 Ejemplos de heurísticas

- Heurísticas para el 8-puzzle



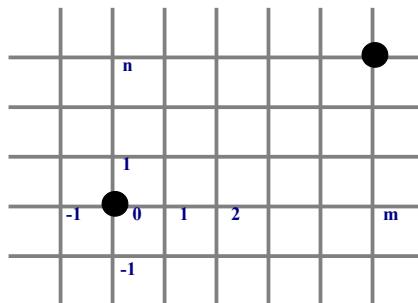
- h_1 = número de piezas mal colocadas
- h_2 = suma de las distancias de las piezas a sus posiciones en el objetivo (distancia de Manhattan)
- Heurística para el VC



h = suma de las distancias de las ciudades aun no visitadas a sus vecinos más cercanos

8.4 Problemas de camino mínimo

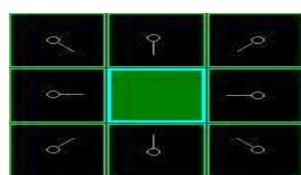
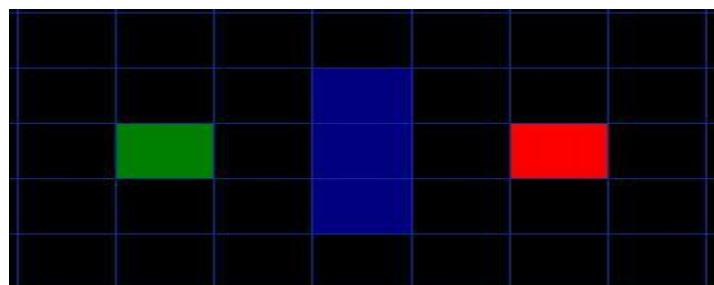
- Coste actual óptimo
 - $g^*((x, y)) = |x| + |y|$
- Heurística admisible
 - $h_1((x,y)) = \sqrt{(m-x)^2 + (n-y)^2}$
- Heurística óptima
 - $h^*((x,y)) = |m-x| + |n-y|$



T2. Estrategias de búsqueda

5

8.4 Un ejemplo detallado (I)



$g(n)$: coste de moverse entre nodos.
recto 10, diagonal 14

$h(n)$: distancia de Manhattan
 $|m-x| + |n-y|$
Atención, no admisible para 8-con.
utilizada por simplicidad pero...

T2. Estrategias de búsqueda

6

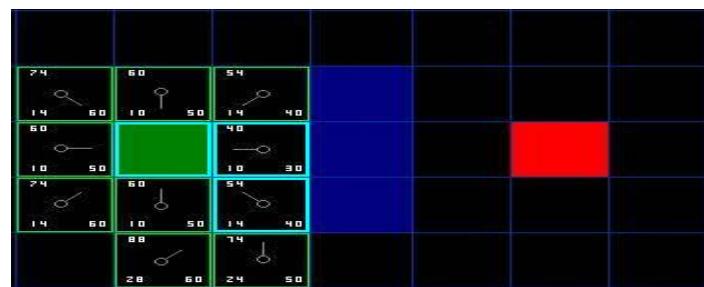
8.4 Un ejemplo (II)



T2. Estrategias de búsqueda

7

8.4 Un ejemplo (III)

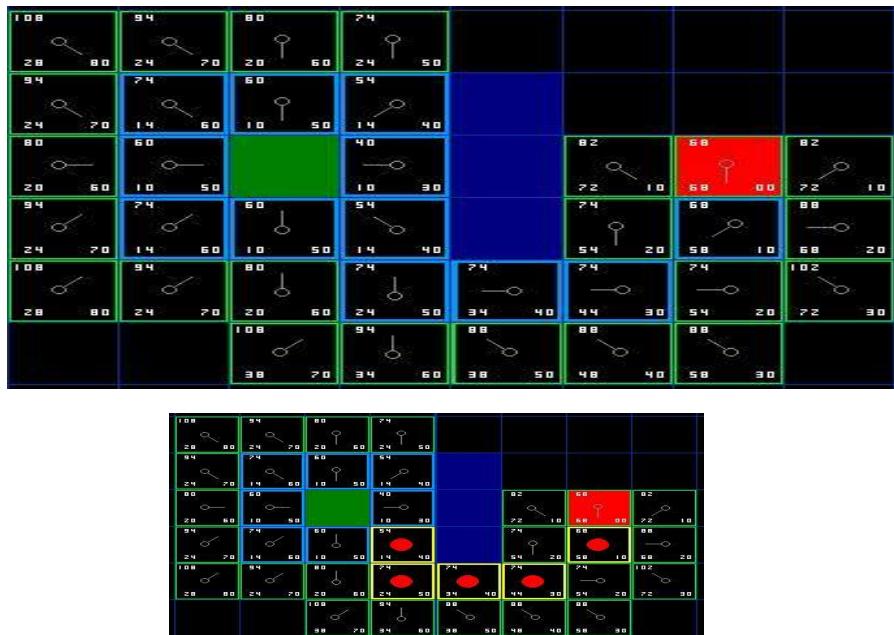


Ejercicio: continua la traza...

T2. Estrategias de búsqueda

8

8.4 Un ejemplo (IV)



T2. Estrategias de búsqueda

9

8.5 Inconvenientes de mantener la admisibilidad

- El **mantenimiento de la admisibilidad** fuerza al algoritmo a **consumir mucho tiempo** en discriminar caminos cuyos costes no varían muy significativamente
- Principal desventaja: se queda sin espacio debido a que mantiene todos los nodos generados en memoria
- No es práctico para problemas grandes
- Dos soluciones
 - Algoritmos que mejoran el coste espacial: A*PI (A* por profundización iterativa), A* SRM (A* acotada por memoria simplificada), búsqueda primero el mejor recursiva
 - Aumentar la velocidad a costa de una pérdida acotada de calidad → **técnicas de relajación de la restricción de optimalidad**

T2. Estrategias de búsqueda

10

8.6 Relajación de la restricción de optimalidad

- Técnica de ajuste de pesos
 - El objetivo de esta técnica es definir una función $f()$ ponderada, $f_w()$, como alternativa a la utilizada en A^*
 - $$f_w(n) = (1-w)g(n)+w h(n)$$
 - $g(n)$
 - Proporciona la componente en anchura de la búsqueda.
 - $h(n)$
 - Nos indica la proximidad al objetivo.
- Variando de forma continua w dentro del rango $0 \leq w \leq 1$ obtenemos **estrategias mixtas intermedias**.
- Si $h(n)$ es admisible tenemos que:
 - En el rango $0 \leq w \leq 1/2$, A^* con $f_w(n)$ también es admisible.
 - Dependiendo de la diferencia existente entre $h(n)$ y $h^*(n)$, A^* con $f_w(n)$ puede perder la admisibilidad en el rango $1/2 < w \leq 1$

8.6 Relajación de la restricción de optimalidad

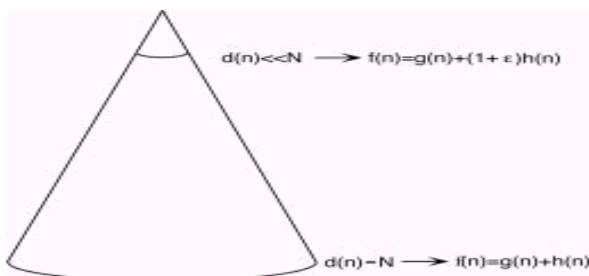
- Técnica de la admisibilidad- ϵ
 - Objetivo: aumentar la velocidad de búsqueda a costa de obtener una solución subóptima
 - Un algoritmo es **admissible- ϵ** cuando para cualquier grafo termina siempre dando como resultado una solución cuyo coste no excede del coste óptimo, C^* , por un factor $1+\epsilon$:
- $$f(\text{sol}) \leq (1+\epsilon)C^*$$
- Técnicas de admisibilidad- ϵ
 - Ponderación Dinámica
 - Estimación de coste de búsqueda

8.6 Relajación de la restricción de optimalidad

- Técnica de ponderación dinámica o APD

$$f(n) = g(n) + h(n) + \epsilon [1 - d(n)/N] h(n)$$

- $d(n)$ es la profundidad del nodo n y N nos proporciona la profundidad de un nodo solución (se supone conocida).
- ¿Qué ocurre en los niveles iniciales?
- ¿Y en los cercanos a la solución?



8.6 Relajación de la restricción de optimalidad

- Algoritmo de estimación de coste de búsqueda A_ϵ^* .
 - Utiliza una lista adicional denominada **Lista_focal (L_f)**
 - Esta lista es una sublistas de Lista_Frontera (LF) que contiene únicamente aquellos nodos cuya $f(n)$ no excede del mejor valor de cualquier $f(n)$ dentro de la Lista_Frontera por un factor $(1+\epsilon)$:

$$L_f = \{n : f(n) \leq (1+\epsilon) \min(f(m))\}$$

$$m \in LF$$
 - A_ϵ^* opera de forma idéntica al algoritmo A^* salvo que **selecciona aquel nodo de Lista_focal con menor valor de $H_f(n)$, una segunda heurística**, además de $h(n)$, que estima el coste computacional requerido para completar la búsqueda a partir del nodo n

8.6 Relajación de la restricción de optimalidad

- Comparación de algoritmos
 - El algoritmo de ponderación dinámica es más sencillo, **pero únicamente es aplicable a problemas donde se conoce la profundidad** en la cual nos va a aparecer la solución, o disponemos de una cota superior de dicha profundidad. Sólo en estos casos se garantiza la admisibilidad- ϵ
 - En cuanto al algoritmo A_ϵ^* la separación en dos heurísticas permite incorporar estimaciones de coste no integradas con las funciones $g(n)$ y $h(n)$ (por ejemplo, proximidad a la solución)

Tema 2. Estrategias de búsqueda.

Bibliografía

- Stuart Russell, Peter Noving. "Inteligencia Artificial. Un enfoque Moderno" Ed. Pearson. Prentice Hall. 2004.



Tema 3. Búsqueda en juegos y Búsqueda para problemas de satisfacción de restricciones

Búsqueda en juegos

- Búsqueda en juegos con adversario
 - Juegos como problemas de búsqueda
 - Estrategias básicas de búsqueda en juegos
 - Técnicas complementarias

Juegos

- XVIII (1760), [Wolfgang Kempelen](#): ajedrecista mecánico



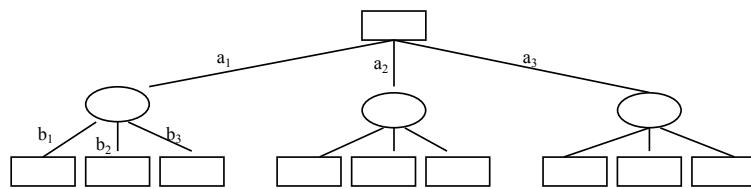
- XIX, [Charles Babbage](#): máquina analítica, ajedrez
- '50 Alan Turing, ajedrez
<http://www.chessgames.com/perl/chessgame?gid=1356927>
- 1963, [Arthur Samuel](#), damas
- 1997, [Deep Blue](#), ajedrez, gana al campeón mundial de ajedrez
- Ajedrez
 - Cada jugada ▶ 35 posibles movimientos
 - Cada partida ▶ aproximadamente 100 jugadas
 - Búsqueda exhaustiva imposible**

Juegos como problemas de búsqueda

- Imposible generar todo el árbol de búsqueda
 - Generar hasta un determinado nivel de profundidad
 - Aplicar alguna [función de evaluación f\(N\)](#)
 - Devuelve un valor numérico que indica cómo de bueno es un estado
 - MAX maximizará esta función y MIN minimizará** dicha función.
 - En algunos casos la función nos puede devolver valores como PIERDE, GANA o EMPATA, siempre referidos a MAX
 - Objetivo del análisis del árbol: determinar valor del nodo raíz (inicio de la jugada). A este valor se le denomina valor MiniMax**

Juegos como problemas de búsqueda

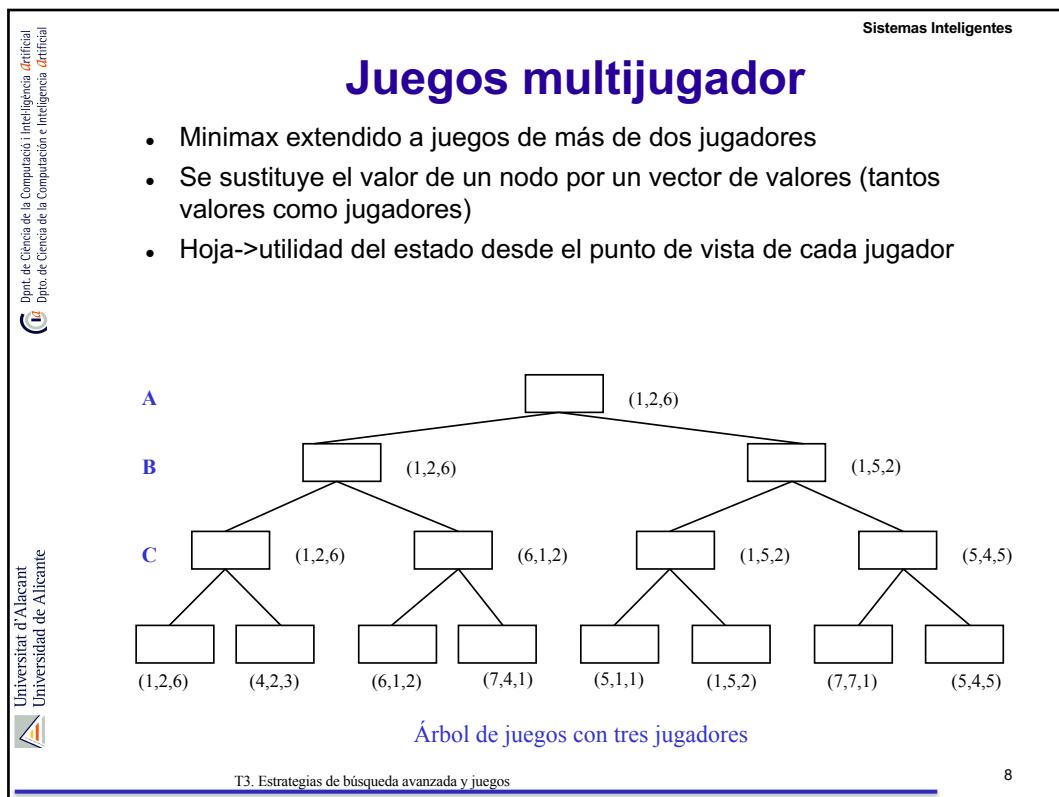
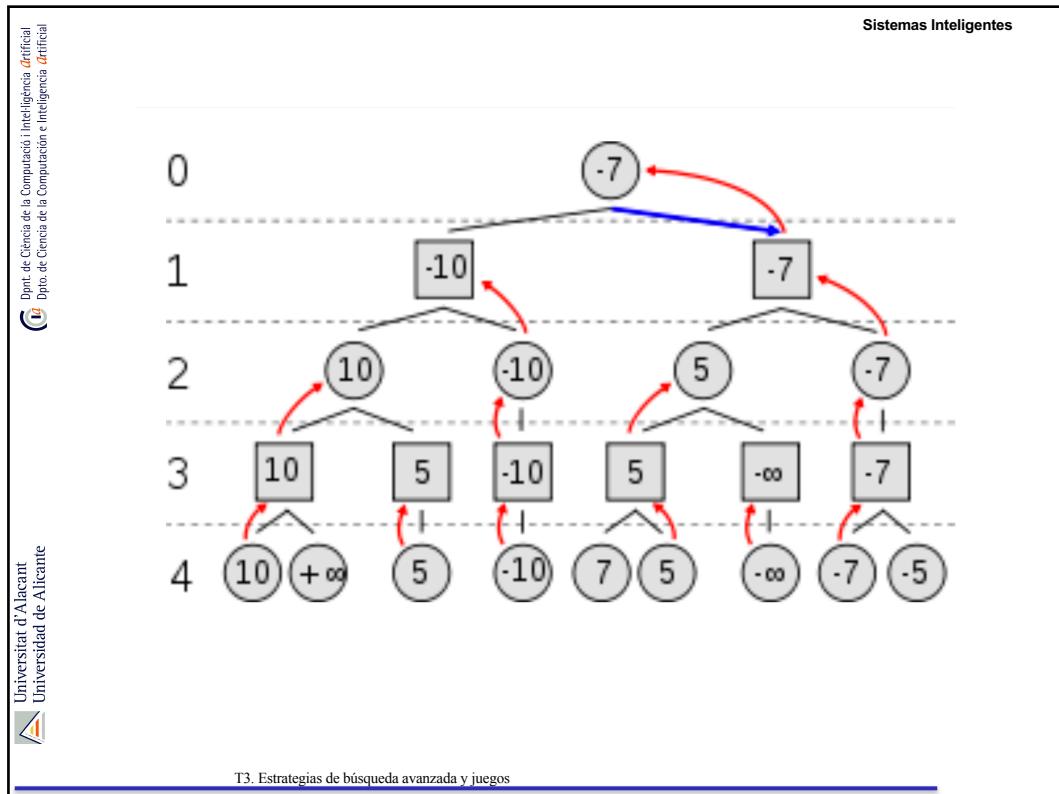
- Estado (N): configuración del juego en un momento dado
- Árbol de juego. Cada arista de ese árbol indica un posible movimiento. Una rama completa contempla una posible jugada



- En cada nivel se van alternando los jugadores
- Factor de ramificación (B): número de posibles movimientos que se pueden realizar

Estrategia exhaustiva: MiniMax

- Genera todos los nodos del árbol hasta la profundidad deseada
- Evalúa cada nodo hoja
- Asigna un valor al nodo raíz
 - o si la decisión la toma el jugador MIN, asociar a ese nodo el mínimo de los valores de sus hijos, y el máximo en caso de MAX
- Cuando decimos que aplicamos el algoritmo lo que realizamos es calcular el valor de $V(N)$ de un determinado nodo.



Estrategia de poda: α - β

La poda alfa beta es una técnica de búsqueda que reduce el número de nodos evaluados en un árbol de juego por el Minimax.

α es el valor de la mejor opción hasta el momento a lo largo del camino para MAX, esto implicará por lo tanto la elección del valor más alto

β es el valor de la mejor opción hasta el momento a lo largo del camino para MIN, esto implicará por lo tanto la elección del valor más bajo.

El valor MiniMax de un nodo estará siempre acotado

$$\alpha \leq V(N) \leq \beta$$

Al principio inicializamos $\alpha = -\infty$ y $\beta = \infty$ al no tener ninguna evidencia.

Esta búsqueda alfa-beta va actualizando el valor de los parámetros según se recorre el árbol. El método realizará la poda de las ramas restantes cuando el valor actual que se está examinando sea peor que el valor actual de α o β para MAX o MIN, respectivamente.

```

Algoritmo  $\alpha$ - $\beta$  -  $V(N, \alpha, \beta)$ 
Entrada: Nodo  $N$ , valores  $\alpha$  y  $\beta$ .
Salida: Valor minimax de dicho nodo.

Si  $N$  es nodo hoja entonces devolver  $f(N)$ .
sino
  Si  $N$  es nodo MAX entonces
    Para  $k = 1$  hasta  $b$  hacer
       $\alpha = \max[\alpha, V(N_k, \alpha, \beta)]$ 
      Si  $\alpha \geq \beta$  entonces devolver  $\beta$  FinSi.
      Si  $k = b$  entonces devolver  $\alpha$  FinSi.
    FinPara.
  sino
    Para  $k = 1$  hasta  $b$  hacer
       $\beta = \min[\beta, V(N_k, \alpha, \beta)]$ 
      Si  $\alpha \geq \beta$  entonces devolver  $\alpha$  FinSi.
      Si  $k = b$  entonces devolver  $\beta$  FinSi.
    FinPara.
  FinSi
FinSi

```

<http://homepage.ufsp.pt/jtorres/ensino/ia/alfabeta.html>

T3. Estrategias de búsqueda avanzada y juegos

Técnicas complementarias

- Uso de movimientos de libro
- Espera del reposo
- Técnica de bajada progresiva
- Poda heurística
- Continuación heurística

T3. Estrategias de búsqueda avanzada y juegos

Uso de movimientos de libro

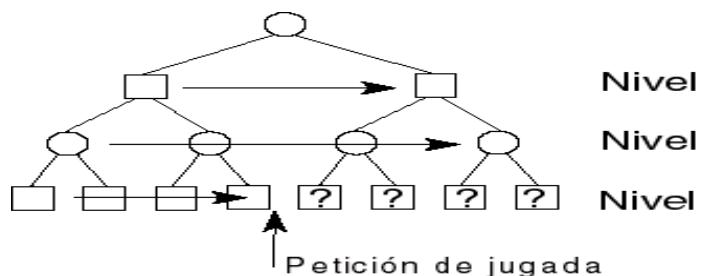
- Imposible seleccionar un movimiento consultando la configuración actual del juego en un catálogo y extrayendo el movimiento correcto.
- Razonable para algunas partes de ciertos juegos.
 - En ajedrez, tanto la secuencia de apertura como los finales están muy estudiados.
- **El rendimiento del programa puede mejorarse si se le proporciona una lista de movimientos** (movimientos de libro) que deberían realizarse en dichos casos.
- Se usa el libro en las aperturas y los finales combinado con el procedimiento MiniMax para la parte central de la partida
- **Conocimiento + búsqueda**

Espera del reposo

- Busca evitar el efecto horizonte
- Condición adicional de corte de recursión en minimax sería alcanzar una situación estable
- Si se evalúa un nodo y éste cambia su valor de manera drástica después de explorar un nivel más, la búsqueda debe continuar

Técnica de bajada progresiva

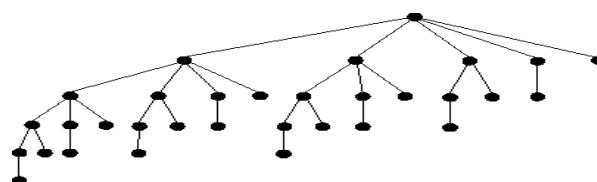
- **Restricciones de tiempo:** algoritmos presentados anteriormente no adecuados.
- **Técnica de bajada progresiva**
 - Recorrer nodos por niveles
 - Al llegar la petición de jugada, devolver la solución del último nivel que se haya completado.



Poda heurística

- Objetivo: **reducir B** desarrollando únicamente los **mejores movimientos** de cada nivel.
- **$g(N)$:** Función adicional de evaluación
 - De bajo coste.
 - Versión simplificada de $f(N)$.
 - Reordenación de nodos: el primer nodo de un nivel es el de mayor $g(N)$.
- Factor de ramificación:

$$\text{Factor}(Nodo) = \text{Factor}(\text{Padre}(Nodo)) - \text{Rango}(Nodo)$$



Continuación heurística

- Intento de evitar el **efecto horizonte**.
 - Provocado por la limitación en profundidad: solo se puede tener conocimiento hasta la profundidad seleccionada.
- Secuencia:
 - Desarrollar en anchura hasta un determinado nivel.
 - Seleccionar un subconjunto de nodos terminales para desarrollar búsquedas más profundas.
 - Selección dada por un conjunto de heurísticas directamente relacionadas con el juego.

Búsqueda para problemas de satisfacción de restricciones

- Formulación de CSPs como redes de restricciones
- Ejemplos
- Métodos de resolución:
 - Esquema backtracking
 - Esquema Forward Checking
 - Esquema de propagación de restricciones

Problemas de satisfacción de restricciones (CSP)

Conjunto de **variables** definidas sobre **dominios** finitos y conjunto de **restricciones** definidas sobre subconjuntos de dichas variables.

(V, D, ρ)

→ un conjunto de **variables**

$$V = \{V_1, V_2, \dots, V_n\} \quad V = \{V_i\}_{i=1..n}$$

→ definidas sobre **dominios** discretos D_i (conjunto finito de posibles valores)

$$D = \{D_1, D_2, \dots, D_n\} \quad D = \{D_j\}_{j=1..n}$$

→ un conjunto de **restricciones** definidas sobre subconjuntos de dichas variables

$$\rho = \{\rho_1, \rho_2, \dots, \rho_n\} \quad \rho = \{\rho_k\}_{k=1..n}$$

Problemas de satisfacción de restricciones (CSP)

Ejemplo:

$$X::\{1,2\}, Y::\{1,2\}, Z::\{1,2\}$$

$$X = Y, X \neq Z, Y > Z$$

Solución: encontrar asignaciones de valor a las variables que satisfagan todas las restricciones.

$$(X = 2, Y = 2, Z = 1)$$

Solución al problema: la relación n-aria que satisface todas las restricciones del problema

Dependiendo de los requerimientos del problema hay que encontrar todas las soluciones o sólo una

Redes de restricciones

- Un CSP se puede representar como un grafo.
- Sobre el grafo se puede definir una red de restricciones:

Quintupla $\langle V, E, c, l, a \rangle$

- V : conjunto de nodos.
- E : conjunto de aristas.
- $c: E \rightarrow V^k$, $k \leq n$; función de asignación de aristas a tuplas de nodos.
- $l: E \rightarrow \rho$; función de asignación de aristas a restricciones.
- a : permutación que define el orden de selección para resolver el problema.

CSP binario

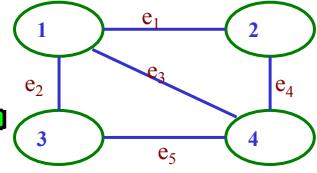
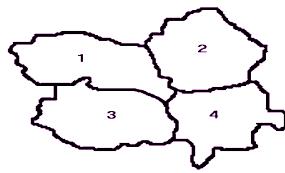
- Variables
$$V = \{V_1, V_2, \dots, V_n\}$$
- Dominios discretos y finitos
$$D = \{D_1, D_2, \dots, D_n\}$$
- Restricciones binarias

$$\{R_{ij}\}$$

Todo problema **n-ario** se puede formular como un problema **binario**

- Ejemplos de CSP binarios:
 - Coloreado de mapas
 - Asignación de tareas para un robot
 - N-reinas
 - Generación de crucigramas

Ejemplo: Coloreado de mapas



$$V = \{V_1, V_2, V_3, V_4\}$$

$$D_i = \{\text{rojo, azul, verde}\}, \forall i, 1 \leq i \leq 4$$

$$E = \{e_1, e_2, e_3, e_4, e_5\}$$

$$\rho_k(V_i, V_j) = \{<v_i, v_j> | v_i \in D_i, v_j \in D_j, v_i \neq v_j\} \forall k, 1 \leq k \leq 5$$

$$c(e_1) = <V_1, V_2>, c(e_2) = <V_1, V_3>, c(e_3) = <V_1, V_4>, \dots$$

$$l(e_j) = \{<\text{azul, verde}>, <\text{azul, rojo}>, <\text{verde, azul}>, <\text{verde, rojo}>, <\text{rojo, azul}>, <\text{rojo, verde}>\} \forall j, 1 \leq j \leq 5$$

$$a = \{V_1, V_4, V_2, V_3\}$$

Generación de crucigramas

- Dada una rejilla y un diccionario, construir un crucigrama legal

Slot horizontal de tres letras

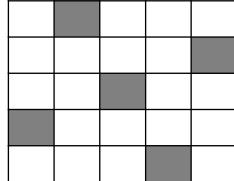
C	O	L
O		
Z		

4 palabras de 1 letra
4 palabras de 3 letras
2 palabras de 5 letras

- Formulación:
 - variables** : grupo de casillas para una palabra (*slots*)
 - dominios** : palabras del diccionario con la longitud adecuada
 - restricciones** : misma letra en la intersección de dos palabras
- Características :
 - CSP binario, discreto (dominios grandes)

N-reinas

- Posicionar n reinas en un tablero de ajedrez $n \times n$, de forma que no se ataquen



$n = 5$

- Formulación: (1 reina por fila)
 - variables : reinas, X_i reina en la fila i -ésima
 - dominios : columnas posibles $\{1, 2, \dots, n\}$
 - restricciones :
 - la misma columna
 - la misma diagonal
- Características :
 - Dominios discretos y restricciones binarias

Criptoaritmética

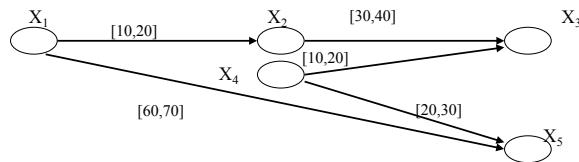
- Sustituir cada letra por un dígito distinto (distinta cifra, distinta letra) de manera que la suma sea correcta
- Formulación:
 - variables : $G, O, T, A, U, C_1, C_2, C_3$
 $(C_1, C_2, C_3$ variables de acarreo)
 - dominios : $O, T, U \in \{0, \dots, 9\}$
 $G, A \in \{1, \dots, 9\}$
 $C_1, C_2, C_3 \in \{0, \dots, 4\}$
 - restricciones :
 - letras distintas $G \neq O, G \neq T, \dots, A \neq U$
 - suma correcta

(unidades) (decenas) (centenas) (unidades millar)	$5^*A = 10^*C_1 + A$ $5^*T + C_1 = 10^*C_2 + U$ $5^*O + C_2 = 10^*C_3 + G$ $5^*G + C_3 = A$
--	--
- Características :
 - Dominios discretos y restricciones múltiples

$$\begin{array}{r}
 \text{GOTA} \\
 \text{GOTA} \\
 + \text{GOTA} \\
 \hline
 \text{AGUA}
 \end{array}$$

Restricciones temporales

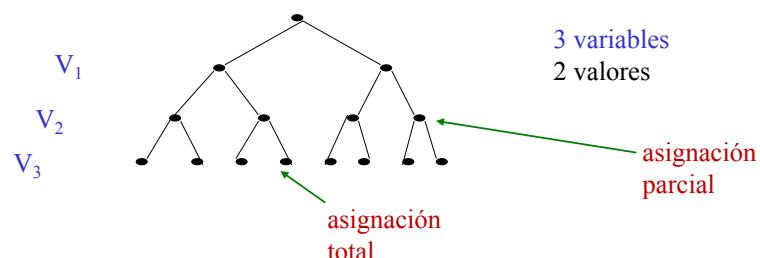
- Dado un conjunto de sucesos que ocurren en intervalos temporales con ciertas relaciones, encontrar un asignación temporal consistente



- Formulación:
 - variables** : sucesos
 - dominios** : intervalo temporal para cada suceso
 - restricciones** : distancia temporal permitida entre sucesos; relaciones temporales antes, después, solapado, etc.
- Características :
 - Dominios continuos y restricciones binarias

Árbol de interpretaciones

- Partimos de un nodo raíz que supervisa el proceso.
- Cada nivel corresponde a una asignación de valor para una característica de datos. El orden de descenso viene especificado por a.
- Cada nodo identifica una posibilidad de asignación (Variable, valor).
- La solución se construye de forma incremental de tal forma que cada hoja es una interpretación.



Métodos de resolución

Búsqueda

Generación y test : generar de forma sistemática y exhaustiva cada una de las posibles asignaciones a las variables y comprobar si satisfacen todas las restricciones. Hay que explorar el espacio definido por el producto cartesiano de los dominios de las variables.

Backtracking : se trata de construir la solución de forma gradual, instanciando variables en el orden definido por la permutación dada



Backjumping: parecido al BT pero el retroceso no se hace a la variable instanciada anteriormente sino a la variable más profunda que está en conflicto con la variable actual.



Explorar el espacio de estados hasta encontrar una solución, demostrar que no existe o agotar los recursos

Métodos de resolución

Inferencia

- **Consistencia de arco**
- Consistencia de caminos
- K-consistencia

Deducir un problema equivalente que sea más fácil de resolver

Algoritmos híbridos

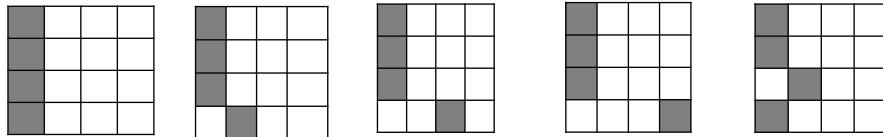
- **Forward Checking**
- Maintaining Arc Consistency
- Heurísticas

Combinación de las aproximaciones anteriores. Sobre un esquema de búsqueda se incorporan métodos de inferencia

Generación y test

Estrategia:

- generación y test de todas las asignaciones totales posibles
 1. Generar una asignación de todas las variables
 2. Comprobar si es solución. Si es, stop, sino ir a 1



Eficiencia:

- es muy poco eficiente
- genera muchas asignaciones que violan la misma restricción

Backtracking

Estrategia:

- Construir una solución parcial : asignación parcial que satisface las restricciones de las variables involucradas
- Extender la solución parcial, incluyendo una variable cada vez hasta llegar una solución total
- Si no se puede extender: backtracking
 - cronológico: se elimina la última decisión
 - no cronológico : se elimina una decisión anterior

Backtracking recursivo

```

procedimiento Backtracking( $k, V[n]$ ) ; Llamada inicial: Backtracking(1,  $V[n]$ )
inicio
     $V[k] = Selección(d_k)$  ; Selecciona un valor de  $d_k$  para asignar a  $x_k$ 
    si Comprobar( $k, V[n]$ ) entonces
        si  $k = n$  entonces
            devolver  $V[n]$  ; Es una solución
        si no
            Backtraking( $k + 1, V[n]$ )
        fin si
    si no
        si quedan_valores( $d_k$ ) entonces
            Backtraking( $k, V[n]$ )
        si no
            si  $k = 1$  entonces
                devolver  $\emptyset$  ; Fallo
            si no
                Backtraking( $k - 1, V[n]$ )
            fin si
        fin si
    fin si
fin Backtracking

```

Limitaciones del backtracking

- **Trashing e inconsistencia de nodo**

Relacionado con las **restricciones unarias**. Sigue cuando un dominio contiene un valor que no satisface una restricción unaria.

- **Inconsistencia de arista**

Relacionado con las **restricciones binarias**. Sigue cuando existe una restricción binaria entre dos variables de tal forma que para un determinado valor de la primera variable no existe ninguna asignación posible para la segunda.

- **Dependencia de la ordenación**

El orden de selección de las variables es un factor crítico. Se han desarrollado diversas heurísticas de selección de variable y de valor.

Variable: Orden estático y Orden dinámico

Valor: p.e. los que conducen a un CPS más simple

Forward checking

- En cada etapa de la búsqueda, FC comprueba hacia delante la asignación actual con todos los valores de las futuras variables que están restringidas con la variable actual.
- Los valores de las variables futuras que son inconsistentes con la asignación actual son temporalmente eliminados de sus dominios.
- Si el dominio de una variable futura se queda vacío, la instanciación de la variable actual se deshace y se prueba con un nuevo valor. Si ningún valor es consistente, entonces se lleva a cabo el backtracking cronológico.

Forward checking

1. Seleccionar x_i .
2. Instanciar $x_i \leftarrow a_i : a_i \in D_i$.
3. Razonar hacia adelante (forward-check):
 - Eliminar de los dominios de las variables (x_{i+1}, \dots, x_n) aún no instanciadas, aquellos valores inconsistentes con respecto a la instanciación (x_i, a_i) , de acuerdo al conjunto de restricciones.
4. Si quedan valores posibles en los dominios de todas las variables por instanciar, entonces:
 - Si $i < n$, incrementar i , e ir al paso (1).
 - Si $i = n$, salir con la solución.
5. Si existe una variable por instanciar, sin valores posibles en su dominio, entonces retractar los efectos de la asignación $x_i \leftarrow a_i$. Hacer:
 - Si quedan valores por intentar en D_i , ir al paso (2).
 - Si no quedan valores:
 - Si $i > 1$, decrementar i y volver al paso (2).
 - Si $i = 1$, salir sin solución.

Forward checking

```

funcion FC(i variable): booleano
    para cada a  $\in$  factibles[i] hacer
        Xi  $\leftarrow$  a
        si i=N solución retorna CIERTO
        sino
            si forward (i,a)
                si FC(i+1) retorna CIERTO
                Restaurar (i)
            retorna FALSO
funcion forward(i variable, a valor): booleano
    para toda j=i+1 hasta N hacer
        Vacío  $\leftarrow$  CIERTO
        para cada b  $\in$  factibles[j] hacer
            si (a,b)  $\in$  Rij vacío  $\leftarrow$  FALSO
            sino eliminar b de factible[j]
            Añadir b a podado[j]
        si vacío retorna FALSO
        retorna CIERTO
procedimiento restaura(i variable)
    para toda j=i+1 hasta N hacer
        para todo b  $\in$  podado[j] hacer
            si Xi responsable filtrado b
            Eliminar b de podado[j]
            Añadir b a factible[j]

```

T3. Estrategias de búsqueda avanzada y juegos

37

Forward Checking

Ejemplo Forward Checking:

Variables *x, y*
 $Dx = Dy = \{1,2,3,4,5\}$
 Restricción $x < y - 1$

Inicialmente $CDx = CDy = \{1,2,3,4,5\}$

Si asignamos $x = 2$, entonces:
 Los únicos valores que puede tomar *y* son 4, 5
 por tanto $CDy = \{4,5\}$

..

..

Si asignamos $x = 4$, entonces:
 No hay asignación posible compatible con la restricción.
 por tanto $CDy = \{\}$
 Deshacer $x = 4$ y backtracking

Propagación de Restricciones

- Transformar el problema en otro más sencillo sin inconsistencias de arco.
- Propiedad de consistencia de arista
 $c(e_p) = \langle V_i, V_j \rangle$ es consistente si y sólo si para todo valor asignable a V_i existe al menos un valor en V_j que satisface la restricción asociada a la arista.
- Un CSP puede transformarse en una red consistente mediante un algoritmo sencillo (AC3) que examina las aristas, eliminando los valores que causan inconsistencia del dominio de cada variable.
- Después del proceso:
 - No hay solución
 - Hay más de una solución
 - Hay una única solución

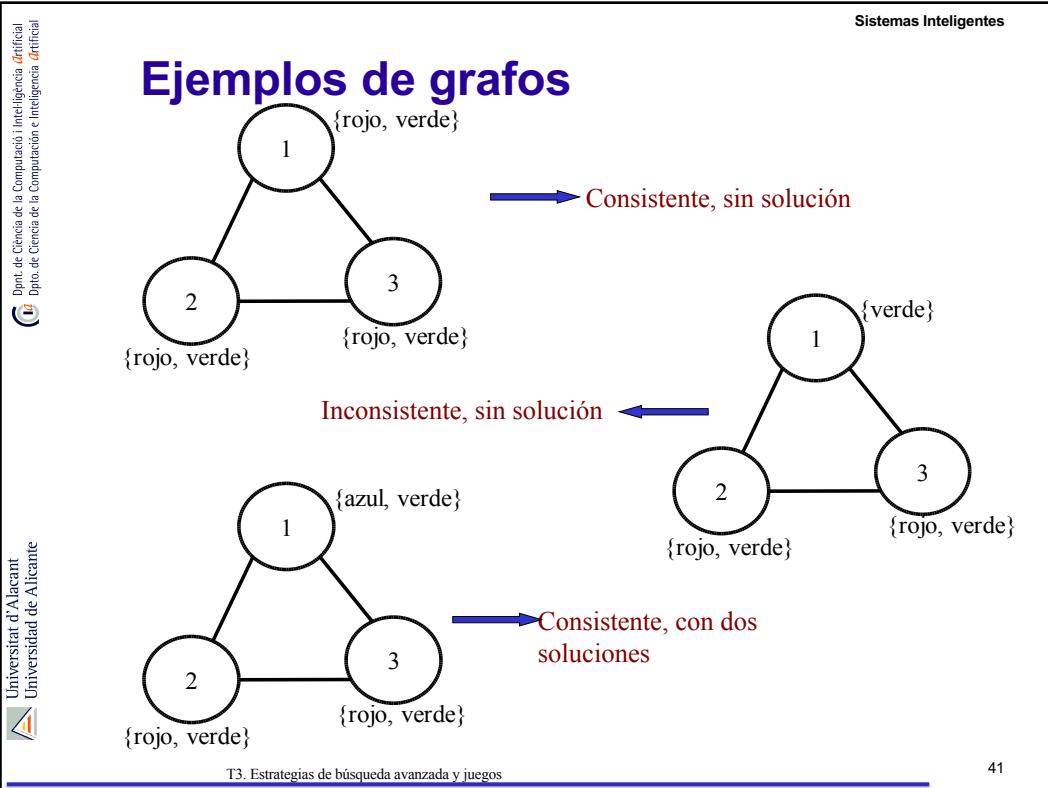
Algoritmo AC3

```

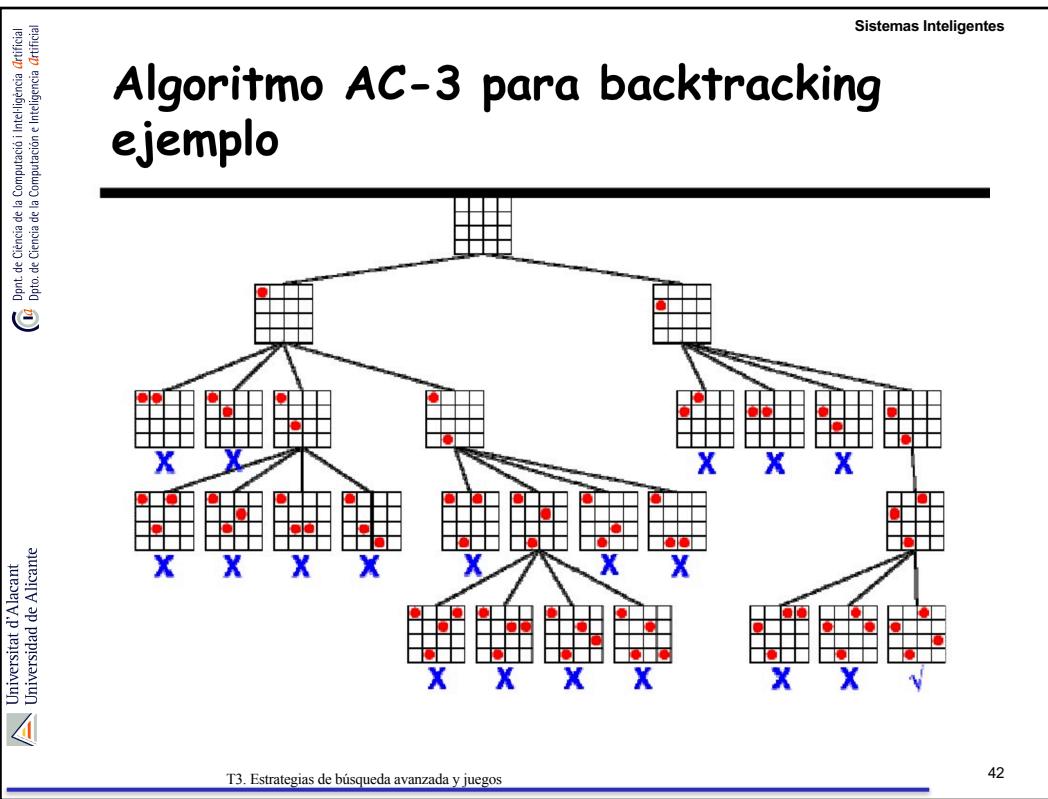
 $Q = \{c(e_p) = \langle V_i, V_j \rangle | e_p \in E, i \neq j\}$ 
Mientras  $Q \neq \emptyset$  hacer
   $\langle V_k, V_m \rangle = \text{seleccionar\_y\_borrar}(Q)$ 
  cambio = falso
  Para todo  $v_k \in D_k$  hacer
    Si no_consistente ( $v_k, D_m$ ) entonces
      borrar ( $v_k, D_k$ )
      cambio = cierto
    FinSi
  FinPara
  Si  $D_k = \emptyset$  entonces salir_sin_solución FinSi
  Si cambio = cierto entonces
     $Q = Q \cup \{c(e_r) = \langle V_i, V_k \rangle | e_r \in E, i \neq k, i \neq m\}$ 
  FinSi
FinMientras

```

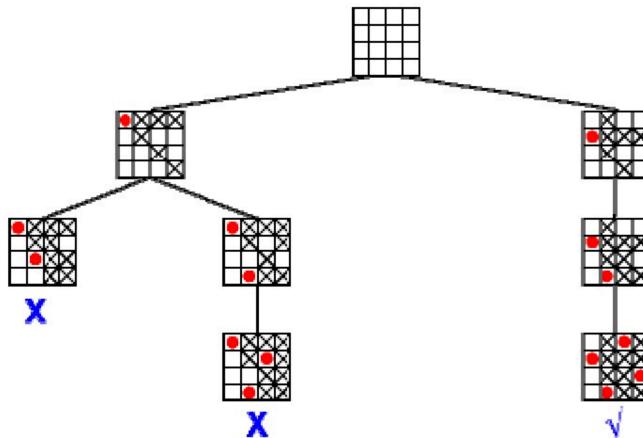
Ejemplos de grafos



Algoritmo AC-3 para backtracking ejemplo



Algoritmo AC-3 para Forward checking: ejemplo



Tema 3. Estrategias de búsqueda. Bibliografía

- Stuart Russell, Peter Noving. "Inteligencia Artificial. Un enfoque Moderno" Ed. Pearson. Prentice Hall. 2004.

Sistemas Inteligentes

Tema 4: Sistemas Expertos Difusos

Precision and Significance in the Real World

Precision

A 1500 kg mass is approaching your head at 45.3 m/s

Significance

LOOK OUT!!

1

Dpto. de Ciencia de la Computación e Inteligencia Artificial
Dpto. de Ciencia de la Computació i Intel·ligència Artificial

Universitat d'Alacant
Universidad de Alicante

Sistemas Expertos Difusos

1

Sistemas Inteligentes

Lógica Difusa. Introducción

LPO

- Especifiquemos qué es “Ser humano”
- Con Lógica de primer orden

• Es monotónica

• Dificultad de representar el conocimiento real

2

Dpto. de Ciencia de la Computación e Inteligencia Artificial
Dpto. de Ciencia de la Computació i Intel·ligència Artificial

Universitat d'Alacant
Universidad de Alicante

Sistemas Expertos Difusos

2

Sistemas Inteligentes

Lógica Difusa. Introducción

LPO

- Especifiquemos qué es “Ser humano”
- Con Lógica multivaluada

- Es monotónica
- Dificultad de representar el **conocimiento real**

3

Universitat d'Alacant
Universidad de Alicante

Dpt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Sistemes Expertos Difusos

Sistemas Inteligentes

Lógica Difusa. Introducción

Fuzzy Logic

- Especifiquemos con LPO “Ser humano”
- Lógica difusa

- Representación del **conocimiento de forma más natural**

4

Universitat d'Alacant
Universidad de Alicante

Dpt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Sistemes Expertos Difusos

Sistemas Inteligentes

Lógica Difusa. Conjuntos difusos (I)

(i) Dpt. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial

Universitat d'Alacant
Universidad de Alicante

$B = \{ (x, \mu_B(x)) / x \in X \}$
 $\mu_B: X \rightarrow [0, 1]$

La función de pertenencia se establece de una manera arbitraria (triangular, gaussiana...)

Sistemas Expertos Difusos

5

5

Sistemas Inteligentes

Lógica Difusa. Conjuntos difusos (II)

- ¿Cómo es la temperatura del Aula?
 - Cálida, Fría, Templada.
 - Definir los conjuntos difusos gráficamente y la función de pertenencia.
 - ¿23 grados es cálida o templada?

Universitat d'Alacant
Universidad de Alicante

Sistemas Expertos Difusos

6

6

Sistemas Inteligentes

Lógica Difusa. Conjuntos difusos (III)

¿23 grados es una temperatura
cálida o templada?

Universitat d'Alacant
 Universidad de Alicante

Sistemas Expertos Difusos

7

Sistemas Inteligentes

Lógica Difusa. Operaciones entre conjuntos (I)

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$$

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$$

$$\mu_{\neg A}(x) = 1 - \mu_A(x)$$

Universitat d'Alacant
 Universidad de Alicante

Sistemas Expertos Difusos

8

Sistemas Inteligentes

Lógica Difusa. Variables lingüísticas

- Una Variable Lingüística: palabras o sentencias que van a enmarcarse en un lenguaje predeterminado.
- Ejemplo:
 - Variable Edad:
 - Valores lingüísticos: Niño, Joven, Adulto, Viejo.
 - Universo del discurso de 0 a 100 años.
 - ¿Qué valor lingüístico tiene una persona de 25 años?

9

Sistemas Expertos Difusos

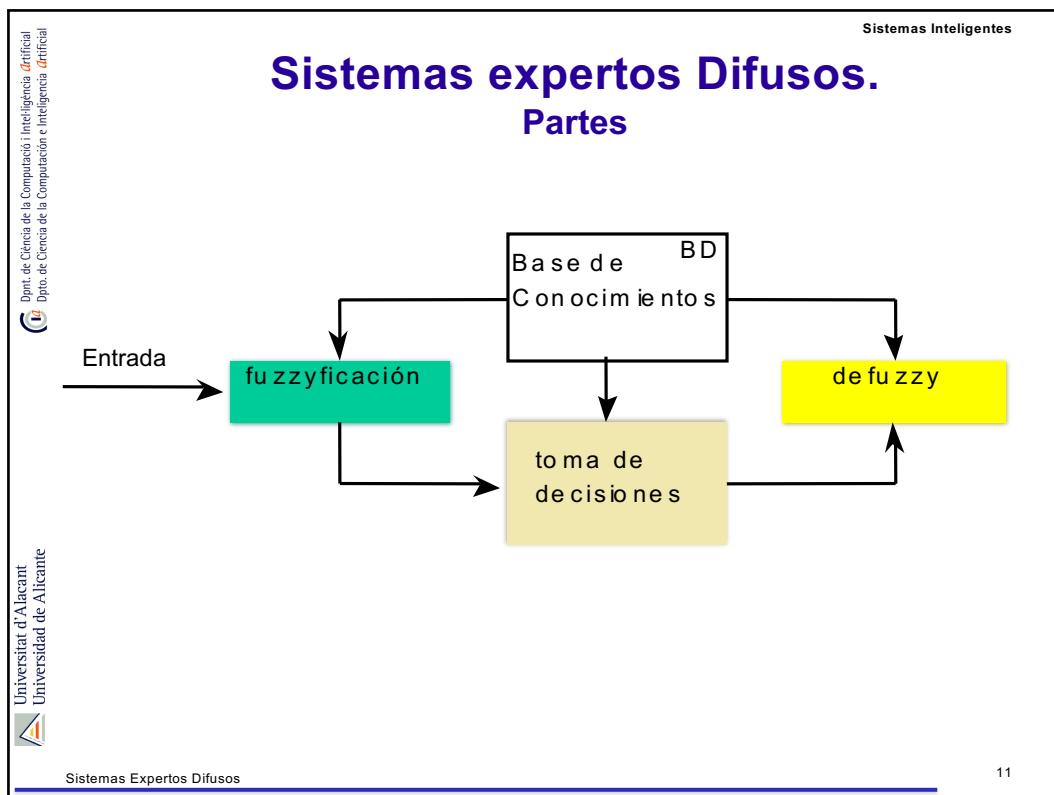
Sistemas Inteligentes

Lógica Difusa. Modificadores Lingüísticos

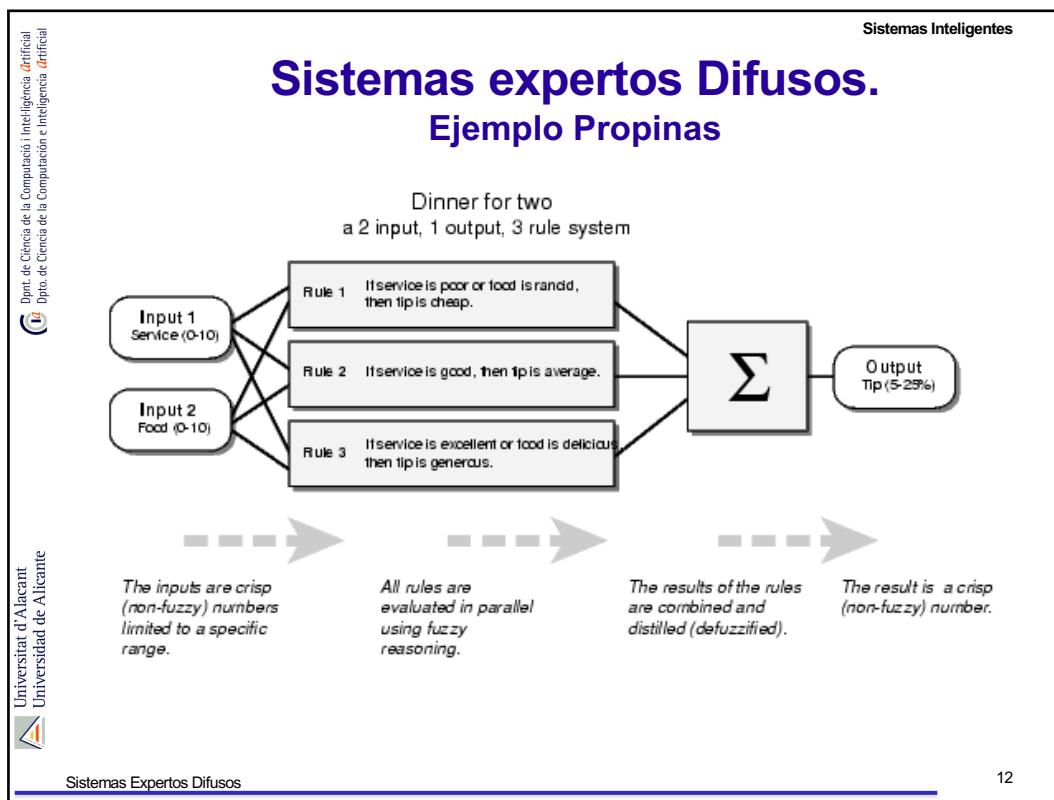
- Modificadores lingüísticos:
 - Operador que modifica el significado de un conjunto difuso:
 - Muy, Más o menos.
 - ¿Cómo lo podríamos describir Muy?
 - ¿Y más o menos?

10

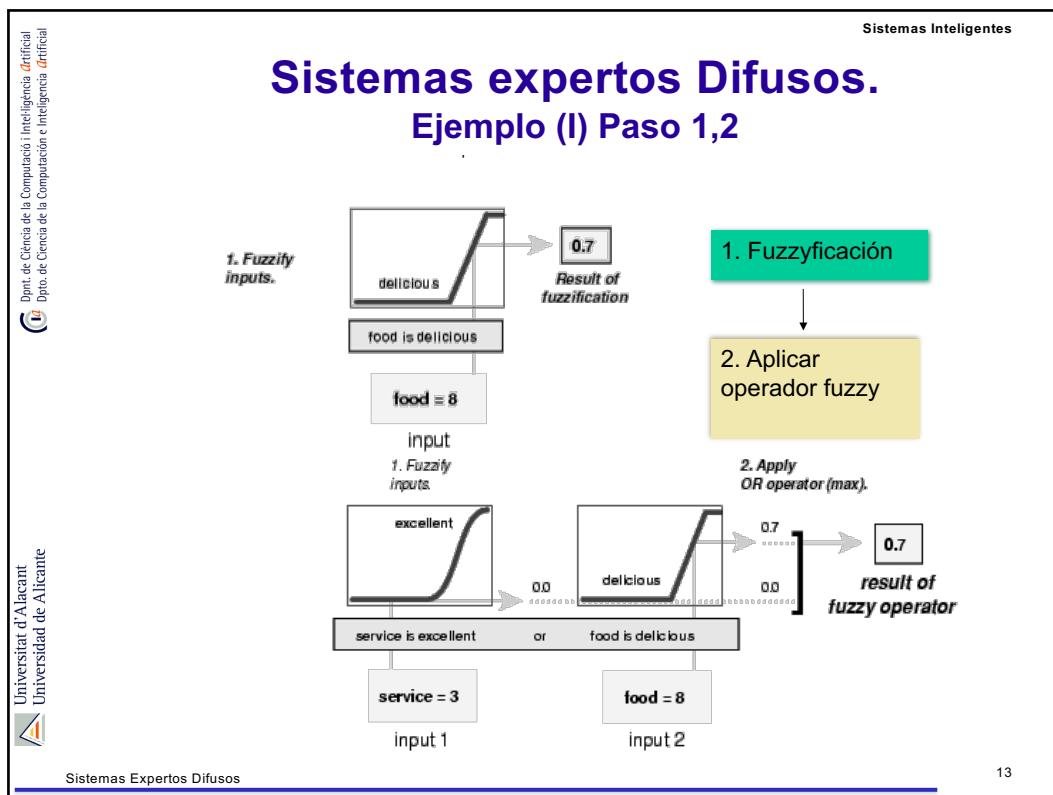
Sistemas Expertos Difusos



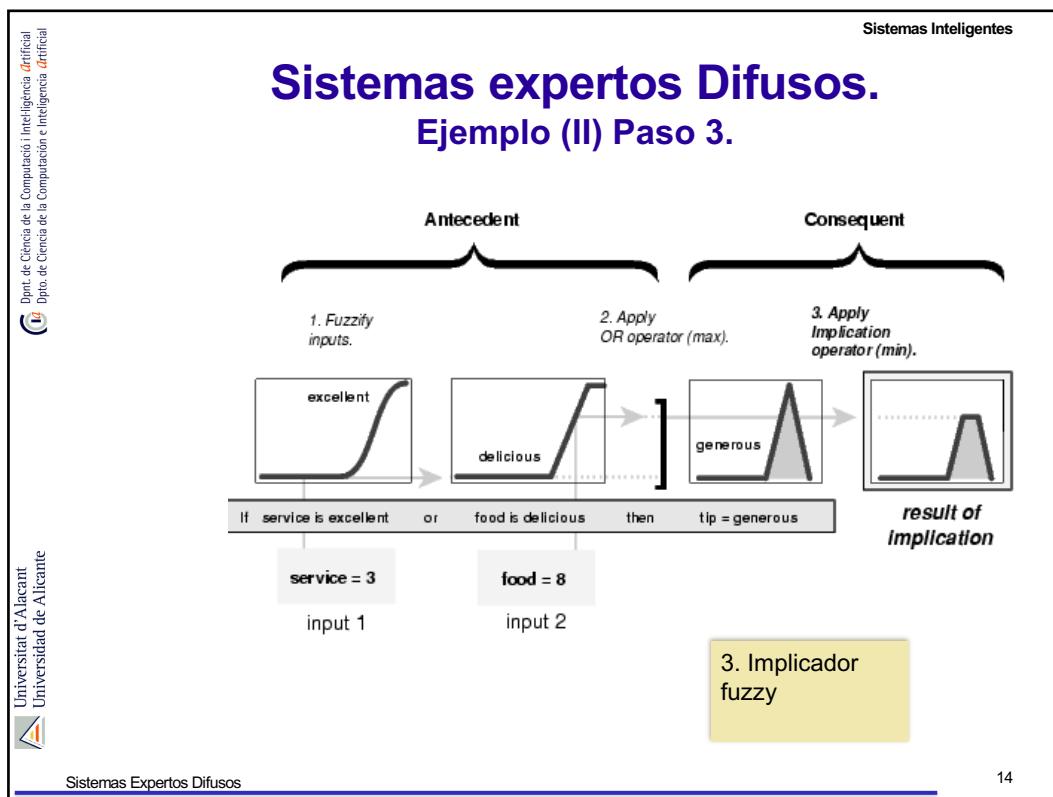
11



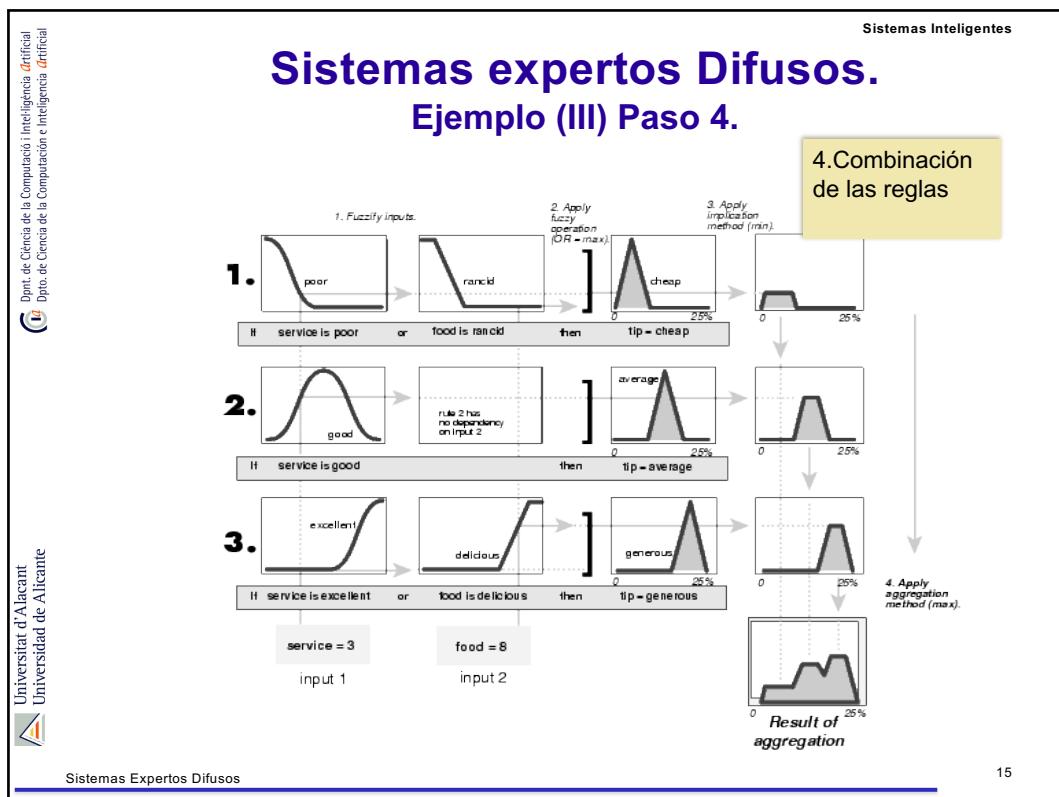
12



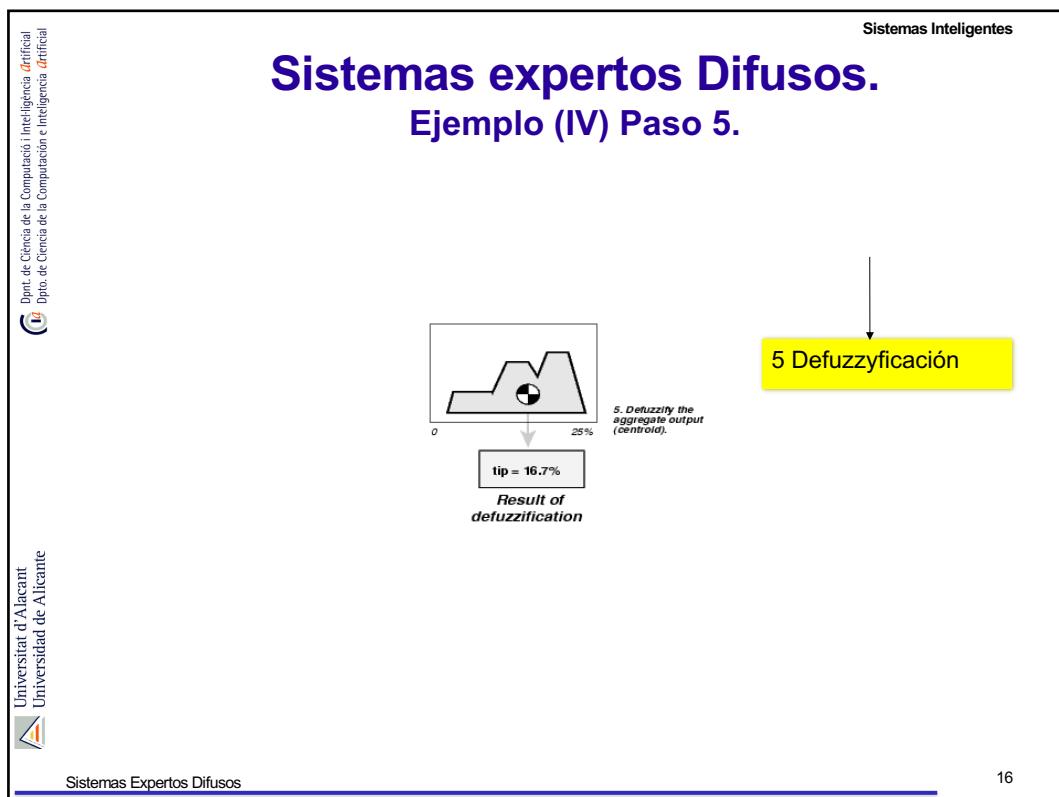
13



14



15



16

Sistemas Inteligentes

Sistemas expertos Difusos

Parámetros a establecer en el SE

- And/Or a utilizar

operator OR keyword for Algorithm	operator AND keyword for Algorithm
MAX	MIN
ASUM	PROD
BSUM	BDIF

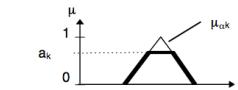
$\mu_1(x) + \mu_2(x) - \mu_1(x) \cdot \mu_2(x)$

$\min(1, \mu_1(x) + \mu_2(x))$

$\max(0, \mu_1(x) + \mu_2(x) - 1)$

- Método de agregación para los conjuntos de variables a defuzzificar

Name	Keyword	Formula
Maximum	MAX	$\max(\mu_1(x), \mu_2(x))$
Bounded Sum	BSUM	$\min(1, \mu_1(x) + \mu_2(x))$
Normalised Sum	NSUM	$\frac{\mu_1(x) + \mu_2(x)}{\max(1, \max(\mu_1(x') + \mu_2(x')))}$



COG

$$U = \frac{\int u \mu(u) du}{\int \mu(u) du}$$

COGS

$$U = \frac{\sum_{i=1}^p [u_i \cdot \mu_i]}{\sum_{i=1}^p [\mu_i]}$$

17

Universitat d'Alacant
Universidad de Alicante

Dpto. de Ciencia de la Computación e Inteligencia Artificial

Dpto. de Ciencia de la Computación e Inteligencia Artificial

Sistemas Expertos Difusos

17

Sistemas Inteligentes

FCL (I)

- Fuzzy Control Lenguaje
- INTERNATIONAL ELECTROTECHNICAL COMMISSION (IEC) TECHNICAL COMMITTEE No. 65: INDUSTRIAL PROCESS MEASUREMENT AND CONTROL SUB-COMMITTEE 65 B: DEVICES
 - IEC 1131 - PROGRAMMABLE CONTROLLERS
 - Part 7 - Fuzzy Control Programming
 - Committee Draft CD 1.0 (Rel. 19 Jan 97)
 - http://jfuzzylogic.sourceforge.net/doc/iec_1131_7_cd1.pdf

18

Universitat d'Alacant
Universidad de Alicante

Dpto. de Ciencia de la Computación e Inteligencia Artificial

Dpto. de Ciencia de la Computación e Inteligencia Artificial

Sistemas Expertos Difusos

18

Sistemas Inteligentes

FCL (II)

```

// Block definition (there may be more than one block per file,
FUNCTION_BLOCK tipper

// Define input variables
VAR_INPUT
    service : REAL;
    food : REAL;
END_VAR

// Define output variable
VAR_OUTPUT
    tip : REAL;
END_VAR

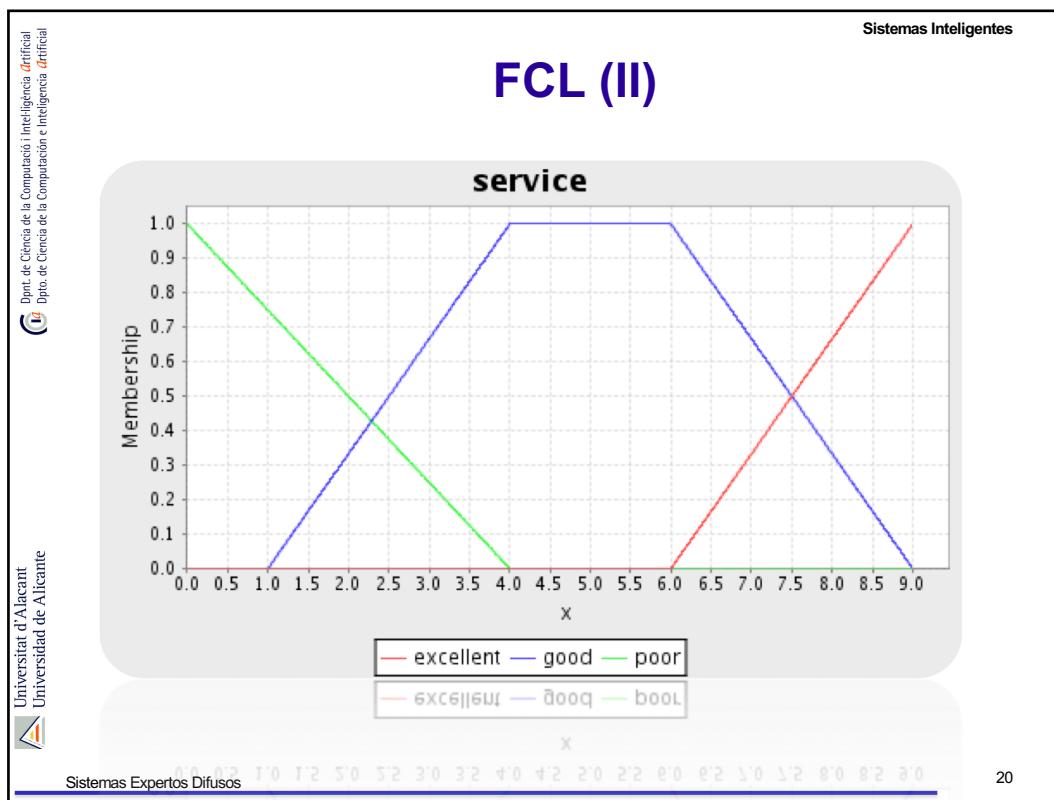
// Fuzzify input variable 'service'
FUZZIFY service
    TERM poor := (0, 1) (4, 0) ;
    TERM good := (1, 0) (4,1) (6,1) (9,0);
    TERM excellent := (6, 0) (9, 1);
END_FUZZIFY
AND_EXCEPT
    TERM excellent := (0, 0) (1,1);
    TERM good := (1, 0) (4,1) (6,1) (9,0);
    TERM poor := (0, 1) (4, 0) ;
END_EXCEPT

```

Sistemas Expertos Difusos

19

19



20

Sistemas Inteligentes

FCL (III)

```
DEFUZZIFY tip
  TERM cheap := (0,0) (5,1) (10,0);
  TERM average := (10,0) (15,1) (20,0);
  TERM generous := (20,0) (25,1) (30,0);
```

tip

Membership

X

— generous — average — cheap

— generos — avrg — chsb

Sistemas Expertos Difusos
0.0 5.2 10.0 15.2 20.0 25.2 30.0
21

21

Sistemas Inteligentes

FCL (IV)

```
Defzzzify output variable 'tip'
DEFUZZIFY tip
  TERM cheap := (0,0) (5,1) (10,0);
  TERM average := (10,0) (15,1) (20,0);
  TERM generous := (20,0) (25,1) (30,0);
// Use 'Center Of Gravity' defuzzification method
METHOD : COG;
// Default value is 0 (if no rule activates defuzzifier)
DEFAULT := 0;
END_DEFUZZIFY

RULEBLOCK N01
  // Use 'min' for 'and' (also implicit use 'max'
  // for 'or' to fulfill DeMorgan's Law)
  AND : MIN;
  // Use 'min' activation method
  ACT : MIN;
  // Use 'max' accumulation method
  ACCU : MAX;

  RULE 1 : IF service IS poor OR food IS rancid
    THEN tip IS cheap;

  RULE 2 : IF service IS good
    THEN tip IS average;

  RULE 3 : IF service IS excellent AND food IS delicious
    THEN tip is generous;
END_RULEBLOCK
```

Sistemas Expertos Difusos
ND_RULEBLOCK
22

22

Bibliografía

- AI Game Engine Programming (Game Development Series) Briam Schwab
- Sobre FCL:
http://jfuzzylogic.sourceforge.net/doc/iec_1131_7_cd1.pdf





TEMA 5: Arboles de Decisión y Redes Bayesianas

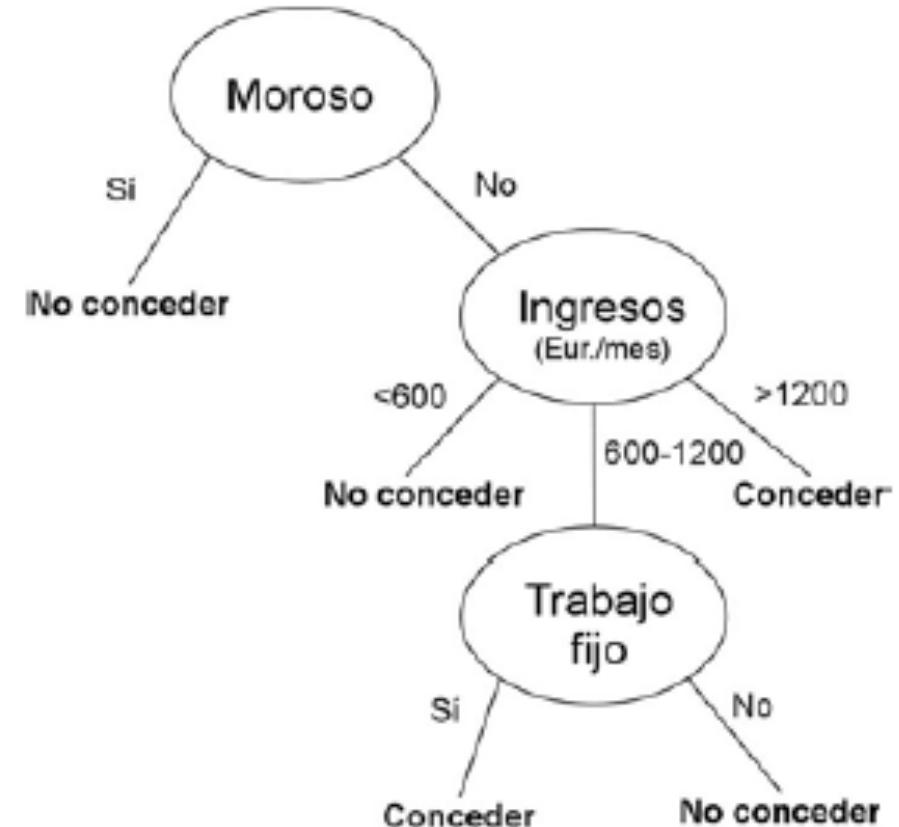


Parte 1: Árboles de decisión

- Árboles de decisión
 - Planteamiento del problema
 - Ejemplo: Concesión de créditos
 - Entropía y Ganancia de Información
- Algoritmo ID3
 - Algoritmo recursivo
 - Aplicación al ejemplo
 - Consideración de atributos numéricos
 - Atributos con un gran número de valores

Árboles de decisión

- Características:
 - Estructura para **clasificación** de vectores de atributos.
 - Establece **en qué orden** testar los atributos para conseguir la clasificación del vector de entrada.
 - Para componer dicho orden se eligen primero aquellos atributos que **mejor ganancia de información** prometen a efectos de descubrir la clase del vector de entrada.
 - Es interesante **aprenderlos** a partir de un conjunto de vectores





Ejemplo “Concesión de créditos”

Cliente	Moroso	Antigüedad (años)	Ingresos (Eur./mes)	Trab.fijo	Conceder
1	sí	>5	600-1200	sí	no
2	no	<1	600-1200	sí	sí
3	sí	1-5	>1200	si	no
4	no	>5	>1200	no	sí
5	no	<1	>1200	sí	sí
6	sí	1-5	600-1200	si	no
7	no	1-5	>1200	sí	sí
8	no	<1	<600	sí	no
9	no	>5	600-1200	no	no
10	Si	1-5	<600	no	no

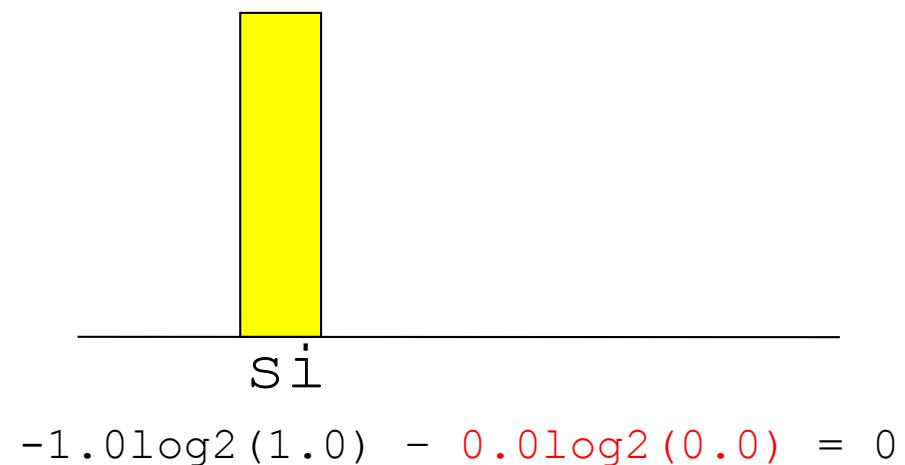
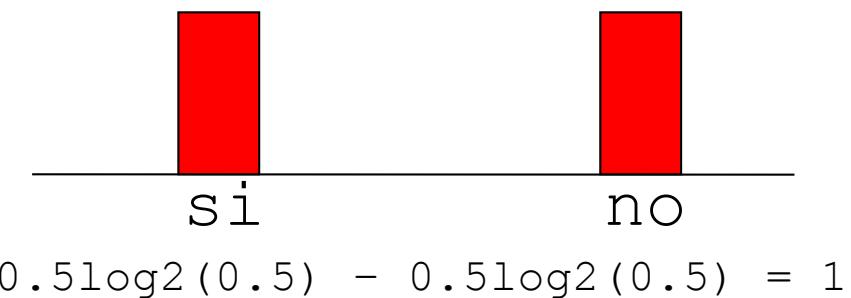
- Aprendizaje:
 - ¿Por qué atributo **comenzar** primero?
 - Esquema voraz: Elegir uno y **filtrar recursivamente**.

Entropía

Definición:

- Medida del **grado de incertidumbre** asociado a una distribución de probabilidad.
- En una **distribución uniforme**, todos los valores son igualmente probables $P_i = 1/N$ y por tanto la **entropía es máxima**, lo cual indica máxima incertidumbre.
- Por el contrario, en una **distribución pico** en la que $P_i = 1$ y $P_j=0$, para todo $j \neq i$ la entropía es mínima lo cual indica mínima incertidumbre o sea **máxima información**.

$$E(S) = \sum_{i \in C} -p_i \log_2 p_i$$



Entropía condicionada

Definición:

- Entropía de la distribución de Y condicionada a X.
- Una entropía condicionada menor que $E(Y)$ indica que el conocimiento de X mejora la información que se dispone sobre Y

$$E(Y | X) = \sum_j \text{Prob}(X = v_j) E(Y | X = v_j)$$

v_j	$\text{Prob}(X = v_j)$	$E(Y X = v_j)$
Math	0.5	1
History	0.25	0
CS	0.25	0

X	Y
Math	Yes
Hist.	No
CS	Yes
Math	No
Math	No
CS	Yes
Hist.	No
Math	Yes

$$E(Y) = 1$$

$$E(Y|X) = 0.5 * 1 + 0.25 * 0 + 0.25 * 0$$



$$E(Y|X) = 0.5$$

Ganancia de información

Definición:

- Medida de **cuanto ayuda el conocer** el valor de una variable aleatoria X para conocer el verdadero valor de otra Y.
- En nuestro caso, **X es un atributo** de un ejemplo dado mientras que **Y es la clase** a la que pertenece el ejemplo.
- Una alta ganancia implica que el atributo X permite **reducir la incertidumbre de la clasificación** del ejemplo de entrada.

$$IG(Y | X) = E(Y) - E(Y | X)$$

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

$$E(Y) = 1$$

$$E(Y|X) = 0.5$$

$$IG(Y | X) = 1 - 0.5 = 0.5$$



Algoritmo recursivo

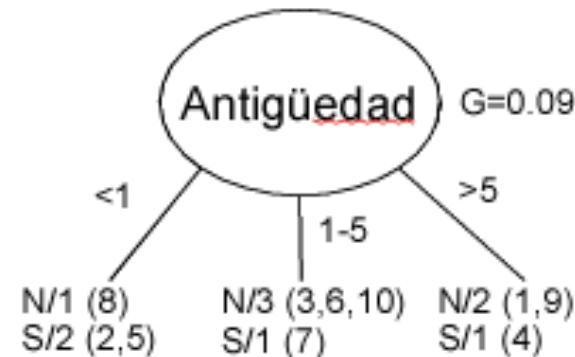
```

Algoritmo ID3(ejemplos, atributos) {
    Si atributos =  $\emptyset$  o MISMACLASE(ejemplos) {
         $C \leftarrow \text{CLASEMAYORITARIA}(\text{ejemplos})$ 
         $N \leftarrow \text{CREARNODOHOJA}(C)$ 
    }
    Sino {
         $a_{max} \leftarrow \max_{\forall A \in \text{atributos}} G(\text{ejemplos}, A)$ 
         $N \leftarrow \text{CREARNODO}(a_{max})$ 
        Para cada  $v_i \in \text{VALORES}(a_{max})$  {
             $\text{ejemplos}_{v_i} \leftarrow \{\text{elementos de ejemplos con valor } v_i \text{ para } a_{max}\}$ 
             $\text{AÑADIRHIJO}(N, \text{ID3}(\text{ejemplos}_{v_i}, \text{atributos} - a_{max}))$ 
        }
    }
    Devolver  $N$ 
}

```

Aplicación al ejemplo

- Entropía inicial:
 - Aplicando la ecuación de entropía a los datos de entrada del ejemplo tenemos:
$$E(S) = -0.4 \log_2(0.4) - 0.6 \log_2(0.6) = 0.971$$
- Para cada atributo (Antigüedad, Moroso, Ingresos, Fijo), calculamos la ganancia de información que obtenemos al seleccionar cada uno de ellos



$$\text{Prob}(S<1)=0.3, \text{Prob}(S1-5)=0.4, \text{Prob}(S>5)=0.3$$

$$E(S<1) = -2/3 \log_2(2/3) - 1/3 \log_2(1/3) = 0.9183$$

$$E(S1-5) = -1/4 \log_2(1/4) - 3/4 \log_2(3/4) = 0.811$$

$$E(S>5) = -1/3 \log_2(1/3) - 2/3 \log_2(2/3) = 0.9183$$

$$E(S<1) * 0.3 = 0.2755$$

$$E(S1-5) * 0.4 = 0.3244$$

$$E(S>5) * 0.3 = 0.2755$$

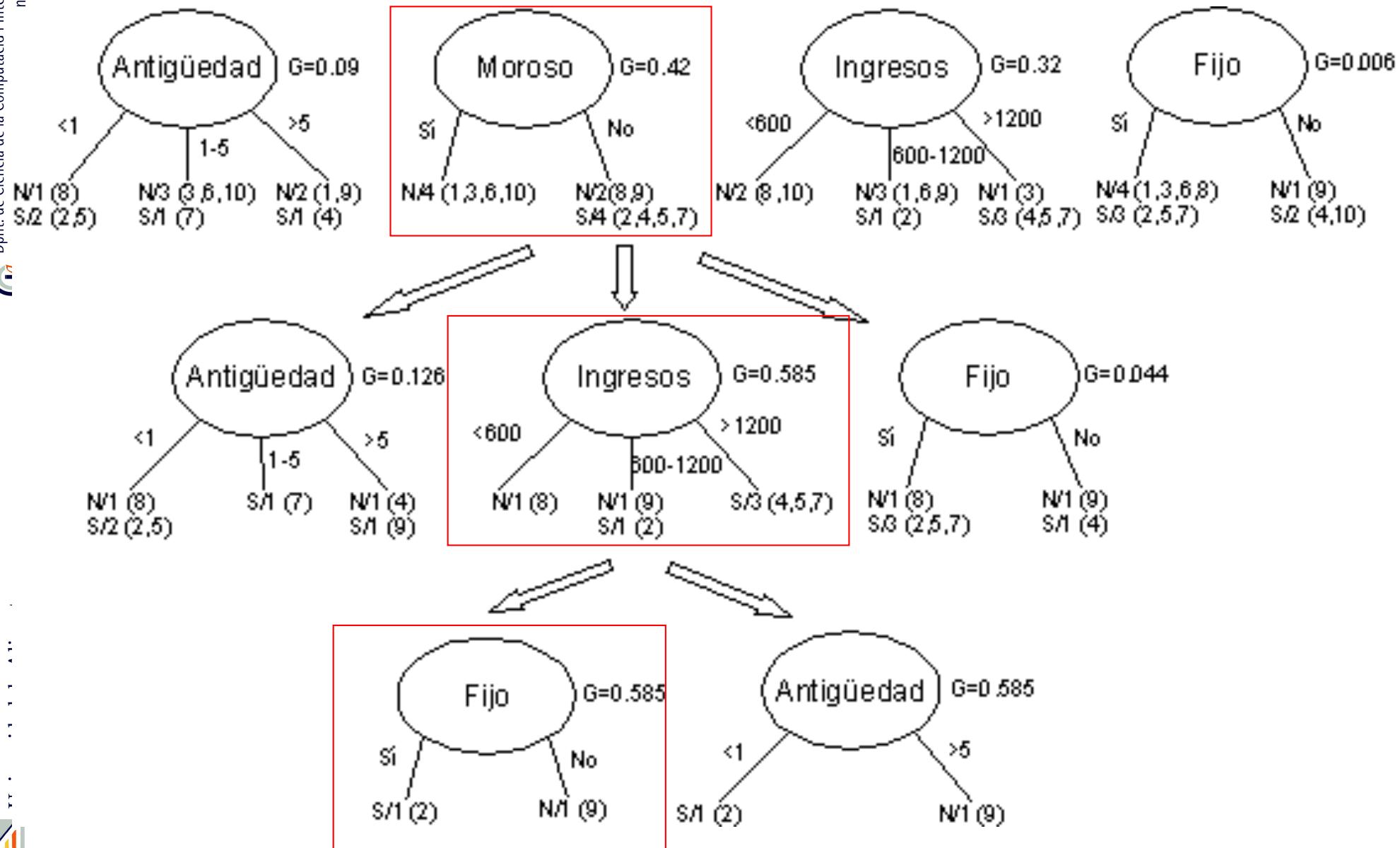
$$H(\text{Conceder} \mid \text{Antigüedad}) =$$

$$0.2755 + 0.3244 + 0.2755 = 0.8754$$

$$\text{Ganancia} = 0.971 - 0.8754 = 0.09$$



Aplicación al ejemplo





Extensiones del algoritmo

- Extensiones:

- Atributos numéricos: ID3 sólo trabaja con atributos discretos. Si se usan **atributos continuos** hay que **descomponerlos en rangos**. Para ello se ordenan los ejemplos según el valor y se toman como puntos límite los puntos medios de aquellos en que se cambie de clase.

Ejemplo	8	10	6	2	1	9	3	5	4	7
Ingresos	450	530	650	800	850	1050	1250	1400	1600	3000
Crédito	no	no	no	no	sí	no	sí	sí	sí	sí

- Atributos con gran número de valores. Se forman grupos pequeños de ejemplos que pueden ser **homogéneos por casualidad**. Debe introducirse un elemento corrector que penalice atributos con un elevado número de valores (**ganancia normalizada**):

$$G_N(S, A) = \frac{G(S, A)}{\sum_{v_i \in V(A)} - p_{v_i} \log_2 p_{v_i}}$$

- Sobre-entrenamiento. Comprobación de capacidad de generalización

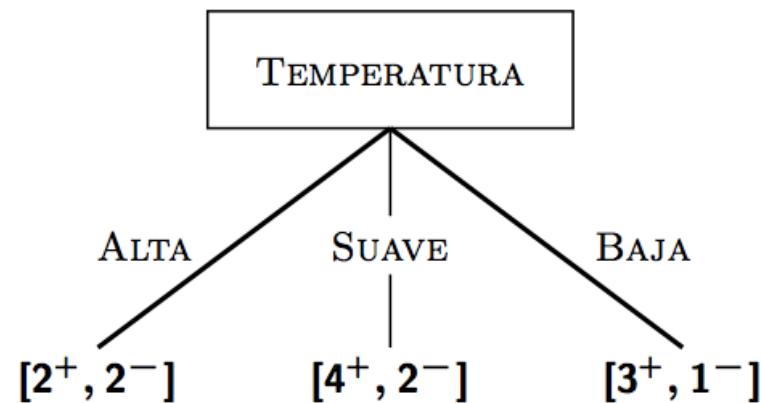
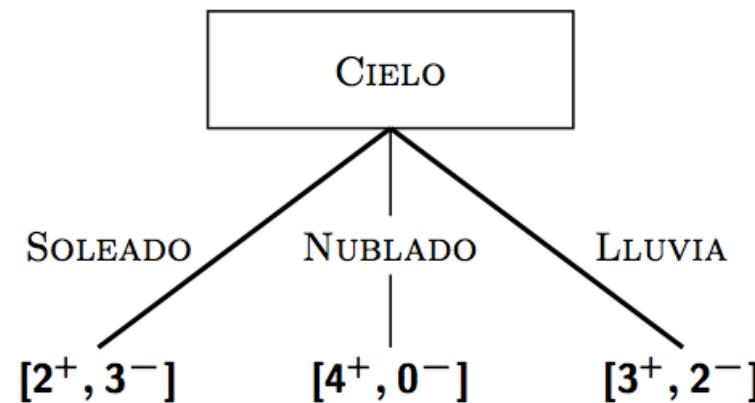
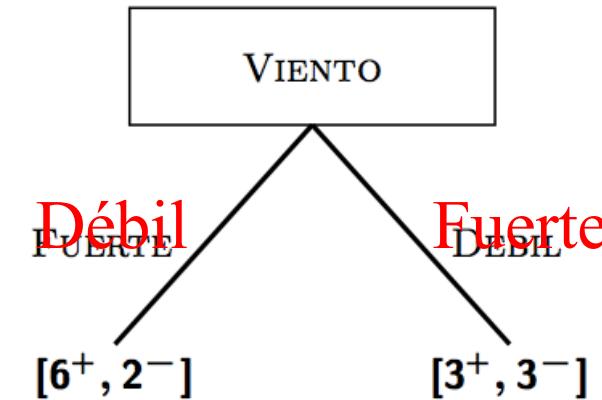
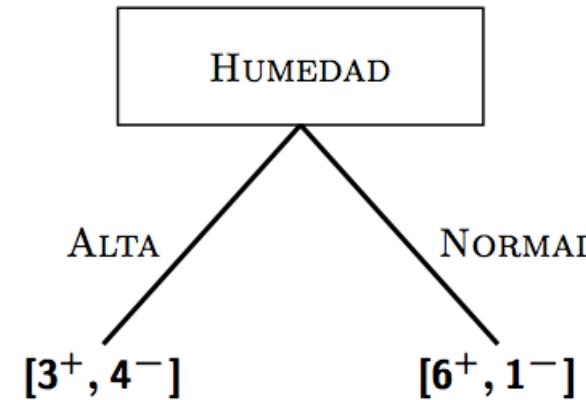
Ejercicios

Objetivo: Dado el conjunto de entrenamiento, aprender el concepto “Días en los que se juega al tenis” obteniendo el árbol de decisión mediante el algoritmo ID3

EJ.	CIELO	TEMPERATURA	HUMEDAD	VIENTO	JUGAR TENIS
D ₁	SOLEADO	ALTA	ALTA	DÉBIL	-
D ₂	SOLEADO	ALTA	ALTA	FUERTE	-
D ₃	NUBLADO	ALTA	ALTA	DÉBIL	+
D ₄	LLUVIA	SUAVE	ALTA	DÉBIL	+
D ₅	LLUVIA	BAJA	NORMAL	DÉBIL	+
D ₆	LLUVIA	BAJA	NORMAL	FUERTE	-
D ₇	NUBLADO	BAJA	NORMAL	FUERTE	+
D ₈	SOLEADO	SUAVE	ALTA	DÉBIL	-
D ₉	SOLEADO	BAJA	NORMAL	DÉBIL	+
D ₁₀	LLUVIA	SUAVE	NORMAL	DÉBIL	+
D ₁₁	SOLEADO	SUAVE	NORMAL	FUERTE	+
D ₁₂	NUBLADO	SUAVE	ALTA	FUERTE	+
D ₁₃	NUBLADO	ALTA	NORMAL	DÉBIL	+
D ₁₄	LLUVIA	SUAVE	ALTA	FUERTE	-



Ejercicios

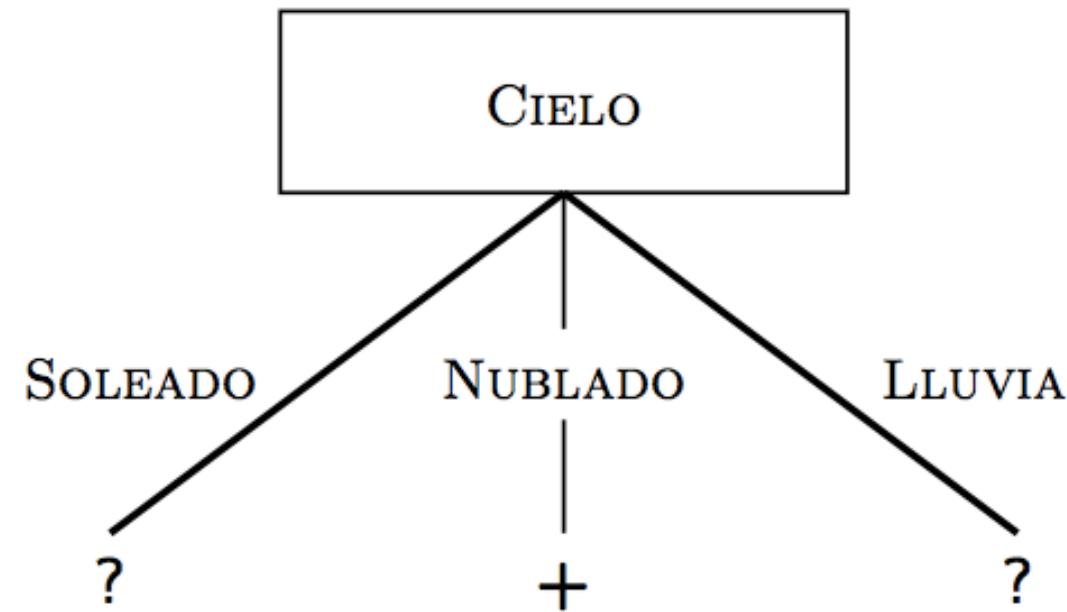


Ejercicios

- Entropía inicial: $\text{Ent}([9^+, 5^-]) = 0,94$
- Selección del atributo para el nodo raíz:
 - **Ganancia(D, HUMEDAD) =**
 $0,94 - \frac{7}{14} \cdot \text{Ent}([3^+, 4^-]) - \frac{7}{14} \cdot \text{Ent}([6^+, 1^-]) = 0,151$
 - **Ganancia(D, VIENTO) =**
 $0,94 - \frac{8}{14} \cdot \text{Ent}([6^+, 2^-]) - \frac{6}{14} \cdot \text{Ent}([3^+, 3^-]) = 0,048$
 - **Ganancia(D, CIELO) =**
 $0,94 - \frac{5}{14} \cdot \text{Ent}([2^+, 3^-]) - \frac{4}{14} \cdot \text{Ent}([4^+, 0^-])$
 $- \frac{5}{14} \cdot \text{Ent}([3^+, 2^-]) = 0,246$ (mejor atributo)
 - **Ganancia(D, TEMPERATURA) =**
 $0,94 - \frac{4}{14} \cdot \text{Ent}([2^+, 2^-]) - \frac{6}{14} \cdot \text{Ent}([4^+, 2^-])$
 $- \frac{4}{14} \cdot \text{Ent}([3^+, 1^-]) = 0,02$
- El atributo seleccionado es CIELO

Ejercicios

Árbol parcialmente construido:





Ejercicios

- Selección del atributo para el nodo CIELO=SOLEADO
- $D_{\text{SOLEADO}} = \{D_1, D_2, D_8, D_9, D_{11}\}$ con entropía
 $\text{Ent}([2^+, 3^-]) = 0,971$
 - **Ganancia**($D_{\text{SOLEADO}}, \text{HUMEDAD}$) =
 $0,971 - \frac{3}{5} \cdot 0 - \frac{2}{5} \cdot 0 = 0,971$ (mejor atributo)
 - **Ganancia**($D_{\text{SOLEADO}}, \text{TEMPERATURA}$) =
 $0,971 - \frac{2}{5} \cdot 0 - \frac{2}{5} \cdot 1 - \frac{1}{5} \cdot 0 = 0,570$
 - **Ganancia**($D_{\text{SOLEADO}}, \text{VIENTO}$) =
 $0,971 - \frac{2}{5} \cdot 1 - \frac{3}{5} \cdot 0,918 = 0,019$
- El atributo seleccionado es HUMEDAD

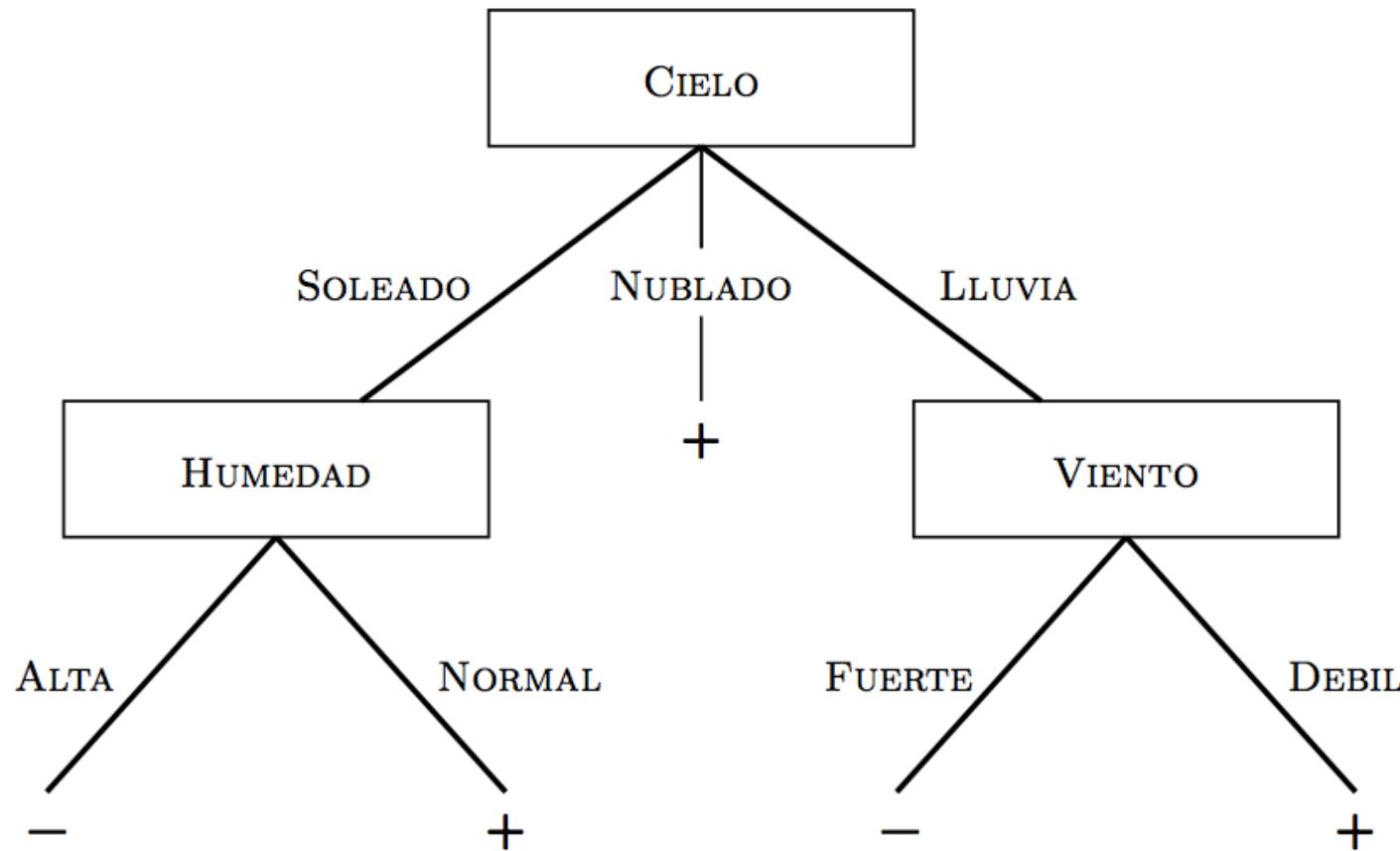
Ejercicios

- Selección del atributo para el nodo CIELO=LLUVIA:
- $D_{LLUVIA} = \{D_4, D_5, D_6, D_{10}, D_{14}\}$ con entropía
 $Ent([3^+, 2^-]) = 0,971$
 - **Ganancia(D_{LLUVIA} , HUMEDAD) =**
 $0,971 - \frac{2}{5} \cdot 1 - \frac{3}{5} \cdot 0,918 = 0,820$
 - **Ganancia(D_{LLUVIA} , TEMPERATURA) =**
 $0,971 - \frac{3}{5} \cdot 0,918 - \frac{2}{5} \cdot 1 = 0,820$
 - **Ganancia(D_{LLUVIA} , VIENTO) =**
 $0,971 - \frac{3}{5} \cdot 0 - \frac{2}{5} \cdot 0 = 0,971$ (mejor atributo)
- El atributo seleccionado es VIENTO



Ejercicios

- Árbol finalmente aprendido:





Parte 2: Redes Bayesinas

Probabilidad como medida de incertidumbre

Teorema de Bayes

Redes Bayesianas

Inferencia mediante redes Bayesianas

- Inferencia Exacta
- Ejemplos
- Inferencia aproximada
 - Muestreo directo
 - Muestreo por rechazo
 - Muestreo Gibbs

Para saber más



Probabilidad como media de incertidumbre (I)

Teoría de la probabilidad

Dos aproximaciones, frecuencial y bayesiana

Aproximación frecuencial

La probabilidad P de un evento a , $P(a)$ se define por la frecuencia de a basada en las observaciones pasadas

60% de los nacimientos en España son niñas

- $a = \text{'Elegir al azar a un bebe y que sea niña'}$
- $P(a) = 0.6$

Utilizamos el pasado para predecir el presente



Probabilidad como media de incertidumbre (II)

Aproximación Bayesiana

Razonar sobre creencias en condiciones de incertidumbre

Deseamos conocer la probabilidad de que una nueva arquitectura de ordenador funcione correctamente

No hay instancias previas

$a = \text{'gana el CD Alcoyano la liga del 2022'}$

$\hat{P}(a)?$

$P(a|Fidel) = 0,7$

$P(a|\text{Conocimiento previo})$: Medida de conocimiento, si este conocimiento previo permanece constante podemos escribir $P(a)$

¿Consistencia interna?



Probabilidad como media de incertidumbre (III)

Axiomas de probabilidad

Axiomas de la probabilidad

- I. $P(a)$ debe ser un nº entre $[0,1]$
- II. Si a es un evento cierto, entonces $P(a)=1$
- III. Si a y b son mutuamente exclusivos entonces
$$P(a \vee b) = P(a) + P(b)$$

De esta manera...

- $P(a + \neg a) = 1$ (por el 2º axioma)





Probabilidad como media de incertidumbre (IV)

Variables y distribuciones de probabilidad

A = 'Ganador de la liga en el 2022'

A = {a₁, a₂, a₃, ...}

$$\text{¿P}(a_1 + a_2) = 1?$$

No es exhaustivo

$$\text{P}(a_1 + a_2 + a_3 + \dots) = 1$$

Probabilidad total

$$\sum_{i=1}^n P(a_i) = 1$$



Probabilidad como media de incertidumbre (V)

Distribuciones conjuntas y marginalización

A: 'funciona el monitor' = {m1,m2,m3}

B: 'funciona la tarjeta de video' = {v1,v2}

?P(A,B)?

Distribución conjunta

$$P(A,B) = \{P(m1,v1), P(m1,v2), P(m2,v1), P(m2,v2), P(m3,v1), P(m3,v2)\}$$

Marginalización

$$P(a) = \sum_i P(a, b_i)$$





Probabilidad como media de incertidumbre (VI)

Probabilidad condicionada

El contexto K: $P(A) = P(A|K)$

$$P(A|B) = P(A|B,K)$$

Sucesos independientes

$$P(A|B) = P(A)$$

Sucesos Condicionalmente Independientes

$$P(A|B,C) = P(A|C) \quad A \text{ y } B \text{ son C.I. dado } C$$

Sucesos dependientes

$$P(A | B) = \frac{P(A, B)}{P(B)}$$

25





Probabilidad como media de incertidumbre (VII)

Probabilidad condicionada

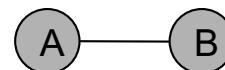
Pepe y Juan lanzan la misma moneda, primero lanza pepe

- a : 'Pepe obtiene cara'
- b : 'Juan obtiene cara'
- $P(A|B) = P(A)$



Igual que antes pero la moneda tiene cierta tendencia a sacar cara (no sabemos cual)

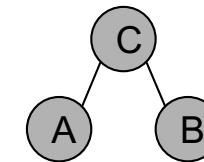
$$P(B|A) > P(A)$$



'A' y 'B' son dependientes con una variable C: 'la moneda tiene tendencia a sacar cara'.

Aunque 'A' y 'B' no son independientes si lo son respecto a 'C'

$$P(A|C) = P(A|B,C)$$





Teorema de Bayes

Sabemos que:

$$\begin{aligned} P(A|B)P(B) &= P(A,B) \\ P(B|A)P(A) &= P(B,A) = P(A,B) \end{aligned}$$

Regla de Bayes

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \alpha \cdot P(B|A)P(A)$$

Constante de normalización $P(B)$

$$P(B) = \sum_i P(B|A_i)P(A_i)$$

Regla de la cadena

$$P(A, B) = P(A)P(B|A)$$

$$P(A, B, C) = P(A)P(B|A)P(C|B,A)$$



Redes Bayesianas (I)

Una red bayesiana es:

- Un grafo acíclico dirigido para representar dependencias entre variables y mostrar una descripción escueta de cualquier distribución de probabilidad conjunta completa

Esta formada por

- Un conjunto de variables aleatorias que forman los nodos de la red. Cada nodo X tendrá adjunta una distribución $P(X|Padres(X))$
- Un conjunto de enlaces que determinan la influencia (dependencia) entre nodos. Si X se conecta con Y se dice que X influencia a Y

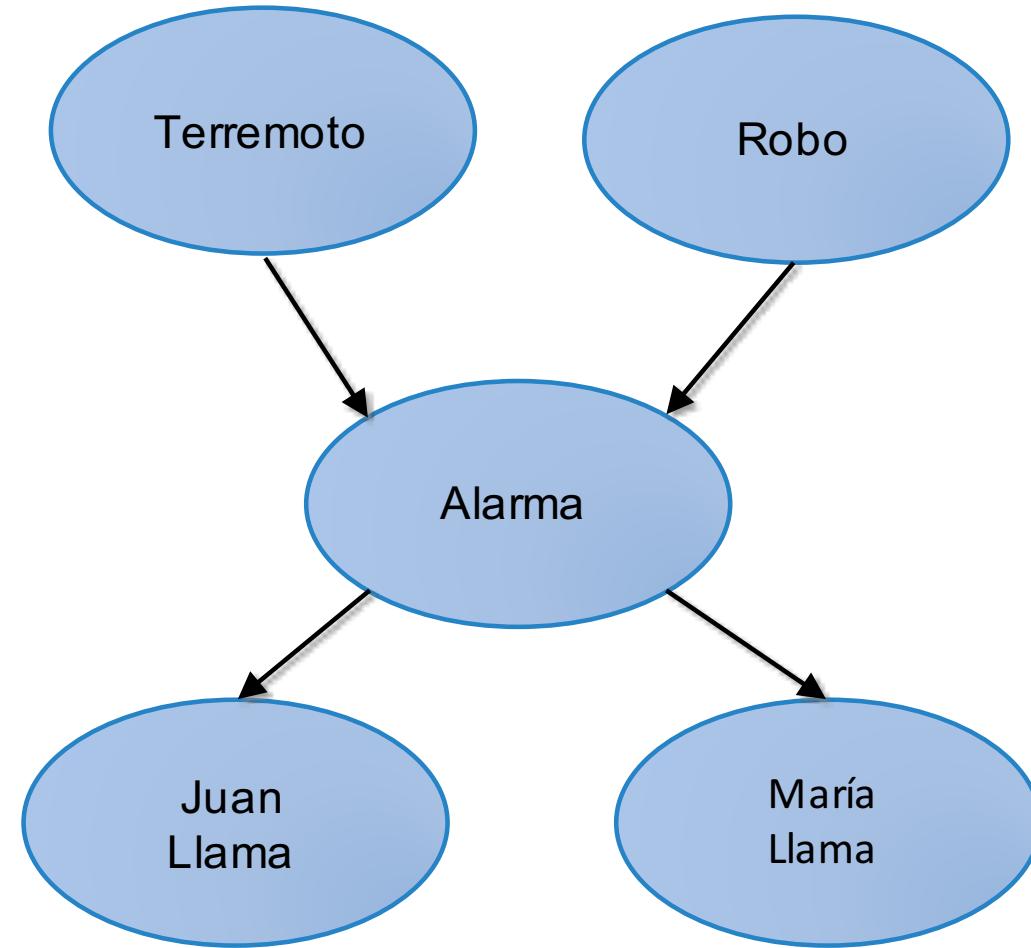
Su finalidad principal es calcular la distribución conjunta de las variables nodo



Redes Bayesianas (II) Semántica

Dada la siguiente red bayesiana,

¿Qué distribución
representa

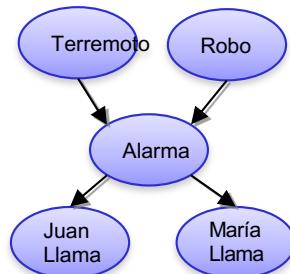




Redes Bayesianas (III)

Semántica

- $P(T, R, A, J, M) =$



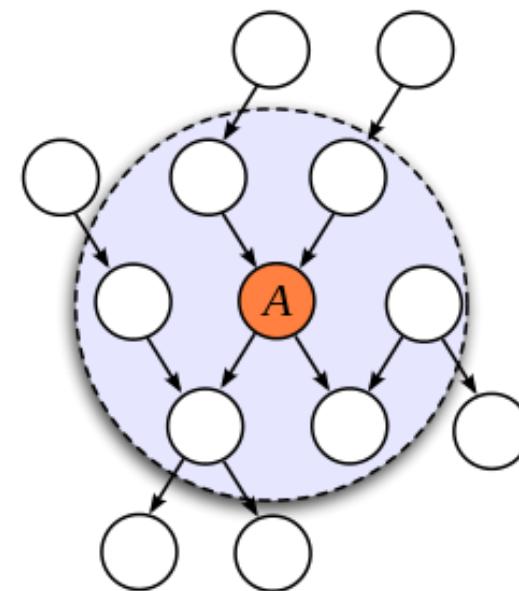
$$P(T) \cdot P(R) \cdot P(A|T,R) \cdot P(J|A) \cdot P(M|A)$$

- ¿ | $P(T, R, A, J, M)$ | sin independencia condicional?
 - $2^5 = 32$
- ¿Y con independencia condicional?
 - $2+2+2^3+2^2 +2^2 = 20$

Redes Bayesianas (IV)

Semántica

- Cobertura de Markov
 - Un nodo A es condicionalmente independiente de todos los nodos de la red dados:
 - Sus padres
 - Sus hijos
 - Los padres de sus hijos





Inferencia

¿Para que queremos la distribución conjunta?

A partir de la distribución conjunta podemos contestar cualquier pregunta relativa a la red...

Varios tipos de inferencia en redes Bayesianas

Exacta (caso general)

Casos especiales (Kim&Pearl...)

Aproximada



Inferencia exacta (I)

Inferencia exacta general (funciona para todas las RR.BB.)

Regla de inferencia general

(Donde B son las variables buscadas, C las conocidas y D las desconocidas)

$$P(B | C) = \alpha \cdot \sum_D P(B, D, C)$$

Problema: Mucha complejidad



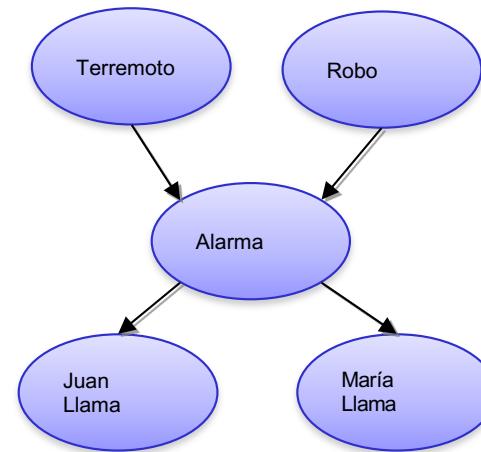
Inferencia exacta(II)

Ejemplo 1

- ¿Cuál es la probabilidad de que suene la alarma si llama María?

$$P(B | C) = \alpha \cdot \sum_D P(B, D, C)$$

- $P(R, T, A, J, M) =$
 $= P(R) \cdot P(T) \cdot P(A|R, T) \cdot P(J|A) \cdot P(M|A)$





Inferencia (III)

Ejemplo 1

De esta manera tenemos que:

$$\begin{aligned}
 P(A | M) &= \alpha \cdot \sum_R \sum_T \sum_J P(R, T, A, J, M) = \\
 &= \alpha \cdot \sum_R \sum_T \sum_J P(R) \cdot P(T) \cdot P(A | R, T) \cdot P(J | A) \cdot P(M | A) = \\
 &= \alpha \cdot P(M | A) \cdot \sum_R \left(P(R) \sum_T \left(P(T) \cdot P(A | R, T) \cdot \underbrace{\sum_J P(J | A)}_1 \right) \right)
 \end{aligned}$$

Inferencia (IV)

Ejemplo 2

¿ $P(R|J+,M+)$? Si sabemos que:

$$P(T) = 0,001$$

$$P(R) = 0,002$$

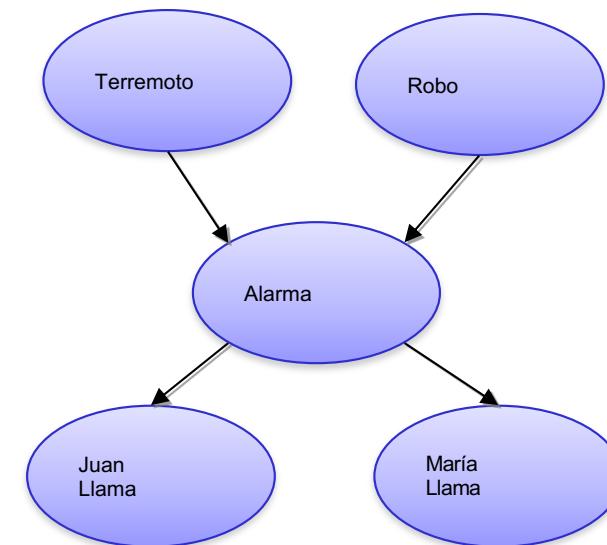
$$P(J|A) =$$

A	J
0	0,05
1	0,9

T	R	A
0	0	0,001
0	1	0,94
1	0	0,29
1	1	0,95

$$P(M|A) =$$

A	M
0	0,01
1	0,7





Inferencia (V)

Ejemplo 2

De esta manera tenemos que:

$$\begin{aligned} P(R \mid J, M) &= \alpha \sum_T \sum_A P(R, T, A, J, M) = \\ &= \alpha \sum_T \sum_A P(R) \cdot P(T) \cdot P(A \mid R, T) \cdot P(J \mid A) \cdot P(M \mid A) = \\ &= \alpha \cdot P(R) \cdot \sum_T \left(P(T) \cdot \sum_A (P(A \mid R, T) \cdot P(J \mid A) \cdot P(M \mid A)) \right) \end{aligned}$$



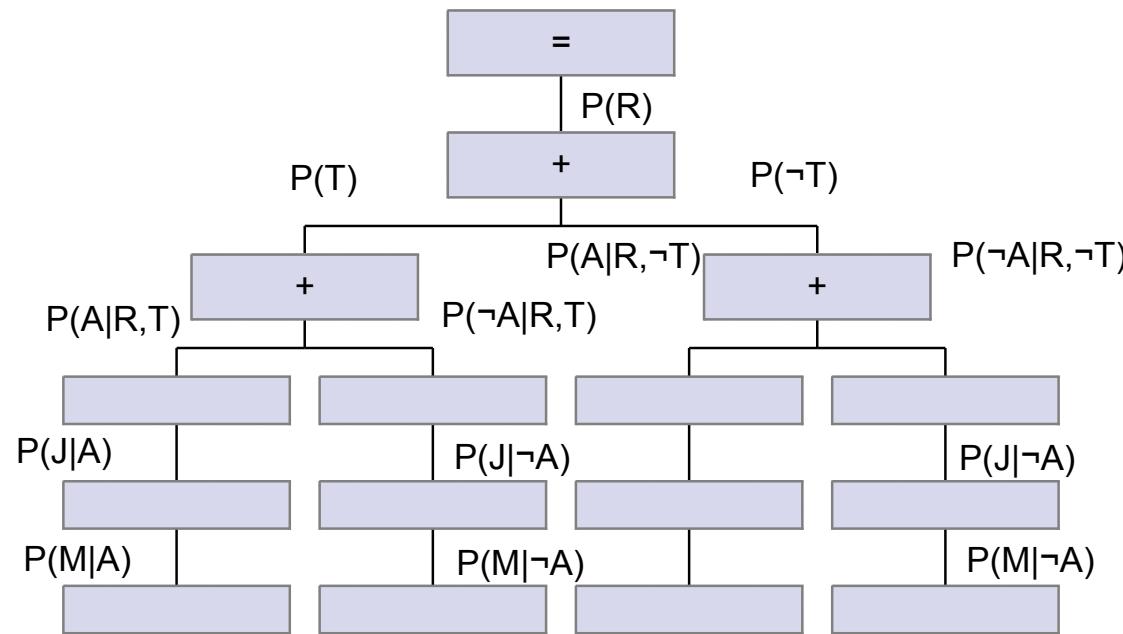


Inferencia (VI)

Ejemplo 2

Para calcular descomponemos utilizando un árbol:

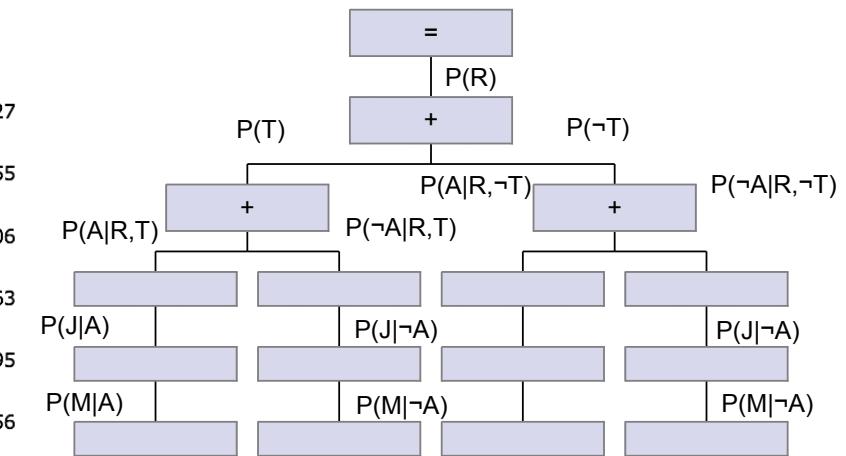
$$\alpha \cdot P(R) \cdot \sum_T P(T) \cdot \sum_A P(A | R, T) \cdot P(J | A) \cdot P(M | A)$$



Inferencia (VII)

Ejemplo 2

$P(R J,M)$	$P(\neg R J,M)$
$1 P(A R,T)*P(J A)*P(M A)$	0,5985
$P(\neg A R,T)*P(J,\neg A)*P(M \neg A)$	0,000025
$P(T)* \text{SUM 1}$	0,00059853
$2 P(A R,\neg T)*P(J A)*P(M A)$	0,5922
$P(\neg A R,\neg T)*P(J,\neg A)*P(M \neg A)$	0,00003
$P(\neg T)* \text{SUM 2}$	0,59163777
TOTAL R+	0,00118447
Cómo alpha*(R,\neg R) = 1	
TOTAL \neg R	0,00130899



$$\begin{aligned} P(R|J,M) = & \quad R & 0,47503178 \\ & \neg R & 0,52496822 \end{aligned}$$

Inferencia (VIII)

Ejercicio 1

¿P(J|R)?

$$\begin{aligned} P(J | R) &= \sum_T \sum_A \sum_M P(R) \cdot P(T) \cdot P(A | R, T) \cdot P(J | A) \cdot P(M | A) = \\ &= P(R) \cdot \sum_T (P(T) \cdot \sum_A (P(A | R, T) \cdot P(J | A) \cdot \sum_M P(M | A))) \end{aligned}$$

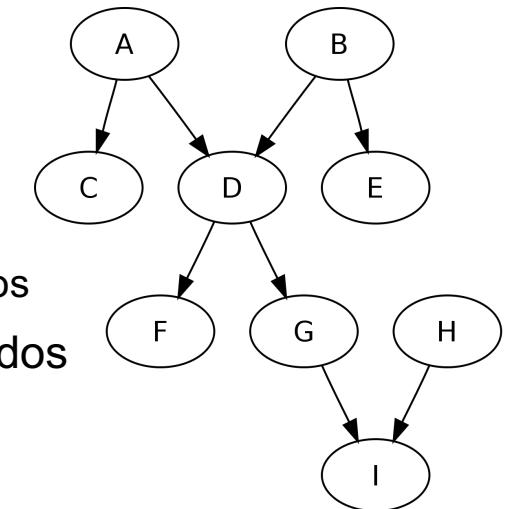


Inferencia exacta en poliárboles

Existen algoritmos más eficientes para tipos específicos de redes

Modelo de Kim y Pearl

- Método de inferencia para redes bayesianas.
- Solo aplicable a un poliárbol.
 - No existe más de un camino entre cada pareja de nodos
- Se basa en el paso de dos tipos de mensajes entre nodos
 - Para actualizar la credibilidad
 - Para introducir nueva evidencia
- Se puede calcular en tiempo lineal



Inferencia aproximada (I)

Sobre la inferencia exacta

- Redes con conexión múltiple son intratables utilizando inferencia exacta
- Complejidad NP-hard en el caso general

Inferencia utilizando algoritmos de muestreo aleatorio (Monte Carlo)

- Existen varios algoritmos
 - Muestreo directo
 - Muestreo por rechazo
 - Gibbs Sampling



Inferencia aproximada (II)

Muestreo directo

- rb: red Bayesiana
- ALGORITMO **Muestreo_Directo**(rb) retorna un evento extraido de rb
 - $X = \langle \text{vector de sucesos con } n \text{ elementos} \rangle$
 - Para cada variable X_i en X_1, \dots, X_n hacer
 - $X_i = \text{Obtener una muestra aleatoria de } P(X_i | \text{Padres}(X_i))$
 - Devolver X

Para responder cualquier pregunta de la red

- Obtener un vector de eventos $X[]$
- Contar apariciones en $X[]$ de las evidencias
- Dividir por suficientes Muestras



Inferencia aproximada (III)

Ejemplo de muestreo de una red mediante muestreo directo

$$P(T) = 0,001; P(R) = 0,002$$

$$P(A|T,R) =$$

T	R	A
0	0	0,001
0	1	0,29
1	0	0,94
1	1	0,95

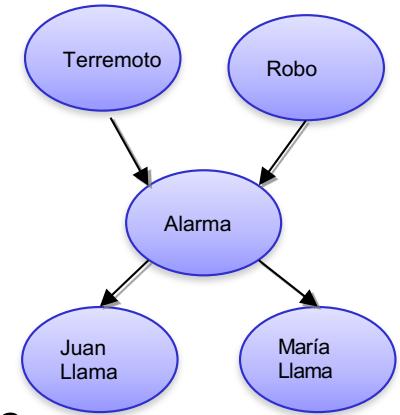
$$P(J|A) =$$

A	J
0	0,05
1	0,9

$$P(M|A) =$$

A	M
0	0,01
1	0,7

1. Muestreo a partir de $P(\text{Terremoto}) = <0,001 \ 0,999>$. **supongamos (s.)** que devuelve *falso*
2. Muestreo($P(\text{Robo})$) s. devuelve *falso*
3. Muestreo($P(\text{Alarma}|\langle\text{Robo}=\text{falso}, \text{Terremoto}=\text{falso}\rangle)$)
 - s. devuelve *cierto*
4. Muestreo($P(\text{Juan}|\langle A=\text{cierto}\rangle)$)
 - s. Devuelve *cierto*
5. Muestreo($P(\text{Maria}|\langle A=\text{cierto}\rangle)$)
 - s. Devuelve *falso*
6. $X = <\text{falso}, \text{falso}, \text{cierto}, \text{cierto}, \text{falso}>$





Inferencia aproximada (IV)

Ejemplo de uso

? $P(R|J,M)$?

$$P(T) = 0,001;$$

$$P(R) = 0,00$$

$$P(A|T,R)=$$

T	R	A
0	0	0,001
0	1	0,29
1	0	0,94
1	1	0,95

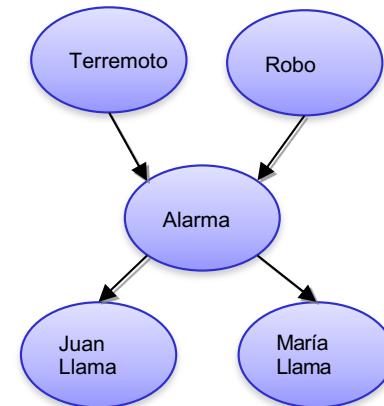
$$P(J|A) =$$

A	J
0	0,05
1	0,9

$$P(M|A) =$$

A	J
0	0,01
1	0,7

1. Para obtener $P(R|J,M)$
2. $C = \text{Contar } X[] \text{ que cumpla este patrón}$
 $X = <?, cierto, ?, cierto, cierto>$
3. Devolver
 $C/\text{numeroDeMuestras}$





Inferencia aproximada (V)

¿Problema del muestreo directo?

Otros tipos de muestreo aleatorio

- Muestreo por rechazo
- Gibbs Sampling



Inferencia aproximada (VI)

Muestreo por rechazo

- ALGORITMO **Muestreo_por_Rechazo(B, c, rb)** retorna estimación $P(B|c)$
- Para $j = 1$ hasta num_muestras hacer
 - $x = \text{Muestro_Directo}(rb)$
 - Si x es consistente con la evidencia c :
 - $N[y] = N[y] + 1$, donde y es el valor de B en x
 - Devolver Normalizar(N)

Entradas:

- B : variable buscada (pregunta)
- c : valores observados de las variables conocidas C
- rb : red bayesiana

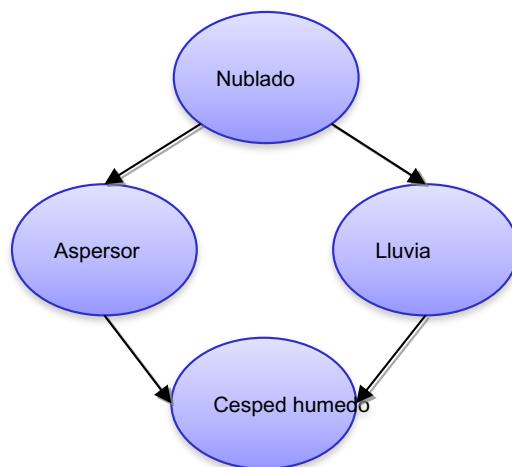
Variables locales:

- N : vector de recuento para cada valor de B , inicialmente 0

Inferencia aproximada (VII)

Muestreo por rechazo

Ejemplo:



Queremos estimar $P(\text{Lluvia}|\text{Aspersor}=\text{cierto})$

Extraemos 100 muestras, de las cuales 73 tienen el aspersor apagado

Nos quedamos con las 27 que coinciden con la evidencia

De las 27:

- En 8 Lluvia = cierto
- En 19 Lluvia es falso

$$P(\text{Lluvia}|\text{Aspersor}=\text{cierto}) = \text{Normalizar}(8, 19) = <0.296, 0.704>$$



Inferencia aproximada (VIII)

Muestreo por Gibbs (MCMC)

- ALGORITMO

Muestreo_por_Gibbs(B, c, rb, N) retorna estimación $P(B|c)$

- Inicializar x con valor aleatorios para las variables en Z
- Para $j = 1$ hasta num_muestras hacer
 - Para cada Z_i en Z hacer
 - $x[Z_i] = \text{muestrear } P(Z_i | mb(Z_i))$
 - $N[x] = N[x] + 1$ donde x es el valor de B en x
- Devolver Normalizar(N)

Entradas:

- B : variable buscada (pregunta)
- c : valores observados de las variables conocidas C
- rb : red bayesiana

Variables locales:

- N : vector de recuento para cada variable B (inicialmente vale 0)
- Z , las variables sin evidencia en rb
- x : el estado de la red, copiado inicialmente de c

Funciones

- $mb()$ retorna la cobertura de Markov un nodo

Inferencia aproximada (IX)

Ejemplo

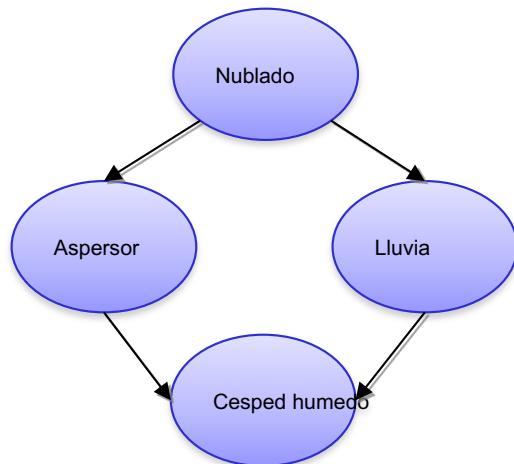
Queremos estimar $P(\text{Lluvia}|\text{Aspersor}=\text{cierto}, \text{Cesped}=\text{cierto})$

Las variables conocidas: Aspersor y Cesped, se fijan a su valor.

Las variables desconocidas: Nublado y Lluvia se establecen aleatoriamente.

Imaginemos que el estado inicial es: [cierto, cierto, falso, cierto]

Ahora las variables sin evidencia se muestran repetidamente en orden arbitrario. Por ejemplo



- Se muestrea Nublado, dado su recubrimiento de Markov. Por tanto extraemos de $P(\text{Nublado}|\text{Aspersor}=\text{cierto}, \text{Lluvia}=\text{Falso})$ Asumamos que el resultado es falso y por tanto el nuevo estado es [falso, cierto, falso, cierto]
- Se muestrea Lluvia dado su recubrimiento: $P(\text{Lluvia}|\text{Nublado}=\text{falso}, \text{aspersor}=\text{cierto}, \text{cesped}=\text{cierto})$. Asumamos que el resultado es cierto. El nuevo estado es [falso, cierto, cierto, cierto]

Todo estado visitado mediante este proceso es una muestra que contribuye a estimar la pregunta

Por ejemplo, si durante este proceso se visitan 20 estados donde lluvia es cierto y 60 donde es falso

$$P(L|A=\text{cierto}, C=\text{cierto}) = \text{Normalizar}(<20, 60>) = <0.25, 0.75>$$

Bibliografía

- Escolano et al. **Inteligencia Artificial**. Thomson-Paraninfo 2003. Capítulo 4.
- Mitchel, **Machine Learning**. McGraw Hill, Computer Science Series. 1997
- Cover, Thomas, **Information Theory**. Wiley & Sons, New York 1991