

# Proyecto **Elvira**

## Árbol explicación (*ExplanationTree*): Abducción usando medidas de información

M. Julia Flores , José A. Gámez  & Serafín Moral



S.I.M.D.

Almería, 19-20 Mayo 2005

1

## Esquema general

- Motivación
- Algoritmo
- Ejemplos
- Implementación
- Breve discusión
- Trabajos en desarrollo y futuros



S.I.M.D.

Reunión Elvira – Almería (Mayo 2005)

2

# Motivación

- En el marco de la *inferencia abductiva* o explicaciones para el **diagnóstico**:

- Abducción total (MPE)

$$x_U^* = \arg \max_{x_U \in \Omega_{X_U}} P(x_U | x_O)$$

- Abducción parcial (MAP)

$$x_E^* = \arg \max_{x_E} P(x_E | x_O) = \arg \max_{x_E} \sum_{x_R} P(x_E, x_R | x_O)$$

- Las técnicas de abducción *clásicas* normalmente siempre el mismo número de literales (simplificación de explicaciones).



## Motivación II

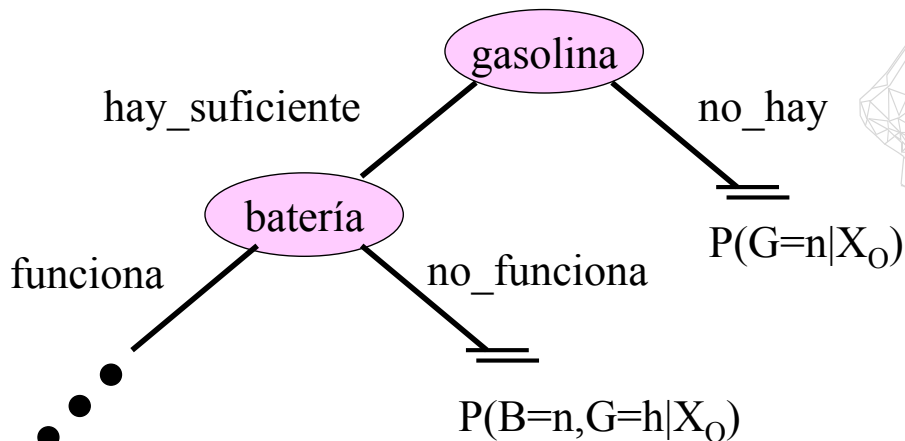
- Nuestro objetivo: conseguir explicaciones simplificadas de **manera directa**.
- Menos costoso y buscamos además una **partición** del conjunto de explicaciones.
- Forma: desarrollar un árbol (no simétrico) donde **ramas = explicaciones**.
- Para ello, nos basaremos en la *(in)estabilidad* del sistema usando la entropía de las variables o el índice de impureza GINI.



# Algoritmo

■ En el ejemplo de un coche que no arranca:

- $X_O$ : {fallo\_arranque = sí}
- $X_E = \{\text{batería, gasolina, sistema\_electrico...}\}$



## Algoritmo II: Planteamiento inicial

■ Para construir este árbol de explicaciones *minimales* necesitamos decidir:

- 1) ¿Cómo seleccionar la siguiente variable?
- 2) ¿Cuándo se abandona la expansión de una rama?

■ Construcción **desde la raíz a las hojas**:

- 1) Escogiendo aquella variable que maximice mi medida de información como siguiente.
- 2) Descartando la continuación de una rama si esta medida de información es demasiado baja ( $\alpha$ ) o la probabilidad de la rama despreciable ( $\beta$ ).

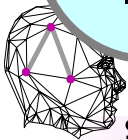


## Algoritmo III: el procedimiento

- Creamos árbol acumulando ramas. Inicio:  
*CreateNewNode* ( $x_o, \emptyset, X_E, \alpha, \beta, null$ )

**CreateNewNode** ( $x_o, path, X_E, \alpha, \beta, ET$ )

- 1 **for all**  $X_j, X_k \in X_E$  **do**  $Info[X_j, X_k] = Inf(X_j, X_k | x_o, path)$
- 2  $X_j^* = \arg \max_{X_j \in X_E} \sum_{X_k} Info[X_j, X_k]$
- 3 **If** *Continue*( $Info[], X_j^*, \alpha$ ) **and**  $P(path | x_o) > \beta$  **then**  
    **for all** estado  $x_j$  de  $X_j$  **do**  
        nuevo\_path  $\leftarrow$  path +  $X_j^* = x_j$   
        *CreateNewNode* ( $x_o, nuevo\_path, X_E \setminus X_j^*, \alpha, \beta, ET$ )  
**Else**  $ET \leftarrow$  *NUEVA\_RAMA*  $\langle path, P(path | x_o) \rangle$



## Algoritmo IV: comentarios

- *Inf* es la medida de información:
  - Información Mutua
  - Índice GINI.  $GINI(X, Y) = 1 - \sum_{x_i, y_j} P(x_i, y_j)^2$
- El vector  $Info[]$  guarda las  $Inf[X_j, X_k]$
- Para la poda en  $\alpha$ , consideramos  $Info[X_j^*, X_k]$  (máximo, mínima, media)
- Si en  $X_E$  sólo queda una variable, expandimos sólo si es más incierta que (1/3, 2/3).

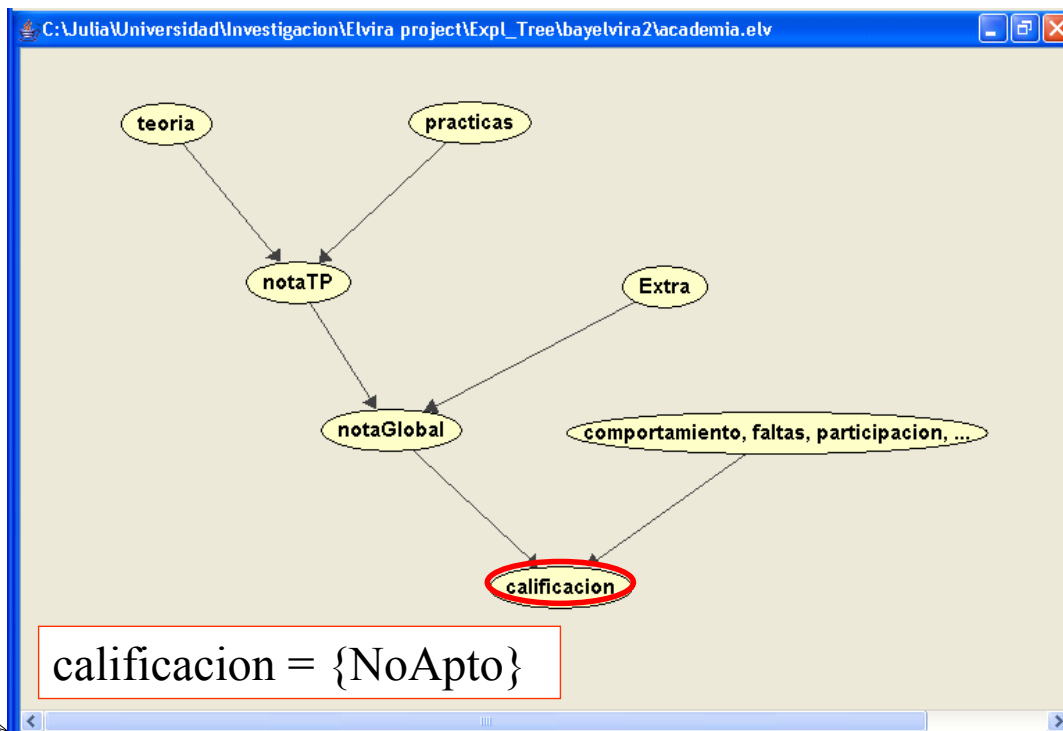


# Ejemplos

- Para un estudio preliminar de la técnica, hemos empleado dos redes sencillas de interpretar
  - *academia.elv*: modela la nota de un alumno en una asignatura en función de distintos factores.
  - *puertas.elv*: modela un circuito con siete puertas lógicas.

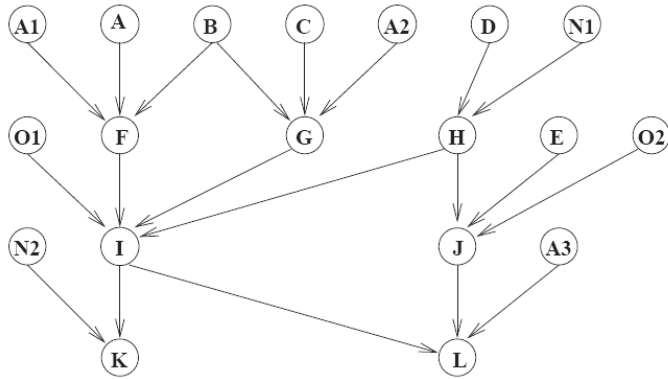
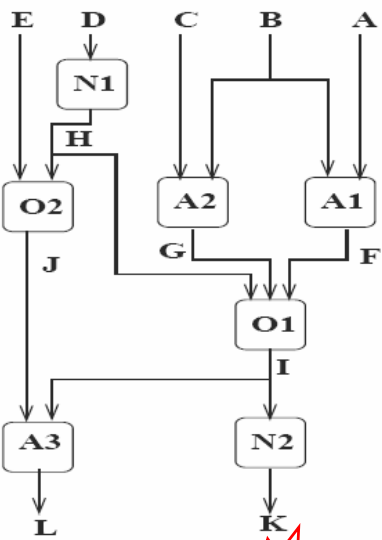


## Ejemplos II : *academia.elv*



# Ejemplos III: circuito

0 1 0 1 0



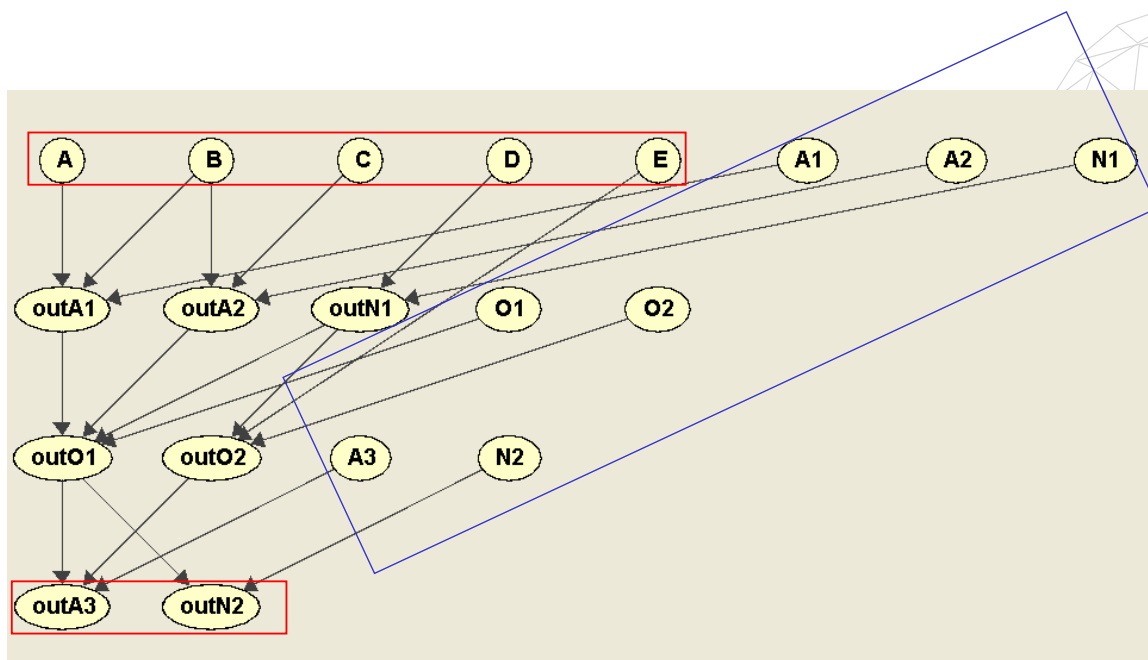
0

1



S.I.M.D.

# Ejemplos IV: *puertas.elv*



S.I.M.D.

# Ejemplos V: ETs para *academia*

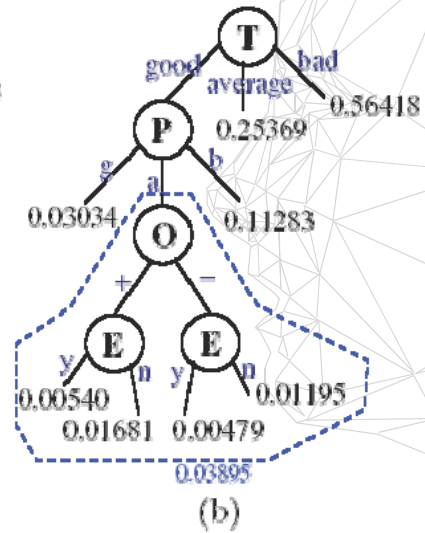
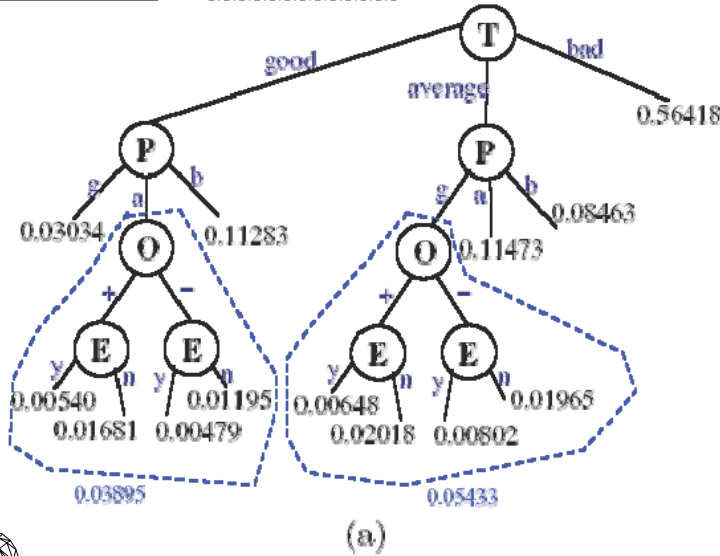
Todos los casos de IM, excepto (IM,  $\alpha=0.05, \min$ )

(gini,  $\alpha=0.05$ )

(gini,  $\alpha=0.07, \max$ )

$\beta=0.0$

$\beta=0.06$



S.I.M.D.

Reunión Elvira – Almería (Mayo 2005)

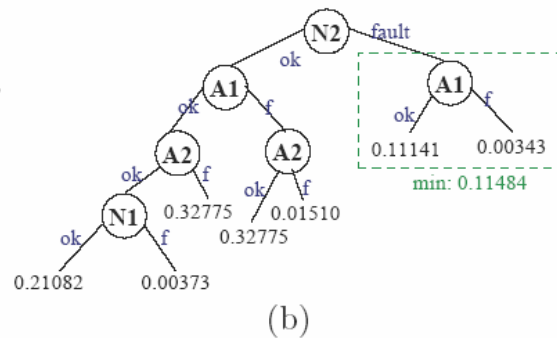
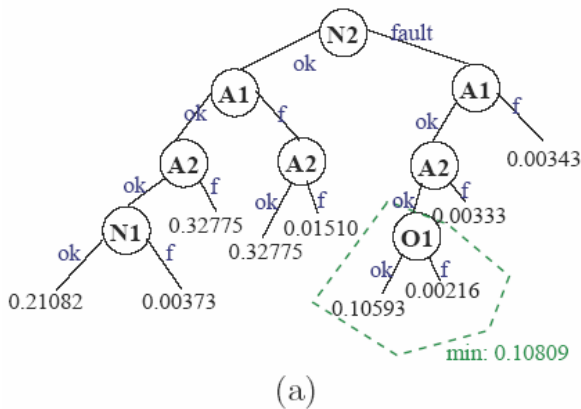
13

# Ejemplos VI: ETs para *puertas* (IM)

(IM,  $\alpha=0.05, \max|avg$ )

(IM,  $\alpha=0.07, \max$ )

(IM,  $\alpha=0.07, avg$ )



$\beta=0.05$



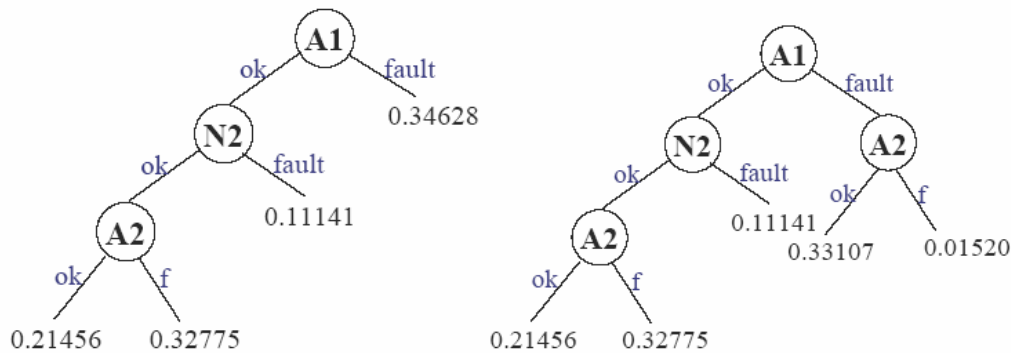
S.I.M.D.

Reunión Elvira – Almería (Mayo 2005)

14

## Ejemplos VII: ETs para *puertas* (gini)

Todos los casos excepto (gini,  $\alpha=0.05$ , max)



## Implementación

- ***ShenoyShaferPropagation.java*** ✓  
(package `elvira.inference.clustering`)
- ***ExplanationTree.java*** funcionando, pero todavía en fase de desarrollo (no subido).  
(en `elvira.inference.clustering.abduction`)
- SS *clásico*, pero con ciertas optimizaciones:
  - SingleCliques y MarginalCliques
  - Incorporar/Eliminar una evidencia sobre una variable es sencillo (`EvidenceItem`), se ahorran cálculos.
  - Se trabaja únicamente con árboles como potenciales.



## Implementación II: *ExplanationTree*:

- Sigue el esquema del algoritmo (CreateNewNode).
- El árbol toma el formato de un ProbabilityTree (sus ramas serían *configuraciones explicación*.)
- Para calcular  $Inf(X, Y)$  si están en distintos cliques usamos VE.
- Empleamos tablas Hash para agilizar el acceso a los cliques y a las variables en  $X_E$ .
- Construcción del árbol en profundidad (mediante SS):
  - Introduce las observaciones o evidencia inicial ( $x_0$ )
  - Selecciona variable  $V$  que maximiza Info  $\rightarrow V_i$
  - Introduce como evidencia (simple)  $V_i=v_{i1} \dots$
  - ... hasta que hay poda o bien no quedan variables en  $X_E$  (retractEvidenceItem)



## Breve discusión

- Las explicaciones obtenidas son *razonables*.
- Interesa estudiar más a fondo y “pulir” la poda.
- Se consigue proporcionar explicaciones con diferente nivel de complejidad.
- Además, obtenemos una partición de los distintos escenarios para las variables explicación.
- Se trata de un paso inicial. Creemos que es necesario:
  - diseñar experimentos más elaborados.
  - refinar y extender el método.



## Trabajos en desarrollo y futuros

- Se están **unificando los parámetros**  $\alpha$  (entropía) y  $\beta$  (probabilidad), en uno único que combina ambos factores:  
$$\gamma = P(X=x|e,path) \cdot H(X=x).$$
- Se está considerando el uso de la **distancia KL** entre dos distribuciones de probabilidad (para Info y para un desarrollo diferente del árbol: k-explicaciones).
- **Extensión y aplicación** del método a técnicas de *clustering*.

