UNIVERSIDAD DE GRANADA

E.T.S. DE INGENIERÍA INFORMÁTICA

Departamento de Ciencias de la Computación e Inteligencia Artificial

PROPAGACIÓN APROXIMADA DE INTERVALOS DE PROBABILIDAD EN GRAFOS DE DEPENDENCIAS

TESIS DOCTORAL

Andrés Cano Utrera

Granada, junio de 1.999

PROPAGACIÓN APROXIMADA DE INTERVALOS DE PROBABILIDAD EN GRAFOS DE DEPENDENCIAS

MEMORIA QUE PRESENTA

ANDRÉS CANO UTRERA

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

JUNIO 1.999

DIRECTOR

SERAFÍN MORAL CALLEJÓN

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
E INTELIGENCIA ARTIFICIAL

E.T.S. DE INGENIERÍA INFORMÁTICA UNIVERSIDAD DE GRANADA

La memoria titulada **Propagación aproximada de intervalos de probabilidad en grafos de dependencias**, que presenta D. Andrés Cano Utrera para optar al grado de DOCTOR, ha sido realizada en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección del Doctor D. Serafín Moral Callejón.

Granada, junio de 1.999

El doctorando

Andrés Cano Utrera

El director

Serafín Moral Callejón

AGRADECIMIENTOS

En primer lugar he de mostrar mi más profundo agradecimiento al doctor D. Serafín Moral Callejón director de esta memoria por su apoyo constante. Sin su ayuda, esfuerzo y dedicación nunca habría sido capaz de realizar este trabajo.

También he de mostrar mi agradecimiento a mis compañeros Silvia Acid, Juan Huete, Luis Miguel de Campos, José Enrique Cano y Juan Francisco Verdegay por su apoyo y el grato ambiente de trabajo que siempre han mantenido conmigo. Quiero también agradecer al resto de miembros del grupo de investigación de "Tratamiento de la Incertidumbre en Inteligencia Artificial" el buen compañerismo que siempre han mostrado conmigo, y su oferta de ayuda en cualquier momento.

También a mis compañeros de la sección de la Facultad de Ciencias de nuestro departamento quiero darles las gracias por su gran compañerismo. Especial mención merece el profesor Javier Mateos Delgado que me ayudó bastante a la hora de configurar el estilo de imprenta de esta memoria.

Quiero hacer también extensiva mi gratitud al resto de miembros del departamento de Ciencias de la Computación, sobre todo por su interés especial en que esta memoria saliese adelante.

Por otra parte agradezco a la Comunidad Económica Europea y concretamente al proyecto ESPRIT III DRUMS II BRA 6156 por el soporte económico, mediante una beca de investigación, para los dos primeros años de trabajo en esta memoria. También agradezco a la CICYT por su financión a través del proyecto "Entorno para el desarrollo de modelos gráficos probabilísticos (TIC97-1135-C04-01)" que han permitido sufragar los gastos de continuación y finalización de esta memoria.

Por último, y no por ello de menor importancia, no puedo dejar de dar las gracias a mi familia y en especial a mis padres y esposa por su paciencia, comprensión, apoyo moral e interés mostrados durante todo el tiempo que ha durado la realización de este trabajo.

A mis padres

y

a Pilar

PROPAGACIÓN APROXIMADA DE INTERVALOS DE PROBABILIDAD EN GRAFOS DE DEPENDENCIAS

ANDRÉS CANO UTRERA

Índice General

Índice de Figuras

Índice de Tablas

Introducción

Motivación

Las redes bayesianas son una herramienta que en los últimos años ha demostrado su potencialidad como modelo de representación de la incertidumbre en Inteligencia Artificial. El éxito de numerosas aplicaciones en campos tan variados como la medicina, recuperación de la información, visión artificial, fusión de información, agricultura, etc. avalan esta afirmación [74, 102, 2, 8, 113, 66, 73, 53, 7, 67, 69].

Una de las dificultades más importantes asociadas a la construcción de sistemas expertos probabilísticos está en la elicitación de los valores de las probabilidades. En general, se necesita especificar un gran número de valores de probabilidad. Como se afirma en Skaaning et al. [122]:

> En la construcción de modelos de redes bayesianas, la adquisición de la información probabilística a partir de expertos en el dominio es a menudo la parte más difícil del proceso de construcción.

Hay que tener en cuenta que el experto tiene que especificar para un gran conjunto de sucesos valores de probabilidad que son números reales, lo que implica una precisión que sobrepasa en muchas ocasiones su grado de conocimiento del problema. A menudo, es mucho más sencillo y más cómodo para el experto dar unos intervalos que contengan al valor de la probabilidad. Pensemos en el caso de un sistema experto para medicina, en el que necesitamos conocer la probabilidad de ocurrencia de una determinada enfermedad en una población dada. El experto médico se puede sentir menos forzado si puede expresar este conocimiento por medio de intervalos: la probabilidad está entre el 1% y el 3%. De hecho, es muy común que en los libros médicos los valores de las frecuencias de ocurrencia aparezcan especificadas de manera imprecisa e incluso, en muchas ocasiones, usando términos de significado vago (*En la mayoría de las ocasiones ..., Muy raramente ocurre que...*).

En general, los sistemas expertos bayesianos están diseñados para trabajar con valores exactos de probabilidades. Se podría pensar en transformar los intervalos de probabilidad en valores exactos, considerando por ejemplo el punto medio del intervalo, pero de esta forma estaríamos trabajando con probabilidades hipotéticas sin tener una idea exacta de cómo afectaría una variación de estas probabilidades en nuestros resultados finales.

Desde el principio de la investigación en redes bayesianas se prestó alguna atención a este problema desarrollando algoritmos para el caso en que las probabilidades se conocían de forma imprecisa [128, 47]. Estos trabajos estaban enfocados como un problema de sensibilidad tratando de analizar cómo cambiaban los resultados finales en función de los parámetros. El procedimiento de cálculo era aproximado: los intervalos calculados contenían a los intervalos que se obtendrían para todos los valores posibles de las probabilidades iniciales (considerando como valores posibles todos los valores del intervalo de probabilidad dado). La razón es que los resultados intermedios (mensajes de los algoritmos de propagación) se representaban también en forma de intervalos. Sin embargo, en la teoría de las probabilidades imprecisas [134] existen informaciones que no se pueden representar mediante intervalos, produciéndose una pérdida de información cada vez que se aproximan por intervalos de probabilidad. De esta forma, los procedimientos empleados, por su propia naturaleza, impedían que los resultados pudiesen ser exactos. De hecho, con mucha facilidad los intervalos terminaban siendo $[0, 1]$, es decir no informaban nada en absoluto.

Cano, Moral y Verdegay-López [24] propusieron por primera vez un algoritmo de propagación exacto para probabilidades imprecisas. Este algoritmo suponía que para cada variable existe un conjunto convexo de probabilidades condicionadas dados los valores de sus padres. Su principal valor fue el de plantear el problema del cálculo con probabilidades imprecisas como un problema de propagación de conjuntos convexos de distribuciones de probabilidad. Esto permitió proponer un cálculo local que proporcionaba resultados finales correctos. Este enfoque tenía dos fundamentos que lo hicieron posible:

- El desarrollo de la teoría de probabilidades imprecisas basada en conjuntos convexos [134, 22, 98]. Aunque los conjuntos convexos ya se habían considerado en 1.967 por Dempster [39], en estos años se avanza notablemente en cuestiones cómo elicitación de informaciones, conceptos básicos como condicionamiento, combinación e independencia.

- La especificación abstracta de los algoritmos de propagación por Shafer y Shenoy [118]

que permitió su generalización a otros modelos de representación de la incertidumbre y, en particular, el que nos ocupa de los conjuntos convexos de distribuciones de probabilidad.

Sin embargo, algunas importantes cuestiones quedaron abiertas en este trabajo:

- La complejidad de los cálculos era demasiado alta, de manera que la imprecisión en la especificación de la red no podía ser elevada si queríamos obtener resultados en tiempo razonable. De hecho, los algoritmos presentados eran útiles sólo cuando la estructura del grafo era muy simple (por ejemplo un poliárbol con pocos valores por variable) o la imprecisión aparecía sólo para unas pocas variables de la red.

- Como se suponía que inicialmente la información venía representada por medio de conjuntos convexos de probabilidad, si teníamos intervalos había que hacer una transformación de los intervalos en convexos. El problema es que los conjuntos convexos resultantes eran demasiado complejos, haciendo totalmente inviable una enumeración de sus puntos extremos, con lo que mucho menos se podrían aplicar los algoritmos de propagación que ya tenían importantes problemas de eficiencia.

Objetivos y Metodología

El objetivo básico de esta memoria es proponer algoritmos de propagación aproximados para grafos de dependencias en los que las distribuciones de probabilidad condicionadas vengan dadas por medio de intervalos de probabilidad.

Para ello se utilizan dos técnicas básicas:

- La especificación del problema de la propagación como un problema de optimización combinatoria mediante la adición de variables adicionales. Dependiendo del tipo de condicionamiento que se quiera calcular, este problema será un problema de optimización clásico o un problema multiobjetivo. De esta forma se aplican los procedimientos usuales para su resolución: algoritmos genéticos y algoritmos de enfriamiento estocástico.

- El uso de árboles de probabilidad para representar las distribuciones de probabilidad [10]. Esta técnica permite sacar partido de las independencias asimétricas de un problema para obtener una representación más compacta que las tablas de probabilidad. Nosotros hemos desarrollado las algoritmos de propagación para este tipo de representación [16]

aprovechando sus posibilidades en el proceso de cálculo y proponiendo además algoritmos aproximados.

Los árboles de probabilidad nos han permitido además, en el caso de que las distribuciones de probabilidad vengan dados mediante intervalos, obtener una representación eficiente de los conjuntos convexos de probabilidad asociados. Esta representación evita el problema de explosión combinatoria que se producía cuando se enumeraban los puntos extremos. De esta forma los algoritmos inicialmente desarrollados para conjuntos convexos se han aplicado también al caso de intervalos.

Finalmente, nos hemos propuesto validar nuestros algoritmos comprobando experimentalmente que su aplicación es posible a redes Bayesianas que se han usado en aplicaciones reales, y en las que todos los valores exactos de probabilidad se han transformado en intervalos. Para ello se ha trabajado con redes de la colección pública de redes Bayesianas mantenida por Friedman y disponible a través de Internet [51], y también con redes proporcionadas con el software HUGIN [75] y disponibles en la dirección de Internet "`http://www.hugin.dk/networks`".

Contenido de la Tesis

En el capítulo 1 se introducirán los conceptos y resultados básicos para trabajar con probabilidades imprecisas, centrándonos más concretamente en los conjuntos convexos de probabilidad como un mecanismo capaz de representar la incertidumbre. Se estudiarán las posibles representaciones de los conjuntos convexos a través de restricciones lineales o puntos extremos, y las relaciones existentes entre ellas. También se hará un repaso a los conceptos básicos de conjuntos convexos de probabilidades como son la combinación, marginalización y condicionamiento. Estudiaremos también las relaciones existentes entre los conjuntos convexos de probabilidad y los intervalos de probabilidad, estudiando métodos para transformar conjuntos convexos en intervalos y viceversa, mostrando además que los conjuntos convexos son más generales que los intervalos, y que el trabajar directamente con intervalos produce pérdidas de información. Finalmente en este capítulo haremos un repaso al concepto de independencia en conjuntos convexos de probabilidad, que es un tema fundamental para poder construir los algoritmos de propagación.

El capítulo 2 está dedicado al estudio de la propagación de probabilidades imprecisas. Se comienza haciendo una clasificación de los algoritmos de propagación que utilizan cálculos

locales basándonos en el tipo de relaciones de independencia que se consideran. Se analizará la axiomática general de Shafer y Shenoy necesaria para poder definir algoritmos de propagación de incertidumbre. Finalmente se darán las particularizaciones necesarias para poder aplicar los algoritmos de propagación descritos en forma abstracta, al caso de conjuntos convexos de probabilidades dados a través de un conjunto de puntos extremos. Veremos que hay dos tipos de problemas que pueden resolverse mediante el esquema general de los algoritmos de propagación.

El capítulo 3 está dedicado al estudio de una serie de algoritmos que hemos desarrollado para permitir realizar propagaciones con conjuntos convexos de probabilidad representados con sus puntos extremos, cuando no es posible realizar propagaciones exactas debido a la complejidad del problema. El problema de la propagación se planteará en términos de un problema de optimización combinatoria. Se han desarrollado dos tipos de algoritmos. En primer lugar los que hacen uso de la técnica de enfriamiento estocástico y en segundo lugar los que hacen uso de algoritmos genéticos. Se ha realizado una experimentación que muestra que estos algoritmos son bastante prometedores y que permiten obtener buenos resultados en tiempos razonables.

El capítulo 4 analiza la aplicación de los árboles de probabilidad para representar potenciales (funciones) probabilísticas. Estos árboles permiten obtener representaciones más compactas que las tablas a la hora de representar el potencial que representa una distribución de probabilidad o bien un conjunto convexo de probabilidades. La razón de esto, es que los árboles de probabilidad son capaces de explotar las independencias asimétricas entre las variables del problema. Como consecuencia, necesitaremos menos cálculos para llevar a cabo los algoritmos de propagación. En este capítulo se hará un estudio de cómo se pueden obtener los árboles de probabilidad asociados a una distribución de probabilidad y de cómo pueden llevarse a cabo las operaciones de combinación y marginalización necesarias para la propagación. Se desarrollarán también algoritmos aproximados que limitan el tamaño de los árboles a un valor máximo, realizando aproximaciones siempre que sea necesario. Finalmente se hará un estudio experimental del comportamiento de dichos algoritmos en el caso de propagaciones probabilísticas sin imprecisión.

Por último en el capítulo 5 se ven cómo los árboles de probabilidad son especialmente útiles cuando se usan para la propagación de informaciones dadas a través de intervalos. Veremos cómo se pueden representar las matrices de intervalos de probabilidad a través de

árboles de probabilidad, lo que permitirá aplicar los algoritmos estudiados en el capítulo 4. En este capítulo se propone la adaptación de los algoritmos de los capítulos 2 y 3 al caso en que se usen árboles como representación de los potenciales probabilísticos. En concreto se proponen algoritmos que explotan esta estructura para representar intervalos, eliminar puntos no extremos, obtener intervalos y acotar el error final. Para terminar se dará una experimentación para ver cómo se comportan los algoritmos de propagación en el caso exacto, en el caso aproximado con árboles limitados en tamaño y con las técnicas de enfriamiento simulado.

Capítulo 1

Conjuntos Convexos e Intervalos de Probabilidad

1.1 Introducción

En este capítulo introducimos los conceptos y resultados básicos para trabajar con probabilidades imprecisas. Actualmente podemos decir que aún no se han unificado en una teoría única todas las investigaciones que se han realizado en el campo de las probabilidades no exactas. Existen distintos grupos que desde ámbitos muy diversos han contribuido a desarrollar una metodología para la representación y uso de la incertidumbre mediante probabilidades imprecisas. Se han usado así distintos formalismos matemáticos y se han propuesto diferentes operadores, incluso para problemas y transformaciones similares.

Existen numerosas situaciones que pueden dar lugar al uso de probabilidades imprecisas. Entre ellas podemos destacar [138]:

- Cuando existe poca información para evaluar una probabilidad, Walley [134, 135, 137].
- Cuando la información disponible no es lo suficientemente específica. Por ejemplo, cuando extraemos bolas de una urna con 10 bolas, de la que sabemos que 5 son rojas y otras 5 son blancas o negras, pero no sabemos la proporción exacta de cada una de ellas, Dempster [39], Shafer [117], Klir y Folger [83].
- En Robustez Estadística cuando queremos representar desconocimiento sobre la distribución 'a priori', Berger [6] DeRobertis y Hartigan [40].

- Para representar conflictos entre expertos o fuentes de información: mayores inconsistencias deben de dar lugar a informaciones más imprecisas, Walley [133] Moral y Sagrado [99].

Existen distintos modelos matemáticos que han sido asociados en la literatura con las probabilidades imprecisas: Probabilidad Comparativa [49, 50], Teoría de la Posibilidad [141, 42], Medidas Difusas [125, 139, 60], Teoría de la Evidencia [117, 123], Capacidades de orden-2 [71, 29], Intervalos de Probabilidad [140, 35], o Conjuntos Convexos de Probabilidades [24, 134, 37]. De todas ellas, pensamos que los Conjuntos Convexos de Probabilidades son la representación más adecuada para las probabilidades imprecisas, salvo en situaciones concretas en las que algún otro modelo pueda estar mejor adaptado a las características del mismo. Y eso por varios motivos: existe una interpretación específica de los valores numéricos [134, 136] (no se puede decir lo mismo de modelos tan generales como el de las Medidas Difusas). Y además, tiene la potencia suficiente como para que el resultado de las operaciones básicas de la información se pueda representar de nuevo dentro de este modelo sin tener que realizar aproximaciones que involucren una pérdida o modificación de la información, como ocurre con los intervalos de probabilidad [131].

Existen dos formas alternativas de representar los conjuntos convexos de probabilidades. La primera es mediante restricciones lineales [134, 131], incluyendo el caso particular de los intervalos de probabilidad, y la segunda es mediante puntos extremos [98, 131]. En general, la segunda es más apropiada para realizar algunas operaciones y, en particular, para los algoritmos de propagación en grafos de dependencias [13].

En este capítulo vamos a dar los elementos básicos para la representación y uso de las probabilidades imprecisas. Insistiremos en las dos formas de representación y en la obtención de los puntos extremos a partir de intervalos de probabilidad [35]. También revisaremos las operaciones y conceptos básicos con conjuntos convexos de probabilidades: combinación, condicionamiento, independencia, y esperanza matemática, entre otros.

1.2 Representación de la Información Mediante Conjuntos Convexos de Probabilidades

Supongamos que tenemos una población, Ω, y una variable de dimensión n, $(X_1, X_2, \ldots, X_n)$, tal que cada X_i toma valores en un conjunto finito U_i. O sea, cada U_i es el espacio muestral

de cada X_i. Para cada $I \subseteq \{1, \ldots, n\}$, X_I denotará la variable $(X_i)_{i \in I}$. Esta variable tomará valores en el conjunto $\Pi_{i \in I} U_i$ que denotaremos como U_I. Una información sobre la variable $(X_1, X_2, \ldots, X_n)$ podría ser una distribución de probabilidad p, es decir, una función:

$$p : \Pi_{i \in \{1,\ldots,n\}} U_i \to [0, 1]$$

que verifica:

$$\sum_{u \in \Pi_{i \in \{1,\ldots,n\}} U_i} p(u) = 1$$

En esta memoria supondremos que no siempre se tiene un conocimiento tan bueno sobre las variables que nos permita determinar de forma exacta los valores de una distribución de probabilidad.

Definición 1.1 Conjunto de distribuciones convexo
Un conjunto de distribuciones de probabilidad H es convexo, si se cumple que si $p_1, p_2 \in H$ y $\alpha \in [0, 1]$, entonces $\alpha p_1 + (1 - \alpha) p_2 \in H$

En general, supondremos que una información será un conjunto convexo y cerrado, H, de distribuciones de probabilidad, con un conjunto finito de distribuciones extremas. Las distribuciones de probabilidad podrán ser absolutas (válidas para todos los casos) o condicionadas.

Desde un punto de vista matemático, una información para las variables del conjunto X_I será un conjunto convexo, H, de funciones:

$$h : U_I \longrightarrow I\!R_0^+ \tag{1.1}$$

con un conjunto finito de puntos extremos.

Puesto que U_I es un conjunto finito, cada una de estas funciones h vendrá dada por un vector de valores $(h(u))_{u \in U_I}$. Por esta razón usaremos la palabra *vector* o *punto* para referirnos a la función h. Cada uno de estos vectores o puntos tiene una dimensión $|U_I|$, donde $|U_I|$ es el número de elementos de U_I.

Veamos un ejemplo en el que no conocemos una única distribución de probabilidad. Si que conocemos un intervalo sobre algunos de los posibles sucesos.

Ejemplo 1.1 Ejemplo de probabilidades imprecisas
Supongamos una urna que contiene bolas de cuatro colores: blancas (B), rojas (R), negras (N)

y verdes (V). La urna contiene 10 bolas. Una persona debe modelizar la incertidumbre sobre el experimento de extraer una de las bolas que contiene la urna. A esta persona se le deja ver durante un instante el contenido de dicha urna, y se da cuenta de lo siguiente: 2 bolas son blancas, 3 son rojas o negras, 2 son verdes, blancas o rojas, y 3 son verdes, blancas o negras.

Esta información se puede representar mediante los siguientes intervalos de probabilidad:

$$\begin{array}{llll}
\underline{P}(\emptyset) = 0 & \overline{P}(\emptyset) = 0 & \underline{P}(\{B\}) = 0.2 & \overline{P}(\{B\}) = 0.7 \\
\underline{P}(\{R\}) = 0 & \overline{P}(\{R\}) = 0.5 & \underline{P}(\{N\}) = 0 & \overline{P}(\{N\}) = 0.6 \\
\underline{P}(\{V\}) = 0 & \overline{P}(\{V\}) = 0.5 & \underline{P}(\{B,R\}) = 0.2 & \overline{P}(\{B,R\}) = 1 \\
\underline{P}(\{B,N\}) = 0.2 & \overline{P}(\{B,N\}) = 1 & \underline{P}(\{B,V\}) = 0.2 & \overline{P}(\{B,V\}) = 0.7 \\
\underline{P}(\{R,N\}) = 0.3 & \overline{P}(\{R,N\}) = 0.8 & \underline{P}(\{R,V\}) = 0 & \overline{P}(\{R,V\}) = 0.8 \\
\underline{P}(\{N,V\}) = 0 & \overline{P}(\{N,V\}) = 0.8 & \underline{P}(\{B,R,N\}) = 0.5 & \overline{P}(\{B,R,N\}) = 1 \\
\underline{P}(\{B,R,V\}) = 0.2 & \overline{P}(\{B,R,V\}) = 1 & \underline{P}(\{B,N,V\}) = 0.3 & \overline{P}(\{B,N,V\}) = 1 \\
\underline{P}(\{R,N,V\}) = 0.3 & \overline{P}(\{R,N,V\}) = 0.8 & \underline{P}(\{B,R,N,V\}) = 1 & \overline{P}(\{B,R,N,V\}) = 1
\end{array} \tag{1.2}$$

El conjunto de probabilidades asociado contiene, entre otras a las siguientes distribuciones de probabilidad en $\{B, R, N, V\}$

$$\begin{array}{llll}
p_1(B) = 0.7, & p_1(R) = 0.3, & p_1(N) = 0, & p_1(V) = 0 \\
p_2(B) = 0.2, & p_2(R) = 0.6, & p_2(N) = 0.3, & p_2(V) = 0 \\
p_3(B) = 0.2, & p_3(R) = 0, & p_3(N) = 0.3, & p_3(V) = 0.5
\end{array} \tag{1.3}$$

■

Veamos ahora un ejemplo de como un conjunto convexo puede especificarse enumerando únicamente sus puntos extremos.

Ejemplo 1.2 Ejemplo de conjunto convexo representado con puntos extremos

Supongamos que tenemos una variable X *que toma valores en* $U = u_1, u_2, u_3$. *Sea* H *un conjunto convexo para la variable* X *dado a través de los 4 puntos extremos de la siguiente tabla:*

	u_1	u_2	u_3
p_1	*1/2*	*1/2*	*0*
p_2	*0*	*0*	*1*
p_3	*1/2*	*0*	*1/2*
p_4	*0*	*1/2*	*1/2*

Entonces podemos ver en la figura 1.1 una representación de este conjunto convexo en un espacio tridimensional, al ser $|U| = 3$. *Esta representación nos muestra que podemos usar un sistema de coordenadas triangulares si* $|U| = 3$ *y además tenemos que los puntos del conjunto convexo están normalizados (suman 1). En un sistema de coordenadas triangulares cada distribución es un punto en un triángulo equilátero de altura 1. La probabilidad* $p(u_i)$ *es la distancia al lado* u_i.

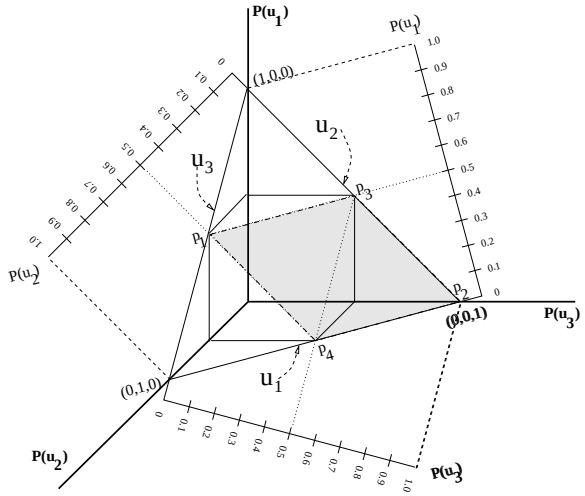


Figura 1.1: Conjunto convexo dado por sus puntos extremos en un sistema de coordenadas triangulares

■

1.2.1 Esperanzas inferiores y superiores

Walley [134] presenta un modelo de representación de la incertidumbre que es equivalente al modelo de los conjuntos convexos de probabilidades. Walley denomina a este modelo *Coherent lower previsions* o bien *Coherent lower expectations*. Este nuevo modelo tiene la ventaja sobre los conjuntos convexos de probabilidades de que tiene una fácil interpretación de su

comportamiento.

Para la definición de la esperanza inferior Walley utiliza funciones X de Ω en $\mathbb{R}$ que se pueden interpretar como juegos en los que se obtendrá como resultado una de las alternativas $\omega \in \Omega$ pero de forma impredecible, y por el que se obtendrá una recompensa $X(\omega)$.

Definición 1.2 Esperanza inferior coherente

Una esperanza inferior se dice que es coherente cuando es la envolvente inferior de algún conjunto de esperanzas lineales, es decir, cuando existe un conjunto no vacío de medidas de probabilidad, $\mathcal{M}$, tal que $\underline{E}(X) = \inf\{E_P(X) : P \in \mathcal{M}\}$ para todos los $X \in \mathcal{K}$, donde $E_P(X)$ denota la esperanza de X respecto a P, $E_P(X) = \sum_{\omega\in\Omega} X(\omega)P(\{\omega\})$.

Walley da la siguiente interpretación para la esperanza inferior: $\underline{E}(X)$ es el supremo del precio de compra para un juego X, o sea, lo máximo que alguien estaría dispuesto a pagar por el juego X.

Definición 1.3 Esperanza superior coherente

Se define la esperanza superior $\overline{E}(X)$ como $\overline{E}(X) = -\underline{E}(-X)$ y que se interpreta como el ínfimo del precio de venta para el juego X.

El siguiente teorema caracteriza las esperanzas inferiores coherentes.

Teorema 1.1 Walley [134]

Sea $\mathcal{K}$ un espacio lineal de funciones X de Ω en $\mathbb{R}$ (es decir si $X \in \mathcal{K}$, $Y \in \mathcal{K}$, $\lambda \in \mathbb{R}$ entonces $\lambda X \in \mathcal{K}, X + Y \in \mathcal{K}$). Entonces $\underline{E}$ es una esperanza inferior coherente si y sólo si satisface (para todo $X, Y \in \mathcal{K}$):

1. $\underline{E}(X) \geq \inf\{X(\omega) : \omega \in \Omega\}$
2. $\underline{E}(\lambda X) = \lambda \underline{E}(X)$ *cuando* $\lambda > 0$
3. $\underline{E}(X + Y) \geq \underline{E}(X) + \underline{E}(Y)$

El siguiente teorema nos muestra que las esperanzas inferiores coherentes son exactamente tan generales como los conjuntos convexos de medidas de probabilidad cerrados. Son formas duales de representar la misma información.

Teorema 1.2 Walley [134]

Existe una correspondencia uno a uno entre las esperanzas inferiores coherentes $\underline{E}$ *(donde* $\underline{E}$ *se define para todas las funciones* X *de* Ω *en* $\mathbb{R}$*) y los conjuntos convexos, no vacíos y cerrados de medidas de probabilidad* $\mathcal{M}$*.*

- $\underline{E}$ *determina de forma única un conjunto convexo* $\mathcal{M}$ *mediante:*

$$\mathcal{M} = \{P \in \mathcal{P} : E_P(X) \geq \underline{E}(X)\}$$

para todas las funciones X *de* Ω *en* $\mathbb{R}$

- $\mathcal{M}$ *determina de forma única una esperanza inferior coherente* $\underline{E}$*, mediante*

$$\underline{E}(X) = \min\{E_P(X) : P \in \mathcal{M}\}$$

para todas las funciones X *de* Ω *en* $\mathbb{R}$

1.2.2 Operaciones básicas en conjuntos convexos

Describimos aquí brevemente algunas operaciones básicas con aplicaciones de un conjunto U_I en los reales y su extensión a conjuntos convexos. Estas operaciones se definen desde un punto de vista puramente matemático sin referencia a las informaciones que representan.

Definición 1.4 Marginal de una función

Si tenemos una función h *de* U_I *en* $\mathbb{R}$*, y* $J \subseteq I$*, entonces la marginal de* h *en el conjunto* U_J *se define como la función* $h^{\downarrow J}$ *definida en* U_J *y dada por* $h^{\downarrow J}(u) = \sum_{v^{\downarrow J}=u} h(v)$*, donde* $v^{\downarrow J} = u$ *es el elemento de* U_J *obtenido al borrar las coordenadas de* $I - J$*.*

Por ejemplo, en el caso probabilístico, de una información global, $p^{X,Y}$, podemos obtener probabilidades para X y probabilidades para Y, de la siguiente manera:

$$p^X(u) = \sum_{v \in V} p^{X,Y}(u,v) = (p^{X,Y})^{\downarrow U}(u)$$

$$p^Y(v) = \sum_{u \in U} p^{X,Y}(u,v) = (p^{X,Y})^{\downarrow U}(u)$$

La definición de marginal para una función puede ser extendida a conjuntos convexos.

Definición 1.5 *(*Marginalización en conjuntos convexos*)*

Si H es un conjunto convexo de funciones en U_I, que tiene como conjunto de puntos extremos a Ext $(H) = \{h_1, \ldots, h_k\}$, y $J \subseteq I$, entonces la marginalización de H a las variables de X_J es el conjunto convexo dado por,

$$H^{\downarrow J} = CH\ \{h_1^{\downarrow J}, \ldots, h_k^{\downarrow J}\} \tag{1.4}$$

donde CH representa el operador de cláusula convexa (o sea el mínimo conjunto convexo que contiene a otro conjunto).

$H^{\downarrow J}$ se obtiene marginalizando en U_J cada una de las funciones h de $Ext(H)$. Sin embargo no todas las marginales, $h_1^{\downarrow J}, \ldots, h_k^{\downarrow J}$, de los puntos extremos de H serán puntos extremos en $H^{\downarrow J}$, aunque si que estamos seguros que Ext $(H^{\downarrow J}) \subseteq \{h_1^{\downarrow J}, \ldots, h_k^{\downarrow J}\}$.

Definición 1.6 Intersección de conjuntos convexos

Si H es un conjunto convexo de funciones en U_I, y H' es un conjunto convexo en U_J, entonces $H \cap H'$ es el conjunto convexo de funciones h definidas en $U_{I \cup J}$ verificando que $h^I \in H$ y $h^J \in H'$.

El conjunto $H \cap H'$ es también un conjunto cerrado y convexo con un número finito de puntos extremos.

Definición 1.7 Multiplicación de funciones

Sea h una función definida de U_I en $\mathbb{R}$ y h' una función definida de U_J en $\mathbb{R}$, entonces definimos la multiplicación de estas dos funciones como la función, $h.h'$, definida sobre $U_{I \cup J}$ y dada por, $h.h'(u) = h(u^{\downarrow I}).h'(u^{\downarrow J})$.

Esta operación también puede ser extendida a conjuntos convexos de funciones.

Definición 1.8 Combinación en conjuntos convexos de probabilidad

Si H es un conjunto convexo de funciones definidas sobre U_I, y H' es el conjunto convexo definido en U_J, siendo $Ext(H) = \{h_1, \ldots, h_k\}, Ext(H') = \{h'_1, \ldots, h'_l\}$, entonces la combinación de H y H' se obtiene mediante la siguiente fórmula:

$$H \otimes H' = \mathrm{CH}\{h_1.h'_1, .., h_1.h'_l, \ldots, h_k.h'_1, .., h_k.h'_l\} \tag{1.5}$$

De la misma manera que en el caso de la marginalización ha sido necesario el uso del operador CH, pues el conjunto $\{h_1.h'_1, .., h_1.h'_l, \ldots, h_k.h'_1, .., h_k.h'_l\}$ podría no ser convexo. También, al igual que con la marginalización, tenemos el hecho que no todas las funciones $h_i.h'_j$ son puntos extremos de $H \otimes H'$, pero estamos seguros que

$$\text{Ext } (H \otimes H') \subseteq \{h_1.h'_1, .., h_1.h'_l, \ldots, h_k.h'_1, .., h_k.h'_l\}$$

1.3 Información Marginal y Condicional en Conjuntos Convexos

En la teoría de la probabilidad, cuando tenemos dos o más variables la forma usual de expresar nuestro conocimiento acerca de ellas no es dando una distribución global de probabilidad para todas ellas. Lo que se suele hacer es dar alguna información sobre las independencias de las variables de forma que podamos construir la distribución global a partir de distribuciones más pequeñas que damos sobre subconjuntos de variables.

Por ejemplo, si tenemos dos variables X e Y que toman valores en dos conjuntos finitos U y V respectivamente, podemos expresar nuestro conocimiento acerca de estas variables mediante una distribución marginal de probabilidad sobre X, p^X, y una distribución condicional de Y dado X, $p^{Y|X}$. O sea con la función:

$$p^{Y|X} : U \times V \to [0, 1] \tag{1.6}$$

verificándose:

$$\sum_{v \in V} p^{Y|X}(u, v) = 1, \forall u \in U \tag{1.7}$$

Mediante estas dos informaciones podemos construir una distribución global de probabilidad para (X, Y), $p^{X,Y}$, según la siguiente expresión:

$$p^{X,Y}(u, v) = p^X(u) \cdot p^{Y|X}(u, v) \tag{1.8}$$

Definición 1.9 Información condicional para conjuntos convexos

Una información condicional sobre Y dado X será un conjunto convexo, $H^{Y|X}$, de funciones, $h : U \times V \to [0, 1]$ que verifican:

$$\sum_{v \in V} h(u, v) = 1, \ \forall u \in U$$

y con un conjunto finito de puntos extremos, $\text{Ext}(H^{Y|X}) = \{h_1, \ldots, h_l\}$.

Esto es más general que suponer que una información condicional es un conjunto convexo de probabilidades para cada posible valor de X, esto es, para cada elemento $u \in U$ [24]. Si tenemos un conjunto convexo, $H^{Y|X=u_i}$, para cada valor, $u_i \in U$, entonces podemos construir un conjunto convexo, $H^{Y|X}$, que represente la información condicional de Y dado X, considerando el conjunto de funciones

$$h : U \times V \to [0, 1]$$

tal que

$$h(u_i, v) \in H^{Y|X=u_i}, \forall u_i \in U$$

Se puede comprobar que la transformación inversa no siempre es posible como muestra el siguiente ejemplo tomado de J. E. Cano [18].

Ejemplo 1.3 J. E. Cano [18]

Supongamos que tenemos $U = \{u_1, u_2\}$ *y* $V = \{v_1, v_2\}$ *y que conocemos los conjuntos convexos* $H^{Y|X=u_1}$ *y* $H^{Y|X=u_2}$. *El primero,* $H^{Y|X=u_1}$, *tiene como puntos extremos,*

	v_1	v_2
p_1	*0.9*	*0.1*
p_2	*0.2*	*0.8*

y el segundo, $H^{Y|X=u_2}$, *tiene los siguientes puntos extremos,*

	v_1	v_2
q_1	*0.6*	*0.4*
q_2	*0.7*	*0.3*

En estas condiciones podemos construir una información condicional, $H^{Y|X}$, *considerando el conjunto convexo generado por los siguientes puntos,*

	(u_1, v_1)	(u_1, v_2)	(u_2, v_1)	(u_2, v_2)
h_1	*0.9*	*0.1*	*0.6*	*0.4*
h_2	*0.9*	*0.1*	*0.7*	*0.3*
h_3	*0.2*	*0.8*	*0.6*	*0.4*
h_4	*0.2*	*0.8*	*0.7*	*0.3*

Sin embargo por medio de un conjunto global condicional podemos expresar relaciones que no pueden ser descompuestas en probabilidades condicionales individuales. Supongamos que conocemos un conjunto convexo $H^{Y|X}$ con el siguiente conjunto de puntos extremos,

	(u_1, v_1)	(u_1, v_2)	(u_2, v_1)	(u_2, v_2)
h_1	*0.9*	*0.1*	*0.6*	*0.4*
h_2	*0.2*	*0.8*	*0.7*	*0.3*

Entonces, las probabilidades condicionales de Y dado $X = u_1$ son las mismas que en el ejemplo anterior, el conjunto convexo con puntos extremos,

	v_1	v_2
p_1	*0.9*	*0.1*
p_2	*0.2*	*0.8*

Para Y, dado $X = u_2$, también tenemos el mismo conjunto de puntos extremos,

	v_1	v_2
q_1	*0.6*	*0.4*
q_2	*0.7*	*0.3*

Pero con la información condicional global, $H^{Y|X}$, hemos expresado que p_1 tiene que ir con q_1, y p_2 con q_2, lo que es imposible de expresar dando simplemente, $H^{Y|X=u_1}$ y $H^{Y|X=u_2}$

■

En el caso de conjuntos convexos de probabilidades si tenemos una información marginal sobre X, H^X, y una información condicional sobre Y dado X, $H^{Y|X}$, podemos obtener una información conjunta H para las variables (X, Y) combinando estas dos informaciones mediante la fórmula 1.5. Podemos formalizar esto mediante la siguiente definición:

Definición 1.10 *Si tenemos un conjunto convexo, H^X, sobre la variable, X, con puntos extremos, $\{p_1, \ldots, p_k\}$ y una información condicional dada por, $H^{Y|X}$, con puntos extremos, $\{h_1, \ldots, h_l\}$, entonces definiremos una información global sobre el par de variables (X, Y), como el conjunto convexo, $H^{X,Y}$, de distribuciones de probabilidad p sobre $U \times V$ generado por los puntos:*

$$H^{X,Y} = CH\{p_1.h_1, \ldots, p_1.h_l, p_2.h_1, \ldots, p_2.h_l, \ldots, p_k.h_1, \ldots, p_k.h_l\}$$

Dos aspectos importantes sobre esta definición son los siguientes:

- En general, no todos los puntos , $p_i.h_j$, son puntos extremos de $H^{Y,X}$. Para calcular una representación mínima del conjunto, $\text{Ext}(H^{Y,X})$, tendremos que aplicar algún algoritmo para obtener la cápsula convexa de todos los puntos y borrar los puntos no extremos del anterior conjunto. En Preparata y Samos [108] y en Edelsbrunner [44] se estudian algunos algoritmos para realizar este cálculo de forma eficiente.

- El conjunto $H^{X,Y}$ no es igual al conjunto

$$\{p.h \,:\, p \in H^X, h \in H^{Y|X}\}$$

  porque este conjunto no es convexo en general (ver [22, 127, 87]). Sin embargo, $H^{X,Y}$, es igual a la cápsula convexa del anterior conjunto y en el peor de los casos estaremos añadiendo alguna probabilidad para ganar en simplicidad ya que los conjuntos convexos se pueden representar por sus puntos extremos.

También es posible el proceso inverso, o sea obtener una información marginal a partir de una información conjunta, o bien una información condicional a partir de una información conjunta. Veamos a continuación como obtener información marginal y condicional a partir de información conjunta para el caso de conjuntos convexos de probabilidad.

La forma de obtener información marginal a partir de la información conjunta es con el uso de la fórmula 1.4. Supongamos dos variables (X, Y), tal que X toma valores en un conjunto finito U, e Y toma valores en el conjunto finito V. Si H es una información sobre (X, Y), o sea un conjunto convexo de distribuciones de probabilidad sobre $U \times V$, con puntos extremos $\{h_1, \ldots, h_n\}$, entonces la marginal de esta información para la variable X, H^X, es el conjunto convexo de probabilidades $H^{\downarrow U}$. O sea, H^X será el conjunto convexo generado por los puntos p_i, $i = 1, \ldots, n$ donde p_i es una función de U en $[0, 1]$ tal que:

$$p_i(u) = \sum_{u \in V} h_i(u, v)$$

O sea,

$$H^X = \text{CH}\{h_1^{\downarrow U}, \ldots, h_n^{\downarrow U}\} \tag{1.9}$$

A partir del conjunto convexo global H definido en $U \times V$ podemos obtener el conjunto condicional, $H^{Y|X}$, de la siguiente forma:

- Sea $\mathcal{H}$ el conjunto de conjuntos convexos condicionales H' sobre Y condicionados a X tal que $H \subseteq H^{\downarrow U} \otimes H'$ y tal que para todo H'' que verifique las mismas condiciones que H' tenemos que si $H^{\downarrow U} \otimes H'' \subseteq H^{\downarrow U} \otimes H'$ entonces $H^{\downarrow U} \otimes H'' = H^{\downarrow U} \otimes H'$.
- $H^{Y|X} = \text{CH}\ (\bigcup_{H' \in \mathcal{H}} H')$

Así pues, si partimos de un conjunto convexo global H definido en $U \times V$, podemos calcular la marginal en U, H^X, y el conjunto condicional $H^{Y|X}$. Pero en general el conjunto global inicial H no podrá ser recuperado a partir de la combinación de H^X y $H^{Y|X}$. Si $H' = H^X \otimes H^{Y|X}$ tendremos en general que H está incluido en H', pero no siempre será igual. En el caso de la teoría de la probabilidad sí que se da siempre la igualdad, o sea a partir de la marginal y la condicional se obtiene la información global original.

En el caso de que $H = H^X \otimes H^{Y|X}$ diremos que X es una *causa* de Y de acuerdo con el conjunto H. Esta definición se basa en la idea intuitiva de que en el caso de una relación causal las imprecisiones sobre X e $Y|X$ no deben de estar correladas [124]. Esto permite en algunos casos determinar relaciones de causalidad incluso cuando sólo existan dos variables. Está claro que pueden existir relaciones de causalidad en ambas direcciones (siempre ocurre en el caso probabilístico), pero también habrá situaciones en las que la relación de causalidad tenga sólo sentido en una dirección.

A partir de ahora consideraremos que si tenemos una información marginal sobre X, H^X, y una condicional sobre Y dado X, $H^{Y|X}$, entonces $H^X \otimes H^{Y|X}$ es el conjunto convexo global para (X, Y). Este conjunto global es el conjunto maximal determinado por la información marginal y la información condicional.

1.4 Conjuntos convexos y restricciones lineales

El método fundamental para tratar las probabilidades imprecisas en la Inteligencia Artificial en la literatura han sido los intervalos de probabilidad [34, 35, 43, 45, 47, 48, 62, 101, 105, 109, 127, 128, 130] también conocidos como probabilidades inferiores y superiores. Sin embargo éste es un modelo menos general que el de los conjuntos convexos de probabilidades [22, 39]. La razón principal es que en el modelo de los intervalos de probabilidad sólo conocemos los límites para sucesos y en conjuntos convexos podemos especificar límites para cualquier función lineal de probabilidades de los sucesos elementales. Más adelante, en el ejemplo 1.8, veremos efectivamente que los conjuntos convexos son más generales que los intervalos de

probabilidad. Un conjunto convexo puede representarse por sus puntos extremos. También es posible especificar un conjunto convexo a través de un conjunto finito de *restricciones* arbitrarias sobre los sucesos elementales. En ambos casos nos interesa que la representación sea mínima. O sea que los puntos sean sólo los puntos extremos y que el conjunto de restricciones sea minimal.

Definición 1.11 Conjunto de restricciones
Sea X una variable que toma valores en un conjunto finito U, entonces un conjunto de restricciones sobre los valores de X, es un conjunto de desigualdades de la forma:

$$R \equiv \sum_{u \in U} \alpha_u . p(u) \leq \beta \tag{1.10}$$

El conjunto de distribuciones de probabilidad que verifica el conjunto de restricciones R es siempre convexo, y lo denotaremos como $\mathcal{H}(R)$. El conjunto de todas las restricciones que verifican un conjunto convexo H lo denotaremos como $\mathcal{R}(H)$. Es evidente que $R \subseteq \mathcal{R}(\mathcal{H}(R))$ y que $\mathcal{H}(\mathcal{R}(H)) = H$.

Si $\mathcal{H}(R) = H$ diremos que el conjunto de restricciones R define el conjunto convexo H. En general, dado un conjunto convexo de distribuciones de probabilidad con un conjunto finito de puntos extremos, existe un conjunto finito minimal de restricciones que lo definen.

Definición 1.12 Conjunto de restricciones minimal
Un conjunto de restricciones R se dice que es minimal si y sólo si para todo conjunto de restricciones $R' \subseteq R$ siendo $\mathcal{H}(R) = \mathcal{H}(R')$ tenemos que $R = R'$.

Definición 1.13 Restricción redundante
Una restricción $r \in R$ es redundante si y sólo si $\mathcal{H}(R) = \mathcal{H}(R - \{r\})$.

En la literatura podemos encontrar algoritmos para buscar la representación mínima de un conjunto convexo de probabilidades. En concreto podemos mencionar los siguientes:

- *Algoritmos de cálculo de la cápsula convexa*: [44, 108] Se usan para borrar los puntos no extremos de un conjunto.

- *Algoritmos de eliminación de la redundancia*: [78] Borran el conjunto de restricciones redundantes de un conjunto finito.

- *Algoritmos de enumeración de vértices*: [94] Calculan todos los puntos extremos de un conjunto convexo definido con un conjunto de restricciones lineales.

A lo largo de esta memoria usaremos la representación dada a través de puntos extremos puesto que es la más apropiada para llevar a cabo las operaciones de marginalización y combinación en conjuntos convexos, aunque con estas operaciones no todos los puntos resultantes son extremos, y será preciso del uso de algún algoritmo de cálculo de cápsula convexa para borrar las probabilidades no extremas. Cuando tengamos una representación dada por restricciones la convertiremos a la representación de puntos extremos.

1.5 Intervalos de probabilidad

Un conjunto de intervalos elementales es un par de funciones $\alpha, \beta : U \to [0,1]$ tal que $\alpha(u) \leq \beta(u), \forall u \in U$. Estas funciones definen la familia de restricciones lineales, $R_{\alpha,\beta}$ dada por,

$$p(u) \geq \alpha(u), \ , p(u) \leq \beta(u), \ \forall u \in U \tag{1.11}$$

Definición 1.14 Conjunto de restricciones consistente
Un conjunto de restricciones elementales $R_{\alpha,\beta}$ es consistente si y sólo si existe un conjunto H no vacío de distribuciones de probabilidad, en el que cada distribución $p \in H$ verifica,

$$\forall u \in U \ \alpha(u) \leq p(u) \leq \beta(u)$$

Sea $R_{\alpha,\beta}$ un conjunto de restricciones lineales, entonces notaremos con $\mathcal{H}(R_{\alpha,\beta})$ al conjunto de distribuciones de probabilidad de dimensión $|U|$ en el que se verifica 1.11. Obviamente $\mathcal{H}(R_{\alpha,\beta})$ debe ser siempre convexo: Cualquier combinación lineal de los elementos de $\mathcal{H}(R_{\alpha,\beta})$ con coeficientes no negativos que sumen 1 pertenecerá a $\mathcal{H}(R_{\alpha,\beta})$.

Ejemplo 1.4 Conjunto de restricciones consistente pero no alcanzable
Supongamos un conjunto $U = \{u_1, u_2, u_3\}$ y sea α y β las funciones definidas como:

	u_1	u_2	u_3
α	0.3	0.3	0.3
β	0.4	0.5	0.5

Hay muchas distribuciones de probabilidad que satisfacen estas restricciones. Por ejemplo:

	u_1	u_2	u_3
p_1	*0.3*	*0.3*	*0.4*
p_2	*0.35*	*0.35*	*0.3*

Sin embargo el conjunto de restricciones que proporcionan α y β no son del todo satisfactorios ya que si $p(u_1)$ y $p(u_2)$ no pueden ser menores que 0.3, no es posible que $p(u_3)$ alcance el valor 0.5. Se puede decir que los intervalos son demasiado grandes. *Parte de los intervalos no pueden alcanzarse nunca por una distribución de probabilidad.* ■

El anterior ejemplo nos muestra que es deseable que el conjunto de restricciones lineales cumpla una propiedad adicional. Esta propiedad adicional la definimos a continuación.

Definición 1.15 Conjunto de restricciones alcanzable
Un conjunto de restricciones lineales $R_{\alpha,\beta}$ que sea consistente será alcanzable, si para cada $u \in U$ existen dos distribuciones de probabilidad p y p' en $\mathcal{H}(R_{\alpha,\beta})$ tal que:

$$\alpha(u) = p(u)$$
$$\beta(u) = p'(u)$$

Campos y otros [35] hacen un estudio general de este tipo de restricciones lineales dadas mediante intervalos. Los resultados más relevantes son los siguientes.

Sea $U = \{u_1, \ldots, u_k\}$ un conjunto de dimensión k y $R_{\alpha,\beta}$ un conjunto de restricciones lineales para U entonces:

- El conjunto de restricciones lineales $R_{\alpha,\beta}$ es consistente si y sólo si

$$\sum_{u_i \in U} \alpha(u_i) \leq 1 \leq \sum_{u_i \in U} \beta(u_i) \tag{1.12}$$

- Todas las restricciones de $R_{\alpha,\beta}$ son alcanzables si y sólo si,

$$\forall u \in U,\ \alpha(u) + \sum_{v \neq u} \beta(v) \geq 1,\ \text{y}\ \beta(u) + \sum_{v \neq u} \alpha(v) \leq 1 \tag{1.13}$$

- Si $R_{\alpha,\beta}$ es un conjunto de restricciones consistente entonces existe un único conjunto de restricciones $R'_{\alpha',\beta'}$ alcanzable asociado que se obtiene mediante:

$$\alpha'(u_j) = \text{Max}(\alpha(u_j), 1 - \sum_{i=1, i \neq j}^{k} \beta(u_i)) \tag{1.14}$$

$$\beta'(u_j) = \text{Min}(\beta(u_j), 1 - \sum_{i=1, i\neq j}^{k} \alpha(u_i)) \tag{1.15}$$

Ejemplo 1.5 Obtención de restricciones alcanzables

Supongamos un conjunto $U = \{u_1, u_2, u_3, u_4\}$ *y sea* α *y* β *las funciones definidas como:*

	u_1	u_2	u_3	u_4
α	*0.2*	*0.1*	*0.4*	*0.1*
β	*0.25*	*0.5*	*0.45*	*0.15*

Es evidente que para u_2 *no se cumple la expresión 1.13. Por tanto es necesario calcular unos nuevos intervalos* $\alpha(u_2), \beta(u_2)$ *a partir de los antiguos mediante el uso de las expresiones 1.14 y 1.15.*

Los nuevos intervalos son:

	u_1	u_2	u_3	u_4
α	*0.2*	*0.15*	*0.4*	*0.1*
β	*0.25*	*0.3*	*0.45*	*0.15*

■

La obtención de los puntos extremos asociados a un convexo definido por intervalos de probabilidad se puede calcular de forma eficiente. Veamos un algoritmo que permite obtener los puntos extremos de $\mathcal{H}(R_{\alpha,\beta})$, visitando cada uno de ellos una sola vez. En este algoritmo se supone que $U = \{u_1, \ldots, u_n\}$. *Prob* es una lista de probabilidades extremas encontradas hasta el momento, y p es la actual probabilidad parcial (o sea $\alpha(u_i) \leq p(u_i) \leq \beta(u_i), \forall i$, donde no necesariamente se cumple la restricción de $\sum_i p(u_i) = 1$. *Expl* es una lista de nodos ya explorados, y λ es la cantidad de probabilidad aun no asignada (o sea $1 - \sum_i p(u_i)$).

Algoritmo 1.1 Obtención de $\mathcal{H}(R_{\alpha,\beta})$ a partir de $R_{\alpha,\beta}$

1. Inicialización:

   $Prob \leftarrow \emptyset$

   $Expl \leftarrow \emptyset$

   $\lambda \leftarrow 1 - \sum_{i \leq n} \alpha(u_i)$

   Para $i = 1$ hasta n

   $p(u_i) \leftarrow \alpha(u_i)$

2. Llamar a $Getprob(p, \lambda, Expl)$ que calcula y añade las probabilidades extremas a $Prob$

■

Algoritmo 1.2 ***Getprob*** **$(p,\lambda,Expl)$**

1. Para $i = 1$ hasta n

   Si no pertenece$(i,Expl)$
   Entonces Si $\lambda < \beta(u_i) - \alpha(u_i)$
   Entonces
   $v \leftarrow p(u_i)$
   $p(u_i) \leftarrow p_i + \lambda$
   Si no pertenece$(p,Prob)$
   Entonces Añadir$(p,Prob)$
   $p(u_i) \leftarrow v$
   En caso contrario
   $v \leftarrow p(u_i)$;
   $p(u_i) \leftarrow \beta(u_i)$;
   Getprob $(p,\lambda - \beta(u_i) + \alpha(u_i),Expl \cup \{i\})$;
   $p(u_i) \leftarrow v$;

■

Este algoritmo usa un árbol de búsqueda implícito donde cada nodo es una probabilidad parcial y un nodo hijo representa un refinamiento de su nodo padre incrementando un componente $p(u_i)$. Los nodos hoja del árbol son probabilidades extremas. El algoritmo puede ser mejorado quitando la necesitad de comprobar si la probabilidad ya está incluida en $Prob$, evitando por tanto una búsqueda innecesaria. El principal cambio en el algoritmo es que debemos mantener cual es el índice que está siendo explorado. El nuevo algoritmo modificado es el siguiente:

Algoritmo 1.3 Obtención de $\mathcal{H}(R_{\alpha,\beta})$ a partir de $R_{\alpha,\beta}$

1. Inicialización:

$Prob \leftarrow \emptyset$

$Expl \leftarrow \emptyset$

$\lambda \leftarrow 1 - \sum_{i \leq n} \alpha(u_i)$

Para $i = 1$ hasta n

$p(u_i) \leftarrow \alpha(u_i)$

2. Si $\lambda = 0$

Entonces Añadir(p,$Prob$)

En caso contrario $Getprob(p, \lambda, Expl, Current)$

■

Getprob es de nuevo una función recursiva que calcula y añade las probabilidades extremas a $Prob$.

Algoritmo 1.4 ***Getprob(p,λ,Expl,Current)***

1. Para $i = 1$ hasta n

Si no pertenece(i,$Expl$)

Entonces Si $\lambda < \beta(u_i) - \alpha(u_i)$

Entonces

$v \leftarrow p(u_i)$

$p(u_i) \leftarrow p(u_i) + \lambda$

Añadir(p,$Prob$)

$p(u_i) \leftarrow v$

En caso contrario

Si $i > Current$

Entonces Si $\lambda - \beta(u_i) + \alpha(u_i) > 0$ y $\beta(u_i) - \alpha(u_i) > 0$

Entonces

$v \leftarrow p(u_i)$;

$p(u_i) \leftarrow \beta(u_i)$;

Getprob (p,$\lambda - \beta(u_i) + \alpha(u_i)$,$Expl \cup \{i\}$,i);

$p(u_i) \leftarrow v$;
En caso contrario Si $(\beta(u_i) - \alpha(u_i)) > 0$
Entonces
$v \leftarrow p(u_i)$
$p(u_i) \leftarrow p(u_i) + \lambda$
Añadir($p, Prob$)
$p(u_i) \leftarrow v$

■

La complejidad del algoritmo en el peor de los casos es exponencial en el número de elementos de U. De hecho el número de puntos extremos puede llegar a ser exponencial. Sin embargo, en la práctica suele haber un número bastante más reducido de puntos extremos, y el algoritmo es óptimo en el sentido que su complejidad es lineal en función del resultado que se calcula.

Ejemplo 1.6 Obtención de puntos extremos a partir de los intervalos
Aplicando el algoritmo anterior a los intervalos obtenidos en el ejemplo 1.5, o sea a:

	u_1	u_2	u_3	u_4
α	*0.2*	*0.15*	*0.4*	*0.1*
β	*0.25*	*0.3*	*0.45*	*0.15*

obtendremos los siguientes puntos extremos:

	u_1	u_2	u_3	u_4
p_1	*0.25*	*0.25*	*0.40*	*0.10*
p_2	*0.25*	*0.20*	*0.45*	*0.10*
p_3	*0.25*	*0.20*	*0.45*	*0.15*
p_4	*0.25*	*0.20*	*0.40*	*0.15*
p_5	*0.20*	*0.30*	*0.40*	*0.10*
p_6	*0.20*	*0.25*	*0.45*	*0.10*
p_7	*0.20*	*0.20*	*0.45*	*0.15*
p_8	*0.20*	*0.25*	*0.40*	*0.15*

■

El algoritmo 1.3 puede ser aplicado cuando el conjunto de restricciones $R_{\alpha,\beta}$ se da sobre una distribución global de probabilidad, la cual puede corresponder a varias variables. Cuando tenemos una distribución de probabilidad condicional de una variable Y, que toma valores en el conjunto V, a otra variable X, que toma valores en el conjunto U, podremos aplicar también este algoritmo. Para ello se aplicará de forma independiente el algoritmo 1.3 a las distribuciones de probabilidad: $P(Y|X = u_i)$, $u_i \in U$, obteniendo los puntos extremos de los conjuntos convexos $H^{Y|X=u_i}$ $u_i \in U$. Para construir el conjunto convexo global $H^{Y|X}$ para la información condicional podemos proceder de la misma forma que en el ejemplo 1.3. Veamos un ejemplo de como hacer todo esto.

Ejemplo 1.7 Obtención de puntos extremos a partir de una distribución de probabilidad condicional expresada con intervalos

Supongamos una distribución de probabilidad condicional $P(Y|X)$ sobre la que se establecen un conjunto de restricciones $R_{\alpha,\beta}$ que establecen un intervalo para cada probabilidad. Supongamos que Y toma valores en el conjunto $V = \{v_1, v_2\}$ y X en el conjunto $U = \{u_1, u_2\}$. Sea $R_{\alpha,\beta}$ el conjunto de restricciones dado por la siguiente tabla:

	v_1	v_2
u_1	$[0.2, 0.3]$	$[0.7, 0.8]$
u_2	$[0.4, 0.6]$	$[0.4, 0.6]$

en la que el valor izquierdo de cada intervalo es el valor α y el valor derecho es el valor β. Esta tabla puede ponerse también de la siguiente forma:

	u_1		u_2	
	v_1	v_2	v_1	v_2
α	0.2	0.7	0.4	0.4
β	0.3	0.8	0.6	0.6

Aplicando el algoritmo 1.3 para $X = u_1$ obtenemos el conjunto convexo $H^{Y|X=u_1}$ dado por los siguientes puntos extremos:

	v_1	v_2
p_1	0.2	0.8
p_2	0.3	0.7

y el segundo, $H^{Y|X=u_2}$, tiene los siguientes puntos extremos,

	v_1	v_2
q_1	*0.4*	*0.6*
q_2	*0.6*	*0.4*

La información global $H^{Y|X}$ se puede dar mediante el conjunto convexo con los siguientes puntos extremos:

	(u_1, v_1)	(u_1, v_2)	(u_2, v_1)	(u_2, v_2)
h_1	*0.2*	*0.8*	*0.4*	*0.6*
h_2	*0.2*	*0.8*	*0.6*	*0.4*
h_3	*0.3*	*0.7*	*0.4*	*0.6*
h_4	*0.3*	*0.7*	*0.6*	*0.4*

Obsérvese como el número de puntos extremos en $H^{Y|X}$ es el producto de los puntos extremos en cada conjunto $H^{Y|X=u_i}$. ■

1.5.1 Conjuntos convexos determinados por probabilidades inferiores y superiores

Otro caso particular de conjuntos convexos son los que se especifican con un par de probabilidades inferiores y superiores (P_*, P^*). Este tipo de conjuntos convexos han sido estudiados en [34, 36, 39, 71].

Sea el conjunto finito $U = \{u_1, \ldots, u_n\}$, y sean P_* y P^* un par de funciones (P_*, P^*) que asignan un intervalo de valores a cada suceso de la siguiente forma:

$$(P_*, P^*) : 2^U \rightarrow [0, 1] \tag{1.16}$$

donde $P_*(A) \leq P^*(A),\ \forall A \subseteq U$.

En la práctica, cuando no asignemos un intervalo para alguno de los sucesos (subconjuntos de U) supondremos que tiene asignado el intervalo $[0, 1]$.

El sistema de intervalos define un conjunto convexo de distribuciones de probabilidad que viene dado por la siguiente expresión:

$$H_{(P_*,P^*)} = \{p : p \in \mathcal{P}, P_*(A) \leq P(A) \leq P^*(A)\} \tag{1.17}$$

donde $\mathcal{P}$ es el conjunto de todas las distribuciones de probabilidad posibles en U y P es la medida de probabilidad asociada a la distribución p.

A su vez, dado un conjunto, H, podemos definir para cada subconjunto, $A \subseteq U$, dos valores extremos de probabilidades (un intervalo) de la siguiente forma:

$$\underline{P}(A) = \inf_{p \in H} \sum_{a \in A} p(a) = \inf_{p \in H} P(A) \tag{1.18}$$

$$\overline{P}(A) = \sup_{p \in H} \sum_{a \in A} p(a) = \sup_{p \in H} P(A) \tag{1.19}$$

siendo P la medida de probabilidad asociada a la distribución p. Este par de funciones no son equivalentes al conjunto original, H, pues diferentes H pueden definir el mismo par de funciones.

En general, si partimos de un sistema de intervalos (P_*, P^*), calculamos su conjunto convexo asociado $H_{(P_*, P^*)}$, y a partir de él, un nuevo sistema de intervalos $(\underline{P}, \overline{P})$ tendremos que:

$$P_*(A) \leq \underline{P}(A) \leq \overline{P}(A) \leq P^*(A) \tag{1.20}$$

Si partimos de un conjunto convexo H, calculamos los intervalos asociados $(\underline{P}, \overline{P})$ mediante las expresiones 1.18 y 1.19, y a partir de los intervalos $(\underline{P}, \overline{P})$ volvemos a calcular un conjunto convexo H' con la expresión 1.17 tendremos en general que $H \subseteq H'$.

Veamos un ejemplo tomado de J. E. Cano [18] sobre las relaciones entre intervalos de probabilidad y los conjuntos convexos.

Ejemplo 1.8 Los conjuntos convexos son más generales que las probabilidades inferiores y superiores (J. E. Cano [18])
Supongamos que tenemos $U = \{u_1, u_2, u_3\}$ *y el conjunto convexo asociado, cuyos puntos extremos son los siguientes:*

	u_1	u_2	u_3
p_1	*1/2*	*1/2*	*0*
p_2	*0*	*0*	*1*

Este conjunto convexo está representado gráficamente en la figura 1.2 mediante un sistema de coordenadas triangulares. Puede verse que el conjunto convexo está formado por todos los puntos que hay en la línea que une p_1 *y* p_2.

Se puede ver que este conjunto convexo determina los siguientes intervalos de probabilidad,

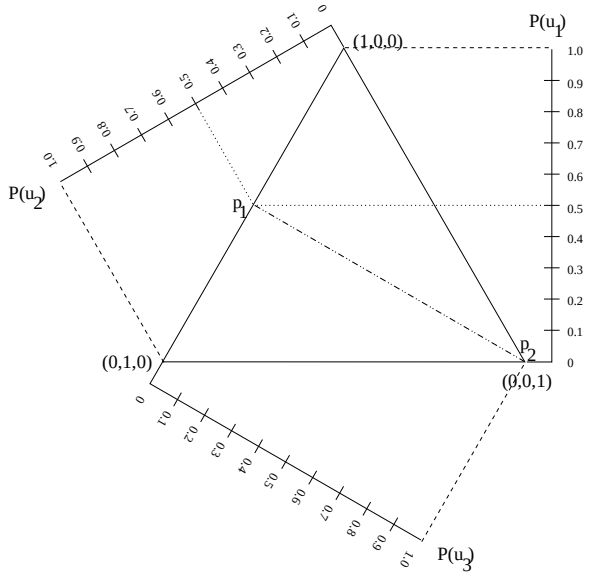


Figura 1.2: Conjunto convexo original

$$P(u_1) \in [0, 1/2] \quad P(u_1, u_2) \in [0, 1]$$
$$P(u_2) \in [0, 1/2] \quad P(u_2, u_3) \in [1/2, 1]$$
$$P(u_3) \in [0, 1] \quad P(u_1, u_3) \in [1/2, 1]$$

Estos intervalos de probabilidad definen a su vez el siguiente conjunto convexo de probabilidades;

	u_1	u_2	u_3
p_1	*1/2*	*1/2*	*0*
p_2	*0*	*0*	*1*
p_3	*1/2*	*0*	*1/2*
p_4	*0*	*1/2*	*1/2*

Este segundo conjunto convexo está representado gráficamente mediante un sistema de coordenadas triangulares en la figura 1.3. Este segundo conjunto convexo es mayor que el de la figura 1.2 del que partimos originalmente. Es decir, hemos aumentado el número de distribuciones de probabilidad posibles.

■

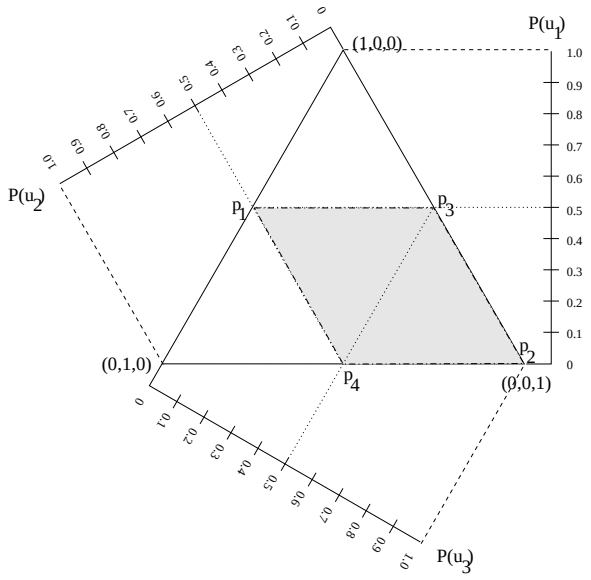


Figura 1.3: Conjunto convexo obtenido a partir de los intervalos para cada suceso

1.6 Información 'a posteriori': Condicionamiento

Hemos visto hasta ahora reglas generales que nos permiten transformar información que es válida para todos los elementos de una población. Veamos cómo particularizar las reglas generales cuando disponemos de observaciones sobre algunas de las variables de nuestro problema en el caso de conjuntos convexos. Esto es conocido como *condicionamiento* [97]. Cuando calculamos información marginal teniendo en cuenta las observaciones disponibles para algunas de las variables de nuestro problema, la información resultante la conoceremos como información 'a posteriori'.

Definición 1.16 Condicionamiento de Moral y Campos [98, 22, 97]

Supongamos un conjunto convexo para la variable X*:* $H = CH\{p_1, \ldots, p_k\}$ *y que hemos observado que* X pertenece a A*, entonces el resultado del condicionamiento es el conjunto convexo,* $H|_1 A$*, generado por los puntos* $\{p_1.l_A, \ldots, p_k.l_A\}$ *donde* l_A *es la verosimilitud asociada al conjunto* A*:*

$$l_A(u) = 1, \qquad si \quad u \in A$$
$$l_A(u) = 0, \qquad en\ caso\ contrario$$

El conjunto $H|_1 A$ también se puede representar como $H \otimes \{l_A\}$.

Es importante remarcar que $H|_1 A$ es un conjunto convexo de funciones con distintos factores de normalización. Si llamamos $r = \sum_{u \in U} p(u).l_A(u) = p(A)$, entonces calculando $(p.l_A)/r$ (cuando $r \neq 0$) obtenemos la distribución de probabilidad condicional $p(.|A)$.

Definición 1.17 Condicionamiento de Dempster [39]

Supongamos un conjunto convexo para la variable X: $H = CH\{p_1, \ldots, p_k\}$ *y que hemos observado que* X pertenece a A, *entonces el resultado del condicionamiento es el conjunto convexo,* $H|_2 A$, *generado por los puntos* $\{p(.|A) : p \in H, p(A) \neq 0\}$ *donde* $p(.|A) = (p.l_A)/r$, $r = \sum_{u \in U} p(u).l_A(u) = p(A)$.

El conjunto $H|_2 A$ fue propuesto por Dempster [39] como conjunto de condicionamiento, y ha sido ampliamente usado en la literatura. El uso de este conjunto tiene el inconveniente de que se pierde información, ya que no se tienen en cuenta los valores de los factores de normalización, $r = p(A)$, los cuales son una verosimilitud o posibilidad inducida por la observación en el conjunto de posibles distribuciones de probabilidad. El resultado del primer condicionamiento es un conjunto de distribuciones de probabilidad $p(.|A) \in H|_2 A$ con un valor de verosimilitud o posibilidad $p(A)$ asociado a cada una de estas probabilidades.

Para asociar intervalos al conjunto convexo $H|_2 A$ mediante el condicionamiento de Dempster podemos usar las fórmulas 1.18 y 1.19, ya que $H|_2 A$ es un conjunto convexo donde todos los puntos tienen el mismo factor de normalización. Para asociar intervalos de probabilidad al conjunto convexo $H|_1 A$, Campos y Moral [98] proponen el siguiente procedimiento:

- Considerar el conjunto convexo condicional $H|_1 A$ formado por los puntos extremos: $\{p_1.l_A, \ldots, p_k.l_A\}$
- Normalizar cada punto extremo, calculando $p_i(.|A)$, y asignándole al mismo tiempo un valor de posibilidad, $\pi(p_i(.|A))$, igual a $r_i/(\max r_k)$.
- Si Π es la medida de posibilidad definida por el anterior valor de posibilidad y N es su medida dual de necesidad en el conjunto de puntos extremos:

$$\Pi(H) = \sup \pi(h), h \in H$$

$$N(H) = 1 - \Pi(\overline{H})$$

entonces los intervalos inferiores y superiores se calculan como sigue:

$$P_*(B|A) = I(P_i(B|A)|N) \tag{1.21}$$

$$P^*(B|A) = I(P_i(B|A)|\Pi) \tag{1.22}$$

donde I es la integral de Choquet [30], que tiene la siguiente expresión:

$$I(P_i(B|A)|N) = \int_0^{+\infty} N(\{P_i(.|A) \,:\, P_i(B|A) \geq \alpha\})d\alpha$$

$$I(P_i(B|A)|\Pi) = \int_0^{+\infty} \Pi(\{P_i(.|A) \,:\, P_i(B|A) \geq \alpha\})d\alpha$$

Cuando tenemos dos variables X e Y que toman valores en U y V respectivamente y H es un conjunto convexo global de probabilidades para estas dos variables, entonces denotaremos como $H^U|_1(Y \in B)$ el condicionamiento de Moral y Campos [98] de H al conjunto $U \times B$ y la marginalización del resultado a U. En otros términos,

$$H^U|_1(Y \in B) = (H|_1 U \times B)^{\downarrow U} \tag{1.23}$$

De forma análoga, para el condicionamiento de Dempster [39] consideraremos

$$H^U|_2(Y \in B) = (H|_2 U \times B)^{\downarrow U} \tag{1.24}$$

Si $B = \{v\}$, entonces $H^U|_i(Y \in B)$ lo denotaremos como $H^U|_i(Y = v)$ $(i = 1, 2)$.

Si l_Y es la función de verosimilitud sobre Y, $l_Y : V \to [0, 1]$, entonces $H^U|_1 l_Y$ denotará el condicionamiento de Moral y Campos [98] de H a la verosimilitud en $U \times V$ dada por $l(u, v) = l_Y(v)$ y la marginalización del resultado a U:

$$H^U|_1 l_Y = (H \otimes \{l_Y\})^{\downarrow U} \tag{1.25}$$

De igual forma, $H^U|_2 l_Y$ denotará el condicionamiento de Dempster [39] a l y la posterior marginalización a U.

Cuando disponemos de una información marginal H^X sobre la variable X y una información condicional $H^{Y|X}$ sobre la variable Y dado X, y una función de verosimilitud l_Y sobre Y podemos obtener la información 'a posteriori' sobre X condicionada a esta función de verosimilitud con una extensión del Teorema de Bayes para conjuntos convexos de probabilidad mediante la siguiente expresión:

$$H^X|_i l_Y = [(H^X \otimes H^{Y|X}) \otimes \{l_Y\}]^{\downarrow U} \quad (i = 1, 2) \tag{1.26}$$

1.7 Independencia

La independencia en conjuntos convexos de probabilidades tiene un papel relevante en el desarrollo de algoritmos de propagación. Una dificultad que aparece es que no existe un único concepto de independencia para conjuntos convexos [37]. Existen varias definiciones distintas que no son equivalentes. Incluso tienen sentido conceptos de independencia que no son una generalización de la independencia probabilística. En este apartado recogeremos dos definiciones de independencia, aquellas que se han usado fundamentalmente para los algoritmos de propagación.

La primera de ellas no incluye como caso particular a la independencia probabilística. De hecho, en el caso de una única distribución de probabilidad siempre se verifica.

Definición 1.18 Independencia marginal

Dado un convexo de probabilidades H sobre las variables (X, Y, Z), que toman valores en los conjuntos U, V, W, respectivamente, diremos que existe independencia marginal de X e Y dada Z si y sólo si,

$$H = H^{X,Z} \cap H^{Y,Z} = \{p : p^{\downarrow U,W} \in H^{X,Z}, p^{\downarrow V,W} \in H^{Y,Z}\} \tag{1.27}$$

donde $H^{X,Z}$ y $H^{Y,Z}$ son los conjuntos marginales de H en $U \times W$ y $V \times W$.

La idea intuitiva de esta definición se entiende mejor en el caso de independencia no condicionada. Si tenemos dos variables, se da independencia no condicionada cuando $H = H^X \cap H^Y$. La situación es: tenemos dos variables para las que sólo conocemos la información marginal. Es decir no conocemos ninguna interacción entre ellas. Pero eso no implica que no exista esta interacción. Así nuestro conocimiento global está formado por todas las distribuciones bidimensionales que tengan las marginales dadas. En algunas de estas habrá independencia probabilística clásica y en otras no.

El siguiente ejemplo de este tipo de independencia ha sido dado por Couso, Moral y Walley [31].

Ejemplo 1.9 Independencia marginal

Supongamos que tenemos dos urnas. Cada una de las urnas contiene 10 *bolas que son blancas o rojas. Sabemos que la primera urna contiene* 5 *rojas,* 2 *blancas, y* 3 *de color desconocido, y que la segunda urna contiene* 3 *rojas,* 3 *blancas, y* 4 *de color desconocido. Una bola se*

elige aleatoriamente de cada una de las urnas, pero no se supone que existe independencia estocástica y es posible que se use un procedimiento conjunto para elegir las bolas en las urnas. Por ejemplo, podría ocurrir que en cada urna las bolas estén numeradas del 1 *al* 10 *en cada urna y que las bolas en ambas urnas se elijan de acuerdo con un número aleatorio i entre* 1 *y* 10. *La independencia entre los colores de las urnas está en nuestra completa falta de información sobre la interacción entre las dos extracciones. Sólo tenemos información sobre las distribuciones marginales.* ■

La segunda de ellas es una generalización de la independencia probabilística. De entre las de este tipo que se han estudiado para conjuntos convexos nosotros vamos a considerar la más restrictiva, que es por otra parte la que más se ha aplicado en los algoritmos de propagación con probabilidades imprecisas. Comenzaremos por la independencia incondicional.

Definición 1.19 Independencia fuerte

Dado un convexo de probabilidades H sobre las variables (X, Y), que toman valores en los conjuntos U, V, respectivamente, diremos que existe independencia fuerte de X e Y si y sólo si,

$$H = H^X \otimes H^Y = CH\{p_1.p_2 : p_1 \in H^X, p_2 \in H^Y\} \tag{1.28}$$

donde H^X y H^Y son los conjuntos marginales de H en U y V.

La idea es que el conjunto de distribuciones de probabilidad conjuntas es el que se obtiene considerando todas las marginales en H^X, todas las marginales en H^Y, y formando todas las conjuntas que se pueden obtener a partir de estas marginales y la condición de independencia probabilística. Es importante hacer notar que no es suficiente que todos los puntos extremos, p, de H verifiquen la condición de independencia probabilística. Es necesario además que para cada extremo, p_1 de H^X, y cada extremo p_2 de H^Y, haya un punto en H que los tenga como marginales.

El siguiente ejemplo se puede encontrar en [31].

Ejemplo 1.10 Independencia fuerte

Consideremos las dos urnas del ejemplo anterior con la misma distribución de bolas, y supongamos que elegimos una bola de cada urna de forma independiente. Las posibles frecuencias de

obtener una bola roja en cada urna son $0.5, 0.6, 0.7$ *y* 0.8 *para la primera urna, y* $0.3, 0.4, 0.5, 0.6$ *y* 0.7 *para la segunda. Como los dos sucesos son estocásticamente independientes, la probabilidad de elegir dos bolas rojas es el producto de las dos frecuencias relativas. El intervalo* $[0.15, 0.56]$ *es la cláusula convexa de estas probabilidades, y representa nuestra incertidumbre sobre este suceso en el caso de independencia fuerte.* ■

La definición de independencia condicional que damos aquí es un poco más restrictiva que la dada por Campos y Moral [37], pero es la que usualmente se aplica cuando se consideran grafos dirigidos acíclicos para representar las relaciones de independencia entre las variables de un problema. Además en este caso es posible demostrar los axiomas de independencia condicional propuestos por Geiger y Pearl [54] y que estudiaremos en el siguiente capítulo.

Definición 1.20 Independencia condicional fuerte

Si $H^{X,Y,Z}$ *es un conjunto convexo de probabilidades sobre las variables* (X, Y, Z)*, que toman valores en los conjuntos* U, V, W*, respectivamente, diremos que existe* independencia fuerte *de* X *e* Y *dada* Z *si y sólo si:*

$$H^{X,Y,Z} = H_1 \otimes H_2$$

donde H_1 *es un conjunto convexo de funciones sobre* $U \times W$ *y* H_2 *es el conjunto convexo de funciones sobre* $V \times W$*.*

En la anterior definición no hay ninguna necesidad que H_1 ó H_2 sean un conjunto convexo marginal o condicional obtenidos a partir de $H^{X,Y,Z}$.

Capítulo 2

Propagación de Probabilidades Imprecisas

2.1 Introducción

Los algoritmos de propagación en estructuras gráficas fueron desarrollados originalmente para el caso probabilístico [79, 90, 106, 115]. Con ellos se logra calcular probabilidades condicionadas de forma eficiente en distribuciones que involucran un gran número de variables, lo que implica que un cálculo directo es totalmente inviable.

Estos algoritmos se basan en aprovechar las relaciones de independencia de las variables para obtener una factorización de la distribución de probabilidad global. Esta factorización se explota para desarrollar un procedimiento de cálculo local, que trabaja directamente con los factores, y que es correcto desde el punto de vista de la distribución global.

Pronto se descubrió que los algoritmos de propagación eran aplicables también a otros formalismos. Shafer y Shenoy [118, 120, 119] introdujeron un sistema de axiomas que debían verificarse en una teoría de representación de la incertidumbre para poder usar los algoritmos de propagación en estructuras gráficas. Esta axiomática ha sido también estudiada por Cano, Delgado y Moral [19, 20] haciendo énfasis en los problemas relacionados con la especificación de la información en grafos dirigidos.

Una de estas particularizaciones del sistema de Shafer y Shenoy la dieron Cano, Moral y Verdegay-López [23, 19] para el caso de los conjuntos convexos de probabilidades, proporcionando el primer algoritmo exacto para llevar a cabo el cálculo local de probabilidades

condicionadas imprecisas bajo relaciones de independencia.

Sin embargo, el cálculo local con probabilidades imprecisas ha sido objeto en la literatura de distintas aproximaciones, según se presenten o no relaciones de independencia, o si el cálculo es exacto o aproximado. Un estudio de los distintos problemas planteados y de los métodos utilizados para resolverlos puede encontrarse en Cano y Moral [17]. En resumen, se puede considerar la siguiente clasificación:

- *Métodos sin relaciones de independencia.*- Se dan restricciones sobre los valores de probabilidad de otros sucesos de interés. Se trata de calcular los límites inferior y superior para la probabilidad de ciertos sucesos. Se incluye también el caso de probabilidades condicionadas. Existen dos tipos de enfoques:
  - *Exactos.*- Algoritmos de propagación de restricciones exactos se dan por Verdegay-López [131] basados en la axiomática de Shafer y Shenoy. Otro método totalmente distinto para este problema es el proporcionado por Hansen y Jaumard [64], basado en programación lineal con generación de columnas implícitas.
  - *Aproximados.*- Estos algoritmos están basados en la aplicación de reglas de inferencia que permiten la obtención de nuevas cotas a partir de las existentes. En la mayoría de los casos restringen el tipo de las informaciones que pueden aparecer en nuestra base de conocimiento, y además no son completos, en el sentido de que las cotas obtenidas son correctas pero no óptimas en general. En esta línea se encuentran los trabajos de Amarger, Dubois y Prade [3], Thöne [129], Lukasiewicz [92], y Salo [114].
- *Algoritmos bajo relaciones de independencia.*- En este caso se suponen unas relaciones de independencia que vienen dadas por un grafo dirigido acíclico y una información imprecisa sobre las probabilidades condicionadas. Se trata de calcular cotas para las probabilidades con las que distintas variables toman sus valores condicionadas a ciertas observaciones en otras variables. Existen algoritmos basados en la generalización del algoritmo de Pearl [106]. El problema fundamental es que está restringido al caso de hiperárboles, lo que constituye una limitación importante. En esta línea se encuentran el algoritmo aproximado de Tessem [128] y el exacto para el caso de variables bivaluadas de Fagiouli y Zaffalon [46]. El algoritmo exacto de Shacter [119] se ha generalizado de forma aproximada al caso de probabilidades imprecisas por Fertig y Brease [47, 11, 48].

Como ya hemos indicado antes, un algoritmo de propagación exacto es el presentado por Cano, Moral y Verdegay-López [23, 19]. El problema fundamental es la complejidad debido al tamaño de los convexos calculados. Los algoritmos aproximados que hemos mencionado trabajan con intervalos de probabilidad, una representación que resulta insuficiente ya que la combinación de representaciones intervalares puede dar lugar a un convexo que no es representable mediante intervalos de probabilidad. Existen algoritmos aproximados que trabajan con conjuntos convexos directamente. Entre ellos podemos destacar los de Cano, Cano y Moral [12, 13], basados en la técnica de enfriamiento simulado, los de Cano y Moral [15], basados en algoritmos genéticos, y los de Cozman [32, 33], que usan técnicas de búsqueda basadas en el gradiente o el algoritmo EM (media y aproximación).

En este capítulo nos centraremos en los algoritmos exactos para conjuntos convexos de probabilidades, dejando para el siguiente capítulo la descripción de los algoritmos aproximados que hemos desarrollado. Previamente describiremos de forma abstracta los algoritmos de propagación en estructuras gráficas.

En la sección 2.2 estudiaremos la axiomática de Shafer y Shenoy necesaria para poder construir algoritmos de propagación de incertidumbre. También se estudiarán los grafos de dependencias como un método válido para poder representar relaciones de independencia entre variables. En la sección 2.3 describiremos de forma abstracta los algoritmos de propagación. En la sección 2.4 daremos un breve repaso a algoritmos que propagan conjuntos convexos representados por conjuntos de restricciones, donde no se consideran relaciones de independencia entre las variables del problema. En la sección 2.5 se estudiará un algoritmo de propagación exacto de conjuntos convexos aplicable cuando existen relaciones de independencia entre las variables del problema. Se considerará la particularización de la axiomática general de Shafer y Shenoy al caso de conjuntos convexos, mostrando las diferencias del algoritmo en el caso de convexos respecto al de las probabilidades.

2.2 Axiomática para la propagación de incertidumbre en estructuras gráficas

A continuación estudiamos una axiomática propuesta por Cano, Delgado y Moral [19, 20] que incluye la axiomática de Shafer y Shenoy [118] para los sistemas basados en valuaciones.

Cualquier modelo de representación de la incertidumbre en el que se verifique este sistema de axiomas permitirá el desarrollo de algoritmos de propagación en estructuras gráficas. Por ejemplo, en el caso de la Teoría de la Probabilidad podremos usar las independencias entre las variables de nuestro problema para factorizar la distribución de probabilidad conjunta como el producto de una serie de distribuciones condicionales $\prod P(X_i|pa(X_i))$. En este caso se cumplen los axiomas que veremos a continuación, y por tanto pueden construirse algoritmos [79, 90, 106, 115] que permiten calcular la distribución 'a posteriori' de una variable X_i dada una evidencia E, $P(X_i|E)$, sin necesidad de calcular previamente la distribución de probabilidad conjunta $\prod P(X_i|pa(X_i))$. Con tales algoritmos de propagación se consigue calcular $P(X_i|E)$ realizando cálculos sobre conjuntos más pequeños de variables en lugar de sobre todas ellas.

Sea $X = (X_1, \ldots, X_n)$ una variable n-dimensional tal que cada X_i toma valores en el conjunto finito U_i. En esta axiomática se hace uso del concepto de *valuación*. Una valuación es un concepto primitivo que puede ser considerado como la representación matemática de una información. Una valuación se puede particularizar a una distribución de posibilidad, a una distribución de probabilidad, a funciones de creencia, etc. Los algoritmos de propagación se expresarán en términos de operaciones con valuaciones. Para cada $I \subseteq \{1, \ldots, n\}$ existe un conjunto $\mathcal{V}_I$ de valuaciones definidas en el producto cartesiano:

$$\prod_{i \in I} U_i$$

$\mathcal{V}$ es el conjunto de valuaciones $\mathcal{V} = \cup_{I \subseteq \{1,\ldots,n\}} \mathcal{V}_I$. A lo largo de esta memoria usaremos $s(V)$ para denotar el conjunto de índices de las variables para las que está definida la valuación V.

Ejemplo 2.1 Valuaciones en la teoría de la probabilidad

En la teoría de la probabilidad una valuación sobre una variable X_I es una función no negativa

$$h : U_I \to I\!R_0^+$$

que no se le exige que esté normalizada.

Por ejemplo, si tres variables $\{X_1, X_2, X_3\}$ que toman valores en U_1, U_2, U_3, donde $U_i = \{u_i^1, u_i^2\}$, entonces una valuación puede ser una distribución de probabilidad sobre X_1:

$$p(u_1^1) = 0.8$$
$$p(u_1^2) = 0.2$$

También puede ser una distribución de probabilidad condicional de X_3 dada X_2:

$$p(u_3^1|u_2^1) = 0.9 \quad p(u_3^2|u_2^1) = 0.1$$
$$p(u_3^1|u_2^2) = 0.6 \quad p(u_3^2|u_2^2) = 0.4$$

Una valuación también puede ser una observación para una variable X_i. En este caso la función asigna el valor 1 para el caso u_i observado y 0 para el resto de los casos de X_i

Dos valuaciones se considerarán equivalentes cuando una es el producto de la otra por una constante. O sea, dos valuaciones h_1 y h_2 definidas sobre U_I son equivalentes cuando existe una constante $\alpha > 0$ tal que

$$\forall u \in U_I, h_1(u) = \alpha \cdot h_2(u)$$

■

Sobre las valuaciones son necesarias dos operaciones básicas: (ver Zadeh [142] y los trabajos de Shafer y Shenoy [120])

- *Marginalización* Si $J \subseteq I$ y $V_1 \in \mathcal{V}_I$ entonces la marginalización de V_1 sobre J es una valuación $V_1^{\downarrow J}$ definida en $\mathcal{V}_J$
- *Combinación* Si $V_1 \in \mathcal{V}_I$ y $V_2 \in \mathcal{V}_J$, entonces su combinación es una valuación, $V_1 \otimes V_2$, definida en $\mathcal{V}_{I \cup J}$.

Ejemplo 2.2 Combinación y marginalización de valuaciones probabilísticas
Dadas dos valuaciones probabilísticas, h_1 y h_2, definidas en U_I y U_J respectivamente, entonces la combinación de h_1 y h_2 es una valuación $h_1 \otimes h_2$ definida en $U_{I \cup J}$ por medio de la multiplicación punto a punto:

$$h_1 \otimes h_2(u) = h_1(u^{\downarrow I}).h_2(u^{\downarrow J}) \tag{2.1}$$

donde $u^{\downarrow I}$ es el elemento obtenido a partir de u sin tener en cuenta las coordenadas que no estén en I.

Si h es una valuación definida en U_I y $J \subseteq I$, entonces la marginalización de h en J, $h^{\downarrow J}$ se calcula mediante suma

$$h^{\downarrow J}(u) = \sum_{v^{\downarrow J} = u} h(v) \tag{2.2}$$

■

En [120] Shenoy y Shafer consideran que las valuaciones verifican los axiomas 1-3. Cano, Delgado y Moral [19] añadieron además los axiomas 4-6.

1. $V_1 \otimes V_2 = V_2 \otimes V_1, \quad (V_1 \otimes V_2) \otimes V_3 = V_1 \otimes (V_2 \otimes V_3)$.
   Este axioma nos da la conmutatividad y la asociatividad de la combinación.

2. Si $I \subseteq J \subseteq K$, y $V \in \mathcal{V}_J$ entonces $(V^{\downarrow J})^{\downarrow I} = V^{\downarrow I}$.
   Este axioma nos dice que no importa cual sea el orden de borrado de las variables en la marginalización de varias variables, ya que el resultado final será el mismo.

3. Si $V_1 \in \mathcal{V}_I$, $V_2 \in \mathcal{V}_J$, entonces $(V_1 \otimes V_2)^{\downarrow I} = V_1 \otimes V_2{}^{\downarrow (J \cap I)}$.
   Este axioma nos da la propiedad distributiva de la marginalización sobre la combinación. Este axioma es particularmente importante para el desarrollo de los algoritmos de propagación, ya que nos permite calcular $(V_1 \otimes V_2)^{\downarrow I}$ sin necesidad de hacer el cálculo explícito de $(V_1 \otimes V_2)$, el cual en la práctica puede llegar a ser muy costoso computacionalmente debido al gran número de variables que puede tener esta valuación. Este cálculo se puede hacer calculando $V_2{}^{\downarrow (J \cap I)}$ y combinando el resultado con V_1. De esta forma sólo necesitamos manejar valuaciones sobre U_J, $U_{I \cap J}$ y U_I, que es más eficiente computacionalmente.

4. *Elemento neutro.-* Existe una y sólo una valuación V_0 definida en $U_1 \times \ldots \times U_n$ tal que $\forall V \in \mathcal{V}^I, \forall J \subseteq I$, tenemos que $V_0{}^{\downarrow J} \otimes V = V$.

5. *Contradicción.-* Existe una y sólo una valuación, V_c, definida en $U_1 \times \ldots \times U_n$, tal que $\forall V$, $V_c \otimes V = V_c$.

6. $\forall V \in \mathcal{V}_\emptyset$, si $V \neq V_c{}^{\downarrow \emptyset}$, entonces $V = V_0{}^{\downarrow \emptyset}$.
   Este axioma nos dice que en el conjunto vacío de variables, $U_\emptyset$, todas las valuaciones son la contradicción o el elemento neutro.

Los tres primeros axiomas son las condiciones necesarias para deducir los algoritmos de propagación.

En el cuarto axioma V_0 representa la valuación neutra y en el quinto axioma V_c representa la información contradictoria. V_0 se usa para definir información condicional, y V_c para definir

valuaciones absorbentes. $(V_0)^{\downarrow I}$ y $(V_c)^{\downarrow I}$ se dice que son la valuación neutra y contradictoria en $\mathcal{V}_I$, respectivamente. Cuando no hay problema de confusión se denotan como V_0 y V_c, simplemente.

Ejemplo 2.3 Valuación neutra y contradicción en la teoría de la probabilidad
La valuación neutra es una función constante y distinta de 0, y la valuación contradicción es la función constante igual a 0. ■

Definición 2.1 (Valuación absorbente)

Una valuación $V \in \mathcal{V}_I$ es una valuación absorbente si y sólo si es diferente de la valuación contradicción y $\forall V' \in \mathcal{V}_I$, $(V \otimes V' = V)$ o bien $(V \otimes V' = V_c)$.

Si una valuación V representa una información sobre los valores de las variables $(X_i)_{i\in I}$, entonces una valuación absorbente representa un conocimiento perfecto sobre estos valores: no puede refinarse de forma consistente con la combinación con otra valuación. Podemos obtener únicamente la misma información o la contradicción.

Definición 2.2 *Dada $V \in V_{I\cup J}$, se dice que V es una valuación sobre U_I condicionada a U_J, si y sólo si se cumple que $V^{\downarrow J} = V_0 \in \mathcal{V}_J$, el elemento neutro sobre $\mathcal{V}_J$. El subconjunto de $\mathcal{V}_{I\cup J}$ que son valuaciones sobre U_I condicionadas a U_J se representará como $\mathcal{V}_{I|J}$.*

Intuitivamente, si V es una valuación sobre U_I condicionada a U_J, entonces V nos informa sobre las variables de X_I y su relación con las variables X_J, pero no debe informar sobre las variables X_J. La condición anterior expresa el hecho de que al marginalizar sobre U_J se obtiene el elemento neutro, lo cual recoge la idea de que esa valuación no nos da ninguna información sobre las variables X_J.

Ejemplo 2.4 Valuación condicionada en la teoría de la probabilidad
Supongamos dos variables, X_1 y X_2, que toman valores en U_1 y U_2, donde $U_i = \{u_i^1, u_i^2\}$. Una valuación condicionada probabilística sobre U_2 dada U_1, es una función

$$p : U_1 \times U_2 \to [0, 1]$$

tal que, marginalizada sobre U_1, obtenemos la valuación neutra (una valuación constante). ■

2.2.1 Modelos de dependencias

A veces, cuando tenemos un problema en el que intervienen varias variables, se presentan relaciones de independencia entre las variables. Estas relaciones de independencia pueden aprovecharse para conseguir algoritmos de propagación más eficientes que si no se consideraran dichas independencias. Además, estas independencias permiten expresar el conocimiento inicial sobre el problema de forma más resumida que si no tuviésemos independencias.

En esta sección estudiamos un modelo propuesto por Geiger y Pearl [54] para caracterizar las independencias entre variables y la independencia condicional.

Definición 2.3 Estructura de dependencias

Dada una familia de variables $(X_1, ..., X_n)$ *una estructura de dependencias sobre ellas es una función* $D : \wp(\{1, ..., n\}) \times \wp(\{1, ..., n\}) \times \wp(\{1, ..., n\}) \longrightarrow \{0, 1\}$ *donde si D(I,J,K) = 0 se dice que* X_I *es independiente de* X_K *dado* X_J*, verificando los siguientes axiomas*

- Simetría.-
  Si $D(I, J, K) = 0$ *entonces* $D(K, J, I) = 0$ *y viceversa.*
- Descomposición.-
  Si $D(I, J, K \cup L) = 0$ *entonces* $D(I, J, K) = 0$ *y* $D(I, J, L) = 0$
- Unión débil.-
  Si $D(I, J, K \cup L) = 0$ *entonces* $D(I, J \cup L, K) = 0$
- Contracción.-
  Si $D(I, J, K) = 0$ *y* $D(I, J \cup K, L) = 0$ *entonces* $D(I, J, K \cup L) = 0$

Puede encontrarse una interpretación intuitiva de estos axiomas en Pearl [106].

Definición 2.4 Sistema de información

Sea $(X_1, \ldots, X_n)$ *una variable n-dimensional que toma sus valores en* $U_1 \times \cdots \times U_n$*, y sea D una estructura de dependencias asociada con esta variable. Un sistema de información sobre esta variable con respecto a D es una familia de valuaciones* $\mathcal{H} \subseteq \mathcal{V}$ *y una función* $h : \mathcal{H} \to \wp((\{1, \ldots, n\}) \times \wp(\{1, \ldots, n\})$ *con las siguientes propiedades:*

1. $\forall V \in \mathcal{H}$*, si* $h(V) = (I, J)$ *entonces* $V \in \mathcal{V}_{I \cup J}$ *y* $I \cap J = \emptyset$*. Se dice que V es una valuación válida sobre las variables* $(X_i)_{i \in I}$ *condicionada a las variables* $(X_j)_{j \in J}$*. Si* $J = \emptyset$*, se dice simplemente que V es una valuación válida sobre* $(X_i)_{i \in I}$*.*

2. *Si* $V_1, V_2 \in \mathcal{H}$ *y* $h(V_1) = (J, \emptyset)$ *y* $h(V_2) = (I, J)$ *entonces* $V_1 \otimes V_2 \in \mathcal{H}$ *con* $h(V_1 \otimes V_2) = (I \cup J, \emptyset)$.

3. *Si* $V \in \mathcal{H}$ *con* $h(V) = (I, J)$ *y* $D(I, J, K) = 0$ *entonces* $V \otimes (V_0)^{\downarrow K} \in \mathcal{H}$ *con* $h(V \otimes (V_0)^{\downarrow K}) = (I, J \cup K)$.

4. *Si* $V \in \mathcal{H}$ *con* $h(V) = (I, J)$, *entonces si* $K \subseteq I$, $V^{\downarrow (K \cup J)} \in \mathcal{H}$ *con* $h(V^{\downarrow K}) = (K, J)$.

Definición 2.5 Sistema de información completo y determinista
Un sistema de información $(\mathcal{H}, h)$ *es completo y determinista si y sólo si existe una y sólo una valuación* $V \in \mathcal{H}$ *tal que* $h(V) = (\{1, \ldots, n\}, \emptyset)$.

Las estructuras gráficas son muy apropiadas para representar relaciones de independencia entre variables. Nosotros trabajaremos con grafos acíclicos dirigidos. Un grafo acíclico dirigido (de forma abreviada DAG) está asociado con una variable n-dimensional $(X_1, \ldots, X_n)$ si el conjunto de vértices es $\{X_1, \ldots, X_n\}$ (ver figura 2.1). O sea, tenemos un vértice para cada variable. Estos grafos pueden usarse para representar una estructura de dependencias para las variables $\{X_1, \ldots, X_n\}$, en la forma introducida por Pearl [106].

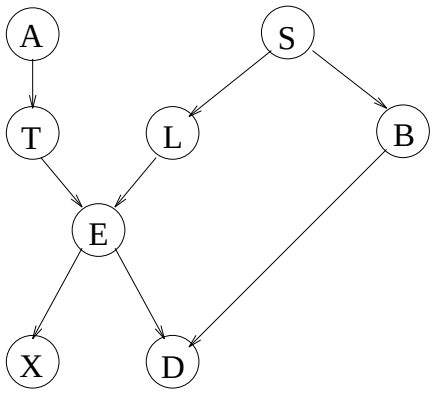


Figura 2.1: Un grafo acíclico dirigido para la red Asia

Definición 2.6 Estructura de dependencias asociada a un DAG
Dada una variable n-dimensional $(X_1, \ldots, X_n)$ *y un grafo dirigido acíclico* (T, E) *donde* $T = \{X_i\}_{i \in \{1, \ldots, n\}}$, *llamaremos estructura de dependencias asociada con* (T, E) *a aquella que verifica:*

$$D(I, J, K) = 0 \text{ si y sólo si } \{X_i\}_{i \in I} \text{ y } \{X_k\}_{k \in K} \text{ están D-separadas por } \{X_j\}_{j \in J}$$

o sea, [106], para todo camino no dirigido entre entre $\{X_i\}_{i\in I}$ *y* $\{X_k\}_{k\in K}$ *es cierta una de las siguientes condiciones:*

- *Existe un nodo cabeza a cabeza* $(\rightarrow X \leftarrow)$ *en el camino que no pertenece a* $\{X_j\}_{j\in J}$ *y que no tiene tampoco ningún descendiente que pertenezca a* $\{X_j\}_{j\in J}$
- *Existe un nodo que no es cabeza a cabeza en el camino que pertenece a* $\{X_j\}_{j\in J}$.

Por ejemplo, en el grafo de la figura 2.1, A y E son dependientes, pero se vuelven independientes cuando conocemos el valor de T. También X y D son dependientes pero se vuelven independientes cuando conocemos el valor de E. Por último T y L son independientes, aunque se vuelven dependientes al conocer el valor de E.

Un grafo acíclico de dependencias puede ser la base para definir un sistema de información.

Definición 2.7 Sistema de información correspondiente a un DAG
Si $(X_1,\ldots,X_n)$ *es una variable n-dimensional y* (T,E) *es un grafo* $T=\{X_i\}_{i\in\{1,\ldots,n\}}$, *decimos que* $(\mathcal{H},h)$ *es un sistema de información definido sobre* (T,E) *si y sólo si* $\mathcal{H}$ *es el mínimo sistema de información generado por la familia* $\{V_i\}_{i\in\{1,\ldots,n\}}$ *donde* $h(V_i)=(i,P(i))$ *y* $P(i)=\{j : (X_j,X_i)\in E\}$, *o sea el conjunto de padres de* X_i, *y donde la estructura de dependencias es la asociada con* (T,E).

Los sistemas de información definidos sobre grafos acíclicos dirigidos son muy apropiados porque son siempre completos y determinísticos. Introduciendo una valuación condicionada para cada variable dados sus padres, obtendremos una información global única para todas las variables como se puede ver en el siguiente teorema de Cano, Delgado y Moral [19].

Teorema 2.1 *Si* $(\mathcal{H},h)$ *es un sistema de información definido sobre el grafo* (T,E) *con* $T=\{X_i\}_{i\in\{1,\ldots,n\}}$ *entonces el sistema de información es completo y determinista sobre* $(X_1,\ldots,X_n)$.

Para completar esta descripción abstracta de información y de información condicional tenemos que dar dos definiciones adicionales.

Definición 2.8 Familia de observaciones
Una familia de observaciones sobre una variable n-dimensional $(X_1,\ldots,X_n)$ *es un conjunto de valuaciones* $\{O_i\}_{i\in I}$, *donde* $I\subseteq\{1,\ldots,n\}$, *y* O_i *es una valuación absorbente sobre* U_i, $\forall i\in I$.

Definición 2.9 Información 'a posteriori'
Si $(\mathcal{H}, h)$ es un sistema de información completo y determinista sobre $(X_1, \ldots, X_n)$ y $\{O_i\}_{i \in I}$ es una familia de observaciones sobre estas variables, llamaremos información 'a posteriori' a la familia de valuaciones $((\otimes_{i \in I} O_i) \otimes V)^{\downarrow J}$, donde $J \subseteq \{1, \ldots, n\}$ y V es la única información global sobre $(X_1, \ldots, X_n)$.

2.3 Algoritmos de Propagación

En general, el problema que abordan los conocidos como algoritmos de propagación es el siguiente: tenemos un conjunto finito de valuaciones $R = \{V_1, \ldots, V_m\}$, donde cada V_i se define en un referencial I_i. Estamos interesados en calcular la proyección o marginalización en una variable de interés X_j de la combinación de todas las valuaciones en R. Es decir en calcular [118]:

$$PS_j = \left(\bigotimes R\right)^{\downarrow \{j\}} = (V_1 \otimes \ldots \otimes V_m)^{\downarrow \{j\}} \tag{2.3}$$

para un valor $j \in \{1, \ldots, n\}$.

El conjunto R representa toda la información de que disponemos. Por ejemplo, en el caso de las valuaciones que son subconjuntos, en R estarán todas las restricciones que conozcamos sobre los valores de las variables del problema.

Para desarrollar los algoritmos de propagación en la arquitectura de Shafer y Shenoy [120], sólo hace falta que se verifiquen los axiomas 1-3 de la sección 2.2 para las operaciones con valuaciones, y esos se supondrán verificados a lo largo de esta sección.

Ejemplo 2.5 Algoritmos de propagación de probabilidades
En el contexto de las probabilidades, en R suele haber dos tipos de informaciones: informaciones genéricas que determinan una distribución de probabilidad global y observaciones sobre algunas de estas variables para un caso particular. Normalmente es muy difícil poder especificar la distribución global de forma directa. Si partimos de un grafo dirigido acíclico (ver figura 2.1 que expresa las independencias del problema mediante el criterio de D-separación [132, 106], entonces una distribución global $P(X_1, \ldots, X_n)$ puede obtenerse combinando todas las distribuciones de probabilidad $P(X_i|pa(X_i))$ de cada nodo X_i condicionada a los valores de sus padres $pa(X_i)$ en el grafo. Esta distribución de probabilidad global se calcula entonces con la expresión:

$$P(X_1, \ldots, X_n) = \Pi_{i=1}^{n} P(X_i | pa(X_i)) \tag{2.4}$$

El objetivo de los algoritmos de propagación de probabilidades es el cálculo de una distribución de probabilidad para una variable X_j dada una familia de observaciones $E = \{O_k\}_{k \in K}$, sobre un conjunto de variables K. Esta distribución de probabilidad 'a posteriori' para la variable X_j es proporcional a $P(X_1, \ldots, X_n) \otimes (\bigotimes_{k \in K} O_k))^{\downarrow \{j\}}$. O sea,

$$P(X_j = u_j \mid E) \propto P(X_1, \ldots, X_n) \otimes (\bigotimes_{k \in K} O_k))^{\downarrow \{j\}}(u_j) \tag{2.5}$$

De hecho tenemos que

$$P((X_j = u_j) \cap E) = P(X_1, \ldots, X_n) \otimes (\bigotimes_{k \in K} O_k))^{\downarrow \{j\}}(u_j) \tag{2.6}$$

Los grafos de dependencias en el caso probabilístico se han hecho populares bajo distintos nombres entre los que cabe destacar diagramas de influencia*,* redes de creencia*,* redes bayesianas *y* redes causales *[106, 90, 100]. En esta sección usaremos* redes bayesianas *para referirnos a tales grafos de dependencias.*

La red bayesiana permite un gran ahorro en el espacio necesario para almacenar la distribución de probabilidad conjunta al no tener que almacenar directamente esta distribución, sino las distribuciones condicionales que suelen involucrar pocas variables y que por tanto ocuparán poco espacio. Además el uso de redes bayesianas en la propagación permite obtener distribuciones de probabilidad 'a posteriori' de una variable de la red conocidas algunas de las demás variables, con un número de operaciones (sumas y multiplicaciones) mucho menor que si se hiciera calculando previamente la distribución de probabilidad conjunta. El problema que resuelven los algoritmos de propagación en redes bayesianas es el cálculo de $P(X_i|E)$ para una o varias variable objetivo X_i, donde E es el conjunto de observaciones (o conjunto de evidencia). $P(X_i|E)$ podría obtenerse a partir de la distribución conjunta como se hace en la expresión 2.5, pero suponemos que esto es difícil de llevar a cabo tanto por razones de espacio como de coste computacional.

Por ello se pueden encontrar en la literatura varios métodos alternativos [79, 106, 90, 120, 91] al que propone la expresión 2.5, que aprovechan las independencias reflejadas por la red bayesiana para realizar los cálculos de forma local, sin tener que calcular la distribución de probabilidad conjunta.

También se pueden resolver mediante algoritmos de propagación otros problemas, como el cálculo de la configuración de máxima probabilidad, es decir un valor u_i^* para cada variable X_i tal que,

$$P(X_1 = u_1^*, \ldots, X_n = u_n^*|E) = \max_{u_1,\ldots,u_n} P(X_1 = u_1, \ldots, X_n = u_n|E)$$

También se pueden resolver problemas de decisión complejos, en los que hay que seleccionar la secuencia de decisiones que maximizan la utilidad esperada. ■

El cálculo se realiza mediante los llamados algoritmos de propagación [20, 90, 106, 118, 120] que permiten obtener la valuación PS_j de la expresión 2.3 utilizando cálculos locales, los cuales van a involucrar operaciones con valuaciones de menos variables que por tanto serán más eficientes.

2.3.1 Algoritmo de eliminación de variables

Un primer algoritmo bastante simple que puede aplicarse es el conocido como de *eliminación de variables* que fue propuesto de forma independiente por Li y D'Ambrosio [91], Zhang y Poole [143] y por Dechter [38], para calcular distribuciones de probabilidad 'a posteriori' de una variable dado un conjunto de evidencias, $P(X_i|E)$. Esencialmente este algoritmo trabaja transformando el conjunto $R = \{V_1, \ldots, V_m\}$ de acuerdo con el siguiente paso básico (*Borrado de k*):

- Sea k un índice, $k \neq j$ donde j es el índice de la variable X_j de interés. Consideremos $K = \{V_i \in R : k \in s(V_i)\}$ y $L = s(\bigotimes K) - \{k\}$. Entonces R se transforma en el conjunto

$$(R - K) \cup \{(\otimes K)^{\downarrow L}\} = (R - K) \cup \{(\otimes K)^{\downarrow s(\bigotimes K) - \{k\}}\} \tag{2.7}$$

Este paso se repite (borrando todos los índices k distintos de j) hasta que todas las valuaciones estén definidas en el referencial $\{j\}$. La valuación que buscamos, PS_j, es la combinación de todas las valuaciones que quedan en R.

Este procedimiento es, en general, más eficiente que combinar todas las valuaciones y marginalizar después. Pensemos que el tamaño de una representación es en la mayoría de los casos, al menos, proporcional al tamaño del referencial U_I, y este tamaño es el producto de los elementos de cada conjunto U_i. Este es el caso, por ejemplo, de una distribución

de probabilidad. Si combinamos todas las valuaciones en R obtenemos una valuación en $U_1 \times \ldots \times U_n$ lo que, para valores moderados de n, ya no se puede representar en la memoria de un ordenador. Sin embargo, en el algoritmo anterior las valuaciones están definidas en referenciales más pequeños (involucran menos variables): $s(\bigotimes K)$.

Veamos en el siguiente ejemplo como podemos particularizar este algoritmo para el caso probabilístico:

Ejemplo 2.6 Algoritmo de borrado para el caso probabilístico

Este algoritmo funciona muy bien cuando estamos interesados en calcular la probabilidad 'a posteriori' sólo para una variable o conjunto de variables del problema. Sea $V = \{h_1, \ldots, h_n\}$ *el conjunto de valuaciones correspondientes cada una de ellas a las distribuciones de probabilidad condicional* $\{P(X_1|pa(X_1)), \ldots, P(X_n|pa(X_n)\}$. *Sea* E *el conjunto de variables observadas y sea* X_j *la variable de interés (el algoritmo puede generalizarse fácilmente si queremos calcular la probabilidad 'a posteriori' de un conjunto de variables* X *en lugar de sólo la de* X_j*).*

Supongamos que tenemos el grafo de dependencias de la figura 2.2, el cual representa una red causal ya que cada nodo almacena una distribución de probabilidad condicional o bien una distribución 'a priori'. Supongamos que cada variable toma dos posibles valores, que no tenemos ninguna variable observada, y que estamos interesados en calcular $P(D, E)$. *Podríamos calcular* $P(D, E)$ *con la siguiente expresión:*

$$P(D,E) = \{P(A)P(B|A)P(C|A)P(D|B,C)P(E|C)\}^{\downarrow\{D,E\}}$$

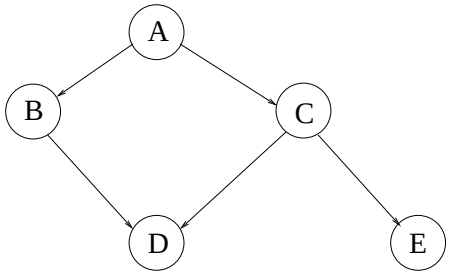


Figura 2.2: Una red causal

Este cálculo conlleva calcular la distribución de probabilidad conjunta expresada por la red causal. Sin embargo $P(D, E)$ *podemos calcularlo también de la siguiente forma:*

$$P(D,E) = [\sum_a [\sum_b [\sum_c [P(E|C)P(D|B,C)]P(C|A)]P(B|A)]P(A)]$$

En este caso no ha hecho falta calcular la distribución de probabilidad conjunta para calcular $P(D,E)$. *El cálculo anterior necesita 72 multiplicaciones. De la expresión anterior deducimos que primero se elimina la variable* C *de todas las valuaciones, combinando previamente todas estas valuaciones donde aparece en una sola. Después se elimina la variable* B *de la misma forma, y finalmente la variable* A.

Otro posible orden de eliminación de la variables nos lo daría la siguiente expresión:

$$P(D,E) = [\sum_c [P(E|C)[\sum_b P(D|B,C)[\sum_a P(C|A)[P(B|A)P(A)]]]]]$$

En este caso sólo se necesitan 28 multiplicaciones. Ahora se elimina primero la variable A, *después la* B *y finalmente la* C.

Podemos entonces deducir que diferentes órdenes de borrado pueden resultar en significativos costos computacionales.

Sea ρ *un orden de eliminación de las variables que no están en* $X_j \cup E$. *Entonces el algoritmo es como sigue:*

Algoritmo *2.1* Propagación de probabilidades por eliminación de variables

1. Para cada variable $X_k \in E$ actualizar en todos los potenciales al valor observado poniendo 0 en los valores no observados.
2. Mientras ρ no esté vacío:
   (a) Borra la primera variable X_i de ρ
   (b) Llamar a $suma(V, X_i)$
3. Sea $h = \bigotimes_{h_i \in V} h_i$
4. Sea $fn = \sum_{u_j \in U_j} h(u_j)$
5. Devolver $P(X_j = u_j|E) = h(u_j)/fn$

■

El procedimiento $suma(V, X_i)$ *usado en el algoritmo anterior, se describe a continuación:*

Algoritmo *2.2* Eliminación de una variable

1. Sean las valuaciones $h_1, \ldots, h_k$ que contienen a X_i y definidas sobre $U_{I_1}, \ldots, U_{I_k}$. Sea $I = \cup_{j=1}^{k} I_j$
2. Borrar de V todas las valuaciones $h_1, \ldots, h_k$ que contienen a X_i
3. Añadir la nueva valuación calculada como $(\bigotimes_{j=1}^{k} h_j)^{\downarrow\{I-\{i\}\}}$ al conjunto de valuaciones V

■

■

El algoritmo de eliminación de variables puede ser mejorado si tenemos en cuenta que a veces no será necesario introducir todas las valuaciones originales en el conjunto V, ya que algunas de las variables del modelo pueden ser irrelevantes para el cálculo de la valuación 'a posteriori' de la variable objetivo. Se pueden evitar operaciones innecesarias hallando el conjunto de variables irrelevantes para el cálculo de esta valuación 'a posteriori', eliminándolas del grafo, permitiendo obtener un modelo equivalente más sencillo, que sólo contenga los nodos relevantes. Los nodos irrelevantes pueden ser obtenidos utilizando la estructura de independencia contenida en el grafo de dependencias, como se muestra en Shachter [116] y Geiger,Verma y Pearl [55, 55]. Estos autores proponen un algoritmo simple para hallar el conjunto de nodos relevantes para el cálculo de la distribución 'a posteriori' $P(X_i|E)$. En este algoritmo se añade un nodo auxiliar Θ_j por cada nodo X_j del grafo de dependencias. Cada nodo Θ_j se hace padre del correspondiente nodo X_j. El conjunto de nodos relevantes puede ser obtenido utilizando el criterio de *D-separación* en el grafo resultante mediante el algoritmo siguiente:

Algoritmo 2.3 Identificación de los nodos relevantes

1. Datos de entrada y de salida
   - **Datos:** Un grafo de dependencias (D, P) y una variable objetivo: variable objetivo X_j y el conjunto de evidencia E.
   - **Resultados:** El conjunto de nodos relevantes R necesarios para calcular $P(X_j|E)$

2. Construir un nuevo grafo dirigido, D', añadiendo un nodo auxiliar, Θ_i, y una arista $\Theta_i \rightarrow X_i$ a D para cada nodo X_i.

3. Identificar el conjunto Θ de nodos auxiliar que no están D-separados de X_j por E en D'.

4. Asignar a R los nodos X_i cuyos nodos auxiliares Θ_i, están contenidos en Θ

■

La etapa 3 del anterior algoritmo puede desarrollarse en tiempo polinomial en el número de variables con un algoritmo propuesto por Geiger, Verma y Pearl [55, 56]. Considerando el grafo reducido resultante de eliminar los nodos irrelevantes obtenidos mediante este algoritmo, se puede simplificar de forma importante el número de cálculos necesarios para calcular la probabilidad 'a posteriori' $P(X_j|E)$.

Ejemplo 2.7 Obtención de nodos relevantes
Supongamos el grafo de dependencias de la figura 2.2. Veamos que valuaciones influyen en el cálculo de alguna valuación 'a posteriori'.

- *Para el cálculo del conjunto convexo 'a posteriori' en la variable E si no disponemos de ninguna evidencia, sólo son relevantes las valuaciones correspondientes a los nodos A, C y E.*

- *Si sólo variable E está observada entonces para el cálculo de la valuación 'a posteriori' de la variable B es necesario utilizar las valuaciones correspondientes a los nodos A, B, C y E.*

- *Si sólo la variable B está observada entonces para el cálculo de la valuación 'a posteriori' de la variable A sólo es necesario utilizar las valuaciones correspondientes a los nodos A y B.*

Vemos por tanto que dependiendo de la situación el uso de esta simplificación puede ahorrar muchos cálculos en el algoritmo de eliminación de variables. ■

2.3.2 Algoritmos de propagación en árboles de grupos

A veces, por ejemplo cuando se quiere calcular la información marginal PS_j para varias variables X_j, es conveniente organizar los cálculos en lo que se llama un *árbol de grupos*. Este árbol de grupos, T_G, es un árbol no dirigido en el que los nodos son grupos de variables, G, y en los que se cumplen las dos condiciones siguientes [90]:

1. Para toda valuación $V \in R$, existe un nodo G del árbol tal que $s(V) \subseteq G$.

2. Si G_1 y G_2 son dos nodos, entonces para todo nodo G en el camino que une G_1 y G_2 se tiene que $(G_1 \cap G_2) \subseteq G$.

2.3.2.1 Obtención de un árbol de grupos

Veamos como podemos llevar a cabo el proceso de obtención de un árbol de grupos a partir de un conjunto de valuaciones $R = \{V_1, \ldots, V_n\}$. Partimos de un grafo no dirigido que tiene un nodo por cada variable $X_1, \ldots, X_n$ y tal que X_i y X_j están unidos en este grafo no dirigido si y sólo si existe una valuación $V \in R$ para la que $\{i, j\} \subseteq s(V)$. Este grafo no dirigido coincide en el caso probabilístico, con el *grafo moral* de una red causal. El grafo moral se obtiene a partir del grafo dirigido de la red causal ignorando la dirección de los arcos de la red causal, y uniendo entre sí todos los padres de cada variable X_i. En la Figura 2.3 se puede ver un ejemplo de un grafo no dirigido, que corresponde al grafo moral asociado al grafo dirigido acíclico de la figura 2.1.

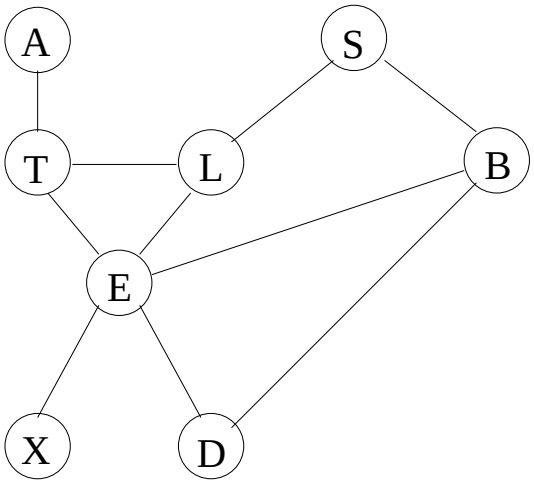


Figura 2.3: Grafo No-Dirigido

El primer paso para obtener el árbol de grupos consiste en obtener un grafo triangular [82, 14] a partir del grafo dirigido mencionado más arriba. Para el grafo no dirigido de la figura 2.3 podemos construir el grafo triangulado de la figura 2.4.

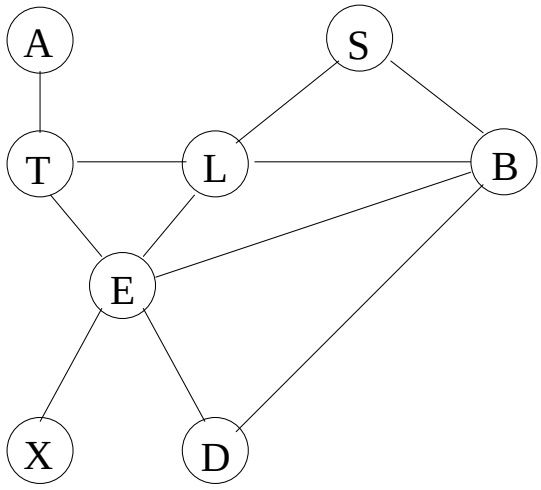


Figura 2.4: Grafo triangulado

2.3.2.2 Obtención de un grafo triangular

La eficiencia en los algoritmos de propagación en un árbol de grupos está estrechamente relacionada con el tamaño del árbol de grupos. Dicho tamaño puede definirse como la suma de los tamaños de cada grupo. Por ejemplo, en el caso probabilístico la eficiencia es una función polinómica del tamaño del árbol de grupos.

El punto clave para obtener un árbol de grupos de pequeño tamaño está en la triangulación del grafo no dirigido. Los algoritmos de propagación óptimos se obtendrán al considerar la triangulación óptima. Sin embargo, se conoce que la construcción de la triangulación óptima es un problema NP-completo [4]. No obstante, existen diversos algoritmos que permiten obtener buenas triangulaciones usando un tiempo razonable [81, 82, 104, 68]. En Cano y Moral [14] se presentaron técnicas heurísticas para la obtención de buenas triangulaciones de grafos no dirigidos, y se comparan con la mejor heurística presentada por Kjærulff en [81]. Describamos a continuación estas técnicas. Veamos en primer lugar la definición de grafo triangular.

Definición 2.10 (Grafo triangular) *Un grafo no dirigido se dice que está triangulado si y sólo si todo ciclo de longitud cuatro o más tiene un arco entre cada par de nodos no adyacentes.*

El grafo de la figura 2.5 no está triangulado.

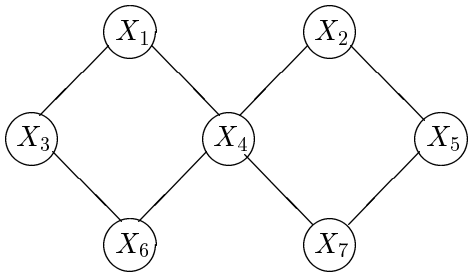


Figura 2.5: Un grafo no dirigido que no está triangulado

La forma de triangular un grafo no dirigido es añadiendo arcos. Pueden obtenerse diferentes triangulaciones con las diferentes permutaciones de los nodos del grafo, $\sigma : \{1, \ldots, n\} \longrightarrow \{1, \ldots, n\}$. Dada una permutación, σ, la triangulación asociada se construye con el siguiente algoritmo:

Algoritmo 2.4 Construcción de un grafo triangular
Entrada: Un grafo no dirigido $G = (\mathcal{U}, E)$.
Salida: El grafo triangular G_T y la lista $\mathcal{L}$ con los grupos maximales asociados.

1. $\mathcal{L} \leftarrow \emptyset$
2. Para i=1 hasta n
   - Añadir arcos entre cada par de nodos adyacentes a $X_{\sigma(i)}$. Sea L_i el conjunto de arcos añadidos.
   - Borrar el nodo $X_{\sigma(i)}$ y todos los arcos que conectan $X_{\sigma(i)}$ con otros nodos del grafo.
   - $G_i \leftarrow \{X_{\sigma(i)}\} \cup Ady(X_{\sigma(i)})$
   - Si $\nexists$ un $G_j \in \mathcal{L}$ tal que $G_i \subseteq G_j$
     entonces $\mathcal{L} \leftarrow \mathcal{L} \cup \{G_i\}$
3. Para obtener el grafo triangulado G_T a partir del grafo original G, añadir todos los arcos de los conjuntos L_i $(i = 1, \ldots, n)$ a G. O sea $G_T \leftarrow (\mathcal{U}, E \cup \{\cup_{i=1,\ldots,n} L_i\})$
4. La salida del algoritmo es G_T y $\mathcal{L}$.

■

La permutación σ se llama una *secuencia de borrado*. Considerando la secuencia de borrado $(\sigma(1), \sigma(2), \sigma(3), \sigma(4), \sigma(5), \sigma(6), \sigma(7)) = (3, 2, 7, 4, 1, 5, 6)$ en el grafo de la figura 2.5, obtenemos el grafo triangulado de la figura 2.6.

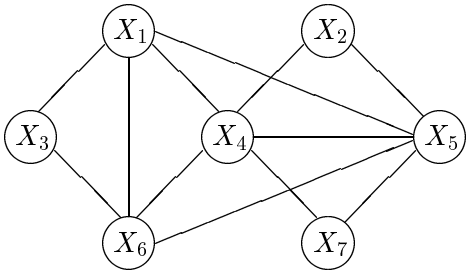


Figura 2.6: Grafo triangulado asociado bajo el orden de eliminación de variables σ

El grafo triangular nos sirve para obtener una descomposición del grafo en un conjunto de subgrupos maximales. Los subgrafos maximales completos del grafo triangulado se conocen en la literatura como *cliques*. A lo largo de esta memoria nosotros hemos preferido utilizar el término *grupo*. Por ejemplo, en el grafo de la figura 2.6 los grupos maximales son $\{X_1, X_3, X_6\}$, $\{X_1, X_4, X_5, X_6\}$, $\{X_2, X_4, X_5\}$, $\{X_4, X_5, X_7\}$.

El tamaño de una triangulación se obtiene entonces como suma del tamaño de los grupos maximales que la forman. El anterior algoritmo de *Construcción de un grafo triangular* es capaz de encontrar todas las triangulaciones minimales posibles que existan, así como las que no lo son, dependiendo de la secuencia de eliminación utilizada. Los grupos maximales obtenidos se ve claramente que dependen de la secuencia de eliminación considerada, ya que cada vez que se elimina un nodo se obtiene un conjunto completo que podrá o no ser un grupo maximal dependiendo de los ya obtenidos previamente. Si el conjunto completo obtenido es un subconjunto de otro conjunto completo previamente obtenido, entonces no es un grupo maximal, y no será entonces considerado a la hora de construir el árbol de grupos.

Cano y Moral [14] desarrollaron varias heurísticas de triangulación que intentaban obtener árboles de grupos lo más pequeños posible. Estas heurísticas se describen en la sección 2.6.

2.3.2.3 Construcción del árbol de grupos

Una vez que hemos construido el grafo triangular G_T y la lista de $\mathcal{L}$ de grupos maximales contenidos en él, hemos de seguir los siguientes pasos para construir el árbol de grupos:

1. Numerar los nodos del grafo no dirigido:

Para ello se emplea el *algoritmo de máxima cardinalidad* [126] que asigna un número de orden a cada nodo del grafo.

2. Numerar los grupos maximales:
   Cada grupo maximal de la lista $\mathcal{L}$ se numera teniendo en cuenta el nodo de menor orden que contenga. Los empates se deshacen teniendo en cuenta el siguiente nodo contenido en cada grupo.

3. Obtener los conjuntos residuales y separadores:
   Dada la ordenación anterior, cada grupo G_i se divide en dos conjuntos disjuntos llamados residual R_i y separador S_i. Estos conjuntos se calculan atendiendo a las siguientes reglas:

$$S_i = \begin{cases} G_i \cap (G_1 \cup \ldots \cup G_{i-1}) & si\ i \geq 2 \\ \emptyset & si\ i = 1 \end{cases} \tag{2.8}$$

$$R_i = \begin{cases} G_i \backslash S_i & si\ i \geq 2 \\ G_1 & si\ i = 1 \end{cases} \tag{2.9}$$

4. Establecer la estructura del árbol de grupos:
   Para dotar de estructura al conjunto de grupos maximales se toma la siguiente consideración:

   > Cualquier grupo G_j que contenga al separador S_i con $j < i$ será un posible padre del grupo G_i.

   Por tanto, el grupo G_1 será la raíz del árbol. Para aquellos grupos que tengan más de un posible padre seleccionaremos uno de ellos arbitrariamente. La representación gráfica del árbol de grupos maximales suele hacerse de dos formas distintas:

   - Mediante un árbol propiamente dicho, con la dirección de los arcos marcada por la elección de los padres para cada grupo.
   - Incluyendo los separadores en la representación como otro tipo de nodos del árbol y omitiendo la dirección de las aristas. En este caso la elección en un momento determinado de un nodo como raíz determina la direccionalidad de las aristas.

En la figura 2.7 mostramos un árbol de grupos asociado al grafo triangulado de la figura 2.4 y en la figura 2.8 mostramos otro árbol de grupos, en este caso para el grafo triangulado de la figura 2.6.

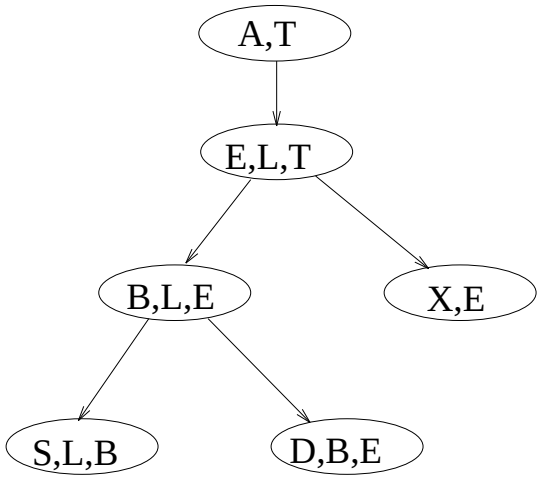


Figura 2.7: Un árbol de grupos para el grafo triangulado de la figura 2.4

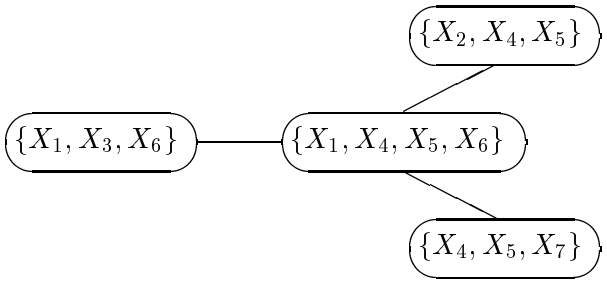


Figura 2.8: Un árbol de grupos para el grafo triangulado de la figura 2.6

2.3.2.4 Asignación de las valuaciones a los grupos del árbol

En general, todos los algoritmos comienzan asignando cada valuación $V \in R$ a un nodo del árbol de grupos G tal que $s(V) \subseteq G$. Sea $R(G)$ el conjunto de valuaciones asignadas a G. A continuación se calcula, para cada nodo G la valuación

$$V_G = \otimes_{V \in R(G)} V \tag{2.10}$$

2.3.2.5 Propagación en el árbol de grupos

Una vez que hemos construido el árbol de grupos y asignado cada una de las valuaciones a un grupo, los algoritmos trabajan entonces mandando mensajes entre los nodos adyacentes del árbol de grupos. Hay dos modelos esenciales: el de Shafer y Shenoy [118] y el conocido como HUGIN [76]. Ambos serán brevemente descritos a continuación. Más detalle puede encontrarse en los libros de Jensen [75] y Castillo, Hadi y Gutiérrez [25].

En general, puede decirse que el algoritmo HUGIN es el más eficiente (si la división tiene la misma complejidad que la combinación o la marginalización). Sin embargo, requiere unas condiciones de aplicación un poco más fuertes que el algoritmo de Shafer y Shenoy.

En nuestros desarrollos emplearemos fundamentalmente el algoritmo de Shafer y Shenoy. La razón es que es metodológicamente más sencillo y además que no existe una operación de división para los conjuntos convexos de probabilidades.

2.3.2.6 Arquitectura de Shafer y Shenoy

Se supone que existen dos mensajes para cada par de nodos adyacentes G_1 y G_2, uno de G_1 a G_2: V_{G_1,G_2} y otro de G_2 a G_1: V_{G_2,G_1}. La operación fundamental de mandar mensaje entre dos nodos G_1 y G_2 consiste en realizar el siguiente cálculo:

$$V_{G_1,G_2} = \left(V_{G_1} \otimes \left(\bigotimes_{G \in Ady(G_1,G_2)} V_{G,G_1} \right) \right)^{\downarrow G_1 \cap G_2} \tag{2.11}$$

donde $Ady(G_1, G_2)$ es el conjunto de todos los grupos adyacentes a G_1 excepto G_2. $Ady(G_1)$ es el conjunto de todos los nodos adyacentes a G_1.

El algoritmo de propagación asociado a esta arquitectura consiste en el cálculo de todos los mensajes lo que se hace mediante dos recorridos del árbol de grupos. Para ello se elige un nodo G como raíz o pivote y en una primera fase se mandan mensajes de las hojas al nodo raíz, y en la segunda se distribuye la información desde el nodo raíz a las hojas. De manera más concreta la primera parte puede representarse como:

Primera(G)
- Para todos los nodos $G' \in Ady(G)$
  - Pide(G', G)

donde $Pide(G', G)$ es como sigue:

Pide(G', G)
- Para todo nodo $G'' \in Ady(G', G)$
  - Pide(G'', G')
- Mandar mensaje de G' a G.

El segundo recorrido en el grafo distribuye la información a partir del nodo raíz G. El algoritmo se puede expresar de la siguiente forma.

Segunda(G)
- Para todo nodo $G' \in Ady(G)$
  - Distribuye(G', G)

El procedimiento *Distribuye* es como sigue,

Distribuye(G', G)
- Mandar un mensaje de G a G'
- Para cada $G'' \in Ady(G', G)$
  - Distribuye(G'', G')

Después de aplicar este algoritmo, para calcular PS_j sólo tenemos que elegir un grupo de variables G al que pertenezca X_j y calcular,

$$PS_j = \left(V_G \otimes \left(\bigotimes_{G' \in Ady(G)} V_{G',G} \right) \right)^{\downarrow j} \tag{2.12}$$

2.3.2.7 Arquitectura HUGIN

Para aplicar la arquitectura HUGIN es necesario que se pueda definir una operación de división /. La operación de división se podrá definir en un formalismo particular cuando V_1/V_2 exista para cada par de valuaciones V_1 y V_2 tal que $\sigma(V_2) \preceq \sigma(V_1)$, donde σ es una aplicación de V en $(G, \preceq)$, y $\preceq$ es una relación de orden parcial tal que $\sigma(V_1 \otimes V_2) = \inf\{\sigma(V_1), \sigma(V_2)\}$. Intuitivamente la aplicación σ representa el soporte de un conjunto.

Por ejemplo, en el caso probabilístico, si se puede definir la operación de división entre valuaciones, y por tanto se podrá aplicar esta arquitectura. En [77, 76] se puede ver detalladamente la aplicación de esta arquitectura al caso probabilístico.

Aquí se supone que existe un sólo mensaje para cada par de nodos adyacentes G_1 y G_2. Este mensaje se denota como $V^{G_1,G_2} = V^{G_2,G_1}$. Inicialmente este mensaje es igual a la valuación neutra en el referencial $G_1 \cap G_2$.

El mismo mensaje se usará en la primera y en la segunda fase. Otro aspecto que distingue al algoritmo de Shafer y Shenoy y al algoritmo HUGIN es que, en el primero la valuación V_G no cambia, mientras que en el segundo si va a cambiar. Pero lo más importante es que ahora los mensajes se mandan de diferente forma. El mensaje de G_1 a G_2 se manda de la siguiente forma:

*Manda*_H(G_1, G_2)

- Calcular $W = (V_{G_1})^{\downarrow G_1 \cap G_2}$
- Hacer $V_{G_2} = V_{G_2} \otimes (W / V^{G_1,G_2})$
- Hacer $V^{G_1,G_2} = V^{G_1,G_2} \otimes W$

El resto del algoritmo es exactamente igual con las mismas fases de petición y distribución. Finalmente, para calcular PS_j se determina un grupo cualquiera G conteniendo X_j y se determina:

$$PS_j = (V_G)^{\downarrow j} \tag{2.13}$$

2.4 Algoritmos de propagación para conocimiento general

En esta sección comenzamos la descripción de los algoritmos de propagación aplicados a conjuntos convexos de probabilidades. En primer lugar, consideramos el caso en que no hay independencias en el sentido probabilístico del término. Sólo se verifica la definición de independencia marginal.

Sea una variable de dimensión n, $(X_1, \ldots, X_n)$, donde cada X_i toma valores en un conjunto finito U_i. Supongamos que disponemos de m informaciones, en la que cada una es un conjunto convexo de probabilidades H_j sobre un subconjunto de las variables del problema, X_{I_j} ($I_j \subseteq 1, \ldots, n$), o sea H_j es un conjunto convexo de probabilidades en U_{I_j}. Supongamos además que cada conjunto convexo viene representado por un conjunto de restricciones lineales sobre las variables para las que viene definido. En los algoritmos que revisamos en esta sección el objetivo será calcular la información inducida sobre un conjunto de variables de interés, X_J, mediante la siguiente expresión:

$$(H_1 \cap H_2 \cap \ldots \cap H_m)^{\downarrow J} \tag{2.14}$$

Cada H_j se da a través de un conjunto de restricciones lineales, R_j, que puede representar límites sobre sucesos; o límites en probabilidades condicionales; o en el valor esperado de alguna función definida de U_{I_j} en $\mathbb{R}$.

El objetivo de los algoritmos que revisamos en esta sección será conocer cuales son las probabilidades posibles inducidas por $H_1, \ldots, H_m$ en el conjunto de variables X_J. Esto es más general que calcular los límites para la probabilidad de un suceso $p(a_J)$. Si tenemos calculado H_J entonces los límites para $p(a_J)$ se pueden calcular fácilmente por programación lineal con las restricciones lineales que definen H_J. Estas técnicas se pueden aplicar incluso cuando queremos calcular la probabilidad condicional $p(a_I|a_J)$, ya que $p(a_I|a_J)$ puede obtenerse a partir de $H^{\downarrow I\cup J}$ por programación lineal. Sin embargo, cuando el objetivo es calcular límites en sucesos dependientes de un conjunto de variables, o bien en sucesos condicionales de un conjunto de variables a otro, será más eficiente aplicar las técnicas de programación lineal al problema original $H_1 \cap \ldots \cap H_m$, en vez de hacer la marginalización al conjunto de variables de J y luego calcular los límites en el convexo marginalizado. Los algoritmos que repasaremos en esta sección deben aplicarse cuando nuestro objetivo es calcular las restricciones que definen $H^{\downarrow J}$.

El esquema general del algoritmo de propagación es similar al algoritmo de eliminación de variables [38, 91, 143]. Podría también aplicarse el algoritmo obtenido a partir del marco abstracto presentado por Lauritzen y Shenoy [89] para propagar incertidumbre en sistemas que pueden representarse por valuaciones, usando un *árbol de grupos*. En ambos algoritmos de propagación necesitamos definir las operaciones de combinación y marginalización de informaciones. En nuestro caso la combinación es la intersección y la marginalización es la dada por la definición 1.5. Ambos algoritmos de propagación pueden aplicarse en el problema que estudiamos en esta sección porque se cumplen los axiomas que hemos visto en la sección anterior. Los tres primeros axiomas particularizados tomando la intersección como combinación y la marginalización de la definición 1.5.

1. $H_1 \cap H_2 = H_2 \cap H_1, \quad H_1 \cap (H_2 \cap H_3) = (H_1 \cap H_2) \cap H_3.$
2. $\left(H^{\downarrow I}\right)^{\downarrow J} = H^{\downarrow J}.$
3. Si H_1 está definido en U_I y H_2 está definido en U_J, entonces $(H_1 \cap H_2)^{\downarrow I} = H_1 \cap H_2^{\downarrow I\cap J}$

El elemento neutro que postula el axioma 4 sería la ignorancia, I, definida sobre el referencial U, la cual corresponde al conjunto convexo de todas las distribuciones de probabilidad sobre

U. La contradicción en el axioma 5 sería el conjunto convexo vacío, es decir, aquel conjunto convexo cuyo conjunto de restricciones sea incompatible.

Verdegay-López [131] demuestra que los seis axiomas de Cano, Delgado y Moral se cumplen para el caso de conjuntos convexos de probabilidad representado como conjuntos de restricciones con las operaciones de combinación y marginalización mencionadas más arriba. La definición de independencia implícita en esta propagación es la definición de independencia marginal. Es fácil comprobar que se verifica el siguiente teorema.

Teorema 2.2 *La independencia marginal (definición 1.18) cumple los cuatro axiomas de una estructura de dependencias dados en la definición 2.3.*

Demostración:

- La simetría es inmediata a partir de la definición de independencia marginal.
- Para la descomposición, tenemos que probar que si X es independiente de (Y_1, Y_2) dado Z, entonces X es independiente de Y_1 dado Z. De acuerdo con la definición, si se da la primera independencia, tenemos que el conjunto global de probabilidades posibles para (X, Y_1, Y_2, Z) es

$$H = \{p : p^{\downarrow X,Z} \in H^{X,Z}, p^{\downarrow Y_1,Y_2,Z} \in H^{Y_1,Y_2,Z}\}$$

  El conjunto global de probabilidades para (X, Y_1, Z) es

$$H^{\downarrow X,Y_1,Z} = \{p^{\downarrow X,Y_1,Z} : p \in H\} = \{p^{\downarrow X,Y_1,Z} : p^{\downarrow X,Z} \in H^{X,Z}, p^{\downarrow Y_1,Y_2,Z} \in H^{Y_1,Y_2,Z}\}$$

  Si llamamos q a $p^{\downarrow X,Y_1,Z}$, entonces $p^{\downarrow X,Z} \in H^{X,Z}$ es equivalente a $q^{\downarrow X,Z} \in H^{X,Z}$. Y, por otra parte, dada una distribución q, existe p tal que $p^{\downarrow X,Y_1,Z} = q$ y $p^{\downarrow Y_1,Y_2,Z} \in H^{Y_1,Y_2,Z}$ si y sólo si $q^{\downarrow Y_1,Z} \in H^{Y_1,Z}$. Por tanto la anterior expresión se puede escribir como:

$$H^{\downarrow X,Y_1,Z} = \{q : q^{\downarrow X,Z} \in H^{X,Z}, q^{\downarrow X,Y_1,Y_2} \in H^{Y_1,Y_2,Z}\} = H^{X,Z} \cap H^{Y_1,Z}$$

  Con lo que queda demostrado que existe independencia marginal de X e Y_1 dado Z.

- Para la unión débil, tenemos que probar que si X es independiente de (Y_1, Y_2) dado Z, entonces X es independiente de Y_2 dado (Z, Y_1). Siguiendo el mismo proceso que en el caso anterior el conjunto global de probabilidades es:

$$H = \{p : p^{\downarrow X,Z} \in H^{X,Z}, p^{\downarrow Y_1,Y_2,Z} \in H^{Y_1,Y_2,Z}\}$$

Si p es una distribución de probabilidad tal que $p^{\downarrow X,Z} \in H^{X,Z}$ y $p^{\downarrow Y_1,Y_2,Z} \in H^{Y_1,Y_2,Z}$, tendremos además que $p^{\downarrow Y_1,Z} \in H^{Y_1,Z}$. Según hemos demostrado en el caso anterior, $H^{X,Y_1,Z} = H^{X,Z} \cap H^{Y_1,Z}$. Por tanto, podemos deducir de $p^{\downarrow Y_1,Z} \in H^{Y_1,Z}$ y $p^{\downarrow X,Z} \in H^{X,Z}$ que $p^{X,Y_1,Z} \in H^{X,Y_1,Z}$. Como por otra parte, siempre que $p^{X,Y_1,Z} \in H^{X,Y_1,Z}$ se tiene que $p^{\downarrow X,Z} \in H^{X,Z}$, entonces se puede expresar

$$H = \{p : p^{\downarrow X,Y_1,Z} \in H^{X,Y_1,Z}, p^{\downarrow Y_1,Y_2,Z} \in H^{Y_1,Y_2,Z}\}$$

Y esta expresión es equivalente a lo que queremos demostrar: X es marginalmente independiente de Y_2 dados (Y_1, Z).

- Para la contracción, hemos de probar que si X es independiente de Y_1 dado Z y X independiente independiente de Y_2 dados (Y_1, Z) entonces tenemos que X es independiente de (Y_1, Y_2) dado Z.

  Si X es independiente de Y_2 dados (Y_1, Z), entonces podemos expresar el conjunto global de probabilidades como

  $$H = \{p : p^{\downarrow X,Y_1,Z} \in H^{X,Y_1,Z}, p^{\downarrow Y_1,Y_2,Z} \in H^{Y_1,Y_2,Z}\}$$

  Si X es independiente de Y_1 dado Z, el conjunto $H^{X,Y_1,Z}$ se puede expresar como $H^{X,Z} \cap H^{Y_1,Z}$. Entonces, tenemos que

  $$H = \{p : p^{\downarrow X,Z} \in H^{X,Z}, p^{\downarrow Y_1,Z} \in H^{Y_1,Z}, p^{\downarrow Y_1,Y_2,Z} \in H^{Y_1,Y_2,Z}\}$$

  Como $p^{\downarrow Y_1,Y_2,Z} \in H^{Y_1,Y_2,Z}$ implica que $p^{\downarrow Y_1,Z} \in H^{Y_1,Z}$, la expresión anterior se puede simplificar a:

  $$H = \{p : p^{\downarrow X,Z} \in H^{X,Z}, p^{\downarrow Y_1,Y_2,Z} \in H^{Y_1,Y_2,Z}\}$$

  Y esto es lo que queríamos demostrar: X es marginalmente independiente de (Y_1, Y_2) dado Z.

■

La base para la reducción de los cálculos en el algoritmo de propagación es la aplicación de la regla que se daba en la sección 2.3.1 mediante la expresión 2.7. Según ella, en cuanto haya una variable que aparece en una sola valuación, ésta puede eliminarse por marginalización en esa valuación. Para ello en el algoritmo de eliminación de variables se selecciona una variable X_i, y se combinan todas las valuaciones que contienen a esta variable. En el resultado de esta combinación podemos eliminar X_i mediante marginalización pues ya sólo aparece en una valuación. Al eliminar una variable en el problema estamos reduciendo el tamaño de nuestros datos, es decir, el tamaño de las valuaciones que tenemos que combinar y por tanto el tamaño de los cálculos. Diferentes órdenes de eliminación de variables pueden dar lugar a algoritmos más o menos eficientes. Podemos resumir este algoritmo de forma esquemática de la siguiente forma para el caso de propagación de conjuntos convexos de probabilidad representados por restricciones:

Algoritmo 2.5 Propagación de restricciones por eliminación de variables

1. Sea $\mathcal{V}$ el conjunto de valuaciones del problema.
2. Establecer un orden ρ de eliminación de las variables del conjunto $\{1, 2, \ldots, n\} - J$
3. Mientras ρ no esté vacío:
   (a) Borra la primera variable X_k de ρ
   (b) Calcular $V = \bigcap V_i$ tal que V_i son las valuaciones que contienen a la primera variable X_k. Sea I el conjunto de variables para las que está definida V.
   (c) Añade $V^{\downarrow I-k}$ al conjunto de valuaciones $\mathcal{V}$ y elimina las valuaciones V_i del paso anterior.
4. Devolver la única valuación V que quede en $\mathcal{V}$

■

El aspecto más importante a la hora de implementar el algoritmo es ver como se realizan las dos operaciones básicas de combinación y marginalización. Si los convexos los estamos representando por un conjunto de restricciones lineales, la combinación es fácil de realizar,

pues sólo tenemos que hacer la unión de conjuntos de restricciones. Si queremos mantener siempre un conjunto minimal de restricciones deberíamos aplicar un algoritmo de eliminación de redundancias. Pero, en general, la ganancia que se consigue al eliminar las redundancias con algún algoritmo de este tipo, no compensa el costo de aplicarlo. Si que podríamos aplicar un algoritmo que detecte algunas de las redundancias en un corto periodo de tiempo, aunque sólo reduzca las redundancias más simples. Un ejemplo de este tipo de algoritmo ha sido dado por Imbert y Van Hentenryck [72]. La operación de marginalización puede implementarse también de forma que no genere restricciones redundantes. Por tanto, podríamos esperar y no eliminar redundancias hasta que no hagamos una marginalización.

La marginalización es más difícil de llevar a cabo cuando el conjunto convexos viene expresado en términos de restricciones lineales. Hansen y Jaumard [64] argumentan que esta operación involucra la enumeración de los vértices del conjunto convexo, lo cual es una operación muy costosa computacionalmente. Sin embargo puede conseguirse un ahorro computacional, ya que la marginalización puede realizarse enumerando solamente los puntos del conjunto marginalizado, $H^{\downarrow I}$, en lugar de hacerlo sobre el conjunto convexo original H. Esto hace que la marginalización sea mucho más eficiente ya que el número de puntos extremos en $H^{\downarrow I}$ es mucho menor que en H. Una forma de efectuar la marginalización de esta forma es realizando la marginalización como un caso particular de *proyección* a un determinado subespacio [88, 131]. Un ejemplo de este tipo de algoritmos puede obtenerse con la aplicación de la técnica de eliminación de cuantificadores de Lassez y Lassez [88]. Este algoritmo realiza la proyección de un conjunto convexo a un determinado subespacio, recorriendo los puntos extremos de la proyección en lugar de los puntos del conjunto convexo que se quiere proyectar. Lassez y Lassez establecen un algoritmo que permite realizar la proyección de un politopo convexo determinado por un conjunto de restricciones. Esta se realiza por aproximaciones sucesivas de una aproximación inicial que se obtiene al calcular suficientes puntos extremos para que la cápsula convexa sea de dimensión total en el espacio de proyección. Estos puntos se obtienen al maximizar hiperplanos ortogonales al subespacio de proyección sobre el politopo convexo original. Al realizar esta maximización se alcanza un punto extremo cuya proyección será un punto extremo del politopo proyectado. Posteriormente se realizan refinamientos sucesivos que consisten en añadir nuevos puntos extremos según el proceso anterior, y actualizar la cápsula convexa: Si una restricción no pertenece a la proyección entonces se determina un nuevo punto extremo y actualizamos la cápsula convexa.

Verdegay-López [131] presenta una modificación al algoritmo de Lassez y Lassez para realizar la proyección, que hace que cuando se determina un nuevo punto extremo de la proyección, siempre se pasa a un punto extremo adyacente, mientras que el algoritmo de Lassez y Lassez puede dar lugar a un punto extremo que se encuentre muy alejado del anterior. Esto conlleva un mayor gasto computacional debido a que hay que realizar varios cambios de base para pasar de un punto extremo a otro. En el algoritmo de Verdegay-López sólo hay que realizar un cambio de base. Por otra parte al determinar los semiespacios de la cápsula convexa actualizada de la proyección, en el algoritmo de Lassez se pueden obtener muchos semiespacios que, al final, no pertenecen al convexo proyectado y por lo tanto deben ser refinados en pasos posteriores. En el algoritmo de Verdegay, al considerar puntos extremos adyacentes, siempre se van a determinar semiespacios que van a pertenecer al politopo proyectado.

2.4.1 Otros algoritmos de propagación de restricciones

Podemos encontrar en la literatura algunos algoritmos de propagación de conocimiento general (restricciones) basados en la aplicación de reglas locales. En general, en todos estos procedimientos no es posible propagar todo tipo de restricción, sino sólo algunos tipos particulares, usualmente límites en las probabilidades condicionales. Las reglas propuestas son siempre correctos pero, en la mayoría de los casos, no son completos (no hay garantía que obtengamos los límites óptimos). Los trabajos más relevantes en esta dirección pueden encontrarse en Amarger, Dubois, y Prade [3], Dubois y otros [41], Thöne [129], Lukasiewicz [92], y Salo [114].

Amarger, Dubois, y Prade [3], Thöne [129], y Lukasiewicz [92] consideran variables proposicionales $\{A_1, \ldots, A_n\}$, que toman sólo dos posibles valores: *true* y *false*. Luego, consideran reglas de la forma $A \overset{u_1,u_2}{\longrightarrow} B$, con el significado que $P(A) > 0$ y $0 \leq u_1 \leq P(B|A) \leq u_2 \leq 1$. Es posible también con reglas bidireccionales $A \underset{y_1,y_2}{\overset{u_1,u_2}{\longleftrightarrow}} B$ donde $(u_2 = 0 \Leftrightarrow y_2 = 0)$, con el significado que $A \overset{u_1,u_2}{\longrightarrow} B$ y $B \overset{y_1,y_2}{\longrightarrow} A$. Estos autores proporcionan reglas para obtener nuevos límites para sucesos de este tipo a partir de los límites dados. Dubois y otros [41] generalizan estos límites al caso de probabilidades lingüísticas (sólo un valor lingüístico se asigna a la probabilidad de un suceso).

Salo [114] da otras reglas locales que no son siempre precisas, pero que sirven para casos más generales de restricciones lineales. Se consideran sólo dos variables, que llamaremos X e Y. Primero, tenemos un conjunto de restricciones lineales

$$\sum_{i=1}^{n} a_i^k p(u_i) \leq \alpha_k, \quad k = 1, \ldots, K \tag{2.15}$$

que representa un conjunto convexo H^X (información marginal sobre X).

Para cada probabilidad $p(v_j|u_i)$ para un valor fijo v_j, tenemos un conjunto convexo, $H^{Y=v_j|X}$, dados por un conjunto de restricciones lineales:

$$\sum_{i=1}^{n} b_i^l p(v_j|u_i) \leq \beta_l, \quad l = 1, \ldots, L \tag{2.16}$$

Salo da procedimientos para calcular los conjuntos convexos generados para los valores de $p(u_i|v_j)$.

2.5 Propagación de conjuntos convexos usando relaciones de independencia

Particularicemos ahora los algoritmos de propagación al caso en que se da la independencia fuerte (definición 1.19) y tengamos un sistema de independencias que venga dado por un grafo dirigido acíclico. Estos algoritmos permitirán calcular conjuntos convexos 'a posteriori' dado un conjunto de observaciones sobre ciertas variables del problema y suponiendo que existen relaciones de independencia entre las variables del problema que vendrán representadas a través de un grafo de dependencias.

Supongamos que tenemos un conjunto convexo de probabilidades H sobre una variable n-dimensional $(X_1, \ldots, X_n)$. Este conjunto convexo se va a especificar a través de informaciones locales $H_1, \ldots, H_n$, de forma que $H = H_1 \otimes \ldots \otimes H_n$. También es posible que conozcamos el valor de algunas de las variables de $(X_1, \ldots, X_n)$. El conjunto de variables observadas junto con su valor lo denotaremos con E.

El proceso para obtener la descomposición será el descrito en la sección 2.2.1. Para aplicarlo tenemos que probar que la definición de independencia condicional fuerte verifica los axiomas de independencia condicional.

Teorema 2.3 *La independencia condicional fuerte (definición 1.20) cumple los axiomas de Simetría, Descomposición y Unión Débil dadas en la definición 2.3.*

Demostración:

- La simetría es inmediata a partir de la definición de independencia fuerte.
- Para la descomposición, tenemos que probar que si X es independiente de (Y_1, Y_2) dado Z, entonces X es independiente de Y_1 dado Z. De acuerdo con la definición, si se da la primera independencia, tenemos que el conjunto global de probabilidades posibles para (X, Y_1, Y_2, Z) se puede descomponer como:

  $$H = H_1^{X,Z} \otimes H_2^{Y_1,Y_2,Z}$$

  Marginalizando este producto en Y_2 y teniendo en cuenta que $H^{X,Z}$ no depende de Y_2, tenemos que:

  $$H^{X,Y_1,Z} = H_1^{X,Z} \otimes (H_2^{Y_1,Y_2,Z})^{\downarrow(Y_1,Z)}$$

  De esta expresión se deduce que X es independiente de Y_1 dado Z.
- Para la unión débil, supongamos X es independiente de (Y_1, Y_2) dado Z, es decir que se verifica la descomposición

  $$H = H_1^{X,Z} \otimes H_2^{Y_1,Y_2,Z}$$

  Hagamos $H\prime_1 = H_1^{X,Z} \otimes \{1_{Y_2}\}$. Este convexo se obtiene a partir de $H_1^{X,Z}$ extendiendo cada uno de sus puntos $h(x,z)$ a una función $h(x,z,y_1)$ definida por $h(x,z,y_1) = h(x,z), \forall y_1$. Es evidente, que se verifica la descomposición:

  $$H = H\prime_1 \otimes H_2^{Y_1,Y_2,Z}$$

  Como $H\prime_1$ es un convexo sobre (X, Y_1, Z) se obtiene que X es independiente de Y_2 dados (Y_1, Z).

■

El axioma de contracción no siempre se verifica en esta definición. En nuestro enfoque esto no será un problema, ya que las independencias que usemos estarán siempre asociadas a un grafo dirigido acíclico y es bien sabido que las independencias que se obtienen a partir de un grafo por el criterio de D-separación si verifican todos los axiomas de independencia.

2.5.1 Particularización de la axiomática para conjuntos convexos de probabilidades

Las valuaciones en U_I en el caso de probabilidades imprecisas son conjuntos convexos, H, de funciones no necesariamente normalizadas de U_I en $\mathbb{R}_0^+$, o sea, vectores en $\mathbb{R}^{\prod_{i\in I} n_i}$ (donde n_i es el número de elementos de U_i), con componentes no negativos. Dos valuaciones V_1 y V_2, que representen los conjuntos convexos, H_1 y H_2 serán equivalentes, si añadiendo la función nula, $h_c(u) = 0,\ \forall u \in U_1 \times \ldots \times U_n$, sus cápsulas convexas son proporcionales, o sea, $\exists \alpha \in \mathbb{R}^+$, tal que

$$\mathrm{CH}(H_1 \cup \{h_c\}) = \alpha.\mathrm{CH}(H_2 \cup \{h_c\})$$

En Cano, J.E. [18] puede verse una completa justificación de la inclusión de h_c en la definición de equivalencia de valuaciones para conjuntos convexos de probabilidades. Resumiendo brevemente, h_c representa la presencia de una probabilidad imposible, y su inclusión no modifica el estado de conocimiento que tenemos sobre un problema. El añadir o eliminar, h_c, no cambiará nunca los intervalos de probabilidad finales. Sin embargo, su inclusión tiene la ventaja de que, a veces, reduce el número de puntos extremos necesarios para representar la misma información.

Una observación de un valor para una variable se representará de la misma forma que en el caso probabilístico.

Las operaciones de combinación y marginalización en términos de valuaciones para conjuntos convexos de probabilidades están basadas en las definidas en el capítulo 1 mediante las expresiones 1.4 para la marginalización y 1.5 para la combinación.

- Si V_1 está definida en $\prod_{i\in I} U_i$ y dada por los puntos extremos $\{h_1, \ldots, h_n\}$ y V_2, está definida en $\prod_{i\in I} U_i$, y dada por los puntos extremos $\{f_1, \ldots, f_m\}$, entonces $V_1 \otimes V_2$ es el conjunto convexo, definido en $\prod_{i\in(I\cup J)} U_i$ generado por las funciones,

  $$\{h_1.f_1, \ldots, h_1.f_m, \ldots\ldots, h_n.f_1, \ldots, h_n.f_m\}$$

  donde $(h_k.f_l)(u) = h_k(u^{\downarrow I}).f_l(u^{\downarrow J})$.

- Si V está definida en $\prod_{i\in I} U_i$ y dada por las funciones $\{h_1, \ldots, h_n\}$ y $J \subseteq I$ entonces $V^{\downarrow J}$ es el conjunto convexo generado por las funciones

  $$\{h_1^{\downarrow J}, \ldots, h_n^{\downarrow J}\}$$

El elemento neutro es el conjunto con un solo elemento, la función h_0, donde $h_0(u) = 1, \forall u \in U_1 \times \ldots \times U_n$. La contradicción tiene un único punto, la función nula, h_c.

Cano, J.E [18] demuestra que las anteriores valuaciones con sus correspondientes operadores de marginalización y combinación verifican los axiomas 1-6.

2.5.2 Algoritmos de propagación para conjuntos convexos

En nuestro problema tenemos un conjunto de valuaciones $\{H_i\}_{i\in\{1,\ldots,n\}}$ correspondientes a conjuntos convexos de probabilidad, cada una de estas valuaciones se corresponde con uno de los nodos del grafo de dependencias. Tendremos valuaciones que corresponden a informaciones marginales (nodos raíz) y valuaciones condicionales. También existe un conjunto de observaciones $E = \{O_k\}_{k\in K}$ sobre algunas de las variables de $\{X_1, \ldots, X_n\}$. Queremos calcular de forma eficiente las valuaciones 'a posteriori':

$$PS_j = ((\otimes_{k\in K} O_k) \otimes (\otimes_{i\in\{1,\ldots,n\}} V_i))^{\downarrow\{j\}}$$

para cada $j \in \{1, \ldots, n\}$

La propagación se puede obtener particularizando los algoritmos de borrado o la arquitectura de Shafer y Shenoy. El resultado de la propagación para esta variable X_j será un conjunto convexo PS_j de funciones de U_j en $[0, 1]$. Si suponemos que X_j tiene dos casos $\{u_j^1, u_j^2\}$ entonces el resultado de la propagación es un conjunto convexo sobre $\mathbb{R}^2$ de la forma de la figura 2.9. Los puntos de este conjunto convexo PS_j pueden obtenerse de la siguiente forma: Si P es una distribución de probabilidad global, formada seleccionando una distribución de probabilidad para cada conjunto convexo, entonces obtendremos un punto $(t_1, t_2) \in PS_j$ asociado a esta distribución P donde,

$$t_1 = P(u_j^1 \cap E)$$
$$t_2 = P(u_j^2 \cap E)$$

y E es la evidencia dada o familia de observaciones $\{O_k\}_{k\in K}$.

En esta figura podemos apreciar que tenemos cinco puntos extremos en el conjunto convexo (más el punto $(0, 0)$).

A partir de PS_j podemos calcular un intervalo de probabilidad utilizando algunos de los métodos de condicionamiento descritos en el capítulo 1. En la figura 2.10 podemos ver gráficamente el resultado de aplicar el condicionamiento de Dempster para la obtención de intervalos

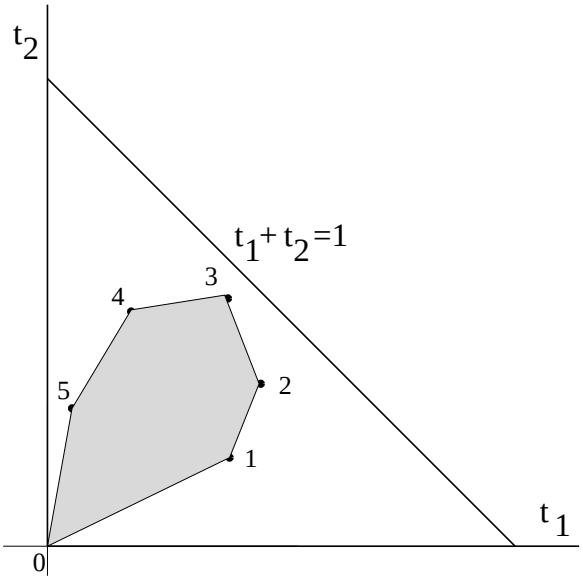


Figura 2.9: Conjunto convexo 'a posteriori'

para la variable objetivo X_j cuando se tiene el conjunto convexo 'a posteriori' de la figura 2.9. Se pueden observar en esta figura los intervalos para $P(X_j = u_j^1)$ y $P(X_j = u_j^2)$. Se puede observar que la inclusión del punto $(0, 0)$ no ha influido en los intervalos obtenidos.

En la figura 2.11 se puede observar gráficamente la aplicación del condicionamiento de Moral y Campos para la obtención de los intervalos para $P(X_j = u_j^1)$ y $P(X_j = u_j^2)$. Aquí se puede comprobar que tampoco la inclusión del punto $(0, 0)$ afecta a los intervalos obtenidos. En este caso los intervalos obtenidos se puede observar que son más pequeños que los obtenidos con el condicionamiento de Dempster. Esto es así porque en el condicionamiento de Moral y Campos aquellos puntos que están más alejados de la recta $t_1 + t_2 = 1$ afectarán con menos fuerza al resultado que los más cercanos. En la figura se ve como el punto número 5, que está muy alejado, no afectará a los intervalos obtenidos. Este punto número 5 en el condicionamiento de Dempster determinaba el límite inferior para $P(X_j = u_j^1)$ y el límite superior para $P(X_j = u_j^2)$.

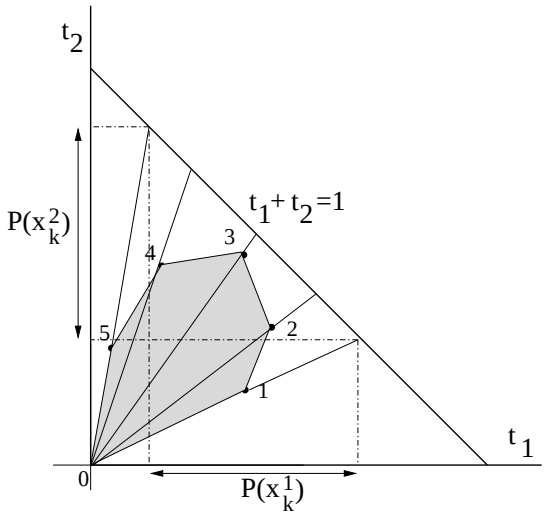


Figura 2.10: Intervalos para el conjunto convexo 'a posteriori' con el condicionamiento de Dempster

2.5.3 Otros algoritmos de propagación de probabilidades imprecisas bajo independencia fuerte

2.5.3.1 Algoritmos de propagación simbólica

En la sección 2.3 se han estudiado algunos métodos disponibles en la literatura aplicables cuando tenemos un grafo de dependencias en donde en cada nodo se tiene una distribución de probabilidad condicional y queremos calcular la probabilidad ´a posteriori´ en una de las variables del grafo de dependencias. Cuando algunas de estas probabilidades sean especificadas con un símbolo en lugar de con un número, entonces a estos algoritmos se les conoce como *algoritmos de propagación simbólica* [26]. Los métodos de propagación simbólica conducen a soluciones generales que se expresan como funciones de los parámetros. Con estas funciones generales podemos obtener respuestas específicas sin más que sustituir los valores de los parámetros en la solución simbólica, sin necesidad de rehacer la propagación. Según Castillo et al. [26] la propagación simbólica es útil en los siguientes casos:

- Cuando no se dispone de la especificación numérica del modelo probabilístico.
- Cuando los especialistas sólo son capaces de especificar intervalos de los parámetros en vez de valores concretos.
- Cuando se requiere un análisis de la sensibilidad.

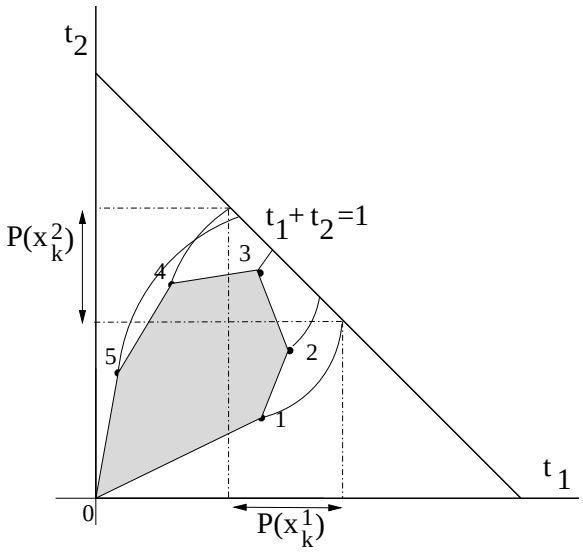


Figura 2.11: Intervalos para el conjunto convexo 'a posteriori' con el condicionamiento de Moral y Campos

Aquellos nodos X_i en los que su distribución de probabilidad condicional contenga al menos un símbolo, o sea que una probabilidad no venga dada con un número, se conocen como *nodos simbólicos* [26].

Para obtener la expresión simbólica para la probabilidad ´a posteriori´ en una variable de interés X_j podríamos aplicar la expresión 2.3 y luego simplificar la expresión obtenida. Este método de resolver el problema es muy costoso computacionalmente y resulta ineficiente incluso con un número reducido de variables ya que conlleva calcular la distribución de probabilidad conjunta para todas las variables. Una alternativa a este método la dan Castillo, Gutiérrez y Hadi [28] que han adaptado los métodos numéricos de propagación de probabilidades descritos en la sección 2.3 a la propagación con símbolos. Para ello hacen uso de una herramienta matemática, tal como *Mathematica*. Por otra parte Castillo, Gutiérrez y Hadi [27] desarrollan un nuevo método de propagación simbólica que aprovecha la estructura algebraica de las probabilidades según la cual la probabilidad ´a posteriori´ de una variable puede expresarse con un polinomio en los parámetros simbólicos de grado menor o igual al número de nodos simbólicos, o bien, con una función racional (cociente de dos funciones polinomiales) de los parámetros. Este nuevo método llamado *método de las componentes canónicas* requiere utilizar un método de propagación de probabilidades tantas veces como número de posibles combinaciones de los parámetros simbólicos haya, y posteriormente resolver un sistema lineal de ecuaciones para poder calcular los coeficientes de los polinomios.

2.5.3.2 Algoritmos de propagación con intervalos de probabilidad

Un caso particular importante que ha recibido cierta atención en la literatura es cuando se conoce un intervalo de probabilidad para cada caso de una variable X_i dada cada configuración de sus padres [11, 48, 128, 46]. La situación en este caso es diferente a la de la propagación de restricciones dadas por intervalos (ver sección 2.4) ya que aquí consideramos relaciones de independencia fuerte representadas con un grafo dirigido acíclico.

Supongamos que para cada posible valor u de la variable X_i y cada configuración v de sus padres $pa(X_i)$ tenemos un par de valores $(\alpha(.|v), \beta(.|v))$ que representan un intervalo de probabilidad. Para cada configuración v de $pa(X_i)$, sea n_v el número de puntos extremos del conjunto convexo $H^{X_i|pa(X_i)=v}$. Entonces, el conjunto convexo condicional global $H^{X_i|pa(X_i)}$ tendrá $\prod_{v \in U_{pa(X_i)}} n_v$ como número de puntos extremos. Teniendo en cuenta que el número de elementos de $U_{pa(X_i)}$ es $\prod_{k \in pa(X_i)} |U_k|$, tenemos una explosión combinatoria del número de puntos extremos de la información condicional global $H^{X_i|pa(X_i)}$. Debido a ello, ninguna de las aproximaciones que hay en la literatura para tratar el problema de la propagación de este tipo de informaciones ha considerado el problema en toda su generalidad. Tessen [128] ha considerado el caso de grafos acíclicos dirigidos sin ciclos (ciclos no dirigidos). Por otra parte, Fertig y Brease [47, 11, 48] sólo consideran los límites inferiores $\alpha(.|v)$. Fagiuoli y Zaffalon [46] han desarrollado algoritmos exactos para grafos sin ciclos con variables binarias, los cuales son lineales en el tamaño de las redes.

2.5.4 Algoritmos que hacen explícita la imprecisión de las probabilidades

Describamos en esta sección una transformación del problema de la propagación de probabilidades imprecisas que hacen explícita mediante la adición de variables adicionales nuestro desconocimiento de los valores exactos de las probabilidades. En estas variables se considerará un posible valor por cada posibilidad extrema para una probabilidad dada. Esto será especialmente útil para los algoritmos aproximados del próximo capítulo.

Originalmente, cada variable X_i tiene asociada una valuación condicional H_i de esa variable X_i a sus padres $pa(X_i)$, que lógicamente estará definida para el conjunto de variables $X_i \cup pa(X_i)$. Esta valuación es un conjunto convexo de posibles distribuciones de probabilidad condicionales $H_i = \{h_1, \ldots, h_{k_i}\}$. En estas condiciones añadimos una nueva variable ficticia a la valuación H_i. Esta nueva variable ficticia la llamaremos *variable transparente* T_i y tendrá tantos posibles valores $\{c_1, \ldots, c_{k_i}\}$ como puntos extremos tenga la valuación condicional H_i.

De esta forma fijando un valor de la variable transparente en la valuación condicional H_i nos da una subvaluación que corresponde con uno de los puntos extremos de H_i. Esta variable T_i se puede poner como un nuevo padre de X_i en el grafo de dependencias. Con esta transformación del problema, la probabilidad de X_i dados sus padres es una única y determinada distribución de probabilidad. Sobre la variable T_i no se conoce nada, o sea, consideramos que todas las distribuciones de probabilidad sobre este nodo son posibles, o lo que es lo mismo, es como si el nodo transparente T_i (nodo raíz en el grafo de dependencias) tuviese un conjunto convexo con k_i puntos extremos, cada uno de ellos degenerado en uno de los posibles casos de T_i. En otras palabras consideramos que en la variable T_i tenemos una valuación V_{T_i} que representa la *ignorancia*.

En la figura 2.12 podemos observar la transformación que se ha hecho sobre el grafo de dependencias de la figura de la izquierda, considerando que en este grafo se tenía un conjunto convexo para cada distribución condicional

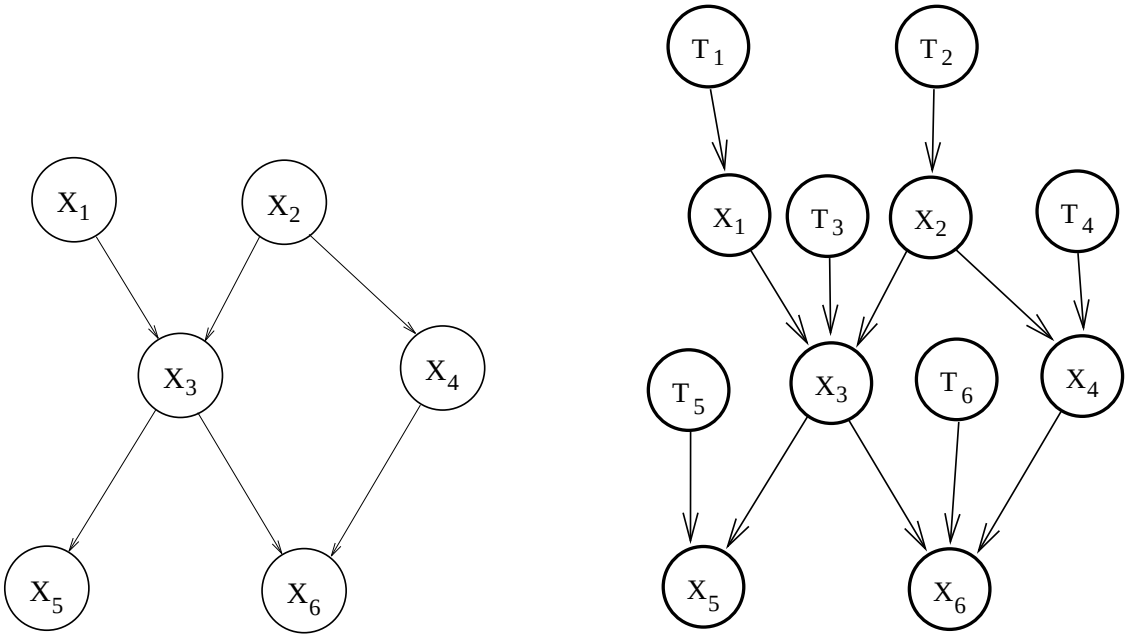


Figura 2.12: Transformación del grafo de dependencias

Puede comprobarse que la estructura del problema no cambia con esta transformación. La única cosa que hemos hecho es que nuestro estado de conocimiento sobre las probabilidades condicionales se ha hecho explícito con la ayuda de un nuevo nodo que expresa todas las posibles distribuciones de probabilidad condicionales. Nada es conocido sobre este nodo.

Para adaptar los algoritmos de propagación haremos lo siguiente. En el proceso para construir el árbol de grupos (obtención del grafo moral, triangulación y obtención de los grupos) no se tendrán en cuenta los nodos transparentes. Los grupos estarán definidos entonces sólo para subconjuntos de variables de $\{X_1, \ldots, X_n\}$. Una vez que hemos construido el árbol de grupos añadiremos a cada uno de los grupos todo el conjunto de variables transparentes $\{T_1, \ldots, T_n\}$ y se asignará cada una de las valuaciones H_i a un grupo que contenga todas sus variables. Las valuaciones V_{T_i} que representan la ignorancia no se introducen. De igual forma será preciso incluir todas las variables transparentes en los dos mensajes que hay entre cada par de grupos. En la figura 2.13 podemos observar cual sería el árbol de grupos correspondiente al grafo de dependencias de la figura 2.12.

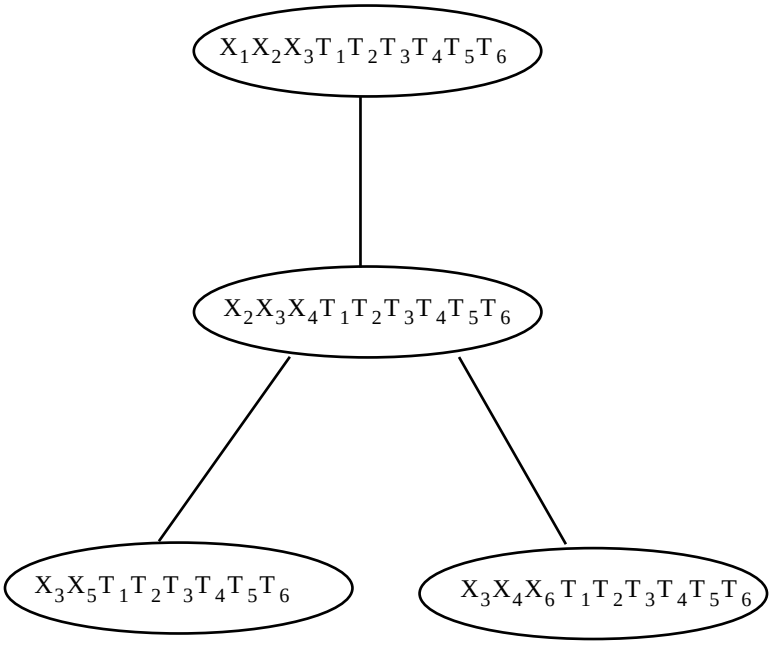


Figura 2.13: Árbol de grupos para conjuntos convexos: Inclusión de variables transparentes en cada grupo

Ahora, lo que ocurre es que en cada valuación inicial, para cada punto extremo del convexo tendremos una configuración de nodos transparentes tal que fijándola en la valuación a un determinado valor, se obtiene dicho punto extremo. El algoritmo resultante es equivalente al de propagación de la sección 2.3 sin eliminar puntos extremos.

De igual forma que el algoritmo de árboles de grupos se puede aplicar a la propagación de conjuntos convexos, también se puede aplicar el algoritmo de eliminación de variables. En este caso también hacemos la misma transformación del problema original añadiendo una variable

transparente T_i por cada conjunto convexo condicional. En el algoritmo de eliminación de variables sólo hay que tener en cuenta que las variables transparentes no se eliminarán nunca. Al final del algoritmo tendremos una valuación definida para la variable objetivo X_j y todas las variables transparentes $\{T_1, \ldots, T_n\}$. Esta valuación nos define el conjunto convexo 'a posteriori'. O sea, por cada posible combinación de los casos de las variables transparentes en la valuación resultado se obtiene un vector o punto. No todos los puntos serán puntos extremos del conjunto convexo 'a posteriori', pero seguro que todos los puntos extremos se pueden obtener de esta forma.

2.6 Experimentación: algoritmos heurísticos para la triangulación de grafos

Olmsted [103], Kong [84], Kjærulff [81] propusieron la siguiente heurística para obtener una buena secuencia de borrado:

H1 *En cada paso, entre las variables aun no borradas seleccionar aquella que produce un grupo de mínimo tamaño y borrarla*

Esta heurística produce resultados muy buenos en el caso general. Intenta minimizar la suma de los tamaños de los grupos minimizando en cada etapa, el tamaño de cada uno de los grupos que van siendo creados. Esto no garantiza que el tamaño del árbol de grupos sea óptimo, ya que al seleccionar una variable que produce un grupo de mínimo tamaño puede forzar a producir grupos de mayor tamaño posteriormente al borrar las otras variables, pero en general produce árboles de grupos con tamaños cercanos al óptimo.

Nosotros proponemos nuevas heurísticas en las que hemos desarrollado la idea subyacente es que cuando se borra una variable estamos creando un grupo con un tamaño que debe minimizarse. Sin embargo, al mismo tiempo, estamos borrando esta variable y todos los arcos que unen esta variable con otras variables, y por tanto simplificando el grafo. Esta simplificación en el grafo resultante puede servir también de guía para obtener triangulaciones eficientes. Según esta idea introducimos las siguientes técnicas heurísticas para encontrar la secuencia de borrado, y por tanto la triangulación:

Si X_i es una posible variable para borrarse entonces,

- $S(i)$ es el tamaño del grupo creado al borrar esta variable.

- $E(i)$ es el número de elementos de U_i.
- $M(i)$ es el tamaño máximo de los grupos del subgrafo dado por X_i y sus nodos adyacentes.
- $C(i)$ es la suma de los tamaños de los grupos del subgrafo dado por X_i y sus nodos adyacentes.

Los algoritmos heurísticos siguen las siguientes reglas:

- **H2** *En cada etapa seleccionar la variable,* X_i*, que tenga mínimo valor de* $S(i)/E(i)$.
- **H3** *En cada etapa seleccionar la variable,* X_i*, que tenga mínimo valor de* $S(i) - M(i)$.
- **H4** *En cada etapa seleccionar la variable,* X_i*, que tenga mínimo valor de* $S(i) - C(i)$.
- **H5** *En cada etapa seleccionar la variable,* X_i*, que tenga mínimo valor de* $S(i)/M(i)$.
- **H6** *En cada etapa seleccionar la variable,* X_i*, que tenga mínimo valor de* $S(i)/C(i)$.

En las anteriores heurísticas, cuando tenemos dos o más variables igualmente buenas para borrar, borramos una de ellas.

Las heurísticas H_5 y H_6 son similares a las heurísticas H_3 y H_4 respectivamente, pero en lugar de calcular una diferencia, se calcula el factor en el incremento de los grupos. La complejidad de H_2 es igual a la complejidad de H_1. La idea de simplificar el grafo resultante es sólo parcialmente considerada en H_2: la causa de borrar una variable X_i es el tamaño del grupo creado, $S(i)$. Se considera que el grafo resultante es más simple si borramos la variable que tiene más casos. Dividiendo $S(i)$ por el número de casos, $E(i)$, equilibramos el costo de borrar una variable con la simplicidad del grafo: indicadores más precisos de esta simplicidad se consideran en las heurísticas H_3, H_4, H_5 y H_6. Las bases intuitivas de estas reglas son las siguientes: si borramos la variable X_i, los arcos del grafo antiguo, no presentes en el nuevo grafo, son los arcos que unen X_i con sus nodos adyacentes. $C(i)$ y $M(i)$ intentan medir la complejidad del subgrafo dado por estos arcos. Pero estas reglas son más complejas que H_1 y H_2 porque debemos calcular los grupos del grafo, aunque este grafo será generalmente más pequeño porque es el grafo dado por un nodo y sus nodos adyacentes.

Evaluación de las heurísticas de triangulación: Presentamos aquí unos experimentos que hemos realizado para evaluar las heurísticas explicadas más arriba. En la evaluación de estos algoritmos heurísticos se generaron 500 grafos de cada uno de los siguientes grupos, que fueron triangulados con cada una de las seis heurísticas:

a) Los grafos tienen 50 nodos que se numeran desde 1 hasta 50. Cada nodo tiene 2 casos, un número de padres entre 0 y 5, y generado de acuerdo a una distribución uniforme de media $2,5$ redondeando al entero más cercano. Los padres son los nodos inmediatamente predecesores al nodo dado. Este método produce grafos tipo cadena.

b) El mismo que a), pero ahora los padres se seleccionan aleatoriamente entre los nodos predecesores. Los nodos más cercanos tendrán una probabilidad más alta de ser elegidos entre los padres.

c) El mismo que a), pero ahora los padres se seleccionan aleatoriamente entre todos los nodos predecesores. Todos los nodos tienen la misma probabilidad de ser elegidos como padres.

d) El mismo que a), pero ahora el número de casos de cada nodo se selecciona de acuerdo a una distribución de Poisson de media 2 con un mínimo de 2.

e) El mismo que b), pero ahora el número de casos de cada nodo se selecciona con una distribución de Poisson de media 2 con un mínimo de 2.

f) El mismo que c), pero ahora el número de casos de cada nodo se selecciona con una distribución de Poisson de media 2 con un mínimo de 2.

g) El mismo que a), pero ahora el número de casos de cada nodo se selecciona con una distribución de Poisson de media 3 con un mínimo de 2.

h) El mismo que b), pero ahora el número de casos de cada nodo se selecciona con una distribución de Poisson de media 3 con un mínimo de 2.

i) El mismo que c), pero ahora el número de casos de cada nodo se selecciona con una distribución de Poisson de media 3 con un mínimo de 2.

j) El mismo que a), pero ahora el número de casos de cada nodo se selecciona con una distribución de Poisson de media 4 con un mínimo de 2.

Método	H1	H2	H3
a)	489	489	428
b)	19.965	19.965	18.027
c)	23.378	23.378	22.826
d)	2.700	2.613	2.278
e)	2.158.184	1.606.454	2.199.538
f)	4.457.325	2.420.105	2.890.733
h)	8.377.588	7.981.114	7.943.450
g)	6.606	6.407	5.548
i)	65.147.965	32.137.124	53.160.398
j)	16.825	16.275	14.189
k)	130.974.850	128.772.625	131.136.043
l)	1.627.641.198	1.506.177.237	1.605.723.127
	H4	H5	H6
a)	483	428	483
b)	17.564	20.153	15.613
c)	22.660	26.046	22.152
d)	2.301	2.278	2.301
e)	2.091.678	1.691.858	1.052.726
f)	2.872.668	1.820.238	1.423.942
g)	5.558	5.548	5.558
h)	8.002.658	6.290.989	5.197.635
i)	53.028.113	28.502.411	24.144.396
j)	14.196	14.189	14.196
k)	133.913.541	123.910.850	77.734.851
l)	1.606.452.599	528.890.374	475.232.064

Tabla 2.1: Tamaños medios de los árboles de grupos.

k) El mismo que b), pero ahora el número de casos de cada nodo se selecciona con una distribución de Poisson de media 4 con un mínimo de 2.

l) El mismo que c), pero ahora el número de casos de cada nodo se selecciona con una distribución de Poisson de media 4 con un mínimo de 2.

El tamaño medio y la desviación típica del árbol de grupos obtenido con el uso de cada heurística se dan en las tablas 2.1 y 2.2.

De los anteriores experimentos podemos deducir las siguientes conclusiones:

- Las nuevas heurísticas, con la excepción de $\mathbf{H_3}$ y $\mathbf{H_4}$, obtienen árboles de grupos más pequeños que $\mathbf{H_1}$.

Método	H1	H2	H3
a)	64	64	54
b)	34.332	34.332	27.561
c)	39.452	39.452	38.911
d)	1.097	1.063	956
e)	7.423.401	5.080.951	8.543.176
f)	50.441.890	13.401.449	18.944.797
g)	2.630	2.546	2.188
h)	21.490.256	23.676.980	21.069.018
i)	705.106.339	288.762.073	466.932.003
j)	7.940	7.636	6.916
k)	344.446.921	451.214.399	360.803.452
l)	25.081.485.346	24.760.156.861	24.797.026.066
	H4	H5	H6
a)	71	54	64
b)	28.929	30.047	23.179
c)	31.731	39.188	32.630
d)	954	956	954
e)	8.367.031	7.670.390	3.276.971
f)	18.946.028	7.086.978	5.472.380
g)	2.187	2.188	2.187
h)	21.380.543	19.737.130	16.726.049
i)	466.839.764	201.002.001	209.095.702
j)	6.914	6.916	6.914
k)	380.197.254	500.892.991	570.893.175
l)	24.796.998.549	4.180.926.330	4.280.997.524

Tabla 2.2: Desviaciones típicas del tamaño de los árboles de grupos.

- La mejora relativa crece en función de la complejidad del grafo. El tamaño medio aplicando $\mathbf{H_1}$ dividido por el tamaño medio aplicando $\mathbf{H_6}$ es cercano a 1 en el caso de los casos más simples (tipo a). En el caso de grafos más complicados (tipo l) es mayor que 3.

- La mejor heurística es $\mathbf{H_6}$. Esta heurística tiene un mayor costo computacional que $\mathbf{H_1}$. Sin embargo, $\mathbf{H_2}$ tiene el mismo costo computacional que $\mathbf{H_1}$, y produce, en general, mejores resultados que $\mathbf{H_1}$ (algunas veces consigue árboles de tamaño dos veces menos).

- Cuando los padres de un nodo se eligen entre los nodos predecesores (casos a, d, g y j) los grafos son relativamente más simples y todos los algoritmos heurísticos producen resultados similares.

- Cuando los nodos más cercanos tienen una probabilidad más alta de ser elegidos como padres, (b, e, h, k), los grafos tienen un tamaño más pequeño que cuando los nodos tienen la misma probabilidad (c, f, i, l), pero obtenemos tamaños mayores si lo comparamos con los tamaños de los casos más simples (a, d, g, j).

Capítulo 3

Algoritmos de Propagación Basados en Técnicas de Optimización Combinatoria

3.1 Introducción

En el capítulo anterior se consideraron algoritmos exactos para la propagación de conjuntos convexos de probabilidades. Estos algoritmos sin embargo, no son realizables desde el punto de vista práctico en problemas de mediano tamaño. El problema fundamental es que si combinamos dos conjuntos convexos, el número de posibles puntos extremos es el producto de los puntos extremos de los convexos que se combinan. Además eliminar los puntos no extremos que aparecen también implica la aplicación de un algoritmo de cláusula convexa, que conllevan un importante costo computacional. Cano y Moral [21] consideraron el uso del algoritmo conocido como envolvente de regalos (*gift-wrapping*) [108, 44]. Sin embargo, este algoritmo sólo puede aplicarse en espacios con una dimensión no demasiado alta por las siguientes dos razones:

- La complejidad de aplicar el algoritmo de cápsula convexa a n puntos en un espacio de dimensión k es $O(n^{[k/2]}\log(n))$, donde $[k/2]$ es la parte entera de $k/2$. Por tanto el costo de aplicar este algoritmo crece mucho con la dimensión del espacio.

- El número de puntos no extremos decrece con la dimensión del espacio.

En esta situación, ha habido dos tendencias en la literatura: una de ellas consiste en resolver problemas menos generales que el que hemos planteado, y que garanticen que la solución se puede alcanzar en un tiempo razonable: idealmente de un orden comparable al de una propagación probabilística. En esta dirección podemos destacar el trabajo de Fagiuoli y Zaffalon [46], que resuelven el problema de forma exacta para el condicionamiento de Dempster [39], en poliárboles y con variables bivaluadas. La otra tendencia, mucho más importante, ha sido la de intentar resolver el problema de forma aproximada, y dentro de esta línea su resolución como un problema de *optimización combinatoria*. En Cano, Cano y Moral [13] se introdujo por primera vez esta visión del problema de propagación de probabilidades imprecisas como un problema de optimización, aplicando técnicas de enfriamiento simulado, posteriormente se han empleado otros procedimientos como algoritmos genéticos [15], o técnicas de gradiente descendente [32, 33].

En este capítulo analizaremos *métodos generales* de solución de problemas de optimización combinatoria, que hemos aplicado al problema de la propagación de conjuntos convexos cuando tenemos relaciones de independencia y que intentan buscar buenas aproximaciones al resultado exacto. Estos métodos se basarán en las técnicas de *enfriamiento simulado* y en las de *algoritmos genéticos*. Es importante remarcar que si el método de condicionamiento que se utiliza es el de Moral y Campos [98], el problema de optimización tiene dos objetivos (máxima probabilidad condicionada y máxima verosimilitud de la probabilidad dadas las observaciones), que usualmente no se pueden satisfacer de forma simultánea. Consideraremos entonces una adaptación de estas técnicas que tratarán de obtener una familia de puntos no dominados. Para este caso, los algoritmos genéticos, que trabajan con una familia de puntos, serán especialmente útiles.

Un aspecto muy importante será la evaluación de la función objetivo para un caso dado, que en el peor de los casos se podrá realizar mediante una propagación probabilística, pero que en muchas ocasiones, como en el caso del enfriamiento simulado, se propondrán métodos para hacerlo de forma mucho más eficiente.

El capítulo se estructura de la siguiente forma: en la sección 3.2 se introduce el problema de la propagación cómo un problema de optimización combinatoria, en las secciones 3.3 y 3.4 se introducen brevemente las técnicas de enfriamiento simulado y algoritmos genéticos. Las secciones 3.5 y 3.6 presentan la aplicación de estas técnicas a nuestro problema, incluyendo la realización de experimentos. Finalmente en la sección 3.7 se dan las conclusiones del capítulo.

3.2 Planteamiento del Problema

Supongamos un grafo dirigido acíclico en el que en cada nodo existe un conjunto convexo de distribuciones de probabilidad condicionadas a sus padres, H_i. Supongamos que el grafo representa relaciones de independencia fuertes, de tal manera que la información global para todas las variables se puede expresar como:

$$H = H_1 \otimes \ldots \otimes H_n \tag{3.1}$$

Supongamos que tenemos una familia de observaciones $e = (O_i)_{i \in I}$, y estamos interesados en la información 'a posteriori' para la variable X_j. Esta información 'a posteriori' puede tener, dependiendo de la situación, distintas modalidades:

- El convexo 'a posteriori' :

  $$PS_j = ((\otimes_{i \in I} O_i) \otimes (\otimes_{k \in \{1,\ldots,n\}} H_k))^{\downarrow \{j\}}$$

  Este conjunto convexo corresponde al condicionamiento de Moral y Campos [98], marginalizado en la variable de interés.

  A partir de este convexo se pueden calcular los intervalos de probabilidad para todos los valores de la variable X_j usando el procedimiento explicado en la sección 1.6.

- Los intervalos 'a posteriori' para la variable X_j calculados de acuerdo con el condicionamiento de Dempster. Estos se podrían obtener a partir del convexo

  $$(H^j|_2 e) = ((H_1 \otimes \ldots \otimes H_n)|_2 e)^{\downarrow j}$$

  Sin embargo, en la mayoría de las ocasiones se calcularán directamente.

Para resolver este problema realizamos la misma transformación que en el capítulo anterior (sección 2.5.2) asociando a cada variable X_i, una variable transparente T_i. Si el convexo de X_i condicionado a sus padres, H_i, tiene l puntos extremos, entonces T_i tendrá l casos, uno para cada punto extremo: $\{c_1, \ldots, c_l\}$. Este nodo será un padre de la variable X_i. Ahora la probabilidad de X_i dados sus padres será una única y bien determinada distribución de probabilidad. Si $pa(i)$ son los padres originales del nodo X_i, entonces la probabilidad condicionada

de X_i dados $pa(i) \cup T_i$ se determina de la siguiente forma: si $T_i = c_k$ entonces la probabilidad condicional de X_i dados $pa(i)$ es h_k, el k-ésimo punto extremo de H_k.

Suponemos que cada nodo transparente tiene como información 'a priori' la ignorancia: todas las distribuciones de probabilidad posibles. Puede comprobarse que la estructura del problema no cambia, lo único que hemos hecho es hacer explícito la imprecisión en nuestro conocimiento de las probabilidades condicionadas.

Un punto importante sobre los nodos transparentes es que si le asignamos un valor a cada uno de ellos obtenemos una única distribución de probabilidad. Si tenemos $c \equiv (T_1 = c_1, \ldots, T_n = c_n)$, y P_c es la distribución de probabilidad asociada a esta configuración, entonces P_c es un punto de $H = H_1 \otimes \ldots \otimes H_n$ y, por tanto, $f_c(x) = p_c(x \cap e)$ un punto de $H|_1 e$ y $f_c^{\downarrow j}$ un punto de PS_j. Variando c de manera que las variables transparentes tomen todos los posibles valores, obtendríamos todos los puntos extremos de PS_j (quizás algunos no extremos también). Lo importante es que $f_c^{\downarrow j} = P_c^{\downarrow j}(. \cap e)$ puede calcularse mediante una propagación probabilística simple.

Sea C_i el conjunto de valores posibles de T_i y $C = C_1 \times \ldots \times C_n$. El problema de calcular los límites de los intervalos de acuerdo con el condicionamiento de Dempster se puede expresar cómo obtener:

$$P_*(u_j|e) = \inf_{c \in C} P_c^{\downarrow j}(.|e)$$

para el límite inferior, y

$$P^*(u_j|e) = \sup_{c \in C} P_c^{\downarrow j}(.|e)$$

para el límite superior.

Esta representación del cálculo de los extremos intervalares como problemas de optimización combinatoria permitirá el uso de técnicas clásicas como el enfriamiento estocástico y los algoritmos genéticos.

Para los intervalos que provienen del condicionamiento de Moral y Campos será necesario el cálculo del convexo PS_j, o más concretamente de todos los puntos que tengan una influencia directa en el valor de los intervalos calculados. Si estamos calculando el extremo inferior para $u_j \in U_j$, entonces nos interesarán puntos $c \in C$ que tengan un valor lo más pequeño posible de $P_c(u_j|e)$ y, al mismo tiempo el valor lo más alto posible de $P_c(e)$. Todos los puntos que no sean dominados en relación con estos dos objetivos influirán en el cálculo final. Se convierte

así el problema en uno de optimización combinatoria multiobjetivo al que habrá que adaptar los algoritmos aproximados habituales. Análogamente, para otros valores $u_j \in U_J$ y para los límites superiores.

3.3 Enfriamiento simulado

3.3.1 Introducción

El método del enfriamiento simulado (también conocido como enfriamiento estocástico, temple simulado o recocido simulado), más conocido por su nombre en inglés *Simulated Annealing*, fue introducido por Kirkpatrick y otros [80]. El enfriamiento simulado está basado en el algoritmo de Metropolis [95] que es un algoritmo de *Monte Carlo* para simular la evolución de un sólido hasta equilibrio termal, desde un determinado valor de temperatura. Es una técnica que ha tenido una significativa atención como método general para resolver problemas de optimización combinatoria de gran escala, especialmente aquellos donde la solución global del problema está oculta entre muchas, más pobres, soluciones óptimas locales. Es una técnica de optimización potente, general y de amplia aplicación en diversos ámbitos de la Inteligencia Artificial. Sorprendentemente, la implementación del algoritmo es relativamente simple.

La técnica del enfriamiento simulado incorpora algunos aspectos de las técnicas de *mejora iterativa*, también conocidos como algoritmos de *búsqueda local* o de *ascensión de colinas*. En ellas también se presupone la definición de configuraciones, una función de costo y un mecanismo de generación de configuraciones. En los algoritmos de mejora iterativa se comienza con una configuración inicial, desde la cual se pasa a la configuración vecina que optimice la función de costo (o sea, que tenga menor valor para la función de costo). El algoritmo termina cuando se obtiene una configuración cuyo costo es menor que todos sus vecinos. El enfriamiento simulado intenta resolver el problema básico de los algoritmos de mejora iterativa, que consiste en que encuentran mínimos locales los cuales no se sabe cuanto se han desviado del mínimo global. Con la técnica de ascensión de colinas se alcanza rápidamente una solución. Pero esta solución no será probablemente la solución mínima global, sino que será un óptimo local. Además los mínimos locales encontrados por los algoritmos de mejora iterativa dependen de la configuración inicial elegida. El enfriamiento simulado resuelve estos problemas aceptando incrementos en la función de costo en determinados momentos (en los algoritmos de mejora iterativa sólo se aceptan decrementos de la función de costo). Las soluciones obtenidas por el

enfriamiento simulado no dependen de la configuración inicial.

3.3.2 Algoritmo de enfriamiento simulado

Sea un problema de optimización combinatoria con una función de costo $C : \mathcal{S} \rightarrow I\!R^+$, definida en algún conjunto finito (espacio de búsqueda) $\mathcal{S}$, donde cada $s \in \mathcal{S}$ es una solución o configuración del problema. El objetivo del algoritmo de enfriamiento simulado va a ser encontrar aquella configuración s que minimice la función de costo C. Para cada configuración $s \in \mathcal{S}$, existe un conjunto de configuraciones vecinas $\mathcal{N}(s) \subseteq \mathcal{S}$ que se pueden generar con una pequeña perturbación de s.

En el algoritmo tendremos que la probabilidad de hacer una transición desde un estado $s(k)$ a otro estado $s'(k)$ está controlado por el *criterio de Metropolis* [95]. Según este criterio cuando el nuevo estado $s'(k)$ es mejor al anterior $s(k)$, entonces se pasa con probabilidad 1 al estado $s'(k)$, pero cuando $s'(k)$ es peor a $s(k)$ entonces se pasará al estado $s'(k)$ con una probabilidad dada con la siguiente expresión:

$$P[s(k), s'(k)] = Pr\{s(k) \rightarrow s'(k)\} = e^{[-\frac{C(s'(k))-C(s(k))}{t}]} \tag{3.2}$$

donde $s'(k)$ se genera a partir de $\mathcal{N}(s)$ con un mecanismo de perturbación
k denota la k-ésima iteración, y
t denota la temperatura actual

O sea, la probabilidad de pasar desde un estado $s(k)$ a un estado $s'(k)$ a una temperatura t viene dado por la siguiente expresión (criterio de Metropolis):

$$Pr\{s(k+1) = s'(k)\} = \begin{cases} 1 & si\ C(s'(k)) < C(s(k)) \\ e^{(-\frac{C(s'(k))-C(s(k))}{t})} & en\ otro\ caso \end{cases} \tag{3.3}$$

Podemos ver a partir del criterio de Metropolis (ecuación 3.3) que a una determinada temperatura t el algoritmo del enfriamiento simulado se comporta como un algoritmo de búsqueda local permitiendo que la búsqueda pase de configuraciones con un costo bajo a configuraciones con costo más alto con cierta probabilidad, previniendo de esta forma que la búsqueda quede atrapada en mínimos locales. Al principio de la ejecución del algoritmo la temperatura t será alta y por tanto también la posibilidad de aceptar configuraciones "peores" a la actual. Durante la ejecución del algoritmo se hace disminuir el parámetro t, con lo que también disminuye la posibilidad de aceptar el paso a una configuración "peor".

El algoritmo del enfriamiento simulado puede ser visto como un algoritmo que continuamente intenta transformar la configuración actual en una de sus vecinas. Este mecanismo se puede describir matemáticamente a través de una *cadena de Markov* [1]: una secuencia de simulaciones, donde el resultado de cada simulación depende sólo del resultado de la anterior. En el caso del enfriamiento simulado, las simulaciones corresponden a las transiciones y está claro que el resultado de una simulación depende sólo del resultado de la simulación previa (o sea, de la actual configuración), con lo cual la cadena de Markov que resulta es homogénea a temperatura constante. Esta comparación del Enfriamiento Simulado con una cadena de Markov permite comprobar matemáticamente la convergencia del método [1, 57, 58, 63] a las configuraciones con mínimo costo. Si la temperatura es decrementada lentamente y en cada temperatura se permite el suficiente número de transiciones, entonces las configuraciones (soluciones) con costo global mínimo serán encontradas con probabilidad 1. Evidentemente, no se pueden efectuar un número infinito de iteraciones en un ordenador. Por ello hay que realizar aproximaciones que permitan ejecutar el algoritmo en tiempo finito. Al hacer esto perdemos la garantía teórica de encontrar la solución óptima, pero en la mayoría de los casos la solución que nos proporciona el algoritmo será casi óptima.

Resumiendo, para usar un algoritmo de enfriamiento simulado en problemas de optimización combinatoria debemos proporcionar los siguientes elementos:

- Una descripción de las posibles configuraciones del sistema $s \in \mathcal{S}$.
- Un generador aleatorio de cambios en la configuración. O sea, un mecanismo para generar una *transición* desde una configuración a otra. Este mecanismo de generación define una vecindad $\mathcal{N}(s)$ para cada configuración s, consistente de todas las configuraciones que pueden alcanzarse desde s en una transición.
- Una función objetivo C cuya minimización es el objetivo del algoritmo.
- Un parámetro de control t (temperatura) que tendrá un valor inicial t_0 suficientemente alto, y un *esquema de enfriamiento*, que puede representarse con una función $f(t)$, que especifica como pasar de alta a baja temperatura, y que debe hacer bajar la temperatura lentamente. Hace falta decidir el valor inicial de la temperatura t_0 y el valor final.
- Un criterio para determinar el número de transiciones de estado por cada temperatura.

A continuación podemos ver de forma esquemática el algoritmo del enfriamiento simulado:

Algoritmo 3.1 Esquema básico del enfriamiento simulado

1. Inicializar la temperatura t
2. Introducir el número de iteraciones para cada temperatura, n.
3. Seleccionar aleatoriamente una configuración s de $\mathcal{S}$
4. Repetir hasta condición de parada
   - Para $i = 1$ hasta n
     Seleccionar $s' \in N(s)$
     Si $C(s') \leq C(s)$
       Entonces $s = s'$
       Sino $s = s'$ con probabilidad $e^{(-\frac{C(s')-C(s)}{t})}$
   - $t = f(t)$
5. Devolver como solución la mejor configuración s visitada

■

3.4 Algoritmos genéticos

Los algoritmos genéticos [96] han sido utilizados en los últimos años para la resolución de problemas de optimización combinatoria al igual que el algoritmo del enfriamiento simulado. En los algoritmos genéticos, la idea subyacente es la de aplicar los principios en los que se basan los procesos genéticos de los organismos biológicos y los procesos de la evolución natural como reglas en el diseño de sistemas artificiales para la resolución de problemas. La terminología usada en los algoritmos genéticos se toma de la *evolución natural* o *genética*. Aunque en un principio la representación del problema se realizaba siempre mediante cadenas de bits, hoy en día está aceptado que se use cualquier tipo de estructura de datos apropiada para el problema dado y un conjunto de operadores genéticos apropiados para la estructura de datos elegida.

En un modelo estándar de evolución tenemos las fases descritas en el algoritmo 3.2.

Algoritmo 3.2 Estructura de un algoritmo genético

1. Inicialización de la población de posibles soluciones P(t) (inicialmente t=0).
2. Evaluar la población inicial P(t) según alguna función de evaluación.
3. Mientras no se de la condición de parada hacer:
   (a) t=t+1
   (b) Seleccionar una nueva población P(t) desde P(t-1).
   (c) Recombinar P(t) mediante operadores como cruce y mutación.
   (d) Evaluar la población P(t).

■

Analicemos brevemente los elementos que intervienen en un algoritmo genético:

- *Población*: Conjunto de individuos o *cromosomas*, cada uno de los cuales define una posible solución al problema. $P(t)$ representará la población que se tiene en la iteración t del algoritmo. Cada cromosoma tendrá asociada una adaptación que describe la adecuación (conocido en inglés con *fitness*) de la correspondiente solución que representa. Un cromosoma se representará como una cadena de genes $\{c_1, c_2, \ldots, c_n\}$ en la que cada $c_i \in \Omega_i$, siendo Ω_i el conjunto de posibles valores que puede tomar el gen c_i. La figura 3.1 nos muestra esta representación de un cromosoma como una cadena de genes.

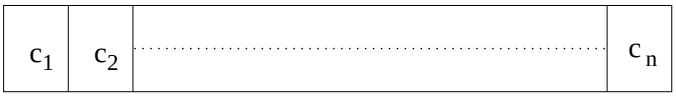


Figura 3.1: Representación de un cromosoma mediante una cadena de genes

La población inicial se suele generar de forma aleatoria, y mediante el algoritmo genético va a evolucionar, a través de un proceso de competición y variación controlada, hacia una población que contenga mejores individuos, o sea, que estén más cerca del individuo óptimo de la población.

- *Función de adaptación*: En cada generación del algoritmo evolutivo las soluciones relativamente buenas se reproducen, y las soluciones relativamente malas mueren para ser

reemplazadas por otras con las características de buenas soluciones. Para distinguir entre las soluciones se utiliza una función de evaluación o función objetivo f que permita distinguir las soluciones relativamente buenas de las malas para el problema tratado. En cada iteración del algoritmo genético se evalúan los individuos de la población para ver cual es su factor de adecuación a la solución global.

- *Selección*: En la naturaleza, los individuos más fuertes de una especie suelen sobrevivir con mayor posibilidad que los más débiles. Usando esta noción de la adecuación (fitness), el proceso de selección genera una nueva población *aleatoriamente* seleccionando cromosomas (individuos) de la población anterior. La selección no será totalmente aleatoria, sino que estará basada en la adecuación relativa del cromosoma con respecto a los valores de adecuación de la población completa.

  Los individuos de una población no tienen que ser únicos. La proporción $f_i / \sum_i f_i$ se considera como la probabilidad de que el individuo i sobreviva a la siguiente generación. Eso significa que los individuos con un valor más alto de valor de adecuación, f_i tendrán una probabilidad más alta de sobrevivir, y de tener más copias en la nueva población. Además, el número de individuos en una población se mantiene como constante durante toda la ejecución del algoritmo, y el operador de selección generará siempre una nueva población del mismo tamaño.

  Considerando una población actual P con n cromosomas $C_1, \ldots, C_n$, el mecanismo de selección generará una nueva población P' que tendrá también n cromosomas. La selección consistirá entonces de los dos siguientes pasos:

  1. Calcular para cada cromosoma C_i de la población actual P, la probabilidad $p_s(C_i)$, de que este cromosoma pase a la siguiente generación.

     $$p_s(C_i) = \frac{f(C_i)}{\sum_{j=1}^{n} f(C_j)} \tag{3.4}$$

  2. Calcular la probabilidad acumulada para cada cromosoma C_i mediante la siguiente expresión:

     $$p_{acum}(C_i) = \sum_{j=1}^{i} p_s(C_j)$$

  3. Seleccionar n cromosomas de la actual población de la siguiente forma:

     - Generar un número aleatorio uniforme $r \in [0, 1]$

- Si $r < p_{acum}(C_1)$ seleccionar C_1 para la nueva población. Si no seleccionar C_i tal que $p_{acum}(C_{i-1}) < r \leq p_{acum}(C_i)$

Claramente puede verse que si el mecanismo de selección se aplica de forma repetida, entonces los individuos con alto valor de adecuación dominarán la población completamente. También podemos observar que este operador no obtiene individuos nuevos en la nueva población. O sea todos los individuos de la nueva población ya estaban en la antigua.

En el mecanismo de selección a veces se utiliza un mecanismo complementario conocido como *elitismo* que consiste en que el mejor elemento C_m de una población en la iteración t estará presente en la iteración $t+1$. Esto a veces resulta conveniente, pues es posible que en la iteración t el mejor individuo desaparezca debido a la aplicación del mecanismo de selección, o debido a la aplicación de los mecanismos de cruce y mutación.

- *Operadores de recombinación*: El procedimiento para alterar la población consiste en aplicar los operadores genéticos a la población seleccionada. Los dos operadores genéticos básicos son:

  - *Cruce*: El operador de cruce usa el concepto de *emparejamiento* donde los genes de cada uno de individuos (cromosomas) diferentes se mezclan para producir nuevos individuos. Como los genes representan las características de un individuo, mezclando *buenas* características de individuos diferentes, obtendremos individuos aun mejores que sus progenitores.

    Suponiendo que los genes de un cromosoma se ordenan en una línea recta, entonces el cruce de dos cromosomas puede llevarse a cabo en las siguientes etapas mediante el denominado *cruce simple*:

    1. Seleccionar de forma aleatoria un punto para romper cada cromosoma en dos partes.
    2. Intercambiar las partes de más a la izquierda de cada cromosoma.

    En la figura 3.2 podemos ver un ejemplo de aplicación de este operador a los cromosomas A y B.

    De esta forma se obtienen dos nuevos cromosomas a partir de dos cromosomas progenitores. Estos nuevos cromosomas tendrán genes de ambos progenitores. Este

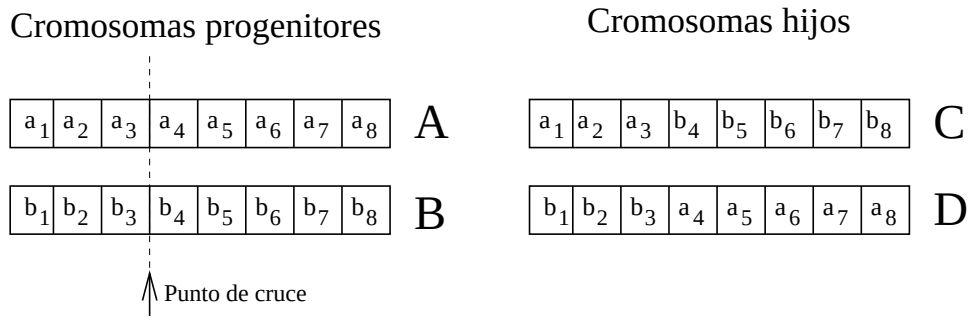


Figura 3.2: Aplicación del operador de cruce a los cromosomas A y B, obteniendo los cromosomas C y D

operador de cruce, aunque es simple de llevar a cabo, es sin embargo una forma bastante potente de llevar a cabo el intercambio de información y de crear nuevas soluciones. Sin embargo, un pequeño problema que podría plantearse es tener una población compuesta de copias del mismo individuo. En tal caso el cruce de dos individuos siempre produce individuos iguales. Una forma de escapar de tal situación es utilizar el operador de mutación.

Este operador necesita un parámetro de control, denominado *probabilidad de cruce*, p_c, que determina la probabilidad de aplicar el operador de cruce sobre parejas de cromosomas de la población intermedia.

– *Mutación*: El operador de mutación selecciona de forma aleatoria uno o más genes en un cromosoma, y lo (los) sustituye por un alelo diferente, creando un nuevo cromosoma. Este operador introduce variabilidad en la población, y proporciona un mecanismo para escapar de los mínimos locales. Se consigue así restaurar material genético perdido o no explorado en la población. Puede verse claramente que un algoritmo que usase sólo este operador equivaldría a un algoritmo de búsqueda aleatoria. En la figura 3.3 podemos ver un ejemplo de aplicación de la mutación donde se mutan dos genes de un cromosoma.

Cada gen en cada cromosoma sufrirá una mutación de acuerdo a un parámetro de control, denominado la *probabilidad de mutación*, p_m.

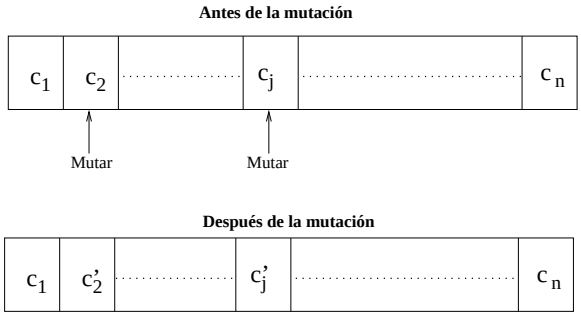


Figura 3.3: Aplicación del operador de mutación, para mutar los genes c_2 y c_j

3.5 Algoritmos de enfriamiento simulado aplicados a la propagación de conjuntos convexos

Cano y Moral [21] dieron un primer algoritmo de Monte Carlo para aproximar el conjunto convexo 'a posteriori' PS_j de una variable dada X_j. Este algoritmo hacía la transformación del problema original en un problema equivalente que fue explicada en la sección 3.2, mediante la adición de un nodo transparente por cada conjunto convexo.

En esta sección asumiremos también que se realiza la misma transformación mencionada más arriba. Haremos uso del esquema propuesto por el algoritmo exacto de propagación en un árbol de grupos. Pero en este caso no será necesario tener valuaciones tan enormes como lo era en el algoritmo de propagación de conjuntos convexos explicado en el capítulo anterior. Ahora cada variable transparente sólo aparecerá en un grupo, con lo que el tamaño del árbol de grupos es mucho menor, y por tanto el tiempo de cada propagación será también mucho menor. Seleccionando un valor para cada una de las variables transparentes del árbol de grupos se tiene determinado un punto del conjunto convexo 'a posteriori' de cada una de las variables de interés.

En esta sección presentaremos varios algoritmos que hemos desarrollado que seleccionan de forma aleatoria un valor para cada nodo transparente, llevando a cabo una propagación probabilística a continuación. De esta forma se puede obtener un vector 'a posteriori' para todas las variables del problema. Repitiendo este experimento muchas veces, podemos obtener aquellos vectores que correspondan a los puntos extremos del conjunto convexo 'a posteriori', o al menos podemos llegar a obtener puntos que estén muy próximos a los puntos extremos, obteniendo así una aproximación del conjunto convexo 'a posteriori'.

En el capítulo 2 para aplicar el algoritmo exacto de propagación de conjuntos convexos, se construía un árbol de grupos en el que cada grupo tendría todas las variables transparentes del problema. En los algoritmos que estudiaremos en esta sección también se construirá un árbol de grupos, pero ahora cada variable transparente estará nada más que en un grupo del árbol. Para obtener este árbol de grupos podemos proceder de la misma forma que en el caso probabilístico. Primero se construye el grafo moral, luego se triangula el grafo moral y finalmente se obtiene el árbol de grupos con el algoritmo de máxima cardinalidad. En este caso sólo cabe comentar que para triangular el grafo moral podemos eliminar primero las variables transparentes del grafo, con lo que no se introducen nuevos arcos en el grafo triangulado, y posteriormente eliminar el resto de las variables según algún criterio más o menos óptimo. Para la obtención del árbol de grupos se procederá de la misma forma que en el caso probabilístico, usando el grafo triangulado. Finalmente habría que asignar cada valuación condicional a uno de los grupos que contengan todas sus variables. Como hemos visto en el capítulo 2, para las variables transparentes no se conoce nada acerca de ellas, o lo que es lo mismo, que todas las distribuciones de probabilidad sobre ese nodo son posibles. Es como si cada nodo transparente T_i tuviese un conjunto convexo con k puntos extremos, cada uno de ellos degenerado en uno de los posibles casos de T_i.

Por ejemplo, para el grafo de dependencias de la figura 2.12, donde ya se ha hecho la trasformación de incluir las variables transparentes podemos tener el grafo moral de la figura 3.4 que en este caso también está triangulado. A partir de este grafo triangulado podemos obtener el árbol de grupos de la figura 3.5. Comparando este nuevo árbol con el que usábamos en la figura 2.13 para el algoritmo exacto, podemos ver que los grupos tienen un número mucho menor de variables.

Básicamente los pasos para construir el árbol de grupos nos los da el siguiente algoritmo:

Algoritmo 3.3 Construcción de un árbol de grupos para los algoritmos de enfriamiento simulado

1. Construcción del grafo moral.

2. Triangulación del grafo moral siguiendo una secuencia de eliminación de variables. Eliminaremos primero las variables transparentes, pues de esta forma conseguimos que no se añadan arcos nuevos al grafo, y además que las variables transparentes sólo aparezcan en un grupo. El algoritmo de triangulación permite al mismo tiempo obtener los grupos maximales.

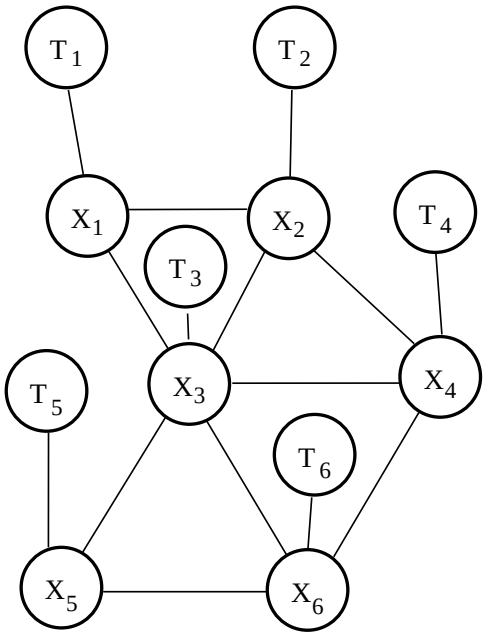


Figura 3.4: Grafo moral y triangulado para los algoritmos aproximados

3. Ordenar los grupos maximales utilizando el algoritmo de *máxima cardinalidad*.

4. Utilizar el orden anterior de los grupos para darle estructura al árbol de cliques. Con este orden se consigue que el árbol cumpla la *propiedad de la intersección*: Para cada par de grupos C_i y C_j entonces se cumple que las variables comunes que hay en ellos, $C_i \cap C_j$, están presentes en todos los grupos que hay en el único camino que va de C_i hasta C_j.

5. Asignar cada conjunto convexo condicional a uno de los grupos que contenga todas las variables para las que está definido este conjunto convexo (sólo habrá un grupo que cumpla tal condición debido a que la variable transparente que tiene el conjunto convexo condicional sólo va a estar en un grupo del árbol).

■

Cuando hacemos uso del condicionamiento de Moral y Campos (definición 1.6), es importante volver a remarcar (como ya hicimos en la sección 1.6 y más tarde en la sección 2.5.2) que no podemos normalizar los vectores 'a posteriori' ya que el factor de normalización de cada punto del conjunto convexo 'a posteriori' es necesario a la hora de hacer el cálculo para dar un intervalo para la variable correspondiente.

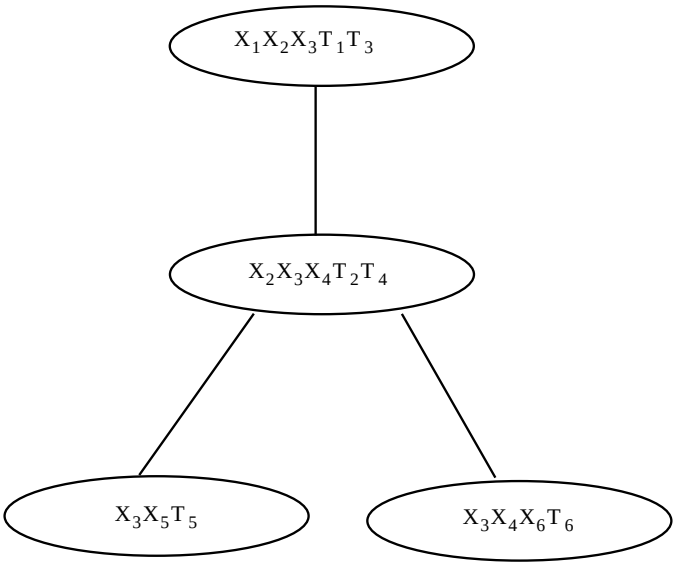


Figura 3.5: Un árbol de grupos para los algoritmos aproximados

Sea una variable X_j que toma sus valores en el conjunto finito U_j, y sea $p = (p(u_j^1), \ldots, p(u_j^{|U_j|})$ un punto perteneciente al conjunto convexo 'a posteriori' de la variable X_j (punto que no está normalizado). Como vimos en la sección 1.6 el factor de normalización $r = \sum_{u_j^i \in U_j} p(u_j^i)$ de un punto p permite obtener un valor de posibilidad $\pi(p)$ proporcional al factor de normalización. Este valor de posibilidad nos puede dar una idea de la verosimilitud del punto a la hora de asociar intervalos a un conjunto convexo 'a posteriori'. Puntos con alto valor de posibilidad serán más importantes que los puntos con valor bajo. En un algoritmo exacto de propagación de conjuntos convexos se obtenían todos los puntos extremos del conjunto convexo 'a posteriori', pero en un algoritmo aproximado no se obtendrán todos, pero interesará que los que se encuentren, sean aquellos que tengan alta posibilidad, para que así el error al calcular los intervalos sea menor.

Por cuestiones de simplicidad en la nomenclatura supongamos también que la variable X_j toma sólo dos posibles valores, con lo cual $U_j = \{u_j^1, u_j^2\}$. Entonces el resultado de la propagación es un conjunto convexo sobre $\mathbb{R}^2$ de la forma de la figura 2.9 como ya explicamos en el capítulo 2. En la sección 2.5.2 indicábamos que, si $P_{c_{i_1},\ldots,c_{i_n}}$ es la distribución de probabilidad global, obtenida seleccionando un caso c_{i_k} para cada nodo transparente T_k , entonces el conjunto convexo 'a posteriori' PS_j para X_j contendrá el punto (t_1, t_2) donde,

$$\begin{aligned} t_1 = &\quad P_{c_{i_1},\ldots,c_{i_n}}(u_j^1 \cap e) \\ t_2 = &\quad P_{c_{i_1},\ldots,c_{i_n}}(u_j^2 \cap e) \end{aligned} \tag{3.5}$$

y e es la evidencia dada, o familia de observaciones $\{O_i\}_{i\in I}$

El punto (t_1, t_2) del conjunto convexo 'a posteriori' PS_j de la variable X_j puede obtenerse a partir del árbol de grupos como se indicó en la sección 2.3.2.5 donde se explicaba la propagación probabilística en un árbol de grupos, una vez que éste está en equilibrio mediante el uso de la expresión 2.12 en la siguiente forma:

$$\begin{aligned} t_1 = &\quad PS_j(u_j^1) = (\bigotimes_{m=1,\ldots,l} V_{C_{j_m}\to C_i} \otimes V_{C_i})^{\downarrow\{j\}}(u_j^1) \\ t_2 = &\quad PS_j(u_j^2) = (\bigotimes_{m=1,\ldots,l} V_{C_{j_m}\to C_i} \otimes V_{C_i})^{\downarrow\{j\}}(u_j^2) \end{aligned} \tag{3.6}$$

A continuación presentamos algunos de los puntos del conjunto convexo que se consideran importantes para calcular los intervalos asociados al conjunto convexo 'a posteriori' de X_j.

- Puntos con alto valor de $r = t_1 + t_2$. O sea, puntos con alta posibilidad.
- Puntos con alto valor de t_1 y puntos con alto valor de t_2.
- Puntos con alto valor de $\frac{t_1}{t_1+t_2}$.
- Puntos con alto valor de $\frac{t_2}{t_1+t_2}$.

En el caso del condicionamiento de Dempster, sólo los puntos de alto valor de $\frac{t_1}{t_1+t_2}$ y $\frac{t_2}{t_1+t_2}$ son necesarios.

Hay otros puntos intermedios que también son importantes para que los intervalos asociados al conjunto convexo 'a posteriori' sean lo más exactos posibles. Por ejemplo, un punto con un valor intermedio de $\frac{t_1}{t_1+t_2}$ y un valor intermedio de $t_1 + t_2$. En esta sección expondremos distintos métodos aproximados que intentan encontrar los puntos del conjunto convexo 'a posteriori' que cumplen alguna de las buenas características expuestas más arriba. Los puntos que encuentran los algoritmos pertenecerán al conjunto convexo 'a posteriori', aunque es posible que tales puntos no sean extremos del convexo 'a posteriori'.

Los métodos que expondremos en esta sección instancian todas las variables transparentes del árbol de grupos. De esta forma al realizar una propagación probabilística se puede obtener un punto del conjunto convexo 'a posteriori' de las variables de interés. Un método exacto que podría aplicarse según este esquema, es instanciar las variables para todas las posibles

combinaciones posibles y realizando la correspondiente propagación obtendríamos todos los puntos del conjunto convexo 'a posteriori'. Esto es inmanejable debido al gran número de propagaciones que tendríamos que llevar a cabo.

En los algoritmos que veremos se usa el método de propagación de probabilidades de Shafer y Shenoy, el cual fue explicado en la sección 2.3.2.5. Recordemos que en este método se construía un árbol de grupos. En cada grupo C_i se mantenía una valuación V_{C_i} y entre cada par de grupos adyacentes C_i y C_j se mantenían dos valuaciones que corresponden a los dos mensajes $M_{C_i \to C_j}$ y $M_{C_j \to C_i}$. La idea fundamental de los algoritmos será que si tenemos una configuración $c \in C$ de nodos transparentes y nos movemos a una configuración $c' \in C$ similar a c, entonces sólo una parte de la propagación tendrá que ser recalculada aprovechando gran parte de los cálculos de c. Incluso, podemos elegir los cambios posibles (cambiar sólo el valor de una transparente que esté en el mismo clique o uno conectado al clique donde estuviese la última variable modificada) de tal manera que, a lo sumo, sólo haya que mandar un mensaje para rehacer los cálculos.

3.5.1 Algoritmo de búsqueda de puntos con alto factor de normalización

En el primer algoritmo se intentan buscar puntos con alto valor de $t_1 + t_2$, o sea puntos con alta posibilidad.

Sean $H_1, \ldots, H_n$ los conjuntos convexos iniciales donde cada H_i representa una información condicional de una variable X_i dados sus padres $pa(X_i)$ en el grafo de dependencias. Cada H_i vendrá representado por una valuación V_i que estará definida para el conjunto de variables $\{X_i, pa(X_i), T_i\}$ donde T_i es una variable transparente que tiene tantos casos como puntos extremos tenga el conjunto convexo H_i, según se explicó en la sección 2.5.2. Sobre los nodos transparentes consideraremos que tenemos definida una distribución de probabilidad uniforme representada por una valuación V_{T_i} con una única variable, la variable transparente T_i.

El algoritmo que construimos en esta sección hará una propagación probabilística inicial en donde se consideran las variables observadas, así como una distribución uniforme sobre las variables transparentes que permite obtener una *configuración inicial* para los potenciales de los grupos y los mensajes entre grupos. Luego, se realizan el número de iteraciones deseado comenzando cada vez con esta configuración inicial. En cada iteración se hace un recorrido del árbol de grupos donde se va simulando un valor para cada una de las variables transparentes, insertando dicho valor como una observación más en el árbol de grupos. De esta forma, cuando

todas las variables transparentes están simuladas tenemos determinado un punto del conjunto convexo 'a posteriori' de las variables de interés, que puede obtenerse encontrando un grupo en el que esté incluida la variable de interés y aplicando la expresión 2.12.

Los pasos básicos de este algoritmo los podemos ver a continuación.

Algoritmo 3.4 Algoritmo de búsqueda de puntos con alto factor de normalización

1. Construir el árbol de grupos para el grafo de dependencias según hemos explicado anteriormente.

2. Sea V_{C_i} la valuación del grupo C_i inicialmente con todos los valores a 1.

3. Asignar cada valuación inicial V_i, cada V_{T_i} y cada valuación observación O_i a un grupo que contenga todas sus variables.

4. En cada grupo, combinar todas las valuaciones asignadas a ese grupo en la valuación V_{C_i}.

5. Llevar a cabo una propagación probabilística, dejando todos los nodos en equilibrio, en la forma que se explicó en la sección 2.3.2.5. Obtenemos así una valuación V_{C_i} actualizada en cada grupo y dos valuaciones $M_{C_i \to C_j}$ y $M_{C_j \to C_i}$ para los mensajes entre cada par de grupos adyacentes.

6. Para $i = 0$ hasta N (número de iteraciones deseado)

   - Considerar en cada grupo del árbol las valuaciones V_{C_i} de los grupos y los pares de mensajes $M_{C_i \to C_j}$ y $M_{C_j \to C_i}$ entre grupos adyacentes, que fueron obtenidos antes de este bucle.
   - Seleccionar un grupo raíz, C_0
   - Llamar a **Select-trans**(C_0,C_0)
   - Llevar a cabo una propagación probabilística, dejando todos los nodos en equilibrio.
   - Para cada variable X_j calcular el vector PS_j, resultante de la propagación probabilística, usando la expresión 2.12.

■

La función **Select-trans**(C_1,C_2) es una función recursiva definida a continuación. que básicamente se encarga de hacer un recorrido del árbol de grupos simulando un valor para cada variable transparente y actualizando los potenciales de los grupos y los mensajes entre ellos para tener en cuenta dichos valores simulados:

Algoritmo 3.5 Select-trans(C_1,C_2)

1. Para todas las variables transparentes T_{1_i} de C_1
   - Calcular el vector 'a posteriori' $PS_{T_{1_i}}$ de la variable transparente T_{1_i} con la fórmula 2.12.
   - Simular un valor para el nodo transparente usando el vector 'a posteriori' $PS_{T_{1_i}}$.
   - Añadir al grupo C_1 una observación correspondiente al valor simulado para el nodo transparente anterior.
2. Para todos los grupos vecinos, C', de C_1 excepto C_2.
   - Enviar un mensaje desde C_1 hasta C' calculado mediante la expresión 2.11.
   - Llamar a **Select-trans**(C',C_1)
3. Enviar un mensaje desde C_1 hasta C_2 calculado mediante la expresión 2.11.

■

En el anterior algoritmo cuando se simula cualquier variable transparente T_i con m_i casos $\{c_1, \ldots, c_{m_i}\}$ usamos el vector 'a posteriori' PS_{T_i} de esta variable. Este vector contiene un número real por cada caso c_i de la variable transparente T_i. Cada uno de estos valores representa la probabilidad de la evidencia $P(e)$ asociada a la distribución de probabilidad que se obtiene fijando la transparente T_i con el caso c_i. Así pues, PS_{T_i} representa un vector de distintos valores $P(e)$. Para simular con el vector PS_{T_i} hacemos lo siguiente: Generamos un número uniforme u entre 0 y 1 y entonces seleccionamos el caso c_k de la variable transparente si se cumple la siguiente condición:

$$\sum_{l=1}^{k} PS_{T_i}(c_l) \leq u < \sum_{l=1}^{k+1} PS_{T_i}(c_l) \tag{3.7}$$

De esta forma el mecanismo anterior (expresión 3.7) simula más frecuentemente casos de T_i correspondientes a valores altos de $P(e)$, con lo que vemos que el objetivo de este algoritmo es maximizar la probabilidad de la evidencia $P(e)$, que según hemos visto anteriormente es el factor de normalización de la distribución de probabilidad 'a posteriori'.

El costo de este algoritmo es equivalente a $1/2$ propagaciones probabilísticas por cada iteración del algoritmo.

3.5.2 Algoritmo de búsqueda de puntos con alto factor de normalización por muestreo de Gibbs

El segundo algoritmo es una modificación del anterior de acuerdo a las ideas de simulación estocástica dadas por Pearl para la propagación probabilística [106]. Este algoritmo es un caso particular del muestreo de Gibbs [57, 70] con una temperatura constante. También selecciona puntos con una probabilidad proporcional a $t_1 + t_2$, o sea al factor de normalización del punto obtenido del conjunto convexo 'a posteriori', pero en este caso las muestras no van a ser independientes como en el algoritmo anterior. O sea, no se comenzará en cada iteración con el resultado de la propagación inicial de la distribución uniforme de los nodos transparentes como hacíamos en el algoritmo de la sección anterior.

Para este segundo algoritmo haremos las mismas consideraciones que con el primer algoritmo. Es decir, sean $H_1, \ldots, H_n$ los conjuntos convexos iniciales donde cada H_1 representa una información condicional de una variable X_i dados sus padres $pa(X_i)$ en el grafo de dependencias. Cada H_i vendrá representado por una valuación V_i que estará definida para el conjunto de variables $\{X_i, pa(X_i), T_i\}$ donde T_i es una variable transparente que tiene tantos casos como puntos extremos tenga el conjunto convexo H_i, según se explicó en la sección 2.5.2. Sobre los nodos transparentes consideraremos que tenemos definida una distribución de probabilidad uniforme representada por una valuación V_{T_i} que está definida sólo para la variable T_i.

En este algoritmo se hace preciso ignorar el último valor simulado para una variable transparente cuando se va a simular de nuevo dicha variable transparente. En el algoritmo anterior cuando se simulaba un valor para una variable transparente se combinaba la valuación correspondiente a este valor simulado con la valuación de un grupo V_{C_i} que contuviese a esta variable transparente. En este caso, para poder ignorar el último valor simulado de la variable transparente, tendremos en cada grupo por separado la valuación del grupo V_{C_i} y por otro lado el conjunto de valuaciones de observaciones de las variables transparentes del grupo

$\{O_{T_{i_1}}, \ldots, O_{T_{i_k}}\}$. Cuando sea necesario enviar un mensaje a otro grupo adyacente C_{j_l} desde el grupo actual C_i, en lugar de aplicar la fórmula 2.11 aplicaremos la siguiente expresión que la modifica ligeramente.

$$M_{C_i \to C_{j_l}} = \left(\left(\bigotimes_{m=1,\ldots,k, m \neq l} M_{C_{j_m} \to C_i} \right) \otimes (V_{C_i} \otimes O_{T_{i_1}} \otimes \ldots \otimes O_{T_{i_k}}) \right)^{\downarrow X_{C_i} \cap X_{C_{j_l}}} \tag{3.8}$$

De la misma forma, cuando sea preciso obtener el vector 'a posteriori' para una variable X_j, en lugar de aplicar la expresión 2.12, aplicaremos la siguiente expresión que la modifica:

$$PS_j = (\bigotimes_{m=1,\ldots,l} V_{C_{j_m} \to C_i} \otimes (V_{C_i} \otimes O_{T_{i_1}} \otimes \ldots \otimes O_{T_{i_k}}))^{\downarrow \{j\}} \tag{3.9}$$

De esta forma se hace bastante sencillo poder ignorar el último valor simulado para una variable transparente T_i. Sólo hay que desechar la valuación observación O_{T_i} del grupo en el que esté incluida.

El esquema básico del algoritmo es el siguiente:

Algoritmo 3.6 Algoritmo de búsqueda de puntos con alto factor de normalización por muestreo de Gibbs

1. Construir el árbol de grupos para el grafo de dependencias.
2. Sea V_{C_i} la valuación del grupo C_i inicialmente con todos los valores a 1.
3. Asignar cada valuación inicial V_i, cada V_{T_i} y cada valuación observación a un grupo que contenga todas sus variables.
4. En cada grupo combinar todas las valuaciones asignadas a ese grupo en la valuación V_{C_i}.
5. Llevar a cabo una propagación probabilística en la forma que se explicó en la sección 2.3.2.5 pero usando la expresión 3.8 a la hora de calcular los mensajes.

   Obtenemos así una valuación V_{C_i} actualizada en cada grupo y dos valuaciones $M_{C_i \to C_j}$ y $M_{C_j \to C_i}$ para los mensajes entre cada par de grupos adyacentes.
6. Seleccionar un grupo raíz, C_0.
7. Llamar a Select-trans(C_0,C_0)

8. Para i=1 hasta N (Número de iteraciones deseado)
    - Llamar Select-Calc(C_0,C_0)

■

El procedimiento Select-Calc(C_1,C_2) es el procedimiento recursivo dado continuación:

Algoritmo 3.7 Select-Calc(C_1,C_2)

1. Para todos las variables transparentes T_{1_i} de C_1
    - Ignorar la observación anterior de esta variable transparente T_{1_i}
    - Calcular el vector 'a posteriori' $PS_{T_{1_i}}$ de la variable transparente T_{1_i} con la fórmula 3.9.
    - Simular un valor para el nodo transparente usando el vector 'a posteriori' $PS_{T_{1_i}}$.
    - Añadir al grupo C_1 una observación correspondiente al valor simulado para el nodo transparente anterior.
    - Para todas las variables no transparentes del grupo C_1 calcular los vectores 'a posteriori', teniendo en cuenta los casos seleccionados para las variables transparentes.
2. Para todos los grupos vecinos, C' de C_1 excepto C_2
    - Para todas las variables transparentes T_{1_i} de C_1
        - Ignorar la observación anterior de esta variable transparente T_{1_i}
        - Calcular el vector 'a posteriori' $PS_{T_{1_i}}$ de la transparente T_{1_i} con la fórmula 3.9.
        - Simular un valor para la variable transparente usando el vector 'a posteriori' $PS_{T_{1_i}}$.
        - Añadir al grupo C_1 una observación correspondiente al valor simulado para T_{1_i}.
        - Para todas las variables no transparentes del grupo C_1 calcular los vectores 'a posteriori', teniendo en cuenta los casos seleccionados para las variables transparentes.
        - Enviar un mensaje desde C_1 hasta C' calculado mediante la expresión 3.8.
        - Llamar a Select-Calc(C',C_1)
    - Enviar un mensaje desde C_1 hasta C_2 calculado mediante la expresión 3.8.

■

Aquí de nuevo se usa el mismo mecanismo que en el algoritmo anterior para simular las variables transparentes mediante el uso de un número aleatorio con la fórmula 3.7.

El coste de este algoritmo es similar al anterior. La principal diferencia de este algoritmo con el anterior es que ahora las muestras no son independientes como dijimos anteriormente. Además, ahora se obtienen más puntos 'a posteriori' para cada variable de interés. O sea, se obtiene un vector 'a posteriori' para todas las variables de un clique cada vez que lo visitamos.

Una condición necesaria que debe cumplirse para poder aplicar este algoritmo y que nos asegure que estamos seleccionando puntos del conjunto convexo 'a posteriori' de las variables de interés, con una probabilidad proporcional a $t_1 + t_2$ (o sea, su factor de normalización) es que la probabilidad de obtener cualquier configuración de las variables transparentes desde cualquier otra configuración es estrictamente positiva. Esta condición se verifica de forma evidente si para cada configuración de variables transparentes $(c_{i_1}, \ldots, c_{i_n})$ se cumple que $P_{c_{i_1},\ldots,c_{i_n}}(e) > 0$.

3.5.2.1 Aplicación de la técnica de enfriamiento simulado

Este último algoritmo puede ser modificado para adoptar la técnica del enfriamiento simulado explicada en la sección 3.3 de este capítulo. En la versión con enfriamiento simulado se necesita un parámetro adicional, la temperatura t que será usada a la hora de simular las variables transparentes y que debería ir decrementándose lentamente de acuerdo a algún mecanismo de enfriamiento adecuado según indicamos en la sección 3.3. La temperatura debe ser alta al principio del algoritmo para permitir a las configuraciones de variables transparentes moverse libremente. Luego, la temperatura debe decrementarse lentamente hasta valores cercanos a 0. El mecanismo de enfriamiento debería ser logarítmico en teoría, para asegurar que el algoritmo converge a la configuración de variables transparentes con máximo factor de normalización, pero en la práctica este mecanismo hace que el algoritmo tenga un costo de computación muy duro. El mecanismo de enfriamiento que nosotros hemos utilizado es el de Kirkpatrick, Gelatt y Vecchi [80], donde en cada iteración la temperatura se decrementa según la siguiente expresión:

$$t_{i+1} = \alpha \cdot t_i \tag{3.10}$$

donde $\alpha < 1$

Otra modificación que puede mejorar la eficiencia del algoritmo fue propuesta por Green y

Supowit [61]. De acuerdo con este algoritmo, si podemos calcular el costo de todas las configuraciones vecinas $N(x)$ de una configuración dada x, entonces en lugar de elegir aleatoriamente una configuración de $N(x)$ y aceptarla según el mecanismo detallado en la sección 3.3, es mejor elegir una configuración y de $N(x)$ usando una probabilidad proporcional a $e^{-(C(y)-C(x))/t}$ donde e representa el número e, que es la base para los logaritmos neperianos. Este mecanismo modificado será el que apliquemos en el algoritmo que estamos comentando y en los que comentaremos después.

Supongamos que en el algoritmo que estamos explicando en esta sección vamos a simular la variable transparente T_i y hemos obtenido el vector 'a posteriori' PS_{T_i}:

$$PS_{T_i} : T_i \rightarrow I\!R_0^+$$

Entonces a una determinada temperatura t este vector será transformado en el siguiente vector antes de la simulación de la variable transparente T_i:

$$e^{-PS_{T_i}/t} \tag{3.11}$$

donde $e^{-PS_{T_i}/t}(c_j) = e^{-PS_{T_i}(c_j)/t}$

3.5.3 Algoritmo de búsqueda de puntos con alto valor de t_1 y t_2

Este algoritmo está enfocado a buscar puntos del conjunto convexo 'a posteriori' que tengan alto valor de t_1 y de t_2 para alguna variable de interés X_j. Esta es la principal diferencia con los dos algoritmos anteriores, pues ahora el algoritmo va enfocado a buscar puntos para una única variable. Aunque el algoritmo puede obtener puntos del convexo 'a posteriori' de cualquier otra variable, éstos estarán optimizados para una variable determinada X_j. El algoritmo lo describiremos únicamente para una variable con dos casos, pero la generalización a variables con más casos es inmediata, ya que sólo habría que repetir el algoritmo para cada caso de la variable de interés, y para la negación de ese caso.

En este algoritmo se necesita un doble sistema de mensajes entre cada par de grupos adyacentes. En los anteriores algoritmos y en los algoritmos exactos se tenía entre cada dos grupos adyacentes C_i y C_j dos mensajes $M_{C_i \rightarrow C_j}$ y $M_{C_j \rightarrow C_i}$. En este algoritmo necesitaremos cuatro mensajes entre los grupos C_i y C_j para mantener dos sistemas de mensajes en el algoritmo de propagación. El primer sistema de mensajes es el usado para la propagación usual utilizada hasta ahora y que nos permite obtener puntos del conjunto convexo 'a posteriori' de

la variable de interés. El segundo sistema de mensajes es usado para simular las variables transparentes en forma similar a los anteriores algoritmos. Es decir, se usa para calcular de forma local la función a optimizar. Este segundo sistema de mensajes es igual al primero, pero transporta además la observación de uno de los casos de la variable de interés X_j. Para maximizar el valor t_1 incluiremos la observación del caso u_j^1 en el segundo sistema de mensajes. Si calculamos el vector 'a posteriori' PS_{T_i} para una variable transparente utilizando el segundo sistema de mensajes, tendremos un vector en el cada componente representa $P(e \cap u_j^1)$, que corresponde precisamente a t_1. De esta forma, podemos ver que al simular las transparentes con el segundo sistema de mensajes estamos maximizando los valores de t_1. Repitiendo el algoritmo con la observación del caso u_j^2 de X_j obtendremos puntos donde se maximiza t_2.

En este algoritmo también se hace preciso tener las observaciones de las variables transparentes de un grupo por separado de la valuación del grupo. Por tanto usaremos el mismo esquema explicado en el algoritmo anterior.

El algoritmo es el siguiente:

Algoritmo 3.8 Algoritmo de búsqueda de puntos con alto valor de t_1 y de t_2 por muestreo de Gibbs

1. Construir el árbol de grupos para el grafo de dependencias.
2. Sea V_{C_i} la valuación del grupo C_i inicialmente con todos los valores a 1.
3. Asignar cada valuación inicial V_i, cada V_{T_i} y cada valuación observación a un grupo que contenga todas sus variables.
4. En cada grupo combinar todas las valuaciones asignadas a ese grupo en la valuación V_{C_i}.
5. Para cada caso u_j^l de la variable de interés X_j
   - En el segundo sistema de mensajes combinar la valuación observación del caso u_j^l de la variable X_j con la valuación de un grupo que contenga la variable X_j.
   - Llevar a cabo una propagación probabilística en la forma que se explicó en la sección 2.3.2.5 usando los dos sistemas de mensajes pero usando la expresión 3.8 a la hora de calcular los mensajes.

     Obtenemos así para el primer sistema de mensajes una valuación $V_{C_i}^1$ actualizada en cada grupo y dos valuaciones $M_{C_i \to C_j}^1$ y $M_{C_j \to C_i}^1$ para los mensajes entre cada par

de grupos adyacentes y para el segundo sistema de mensajes la valuación $V^2_{C_i}$ en cada grupo y las dos valuaciones $M^2_{C_i \to C_j}$ y $M^2_{C_j \to C_i}$ entre cada par de grupos adyacentes.

- Seleccionar un grupo raíz C_0 que contenga la variable X_j.
- Para i=1 hasta N (número deseado de iteraciones)
  - Select-trans-secondmessages(C_0,C_0)
  - Calcular el vector 'a posteriori' PS_j para la variable X_j considerando el primer sistema de mensajes utilizando la fórmula 3.9.

■

La función Select-trans-secondmessages(C_1,C_2) es también una función recursiva similar a la función Select-Calc(C_1,C_2) con la única diferencia que el segundo sistema de mensajes es usado para simular las variables transparentes y que no se calculan vectores 'a posteriori' para las variables transparentes en cada grupo visitado. Ahora, además, se envían dos mensajes (uno para el primero y otro para el segundo sistema de mensajes) donde en la función Select-Calc(C_1,C_2) se enviaba uno.

Las variables transparentes se simulan de nuevo generando un número aleatorio con una distribución uniforme y aplicando la fórmula 3.7 como en los dos anteriores algoritmos, pero calculando el vector 'a posteriori' con el segundo sistema de mensajes.

3.5.3.1 Aplicación de la técnica de enfriamiento simulado

De la misma manera que en el algoritmo de la sección anterior, ahora también podemos modificar este algoritmo para que utilice la técnica del enfriamiento simulado explicado en la sección 3.3. De nuevo, necesitamos la temperatura t como parámetro adicional, que en este caso se usará para simular las variables transparentes en el segundo sistema de mensajes en forma similar al anterior algoritmo. Usaremos también el mecanismo de Kirpatrik, Gelatt y Vecchi [80] para decrementar la temperatura en cada iteración y el mecanismo para pasar a otra configuración de Green y Supowit [61]. En este caso el vector que se usa para simular una variable transparente es el vector 'a posteriori' obtenido con el segundo sistema de mensajes de la variable transparente que se va a simular.

3.5.4 Algoritmo de búsqueda de puntos con alto valor de $\frac{t_1}{t_1+t_2}$ y $\frac{t_2}{t_1+t_2}$

Este algoritmo es similar al anterior. Ahora el algoritmo está enfocado a la obtención de puntos con alto valor de $t_1/(t_1+t_2)$ y de $t_2/(t_1+t_2)$, o sea, intenta buscar puntos que maximicen los intervalos que se obtienen a partir del vector 'a posteriori' para una variable usando el condicionamiento de Dempster (definición 1.17). O lo que es lo mismo, se busca minimizar y maximizar la siguiente expresión para todos los casos $\{u_k^1, u_k^2, \ldots\}$ de la variable X_j:

$$P^{\downarrow j}_{c_{i_1},\ldots,c_{i_n}}(u_j^i|e) \tag{3.12}$$

También en este caso el algoritmo va dirigido a una variable de interés X_j y también se utiliza un doble sistema de mensajes. El primero se usa para obtener puntos del conjunto convexo 'a posteriori' de la variable X_j al igual que en los algoritmos anteriores y el segundo para simular las variables transparentes, es decir, para poder calcular de forma local la función a optimizar. De la misma forma se hará preciso, al simular una variable transparente, ignorar la observación previa de esta variable. Por ello al igual que en los dos algoritmos anteriores tendremos los potenciales correspondientes a las observaciones de las variables transparentes por separado de los potenciales de los grupos y será preciso aplicar la fórmula 3.8 para enviar un mensaje a un grupo vecino, y la fórmula 3.9 para obtener el vector 'a posteriori' de una variable. El principal cambio respecto al algoritmo anterior es en el vector usado para simular las variables transparentes. Veamos el procedimiento que se sigue ahora para simular las variables transparentes.

Para simular la variable transparente T_i primero se ignora la anterior observación para esta variable. Calculamos entonces dos vectores 'a posteriori' con el uso de la fórmula 3.9. Sea $PS^1_{T_i}$ y $PS^2_{T_i}$ los vectores 'a posteriori' para la variable transparente usando respectivamente el primer y segundo sistema de mensajes. Calculamos entonces el vector $PS_{T_i} = PS^2_{T_i}/PS^1_{T_i}$, en el que la división de vectores representa la división elemento a elemento. Si algún elemento $PS^1_{T_i}(c_i) = 0$ entonces puede comprobarse que también $PS^2_{T_i}(c_i)$ debe ser 0. En este caso consideraremos que la división de los dos elementos es 0 (el mínimo valor posible), ya que en ese caso la probabilidad de la evidencia es 0, y cuando esto ocurre esta probabilidad conjunta no debe usarse para calcular el ínfimo o el supremo de la probabilidad 'a posteriori' de una variable de interés. El vector PS_{T_i} es entonces usado para simular la variable transparente T_i usando el mismo mecanismo que en los algoritmos anteriores, con la generación de un

número aleatorio con distribución uniforme, y con el uso de la fórmula 3.7. Ahora, el vector 'a posteriori' PS_{T_i} usado en para simular un caso para la variable transparente T_i contiene en cada posición el valor $P(u_j^i|e)$. De esta forma, cuando se introduce la observación de u_j^1 en el segundo sistema de mensajes estamos maximizando $\frac{t_1}{t_1+t_2}$ y cuando se introduce u_j^2 estamos maximizando $\frac{t_2}{t_1+t_2}$.

3.5.4.1 Aplicación de la técnica de enfriamiento simulado

También en este caso podemos aplicar la técnica del enfriamiento simulado. El parámetro temperatura t servirá para modificar el vector calculado anteriormente PS mediante la aplicación de la fórmula 3.11.

3.5.5 Evaluación de los algoritmos

Para la evaluación de los algoritmos hemos considerado un grafo de dependencias generado de forma aleatoria. Este grafo puede verse en la figura 3.6. Desde un punto de vista probabilístico este grafo no es demasiado complicado. O sea, si en cada nodo del grafo tuviésemos una única distribución de probabilidad condicional entonces podríamos obtener con un costo pequeño los vectores 'a posteriori' de cualquier variable.

En el grafo usado, cada variable tendrá un número de casos que fue generado aleatoriamente con una distribución de Poisson de media 2, pero donde se tomaba un mínimo de 2 casos. Para asignar los padres a cada nodo establecemos un orden de los nodos, generamos el número de padres de cada nodo con una distribución uniforme en $[0, 2.8]$, redondeando al entero más próximo, y finalmente seleccionamos para cada nodo de forma aleatoria los nodos padres entre los nodos anteriores al actual.

Para cada variable el número de posibles distribuciones de probabilidad condicional se genera con una distribución uniforme en $[0, 3.8]$ redondeando al entero más próximo. En el grafo obtenido se tienen 1 495 686 distribuciones de probabilidad posibles, lo cual significa que para obtener el conjunto convexo 'a posteriori' exacto para una variable, habría que llevar a cabo 1 495 686 propagaciones probabilísticas en el anterior grafo de dependencias.

Para generar los valores de las distribuciones de probabilidad condicionada $P(X_i|pa(X_i)$ se selecciona de forma aleatoria uno de los casos de la variable X_i. A este caso se le asigna una probabilidad de 0.9. El resto de la masa de probabilidad (0.1) se reparte entre el resto de los casos aleatoriamente.

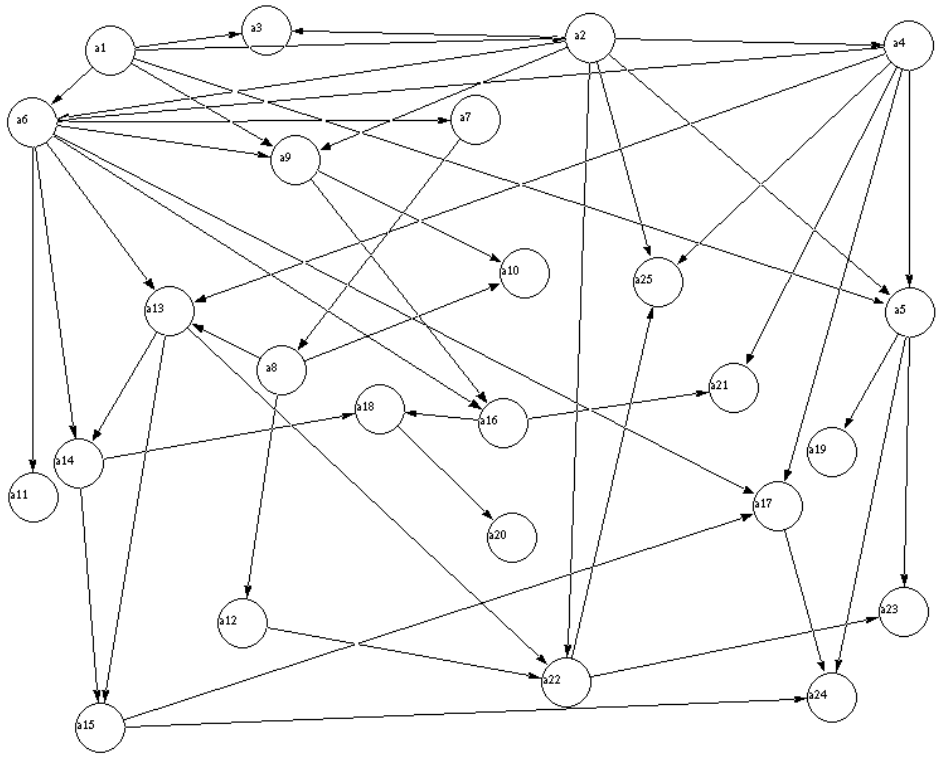


Figura 3.6: Grafo de dependencias usado para evaluar los algoritmos aproximados

Hemos llevado a cabo dos tipos de tests. En primer lugar hemos aplicado unos tests donde se han obtenido puntos del conjunto convexo 'a posteriori' mediante una combinación de los anteriores algoritmos. En el segundo test hemos aplicado el último algoritmo utilizando la técnica de enfriamiento simulado. En ambos casos el condicionamiento aplicado ha sido el de Moral y Campos dado en la definición 1.16.

Para ahorrar espacio al referirse a los algoritmos asignaremos un número a cada algoritmo de la siguiente forma:

- Algoritmo de búsqueda de puntos con alto factor de normalización: Algoritmo 1.
- Algoritmo de búsqueda de puntos con alto factor de normalización por muestreo de Gibbs: Algoritmo 2.
- Algoritmo de búsqueda de puntos con alto valor de t_1 y alto valor de t_2: Algoritmo 3.
- Algoritmo de búsqueda de puntos con alto valor de $t_1/(t_1+t_2)$ y $t_2/(t_1+t_2)$: Algoritmo 4

3.5.5.1 Primer grupo de tests: Obtención de puntos con una combinación de todos los algoritmos

Los algoritmos han sido implementados en una forma tal que para una variable determinada X_j, es posible acumular los vectores 'a posteriori' procedentes de diferentes algoritmos. Cuando hemos hecho uso de la técnica de enfriamiento simulado en un algoritmo se ha usado una temperatura inicial de 2 y un factor de enfriamiento $\alpha = 0.8$.

Se han llevado a cabo dos tipos de tests. En unos no se han utilizado observaciones en ninguna variable del grafo de dependencias y en otros se ha considerado que teníamos observado el primer caso de las variables X_4, X_{10}, X_{17} y X_{23}.

Dentro de este grupo de tests hemos realizado los siguientes experimentos:

1. 80 iteraciones del algoritmo 1.
2. 80 iteraciones del algoritmo 2.
3. 80 iteraciones del algoritmo 2 utilizando la técnica de enfriamiento simulado.
4. 80 iteraciones del algoritmo 2 y 40 iteraciones del algoritmo 3.
5. 40 iteraciones del algoritmo 2 y 40 iteraciones con el algoritmo 3 con enfriamiento simulado.
6. 25 iteraciones con el algoritmo 2, 30 iteraciones con el algoritmo 3, y 25 iteraciones con el algoritmo 4.
7. 25 iteraciones con el algoritmo 2, 30 iteraciones con el algoritmo 3 y 25 iteraciones con el algoritmo 4 con enfriamiento simulado.

Hemos seleccionado 4 variables no observadas de forma aleatoria para llevar a cabo los test: X_3, X_8, X_{15} y X_{21}. Para chequear la bondad de las aproximaciones hemos calculado los intervalos exactos para estas variables con un algoritmo exacto que ha llevado un gran tiempo de computación. Los intervalos se obtienen usando el condicionamiento de Moral y Campos (definición 1.16) tanto para el método exacto como para nuestros experimentos aproximados.

Cada uno de los 8 anteriores experimentos ha sido ejecutado 10 veces. Cada vez hemos medido el error con el límite superior de los intervalos exactos de las variables estudiadas. En las tablas 3.1 hasta la 3.4 expresamos el error medio, el mínimo y el máximo error con los diferentes algoritmos en las 10 ejecuciones de cada experimento.

Test	Media	Min	Max
1	0.0	0.0	0.0
2	0.0	0.0	0.0
3	0.0	0.0	0.0
4	0.0	0.0	0.0
5	0.0	0.0	0.0
6	0.0	0.0	0.0
7	0.0	0.0	0.0

X_3

Test	Media	Min	Max
1	0.0	0.0	0.0
2	0.0	0.0	0.0
3	0.0	0.0	0.0
4	0.0	0.0	0.0
5	0.0	0.0	0.0
6	0.0	0.0	0.0
7	0.0	0.0	0.0

X_8

Tabla 3.1: Errores para las variables X_3 y X_8 sin observaciones

Test	Media	Min	Max
1	0.002	0.0	0.006
2	0.0	0.0	0.0
3	0.001	0.0	0.003
4	0.0	0.0	0.0
5	0.0	0.0	0.0
6	0.0	0.0	0.0
7	0.0	0.0	0.0

X_{15}

Test	Media	Min	Max
1	0.0	0.0	0.0
2	0.0	0.0	0.0
3	0.001	0.0	0.006
4	0.0	0.0	0.0
5	0.0	0.0	0.0
6	0.0	0.0	0.0
7	0.0	0.0	0.0

X_{21}

Tabla 3.2: Errores para las variables X_{15} y X_{21} sin observaciones

Test	Media	Min	Max
1	0.02	0.0	0.064
2	0.03	0.019	0.051
3	0.098	0.01	0.149
4	0.013	0.003	0.027
5	0.018	0.003	0.097
6	0.017	0.002	0.042
7	0.041	0.001	0.117

X_3

Test	Media	Min	Max
1	0.038	0.006	0.103
2	0.021	0.001	0.042
3	0.45	0.055	0.853
4	0.015	0.002	0.034
5	0.011	0.002	0.027
6	0.02	0.006	0.035
7	0.041	0.008	0.099

X_8

Tabla 3.3: Errores para las variables X_3 y X_8 con observaciones

Test	Media	Min	Max
1	0.02	0.0	0.079
2	0.004	0.0	0.021
3	0.034	0.0	0.143
4	0.009	0.0	0.019
5	0.06	0.001	0.143
6	0.007	0.0	0.015
7	0.014	0.0	0.112

X_{15}

Test	Media	Min	Max
1	0.022	0.001	0.047
2	0.022	0.008	0.046
3	0.048	0.006	0.122
4	0.019	0.003	0.034
5	0.01	0.002	0.046
6	0.019	0.003	0.055
7	0.02	0.001	0.081

X_{21}

Tabla 3.4: Errores para las variables X_{15} y X_{21} con observaciones

Algunas conclusiones que podemos obtener de los anteriores experimentos son las siguientes:

- Se pueden obtener buenas aproximaciones de los intervalos de probabilidad asociados al conjunto convexo 'a posteriori' con pocas propagaciones probabilísticas.

- Si no hay observaciones, las aproximaciones son bastante buenas. Los experimentos que hemos realizado en tal caso casi siempre producen los intervalos exactos. Sólo el experimento 3, introduce algún error. La explicación de este hecho puede estar basada en que sin observaciones el espacio de búsqueda es usualmente más pequeño que con observaciones. Con ninguna observación muchas de las distribuciones condicionales son irrelevantes en el cálculo del vector 'a posteriori' de una variable. Las valuaciones irrelevantes pueden obtenerse con la aplicación del algoritmo 2.3 que fue detallado en la sección 2.3.2.5. Además, sin observaciones los conjuntos convexos 'a posteriori' son más simples ya que todas las probabilidades tienen la misma posibilidad (factor de normalización).

- Casi todos los experimentos excepto el número 3, producen resultados comparables. Esto es algo que no era esperado en un principio, pues se esperaba obtener mejores resultados con los experimentos 6 y 7. Esto puede ser debido quizás a la naturaleza de este grafo.

- La técnica del enfriamiento simulado no introduce mejoras sustanciales en los cálculos. De hecho el experimento 3 (el mismo que el 2, pero con enfriamiento simulado) es el peor de todos los experimentos. Esto puede ser debido quizás, al rápido mecanismo de enfriamiento que hemos usado, con lo que el método converge a un mínimo local.

Además, tenemos que tener en cuenta que necesitamos no sólo el valor óptimo (el vector con más alta posibilidad) sino una familia de puntos con alto valor de posibilidad.

- Con el condicionamiento de Dempster (problema de optimización clásico) no se han hecho experimentos en esta sección, pero se harán en el capítulo 5, en una situación más compleja.

3.5.5.2 Segundo grupo de tests: Obtención de puntos con el algoritmo 4 usando enfriamiento simulado

En este segundo grupo de tests hemos vuelto a utilizar el grafo de dependencias de la figura 3.6. Se ha utilizado el algoritmo 4 con la técnica de enfriamiento simulado en un intento de ver si esta técnica podía mejorarse respecto a los malos resultados obtenidos en el grupo de test del apartado anterior.

En este caso hemos considerado también que tenemos las mismas variables observadas que con los experimentos del grupo uno (X_4, X_{10}, X_{17} y X_{23}). Con observaciones el problema es más difícil que sin ellas.

Puesto que el algoritmo 4 está especialmente indicado para obtener el mínimo y máximo de los puntos del conjunto convexo 'a posteriori' de la variable de interés, una vez que los puntos están normalizados, hemos utilizado en este caso el condicionamiento de Dempster (definición 1.17) para chequear la bondad de los intervalos obtenidos.

Los experimentos de este grupo se han aplicado para obtener los conjuntos convexos 'a posteriori' y luego los intervalos para todas las variables no observadas del grafo de dependencias. El número de iteraciones que hemos empleado ha sido de 40 para cada variable. Los mecanismos de enfriamiento utilizados han sidos los siguientes:

1. $t_0 = 2.0$ y $\alpha = 0.9$
2. $t_0 = 3.0$ y $\alpha = 0.92$
3. $t_0 = 0.5$ y $\alpha = 0.92$

Las tablas 3.5 y 3.6 nos muestran los errores medios obtenidos con los diferentes mecanismos de enfriamiento para los límites superiores de los intervalos aplicando los algoritmos 50 veces.

Variable	a1	a2	a3	a5	a6	a7	a8	a9	a11	a12
a)	0.0016	0.0009	0.0016	0.0060	0.0032	0.0157	0.0044	0.0026	0.0002	0.0769
b)	0.0021	0.0012	0.0026	0.0133	0.0046	0.0269	0.0079	0.0034	0.0003	0.0797
c)	0.0029	0.0014	0.0031	0.0226	0.0063	0.0268	0.0125	0.0043	0.0004	0.0871

Tabla 3.5: Media de error en los límites superiores de los intervalos de las variables

Variable	a13	a14	a15	a16	a18	a19	a20	a21	a22	a24	a25
a)	0.0045	0.0079	0.0033	0.0068	0.0484	0.0009	0.0395	0.0157	0.0030	0.0044	0.0011
b)	0.0084	0.0174	0.0048	0.0113	0.0665	0.0010	0.0554	0.0181	0.0049	0.0067	0.0017
c)	0.0101	0.0204	0.0056	0.0163	0.0680	0.0019	0.0624	0.0194	0.0081	0.0102	0.0020

Tabla 3.6: Media de error en los límites superiores de los intervalos de las variables

Podemos observar que para la mayoría de las variables el error medio obtenido es pequeño. Sin embargo, para algunas de ellas, por ejemplo, la variable X_{12}, el error medio puede considerarse demasiado alto. Una conclusión importante que puede observarse a partir de las tablas, es que el primer esquema de enfriamiento es el mejor para todos los intervalos, y el tercero el peor. Podemos pensar que la bondad de un esquema de enfriamiento tenga probablemente bastante independencia del problema concreto.

Una vez que hemos comprobado que es en la variable X_{12} donde peores resultados se obtiene, hemos dedicado un esfuerzo adicional a esta variable. Los mejores resultados se han obtenido entonces con 70 iteraciones tomando $t_0 = 5.0$ y $\alpha = 0.95$. El error medio en este caso ha sido de 0.0433.

3.6 Algoritmos genéticos aplicados a la propagación de conjuntos convexos

En los algoritmos que explicaremos en esta sección se utiliza la técnica general de optimización de los algoritmos genéticos. Los algoritmos que presentaremos en esta sección los usaremos también, al igual que en el caso de los algoritmos de las sección anterior, para buscar puntos que cumplan ciertas *buenas* condiciones del conjunto convexo 'a posteriori' de ciertas variables de interés. La idea de estos algoritmos es manejar una población de puntos del conjunto convexo 'a posteriori' que aunque inicialmente no sean puntos con buenas propiedades, tras ir ejecutando el algoritmo permitan que esta población inicial evolucione hacia una población de puntos en

la que cada uno de ellos cumpla con esas buenas condiciones. En el caso del condicionamiento de Moral y Campos no hay un único punto del conjunto convexo 'a posteriori' que sea el mejor de todos y al que un buen algoritmo deba converger. Existen más bien, varios puntos que son buenos para el cálculo de los intervalos de la variable de interés. Un buen algoritmo debería obtener todos estos buenos puntos. Puesto que éste es un problema en el que no hay un sólo objetivo, o función a maximizar, e incluso para una de esas funciones a maximizar existen varios puntos que alcanzan el máximo, entonces pensamos que los algoritmos genéticos pueden funcionar bien, ya que mantienen una población de soluciones, y no una sola como otros algoritmos.

Al igual que en el caso de los algoritmos de la sección anterior 3.5, también en estos algoritmos haremos la trasformación del problema original por medio del cual se añadía a cada conjunto convexo del problema original una variable transparente con tantos casos como puntos extremos tuviese dicho conjunto convexo. Supongamos que el número de casos de la variable transparente T_i es k_i. No habrá valuación inicial para las variables transparentes, ya que siempre se propagará con ellas observadas, y tener observación en la transparente es equivalente a tener la ignorancia más dicha observación.

También haremos uso del esquema propuesto por el algoritmo exacto de propagación en un árbol de grupos y que fue detallado en la sección 2.3.2.5.

3.6.1 Búsqueda de puntos con alto factor de normalización

El algoritmo genético planteado a continuación se ha utilizado para obtener puntos extremos del conjunto convexo 'a posteriori' de las variables de interés que tengan un alto factor de normalización. O sea, suponiendo que la variable de interés tenga dos casos, nos interesan puntos con alto valor de $t_1 + t_2$ al igual que los algoritmos expuestos en las secciones 3.5.1 y 3.5.2, donde t_1 y t_2 están definidos en la fórmula 3.5.

En este algoritmo el primer paso es la construcción del árbol de grupos. Para ello procederemos de la misma forma que en los algoritmos de la sección anterior, con la utilización del algoritmo 3.3.

La ventaja que presenta este algoritmo evolutivo sobre el algoritmo presentado en la sección 3.5.1 es que ahora los muestreos no van a ser independientes en cada iteración. Cada iteración del algoritmo genético se va a basar en la población previa para construir una nueva población, y además mediante los operadores de mutación y cruce vamos a poder explorar todas las zonas

del espacio de búsqueda.

De la misma forma que en los algoritmos de la sección 3.5, si seleccionamos un caso para cada transparente del árbol de grupos y hacemos una propagación probabilística se puede obtener un punto del conjunto convexo 'a posteriori' de todas las variables no observadas X_j en el árbol de grupos con el uso de la fórmula 2.12. De esta forma definimos un individuo o cromosoma de la población del algoritmo genético como una configuración de los casos seleccionados en todas las variables transparentes. Para representar estos individuos utilizaremos un vector de números enteros. Cada posición del vector corresponderá a una variable transparente del árbol de grupos. O sea la posición i del vector corresponderá al caso seleccionado para la variable transparente T_i. Cuando en la posición i tengamos un 0 quiere decir que tenemos seleccionado el caso c_0 de la variable transparente T_i, si hay un 1 que tenemos seleccionado el caso c_1 y así sucesivamente. De esta forma cada individuo de la población determinará una probabilidad conjunta , con la que se puede obtener un punto del conjunto convexo 'a posteriori' de cualquier variable no observada.

Cada individuo de la población tiene asignado un *valor de adecuación* que en este caso es el factor de normalización de los puntos del conjunto convexo 'a posteriori' en cualquier variable no observada. Este factor de normalización, como ya conocemos, es el mismo en todas las variables no observadas.

Los parámetros que puede seleccionar el usuario en el algoritmo serán:

- **Tamaño de la población** (t): Nos va a dar el número de configuraciones de nodos transparentes que mantenemos en cada población. Puede que algunos individuos de la población sean iguales, sobre todo a medida que el algoritmo genético evoluciona.

- **Número de generaciones**(n_{it}): Es el número de veces que aplicaremos el algoritmo evolutivo. Nos da por tanto la condición de parada. Otra posible condición de parada hubiera sido cuando en varias iteraciones seguidas no se mejorase el mejor individuo.

- **Número inicial de iteraciones**: Es el número de veces que aplicamos el algoritmo de inicialización de la población. Este parámetro sólo es usado en la fase de inicialización de la población.

- **Probabilidad de cruce**: Nos da la probabilidad de que un individuo de la población sea seleccionado para el cruce con otro individuo, produciendo así dos descendientes nuevos.

- **Probabilidad de mutación**: Nos da la probabilidad de cambiar el estado de una de las variables transparentes en un individuo de la población.

El algoritmo que veremos tiene una fase de inicialización de la población que se encarga de construir la población inicial, de individuos. Con esta fase se obtienen t individuos independientes. Cada individuo se construye seleccionando aleatoriamente un caso de cada una de las variables transparentes y llevando a cabo una propagación probabilística para poder calcular el valor de adecuación de ese individuo.

Después de inicializar la población y evaluar cada uno de los individuos de esta población inicial se llevan a cabo n_{it} iteraciones para que la población inicial evolucione hacia una población en la que los individuos tenga alto factor de normalización.

En una iteración del algoritmo genético se desarrollan las siguientes tres fases:

- **Selección**: Una vez que cada cromosoma de la población ha sido evaluado, se asigna a cada cromosoma una probabilidad de supervivencia que va a depender del valor de esta evaluación. La evaluación de un cromosoma se hace tomando como función objetivo a la posibilidad de los puntos obtenidos tras la propagación que como venimos comentando se calcula como el factor de normalización de dichos puntos. Para calcular esta posibilidad basta con sumar la probabilidad de todos los casos de cualquier variable no observada, lo cual nos proporciona el valor $P(e)$ o probabilidad de la evidencia.

  Para calcular las probabilidades de supervivencia sumamos las adecuaciones de todos los cromosomas de la población y posteriormente para a cada cromosoma se le asigna como probabilidad el resultado de dividir su adecuación por la adecuación total. Para hacer la selección sólo tenemos que generar N números aleatorios en el dominio $[0, \ldots, 1]$ y usar una ruleta dividida en N slots para seleccionar N cromosomas que pasarán a la siguiente generación. Evidentemente habrá cromosomas que tendrán varias copias en la siguiente generación y otros que no pasarán a la siguiente generación. O sea la probabilidad de que un cromosoma cr_i pase a la siguiente generación se calcula como:

$$P(Seleccionarcr_i) = Eval(cr_i)/\sum_j Eval(cr_j) \tag{3.13}$$

  donde $Eval(cr_j)$ es la función de evaluación de un cromosoma que según hemos dicho es el factor de normalización.

- **Cruce**: Para cada individuo de la población generaremos un número aleatorio en el intervalo $[0, 1]$. Sea r el número aleatorio generado y p_c la probabilidad de cruce. Si $r < p_c$ entonces el individuo actual se selecciona para el cruce. Al final de este proceso hacemos el cruce de los individuos seleccionados para el cruce seleccionándolos de dos en dos de forma aleatoria. En cada pareja que se va a cruzar seleccionaremos también de forma aleatoria el punto de cruce. Si suponemos que nuestros individuos están formados por 10 números enteros (10 variables transparentes) y nos sale como punto de cruce el 3, entonces el primer descendiente estará formado por las tres primeras posiciones del primer individuo y las últimas 7 del segundo individuo, y el segundo descendiente estará formado por las tres primeras posiciones del segundo individuo y las siete últimas del primer individuo.

- **Mutación**: Para cada individuo de la población y para cada una de sus posiciones generamos un número aleatorio en el dominio $[0, 1]$. Sea r el número aleatorio generado y p_m la probabilidad de mutación. Si $r < p_m$ entonces la posición actual será mutada. O sea que será generado un nuevo caso para esa variable transparente.

Tras cada iteración es necesario hacer una propagación de aquellos individuos que no están propagados, o sea de aquellos individuos que no estaban en la población anterior.

Veamos pues el esquema básico de este algoritmo:

Algoritmo 3.9 Algoritmo genético para la búsqueda de puntos con alto factor de normalización

1. Introducir el tamaño de la población t, probabilidad de cruce p_c, probabilidad de mutación p_m y número de iteraciones N.
2. Construir un árbol de grupos para el grafo de dependencias en la forma que se explicó en la sección 3.5.
3. Sea V_{C_i} la valuación del grupo C_i, inicialmente con todos los valores a 1.
4. Asignar cada valuación observación O_i a un grupo que contenga la variable observada.
5. En cada grupo, combinar todas las valuaciones asignadas a ese grupo en la valuación V_{C_i}.
6. Inicializar la población con t elementos.

7. Para $i = 1$ hasta N

   (a) Seleccionar los cromosomas para la siguiente generación.

   (b) Llevar a cabo la operación de cruce de individuos de la población.

   (c) Llevar a cabo la operación de mutación de elementos de la población.

   (d) Calcular el valor de adecuación de los nuevos individuos de la población.

8. Calcular los intervalos para las variables de interés.

■

3.6.1.1 Evaluación del algoritmo

En este caso hemos usado el grafo de la figura 3.7 que ha sido generado aleatoriamente al igual que el usado en la sección anterior. Este grafo tiene 15 variables, cada una con dos casos. El número de padres de cada variable se ha generado de acuerdo a una distribución uniforme de media 2. El número de posibles distribuciones de probabilidad condicional en cada variable se generó con una distribución uniforme de media 3. El número de posibles distribuciones de probabilidad global es 2 732 480. Este grafo no es muy complicado desde un punto de vista probabilístico, pero el número de posibles distribuciones de probabilidad hace que el problema sea inmanejable en los ordenadores de que disponemos actualmente. Consideramos que las variables X_7, X_9 y X_{12} tienen observado su primer caso.

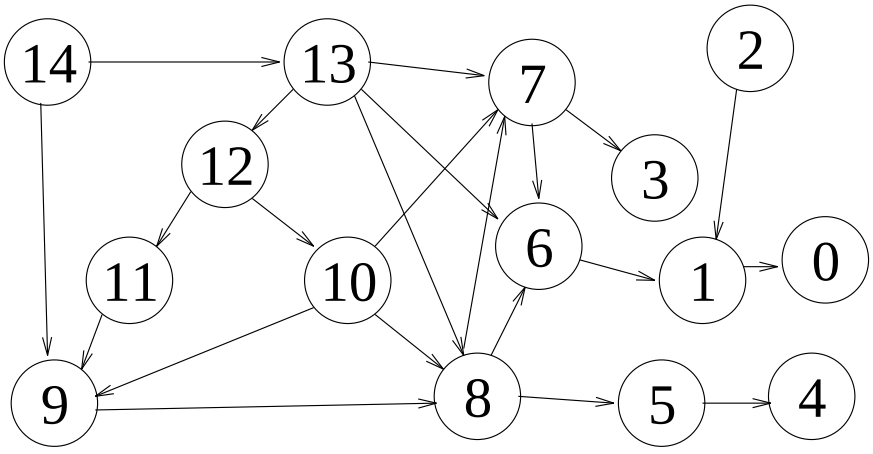


Figura 3.7: Grafo de dependencias usado en la experimentación

En la tabla 3.7 mostramos los intervalos obtenidos usando el condicionamiento de Dempster y el condicionamiento de Moral y Campos usando un método exacto de propagación. Mostramos sólo los intervalos para el primer caso de cada variable. Los intervalos para el segundo caso pueden obtenerse a partir los intervalos del primer caso, al tener las variables sólo dos casos.

Los intervalos proporcionados por nuestro algoritmo deben de parecerse a los del condicionamiento de Campos y Moral, ya que se obtienen puntos con altos valores de posibilidad no tratando de optimizar las probabilidades condicionales.

Los parámetros usados en el algoritmo genético han sido los siguientes: $p_c = 0.25$ y $p_m = 0.01$.

Tabla 3.7: Intervalos exactos obtenido con la aplicación del condicionamiento de Moral y Campos (Ch) y el condicionamiento Dempster (De)

	X0	X1	X2	X3
Ch	[0.174,0.692]	[0.258,0.778]	[0.020,0.670]	[0.080,0.692]
De	[0.171,0.697]	[0.247,0.787]	[0.020,0.670]	[0.080,0.692]
	X4	X5	X6	X8
Ch	[0.425,0.737]	[0.315,0.719]	[0.131,0.827]	[0.182,0.932]
De	[0.404,0.738]	[0.261,0.719]	[0.094,0.871]	[0.052,0.933]
	X10	X11	X13	X14
Ch	[0.017,0.843]	[0.001,0.896]	[0.024,0.820]	[0.006,0.835]
De	[0.005,0.980]	[0.001,0.955]	[0.012,0.995]	[0.006,0.986]

En la tabla 3.8 mostramos los intervalos obtenidos mediante el condicionamiento de Dempster con una población de tamaño 25 y diferentes números de iteraciones (10 hasta 50). En la tabla 3.9 mostramos los intervalos obtenidos mediante el condicionamiento de Dempster con 25 iteraciones y diferentes tamaños de población (10 hasta 50).

El número medio de propagaciones probabilísticas llevadas a cabo por el algoritmo se puede calcular como $t + n_{it} \cdot (t \cdot p_c + t \cdot p_m)$. Por ejemplo, si $t = 25$, $n_{it} = 40$, $p_c = 0,25$ y $p_m = 0,01$ entonces el número medio de propagaciones sería 51.

Algunas conclusiones que podemos obtener de los anteriores experimentos son las siguientes:

Tabla 3.8: Intervalos obtenidos con un algoritmo genético.(Tamaño de la población=25)

	X0	X1	X2	X3
10	[0.187,0.663]	[0.281,0.760]	[0.020,0.670]	0.080,0.692]
20	[0.187,0.658]	[0.283,0.763]	[0.020,0.670]	[0.080,0.692]
30	[0.180,0.667]	[0.270,0.764]	[0.020,0.670]	[0.080,0.692]
40	[0.181,0.675]	[0.271,0.769]	[0.020,0.670]	[0.080,0.692]
50	[0.184,0.669]	[0.266,0.774]	[0.020,0.670]	[0.080,0.692]
	X4	X5	X6	X8
10	[0.414,0.719]	[0.276,0.701]	[0.139,0.806]	[0.076,0.904]
20	[0.413,0.727]	[0.270,0.702]	[0.127,0.813]	[0.070,0.905]
30	[0.410,0.722]	[0.272,0.705]	[0.127,0.814]	[0.074,0.913]
40	[0.412,0.728]	[0.271,0.709]	[0.120,0.808]	[0.071,0.910]
50	[0.408,0.727]	[0.269,0.710]	[0.113,0.827]	[0.072,0.914]
	X10	X11	X13	X14
10	[0.034,0.950]	[0.001,0.893]	[0.044,0.983]	[0.013,0.970]
20	[0.020,0.957]	[0.001,0.901]	[0.040,0.986]	[0.011,0.967]
30	[0.031,0.954]	[0.001,0.913]	[0.034,0.983]	[0.011,0.971]
40	[0.028,0.951]	[0.001,0.916]	[0.032,0.984]	[0.008,0.968]
50	[0.023,0.954]	[0.001,0.916]	[0.031,0.982]	[0.008,0.973]

- El algoritmo desarrollado en esta sección ha sido diseñado para buscar puntos extremos de los conjuntos convexos 'a posteriori' que tengan alto factor de normalización. Ello hace que otros puntos importantes con bajo factor de normalización no sean encontrados, por lo que los resultados pueden verse afectados.

- Las tablas 3.8 y 3.9 muestran que los intervalos obtenidos con el condicionamiento de Dempster con el algoritmo genético son similares a los obtenidos con un método exacto de propagación aplicando el condicionamiento de Moral y Campos.

- Concretamente la tabla 3.8 nos muestra que los intervalos obtenidos con el algoritmo genético están incluidos dentro de los intervalos exactos, y que conforme crece el número de iteraciones los intervalos obtenidos se van aproximando cada vez más a los exactos.

Tabla 3.9: Intervalos obtenidos con un algoritmo genético con un número de iteraciones de 25

	X0	X1	X2	X3
10	[0.191,0.631]	[0.303,0.730]	[0.020,0.670]	[0.080,0.692]
20	[0.183,0.656]	[0.283,0.766]	[0.020,0.670]	[0.080,0.692]
30	[0.182,0.667]	[0.274,0.766]	[0.020,0.670]	[0.080,0.692]
40	[0.178,0.674]	[0.267,0.769]	[0.020,0.670]	[0.080,0.692]
50	[0.179,0.683]	[0.267,0.774]	[0.020,0.670]	[0.080,0.692]
	X4	X5	X6	X8
10	[0.423,0.701]	[0.285,0.678]	[0.155,0.781]	[0.083,0.855]
20	[0.410,0.723]	[0.272,0.704]	[0.133,0.789]	[0.070,0.908]
30	[0.409,0.728]	[0.269,0.708]	[0.122,0.820]	[0.067,0.908]
40	[0.408,0.728]	[0.269,0.710]	[0.119,0.835]	[0.068,0.919]
50	[0.407,0.734]	[0.268,0.715]	[0.123,0.838]	[0.066,0.925]
	X10	X11	X13	X14
10	[0.061,0.940]	[0.007,0.879]	[0.055,0.967]	[0.020,0.956]
20	[0.033,0.941]	[0.001,0.899]	[0.039,0.981]	[0.011,0.964]
30	[0.022,0.950]	[0.001,0.917]	[0.030,0.989]	[0.011,0.972]
40	[0.020,0.952]	[0.001,0.920]	[0.026,0.989]	[0.008,0.974]
50	[0.018,0.966]	[0.001,0.928]	[0.025,0.990]	[0.007,0.974]

- También la tabla 3.9 nos muestra que haciendo crecer el tamaño de la población se consigue aproximar mejor los intervalos del genético.

- Los resultados obtenidos aunque son buenos, creemos que pueden mejorarse intentando en un futuro próximo otras formas de realizar la operación de cruce. Por ejemplo, pensamos que podría llevarse a cabo la operación de cruce seleccionando aleatoriamente uno de los grupos del árbol, que actuaría como punto de corte para decidir que variables transparentes irían a cada cromosoma hijo.

- También, mediante la técnica que desarrollamos en la siguiente sección podremos obtener otros puntos extremos importantes del conjunto convexo 'a posteriori' para conseguir mejores aproximaciones.

3.6.2 Búsqueda de todos los puntos extremos mediante la combinación de un algoritmo genético y un algoritmo heurístico

En esta sección presentamos de nuevo un algoritmo genético utilizado en la búsqueda de puntos del conjunto convexo 'a posteriori' de las variables de interés. En este caso el algoritmo va enfocado a obtener *buenos* puntos extremos del convexo 'a posteriori' para una de las variables no observadas del problema.

En este caso el objetivo es obtener todos los puntos extremos del conjunto convexo 'a posteriori' para la variable de interés X_j. En lo que sigue supondremos que la variable X_j tiene sólo dos casos para simplificar la notación, aunque la generalización a más casos es sencilla.

El algoritmo se va a dividir en dos fases. En la primera fase se usa un algoritmo genético que busca una población de puntos con alto factor de normalización. Esta fase tiene los mismos pasos que el algoritmo genético explicado en la sección anterior. En la segunda fase se toman los puntos obtenidos en la etapa anterior y se aplicará un algoritmo iterativo para obtener otros puntos importantes del conjunto convexo 'a posteriori' de la variable X_j. Esta segunda fase se divide en varias etapas. En cada una de las etapas de esta segunda fase se intenta maximizar una determinada función mediante un algoritmo de mejora iterativa. Esta función mide la distancia de los puntos del conjunto convexo 'a posteriori' de la variable X_j respecto a un determinado hiperplano. En nuestro caso, al tener la variable X_j sólo dos casos el hiperplano es una línea recta (dimensión 1). En el caso general, cuando X_j tenga n casos, el hiperplano será de dimensión $n-1$. Sigamos considerando que la variable X_j tiene 2 casos. Este hiperplano será inicialmente la recta $t_1 + t_2 = 1$. En la figura 3.8 podemos ver una representación de esta recta. Podemos ver que esta recta forma un ángulo $\alpha = 135^o$ con el eje t_1. Si hacemos girar la recta anterior sobre el punto $t_1 = 1, t_2 = 0$ en el sentido de las agujas del reloj, desde el ángulo $\alpha = 135^o$ hasta el ángulo $\alpha = 0^o$, podemos ir obteniendo otras rectas donde el ángulo α se va decrementando en cada etapa. En cada etapa se intentará buscar el punto del conjunto convexo 'a posteriori' de la variable X_j que minimice la distancia Δ desde la recta que pasa por el punto $t_1 = 1, t_2 = 0$, y que forma un ángulo α con el eje t_1. Minimizar la distancia Δ es equivalente a maximizar la siguiente función:

$$f(x) = sin\alpha \cdot t_1 - cos\alpha \cdot t_2 \tag{3.14}$$

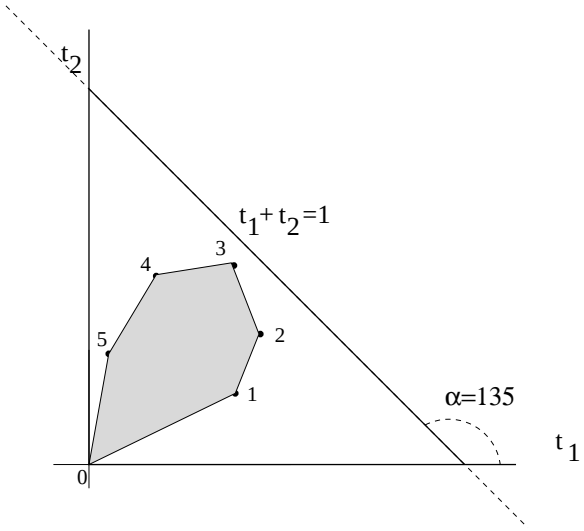


Figura 3.8: Conjunto convexo 'a posteriori' para X_j

donde $x = (t_1, t_2)$ es el punto del conjunto convexo 'a posteriori' de la variable X_j

En la figura 3.9 podemos ver que para $\alpha = 135^o$ el punto que maximiza la fórmula 3.14 es el número 3. Para $\alpha = 90^o$ el mejor punto es el 2, y para $\alpha = 0^o$ el mejor punto es el 1.

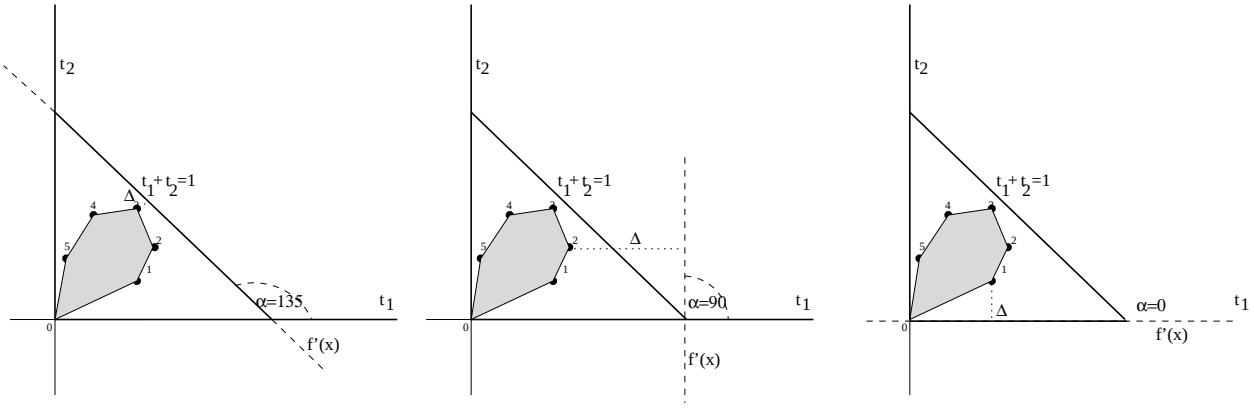


Figura 3.9: Puntos que maximizan la ecuación 3.14 con distintos ángulos

Intercambiando el papel de t_1 y t_2 en la ecuación 3.14 podemos obtener los puntos 4 y 5 de la misma forma que anteriormente.

La segunda fase del algoritmo está enfocada en una variable de interés, mientras que la primera es válida para cualquier variable. Para llevar a cabo esta segunda fase del algoritmo se necesita un segundo sistema de mensajes al igual que en el caso de los algoritmos 3.5.3 y

3.5.4 que usaban la técnica de enfriamiento simulado.

A continuación mostramos el esquema del algoritmo. La primera fase no se detalla pues es básicamente la aplicación del algoritmo 3.9 de la sección anterior.

Algoritmo 3.10 Combinación de un algoritmo genético con un algoritmo iterativo

1. Ejecutar el algoritmo 3.9 obteniendo una población de t individuos correspondientes a puntos del conjunto convexo 'a posteriori' con alto factor de normalización.
2. Considerar dos sistemas de mensajes en el árbol de grupos: M^1 son los mensajes que se utilizaban en la primera fase del algoritmo, y M^2 son los nuevos mensajes.
3. Para cada caso u_j^l de la variable de interés X_j:
   (a) En el segundo sistema de mensajes combinar la valuación observación del caso u_j^l de la variable X_j con la valuación de un grupo que contenga a la variable X_j.
   (b) Para $\alpha = 135$ hasta 0
   - Restaurar la configuración de variables transparentes correspondientes al punto x de la población que maximiza la función 3.14 con el ángulo α.
   - A partir de la anterior configuración aplicar un algoritmo iterativo para maximizar la fórmula 3.14.

■

En el algoritmo anterior, cuando acaba la primera fase, o sea cuando acaba el algoritmo genético se tiene una población de cromosomas. Cada cromosoma nos da una configuración de todas variables transparentes del árbol de grupos. Cada configuración de transparentes nos determina un punto con alto factor de normalización del conjunto convexo 'a posteriori' de cualquier variable no observada del árbol de grupos. Para obtener este punto habría que hacer una propagación probabilística con las observaciones correspondientes a cada variable transparente.

Cada vez que se pasa por 3.b hay que decrementar el valor de α en una cierta cantidad d). Esta cantidad no tiene por qué ser constante para distintos valores de α. De hecho para valores más pequeños (menos verosimilitud de los puntos y menos importancia final) los incrementos pueden ser mayores.

En la segunda fase del algoritmo se hace preciso restaurar aquella configuración de variables transparentes que maximice la expresión 3.14. Se mira para ello sólo en las configuraciones o cromosomas que hay en la población del algoritmo genético tras terminar la primera fase. Para cada cromosoma propagaremos con el primer sistema de mensajes las observaciones de las variables transparentes correspondientes al cromosoma y con el segundo haremos lo mismo, pero además incluiremos la observación del caso u_j^l de la variable X_j. De esta forma usando la fórmula 3.9 podemos obtener dos vectores 'a posteriori' $PS^1_{T_i}$ y $PS^2_{T_i}$ para cualquier variable transparente usando el primer y segundo sistema de mensajes, pero donde se ignora el último valor observado para dicha variable transparente. Para ello, se mantienen las valuaciones observación de las variables transparente por separado de las valuaciones del grupo donde se insertan, al igual que en los algoritmos que explicamos en la sección 3.5. Para cada caso c_{i_k} de T_i, $PS^1_{T_i}(c_{i_k}))$ nos da $P_{c_1,\ldots,c_{i_k},\ldots,c_n}(e)$ y $PS^2_{T_i}(c_{i_k})$ nos da $P_{c_1,\ldots,c_{i_k},\ldots,c_n}(e \cap (X_j = u_j^l)$. O sea, $PS^1_{T_i}(c_{i_k}) = t_1 + t_2$ y $PS^2_{T_i}(c_{i_k}) = t_1$. El valor t_2 de la expresión 3.14 puede calcularse con la fórmula:

$$t_2 = P(e) - t_1 = PS^1_{T_i}(c_{i_k}) - PS^2_{T_i}(c_{i_k}) \tag{3.15}$$

Una vez que hemos conseguido la configuración de variables transparentes que maximizan la función 3.14 el algoritmo anterior aplica un algoritmo iterativo. En este algoritmo se hace un recorrido en profundidad del árbol de grupos. En este recorrido cada vez que visitamos un grupo cambiamos la configuración de sus variables transparentes para obtener mejores valores para la función de la expresión 3.14. El esquema de este algoritmo iterativo puede verse a continuación:

Algoritmo 3.11 Algoritmo de mejora iterativa del punto de la población del algoritmo genético que maximiza la expresión 3.14

1. Seleccionar un grupo raíz C_0 en el árbol de grupos.

2. Repetir mientras que se sigan obteniendo mejores configuraciones:

   - `ObtenerMejorConf`(C_0,NULL)

■

`ObtenerMejorConf` es un procedimiento recursivo que describimos a continuación:

Algoritmo 3.12 ObtenerMejorConf(C_i,C_j)

1. Para todos los grupos vecinos C_k de C_i, excepto C_j
   (a) Para todas las variables transparentes T_{i_j} de C_i
      i. Calcular $PS^1_{T_{i_j}}$ y $PS^2_{T_{i_j}}$ sin tener en cuenta la última observación para la variable T_{i_j}.
      ii. Seleccionar el caso c_{max} de T_i que maximice la expresión 3.14.
      iii. Añadir al grupo T_{i_j} la valuación observación correspondiente al caso c_{max}
      iv. Si X_j está en C_i, calcular el punto del conjunto convexo 'a posteriori' de X_j.
      v. Enviar mensajes al grupo vecino C_k con los dos sistemas de mensajes.
      vi. `ObtenerMejorConf`(C_k,C_i)
2. Para todas las variables transparentes T_i de C_i
   (a) Calcular $PS^1_{T_{i_j}}$ y $PS^2_{T_{i_j}}$ sin tener en cuenta la última observación para la variable T_{i_j}.
   (b) Seleccionar el caso c_{max} de T_i que maximice la expresión 3.14.
   (c) Añadir al grupo C_i la valuación observación correspondiente al caso c_{max} de la variable transparente T_{i_j}
3. Si X_j está en C_i, calcular el punto del conjunto convexo 'a posteriori' de X_j.
4. Enviar mensajes $M_{C_i \to C_j}$ al grupo C_j con los dos sistemas de mensajes.

■

3.7 Conclusiones

En este capítulo se ha mostrado como transformar el problema de la propagación de conjuntos convexos de probabilidades en un problema de optimización combinatoria. Una vez transformado el problema de la propagación en un problema de optimización combinatoria se han usado básicamente dos técnicas para resolverlo. La primera ha sido la del enfriamiento simulado. Se han estudiado varios algoritmos que usan esta técnica. Cada uno de ellos estaba

enfocado a obtener una clase de puntos importantes de los conjuntos convexos 'a posteriori'. Los experimentos realizados para estos algoritmos han mostrado que se obtienen buenos resultados. Además ha sido posible combinar los distintos algoritmos implementados para obtener los puntos que consideramos importantes en el conjunto convexo 'a posteriori'. La segunda técnica aplicada ha sido la de los algoritmos genéticos. Esta técnica también se ha mostrado que funciona bien con los experimentos realizados. Una mejora que ha sido implementada, ha sido la de la combinación de los algoritmos genéticos con un algoritmo de mejora iterativa con la intención de obtener todos los puntos extremos del conjunto convexo 'a posteriori'.

Capítulo 4

Árboles de Probabilidad e Independencias Asimétricas

4.1 Introducción

Recientemente han sido propuestos diferentes métodos para aprovechar las *independencias asimétricas* de una red causal y conseguir una inferencia más eficiente. Uno de estos métodos hace uso de *árboles de probabilidad* [10, 143, 121, 107, 52, 85]. En este capítulo, se propone trabajar directamente con los árboles de probabilidad asociados a cada potencial probabilístico describiendo cómo se pueden implementar los algoritmos de propagación en estructuras gráficas usando árboles de probabilidad en lugar de matrices para representar las probabilidades condicionadas. Esto permite además el diseño de algoritmos aproximados, en los que el tamaño de un potencial tiene un límite fijado de antemano.

En [10] se propone una forma de capturar las independencias asimétricas mediante *árboles de probabilidad*. Sin embargo, en este trabajo [10] el mayor énfasis se ha puesto en aspectos de representación sin proponer métodos para realizar cálculos con potenciales representados por árboles de probabilidad, aunque sí que estudian cómo aprovechar los árboles para realizar cálculos más eficientemente con un método indirecto en el que transforman la red original añadiendo nuevas variables de manera que aparecen más potenciales, pero más pequeños que los originales. De forma independiente Cano y Moral [16] y Kozlov y Koller [85] han dado algoritmos para calcular directamente con árboles de probabilidad. En Friedman y Goldszmidt [52] se da un método de aprendizaje de los árboles de probabilidad a partir de una base de

datos utilizando medidas de entropía condicional. Nosotros utilizaremos un método parecido al que proponen estos autores a la hora de construir los árboles a partir de las tablas de probabilidad.

En este capítulo se propone un algoritmo para construir los árboles de probabilidad a partir de los potenciales y se describen como realizar las operaciones de combinación y marginalización directamente para potenciales representados por árboles de probabilidad. Esto permite propagar árboles directamente. Las ventajas que se consiguen son que se obtiene un método más simple y flexible que el propuesto por Boutilier y otros [10], ya que permite adaptar la estructura de los árboles de forma dinámica de acuerdo con las observaciones, con lo que se puede ganar en eficiencia. Esta adaptación equivaldría en el método de Boutilier a cambiar el árbol de grupos. También nuestro método permite el diseño de operaciones aproximadas, trabajando con árboles de un tamaño limitado a un determinado número máximo de nodos.

El capítulo se organiza de la siguiente forma. En la sección 4.2 se introducen los árboles de probabilidad. En la sección 4.3.1 se describe cómo construir un árbol de probabilidad a partir de un potencial en forma de tabla. En la sección 4.3.2 se describen los algoritmos para realizar la combinación y marginalización directamente con árboles de probabilidad tanto de una forma exacta como de una forma aproximada. En la sección 4.4 se presentan varios experimentos realizados con los algoritmos propuestos que producen aproximaciones de la probabilidad 'a posteriori' de una determinada variable. Finalmente en la sección 4.5 se dan las conclusiones del capítulo.

4.2 Independencias asimétricas y árboles de probabilidad

Como vimos en el capítulo 2, una red causal permite expresar relaciones de independencia entre las variables de la red a través del criterio de la d-separación. Estas independencias permiten conseguir un ahorro en la representación de la distribución de probabilidad conjunta, que puede calcularse con la multiplicación de las distribuciones de probabilidad condicional representadas por cada nodo. Los arcos en la red causal representan relaciones directas entre las variables. La ausencia de arcos entre variables indica la existencia de independencia. De forma más concreta, se dice que las variables Z e Y son independientes dado un conjunto de variables X_I si $P(Z|X_I, Y) = P(Z|X_I)$ para todos los valores de X_I, Y y Z. La red causal codifica la siguiente sentencia de independencia: la variable correspondiente a un nodo de la red causal

es independiente de las variables correspondientes a sus nodos no descendientes dado el estado de los padres [106].

Si tenemos una red causal para el conjunto de variables $\{X_1, \ldots, X_n\}$ donde cada variable X_i toma valores en el conjunto U_i, tendremos una distribución de probabilidad condicional para cada variable X_i dados los padres $pa(X_i)$: $P(X_i|pa(X_i))$. Vimos en el capítulo 2 que la red causal representa la siguiente distribución de probabilidad sobre $\{X_1, \ldots, X_n\}$:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|pa(X_i)) \tag{4.1}$$

Tradicionalmente las distribuciones de probabilidad condicional han sido representadas en forma de tabla mediante un vector de números reales. Sea $U_I = \Pi_{i \in I} U_i, I \subseteq \{1, \ldots, n\}$. Siguiendo la terminología de Shafer y Shenoy [120], llamaremos *potencial* a una función definida del conjunto U_I en $\mathbb{R}_0^+$. Podemos decir entonces que cada distribución de probabilidad condicional $P(X_i|pa(X_i))$ es un potencial h_i definido sobre U_I donde $I = \{pa(X_i) \cup X_i\}$. Para guardar este potencial en forma de matriz necesitaremos $|U_I|$ números reales. Por ejemplo en la red causal de la figura 4.1 si consideramos que todas las variables pueden tomar sólo dos valores se necesitarían 8 valores en la tabla de la distribución condicional de $P(X_1|X_2, X_3)$. Así pues, el tamaño de esta representación es exponencial en el número de parámetros (padres de la variable) de la distribución.

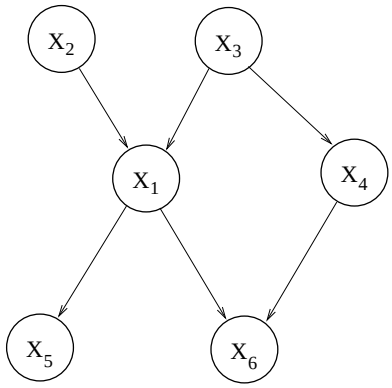


Figura 4.1: Ejemplo de una red causal

A veces se presentan regularidades en las distribuciones de probabilidad que podrían aprovecharse para utilizar una representación más compacta de las tablas. Estas regularidades en las tablas generan cierto tipo de independencias entre las variables de la tabla, que se conocen como *independencias asimétricas* o *basadas en el contexto*.

Un *árbol de probabilidad* es un árbol etiquetado, donde cada nodo interior representa una variable y cada uno de los nodos hoja representa un número real. Cada nodo interior tendrá tantos nodos hijos como posibles casos pueda tomar la variable que representa dicho nodo. Los árboles podremos usarlos para representar distribuciones de probabilidad (potenciales) en cuyo caso los nodos hoja corresponden a probabilidades. Veamos una definición formal de árbol de probabilidad.

Definición 4.1 Árbol de probabilidad
Un árbol de probabilidad sobre las variables X_I es un árbol etiquetado, $\mathcal{T}$, tal que:

- *Cada nodo interior está etiquetado con una variable $X_i \in X_I$*
- *Cada nodo interior tiene tantos hijos como casos tome la variable que representa.*
- *Cada nodo hoja está etiquetado con un número $r \in \mathbb{R}_0^+$*

Definición 4.2 Camino, etiquetado de un árbol y camino consistente
Un camino en un árbol de probabilidad es el conjunto de arcos que hay entre el nodo raíz del árbol y un nodo hoja l. El etiquetado de un camino es la configuración de las variables que hay en el camino de la raíz al nodo hoja. Un camino es consistente con un contexto c si y sólo si el etiquetado del camino es consistente con la asignación de valores de c. Denotaremos con $Anc(l)$ al conjunto de variables que hay desde la raíz al nodo hoja l.

Definición 4.3 *Decimos que un árbol de probabilidad $\mathcal{T}_f$ sobre las variables X_I representa un potencial $f : U_I \to \mathbb{R}$ si y sólo si para cada $u \in U_I$ el valor de $f(u)$ es el número que etiqueta el nodo hoja correspondiente al camino consistente con el contexto u. Para obtener $f(u)$ en el árbol de probabilidad comenzamos por el nodo raíz y vamos seleccionado para cada nodo interior X_i la rama correspondiente al valor que toma X_i en la configuración u hasta llegar a un nodo hoja.*

Esto significa que para cada $u_I \in U_I$, si seguimos la rama que en cada variable X_i elige el camino u_i, entonces encontraremos $f(u_I)$ en el nodo hoja al que llegamos en el árbol.

Cuando $\mathcal{T}_f$ representa un potencial f diremos que $\mathcal{T}_f$ está asociado a f. En tal caso denotaremos como $\mathcal{T}_f(x)$ al valor de $f(x)$.

Veamos la definición de *independencia asimétrica* o *basada en el contexto*.

Definición 4.4 Independencia basada en el contexto [10]
Sean A, B y C conjuntos disjuntos de variables. Sea P una distribución de probabilidad conjunta sobre las variables A, B y C. Decimos que A y B son independientes dado el contexto $c \in U_C$, denotado como $I_c(A, B)$, si $P(A|B = b_1, C = c) = P(A|B = b_2, C = c)$ para todos los $b_1, b_2 \in U_B$ que verifican $P(B = b_1, C = c) > 0$ y $P(B = b_2, C = c) > 0$.

Esta definición es más débil que la que se usa en la codificación de los grafos dirigidos acíclicos, ya que se ha de verificar sólo para algunos de los valores de C y no para todos. Veamos un ejemplo que ilustra las independencias basadas en el contexto.

Ejemplo 4.1 Independencia basada en el contexto
Supongamos que la distribución de probabilidad $P(X_1|X_2, X_3)$ en la red causal de la figura 4.1 viene dada con la tabla de la figura 4.2. Hemos supuesto que cada variable (X_1, X_2 y X_3) toma dos posibles valores. Cuando $X_2 = 2$ la probabilidad condicional es $0,6$ independientemente del valor de X_3. Bajo el contexto de $X_2 = 2$, X_1 y X_3 se vuelven independientes, pero son dependientes bajo el contexto $X_2 = 1$. Esta independencia basada en el contexto es aprovechada por el árbol de probabilidad de la figura 4.2 para representar la misma distribución de probabilidad, pero ahora usando únicamente 5 valores en lugar de 8 como en la tabla. Puede comprobarse con este ejemplo que no estarán todas las variables de la distribución de probabilidad en todos los caminos del árbol. Cuando falten variables en un camino nos está indicando que tenemos una independencia basada en el contexto, pues para todos los valores que puede tomar esa variable ausente tenemos que la distribución de probabilidad toma el mismo valor.

X1	X2	X3	P(X1 |X2 X3)
1	1	1	.3
1	1	2	.5
1	2	1	.4
1	2	2	.4
2	1	1	.7
2	1	2	.5
2	2	1	.6
2	2	2	.6

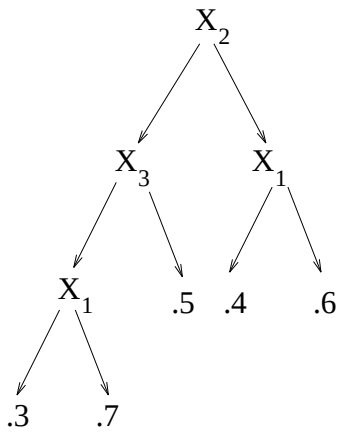


Figura 4.2: Distribución condicional: tabla y árbol

■

La representación de potenciales mediante árboles de probabilidad permite capturar las independencias basadas en el contexto de forma cualitativa de forma similar a como una red causal captura las independencias condicionales en forma cualitativa. Por ejemplo, en la figura 4.3 podemos ver una red causal, la tabla correspondiente a la distribución condicional $P(X|U,V,W)$ y un árbol para esta tabla. Examinando el árbol se puede deducir la independencia contextual $I_c(X;V,W|U=1)$.

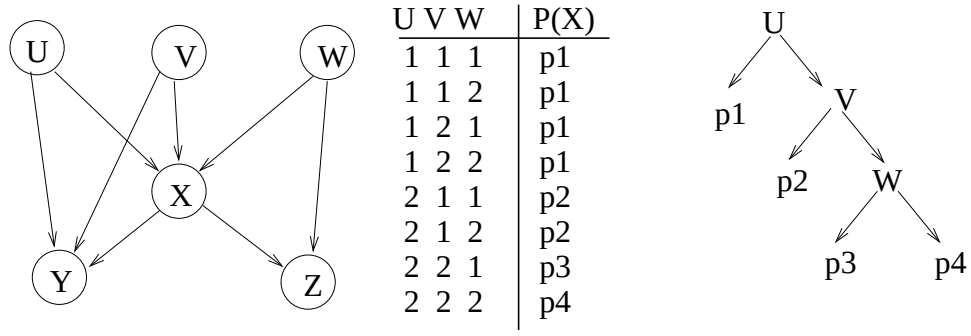


Figura 4.3: Red causal, distribución condicional en forma de tabla y árbol asociado

4.3 Uso de árboles de probabilidad en la propagación en redes causales

Podemos conseguir notables mejoras en la inferencia en redes causales si usamos los árboles asociados a cada distribución de probabilidad condicional, en lugar de los potenciales correspondientes a la tabla completa de cada distribución de probabilidad condicional. En el peor caso el tamaño del árbol tendrá la misma cantidad de números reales que el potencial, pero normalmente existirán independencias asimétricas que reducirán el tamaño de dicho árbol, consiguiendo que la inferencia sea más eficiente.

Para poder usar los árboles en la propagación necesitamos los siguientes algoritmos:

- Algoritmo para construir el árbol de probabilidad a partir de una tabla.
- Algoritmo para llevar a cabo la marginalización (eliminación de variables del árbol).

- Algoritmo para efectuar la combinación de dos árboles de probabilidad obteniendo un nuevo árbol.

4.3.1 Construcción del árbol

Los árboles nos van a servir para representar cualquier potencial, y no solamente las distribuciones de probabilidad condicional.

Cuando una variable tiene un gran número de padres en un grafo dirigido acíclico, es poco natural especificar una distribución de probabilidad para cada configuración de las variables padre. Usualmente muchas distribuciones coinciden y una representación directa mediante un árbol de probabilidad es más adecuada. Para un experto que tenga que proporcionar las probabilidades de una distribución le resulta más simple cuando puede dar un árbol. Por ejemplo, en la figura 4.4 vemos que el experto debería dar muchos menos valores para una distribución condicional correspondiente a una puerta OR, usando un árbol que si directamente diese los valores de todas las configuraciones de la tabla. Sin embargo, en algunas ocasiones es posible que dispongamos de una matriz de probabilidad y nos interese construir un árbol, o bien, que tengamos que construir un árbol que sea el resultado de alguna operación con árboles ya existentes.

A B C D	P(E|ABCD)
0 0 0 0	1.0
0 0 0 1	0.0
0 0 1 0	0.0
0 0 1 1	0.0
0 1 0 0	1.0
0 1 0 1	1.0
0 1 1 0	1.0
0 1 1 1	1.0
1 0 0 0	0.0
1 0 0 1	0.0
1 0 1 0	0.0
1 0 1 1	0.0
1 1 0 0	0.0
1 1 0 1	0.0
1 1 1 0	0.0
1 1 1 1	0.0

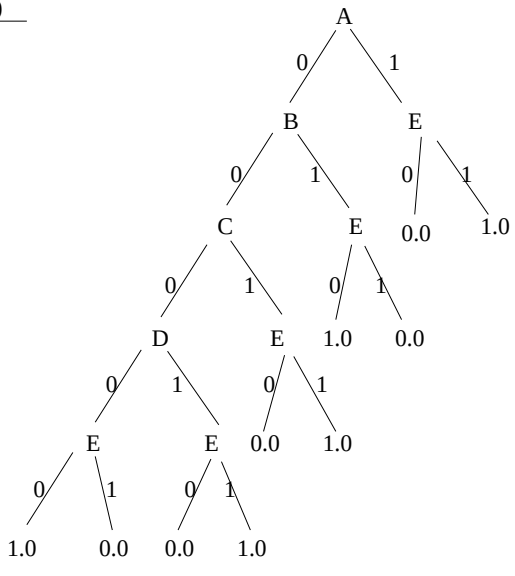


Figura 4.4: Distribución condicional correspondiente a una puerta OR

En general, el problema que nos vamos a plantear es el siguiente. Sea h un potencial definido

para un conjunto de variables X_I. Nuestro propósito será construir un árbol $\mathcal{T}$ que represente el potencial h. No existe un único árbol $\mathcal{T}$ asociado al potencial h, ya que al construir el árbol podemos etiquetar la raíz con cualquier variable del potencial h. A su vez cada uno de los hijos de este nodo raíz podría etiquetarse con cualquiera del resto de las variables. Incluso cada hijo podría etiquetarse con una variable distinta.

Ejemplo 4.2 Construcción de un árbol

En la figura 4.5 podemos ver el proceso de construcción del árbol de la figura 4.2. El orden en que se han realizado las distintas etapas tampoco tiene porqué ser fijo. Detallemos lo que se va haciendo en cada fase:

- *En la fase 1 se crea el nodo raíz, y se decide que será etiquetado con X_2, por lo que se le añaden además dos nodos hijos que ahora mismo están vacíos.*
- *En la fase 2 se decide expandir el árbol por el hijo izquierdo de X_2, y se decide que sea la variable X_3 la que etiquete este nodo.*
- *En la fase 3 se expande el árbol por el hijo izquierdo de X_3 etiquetando este hijo con X_1.*
- *En la fase 4 etiquetamos los nodos hijos de X_1 con números reales, ya que ya hemos usado todas las variables del potencial en el camino de estos nodos hijos de X_1. En estos nodos almacenamos las probabilidades del potencial correspondientes a la configuración que nos proporcionan los caminos que van del raíz a estos dos nodos hoja. También se etiqueta el hijo derecha de X_3 con otro número real. En este caso era posible utilizar la variable X_1 para expandir este nodo, pero no hace falta pues el potencial tomará el mismo valor (0,5) para $X_1 = 1$ y $X_1 = 2$ cuando $X_2 = 1$ y $X_3 = 2$.*
- *Finalmente en la fase 5 se etiqueta el hijo derecha de X_2 con X_1. Por motivo similar al de la fase anterior ahora no se siguen expandiendo los dos nodos hijos de X_1, pues en ambos casos obtenemos idénticos valores en el potencial.*

■

En el proceso de construcción el objetivo es que el árbol resultado sea lo más pequeño posible. El orden en que aparecen las variables en los distintos caminos puede influir en el

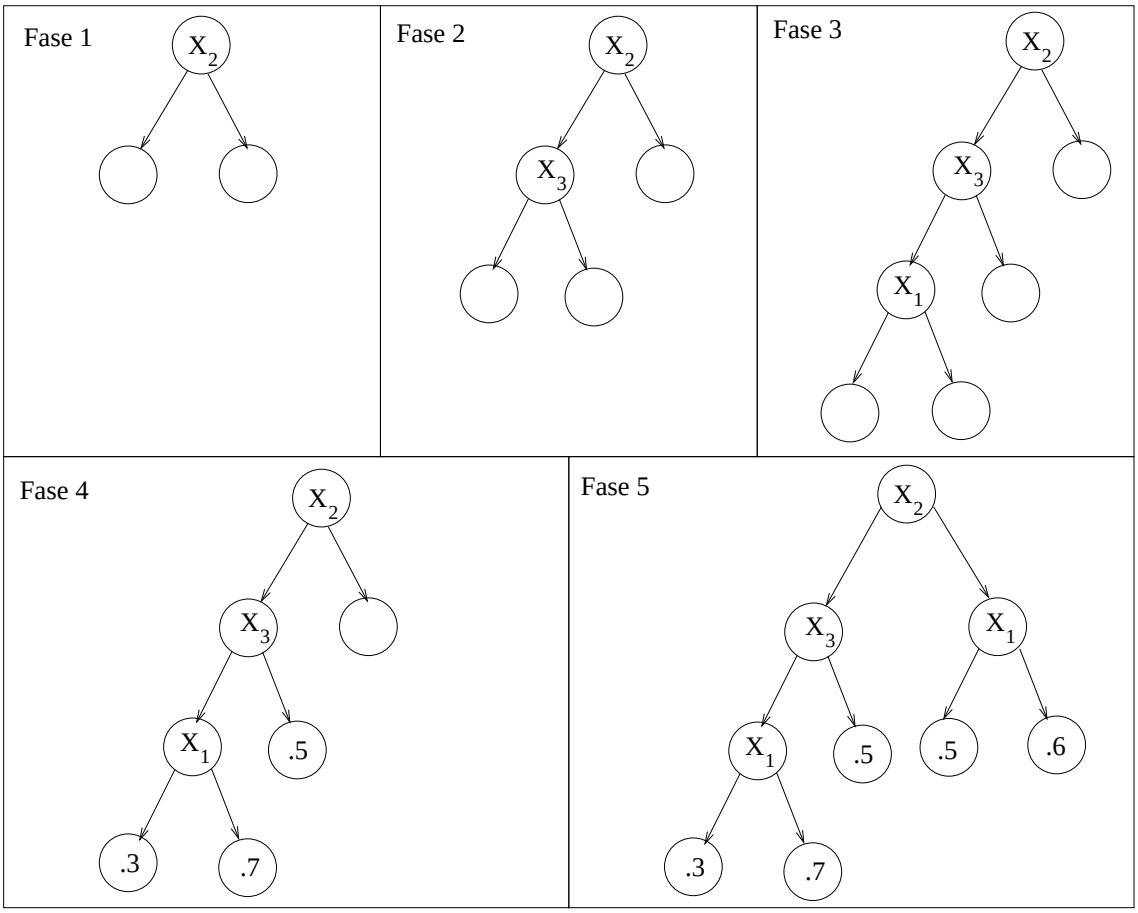


Figura 4.5: Proceso de construcción del árbol de la figura 4.2

tamaño final que tenga el árbol. Por ejemplo, en la figura 4.6 podemos ver otro posible árbol para la misma distribución condicional de la figura 4.2. En este caso el árbol utiliza 7 números reales, por lo que es menos óptimo que el primero.

Algunas veces, por límites de memoria o tiempo no podremos trabajar con representaciones exactas y tendremos que conformarnos con un árbol que aproxime el potencial h. O sea, puede que la tabla que representa el potencial h sea de un tamaño enorme que haga imposible representarlo en un ordenador. En tal caso hará falta el uso de representaciones que aproximen el potencial exacto. Esto se puede conseguir con árboles de probabilidad. Podremos tener árboles tan grandes como permita el ordenador en el que estemos trabajando. Cuando mayores sean estos árboles, mejor aproximarán el potencial exacto. En general el grado de aproximación de un árbol $\mathcal{T}$ a un potencial h lo calcularemos con la *entropía de Kullback-Leibler* [86] de $p_{\mathcal{T}}$

X1	X2	X3	P(X1 |X2 X3)
1	1	1	.3
1	1	2	.5
1	2	1	.4
1	2	2	.4
2	1	1	.7
2	1	2	.5
2	2	1	.6
2	2	2	.6

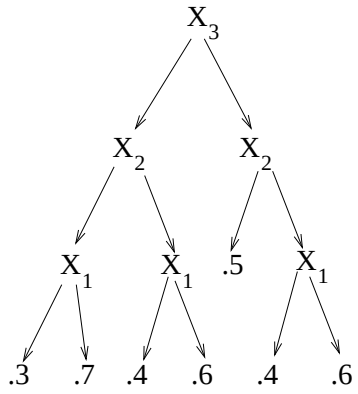


Figura 4.6: Otro posible árbol de probabilidad para la distribución condicional

a p_h donde $p_{\mathcal{T}}$ y p_h representan las distribuciones de probabilidad proporcionales a $\mathcal{T}$ y h respectivamente, esto es:

$$D(h,\mathcal{T}) = D(p_h, p_{\mathcal{T}}) = -\sum_{u_I \in U_I} p_h(u_I) \log \frac{p_h(u_I)}{p_{\mathcal{T}}(u_I)}$$

Definición 4.5 *Dos árboles se dice que tienen la misma estructura si contienen los mismos nodos interiores y sólo difieren como mucho en los números situados en las hojas.*

Definición 4.6 *Si $\mathcal{T}$ es un árbol y X_J es un conjunto de variables, denotaremos por $T^{R(X_J=u_J)}$ al árbol que se obtiene sustituyendo en $\mathcal{T}$ todas las variables $X_k \in X_J$ por los subárboles T_k hijos de X_k correspondientes a $X_k = u_k$*

Cuando X_J contiene todas las variables en el camino hasta una hoja l entonces $T^{R(X_J=u_J)}$ representa el valor almacenado en dicha hoja. Sea $X_J = Anc(l)$, entonces la hoja viene determinada por un valor $X_J = u_J$ que nos indica el hijo u_j que debemos de elegir en cada nodo interior desde la raíz para llegar a dicho nodo hoja.

Por ejemplo en el árbol de la figura 4.2 $\mathcal{T}^{R(X_1=1)}$ nos produce el árbol a) de la figura 4.7, $\mathcal{T}^{R(X_1=1,X_2=1)}$ nos produce el árbol b) de la figura 4.7 y finalmente $\mathcal{T}^{R(X_1=1,X_2=1,X_3=2)}$ nos produce el árbol c) de la figura 4.7 que en este caso es únicamente un nodo hoja con un valor real.

Lo primero que demostraremos será un resultado similar al dado por Boutilier et al. [10] bajo hipótesis ligeramente distintas.

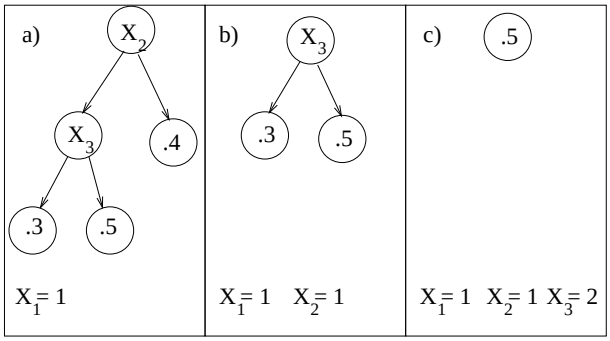


Figura 4.7: Árbol de la figura 4.2 restringidos a diferentes configuraciones

Proposición 4.1 *Sea h un potencial sobre un conjunto de variables X_I. Si $\mathcal{T}$ es un árbol en el que para todo nodo hoja l el valor que etiqueta este nodo hoja es $\sum_{u_I|u_I^{\downarrow J}=u_J} h(u_I)/|U_{I-J}|$ con $X_J = Anc(l)$ y además los valores del potencial asociado a $\mathcal{T}$ suman lo mismo que los valores del potencial h , entonces $\mathcal{T}$ minimiza la distancia de Kullback-Leibler entre todos los árboles con la misma estructura que $\mathcal{T}$ y el potencial h.*

Demostración:

Si particionamos U_I en una serie de subconjuntos A_j, tantos como nodos hoja tenga el árbol $\mathcal{T}$. Entonces, la distancia de Kullback-Leibler entre el potencial asociado al árbol $\mathcal{T}$ y h puede ponerse como:

$$\sum_{A_j} \left(\sum_{u_I \in A_j} h(u_I) \log \frac{h(u_I)}{q_j} \right)$$

donde q_j es el valor que toma el potencial asociado al árbol $\mathcal{T}$ en todas las configuraciones de la partición A_j, que evidentemente son el mismo, ya que cada q_j corresponde a un nodo hoja de $\mathcal{T}$.

En la proposición buscamos los valores q_j con los que etiquetar los nodos hoja de $\mathcal{T}$ que minimicen la anterior distancia. Entonces nuestro objetivo es encontrar que valores q_j hay que utilizar para conseguir:

$$\min \sum_{A_j} \sum_{u_I \in A_j} h(u_I) \log \frac{h(u_I)}{q_j}$$

La anterior expresión es equivalente a:

$$\min \sum_{A_j} \sum_{u_I \in A_j} h(u_I) \log h(u_I) - \sum_{A_j} \sum_{u_I \in A_j} h(u_I) \log q_j$$

Como la primera parte no depende de los q_j entonces es constante respecto a los q_j y por tanto

podemos buscar el mínimo de la anterior expresión de forma equivalente con:

$$\min \; -\sum_{A_j} \sum_{u_I \in A_j} h(u_I) \log q_j$$

Puesto que q_j es igual para todos los u_I de la misma partición A_j, podemos sacar el logaritmo fuera de la sumatoria. Además si denotamos por $h(A_j)$ a $\sum_{u_I \in A_j} h(u_I)$ entonces la anterior expresión quedaría:

$$\min \; -\sum_{A_j} h(A_J) \log q_j$$

Evidentemente $h(A_j) = \sum_{u_I \in A_j} \frac{h(A_j)}{|A_j|}$. Entonces el mínimo quedaría como:

$$\min \; -\sum_{A_j} \sum_{u_I \in A_j} \frac{h(A_j)}{|A_j|}) \log q_j$$

Si denotamos con $j(u_I)$ al índice de la partición en la que está incluida la configuración u_I, entonces podemos transformar la anterior expresión de la siguiente forma:

$$\min \; -\sum_{u_I \in U_I} \frac{h(A_{j(u_I)})}{|A_{j(u_I)}|}) \log q_{j(u_I)}$$

El lema de Gibbs nos dice que:

Dados dos sistemas de números $p_1, p_2, \ldots, p_n$ *y* $q_1, q_2, \ldots, q_n$ *no negativos y tales que*:

$$\sum_{i=1}^{n} p_i = \sum_{i=1}^{n} q_i$$

entonces se verifica

$$-\sum_{i=1}^{n} p_i \log p_i \leq -\sum_{i=1}^{n} p_i \log q_i$$

con la igualdad si y sólo si es $p_i = q_i, \forall i$.

Entonces, según este lema, nuestro mínimo se obtendrá cuando los $q_{j(u_I)}$ se obtienen como:

$$q_{j(u_I)} = \frac{h(A_{j(u_I)})}{|A_{j(u_I)}|}$$

O sea, que cada q_j se obtiene como la media de los valores del potencial h incluidos en la partición A_j, que es lo que queríamos demostrar. ■

Según esta proposición, dada una estructura de árbol $\mathcal{T}^*$, la mejor aproximación al potencial h con esta estructura se consigue etiquetando cada nodo hoja con la media de los valores de h para las configuraciones de las variables de h consistentes con este nodo hoja.

Por ejemplo el árbol b) de la figura 4.8 es el que mejor aproxima el potencial que representa el árbol a) de la misma figura de entre los árboles que tienen la estructura del árbol b).

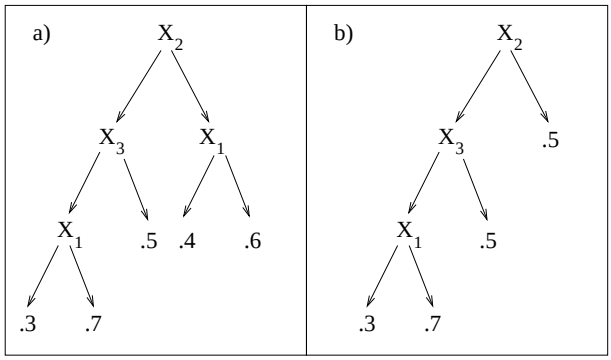


Figura 4.8: Aproximación de un potencial mediante un árbol

El problema es que, en general, la construcción de una estructura óptima $\mathcal{T}^*$ con un tamaño máximo dado, es un problema de optimización combinatoria, difícil de resolver. La metodología para la construcción de un árbol que vamos a usar, está basada en los procedimientos clásicos de aprendizaje como **ID3** [110]. La diferencia está en que ahora no existe una variable de clasificación prefijada. En principio, podría pensarse que la variable hija X_i en la distribución condicional $P(X_i|pa(X_i))$, es una variable de clasificación. Sin embargo, hay que tener en cuenta que no siempre usaremos el árbol para estimar las probabilidades de esta variable X_i y, a veces, pretenderemos estimar las probabilidades de las variables padre $pa(X_i)$, o de algunas otras conectadas con éstas en el grafo de la red bayesiana. Además, después de realizar operaciones de combinación y marginalización, los árboles ya no representarán la probabilidad condicionada de una variable respecto a otras. En algunos casos tendremos probabilidades condicionadas de varias variables, o verosimilitudes, o tendrán otro significado más complejo.

El procedimiento de construcción del árbol de probabilidad será paso a paso, tratando de obtener en cada momento la mejor aproximación posible de h, (aunque esto no garantice la mejor aproximación global al final del proceso). Se partirá de una estructura de árbol inicialmente vacía, y en cada paso se añadirá una variable a uno de los nodos hoja en la estructura actual. Supongamos que hasta ahora hemos generado una estructura $\mathcal{T}^*$ y tenemos

que decidir que nodo hoja de $\mathcal{T}^*$ ramificamos y qué variable usamos en la ramificación. Un nodo hoja l de $\mathcal{T}^*$ vendrá definido por los valores de ciertas variables $X_J = u_J$, donde X_J son las variables que hay en el árbol en el camino hasta el nodo hoja l ($Anc(l)$). Sea $\mathcal{T}^*(X_J = u_J, X_k)$ la estructura que se obtiene aumentando $\mathcal{T}^*$ al ramificar con la variable X_k la hoja definida por $X_J = u_J$. Elegiremos aquella variable X_k que minimice la distancia a h, es decir

$$D(\mathcal{T}(X_J = u_J, X_k), h) = \min_{l' \in L(\mathcal{T}^*)} \{D(\mathcal{T}(X_{J'} = u_{J'}, X_{k'}) : X_{J'} = u_{J'}\} \tag{4.2}$$

donde $\mathcal{T}$ es el árbol asociado a la estructura $\mathcal{T}^*$ según la proposición 4.1 y donde $L(\mathcal{T}^*)$ es el conjunto de nodos hoja de $\mathcal{T}^*$, $X_{J'} = u_{J'}$ es la configuración de variables que definen el nodo hoja l' de la estructura de árbol $\mathcal{T}^*$ y $k' \in I - J'$.

La siguiente proposición demuestra que este mínimo se puede calcular de forma sencilla.

Proposición 4.2 *Sea h un potencial definido en el conjunto de variables X_I y $\mathcal{T}^*$ la estructura actual para el potencial h. Entonces el par $(X_J = u_J, X_k)$ (donde $X_J = u_J$ es una configuración de variables que define una hoja de $\mathcal{T}^*$ y $X_k \in X_I - X_J$ una variable del potencial h) que minimiza (4.2) es aquel que maximiza la medida de información*

$$I(X_J = u_J, X_k|h) = S_{X_J=u_J} \cdot (\log|U_k| - \log S_{X_J=u_J}) - E[X_k|X_J = u_J] \tag{4.3}$$

donde

$$S_{X_J=u_J} = \sum_{(X_I)^{\downarrow J}=u_J} h(u_I)$$

$$E[X_k|X_J = u_J] = -\sum_{u_k \in U_k} h^{\downarrow k}(u_k|X_J = u_J) \log h^{\downarrow k}(u_k|X_J = u_J)$$

$$h^{\downarrow k}(u_k|X_J = u_J) = \sum_{\substack{(X_I)^{\downarrow J}=u_J \\ (X_I)^{\downarrow k}=u_k}} h(u_I)$$

Demostración:

El par $(X_J = u_J, X_k)$ que minimiza (4.2) es el que maximiza el decremento de la distancia antes y después de ramificar en $\mathcal{T}^*$ con una variable X_k. Notaremos con $\mathcal{T}$ al árbol asociado a la estructura $\mathcal{T}^*$ según la proposición 4.1 y que asigna a cada hoja l la media de los valores del potencial h compatibles con la configuración proporcionada por $Anc(l)$ en $\mathcal{T}^*$. Llamemos D_0 a la distancia entre el potencial h y el potencial asociado al árbol actual antes de ramificar. Llamemos D_1 a la distancia después de ramificar en el árbol con una variable X_k. Entonces el objetivo es maximizar $D_0 - D_1$. Supongamos que $X_I = \{X_1, X_2, \ldots, X_n\}$, y que $X_I - X_J =$

$\{X_1, X_2, \ldots, X_k\}$, $k \leq n$. Todas las variables de $X_I - X_J$ son candidatas para ser usadas en la ramificación. Sin pérdida de generalidad podemos suponer que X_1 es la variable con la que ramificaremos. En el nodo hoja de $\mathcal{T}$ determinado por la configuración $(X_J = u_J)$, antes de ramificar tendremos una probabilidad calculada como $p = S_{X_J=u_J}/(|U_1| \cdot |U_2| \cdots |U_k|)$. Entonces, podemos poner D_0 como suma de lo que aporta a D_0 la rama de $\mathcal{T}$ correspondiente a la configuración $(X_J = u_J)$ (nodo hoja que vamos a ramificar) y una cantidad C que aportan el resto de las ramas de $\mathcal{T}$:

$$D_0 = C + \sum_{u_I \in U_I \,:\, u_I^{\downarrow J} = u_J} h(u_I) \log \frac{h(u_I)}{S_{X_J=u_J}/(|U_1| \cdot |U_2| \cdots |U_k|)}$$

Si denotamos como $E_{h^{R(X_J=u_J)}}$ la entropía del potencial h restringido a aquellos valores compatibles con la configuración $X_J = u_J$, entonces podemos escribir la anterior expresión como:

$$D_0 = C - E_{h^{R(X_J=u_J)}} + \sum_{u_I \in U_I \,:\, u_I^{\downarrow J} = u_J} h(u_I) \log \frac{|U_1| \cdot |U_2| \cdots |U_k|}{S_{X_J=u_J}}$$

En la anterior expresión el logaritmo se puede sacar fuera de la sumatoria:

$$D_0 = C - E_{h^{R(X_J=u_J)}} + \log \frac{|U_1| \cdot |U_2| \cdots |U_k|}{S_{X_J=u_J}} \sum_{u_I \in U_I \,:\, u_I^{\downarrow J} = u_J} h(u_I)$$

La sumatoria en la anterior expresión corresponde a $S_{X_J=u_J}$. Por tanto:

$$D_0 = C - E_{h^{R(X_J=u_J)}} + \log(\frac{|U_1| \cdot |U_2| \cdots |U_k|}{S_{X_J=u_J}}) \cdot S_{X_J=u_J}$$

Puesto que el logaritmo del producto es la suma de los logaritmos:

$$D_0 = C - E_{h^{R(X_J=u_J)}} + (\log |U_1| + \log |U_2| + \ldots + \log |U_k| - \log(S_{X_J=u_J})) \cdot S_{X_J=u_J}$$

De la misma forma podemos escribir D_1 como suma de lo que aporta $\mathcal{T}$ con la rama correspondiente a la configuración $(X_J = u_J)$ y la cantidad C que aportan el resto de las ramas:

$$D_1 = C + \sum_{u_1} \sum_{u_2 \ldots u_k} h(u_I) \log \frac{h(u_I)}{S_{(X_J=u_J, X_1=u_1)}/(|U_2| \cdots |U_k|)}$$

donde $S_{(X_J=u_J, X_1=u_1)}$ denota la suma de los elementos del potencial h compatibles con la configuración $X_J = u_J$ y $X_1 = u_1$.

Podemos entonces poner D_1 de la siguiente forma:

$$D_1 = C + \sum_{u_1} \sum_{u_2 \ldots u_k} (h(u_I) \log (h(u_I)(|U_2| \cdots |U_k|))) - \sum_{u_1} \sum_{u_2 \ldots u_k} \left(h(u_I) \log \sum_{u_2 \ldots u_k} h(u_I) \right)$$

En la primera sumatoria tomamos logaritmo del producto como suma de los logaritmos:

$$D_1 = C + \sum_{u_1} \left(\sum_{u_2 \ldots u_k} h(u_I) \log h(u_I) + \sum_{u_2 \ldots u_k} h(u_I) \log(|U_2| \cdots |U_k|) \right) -$$

$$- \sum_{u_1} \sum_{u_2 \ldots u_k} \left(h(u_I) \log \sum_{u_2 \ldots u_k} h(u_I) \right)$$

Si denotamos por $h(X_1)$ al potencial que se obtiene sumando para cada $u_1 \in U_1$ todos los $h(X_I)$ compatibles con la configuración $X_J = u_J$ entonces:

$$D_1 = C - E_{h^{R(X_J=u_J)}} + S_{X_J=u_J}(\log |U_2| + \ldots + \log |U_k|) - \sum_{u_1} h(u_1) \log h(u_1)$$

La sumatoria que aparece en la anterior expresión es la que hemos denominado $E(X_1|X_J = u_J)$ en la proposición. Por tanto:

$$D_1 = C - E_{h^{R(X_J=u_J)}} + S_{X_J=u_J}(\log |U_2| + \ldots + \log |U_k|) + E(X_1|X_J = u_J)$$

Calculemos el decremento de distancia, o sea $D_0 - D_1$:

$$D_0 - D_1 = S_{X_J=u_J}(\log |U_1| + \log |U_2| + \ldots + \log |U_k| - \log(S_{X_J=u_J})) -$$

$$- S_{X_J=u_J}(\log |U_2| + \ldots + \log |U_k|) - E(X_1|X_J = u_J)$$

Simplificando, obtenemos lo que queríamos demostrar:

$$D_0 - D_1 = S_{X_J=u_J} \log |U_1| - S_{X_J=u_J} \log(S_{X_J=u_J}) - E(X_1|X_J = u_J)$$

■

En la anterior proposición $S_{X_J=u_J}$ representa la suma de los valores de probabilidad de aquellas configuraciones del potencial h compatibles con la asignación de valores $X_J = u_J$ a las variables del conjunto X_J. Cuanto mayor sea este valor para una variable X_k mayor será la información que aporte tal variable. Cuando comenzamos a construir el árbol, este valor será el mismo para todas las variables. O sea, será la suma de los valores de probabilidad del potencial. Considerando el potencial dado por las configuraciones del potencial h compatibles con la asignación de valores $X_J = u_J$ a las variables del conjunto X_J, $h^{\downarrow k}(u_k|X_J = u_J)$ es un nuevo potencial obtenido a partir de h, con tantos valores como casos tenga la variable X_k,

en el que cada valor $h^{\downarrow k}(u_k)$ se calcula sumando los valores de h correspondientes a $X_k = u_k$ que sean compatibles con la configuración $X_J = u_J$. $E[X_k|X_J = u_J]$ es entonces la entropía no normalizada del anterior potencial $h^{\downarrow k}(u_k|X_J = u_J)$.

En realidad, $I(X_J = u_J, X_k|h)$ mide $D(\mathcal{T}, h) - D(\mathcal{T}(X_J = u_J, X_k), h)$, o sea, mide la distancia desde un árbol $\mathcal{T}$ a un potencial h antes y después de expandir la hoja l correspondiente a la configuración $X_J = u_J$ con la variable X_k. La proposición 4.2 nos dice que debemos de elegir hojas $X_J = u_J$ de las que cuelga una suma grande de probabilidades y una variable X_k con una pequeña entropía. La razón es que ramificando por esta variable se obtienen valores de probabilidad bastante distintos y con valores importantes de probabilidad. Esto parece bastante intuitivo, puesto que estamos interesados en representar sobre todo valores que sean diferentes entre sí. Los valores que sean muy similares se representarán como uno sólo, correspondiente a su media.

En general el algoritmo va construyendo el árbol maximizando esta medida de información nodo por nodo. Cuando la construcción es exacta el algoritmo se para cuando para cada rama $X_J = u_J$ de la estructura construida hasta el momento, los valores de h compatibles con la configuración $X_J = u_J$ son uniformes, es decir $h(X_I) = h(X_{I'})$ si $(X_I)^{\downarrow J} = (X_{I'})^{\downarrow J} = u_J$. A continuación mostramos un esquema de este algoritmo:

Algoritmo 4.1 Algoritmo de construcción del árbol de probabilidad para un potencial h definido en X_I

1. Crear un nodo de árbol inicialmente sin etiqueta.
2. Mientras no se alcance el tamaño máximo para el árbol:
   (a) Buscar aquel nodo hoja l no etiquetado (definido por una configuración $X_J = u_J$) y aquella variable $X_k \in X_I - X_J$ que maximice la medida de información $I(X_J = u_J, X_k|h)$ calculada con la expresión 4.3.
   (b) Etiquetar el nodo hoja l con la variable X_k.
   (c) Crear tantos nodos hoja no etiquetados como casos tenga la variable X_k, y ponerlos como hijos del nodo hoja l anterior.
3. Etiquetar cada nodo hoja del árbol con el valor calculado según la proposición 4.1.

■

Si queremos construir un árbol aproximado, tenemos distintas opciones:

1. Limitar el número de nodos generados a un número prefijado, y parar la generación cuando este número se ha alcanzado. En este caso aplicaremos la proposición 4.2 para etiquetar cada nodo hoja de la estructura generada.

2. Generar el árbol completo y podarlo posteriormente. Si $\mathcal{T}$ es el árbol inicial, la poda consiste en elegir un nodo tal que todos sus hijos sean hojas, eliminar la variable del mismo, sustituirla por el valor medio de los hijos y borrar los nodos hijos. Una poda vendrá definida por una variable X_k y una sucesión de valores $X_J = u_J$ que nos lleva hasta ella en el árbol. Hemos considerado dos posibilidades distintas:

   (a) Limitar previamente el tamaño del árbol resultado. Se van eliminando variables mientras el tamaño del árbol exceda del tamaño prefijado. En cada momento se elije la variable X_k que minimiza la medida de información $I(X_J = u_J, X_k)$. Con esto en cada paso se minimiza el incremento en la distancia al potencial h.

   (b) Ir eliminando todas aquellas variables X_k situadas en nodos $X_J = u_J$ para los que

   $$I(X_J = u_J, X_k) \leq \delta \tag{4.4}$$

   donde δ es un umbral para el incremento de distancia. Se termina cuando no exista ninguna opción que verifique la condición (4.4).

4.3.2 Operaciones con árboles de probabilidad

Debemos definir como se hacen las operaciones de combinación y de marginalización para poder aplicar los algoritmos de propagación y poder calcular probabilidades condicionadas 'a posteriori'. Kozlov y Koller [85] han dado, de forma independiente, también procedimientos para llevar a cabo estas operaciones en árboles, pero ellos toman la estructura de uno de los árboles como base. Luego van ramificando por los nodos hoja de dicho árbol base de acuerdo a la estructura del otro árbol. Sus operaciones de combinación y marginalización requerirán reestructurar el segundo árbol para que tenga la misma estructura del primero. Además ellos utilizan únicamente árboles binarios. A continuación proponemos nuevos algoritmos que mezclan la estructura de los árboles que se quieren combinar o sumar. Con esto se puede conseguir que el resultado de combinar o marginalizar siga siendo un árbol pequeño, donde las variables *más informativas* van a seguir estando en los primeros niveles del árbol. Además

nuestros métodos pueden aplicarse a árboles generales, donde cada nodo puede tener más de dos hijos.

La combinación actúa sobre dos árboles y obtiene un árbol resultado de la combinación de ambos. El algoritmo que construye este árbol es similar al que construye el árbol a partir de una tabla. La marginalización se realiza sobre un árbol y una variable que desea eliminarse, y construye otro árbol como resultado. Cuando haya que eliminar más de una variable, se aplicará este algoritmo sucesivas veces. De nuevo este algoritmo es muy similar al que construye un árbol a partir de una tabla. Estos algoritmos van construyendo el árbol resultado paso a paso sustituyendo en cada paso nodos hoja del árbol que se está construyendo, por nodos etiquetados con variables de la misma manera que en el caso de la construcción del árbol. Al finalizar el proceso tendremos en cada nodo hoja del árbol resultado, el producto de dos números reales en el caso de la combinación, y la suma de $|U_i|$ números reales para el caso de la marginalización, donde $|U_i|$ es el número de valores que toma la variable X_i que estamos eliminando.

4.3.2.1 Combinación de árboles en forma exacta

Veamos en primer lugar en esta sección un algoritmo que construye un árbol correspondiente a la combinación de dos árboles. Este primer algoritmo nos va a construir un árbol que representa de forma exacta el resultado de la combinación. Más adelante veremos como modificar ligeramente este algoritmo para que permita construir árboles aproximados más pequeños en tamaño.

Sea $\mathcal{T}$ un árbol que representa el potencial h para el conjunto de variables X_I, denotaremos por $Var(\mathcal{T})$ al conjunto de variables que etiquetan los nodos interiores de $\mathcal{T}$. Se debe cumplir que $Var(\mathcal{T}) \subseteq X_I$.

Sean $\mathcal{T}_1$ y $\mathcal{T}_2$ dos árboles que representan los potenciales h_1 y h_2 respectivamente. Queremos calcular el árbol $\mathcal{T}$ que representa el potencial $h = h_1 \otimes h_2$.

Veamos con un ejemplo los pasos a seguir para obtener la combinación de dos árboles de probabilidad de forma exacta.

Ejemplo 4.3 Combinación de dos árboles de probabilidad en forma exacta

Supongamos que queremos combinar los dos árboles que aparecen en la figura 4.9. Podemos ver el proceso paso a paso en la figura 4.10.

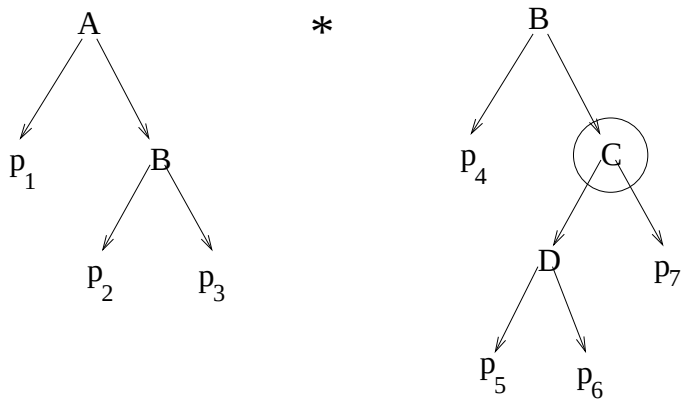


Figura 4.9: Combinación de dos árboles de probabilidad en forma exacta

■

En el algoritmo usaremos una lista $\mathcal{L}_c$ en el que cada elemento contiene tres campos. El primer campo contiene dos subárboles $\mathcal{T}_i$ y $\mathcal{T}_j$ que hay que combinar. Inicialmente la lista contendrá sólo un elemento y este campo contendrá los dos árboles originales $\mathcal{T}_1$ y $\mathcal{T}_2$ que hay que combinar. El segundo campo nos da un nodo hoja no etiquetado del árbol resultado que se está construyendo, que posteriormente en el algoritmo será etiquetado con la variable almacenada en el tercer campo. El tercer campo es una variable X_i perteneciente a $Var(\mathcal{T}_i) \cup Var(\mathcal{T}_j)$ que es la que se usará como etiqueta para el nodo no etiquetado en el segundo campo. Esta variable puede ser cualquier variable de $Var(\mathcal{T}_i) \cup Var(\mathcal{T}_j)$. En el algoritmo que presentamos a continuación se escoge esta variable siguiendo un procedimiento arbitrario. Así pues, los elementos de la lista $\mathcal{L}_c$ nos dan los posibles subárboles a combinar, la etiqueta que pondremos en el raíz del árbol correspondiente a esta combinación, y el nodo hoja del árbol resultado donde colocaremos la combinación de tales subárboles.

En cada etapa del algoritmo, tomaremos un elemento $\{\{\mathcal{T}_i, \mathcal{T}_j\}, l, X_k\}$ de la lista $\mathcal{L}_c$, etiquetaremos el nodo hoja l con la variable X_k almacenada en tal elemento, crearemos tantos nodos hoja como casos tenga X_k haciéndolos hijos del nodo anterior etiquetado con X_k y se insertarán de nuevo en la lista tantos elementos como casos tenga la variable X_k. Cada nuevo elemento que se inserta en la lista $\mathcal{L}_c$ estará compuesto a su vez con tres campos, y corresponde a cada uno de los casos u_k que puede tomar la variable X_k. Los dos árboles del primer campo se calculan como $\mathcal{T}_i^{R\{X_k=u_k\}}$ y $\mathcal{T}_j^{R\{X_k=u_k\}}$. La variable del tercer campo será

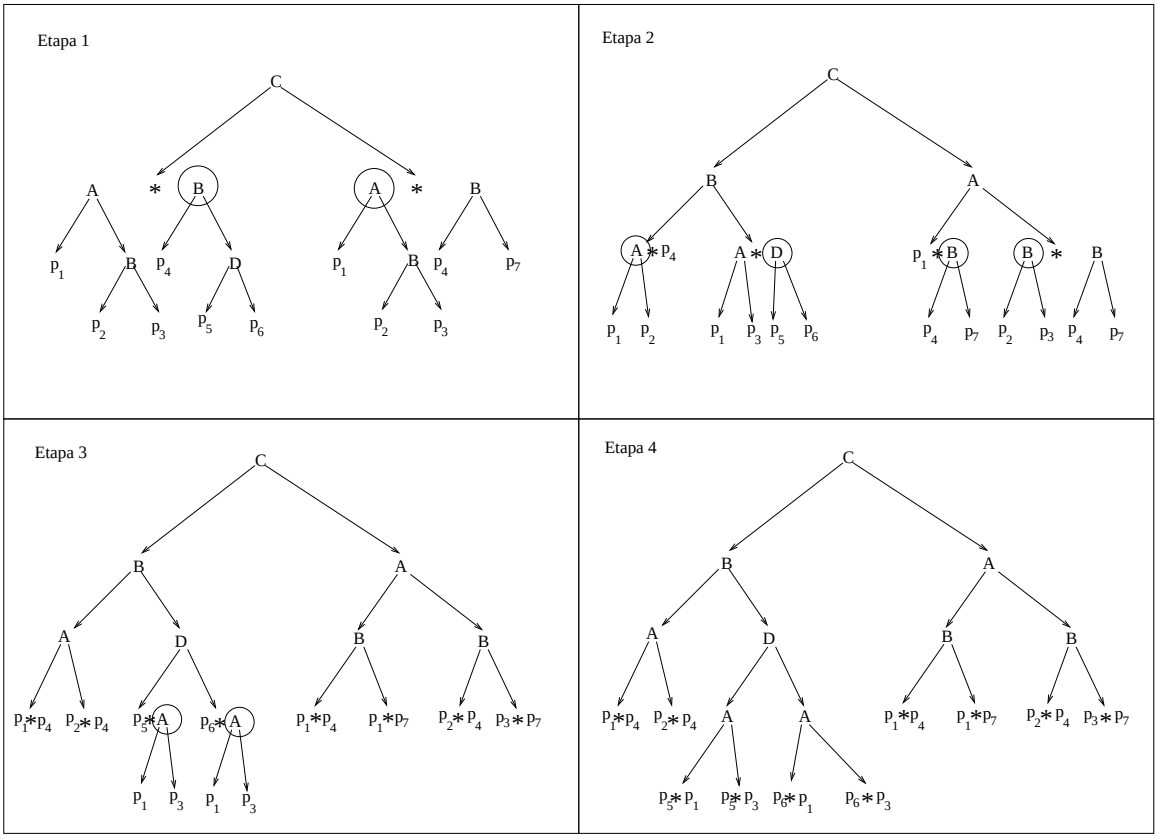


Figura 4.10: Combinación de dos árboles de probabilidad en forma exacta

cualquiera de $Var(\mathcal{T}_i^{R\{X_k=u_k\}}) \cup Var(\mathcal{T}_j^{R\{X_k=u_k\}})$. En caso de que ambos árboles $\mathcal{T}_i^{R\{X_k=u_k\}}$ y $\mathcal{T}_j^{R\{X_k=u_k\}}$ sean árboles correspondientes a un nodo hoja (o sea, que sólo contienen un nodo con un valor numérico) no se insertará en la lista ningún elemento y etiquetaremos el nodo hoja correspondiente a $X_k = u_k$ con el producto de ambos valores numéricos. El algoritmo $Combina(\mathcal{T}_1, \mathcal{T}_2)$ que obtiene $\mathcal{T}$ es el siguiente:

Algoritmo 4.2 Combina$(\mathcal{T}_1, \mathcal{T}_2)$

1. Crear un nodo l de árbol inicialmente sin etiqueta.
2. Seleccionar una variable $X_i \in \{Var(T_1 \cup T_2)\}$
3. Añadir a $\mathcal{L}_c$ el nodo $\{\{T_1, T_2\}, l, X_i\}$
4. Mientras $\mathcal{L}_c$ no esté vacía
   (a) Borrar un nodo $\{\{\mathcal{T}_i, \mathcal{T}_j\}, l, X_k\}$ de $\mathcal{L}_c$

(b) Poner X_k como etiqueta de l.

(c) Para cada $u_k \in U_k$

i. Crear un nodo l' de árbol inicialmente sin etiqueta haciendo que sea hijo del nodo l

ii. Sean L_i y L_j las etiquetas de los nodos $raiz(T_i^{R\{X_k=u_k\}})$ y $raiz(T_j^{R\{X_k=u_k\}})$

iii. Si L_i y L_j son números, sea $L = L_i * L_j$. Pon L como etiqueta de T_h

iv. En caso contrario,

A. Seleccionar una variable $X_l \in \{Var(T_i^{R\{X_k=u_k\}} \cup T_j^{R\{X_k=u_k\}})\}$

B. Añadir a $\mathcal{L}_c$ el nodo $\{T_i^{R\{X_k=u_k\}}, T_j^{R\{X_k=u_k\}}, l', X_l\}$

5. Devolver $\mathcal{T}$

■

4.3.2.2 Marginalización en árboles de forma exacta

Veamos ahora como se haría la operación de marginalización también de forma exacta. Veremos un algoritmo que elimina una sola variable del árbol. La generalización para varias variables es inmediata, llamando al algoritmo de marginalización por cada variable que deseemos eliminar.

Sea $\mathcal{T}$ el árbol de probabilidad que representa el potencial h para el conjunto de variables X_I. Al igual que en el caso de la combinación debe verificarse que $Var(\mathcal{T}) \subseteq X_I$. Queremos calcular el árbol asociado al potencial $h^{\downarrow(I-\{i\})}$ con $i \in I$. O sea, se pretende borrar la variable X_i del árbol $\mathcal{T}$.

El algoritmo de marginalización hará uso de un algoritmo que suma varios subárboles de probabilidad, cuyo funcionamiento es bastante similar al de la combinación. El algoritmo de marginalización hace un recorrido en profundidad partiendo del nodo raíz del árbol original. Cuando visita nodos interiores no hace nada si el nodo está etiquetado con una variable distinta a la variable X_i que se quiere borrar. Si se llega a un nodo interior etiquetado con la variable X_i deberemos sustituir el subárbol que cuelga de este nodo por la suma de los subárboles que cuelgan del nodo. En el recorrido en profundidad del árbol original, en caso que se llegue a un nodo hoja (etiquetado con un número) sin haber pasado por ningún nodo interior etiquetado

con la variable X_i, multiplicaremos el número de tal nodo hoja por el número de casos $|U_i|$ que toma la variable X_i.

Veamos con un ejemplo los pasos a seguir para obtener la marginalización con un árbol de probabilidad de forma exacta.

Ejemplo 4.4 Marginalización en árboles de probabilidad en forma exacta

Supongamos que queremos marginalizar en el árbol de probabilidad de la figura 4.11 al conjunto de variables $\{B, D\}$, o sea que pretendemos eliminar la variable C de dicho árbol. Podemos ver el proceso paso a paso en la figura 4.11.

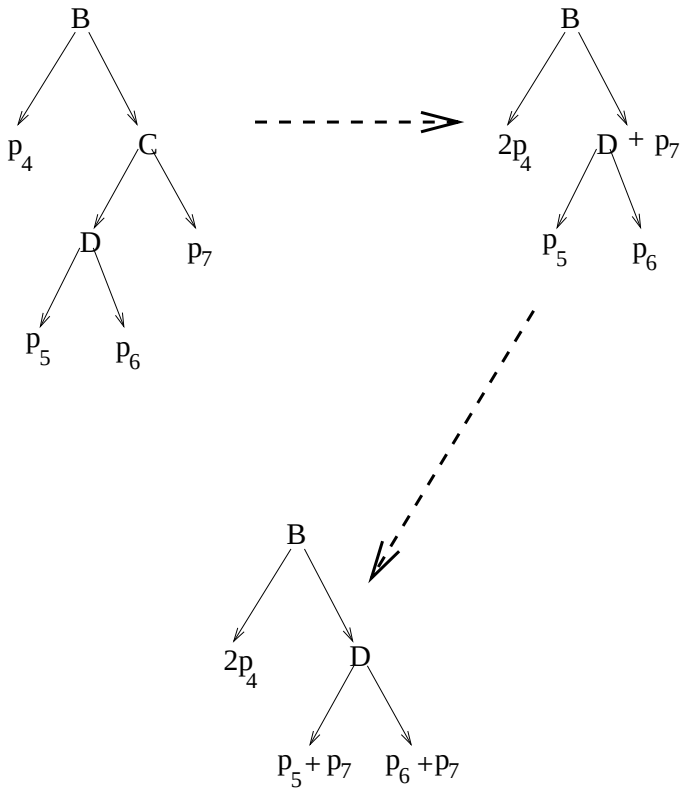


Figura 4.11: Proceso de marginalización de la variable C en forma exacta en un árbol de probabilidad

■

El algoritmo de suma de árboles hará uso de una lista $\mathcal{L}_m$ al igual que el de combinación. Cada elemento de la lista contiene también tres campos. El primer campo contiene los subár-

boles $\mathcal{T}_1, \ldots, \mathcal{T}_j$ que van a sumarse. El segundo campo nos da un nodo hoja no etiquetado del árbol resultado que se está construyendo, que posteriormente en el algoritmo será etiquetado con la variable almacenada en el tercer campo. El tercer campo es una variable X_k perteneciente a $Var(\mathcal{T}_1) \cup \ldots \cup Var(\mathcal{T}_j)$. En el algoritmo que presentamos a continuación se escoge esta variable siguiendo un criterio arbitrario. Por tanto, cada elemento que haya en la lista nos expresa un conjunto de subárboles que deben sumarse, la etiqueta que pondremos en la raíz del árbol suma, y el nodo del árbol resultado de la marginalización que estamos construyendo donde colocaremos el resultado de tal suma.

En cada etapa del algoritmo de suma se toma un elemento $\{\{\mathcal{T}_1, \ldots \mathcal{T}_j\}, l, X_k\}$ de la lista $\mathcal{L}_m$, etiquetaremos el nodo hoja l con la variable X_k, crearemos tantos nodos hoja como casos tenga X_k haciéndolos hijos del nodo anterior etiquetado con X_k y se insertarán de nuevo en la lista tantos elementos como elementos tenga la variable X_k. Cada nuevo elemento que se inserta en la lista $\mathcal{L}_m$ estará compuesto a su vez de tres campos, y corresponde a cada uno de los casos u_k que puede tomar la variable X_k. Los árboles del primer campo se calculan como $\mathcal{T}_1^{R\{X_k=u_k\}} \ldots \mathcal{T}_j^{R\{X_k=u_k\}}$. La variable del tercer campo será cualquiera de $Var(\mathcal{T}_1^{R\{X_k=u_k\}}) \cup \ldots \cup Var(\mathcal{T}_j^{R\{X_k=u_k\}})$. En caso de que todos los árboles $\{\{\mathcal{T}_1, \ldots \mathcal{T}_j\}, l, X_k\}$ sean nodos hojas (o sea, árboles con un sólo nodo etiquetados con un número) no se insertará en la lista ningún elemento y etiquetaremos el nodo hoja correspondiente a $X_k = u_k$ con la suma de todos los valores numéricos. El algoritmo $Marginaliza(\mathcal{T}, X_i)$ es el siguiente:

Algoritmo 4.3 Marginaliza($\mathcal{T}$,X_i)

1. Sea L la etiqueta del nodo $raiz(T)$
2. Si L es un número
   (a) Crear un nodo l de árbol inicialmente sin etiqueta
   (b) Pon en l como etiqueta el valor $L * |U_i|$
3. En caso contrario sea $X_k = L$
   (a) Si $X_k = X_i$ entonces sustituir en el árbol el nodo l por el árbol $Suma_{hijos}(T, X_i)$
   (b) En caso contrario,
      i. Crear un nodo l de árbol inicialmente sin etiqueta
      ii. Pon en l como etiqueta L

iii. Para cada $u_k \in U_k$

A. Poner el árbol $T_h = Marginaliza(T^{R(X_k=u_k)}, X_i)$ como hijo número k de l

4. Devolver T_r

■

Este algoritmo recursivo hace uso de la función Suma($\mathcal{T}$,X_k) que se encarga de sumar varios subárboles y que puede verse a continuación:

Algoritmo 4.4 Suma($\mathcal{T}$,X_k)

1. Crear un nodo l de árbol inicialmente sin etiqueta
2. Sean $L_1, \ldots, L_j$ las etiquetas de los nodos raíz de $T^{R(X_k=u_1)}, \ldots, T^{R(X_k=u_j)}$ $(j = |U_k|)$
3. Si $L_1, \ldots, L_j$ son todas números, poner el valor $L = L_1 + \ldots + L_j$ como etiqueta de l
4. En caso contrario

(a) Seleccionar una variable $X_i \in \{Var(T^{R(X_k=u_{k_1})} \cup \ldots \cup Var(T^{(X_k=u_{k_j})})\}$

(b) Añadir a $\mathcal{L}_m$ el nodo $\{\{T^{R(X_k=u_{k_1})}, \ldots, T^{R(X_k=u_{k_j})}\}, T_r, X_i\}$

(c) Mientras $\mathcal{L}_m$ no esté vacía

i. Borrar un nodo $\{\{T_1, \ldots, T_h\}, l_s, X_l\}$ de $\mathcal{L}_m$

ii. Poner X_l como etiqueta de l_s.

iii. Para cada $u_l \in U_l$

A. Crear un nodo l_t de árbol inicialmente sin etiqueta haciendo que sea hijo del nodo l_s

B. Sean $L_1, \ldots, L_h$ las etiquetas de los nodos raíz de $T_1^{R(X_l=u_{l_1})}, \ldots, T_h^{R(X_l=u_{l_j})}$

C. Si $L_1, \ldots, L_h$ son todos números, sea $L = L_1 + \ldots + L_h$. Pon L como etiqueta de l_t

D. En caso contrario,

- Seleccionar una variable $X_m \in \{Var(T_1^{R(X_l=u_{l_1})}) \cup \ldots \cup T_h^{R(X_l=u_{l_j})})\}$
- Añadir a $\mathcal{L}_c$ el nodo $\{T_1^{R(X_l=u_{l_1})}, \ldots, T_h^{R(X_k=u_{l_j})}, l_t, X_m\}$

5. Devolver T_r

■

Con estos algoritmos hemos mostrado como calcular de forma exacta la combinación y marginalización con árboles de probabilidad. Para cálculos exactos podríamos haber construido algoritmos recursivos de forma más simple y con una complejidad similar. Estos algoritmos alternativos no necesitarían del uso de las listas $\mathcal{L}_c$ y $\mathcal{L}_m$. La idea de estas listas es mantener una traza de los cálculos que restan por hacer para terminar de hacer la combinación o marginalización. Esta forma es más apropiada para el caso de cálculos aproximados, donde podemos dedicar mayor esfuerzo a los cálculos que resten que hacer que influyan más en el resultado final, o sea que aproximen más al resultado exacto.

4.3.2.3 Combinación y Marginalización aproximadas

Los algoritmos desarrollados en la sección anterior constituyen un método exacto de calcular la combinación y la marginalización de potenciales representados por medio de árboles. Los árboles que se obtienen aplicando un método exacto en la combinación y marginalización requieren un número de productos o de sumas menor (igual en el peor de los casos) que si hubiésemos hecho estas operaciones directamente con los potenciales asociados. Sin embargo en la práctica, cuando trabajamos con problemas medianamente complejos, el tamaño que pueden llegar a tener los árboles puede llegar a ser demasiado grande para poder almacenarlos en la memoria de un ordenador. Por eso, vamos a plantear un método que va a limitar el tamaño de los árboles resultado, haciendo que nunca sobrepasen un tamaño especificado por el usuario. Esto hará que la combinación y la marginalización no produzcan resultados exactos, pero se intentará que éstos sean lo más aproximados posible a la solución correcta.

La idea básica tanto en la combinación como en la marginalización consiste en construir un árbol que aproxime $h_1 \otimes h_2$ o $h^{\downarrow I-\{i\}}$, siguiendo los mismos procedimientos que en la construcción de un árbol. Como resultado se irán colocando en los niveles superiores del árbol construido aquellas variables que sean más informativas. El árbol irá creciendo poco a poco siguiendo por las ramas que más decrementen el error, o distancia al potencial exacto. La forma en que esto puede ser controlado en los algoritmos es haciendo que cuando se selecciona una variable para añadirla a la lista $\mathcal{L}_c$ o $\mathcal{L}_m$ como tercer campo del elemento que se va a

insertar, la variable escogida sea aquella que aporte mayor información de todas las posibles en forma similar a como se hacía para el caso de la construcción del árbol. Esta es una diferencia con los algoritmos exactos, donde no se seguía ningún criterio concreto para elegir la variable a colocar en el árbol resultado. También habrá que controlar la elección de los nodos de $\mathcal{L}_c$ ó $\mathcal{L}_m$ para ramificar. En la implementación de los algoritmos aproximados, lo que hemos hecho es escoger únicamente entre las etiquetas de los nodos raíz de los árboles a combinar o de los árboles a sumar. Esto se basa en que si estos árboles están bien construidos, esos nodos deben de ser los más informativos de cada uno de ellos. El considerar sólo estos nodos hace que los cálculos sean mucho más eficientes.

El principal problema está en calcular la información de una variable X_k en un nodo hoja aún no etiquetado, correspondiente a la configuración $X_J = u_j$. Para esto necesitaríamos conocer el potencial resultado, pero ésto es precisamente lo que estamos tratando de calcular. Una primera idea es hacer un cálculo exacto y después calcular una aproximación basándonos en este resultado exacto, para lo que se aplican directamente los métodos de la sección de construcción de árboles. Nosotros hemos optado por realizar una estimación de la información de una variable basándonos en los potenciales que combinamos o marginalizamos. Más concretamente, la información la medimos de la siguiente forma:

- **Combinación:** Sean h_1 y h_2 dos potenciales que queremos combinar, entonces la información $I'(X_J = u_J, X_k|h_1.h_2)$ que aporta la variable X_k si se coloca en el nodo hoja definido por la configuración $X_J = u_J$ de la actual estructura de árbol $\mathcal{T}^*$ para el potencial $h_1 \cdot h_2$, puede estimarse de la siguiente forma:

$$I'(X_J = u_J, X_k|h_1 \cdot h_2) = I(X_k|(h_1^{R(X_J=u_J)})^{\downarrow k}.(h_2^{R(X_J=u_J)})^{\downarrow k})$$

- **Marginalización:** Sea h el potencial que se quiere marginalizar sobre $I - \{i\}$, entonces la información $I'(X_J = u_J, X_k|h^{\downarrow I-\{i\}})$ que aportará la variable X_k si se coloca en el nodo hoja definido por la configuración $X_J = u_J$ de la actual estructura de árbol $\mathcal{T}^*$ para el potencial $h^{\downarrow I-\{i\}}$, puede estimarse de la siguiente forma:

$$I'(X_J = u_J, X_k|h^{\downarrow I-\{i\}}) = I(X_k|(h^{R(X_J=u_J)})^{\downarrow k})$$

Las medidas de información siempre hacen referencia a los potenciales respecto a los que están definidas. La notación $h^{R(X_J=u_J)}$ es análoga al caso de árboles: es el potencial que se

obtiene de h fijando las variables X_J al valor u_J. Los valores I' son fáciles de calcular ya que el cálculo de $h^{\downarrow k}$ es sencillo (sólo se necesita un recorrido sobre el potencial). Después nos queda un vector con tantos componentes como valores tenga X_k. Las estimaciones se basan en la siguiente igualdad

$$I(X_J = u_J, X_k|h) = I(X_k|(h^{R(X_J=u_J)})^{\downarrow k})$$

En el caso de la marginalización, esto permite obtener la expresión de la información en el potencial marginalizado en función de la información en el potencial original. En este caso, la estimación es exacta. Es decir $I'(X_J = u_J, X_k|h^{\downarrow I-\{i\}}) = I(X_J = u_J, X_k|h^{\downarrow I-\{i\}})$. Sin embargo, en el caso de la combinación lo que tenemos es una aproximación, ya que para obtener el valor exacto deberíamos de combinar antes de marginalizar, y nosotros marginalizamos antes para hacer más rápido el cálculo de la información.

Al igual que en la construcción del árbol tenemos distintas opciones para construir árboles aproximados para el caso de la combinación y la marginalización.

1. Limitar el número de nodos generados a un número prefijado, aproximando en los nodos hoja por la media de valores correspondientes a esa hoja.

2. Generar el árbol completo y podarlo posteriormente. La forma de hacer la poda es la misma que en el caso de la construcción del árbol.

La primera opción significa ir haciendo la combinación o marginalización con el mismo esquema que con las operaciones exactas mediante el uso de las listas $\mathcal{L}_c$ o $\mathcal{L}_m$. El árbol resultado de la operación va creciendo poco a poco. Este crecimiento se detendrá en el momento en que el árbol resultado construido hasta el momento alcance un tamaño especificado. Cuando acaba el crecimiento del árbol quedan en el árbol resultado operaciones sin realizar. O sea, tenemos nodos hoja aún sin etiquetar donde habría que colocar la combinación o bien suma de dos o más subárboles de los árboles originales. Puesto que el árbol resultado ya no puede crecer más se inserta en tales nodos hoja una estimación del resultado correcto. Esta estimación será el producto del valor medio de los dos subárboles que combinamos en el caso de la combinación, y la suma de los valores medios en el caso de la marginalización.

4.4 Evaluación experimental

Para evaluar los métodos de inferencia propuestos hemos usado un grafo generado aleatoriamente con 50 variables. A cada variable se le ha generado aleatoriamente un número de padres usando una distribución uniforme de media 2'0. El número de valores que puede tomar cada variable se ha generado usando una distribución de Poisson de media 2'0 (en caso que el número generado sea menor a 2 se toman dos valores para la variable). De forma aleatoria hemos considerado observadas 17 variables asignándoles uno de sus posibles valores. Se ha obtenido la probabilidad 'a posteriori' para una de las variables no instanciadas usando diferentes métodos y se ha comparado con la probabilidad que se obtiene por un método exacto, usando el error cuadrático medio que se obtiene con la siguiente fórmula:

$$\text{mse}(X_i) = \sqrt{\frac{\sum_{j=1}^{|U_i|}(P^*(X_i = u_j|E) - P(X_i = u_j|E))^2}{|U_i|}}$$

Para cada experimento realizado se ha medido el tiempo de CPU empleado por el ordenador (un 486 DX2 a 66 MHz). Se han obtenido los siguientes resultados limitando el tamaño del árbol resultado de combinaciones y marginalizaciones para diferentes número de nodos:

Nodos	10	100	600	1000	2500	5000	10000
mse	0.101	0.06	0.02	0.018	0.017	0.014	0.015
Tiempo (seg)	0	6	38	70	180	300	750

En el segundo experimento se hacen la combinación y la marginalización de forma exacta y posteriormente se poda el árbol resultado de combinar o marginalizar usando la expresión 4.3 para aprender un árbol de tamaño limitado a partir de un potencial.

Nodos	10	100	600	1000	2500	5000	10000
mse	0.114	0.084	0.035	0.05	0.026	0.014	0.0042
Tiempo (seg)	0	9	87	145	317	530	1010

En el tercer experimento se hacen también combinación y marginalización exacta podando posteriormente el árbol resultado de combinar o marginalizar. En este caso el árbol se va podando de forma gradual sustituyendo nodos interiores en que todos los hijos sean hojas por una hoja que es la media de los hijos. Estos nodos interiores serán sustituidos por nodos hoja si cumplen la condición 4.4. En este caso δ se obtiene como la entropía del potencial $\{0.5 - \epsilon, 0.5 + \epsilon\}$ para diferentes valores de ϵ

ϵ	0.03	0.025	0.02	0.015	0.01	0.005	0.0025
mse	0.075	0.062	0.055	0.043	0.02	0.0085	0.0058
Tiempo (seg)	13	19	28	45	88	273	737

En la figura 4.12 vemos gráficamente los errores cometidos en cada experimento en función del tiempo empleado en la propagación.

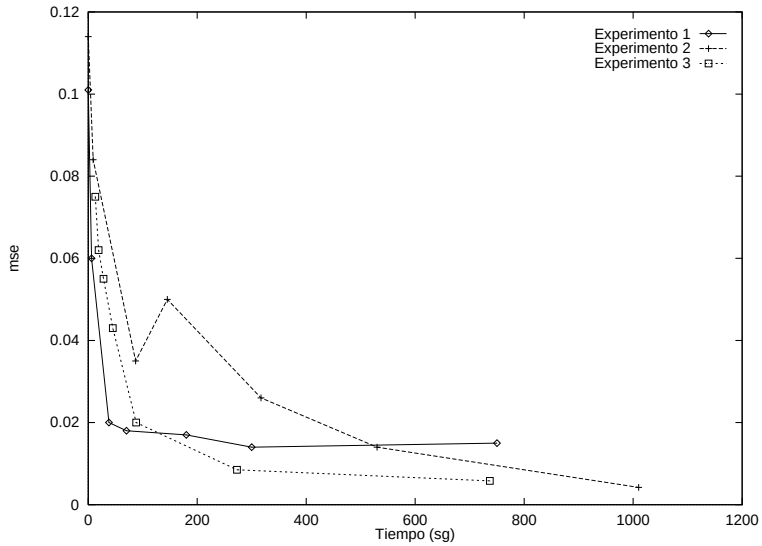


Figura 4.12: Errores cometidos en función del tiempo

4.5 Conclusiones

En este capítulo hemos mostrado cómo las algoritmos de propagación de probabilidades en grafos de dependencias se pueden realizar con los potenciales representados mediante árboles de probabilidad. Esto permite una representación mucho más compacta de las distribuciones de probabilidad que se traduce en una implementación más eficiente de los algoritmos. Esto es especialmente así cuando una variable tiene numerosos padres en el grafo dirigido acíclico original: suele ocurrir entonces que conociendo unos cuantos valores de los padres somos capaces de determinar cual es la distribución de probabilidad del nodo hijo, y esta información se puede representar directamente mediante un árbol de probabilidad.

También hemos descrito algoritmos aproximados que limitan el tamaño máximo de la representación de un potencial o que aproximan valores de potencial similares por su valor medio. En los experimentos que hemos hecho son estos últimos los que producen un mejor compor-

tamiento, ya que dan la mejor aproximación para un tiempo de cálculo dado. Sin embargo, los primeros permiten ajustar el algoritmo a los recursos disponibles: siempre obtendremos la mejor aproximación que nos permita el tiempo y espacio con el que contamos para obtener una respuesta.

En los experimentos realizados los mejores resultados se obtienen cuando los cálculos se hacen de forma exacta aproximando después los resultados, y dentro de este esquema cuando se sigue el criterio de aproximar siempre que la pérdida de información sea menor que un umbral dado.

En el futuro realizaremos una experimentación más extensiva de estos algoritmos, estudiando su comportamiento en una mayor variedad de situaciones. También consideraremos la organización de los cálculos en árboles de grupos como en los algoritmos clásicos de propagación de probabilidades [90, 106, 120].

Capítulo 5

Uso de Árboles de Probabilidad para Representar Intervalos de Probabilidad

5.1 Introducción

En los capítulos segundo y tercero se estudió el problema de la propagación probabilística cuando las distribuciones de probabilidad condicionada no eran conocidas con exactitud. Sólo se sabía de su pertenencia a un conjunto convexo de probabilidades condicionadas. Sin embargo, la forma más natural en la que suelen presentarse las probabilidades imprecisas en la práctica es a partir de intervalos de probabilidad. Supongamos que tenemos que especificar la distribución de probabilidad condicionada $P(Y|X)$. Lo habitual es que para cada valor de X, u, y para cada valor de Y, v tengamos un intervalo de probabilidad $P(Y = v|X = u) \in [\alpha(v|u), \beta(v|u)]$. En el capítulo 1, vimos cómo para cada valor $X = u$ estos intervalos se pueden transformar en un conjunto convexo de probabilidades que notaremos $H^{Y|X=u}$. El conjunto convexo de distribuciones de probabilidad condicionadas de Y dado X, $H^{Y|X}$, se puede calcular a partir de estos conjuntos convexos de la siguiente forma: Una distribución de probabilidad $p(.|.) \in H^{Y|X}$ si y sólo si $\forall u, p(.|u) \in H^{Y|X=u}$. Esta expresión es simple, pero hace que el número de puntos extremos de $H^{Y|X}$ sea demasiado grande: el producto del número de puntos extremos de todos los conjuntos $H^{Y|X=u}$, para los distintos posibles valores u de X. Esto hace que la aplicación directa de los algoritmos de los capítulos 2 y 3 sea inviable: los conjuntos convexos de los que

parten estos algoritmos tienen demasiados puntos extremos.

En este capítulo proponemos una transformación del problema original añadiendo variables transparentes que permitirá aplicar los algoritmos. La técnica consistirá en añadir una variable transparente para cada valor $X = u$ y después usar las independencias asimétricas existentes para, usando la representación de árboles desarrollada en el capítulo 4, obtener una especificación del problema a la que se puedan aplicar algoritmos exactos y aproximados.

El capítulo está estructurado de la siguiente forma. En la sección 2 proponemos la representación de las matrices de probabilidad condicionadas mediante variables transparentes y árboles de probabilidad. Esto permitirá aplicar los algoritmos de los capítulos 2 y 3. La sección 3 considera la adaptación del algoritmo exacto del capítulo 2, basado en la no eliminación de las variables transparentes. En concreto se considera el procedimiento de obtención de puntos extremos que se puede realizar directamente en el árbol sin necesidad de construir previamente una tabla, lo que sería demasiado costoso. También se estudia el algoritmo aproximado que se obtiene cuando se limita el tamaño de los árboles y la obtención de intervalos de probabilidad que incluyan necesariamente al intervalo exacto (en el caso del condicionamiento de Dempster). Estos se pueden usar en conjunción con los intervalos que se obtienen mediante los algoritmos de optimización combinatoria para obtener una cota del error. Finalmente la sección 4 está dedicada a la experimentación. Se usan redes Bayesianas conocidas (redes *Alarm*, *Boblo*, *Car Starts* y *Boerlage92*) en las que cada probabilidad inicial se modifica de forma aleatoria en un intervalo de probabilidad que la contiene. Se comprueba cómo, en estas redes, el cálculo con intervalos de probabilidad es factible de forma aproximada obteniéndose cotas de error bastante moderadas.

5.2 Representación de intervalos con árboles de probabilidad

Supongamos una distribución de probabilidad condicional $P(Y|X_I)$ de una variable Y dado un conjunto de variables X_I donde cada probabilidad se da a través de un sistema de intervalos.

Vemos en la sección 1.4 del capítulo 1 cómo obtener los puntos extremos asociados a esta distribución de probabilidad condicional dada como intervalos para obtener el conjunto convexo condicional $H^{Y|X_I}$. En el ejemplo 1.7 se concretaron los pasos a seguir para obtener dichos puntos extremos. Representando dicho conjunto de puntos extremos con una tabla, según se hace en dicho ejemplo, podemos ver que se obtienen muchos números repetidos en la tabla.

Esto nos hace pensar que es posible buscar una representación más compacta de dicha tabla mediante el uso de árboles de probabilidad.

Teniendo en cuenta la trasformación que venimos haciendo a lo largo de esta memoria, según la cual en cada distribución condicional sobre la que hay definido un conjunto convexo se añade una variable transparente T con tantos casos como puntos extremos tenga dicho conjunto convexo, podemos representar cada conjunto convexo condicional $H^{Y|X_I}$ mediante un árbol de probabilidad $\mathcal{T}$ en el que la variable transparente T aparezca en la raíz del árbol. Cada punto extremo del conjunto convexo condicional se obtiene como cada uno de los subárboles que cuelgan de T. O sea, podemos obtener cada uno de los puntos extremos del conjunto convexo condicional $H^{Y|X_I}$ con el árbol restringido $\mathcal{T}^{R(T=t)}$, donde t es uno de los valores que puede tomar T y que corresponde a uno de los puntos extremos del conjunto convexo condicional.

Cuando disponíamos de una distribución de probabilidad $P(Y|X_I)$ donde cada valor de probabilidad se daba con un intervalo entonces, el conjunto convexo condicional $H^{Y|X_I}$ se construía calculando en primer lugar, los puntos extremos de los conjuntos convexos $H^{Y|X_I=u_I}$ para toda posible configuración u_I de las variables X_I. Esto se hacía con el algoritmo 1.3.

Sea $U_I = \{u_1, \ldots, u_n\}$ el conjunto de configuraciones que puede tomar el conjunto de variables X_I, y sea $Ext_{u_I} = Ext_{H^{Y|X_I=u_I}}$ el conjunto de puntos extremos del conjunto convexo unidimensional $H^{Y|X_I=u_I}$ calculados con el algoritmo 1.3, entonces vimos en el capítulo 1 que el conjunto de puntos extremos del conjunto convexo condicional $H^{Y|X_I}$ (información global) se obtiene como el producto cartesiano de todos los conjuntos de puntos extremos Ext_{u_I}. O sea el conjunto de punto extremos lo obtenemos con la expresión:

$$Ext(H^{Y|X_I}) = Ext_{u_1} \times \ldots \times Ext_{u_n} \tag{5.1}$$

Dicho conjunto convexo condicional $H^{Y|X_I}$ podemos representarlo de forma equivalente, según se indicó en el capítulo 2, usando una variable transparente que tendrá tantos casos como puntos extremos tenga $H^{Y|X_I}$, y que se puede calcular como $\prod_{u_I} Ext_{u_I}$. Veamos un ejemplo en el que se ilustra como construir el árbol de probabilidad para un conjunto convexo obtenido al dar intervalos en una distribución de probabilidad condicional $P(Y|X)$:

Ejemplo 5.1 Representación mediante un árbol de probabilidad de un conjunto convexo
Veamos como representar con un árbol de probabilidad el conjunto convexo que se obtenía a partir de una distribución condicional expresada con intervalos en el ejemplo 1.7. En este

ejemplo disponíamos de una distribución de probabilidad condicional $P(Y|X)$ *sobre la que se establecen un conjunto de restricciones* $R_{\alpha,\beta}$ *que establecen un intervalo para cada probabilidad. Supongamos que* Y *toma valores en el conjunto* $V = \{v_1, v_2\}$ *y* X *en el conjunto* $U = \{u_1, u_2\}$*. Sea* $R_{\alpha,\beta}$ *el conjunto de restricciones dado por la siguiente tabla:*

	u_1		u_2	
	v_1	v_2	v_1	v_2
α	0.2	0.7	0.4	0.4
β	0.3	0.8	0.6	0.6

Vimos en el ejemplo 1.7 que aplicando el algoritmo 1.3 se obtenía para $X = u_1$ *el conjunto convexo* $H^{Y|X=u_1}$ *que tenía dos puntos extremos:*

	v_1	v_2
p_1	0.2	0.8
p_2	0.3	0.7

y el conjunto convexo $H^{Y|X=u_2}$ *con los siguientes puntos extremos:*

	v_1	v_2
q_1	*0.4*	*0.6*
q_2	*0.6*	*0.4*

La información global $H^{Y|X}$ *con la expresión 5.1, con la que se obtiene el conjunto convexo con los siguientes puntos extremos:*

	(u_1, v_1)	(u_1, v_2)	(u_2, v_1)	(u_2, v_2)
h_1	*0.2*	*0.8*	*0.4*	*0.6*
h_2	*0.2*	*0.8*	*0.6*	*0.4*
h_3	*0.3*	*0.7*	*0.4*	*0.6*
h_4	*0.3*	*0.7*	*0.6*	*0.4*

Podemos representar esta tabla utilizando un árbol de probabilidad donde ponemos una variable transparente T *en la raíz del árbol. Esta variable transparente tendrá* $2 \times 2 = 4$ *casos. Por tanto en el árbol tendrá 4 nodos hijo. Tal árbol lo mostramos en la figura 5.1.*

En el árbol de la figura 5.1 podemos obtener los cuatro puntos extremos del conjunto convexo condicional $H^{Y|X}$ *con los cuatro subárboles que cuelgan de la variable transparente* T*.* ■

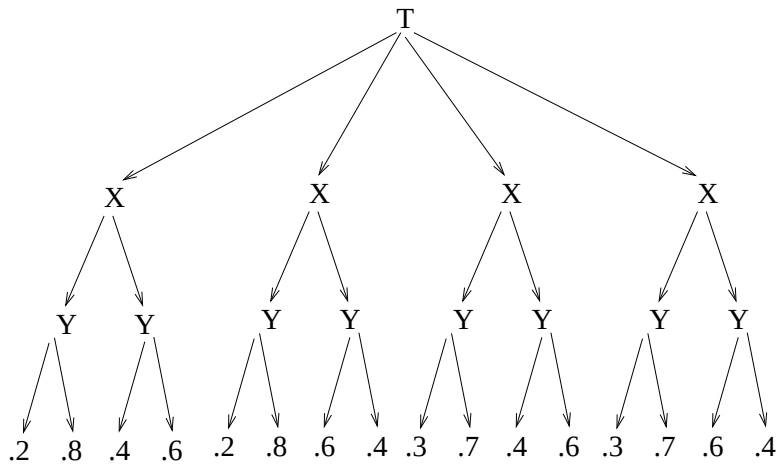


Figura 5.1: Árbol de probabilidad para el conjunto convexo condicional $H^{Y|X}$ del ejemplo 1.7

Una forma equivalente, pero más compacta, de representar la misma información es con el uso de una variable transparente T_{u_I} por cada configuración u_I que puedan tomar las variables X_I sobre las que condicionamos en el conjunto convexo condicional $H^{Y|X_I}$. Estas nuevas variables transparentes T_{u_I} tendrán tantos casos como puntos extremos tenga el conjunto convexo $H(Y|X_I = u_I)$. Así pues cada configuración $X_I = u_I$ tendrá asociada una variable transparente T_{u_I}. De igual forma que con las transparentes que veníamos utilizando hasta ahora, estas nuevas transparentes se incluirán como variables padres de Y en el grafo de dependencias. En la figura 5.2 a) podemos ver la transformación original que hacíamos añadiendo una única variable transparente por cada conjunto convexo, y en la parte b) la nueva transformación donde se añade una variable transparente por cada configuración de las variables padre, que en este caso es una única variable X con dos posibles valores $\{1, 2\}$, por lo que se añaden dos variables transparentes T_1 y T_2. Ahora, cada punto extremo p del conjunto convexo condicional $H^{Y|X_I}$ se obtiene fijando un valor para cada una de las variables transparentes T_{u_I}. O sea cada punto extremo p del conjunto convexo condicional $H^{Y|X_I}$ está determinado por una configuración de las variables transparentes $T_{u_1} = t_{u_1}, \ldots, T_{u_n} = t_{u_n}$.

Podemos de nuevo construir un árbol de probabilidad $\mathcal{T}$ para representar el conjunto convexo condicional $H^{Y|X_I}$ utilizando este nuevo esquema donde tenemos una variable transparente por cada configuración de las variables X_I. Ahora, para construir el árbol, podríamos poner las variables transparentes T_{u_I} en los niveles superiores del árbol. En la figura 5.3 podemos ver un árbol de probabilidad para el conjunto convexo $H^{Y|X}$ del ejemplo 1.7.

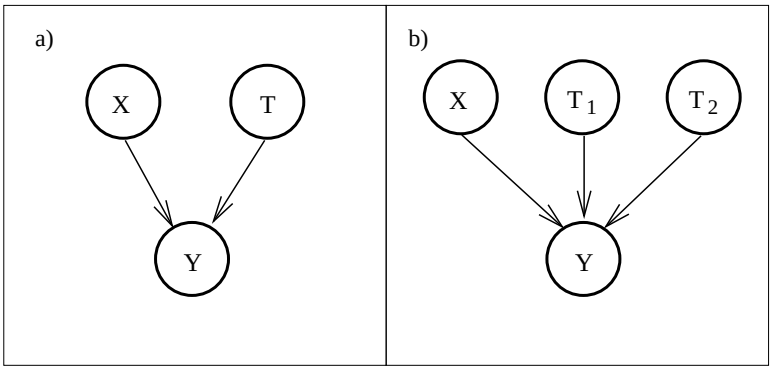


Figura 5.2: Antigua y nueva transformación del grafo de dependencias para representar el conjunto convexo condicional $H^{Y|X}$: En la parte a) se muestra la transformación de adición de una única transparente por cada conjunto convexo condicional. En la parte b) se añade una variable transparente por cada valor que puede tomar la variable padre X

Con este nuevo esquema, con cada asignación de valores a las variables transparentes $T_{u_1} = t_{u_1}, \ldots, T_{u_n} = t_{u_n}$ obtendremos un punto extremo $p \in H^{Y|X_I}$ que corresponde al árbol restringido $\mathcal{T}^{R(T_{u_1}=t_{u_1},\ldots,T_{u_n}=t_{u_n})}$.

Sin embargo con esta transformación del problema no hemos logrado ninguna ventaja respecto a cuando sólo utilizábamos una variable transparente. Se obtiene una representación más resumida de $\mathcal{T}$ si nos fijamos en que existen una serie de independencias asimétricas que podrían ser aprovechadas en un nuevo árbol $\mathcal{T}'$ más pequeño que $\mathcal{T}$. Una vez conocida una configuración $X_I = u_I$ se tiene que Y será independiente de todas las variables transparentes excepto de la variable transparente asociada a esta configuración, que habíamos denominado T_{u_I} más arriba. Estas independencias pueden aprovecharse para construir un árbol más compacto que el anterior. Para ello podemos colocar las variables de X_I en los primeros niveles del árbol. Luego colocaríamos en cada rama la variable transparente T_{u_I} que corresponda a la configuración de las variables de X_I de esa rama. Y por último pondríamos la variable Y. Veamos con un ejemplo de como podemos construir este nuevo árbol más compacto:

Ejemplo 5.2 Construcción de un árbol más resumido para un conjunto convexo condicional *En el árbol de la figura 5.3, podemos comprobar fácilmente que Y es independiente de T_2 cuando $X = u_1$. Para verlo gráficamente podemos fijarnos en el árbol de la parte a) de la figura 5.4. Este árbol representa $\mathcal{T}^{R(X=u_1)}$. En este árbol restringido se ve que el valor de*

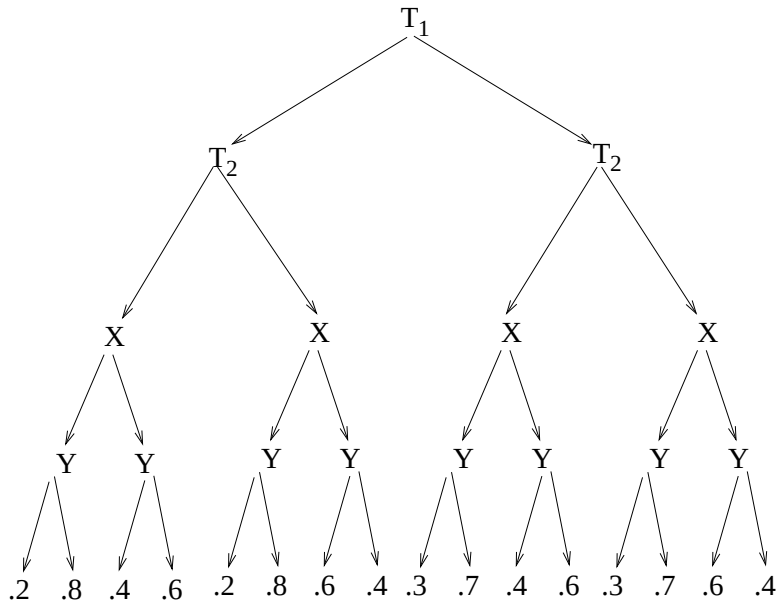


Figura 5.3: Árbol de probabilidad para el conjunto convexo condicional $H^{Y|X}$ con la nueva transformación

T_2 no influye en el valor que tome la variable Y. O sea, se da la independencia asimétrica $I_c(Y;T_2|X=u_1)$. También se puede ver que Y es independiente de T_1 cuando $X=u_2$. De nuevo, en la parte b) de la figura 5.4 podemos ver gráficamente $\mathcal{T}^{R(X=u_2)}$ que nos muestra que el valor de T_1 no influye en el valor de la variable Y. O sea, se da la independencia asimétrica $I_c(Y;T_1|X=u_2)$. De esta forma podríamos construir un árbol más pequeño que aprovechase estas independencias asimétricas. Dicho árbol es el mostrado en la figura 5.5. Este nuevo árbol utiliza solamente 8 nodos hojas (8 valores numéricos), en lugar de los 16 que utilizaba los árboles de las figuras 5.1 y 5.3.

■

5.3 Adaptación de los Algoritmos

5.3.1 Obtención de los puntos extremos eliminando variables transparentes

El resultado de aplicar los algoritmos de eliminación de variables o bien el de propagación en un árbol de grupos es un árbol de probabilidad $\mathcal{T}_j$ cuyas variables serán la variable objetivo X_j y un conjunto de variables transparentes T_J. Este árbol de probabilidad resultado se obtiene

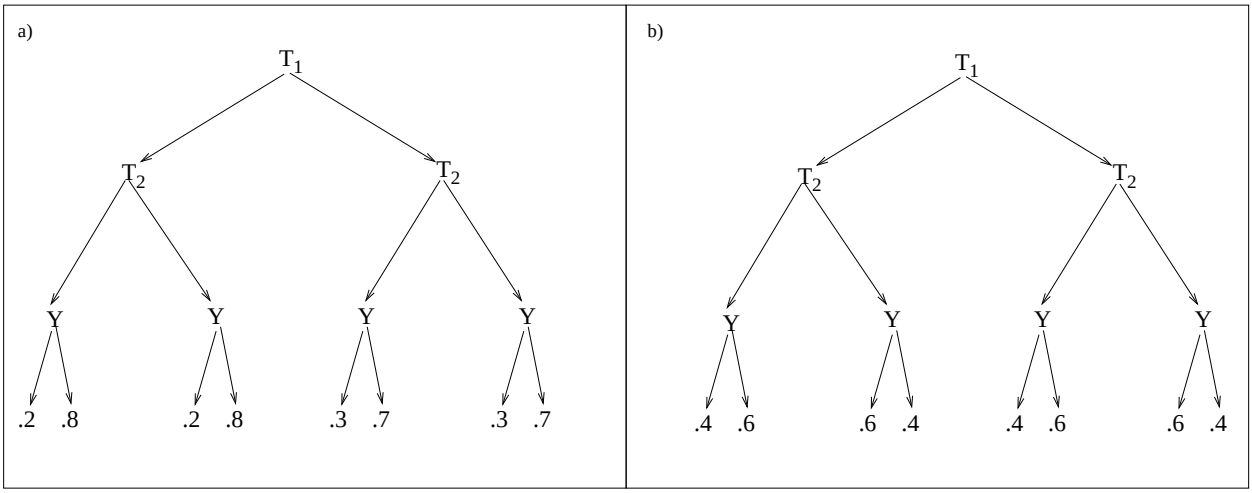


Figura 5.4: Representación de los árboles restringidos $\mathcal{T}^{R(X=u_1)}$ y $\mathcal{T}^{R(X=u_2)}$

tras varias operaciones de combinación y marginalización de los árboles de probabilidad que representan las valuaciones iniciales en el grafo de dependencias. Este árbol resultado nos da el conjunto convexo 'a posteriori' PS_j para la variable de interés X_j. Si X_j puede tomar dos posibles valores entonces este conjunto convexo 'a posteriori' PS_j será un conjunto de puntos extremos $Ext(PS_j)$ pertenecientes a $\mathbb{R}^2$.

A partir del árbol resultado se pueden obtener los puntos extremos para la variable objetivo. La forma más obvia de hacerlo sería transformar el árbol de probabilidad resultado en una tabla bidimensional. En el eje horizontal pondríamos los distintos valores que puede tomar la variable objetivo X_j y en el eje vertical las posibles combinaciones de casos de las variables transparentes del árbol resultado. De esta forma cada una de las filas de esta tabla nos daría un posible punto extremo. Seguramente esta tabla tendría muchísimos puntos repetidos. Además muchos de tales puntos es posible que no fuesen realmente puntos extremos. Así pues, las tablas obtenidas de tal modo podrían ser enormes, y de hecho en las redes que nosotros hemos utilizado el tamaño de dichas tablas según este proceso es tan grande que es imposible almacenarlas en un ordenador.

Podemos aplicar un procedimiento similar que no necesita la creación de una tabla tan enorme, pero que como veremos, sigue necesitando un tiempo exponencial para obtener los puntos debido a que obtiene los mismos puntos que con el anterior procedimiento. Si T_J es el conjunto de variables transparentes del árbol resultado de la propagación $\mathcal{T}_j$, entonces para cada configuración t_J del conjunto de variables transparentes T_J tendremos un punto del

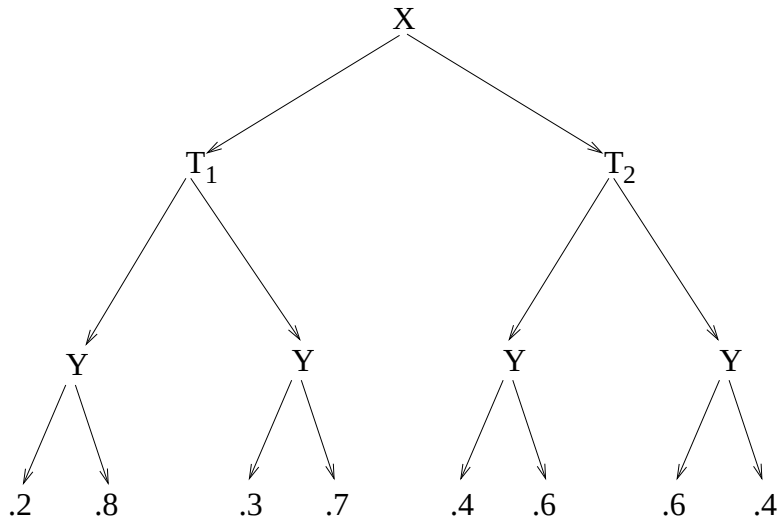


Figura 5.5: Otro posible árbol de probabilidad para el conjunto convexo condicional $H^{Y|X}$ que aprovecha las independencias asimétricas entre Y y las variables transparentes

conjunto convexo 'a posteriori' PS_j que se obtiene a partir del árbol con la expresión $\mathcal{T}_j^{R(T_J=t_J)}$. O sea, seleccionando un valor para cada variable transparente del árbol obtendremos un árbol restringido compuesto por la variable objetivo X_j en la raíz y con tantos nodos hijos (que serán nodos hoja) como casos tenga X_j. Este árbol restringido nos da entonces un posible punto del conjunto convexo 'a posteriori' PS_j. De esta forma no necesitamos tener en la memoria del ordenador nada más que el árbol resultado de la propagación, el cual con sucesivos pasos en los que vamos reduciendo una variable en dicho árbol, se va haciendo más pequeño. Pero como este proceso hay que repetirlo para cada configuración de las variables del árbol resultado, y el número de configuraciones es el producto del número de casos de las variables transparentes, entonces hará falta realizar un número enorme de reducciones en el árbol resultado. Este número crece exponencialmente cuando se aumenta el número de variables transparentes en el árbol resultado. Así pues, podemos concluir que ninguno de los métodos descritos, aunque son correctos, son factibles de utilizar en la práctica para un problema no muy complejo.

Veamos un ejemplo que muestra los puntos que obtendríamos para un árbol resultado y que ilustra los inconvenientes de estos métodos.

Ejemplo 5.3 Obtención de los puntos del conjunto convexo 'a posteriori' a partir del árbol de probabilidad resultado

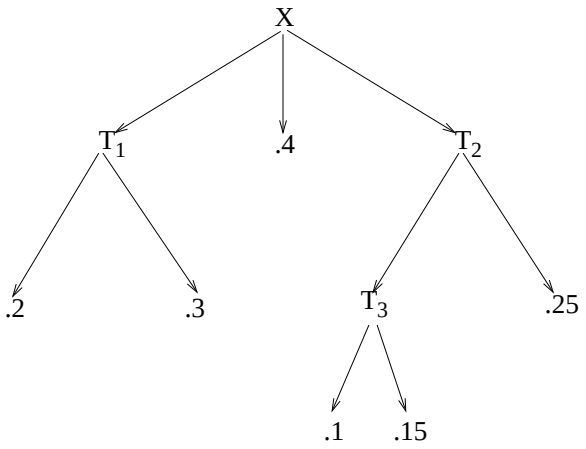


Figura 5.6: Árbol de probabilidad resultado para la variable objetivo X

Supongamos el árbol de probabilidad resultado de la figura 5.6 para la variable objetivo X que suponemos que puede tomar tres posibles valores $\{u_1, u_2, u_3\}$. En este árbol podemos ver que tenemos tres variables transparentes T_1, T_2 y T_3 y la variable objetivo X. Aplicando cualquiera de los dos métodos descritos más arriba obtendríamos los 8 puntos de la tabla que aparece más abajo. Dicha tabla contiene el punto obtenido a partir del árbol de probabilidad para cada combinación de los valores de los puntos extremos. En la parte izquierda de dicha tabla mostramos el valor de cada variable transparente y en la parte derecha el punto obtenido con dicha configuración. Podemos observar que los puntos de la tercera y cuarta fila son el mismo: $(0.2, 0, 4, 0.25)$. También podemos ver que los puntos de las filas séptima y octava son el mismo: $(0.3, 0.4, 0.25)$. La razón por la que hemos obtenido puntos repetidos en la tabla es que en la tabla no se tienen en cuenta las independencias asimétricas que hay entre las variables del árbol resultado, lo cual provoca repetición de puntos en la tabla. Por ejemplo, el árbol de probabilidad resultado nos dice que la variable X es independiente de T_3 cuando conocemos que $T_2 = 2$. La tabla al no utilizar esta independencia asimétrica provoca el tener varias entradas repetidas.

T_1	T_2	T_3	u_1	u_2	u_3
1	*1*	*1*	*0.2*	*0.4*	*0.1*
1	*1*	*2*	*0.2*	*0.4*	*0.15*
1	*2*	*1*	*0.2*	*0.4*	*0.25*
1	*2*	*2*	*0.2*	*0.4*	*0.25*
2	*1*	*1*	*0.3*	*0.4*	*0.1*
2	*1*	*2*	*0.3*	*0.4*	*0.15*
2	*2*	*1*	*0.3*	*0.4*	*0.25*
2	*2*	*2*	*0.3*	*0.4*	*0.25*

La tabla anterior también nos da dos puntos que seguro que no van a ser puntos extremos. Estos puntos aparecen en las filas segunda $(0.2, 0.4, 0.15)$ *y sexta de la tabla* $(0.3, 0.4, 0.15)$. *El punto* $(0.2, 0.4, 0.15)$ *no es extremo debido a que puede obtenerse como combinación lineal de los puntos* $(0.2, 0.4, 0.1)$ *y* $(0.2, 0.4, 0.25)$. *El punto* $(0.3, 0.4, 0.15)$ *tampoco es extremo porque puede obtenerse como combinación lineal de los puntos* $(0.3, 0.4, 0.1)$ *y* $(0.3, 0.4, 0.25)$.

De esta forma de los 8 puntos que tiene la tabla anterior, podemos descartar 4 puntos. ■

Vamos a describir entonces un algoritmo que no obtiene puntos repetidos y que además elimina puntos que son combinación lineal de otros (no todos, pero una gran proporción de ellos). El algoritmo podemos descomponerlo estudiando tres posibles situaciones de menos general a más general que pueden ocurrir en el árbol de probabilidad resultado. Empecemos estudiando la situación menos general y por tanto más simple de tratar.

Primera situación

Sea un árbol de probabilidad $\mathcal{T}_j$ en el que la variable objetivo X_j etiqueta el nodo raíz y que tiene tantos subárboles como casos tenga dicha variable. Además en esta primera situación suponemos que entre los distintos subárboles del nodo raíz no existen variables transparentes en común: $\forall u_j, u'_j \in U_j \; Var(\mathcal{T}^{R(X_j=u_j)}) \cap Var(\mathcal{T}^{R(X_j=u'_j)}) = \emptyset$. Un ejemplo de árbol donde se dan estas condiciones es en el de la figura 5.6. En esta situación, podemos obtener un conjunto de puntos que definen el convexo de la variable objetivo transformando el árbol resultado $\mathcal{T}_j$ en un nuevo árbol $\mathcal{T}'_j$ con la variable objetivo en la raíz, y con tantos hijos como casos tenga dicha variable. Para cada valor u_j de la variable X_j buscaremos en el árbol $\mathcal{T}_j^{R(X_j=u_j)}$ el valor mínimo y máximo. Para ello sólo hay que recorrer todos los nodos hoja de dicho árbol

guardando el mínimo y el máximo. Obtenemos así un mínimo y un máximo por cada valor u_j de X_j. En el nuevo árbol $\mathcal{T}'_j$, si el mínimo y el máximo para el caso u_j son distintos pondremos como hijo de X_j correspondiente al caso u_j una variable transparente con dos hijos. En el primer hijo del nodo para esta variable transparente guardaremos el mínimo y en el segundo el máximo. Si el mínimo y el máximo eran iguales entonces simplemente ponemos como hijo correspondiente al caso u_j el valor del mínimo.

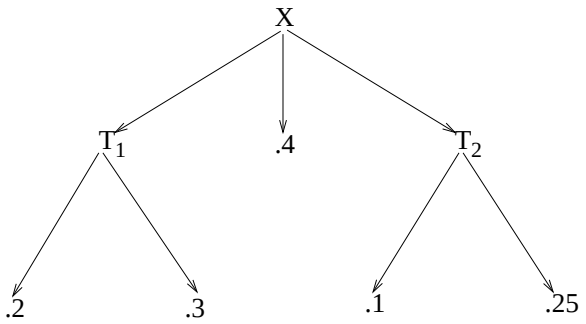


Figura 5.7: Nuevo árbol de probabilidad resultado $\mathcal{T}'_j$ para la variable objetivo X

Este nuevo árbol $\mathcal{T}'_j$ nos permite obtener los puntos para la variable objetivo X_j. Para ello, ahora sí podemos aplicar cualquiera de los dos métodos indicados al principio de esta sección. Por ejemplo, si T'_J son las variables transparentes de $\mathcal{T}'_J$ entonces para cada configuración t'_J de las variables transparentes T'_J obtenemos un punto mediante $\mathcal{T}'^{R(T'_J=t'_J)}_j$. El árbol $\mathcal{T}'_j$ que se obtiene transformando el árbol $\mathcal{T}_j$ de la figura 5.6 lo podemos ver en la figura 5.7. Con este árbol y según el procedimiento explicado más arriba obtendríamos los 4 puntos extremos de la siguiente tabla en lugar de los 8 puntos que se obtenían con el procedimiento aplicado en el ejemplo 5.3:

T_1	T_2	u_1	u_2	u_3
1	1	0.2	0.4	0.1
1	2	0.2	0.4	0.25
2	1	0.3	0.4	0.1
2	2	0.3	0.4	0.25

La simplificación proporcionada por este algoritmo se debe a que es fácil comprobar que cualquier punto intermedio entre el máximo y el mínimo en uno de los subárboles del nodo raíz da lugar a un punto que no es extremo.

El algoritmo que resuelve esta primera situación lo podemos ver a continuación:

Algoritmo 5.1 Algoritmo1 de obtención de puntos con un árbol de probabilidad
Entrada: Un árbol $\mathcal{T}_j$ que tiene a la variable objetivo X_j en el nodo raíz y en donde los subárboles del nodo raíz no tienen variables comunes.
Salida: Conjunto de puntos pertenecientes al conjunto convexo 'a posteriori' PS_j de la variable X_j.

1. Crear un nuevo árbol $\mathcal{T}'$ con un nodo etiquetado con X y con tantos hijos como casos tenga X.

2. Para cada posible valor u_j de X_j

   - Obtener el árbol $\mathcal{T}_j^{R(X_j=u_j)}$.
   - Obtener los valores mínimo y máximo de los nodos hoja del anterior árbol $\mathcal{T}_j^{R(X_j=u_j)}$.
   - Si los valores mínimo y máximo son distintos:
     - Crear un nuevo nodo para el árbol $\mathcal{T}'$ etiquetado con una nueva variable transparente T_{u_j}.
     - Crear dos hijos para T_{u_j}. En el primero insertamos el valor mínimo y en el segundo instalamos el valor máximo.
     - Poner al nodo etiquetado con T_{u_j} como el hijo correspondiente a u_j de la variable X_j.
   - En caso contrario crear un nodo etiquetado con el valor mínimo y ponerlo como el hijo correspondiente a u_j de la variable X_j en el árbol $\mathcal{T}'$.

3. Para cada configuración t'_J de las variables transparentes T'_J de $\mathcal{T}'$:

   - Calcula $\mathcal{T}_j'^{R(T'_J=t'_J)}$.
   - El anterior árbol restringido es un árbol con X_j en la raíz y con tantos hijos como casos tenga X_j, cada uno etiquetado con un número real que nos da un punto del conjunto convexo 'a posteriori'.

■

Segunda situación

Una segunda situación es cuando tenemos la variable objetivo X_j en el nodo raíz del árbol resultado pero la intersección entre los distintos subárboles que cuelgan de X_j no es vacía. Un ejemplo de árbol donde tenemos esta segunda situación lo podemos ver en la figura 5.8, donde vemos que la variable transparente T_2 aparece en el primer y tercer subárbol de la variable objetivo X. En tal caso podemos transformar el problema en una serie de subproblemas pertenecientes al tipo explicado en la primera situación. Sea $Pun(X_j|\mathcal{T})$ un conjunto de puntos que genera el convexo PS_j y que podemos obtener a partir del árbol resultado $\mathcal{T}$. Entonces, para dividir el problema en subproblemas podemos decir que $Pun(X_j|\mathcal{T})$ puede obtenerse como:

$$Pun(X_j|\mathcal{T}) = \bigcup_{t_c} Pun(X_j|\mathcal{T}^{R(T_c=t_c)}) \tag{5.2}$$

donde T_c es una de las variables transparentes del árbol que está presente en dos o más subárboles de la variable objetivo X_j y t_c son los posibles valores que puede tomar dicha variable transparente.

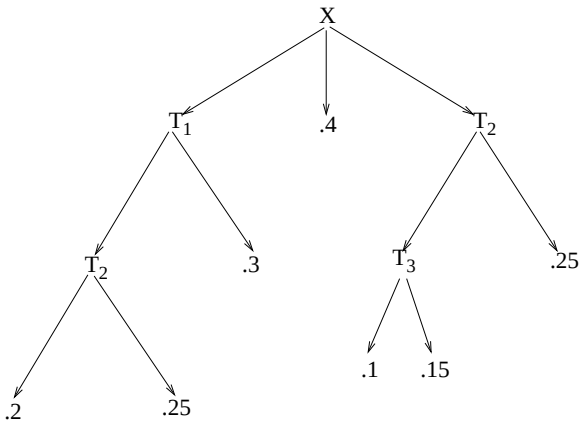


Figura 5.8: Un árbol de probabilidad resultado para la variable objetivo X correspondiente a la segunda situación

De esta forma hemos descompuesto el problema original de obtener una serie de puntos de PS_j en tantos subproblemas como casos tenga la variable transparente T_c. Si en los nuevos árboles obtenidos según la expresión 5.2 seguimos teniendo variables en común entre los distintos subárboles de la variable objetivo X_j aplicaríamos de nuevo la expresión 5.2 para ob-

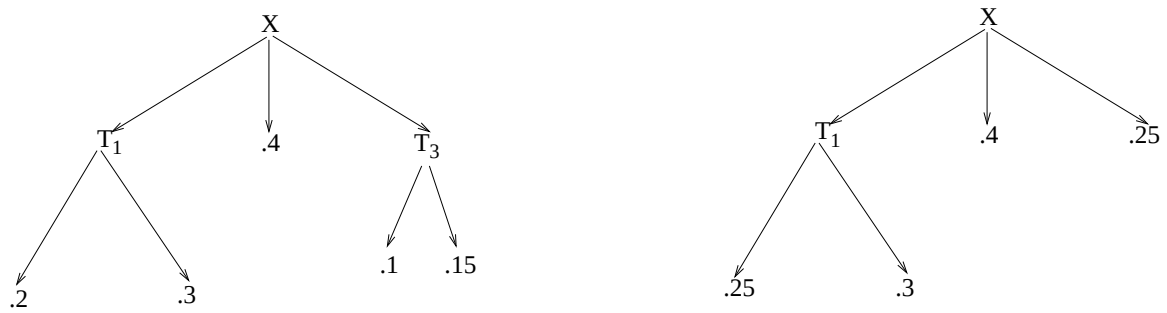


Figura 5.9: Descomposición en dos árboles del árbol de la figura 5.8

tener nuevos subproblemas. Cuando los árboles obtenidos ya no tienen variables comunes en los subárboles de X_j entonces obtenemos los puntos según la primera situación. Por ejemplo para el árbol de la figura 5.8 descomponemos el problema de obtener un conjunto generador de puntos (o sea de calcular $Pun(X|\mathcal{T})$) en el problema de obtener los puntos de cada uno de los subárboles de la figura 5.9 (calcular $Pun(X|\mathcal{T}^{R(T_2=1)})$ y $Pun(X|\mathcal{T}^{R(T_2=2)})$. En este sencillo ejemplo los árboles de la figura 5.9 ya cumplen que no tienen variables en común en los subárboles de la variable objetivo con lo que pueden resolverse según el algoritmo para la primera situación. Con estos árboles obtendríamos entonces los puntos de las siguientes dos tablas. La tabla de la izquierda se corresponde con el árbol de la izquierda de la figura 5.9 y la derecha con el árbol de la derecha de la figura 5.9:

T_1	T_3	u_1	u_2	u_3
1	1	0.2	0.4	0.1
1	2	0.2	0.4	0.15
2	1	0.3	0.4	0.1
2	2	0.3	0.4	0.15

T_1	u_1	u_2	u_3
1	0.25	0.4	0.25
2	0.3	0.4	0.25

De esta forma, a partir del árbol resultado de la figura 5.8, se obtienen los 6 puntos que muestran las dos anteriores tablas.

El algoritmo que resuelve esta segunda situación lo podemos ver a continuación:

Algoritmo 5.2 Algoritmo2 de obtención de puntos con un árbol de probabilidad
Entrada:

1. Un árbol $\mathcal{T}_j$ que tiene a la variable objetivo X_j en el nodo raíz y en donde los subárboles del nodo raíz pueden tener variables transparentes comunes.

2. El conjunto de variables transparentes comunes T_C

Salida: Conjunto de puntos pertenecientes al conjunto convexo 'a posteriori' PS_j de la variable X_j.

1. Si el conjunto de variables comunes T_C no está vacío:
   - Para cada caso t_{c_i} de la primera variable transparente común T_{c_1} del conjunto T_C:
     - Obtener $\mathcal{T}^{R(T_{c_1}=t_{c_i})}$
     - Llamar recursivamente a este algoritmo (Algoritmo2) utilizando el árbol $\mathcal{T}^{R(T_{c_1}=t_{c_i})}$ calculado en la etapa anterior y el conjunto de variables transparentes $T_C - T_{c_1}$.
2. En caso contrario llamar al algoritmo Algoritmo1 con el árbol $\mathcal{T}_j$.

■

Tercera situación

Por último veamos cual sería la situación en la que la variable objetivo X_j no aparece en la raíz del árbol. X_j podrá aparecer en nodos del árbol distintos al raíz, e incluso podrá aparecer en más de un nodo del árbol. Básicamente el algoritmo que resuelve la situación tercera hace un recorrido en profundidad del árbol. En cada paso, si la variable transparente T aparece como etiqueta del nodo raíz, entonces el conjunto de puntos que define PS_j es la unión de los puntos que definen cada uno de los subárboles que corresponden a cada uno de los hijos de T. El recorrido en profundidad se detiene bajo dos condiciones. La más simple es cuando se llega a un nodo hoja etiquetado con un número real r sin haber pasado por X_j. Este caso significa que hemos obtenido un punto del conjunto convexo 'a posteriori' de X_j formado por la repetición del número r tantas veces como casos tenga la variable X_j. La segunda condición de parada del recorrido en profundidad es cuando encontramos la variable objetivo X_j en un nodo interior del árbol. En este caso se detiene el recorrido en profundidad por esta rama y resolvemos el problema de acuerdo con la primera o segunda situación.

Por ejemplo, en el árbol de la figura 5.10 podemos ver un ejemplo de esta situación general. La variable objetivo X ahora no aparece en el nodo raíz, y además aparece en dos nodos del árbol. Para obtener los puntos extremos hacemos un recorrido en profundidad del árbol partiendo del nodo raíz T_4. Visitamos primero la rama izquierda de T_4 llegando al nodo

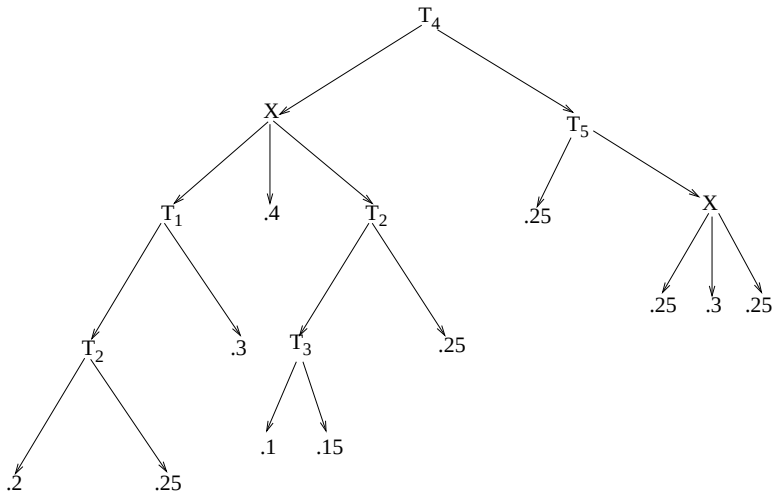


Figura 5.10: Árbol resultado de una propagación donde se da la tercera situación (situación general)

etiquetado con X. En ese momento detenemos el recorrido en profundidad por tal rama y resolvemos este subproblema como en la situación segunda. De hecho, este subproblema es el problema que resolvimos anteriormente para la el árbol de la figura 5.8 que obtenía 6 puntos extremos. Continuamos con el recorrido en profundidad por la rama derecha de T_4 llegando al nodo T_5. Desde T_5 seguimos por la rama izquierda llegando a un nodo hoja etiquetado con un número real $r = 0.25$. Esto nos da el punto $(0.25, 0.25, 0.25)$. Seguimos el recorrido en profundidad por la rama derecha de T_5 llegando al nodo etiquetado con X. De nuevo detenemos el recorrido en profundidad por esta rama y resolvemos este subproblema con el algoritmo de la situación segunda, aunque en este caso el problema pertenece directamente a la situación primera. Este algoritmo nos dará el punto $(0.25, 0.3, 0.25)$. De esta forma hemos obtenido 8 puntos del conjunto convexo 'a posteriori' de la variable X. Dichos puntos los mostramos en la siguiente tabla.

u_1	u_2	u_3
0.2	0.4	0.1
0.2	0.4	0.15
0.3	0.4	0.1
0.3	0.4	0.15
0.25	0.4	0.25
0.3	0.4	0.25
0.25	0.25	0.25
0.25	0.3	0.25

El algoritmo que resuelve la situación tercera (situación general) lo podemos ver a continuación:

Algoritmo 5.3 Algoritmo3 general de obtención de puntos con un árbol de probabilidad

Entrada: Un árbol $\mathcal{T}_j$ que puede tener la variable objetivo X_j en cualquier nodo del árbol, la cual puede aparecer varias veces en el árbol e incluso ninguna vez.

Salida: Conjunto de puntos pertenecientes al conjunto convexo 'a posteriori' PS_j de la variable X_j.

1. Si el nodo raíz de $\mathcal{T}_j$ es un nodo probabilidad devolver un único punto extremo formado por la repetición de este valor.

2. En caso contrario, si el nodo raíz de $\mathcal{T}_j$ es un nodo transparente T

   - Para cada valor t del nodo transparente T
     - Llamar recursivamente a este algoritmo (Algoritmo3) con el árbol $\mathcal{T}_j^{R(T=t)}$

3. En caso contrario, si el nodo raíz de $\mathcal{T}_j$ es la variable objetivo X_j

   - Calcular el conjunto de variables transparentes T_C entre los subárboles del nodo raíz de $\mathcal{T}_j$.
   - Llamar al algoritmo que resuelve la segunda situación (Algoritmo2) con el árbol $\mathcal{T}_j$ y el conjunto de variables transparentes T_C.

■

5.3.2 Un criterio de eliminación de puntos no extremos

En la sección 5.3.1 hemos mostrado que la tabla asociada al árbol de probabilidad resultado $\mathcal{T}$ de una propagación con los algoritmos de eliminación de variables, o bien con el de propagación en un árbol de grupos, puede contener puntos no extremos y que por tanto no influyen en el resultado final del conjunto convexo 'a posteriori' PS_j de la variable X_j. Por tanto, estos puntos no extremos pueden eliminarse de los cálculos sin que el resultado final se altere. Esta situación se presenta cuando se tiene un árbol resultado con la variable objetivo X_j en la raíz de $\mathcal{T}$ y donde se cumple que $\forall u_j, u'_j \in U_j$ se verifica que $Var(\mathcal{T}^{R(X=u_j)}) \cap Var(\mathcal{T}^{R(X=u'_j)}) = \emptyset$. El procedimiento para evitar incluir estos puntos no extremos que se ha mostrado en la sección 5.3.1 es transformando el árbol $\mathcal{T}$ en un nuevo árbol $\mathcal{T}'$ más simple con el que ya no se obtendrán puntos que son combinación lineal de otros. En cada subárbol $\mathcal{T}^{R(X=u_j)}$ del nodo raíz de $\mathcal{T}$ se sustituyen las variables transparentes por una sola con dos nodos hoja, uno para incluir el mínimo de los nodos hoja del subárbol $\mathcal{T}^{R(X=u_j)}$ y otro para el máximo.

La transformación mencionada más arriba de $\mathcal{T}$ en $\mathcal{T}'$ puede llevarse a cabo también en pasos intermedios de la propagación, lo cual además de evitar obtener gran número de puntos no extremos hará que la propagación sea más eficiente. Según vimos en el capítulo 2, la propagación de conjuntos convexos de probabilidad era equivalente a realizar un número de propagaciones probabilísticas determinado por el producto del número de puntos extremos de las informaciones iniciales. Pues bien, si eliminamos variables transparentes sustituyéndolas por una sola con dos valores estamos reduciendo enormemente el número de propagaciones probabilísticas necesarias para llevar a cabo la propagación con los conjuntos convexos.

En los algoritmos de propagación por eliminación de variables y en el de propagación en un árbol de grupos descritos en el capítulo 3, se tiene en un paso intermedio de la propagación un conjunto de árboles de probabilidad que podemos denotar con $Set(\mathcal{T})$, cada uno representando una valuación. En este caso la transformación de un árbol $\mathcal{T} \in Set(\mathcal{T})$ en un nuevo árbol $\mathcal{T}'$ podrá hacerse cuando en el árbol $\mathcal{T}$ exista un subárbol donde sólo hay variables transparentes las cuales además no aparecen en ningún otro subárbol de $\mathcal{T}$ ni en ningún otro árbol del conjunto $Set(\mathcal{T})$. Entonces en estos algoritmos de propagación podemos comprobar tras cada operación de combinación o marginalización si se da esta condición, y en caso de que sea así haremos la transformación del árbol resultado de dichas operaciones.

El proceso descrito en esta sección se puede considerar como un algoritmo aproximado que detecta puntos no extremos eliminándolos del resultado final, aunque puede seguir incluyendo

algunos puntos no extremos.

5.3.3 Obtención de intervalos aproximados

En el capítulo 4 se presentó en la sección 4.3.2.3 cómo podían llevarse a cabo las operaciones de combinación y marginalización con árboles en forma aproximada consiguiendo árboles de un tamaño menor que si se hiciese de forma exacta. Se estudiaron varias posibilidades para hacer esta aproximación. En todas ellas era necesario sustituir en ciertas zonas del árbol resultado, un subárbol por un nodo etiquetado con la media de los valores que representaba este subárbol. La justificación de sustituir por el valor medio en lugar de otro valor distinto nos lo daba la proposición 4.1. Podemos utilizar un mecanismo que nos permita acotar los errores que estamos cometiendo en el resultado final de la propagación al hacer estas aproximaciones. Con el mecanismo que proponemos se utilizará en cada nodo hoja de los árboles de probabilidad, que hasta ahora habían sido etiquetados con números reales $r \in I\!R$, otros dos números reales. El primero de ellos servirá para almacenar un valor mínimo $r_{\min} \in I\!R$ y el segundo para guardar un valor máximo $r_{\max} \in I\!R$. En cada nodo hoja, el valor r estará comprendido entre el valor $r_{\min}$ y $r_{\max}$. Estos valores mínimo y máximo coincidirán y serán iguales al valor r de cada nodo hoja en caso de que se haga una propagación exacta, o sea, en caso de que nunca se aproxime en las operaciones de combinación y marginalización. Cuando sea necesario aproximar en una rama del árbol resultado, el valor r se calcula como siempre con la media del subárbol al que sustituye el nodo hoja que insertamos en el árbol resultado. El valor $r_{\min}$ y $r_{\max}$ representarán el menor y mayor valor que podría tomar este nodo hoja en el árbol expandido que hemos aproximado. Estudiemos en primer lugar los detalles a tener en cuenta al construir el árbol para una valuación cuando hacemos uso de estos valores $r_{\min}$ y $r_{\max}$.

Representemos por $h^{R(X_J=u_J)}$ el conjunto de valores en la valuación h compatibles con la configuración de variables $X_J = u_J$. Sea un árbol $\mathcal{T}$ que se está utilizando para representar una valuación h definida para el conjunto de variables X_I y sea $X_J = u_J$ una configuración de algunas de las variables de h, las cuales conducen a un nodo hoja l en el árbol $\mathcal{T}$. El valor r almacenado en el hoja l puede pertenecer a una de las dos siguientes situaciones:

- r representa un valor exacto para todos los valores de la valuación h compatibles con la configuración $X_J = u_J$, entonces $r_{\min}$ y $r_{\max}$ serán iguales a r. O sea, cuando todos los valores que hay en $h^{R(X_J=u_J)}$ son iguales a r, entonces r está representando de forma exacta esta subvaluación y por tanto $r_{\min}$ y $r_{\max}$ serán iguales a r.

- r es una aproximación de la subvaluación $h^{R(X_J=u_J)}$. Realmente r será el valor medio de $h^{R(X_J=u_J)}$. En este caso el valor $r_{\min}$ y $r_{\max}$ se calcularán como $r_{\min} = \min_{u_J} h^{R(X_J=u_J)}$ y $r_{\max} = \max_{u_J} h^{R(X_J=u_J)}$ respectivamente.

Para las operaciones de combinación y marginalización hace falta también indicar como se opera con estos valores $r_{\min}$ y $r_{\max}$. Supongamos que $\mathcal{T}_1$ y $\mathcal{T}_2$ son dos árboles que van a combinarse obteniendo el árbol $\mathcal{T}$. Cuando no se hace aproximación, se está colocando en un nodo hoja de $\mathcal{T}$ como valor r el producto del valor r^1 de un nodo hoja de $\mathcal{T}_1$ y del valor r^2 de un nodo hoja de $\mathcal{T}_2$. O sea, si $X_J = u_J$, $X_K = u_K$ son las configuraciones de variables que determinan los nodos hoja de $\mathcal{T}_1$ y $\mathcal{T}_2$ que van a multiplicarse, entonces el valor r del nodo hoja de $\mathcal{T}$ se obtiene con el producto de los valores r que hay en los nodos hoja $\mathcal{T}_1^{R(X_J=u_J)}$ y $\mathcal{T}_2^{R(X_K=u_K)}$. El valor $r_{\min}$ se obtiene como el producto de los $r_{\min}$ de los nodos hoja $\mathcal{T}_1^{R(X_J=u_J)}$ y $\mathcal{T}_2^{R(X_K=u_K)}$. De igual forma, el valor $r_{\max}$ se obtiene con el producto de los $r_{\max}$ de $\mathcal{T}_1^{R(X_J=u_J)}$ y $\mathcal{T}_2^{R(X_K=u_K)}$. En el capítulo 4 vimos dos modos de realizar una aproximación en algunos nodos del árbol resultado:

1. Una forma era generando un árbol exacto para la combinación que luego sería podado obteniendo así un árbol aproximado. Para ello el árbol iba siendo podado por las hojas sustituyendo un nodo cuyos hijos fuesen todos hojas, por un nodo hoja con valor r igual a la media de los valores r de los nodos hoja que se podan. Al realizar esta poda el valor $r_{\min}$ del nuevo nodo hoja se obtiene como el valor mínimo de los $r_{\min}$ de los antiguos nodos hoja y el valor $r_{\max}$ se obtiene de forma análoga con el máximo de los $r_{\max}$ de los antiguos nodos hoja.

2. La otra forma de realizar la combinación aproximada hacía crecer el árbol resultado de arriba hacia abajo deteniéndose cuando éste alcanzase cierto tamaño prefijado. Al detener el proceso de combinación quedarían en el árbol resultado ciertos nodos hojas cuyos valores aún no estaban calculados pues debían haberse obtenido combinando dos subárboles de los árboles $\mathcal{T}_1$ y $\mathcal{T}_2$. En cada nodo hoja no calculado obteníamos la media de cada subárbol y colocábamos en tal nodo hoja el producto de ambos valores medios. Si $X_J = u_J$, $X_K = u_K$ determinan los subárboles $\mathcal{T}_1^{R(X_J=u_J)}$ y $\mathcal{T}_2^{R(X_K=u_K)}$ que debían combinarse y colocarse en un nodo hoja de $\mathcal{T}$, entonces el valor r que colocamos en el nodo hoja se obtiene con el producto de los valores medios de cada subárbol. El valor $r_{\min}$ se obtiene como el producto del valor mínimo de los $r_{\min}$ de $\mathcal{T}_1^{R(X_J=u_J}$) y del

valor mínimo de los $r_{\min}$ de $\mathcal{T}_2^{R(X_K=u_K)}$). El valor $r_{\max}$ se obtiene de forma análoga.

Para la operación de marginalización el cálculo de los $r_{\min}$ y $r_{\max}$ es bastante similar al caso de la combinación por lo que no hace falta que sea detallado.

Veamos un ejemplo en el que se ilustra como calcular los $r_{\min}$ y $r_{\max}$ en la combinación aproximada.

Ejemplo 5.4 Combinación de forma aproximada utilizando intervalos para acotar la aproximación

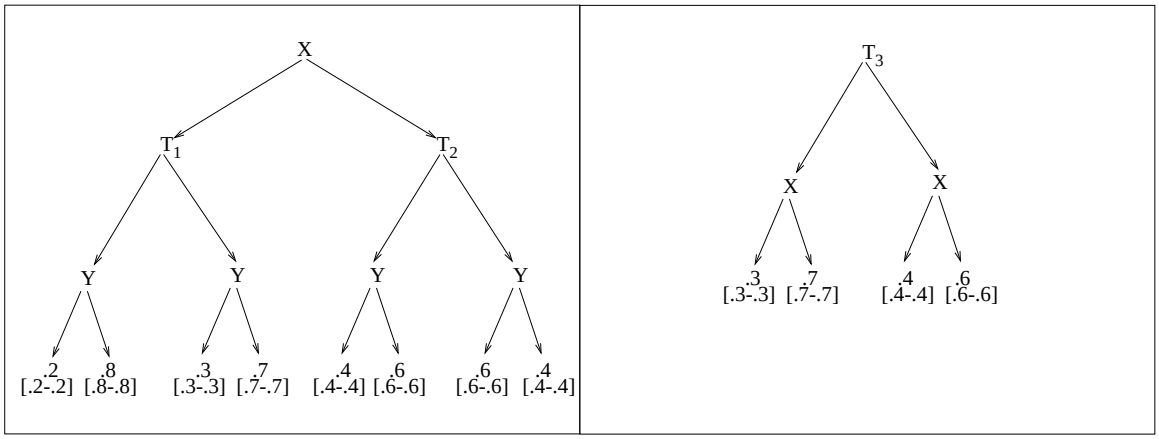


Figura 5.11: Dos árboles que queremos combinar de forma aproximada

Supongamos que queremos combinar de forma aproximada los dos árboles de la figura 5.11 en la segunda forma que hemos explicado más arriba. En estos dos árboles representamos en cada nodo hoja el valor r de siempre y además el intervalo (valor mínimo y máximo) en el que debe estar comprendido r. En este caso los valores $r_{\min}$ y $r_{\max}$ coinciden con r pues estos árboles no han sido aproximados y representan un resultado exacto.

En el algoritmo de combinación aproximada podríamos llegar a un estado del árbol resultado como el que se muestra en la figura 5.12 donde ya se han hecho algunas aproximaciones y donde aún quedan por combinar los dos subárboles encerrados por la línea discontinua. Supongamos que en el algoritmo de combinación aproximada se decide que por esta rama ya no se sigue expandiendo el árbol y que se hará una aproximación sustituyendo por un nodo hoja. La forma de hacer esta aproximación, como se vio en el capítulo 4, es poniendo en el nodo hoja el producto de los valores medios de cada subárbol que deben combinarse. En este caso la media

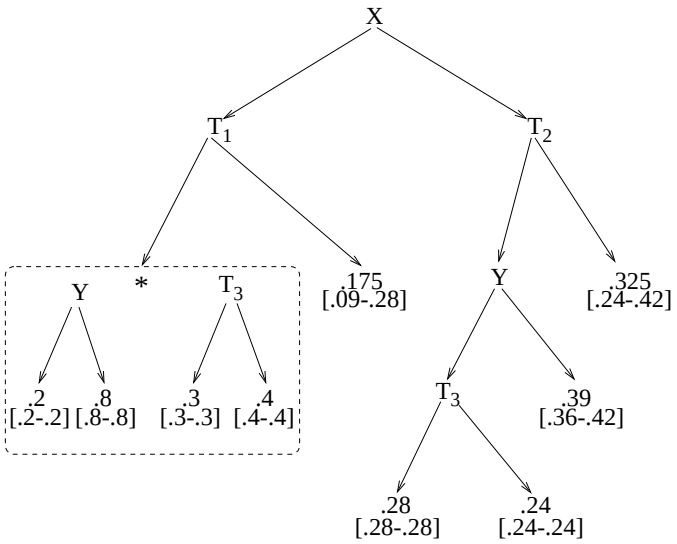


Figura 5.12: Árbol resultado aproximado aún sin acabar para la combinación de los árboles de la figura 5.11

del subárbol de la izquierda es 0.5 *y el de la derecha es* 0.35. *Por tanto el producto de ambos valores, o sea* 0.175, *será el que aparecerá en tal nodo hoja como valor* r. *El valor min se calculará como el producto de los valores mínimos de todos los* $r_{\min}$ *de cada subárbol y el valor max como el producto de los valores máximos de todos los* $r_{\max}$ *de cada subárbol. En nuestro caso tendremos* $r_{\min} = 0.2 \times 0.3 = 0.06$ *y* $r_{\max} = 0.8 \times 0.4 = 0.32$. *El árbol aproximado quedaría entonces como aparece en la figura 5.13.*

■

Por último, nos queda ver como obtener intervalos para cada caso de la variable objetivo con los valores $r_{\min}$ y $r_{\max}$. Estos nuevos intervalos acotarán a los anteriores: los intervalos calculados hasta ahora con el condicionamiento de Dempster estarán incluidos en estos nuevos intervalos. En los algoritmos de propagación por eliminación de variables y de propagación en un árbol de grupos, como resultado final de la propagación, se obtiene un árbol resultado $\mathcal{T}_j$ a partir del cual podemos obtener sus puntos extremos según se detalló en la sección 5.3.1. Ahora, en el árbol resultado $\mathcal{T}_j$ tendremos en cada nodo hoja tres valores (r, $r_{\min}$ y $r_{\max}$). Los valores $r_{\min}$ y $r_{\max}$ nos dan los valores entre los que podría oscilar el valor r. Estos valores nos permiten obtener unos intervalos adicionales a los que se pueden obtener con los valores r. Por ejemplo en la figura 5.14 podemos ver un posible árbol resultado de la propagación para la variable X. Para obtener los intervalos para los casos de la variable X

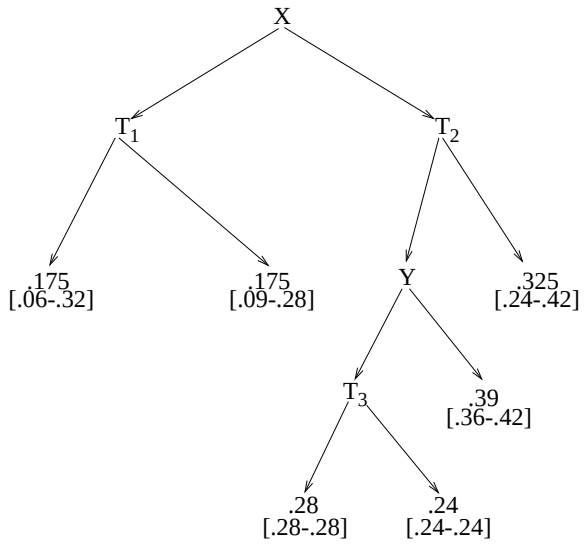


Figura 5.13: Árbol resultado aproximado correspondiente a la combinación de los árboles de la figura 5.11

aplicamos el mismo algoritmo detallado en la sección 5.3.1 pero con una pequeña modificación. Antes para resolver el caso simple (situación 1) se utilizaba una búsqueda del nodo hoja con mínimo valor r y del nodo hoja con máximo valor r. La única modificación que habría que hacer es que el cálculo del mínimo se hace como el menor de los $r_{\min}$ y el máximo como el mayor de los $r_{\max}$. En el caso de la figura 5.3.1 el árbol se descompondría en los dos árboles que se muestran en la figura 5.15 puesto que es un árbol que pertenece a la situación segunda.

Los dos árboles de la figura 5.15 ya corresponden a la situación primera, por lo que haríamos el cálculo de los mínimos y los máximos según hemos indicado más arriba y según se describe en la sección 5.3.1 para la situación primera. De esta forma obtendríamos los puntos de la tabla 5.1, la cual nos permite obtener los intervalos con el condicionamiento de Dempster [39] en todos los casos de la variable X.

5.4 Experimentación

En el caso del condicionamiento de Dempster, si usamos los puntos calculados con $r_{\min}$ y $r_{\max}$ obtendremos intervalos que incluyen al intervalo exacto. Los algoritmos aproximados que se proponen en el capítulo 3 proporcionan intervalos que están incluidos en el intervalo exacto. Podemos obtener así dos aproximaciones, una interior y otra exterior, del intervalo final. Si éstas están muy próximas, sabemos que tendremos una muy buena aproximación del

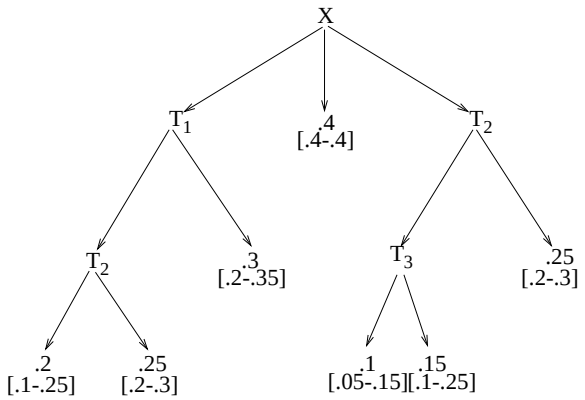


Figura 5.14: Un árbol de probabilidad resultado para la variable objetivo X en donde se muestran los valores r, $r_{\min}$ y $r_{\max}$

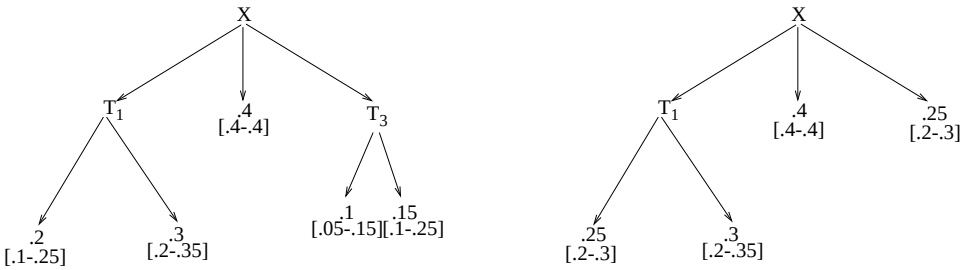


Figura 5.15: Descomposición en dos árboles del árbol de la figura 5.14

intervalo final.

En esta sección vamos a detallar la experimentación realizada con grafos de dependencias donde la información viene dada por intervalos. Hemos realizado una experimentación que ha intentando evaluar el algoritmo de eliminación de variables usando árboles de probabilidad, con un límite a su tamaño, obteniéndose resultados aproximados. También hemos utilizado el algoritmo de enfriamiento simulado descrito en la sección 3.5.4 y que estaba orientado a calcular los intervalos 'a posteriori' para una variable con el condicionamiento de Dempster [39]. En este algoritmo también haremos uso de los árboles de probabilidad. Los programas han sido ejecutados en un ordenador Pentium II de 450 MHz con sistema operativo Linux.

En la experimentación llevada a cabo siempre hemos calculado los intervalos utilizando el condicionamiento de Dempster [39] para el primer caso de alguna de las variables del grafo de dependencias. En las tablas que aparecerán en esta sección denotaremos con L_i al límite

u_1	u_2	u_3
0.1	0.4	0.05
0.1	0.4	0.25
0.35	0.4	0.05
0.35	0.4	0.25
0.2	0.4	0.2
0.2	0.4	0.3

Tabla 5.1: Puntos que se obtienen con el uso de los valores $r_{\min}$ y $r_{\max}$ que permiten obtener los intervalos para X con el condicionamiento de Dempster

inferior de tal intervalo, con L_s al límite superior, y con L_{imin} y L_{smax} los límites inferior y superior respectivamente de los intervalos obtenidos según el método de la sección 5.3.3. Denotaremos por ts el tiempo en segundos en la ejecución de los programas. Las pruebas realizadas se han llevado a cabo con varias redes Bayesianas disponibles a través de Internet en el repositorio de Nir Friedman [51] y en el libro de Jensen [75] cuyas redes están disponibles en la dirección "http://www.hugin.dk/networks". Estas redes contienen probabilidades. Nosotros hemos transformado cada probabilidad p de la red Bayesiana en un intervalo generado aleatoriamente. Para ello se ha construido un pequeño programa que recibe como entrada una desviación máxima d permitida sobre la probabilidad p para generar el intervalo. Para cada número p se genera un número aleatorio r uniforme entre 0 y el mínimo de los siguientes tres valores: p, $1-p$ y d. Con el número r generado se obtiene el intervalo a partir de la antigua probabilidad p con: $[p-r, p+r]$. Esta forma de generar el intervalo asegura que $p-r >= 0$ y $p+r <= 1$. También asegura que si $p = 0.0$ entonces el intervalo generado será $[0.0, 0.0]$ y si $p = 1.0$ el intervalo generado entonces será $[1.0 - 1.0]$.

Los experimentos llevados a cabo han sido de tres tipos:

1. Propagación con el algoritmo de eliminación de variables. Aquí las operaciones de combinación y marginalización utilizadas son las que se describen en la sección 4.3.2.3. Concretamente las que van construyendo el árbol resultado de la operación poco a poco partiendo del nodo raíz del resultado y deteniéndose en el momento que el árbol resultado alcanzase cierto tamaño preestablecido. Estos experimentos se realizarán con diferentes tamaños máximo para los árboles. Llamaremos a este algoritmo PropWithTD1.

2. Propagación con el algoritmo de eliminación de variables. Ahora las operaciones de combinación y marginalización se hacen de forma exacta y posteriormente reducimos el árbol resultado al tamaño preestablecido sustituyendo nodos cuyos hijos son todos hoja por nodos hoja etiquetados con el valor medio de todos sus hijos. Estos experimentos se realizarán con diferentes tamaños máximo para los árboles. Llamaremos a este algoritmo PropWithTD3.

3. Propagación con el algoritmo de enfriamiento simulado de la sección 3.5.4. Este algoritmo estaba enfocado a obtener puntos del conjunto convexo 'a posteriori' para la variable de interés X_j que minimizasen el error cometido al dar los intervalos para esta variable con el condicionamiento de Dempster [39]. El algoritmo necesita un doble sistema de mensajes. Uno es el normal, y el otro el que transportaba la instanciación de uno de los casos de la variable X_j. El algoritmo hay que repetirlo para cada valor de la variable X_j para así ajustar los límites inferiores y superiores de los intervalos de X_j. En los experimentos con este algoritmo veremos las diferencias que se obtienen cuando el algoritmo se enfoca a obtener el límite inferior del intervalo o al límite superior en el primer caso de la variable X_j. El algoritmo se aplicará con diferentes números de iteraciones para ver si converge a la solución exacta. Llamaremos a este método PropSim4. En este algoritmo se necesitan dos parámetros de entrada. Son la temperatura inicial que hemos denotado por t_0 y el factor de enfriamiento que hemos denotado por α. El mecanismo de enfriamiento de la temperatura empleado ha sido el de Kirkpatrick, Gelatt y Vecchi [80]. El mecanismo de transición de una configuración a otra es el propuesto por Green y Supowit [61] que fue detallado en la sección 3.5.2.1 con el uso de la expresión 3.11 que reproducimos a continuación:

$$e^{-PS_{T_i}/t} \tag{5.3}$$

donde T_i es la variable transparente a simular, PS_{T_i} es su vector 'a posteriori', e es la base de los logaritmos neperianos y $e^{-PS_{T_i}/t}(c_j) = e^{-PS_{T_i}(c_j)/t}$,

En los experimentos 1 y 2 hemos propagado además con los mínimos y los máximos según se describe en la sección 5.3.3 para así comprobar como acotan estos nuevos intervalos a los intervalos obtenidos mediante aproximaciones en los árboles.

5.4.1 Red Boerlage92

La red Boerlage92 [9] se muestra en la figura 5.16. A esta red le hemos sustituido las probabilidades por intervalos según se describe más arriba. Sobre la red transformada hemos realizado experimentos para obtener los intervalos con el condicionamiento de Dempster [39] en las siguientes variables:

1. Variable "TD: Tom" (variable con 2 casos) sin observaciones en ninguna otra variable de la red. Llamaremos a esta configuración Boe1.

2. Variable "MT: Molly" (variable con 2 casos) sin observaciones en ninguna otra variable de la red. Llamaremos a esta configuración Boe2.

3. Variable "TD: Tom" (variable con 2 casos) con observación del primer caso de la variable variable "TR: Tom". Llamaremos a esta configuración Boe3.

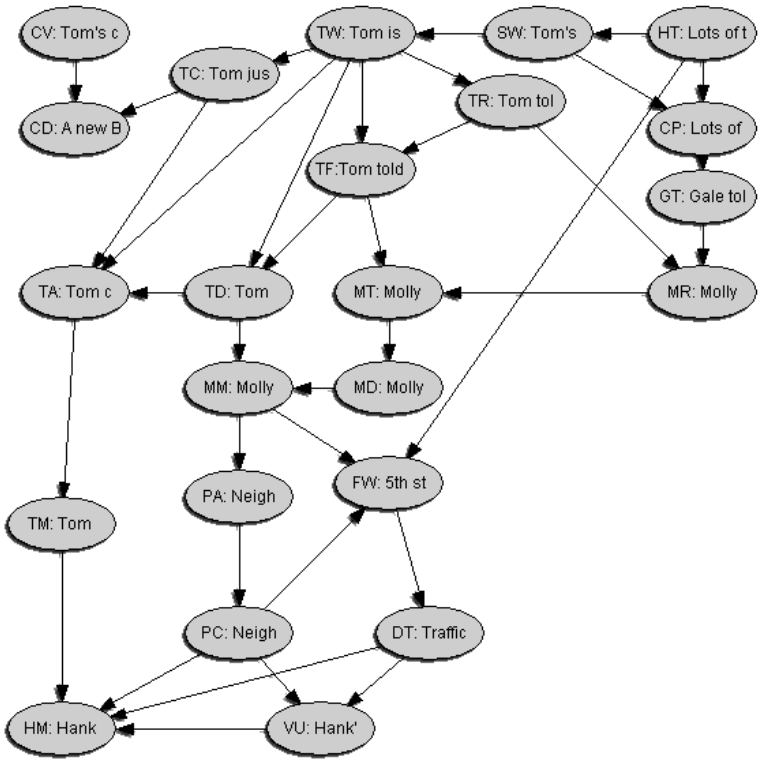


Figura 5.16: Red Boerlage92 [9]: Es un modelo para un escenario particular de sucesos vecinos

Veamos los experimentos realizados:

- En la tabla 5.2 mostramos los intervalos que hemos obtenido con los métodos de propagación por eliminación de variables PropWithTD1 y PropWithTD3 empleando diferentes tamaños máximos para los árboles con la configuración Boe1. También se muestra el tiempo empleado por los algoritmos. La figura 5.17 muestra gráficamente los resultados.

- En la tabla 5.3 mostramos los intervalos que hemos obtenido con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ para diferentes número de iteraciones del algoritmo en la configuración Boe1. También se muestra el tiempo empleado por los algoritmos. La figura 5.18 muestra gráficamente los resultados.

- En la tabla 5.4 mostramos los intervalos que hemos obtenido con los métodos de propagación por eliminación de variables PropWithTD1 y PropWithTD3 empleando diferentes tamaños máximos para los árboles con la configuración Boe2. También se muestra el tiempo empleado por los algoritmos. La figura 5.19 muestra gráficamente los resultados.

- En la tabla 5.5 mostramos los intervalos que hemos obtenido con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ para diferentes número de iteraciones del algoritmo en la configuración Boe2. También se muestra el tiempo empleado por los algoritmos. La figura 5.20 muestra gráficamente los resultados.

- En la tabla 5.6 mostramos los intervalos que hemos obtenido con los métodos de propagación por eliminación de variables PropWithTD1 y PropWithTD3 empleando diferentes tamaños máximos para los árboles con la configuración Boe3. También se muestra el tiempo empleado por los algoritmos. La figura 5.21 muestra gráficamente los resultados.

- En la tabla 5.7 mostramos los intervalos que hemos obtenido con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ para diferentes número de iteraciones del algoritmo con la configuración Boe3. También se muestra el tiempo empleado por los algoritmos. La figura 5.22 muestra gráficamente los resultados.

	10000	20000	30000	40000	50000	65000	70000	100000	200000
L_i	0.131587	0.130855	0.130740	0.130740	0.130740	0.130733	0.124713	0.124487	0.124486
L_s	0.212634	0.213963	0.213963	0.213973	0.213973	0.213973	0.224924	0.225327	0.225327
L_{imin}	0.114457	0.114457	0.114457	0.114457	0.114457	0.114457	0.124486	0.124486	0.124486
L_{smax}	0.243323	0.243323	0.243323	0.243323	0.243323	0.243323	0.225327	0.225327	0.225327
ts	6.5	13.5	29.4	34	34.5	35	37	61	62

	5000	10000	20000	30000	40000	50000	100000
L_i	0.128374	0.124875	0.124724	0.124671	0.124620	0.124509	0.124487
L_s	0.232749	0.224244	0.224336	0.224809	0.225223	0.225312	0.225327
L_{imin}	0.110878	0.111654	0.114568	0.114653	0.124486	0.124486	0.124486
L_{smax}	0.247272	0.246851	0.242907	0.242698	0.225327	0.225327	0.225.327
ts	69.7	153.7	192.7	273.55	366.7	565.6	878.9

Tabla 5.2: Intervalos y tiempos obtenidos con PropWithTD1 (tabla de arriba) y PropWithTD3 (tabla de abajo) en configuración Boe1 en red $Boerlage92$.

	10	25	50	75	100	125	150	175	200	250
L_i	0.126360	0.130738	0.124501	0.124486	0.124486	0.124486	0.124486	0.124486	0.124486	0.124486
L_s	0.193360	0.200900	0.215113	0.213233	0.194150	0.202995	0.198787	0.201897	0.217429	0.218858
ts	89.4	225.2	452.2	678	793	818.5	844.3	861.2	896.3	954.2

	10	25	50	75	100	125	150	175	200	250
L_i	0.139193	0.129224	0.143977	0.134812	0.131861	0.127772	0.139194	0.131488	0.130009	0.140092
L_s	0.202389	0.195358	0.222293	0.225327	0.225327	0.225327	0.225327	0.225327	0.225327	0.225327
ts	89.6	226.3	454.2	681	908.7	953.2	976.6	1003.3	1028.6	1080.7

Tabla 5.3: Intervalos y tiempos obtenidos con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Boe1 en red $Boerlage92$ enfocando a optimizar el límite inferior (tabla de arriba) y el superior (tabla de abajo)

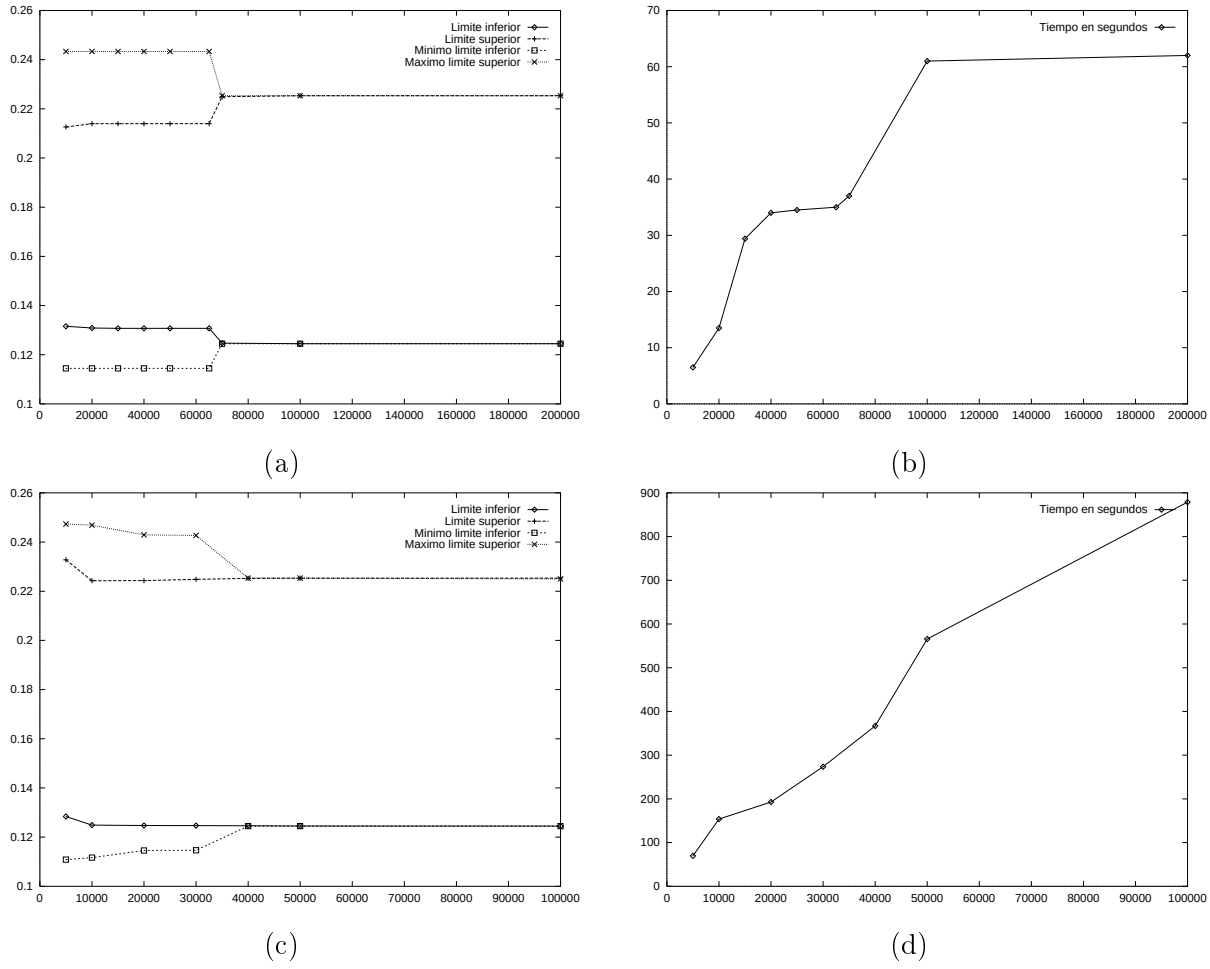


Figura 5.17: Algoritmo PropWithTD1 en configuración Boe1 en red $Boerlage92$: Intervalos (figura a) y tiempos (figura b) para cada tamaño de árbol (tabla 5.2 de arriba). Algoritmo PropWithTD3 en configuración Boe1 en red $Boerlage92$: Intervalos (figura c) y tiempos (figura d) para cada tamaño de árbol (tabla 5.2 de abajo).

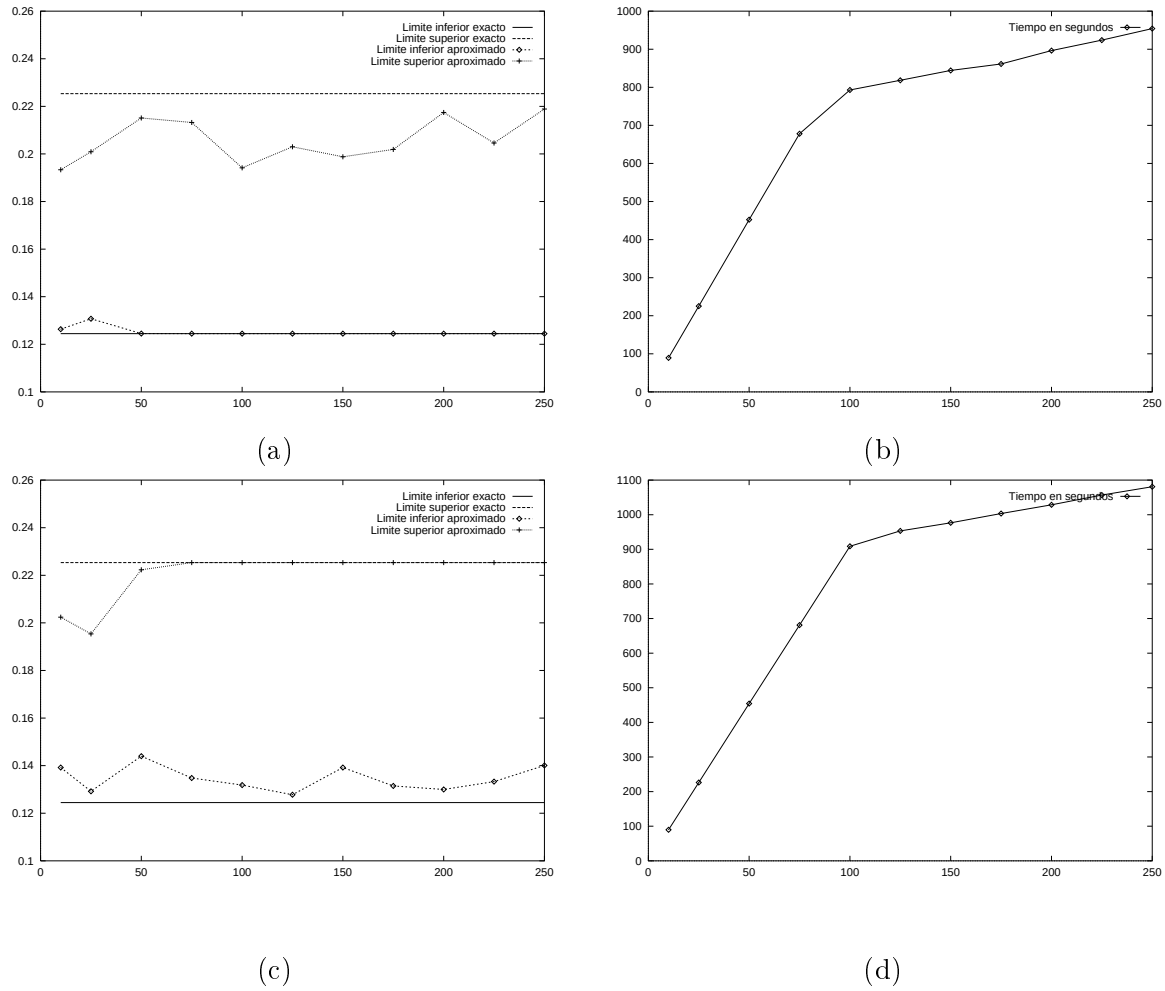


Figura 5.18: Algoritmo PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Boe1 en red $Boerlage92$: Intervalos (figura a) y tiempos (figura b) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite inferior (tabla 5.3 de arriba) e intervalos (figura c) y tiempos (figura d) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite superior (tabla 5.3 de abajo)

	10000	20000	30000	40000	50000	60000
L_i	0.105220	0.102918	0.102548	0.101518	0.099977	0.099885
L_s	0.203419	0.203787	0.204783	0.202381	0.203109	0.200739
L_{imin}	0.042534	0.042997	0.042997	0.042997	0.043098	0.043432
L_{smax}	0.528020	0.527453	0.527453	0.527453	0.527453	0.577453
ts	185	405	818	1497	2793	10529

	5000	10000	20000	30000	40000	50000	60000	100000
L_i	0.100187	0.295483	0.107630	0.412786	0.301805	0.286207	0.230603	0.447639
L_s	0.511125	0.500000	0.516692	0.522224	0.514695	0.514841	0.514103	0.500005
L_{imin}	0.024886	0.020750	0.010792	0.049314		0.029298	0.034785	0.032752
L_{smax}	0.975113	0.979249	0.989207	0.950685		0.970701	0.965214	0.967247
ts	228.6	390.7	1990.6	3180.1	7138.8	7839.7	6447	25571

Tabla 5.4: Intervalos y tiempos obtenidos con PropWithTD1 (tabla de arriba) y PropWithTD3 (tabla de abajo) en configuración Boe2 en red $Boerlage92$

	20	30	40	50	60
L_i	0.106938	0.091696	0.086195	0.083794	0.083479
L_s	0.182036	0.175355	0.167394	0.168782	0.186407
ts	917	1378	1840	2301	2766

	20	30	40	50	60
L_i	0.106115	0.092669	0.102390	0.096349	0.091533
L_s	0.183890	0.165193	0.187181	0.198273	0.200985
ts	940.3	1417.2	1891.1	2378	2838

Tabla 5.5: Intervalos y tiempos obtenidos con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Boe2 en red $Boerlage92$ enfocando a optimizar el límite inferior (tabla de arriba) y el superior (tabla de abajo)

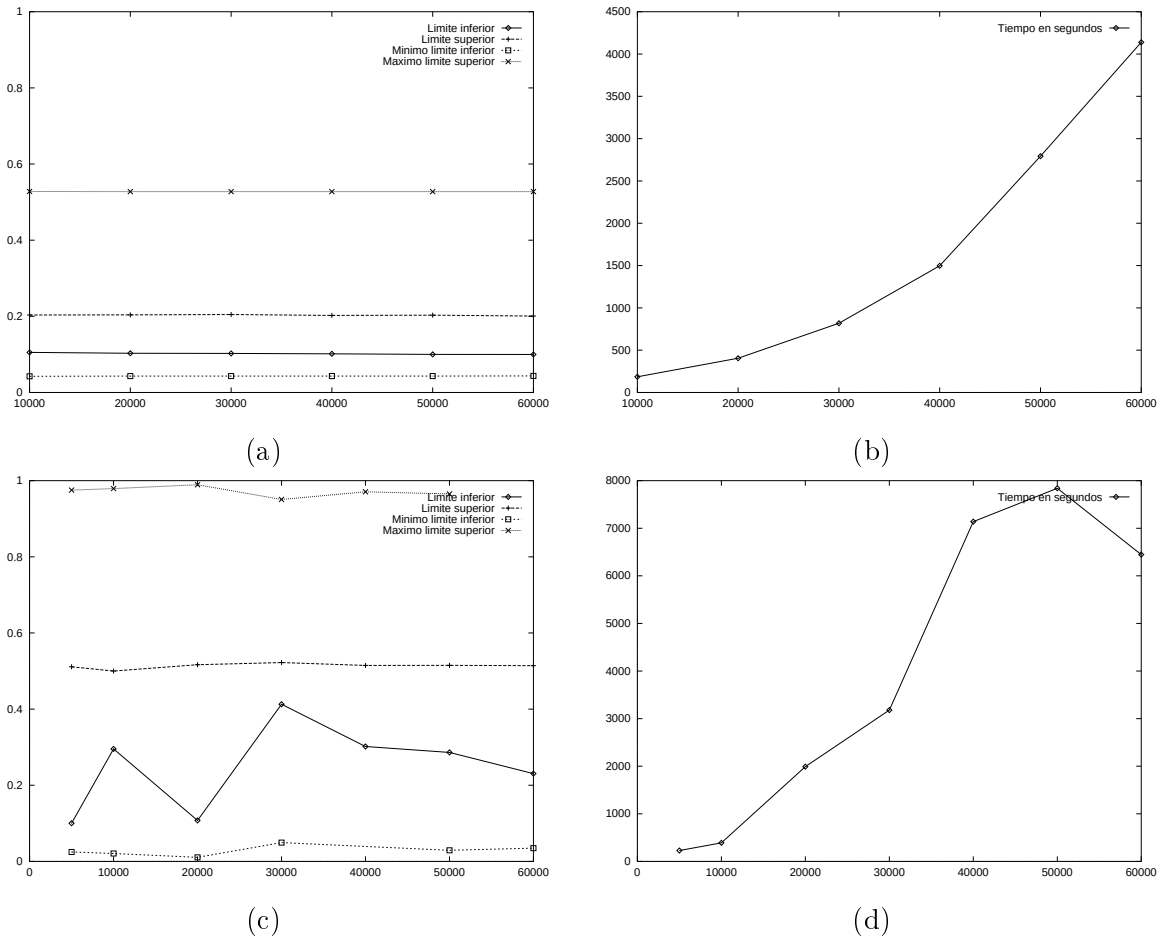


Figura 5.19: Algoritmo PropWithTD1 en configuración Boe2 en red $Boerlage92$: Intervalos (figura a) y tiempos (figura b) para cada tamaño de árbol (tabla 5.4 de arriba). Algoritmo PropWithTD3 en configuración Boe2 en red $Boerlage92$: Intervalos (figura c) y tiempos (figura d) para cada tamaño de árbol (tabla 5.4 de abajo).

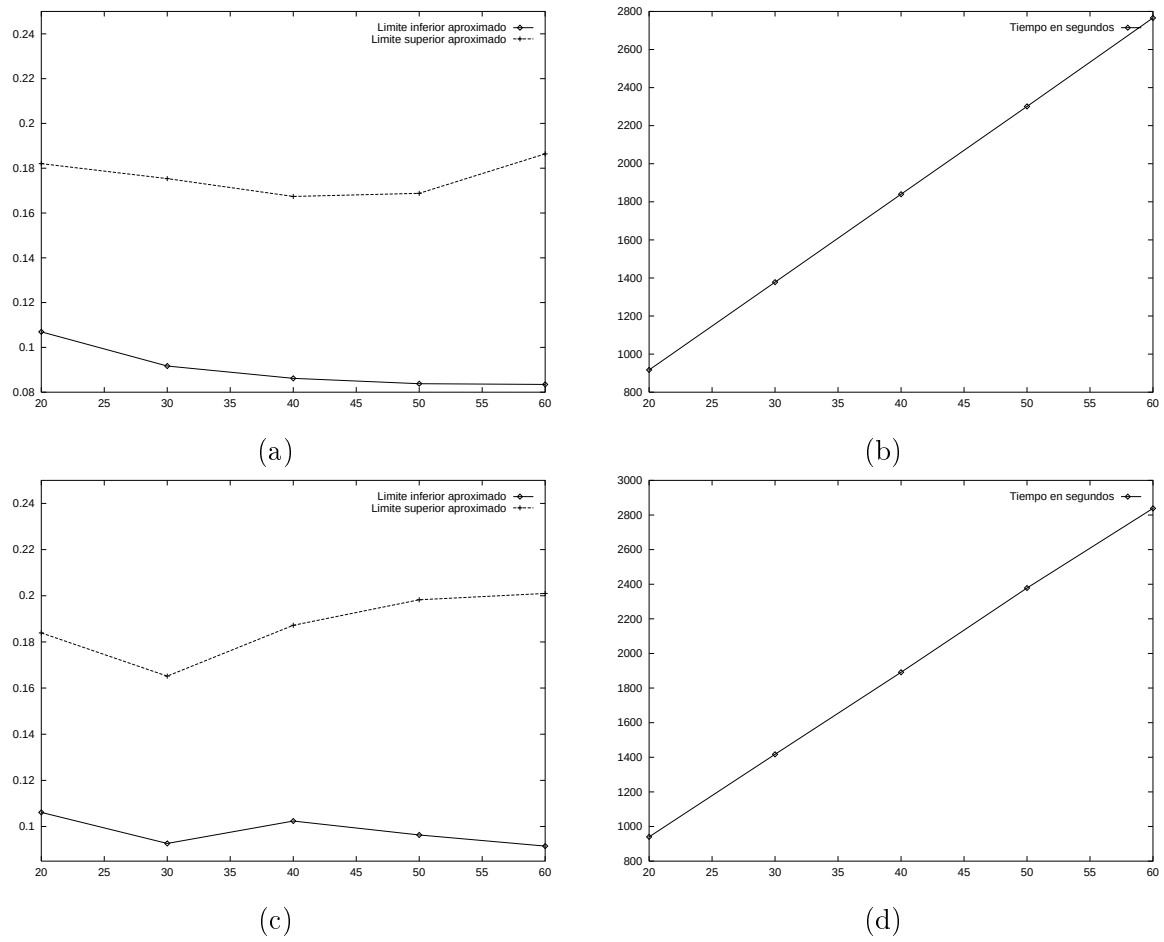


Figura 5.20: Algoritmo PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Boe2 en red $Boerlage92$: Intervalos (figura a) y tiempos (figura b) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite inferior (tabla 5.5 de arriba) e intervalos (figura c) y tiempos (figura d) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite superior (tabla 5.5 de abajo)

	1000	2500	5000	7500	10000	12500	15000	17500	20000	40000
L_i	0.272973	0.270832	0.251653	0.251653	0.251358	0.222883	0.222883	0.222883	0.222883	0.222867
L_s	0.489364	0.489273	0.489545	0.489807	0.495341	0.522341	0.529459	0.531509	0.531531	0.531531
L_{imin}	0.190970	0.193818	0.194207	0.194207	0.194207	0.209124	0.222867	0.222867	0.222867	0.222867
L_{smax}	0.549067	0.544407	0.542507	0.542507	0.542507	0.542507	0.542507	0.531531	0.531531	0.531531
ts	1	2.5	3.1	4	5	6.1	7.4	8.3	9.2	12.4

	1000	2500	5000	7500	10000	12500	17500	22500	32500	35000
L_i	0.477410	0.259620	0.223022	0.222883	0.222883	0.222883	0.222883	0.222883	0.222871	0.222867
L_s	0.553631	0.553631	0.553042	0.531183	0.531418	0.531418	0.531531	0.531531	0.531531	0.531531
L_{imin}	0.121570	0.116982	0.118544	0.132799	0.133030	0.133030	0.139251	0.148725	0.153454	0.222867
L_{smax}	0.878429	0.883220	0.881455	0.870560	0.867066	0.812689	0.810565	0.808564	0.803545	0.531531
ts	1	5.5	20	29	30	30	30	31	63	65

Tabla 5.6: Intervalos y tiempos obtenidos con PropWithTD1 (tabla de arriba) y PropWithTD3 (tabla de abajo) en configuración Boe3 en red $Boerlage92$

	20	30	40	50	60
L_i	0.232708	0.229649	0.226197	0.222883	0.222867
L_s	0.440250	0.519099	0.521723	0.514269	0.517772
ts	918.1	1379.9	1842.1	2304.6	2773.2

	20	30	40	50	60
L_i	0.270542	0.263867	0.262177	0.278811	0.270866
L_s	0.509944	0.525528	0.529359	0.530418	0.529899
ts	921.2	1386.8	1848.6	2309.3	2772

Tabla 5.7: Intervalos y tiempos obtenidos con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Boe3 en red $Boerlage92$ enfocando a optimizar el límite inferior (tabla de arriba) y el superior (tabla de abajo)

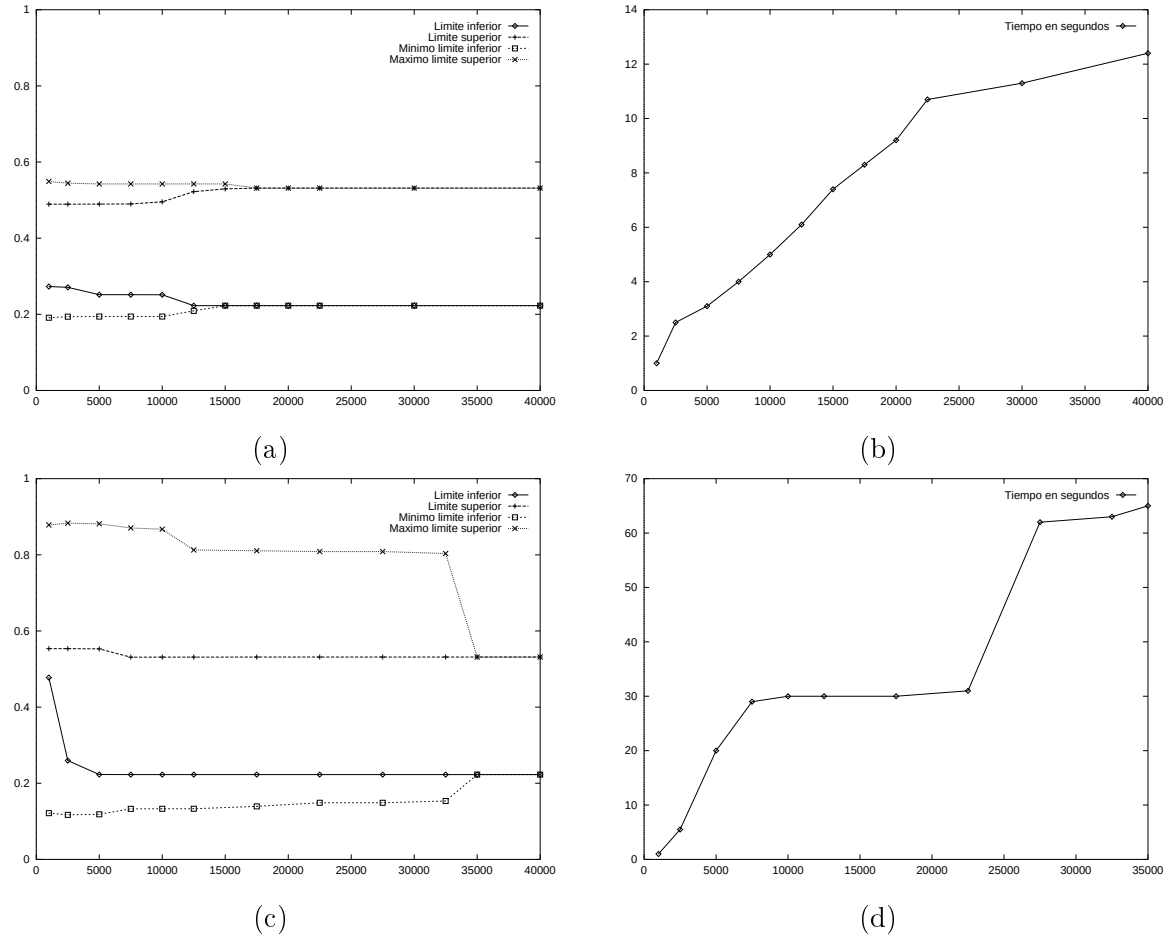


Figura 5.21: Algoritmo PropWithTD1 en configuración Boe3 en red $Boerlage92$: Intervalos (figura a) y tiempos (figura b) para cada tamaño de árbol (tabla 5.6 de arriba). Algoritmo PropWithTD3 en configuración Boe3 en red $Boerlage92$: Intervalos (figura c) y tiempos (figura d) para cada tamaño de árbol (tabla 5.6 de abajo).

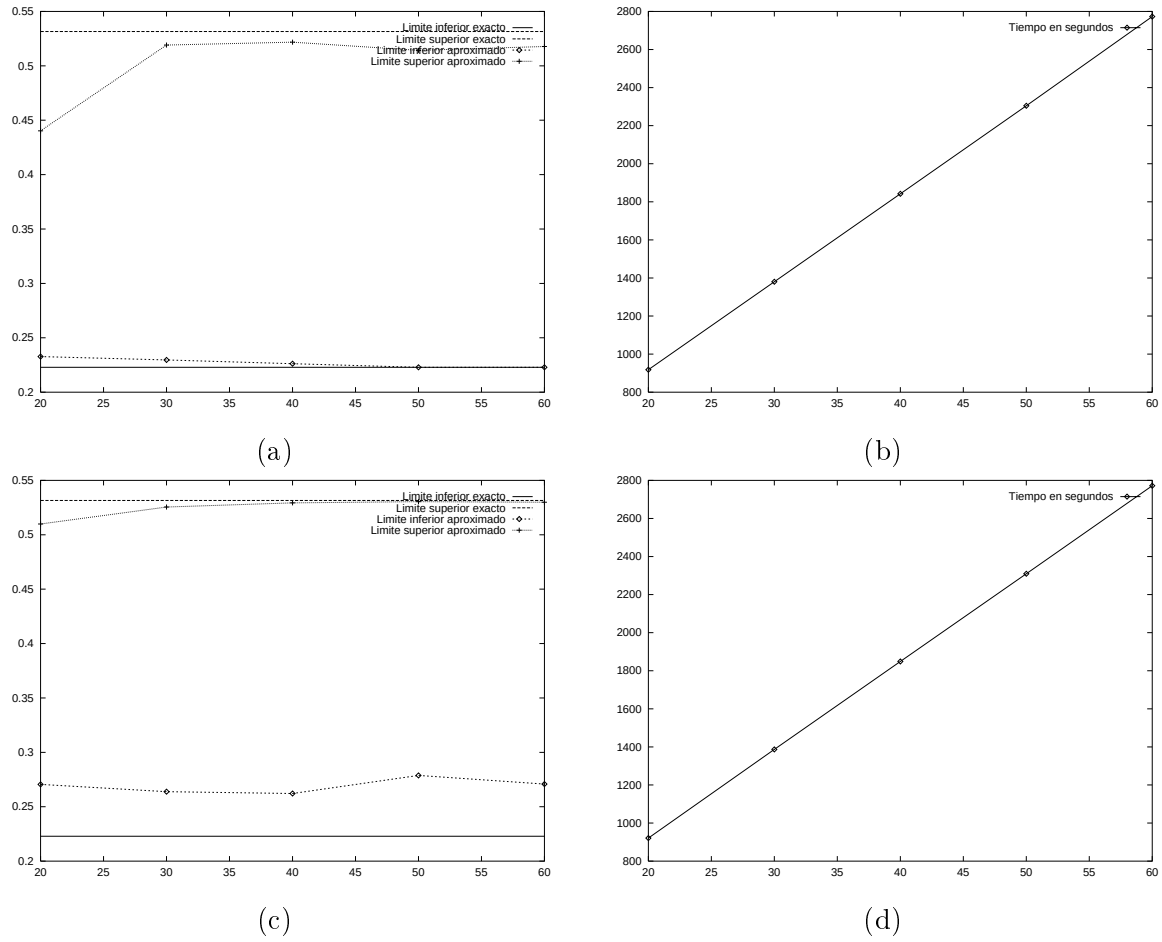


Figura 5.22: Algoritmo PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Boe3 en red $Boerlage92$: Intervalos (figura a) y tiempos (figura b) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite inferior (tabla 5.7 de arriba) e intervalos (figura c) y tiempos (figura d) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite superior (tabla 5.7 de abajo)

5.4.2 Red Boblo

La red *Boblo* [111, 112] se muestra en la figura 5.23. De nuevo hemos sustituido las probabilidades por intervalos de la misma forma que con la red *Boerlage*. Sobre la red transformada hemos realizado experimentos para obtener los intervalos con el condicionamiento de Dempster [39] en las siguientes variables:

1. Variable "factor 1 (F1)" (variable con 2 casos) sin observaciones en ninguna otra variable de la red. Llamaremos a esta configuración Bob1.

2. Variable "factor 1 (F1)" (variable con 2 casos) con observaciones en el grafo de dependencias en las variables "pheno st 1 dam" (caso 3), "pheno st 2 dam" (caso 2) y "pheno true sire" (caso 3). Llamaremos a esta configuración Bob2.

En esta red ha sido imposible aplicar el método de propagación por eliminación de variables PropWithTD3, debido a que el algoritmo consumía toda la memoria de nuestro ordenador y era incapaz de obtener ningún resultado.

Veamos los experimentos realizados:

- En la tabla 5.8 mostramos los intervalos que hemos obtenido con el método de propagación por eliminación de variable PropWithTD1 empleando diferentes tamaños máximos para los árboles con la configuración Bob1. También se muestra el tiempo empleado por los algoritmos. La figura 5.24 muestra gráficamente los resultados.

- En la tabla 5.9 mostramos los intervalos que hemos obtenido con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ para diferentes número de iteraciones del algoritmo en la configuración Bob1. También se muestra el tiempo empleado por los algoritmos. La figura fig5.29 muestra gráficamente los resultados.

- En la tabla 5.10 mostramos los intervalos que hemos obtenido con el método de propagación por eliminación de variable PropWithTD1 empleando diferentes tamaños máximos para los árboles con la configuración Bob2. También se muestra el tiempo empleado por los algoritmos. La figura 5.26 muestra gráficamente los resultados.

- En la tabla 5.11 mostramos los intervalos que hemos obtenido con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ para diferentes número de iteraciones del algoritmo en la configuración Bob2.

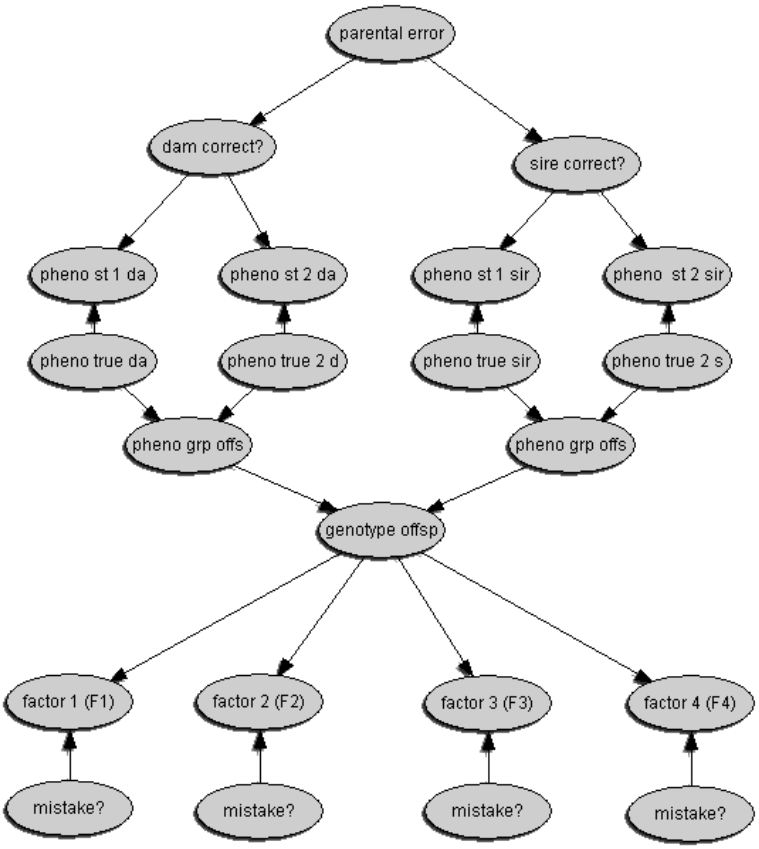


Figura 5.23: Red Boblo [111, 112]: Es un sistema de ayuda para la verificación del parentesco del ganado con el uso de la identificación del tipo de sangre

También se muestra el tiempo empleado por los algoritmos. La figura fig5.30 muestra gráficamente los resultados.

	20000	30000	40000	50000	60000
L_i	0.5	0.5	0.5	0.5	0.5
L_s	0.5	0.5	0.5	0.5	0.5
L_{imin}	0.000402	0.000402	0.000333	0.000333	0.000333
L_{smax}	0.999597	0.999597	0.999666	0.999666	0.999666
ts		4423.4	7092.4	8627	10294

Tabla 5.8: Intervalos y tiempos obtenidos con PropWithTD1 en configuración Bob1 en la red $Boblo$

	20	30	40	50	60	70	80	90
L_i	0.773434	0.777502	0.774442	0.761917	0.755249	0.755270	0.755260	0.755215
L_s	0.866201	0.851082	0.864691	0.866209	0.879773	0.867118	0.875125	0.843488
ts	1089	1644	2193	2758	3291	3857	4391	4936

	20	30	40	50	60	70	80	90
L_i	0.780240	0.797924	0.778357	0.776587	0.783588	0.775994	0.796941	0.791882
L_s	0.864236	0.868573	0.875217	0.882413	0.882814	0.882840	0.882915	0.882919
ts	1148	1720	2297	2871	3460	4030	4600	4999

Tabla 5.9: Intervalos y tiempos obtenidos con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Bob1 en red $Boblo$ enfocando a optimizar el límite inferior (tabla de arriba) y el superior (tabla de abajo)

	20000	30000	40000	50000	60000
L_i	0.5	0.429848	0.429848	0.475508	0.479056
L_s	0.550398	0.552048	0.552048	0.552055	0.543777
L_{imin}	0.0	0.0	0.0	0.0	0.0
L_{smax}	1.0	1.0	1.0	1.0	1.0
ts	1218	2436	4956	9650	14179

Tabla 5.10: Intervalos y tiempos obtenidos con PropWithTD1 en configuración Bob2 en la red $Boblo$

	20	30	40	50	60	70	80	90
L_i	0.231435	0.231910	0.231426	0.231018	0.230938	0.230870	0.230860	0.230847
L_s	0.397681	0.383940	0.375738	0.379450	0.371312	0.366310	0.385237	0.384540
ts	986	1481	1992	2485	2980	3484	3985	4080

	20	30	40	50	60	70	80	90
L_i	0.231526	0.231953	0.231924	0.232731	0.235411	0.232304	0.235498	0.232333
L_s	0.388566	0.401431	0.402646	0.406972	0.414314	0.415828	0.418189	0.418508
ts	1023	1537	2046	2579	3096	3623	4129	4627

Tabla 5.11: Intervalos y tiempos obtenidos con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Bob2 en red $Boblo$ enfocando a optimizar el límite inferior (tabla de arriba) y el superior (tabla de abajo)

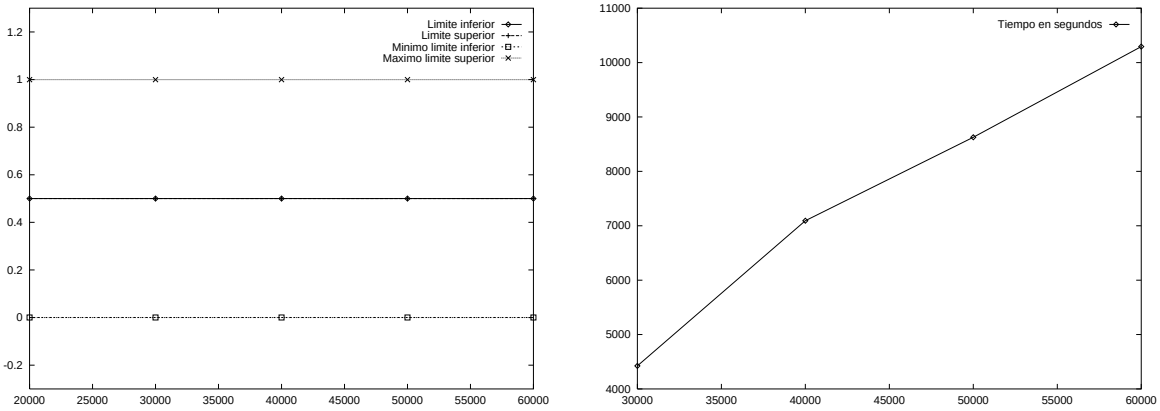


Figura 5.24: Algoritmo PropWithTD1 en configuración Bob1 en red $Boblo$: Intervalos (figura a) y tiempos (figura b) para cada tamaño de árbol (tabla 5.8).

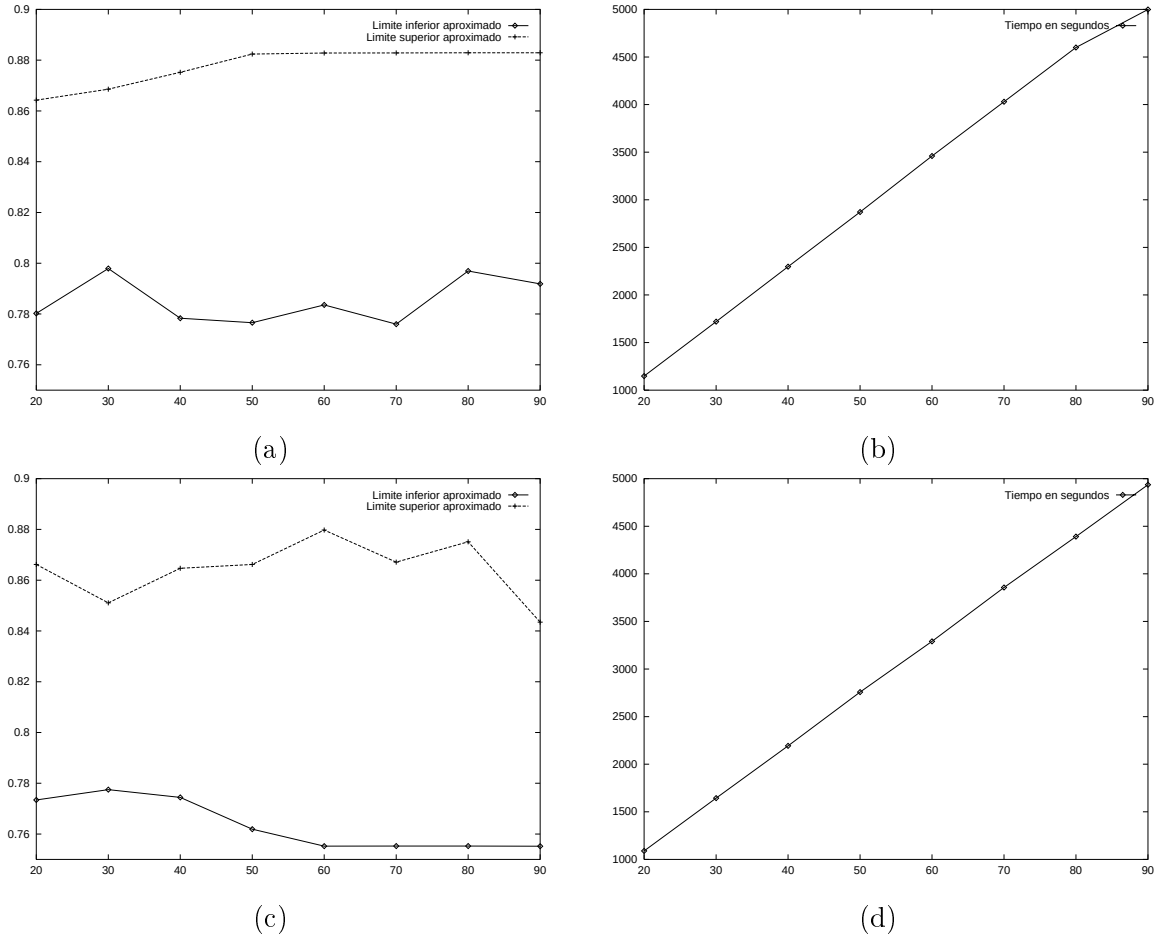
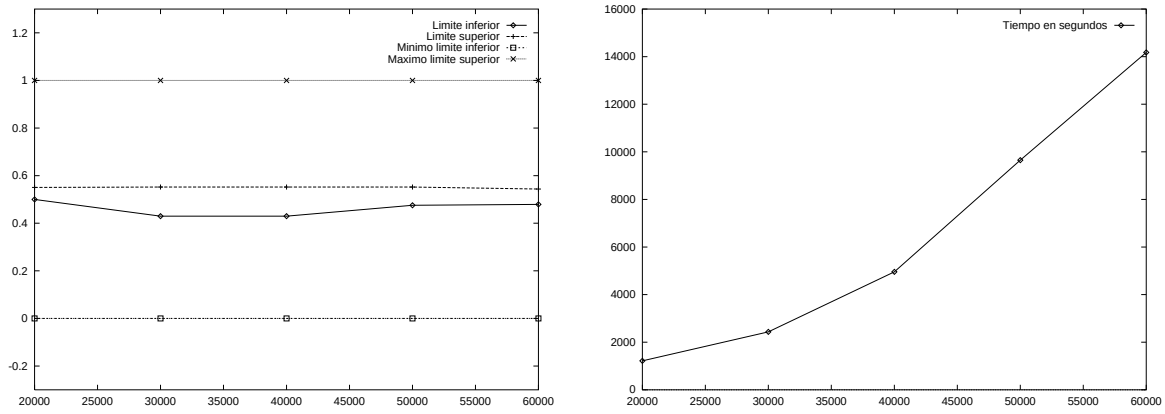


Figura 5.25: Algoritmo PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Bob1 en red $Boblo$: Intervalos (figura a) y tiempos (figura b) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite inferior (tabla 5.9 de arriba) e intervalos (figura c) y tiempos (figura d) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite superior (tabla 5.9 de abajo)



(a) (b)

Figura 5.26: Algoritmo PropWithTD1 en configuración Bob2 en red $Boblo$: Intervalos (figura a) y tiempos (figura b) para cada tamaño de árbol (tabla 5.10).

5.4.3 Red Coche (Car Starts)

La red *Coche* [1] se muestra en la figura 5.23. De nuevo hemos sustituido las probabilidades por intervalos de la misma forma que con las dos redes anteriores. Sobre la red transformada hemos realizado experimentos para obtener los intervalos con el condicionamiento de Dempster [39] en las siguientes variables:

1. Variable "Arranca" (variable con 2 casos) sin observaciones en ninguna otra variable de la red. Llamaremos a esta configuración Coc1.

2. Variable "Arranca" (variable con 2 casos) con observaciones en el grafo de dependencias en las variables "Alternador" (caso 2) y "Faros" (caso 2). Llamaremos a esta configuración Coc2.

Veamos los experimentos realizados:

- En la tabla 5.12 mostramos los intervalos que hemos obtenido con los métodos de propagación por eliminación de variables PropWithTD1 y PropWithTD3 empleando diferentes tamaños máximos para los árboles con la configuración Coc1. También se muestra el tiempo empleado por los algoritmos. La figura 5.29 muestra gráficamente los resultados.

[1] Esta red ha sido extraída del paquete *JavaBayes* que fue programado por Fabio Cozman y que está accesible en Internet en la dirección web "http://www.cs.cmu.edu/javabayes"

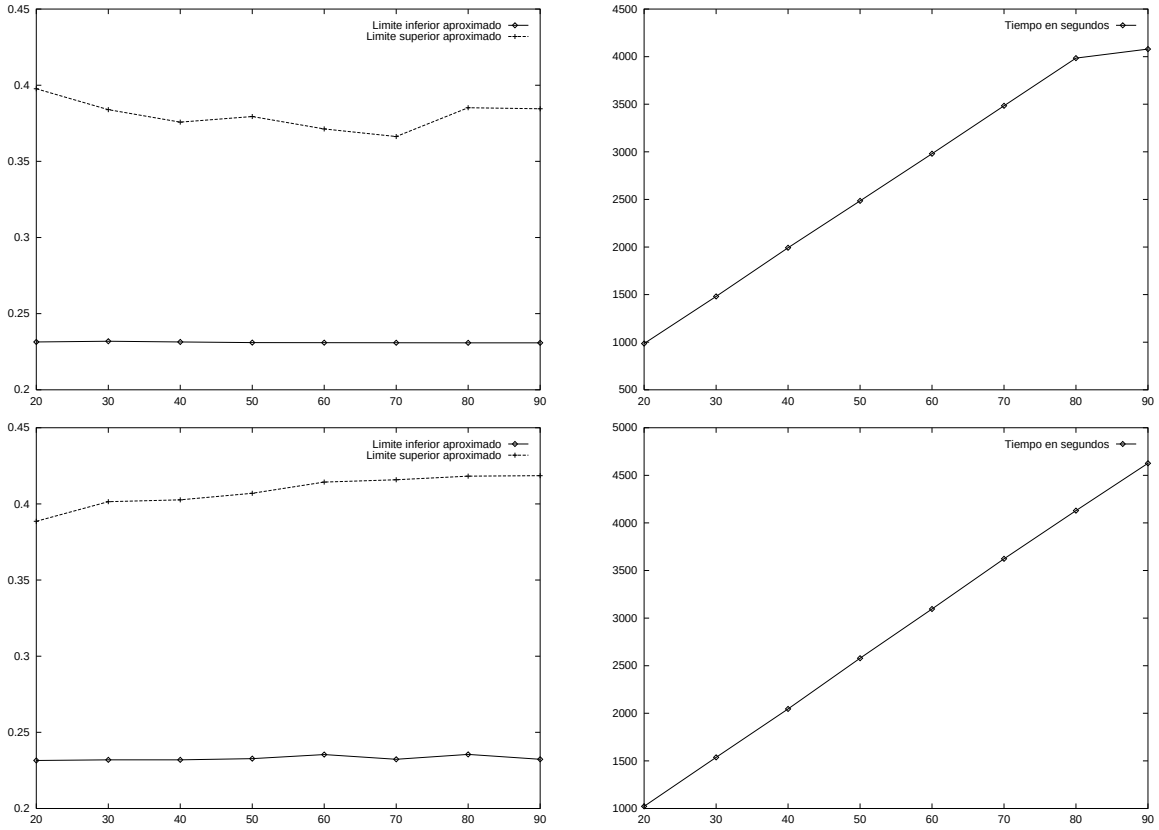


Figura 5.27: Algoritmo PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Bob2 en red $Boblo$: Intervalos (figura a) y tiempos (figura b) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite inferior (tabla 5.11 de arriba) e intervalos (figura c) y tiempos (figura d) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite superior (tabla 5.11 de abajo)

- En la tabla 5.13 mostramos los intervalos que hemos obtenido con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ para diferentes número de iteraciones del algoritmo en la configuración Coc1. También se muestra el tiempo empleado por los algoritmos. La figura fig5.52 muestra gráficamente los resultados.

- En la tabla 5.14 mostramos los intervalos que hemos obtenido con los métodos de propagación por eliminación de variables PropWithTD1 y PropWithTD3 empleando diferentes tamaños máximos para los árboles con la configuración Coc2. También se muestra el tiempo empleado por los algoritmos. La figura 5.31 muestra gráficamente los resultados.

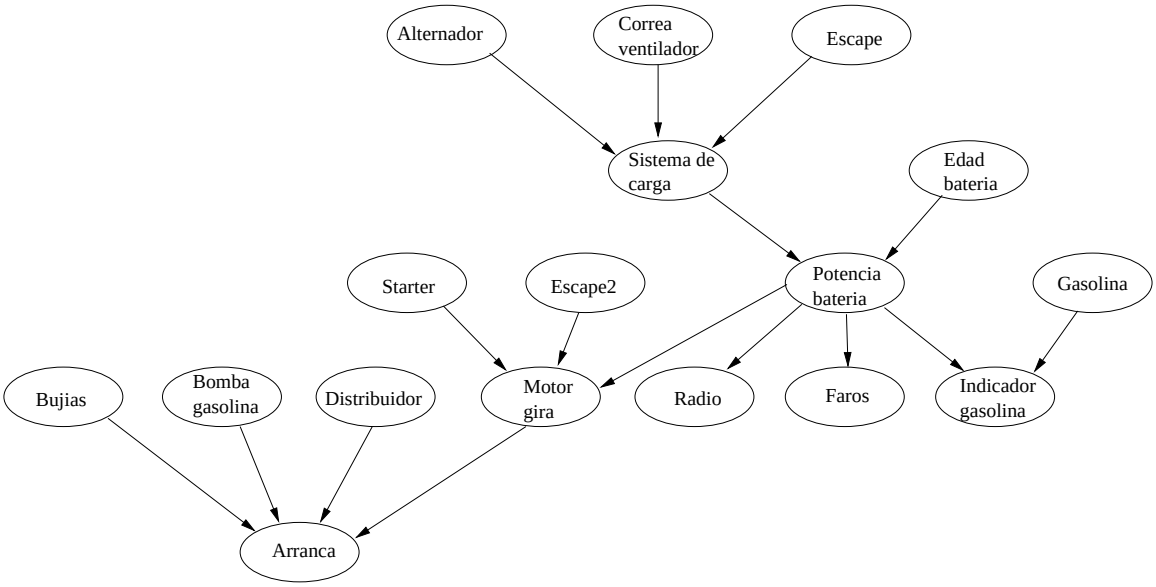


Figura 5.28: Red Coche: Representa las variables relacionadas con el problema del arranque de un coche

- En la tabla 5.15 mostramos los intervalos que hemos obtenido con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ para diferentes número de iteraciones del algoritmo en la configuración Coc2. También se muestra el tiempo empleado por los algoritmos. La figura 5.32 muestra gráficamente los resultados.

	20000	30000	40000	50000	60000
L_i	0.901337	0.901329	0.901327	0.901327	0.901327
L_s	0.957187	0.957197	0.957197	0.957197	0.957197
L_{imin}	0.901322	0.901323	0.901323	0.901323	0.901323
L_{imax}	0.958928	0.958928	0.958928	0.958928	0.958928
ts	764	1301	1494	1799	2048

	20000	30000	40000	50000	60000
L_i	0.496910	0.496910	0.496910	0.496910	0.496910
L_s	0.957758	0.957758	0.957765	0.957765	0.957765
L_{imin}	0.041273	0.041273	0.041273	0.041273	0.041273
L_{imax}	0.958727	0.958726	0.958726	0.958726	0.958726
ts	530	700	1177	1088	1332

Tabla 5.12: Intervalos y tiempos obtenidos con PropWithTD1 (tabla de arriba) y PropWithTD3 (tabla de abajo) en configuración Coc1 en la red $Coche$

	30	40	50	60	70	80	90
L_i	0.904434	0.901352	0.902979	0.901593	0.901330	0.901324	0.901324
L_s	0.953814	0.952766	0.954409	0.957196	0.954730	0.953366	0.956831
ts	3725	4985	6253	7455	8726	10022	11216

	30	40	50	60	70	80	90
L_i	0.905449	0.901710	0.906102	0.912634	0.902740	0.911420	0.917519
L_s	0.955686	0.957020	0.957198	0.957196	0.957198	0.957198	0.957198
ts	5110	6838	8544	10246	11935	13477	13577

Tabla 5.13: Intervalos y tiempos obtenidos con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Coc1 en red $Coche$ enfocando a optimizar el límite inferior (tabla de arriba) y el superior (tabla de abajo)

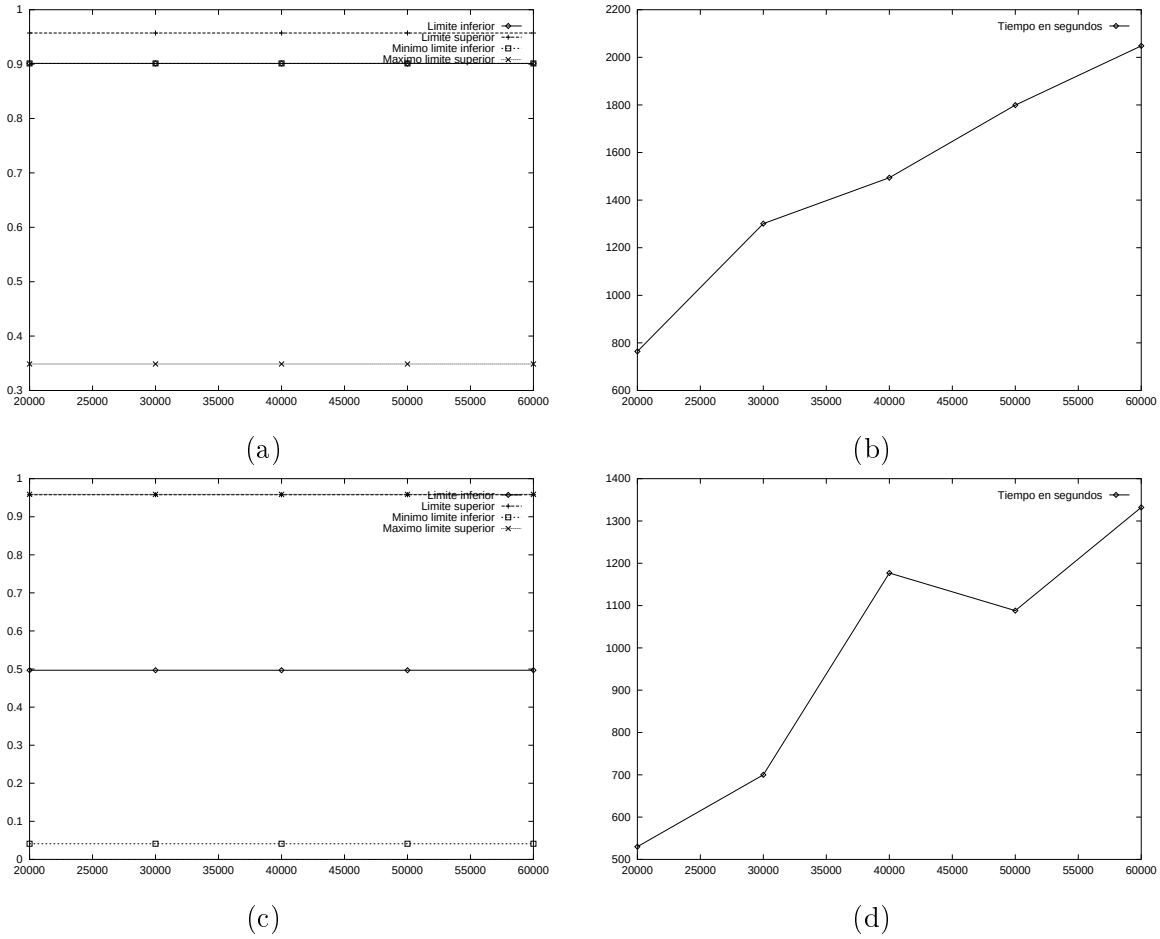


Figura 5.29: Algoritmo PropWithTD1 en configuración Coc1 en red $Coche$: Intervalos (figura a) y tiempos (figura b) para cada tamaño de árbol (tabla 5.12 de arriba). Algoritmo PropWithTD3 en configuración Coc1 en red $Coche$: Intervalos (figura c) y tiempos (figura d) para cada tamaño de árbol (tabla 5.12 de abajo).

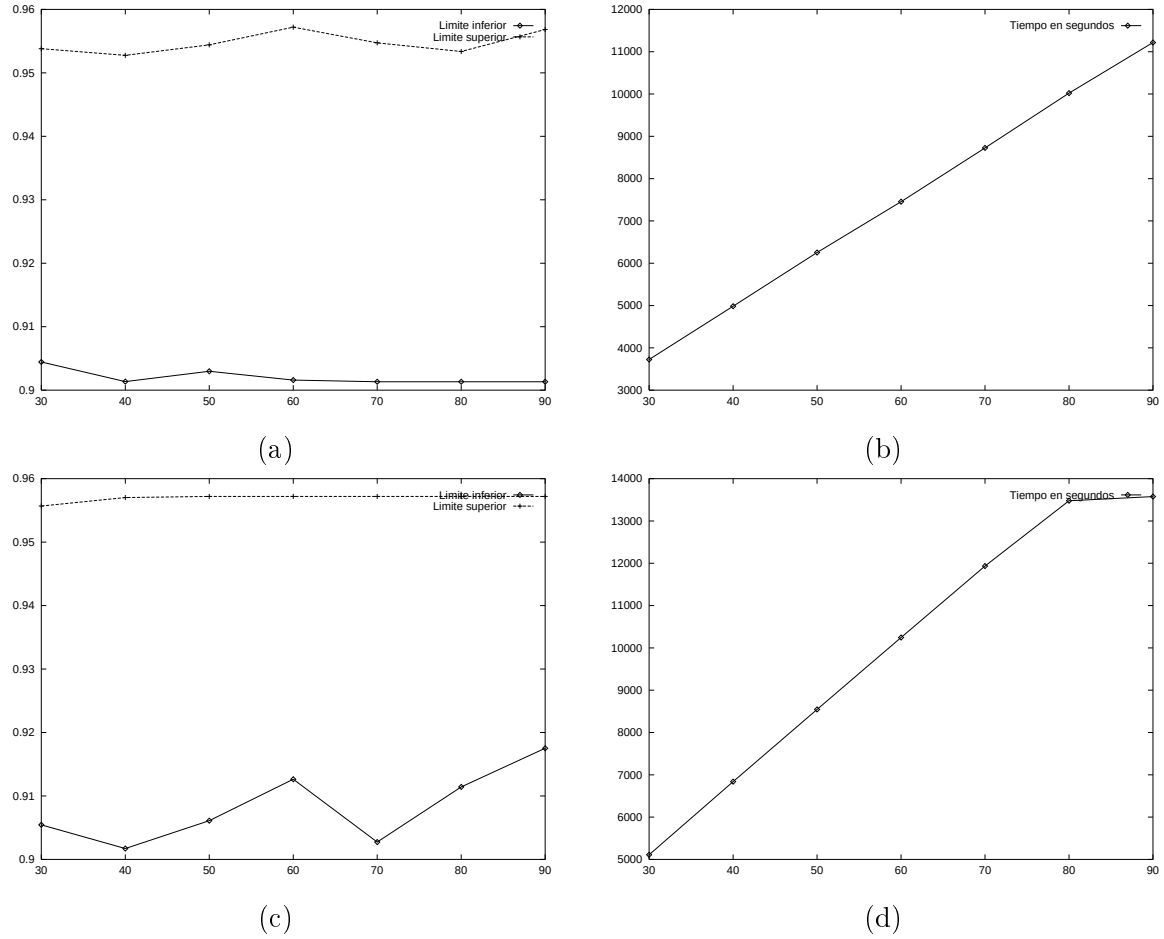


Figura 5.30: Algoritmo PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Coc1 en red $Coche$: Intervalos (figura a) y tiempos (figura b) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite inferior (tabla 5.13 de arriba) e intervalos (figura c) y tiempos (figura d) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite superior (tabla 5.13 de abajo)

	20000	30000	40000	50000	60000
L_i	0.177708	0.177707	0.176318	0.176318	0.176318
L_s	0.343760	0.343762	0.343762	0.343762	0.343762
L_{imin}	0.174571	0.174571	0.174571	0.174571	0.174571
L_{imax}	0.348653	0.348653	0.348653	0.348653	0.348653
ts	1044	1469	1892	2341	2785

	20000	30000	40000	50000	60000
L_i	0.177744	0.176601	0.176586	0.175635	0.176718
L_s	0.345960	0.345960	0.346113	0.348650	0.348656
L_{imin}	0.174024	0.1730291	0.173029	0.173029	0.174024
L_{imax}	0.349661	0.349516	0.349661	0.351259	0.351259
ts	667	787	875	1094	1807

Tabla 5.14: Intervalos y tiempos obtenidos con PropWithTD1 (tabla de arriba) y PropWithTD3 (tabla de abajo) en configuración Coc2 en la red $Coche$

	20	30	40	50	60	70	80	90
L_i	0.179490	0.179214	0.178698	0.179377	0.177229	0.180915	0.176968	0.178903
L_s	0.344572	0.344570	0.344500	0.344615	0.344317	0.344538	0.344569	0.344954
ts	1700	2577	3451	4311	5165	6071	6827	7803

	20	30	40	50	60	70	80	90
L_i	0.177375	0.176985	0.176887	0.177419	0.176810	0.177035	0.176972	0.176746
L_s	0.340261	0.344563	0.342392	0.344001	0.344168	0.340540	0.338345	0.340522
ts	2403	3556	4661	5867	7151	8301	9561	10772

Tabla 5.15: Intervalos y tiempos obtenidos con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Coc2 en red $Coche$ enfocando a optimizar el límite inferior (tabla de arriba) y el superior (tabla de abajo)

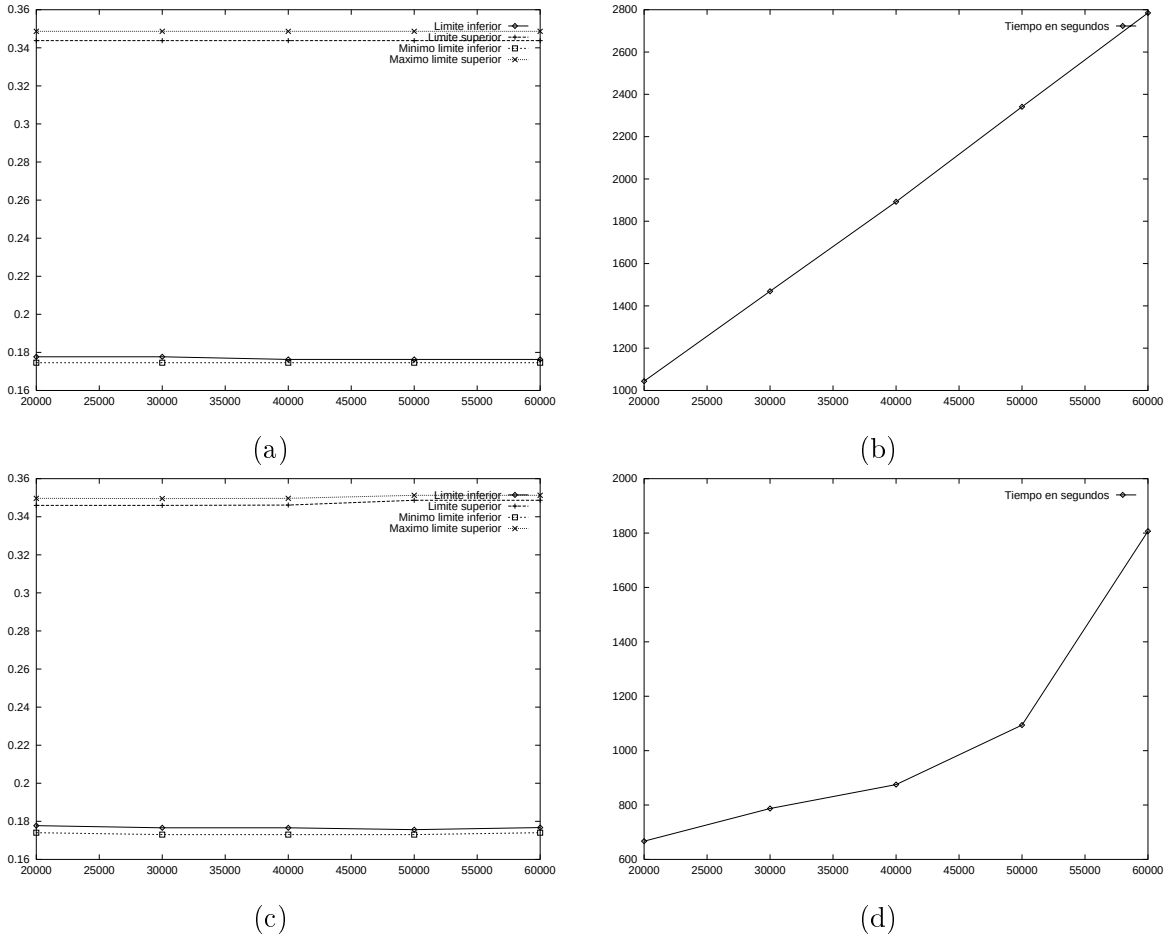


Figura 5.31: Algoritmo PropWithTD1 en configuración Coc2 en red $Coche$: Intervalos (figura a) y tiempos (figura b) para cada tamaño de árbol (tabla 5.14 de arriba). Algoritmo PropWithTD3 en configuración Coc2 en red $Coche$: Intervalos (figura c) y tiempos (figura d) para cada tamaño de árbol (tabla 5.14 de abajo).

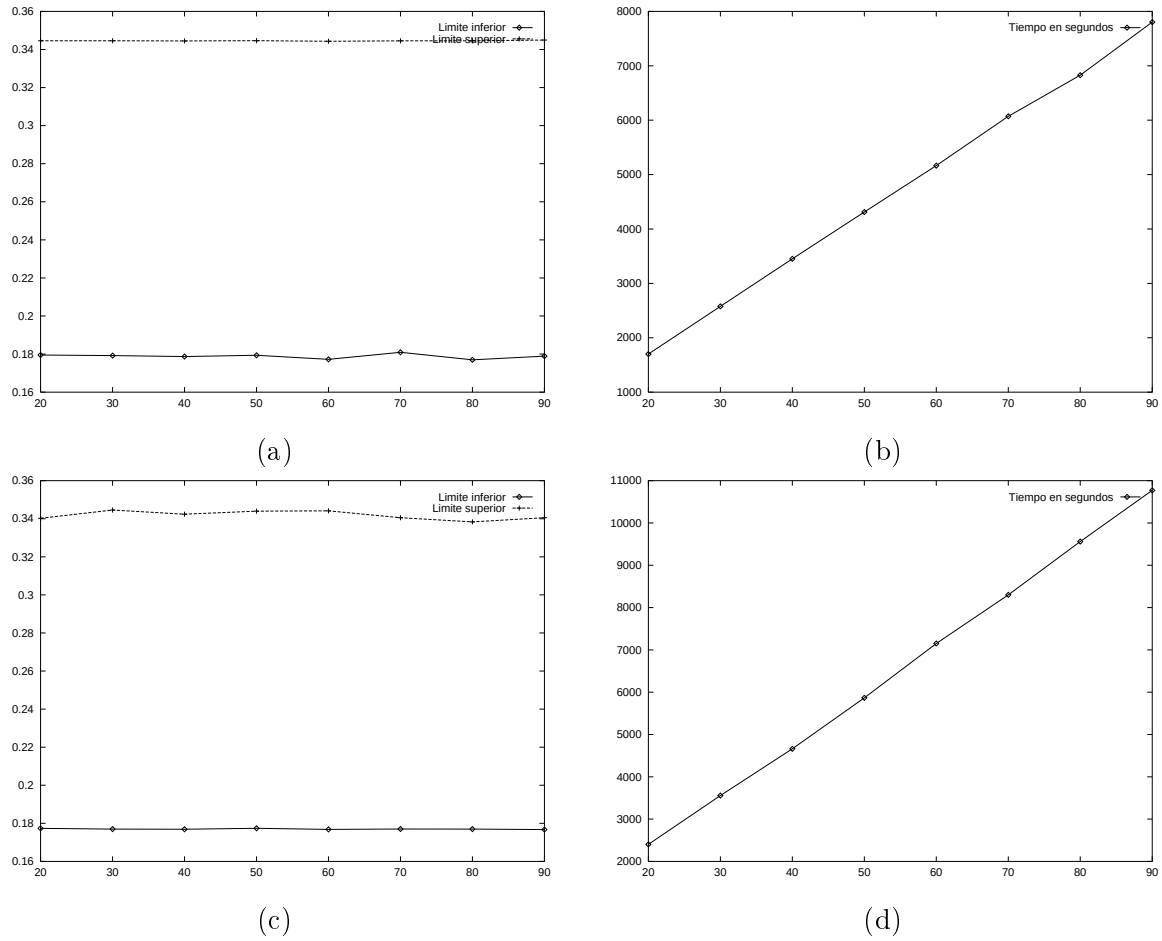


Figura 5.32: Algoritmo PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Coc2 en red $Coche$: Intervalos (figura a) y tiempos (figura b) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite inferior (tabla 5.15 de arriba) e intervalos (figura c) y tiempos (figura d) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite superior (tabla 5.15 de abajo)

5.4.4 Red Alarm

La red *Alarm* [5] se muestra en la figura 5.33. De nuevo hemos sustituido las probabilidades por intervalos de la misma forma que con las redes anteriores. Sobre la red transformada hemos realizado experimentos para obtener los intervalos con el condicionamiento de Dempster [39] en las siguientes variables:

1. Variable "BP" (variable con 2 casos) sin observaciones en ninguna otra variable de la red. Llamaremos a esta configuración Ala1.

2. Variable "BP" (variable con 2 casos) con observaciones en las variables "LVFAILURE" (Caso 2), "PULMEMBOLUS" (Caso 1) y "ARTCO2" (Caso 2). Llamaremos a esta configuración Ala2.

En esta red ha sido imposible aplicar los métodos de propagación por eliminación de variables PropWithTD1 y PropWithTD3, debido a que ambos consumían toda la memoria de nuestro ordenador y era incapaz de obtener ningún resultado.

Veamos los experimentos realizados:

- En la tabla 5.16 mostramos los intervalos que hemos obtenido con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ para diferentes número de iteraciones del algoritmo en la configuración Ala1. También se muestra el tiempo empleado por los algoritmos. La figura fig5.35 muestra gráficamente los resultados.

- En la tabla 5.17 mostramos los intervalos que hemos obtenido con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ para diferentes número de iteraciones del algoritmo en la configuración Ala2. También se muestra el tiempo empleado por los algoritmos. La figura 5.35 muestra gráficamente los resultados.

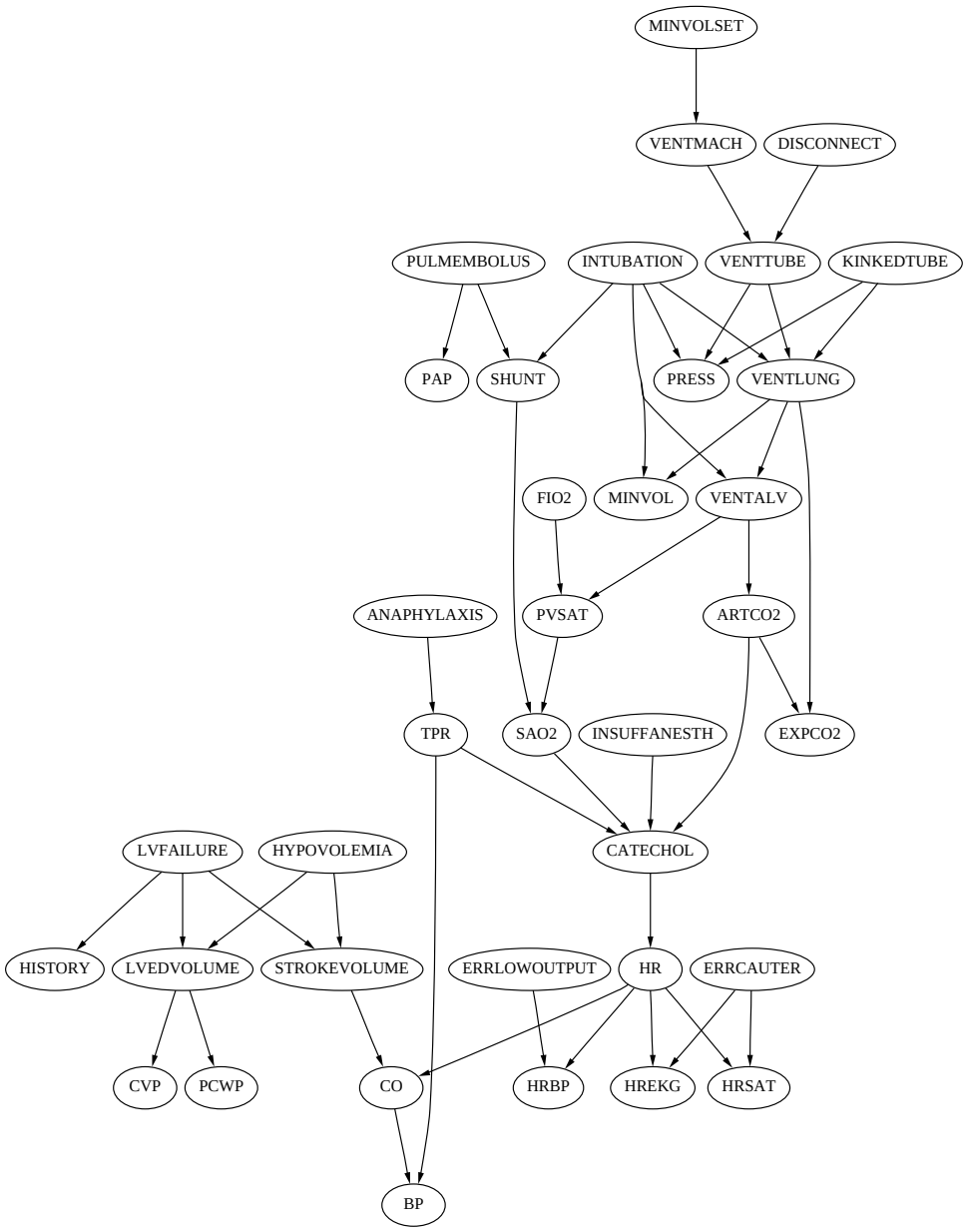


Figura 5.33: Red Alarm (A Logical Alarm Reduction Mechanism): Es una aplicación de diagnóstico usado para explorar técnicas de razonamiento probabilístico en redes Bayesianas

	30	40	50	60	70	80	90
L_i	0.319583	0.315559	0.302474	0.294258	0.293539	0.291204	0.291472
L_s	0.509959	0.488273	0.504944	0.504097	0.500202	0.504162	0.499270
ts	2228	3000	3700	4436	5169	5847	5891

	30	40	50	60	70	80	90
L_i	0.334135	0.342031	0.342866	0.326691	0.312401	0.328256	0.327098
L_s	0.499840	0.508551	0.524216	0.518965	0.517356	0.525373	0.521329
ts	2169	2887	3583	4309	4996	5722	6101

Tabla 5.16: Intervalos y tiempos obtenidos con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Ala1 en red $Alarm$ enfocando a optimizar el límite inferior (tabla de arriba) y el superior (tabla de abajo)

	30	40	50	60	70	80	90
L_i	0.303479	0.305820	0.296274	0.287377	0.284422	0.283146	0.282268
L_s	0.497452	0.470390	0.503964	0.482040	0.468923	0.491886	0.480822
ts	1308	1750	2171	2611	3092	3440	3569

	30	40	50	60	70	80	90
L_i	0.308772	0.321259	0.310396	0.310121	0.321575	0.319159	0.325953
L_s	0.486773	0.498773	0.504750	0.501607	0.506765	0.507039	0.505314
ts	1248	1664	2080	2479	2906	3337	3579

Tabla 5.17: Intervalos y tiempos obtenidos con PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Ala2 en red $Alarm$ enfocando a optimizar el límite inferior (tabla de arriba) y el superior (tabla de abajo)

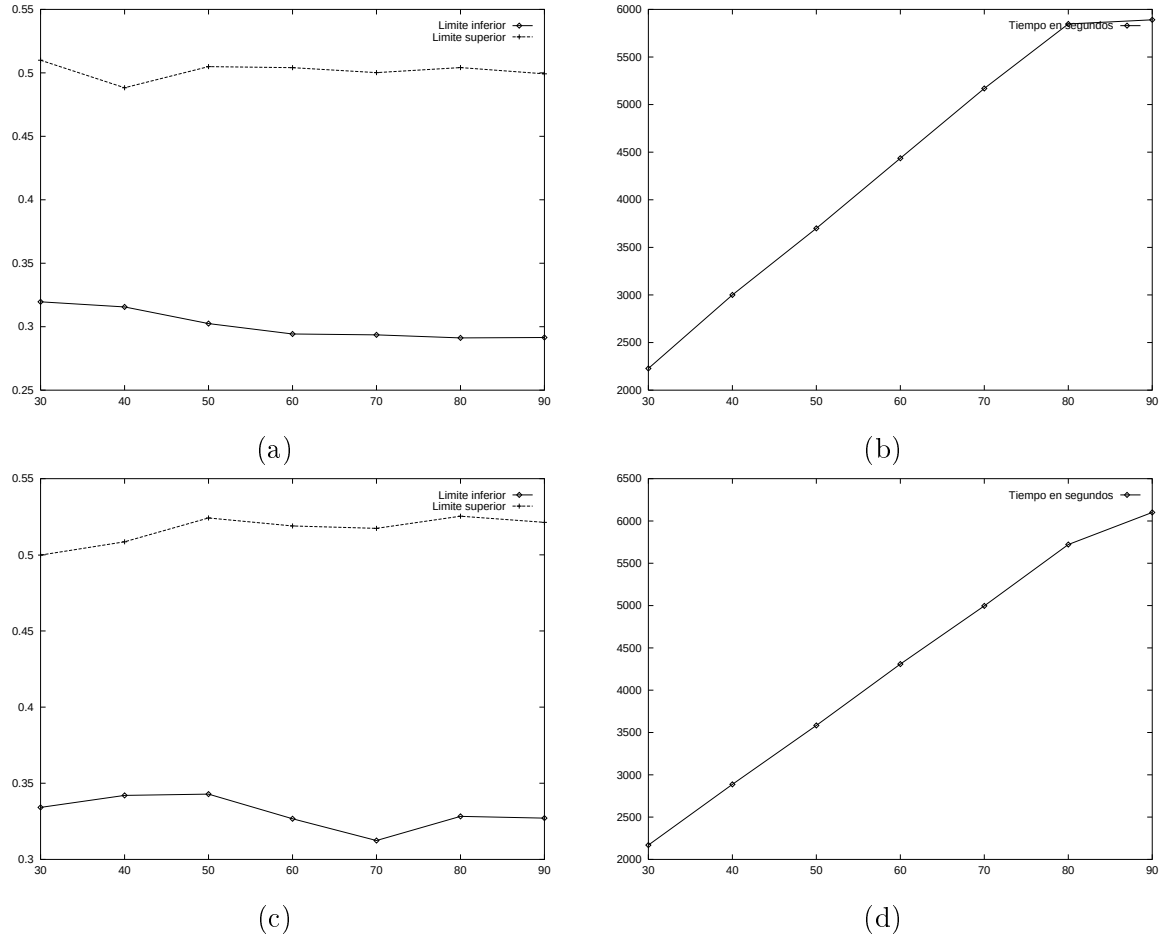


Figura 5.34: Algoritmo PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Ala1 en red $Alarm$: Intervalos (figura a) y tiempos (figura b) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite inferior (tabla 5.16 de arriba) e intervalos (figura c) y tiempos (figura d) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite superior (tabla 5.16 de abajo)

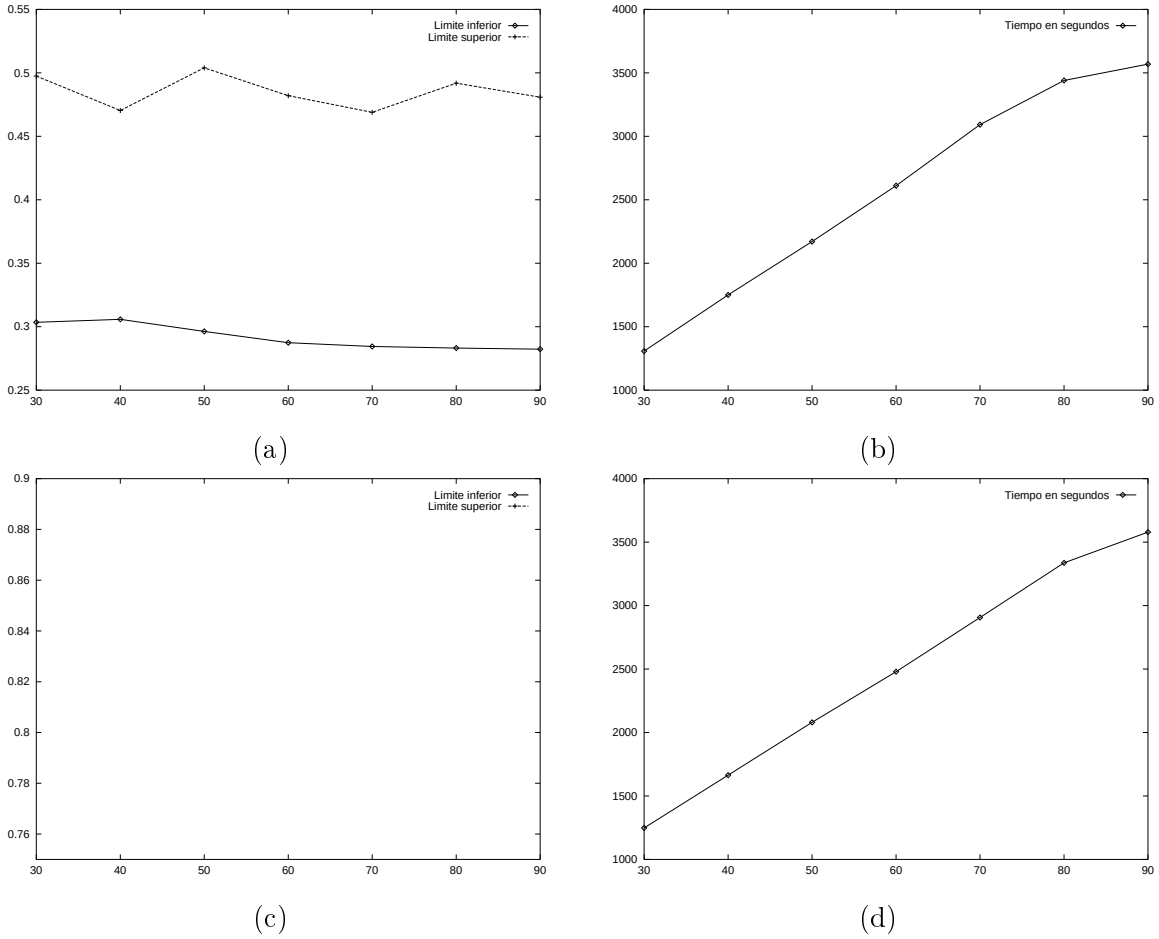


Figura 5.35: Algoritmo PropSim4 con $t_0 = 2.0$ y $\alpha = 0.9$ en configuración Ala2 en red $Alarm$: Intervalos (figura a) y tiempos (figura b) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite inferior (tabla 5.17 de arriba) e intervalos (figura c) y tiempos (figura d) para cada número de iteraciones, enfocando el algoritmo a optimizar el límite superior (tabla 5.17 de abajo)

5.5 Conclusiones

En este capítulo hemos desarrollado una serie de técnicas enfocadas a mejorar algún aspecto de la propagación de intervalos de probabilidad. Estas técnicas se ha demostrado con ejemplos que son importantes incorporarlas en los algoritmos debido a las ventajas de eficiencia en los cálculos que conseguimos con ellas. También hemos realizado una evaluación experimental de los algoritmos implementados tanto exactos como aproximados. Hemos podido ver que los algoritmos exactos funcionan bien, pero que en determinados problemas no son capaces de obtener buenos resultados, e incluso a veces no obtienen ninguno. Limitando el tamaño de los árboles hemos conseguido obtener resultados aproximados con el algoritmo de eliminación de variables. Pero cuando el problema ha sido aún más complejo, tampoco con la limitación del tamaño del árbol se ha resuelto el problema de forma satisfactoria.

En general, destacamos los siguientes puntos de los experimentos:

- Existen casos simples (figuras 5.17,5.21) en los que los algoritmos de eliminación de variables obtienen resultados bastante buenos, obteniéndose intervalos exactos para un determinado tamaño de los árboles. En ese caso, los algoritmos de enfriamiento simulado (figuras 5.18, 5.22) obtienen en pocas iteraciones (50) los intervalos exactos.

- En la mayoría de los casos (figuras 5.19, 5.24, 5.26, 5.32, 5.34, 5.35) los algoritmos de eliminación de variables aproximados no son factibles: el valor medio proporciona intervalos muy pequeños, mientras que los intervalos aproximados con $r_{\min}$ y $r_{\max}$ son demasiado amplios ($[0, 1]$ en muchos casos). Esto viene a corroborar lo que ocurre en los algoritmos de propagación en los que se mantienen sólo dos valores mínimo y máximo para las variables originales del árbol y que son los que se han usado ampliamente en la literatura. Estos corresponden al uso de árboles aproximados donde se eliminan todas las variables transparentes sustituyéndolas por el mínimo y el máximo. Esto es mucho más pobre que nuestro enfoque en el que permitimos árboles de tamaño de hasta 60 000 (es decir 60 000 intervalos para representar el convexo). Si a nosotros nos dan en ejemplos medianos intervalos $[0, 1]$ como aproximación externa del intervalo, no es de extrañar que los algoritmos clásicos produzcan siempre este tipo de aproximaciones tan pobre: el intervalo $[0, 1]$.

- Cuando ha sido posible obtener los intervalos exactos (figuras 5.17,5.21) los algoritmos de enfriamiento simulado (figuras 5.18, 5.22) han convergido rápidamente a estos valores.

En otros casos (figuras 5.20, 5.25, 5.27, 5.30), también se ha dado la convergencia, aunque no podemos estar completamente seguros que los resultados sean exactos (nuestro enfriamiento no garantiza la convergencia).

Conclusiones y Trabajos Futuros

El objetivo global del presente trabajo era la construcción de algoritmos de propagación para grafos de dependencias en los que las distribuciones de probabilidad condicionadas viniesen dadas por medio de intervalos de probabilidad.

Este objetivo general ha sido cubierto mediante la construcción de diversos algoritmos tanto exactos como aproximados. Las dos conclusiones básicas de esta memoria son:

1. Es muy conveniente una especificación global del problema de la propagación de probabilidades imprecisas y el uso de conjuntos convexos para la representación de las informaciones intermedias. Cualquier otra representación menos general que implique el uso de aproximaciones en cada paso del problema, da lugar a intervalos muy poco informativos.

2. La obtención de buenas aproximaciones al problema global es factible para problemas medianos. La aplicación de técnicas de optimización combinatoria produce buenas aproximaciones en tiempos razonables.

Para ello hemos empleado básicamente las siguientes técnicas:

- En el capítulo 1 se ha visto como obtener los conjuntos convexos asociados a distribuciones de probabilidad condicionadas proporcionadas por medio de intervalos. El trabajar con conjuntos convexos, en lugar de directamente con los intervalos, nos asegura que durante los algoritmos de propagación no perderemos información al hacer operaciones con las informaciones disponibles. Se ha demostrado con un ejemplo que tales pérdidas de información son posibles y de hecho bastante frecuentes.

- En el capítulo 2 hemos estudiado dos algoritmos de propagación exactos de conjuntos convexos de probabilidades que hacen uso de las independencias entre las variables del

problema para poder hacer la propagación con cálculos locales en un grafo de dependencias. Hemos clasificado los algoritmos de propagación existentes de acuerdo con las relaciones de independencia usadas. Respecto a los algoritmos exactos implementados para este capítulo concluimos que pueden utilizarse en problemas de muy pocas variables, o en los que sólo tenemos imprecisión sobre algunas de ellas. Por tanto, aunque en teoría estos algoritmos son correctos, el gran número de puntos extremos que llegan a tener los conjuntos convexos hace que la propagación exacta sea infactible en la práctica. Por ello se estima necesaria la construcción de algoritmos aproximados que intentan obtener resultados en los casos en que la propagación exacta no puede realizarse.

- En el capítulo 3 se ha estudiado la especificación del problema de la propagación de conjuntos convexos de probabilidades como un problema de optimización combinatoria mediante el empleo de variables adicionales, que nosotros hemos llamado *transparentes*. En este capítulo se han presentado varios algoritmos que utilizaban la técnica del *enfriamiento simulado* y un *algoritmo genético* que permite obtener resultados cercanos a los exactos, pero en tiempo razonable. En la implementación de estos algoritmos las distribuciones de probabilidad vienen almacenadas en tablas de números reales.

- En el capítulo 4 se presentan los *árboles de probabilidad* para representar las distribuciones de probabilidad. Con esta técnica podemos almacenar distribuciones de probabilidad condicionadas de manera más eficiente que usando tablas. Además, hemos desarrollado algoritmos de propagación de probabilidades que se basan en esta representación realizando las operaciones de combinación, marginalización y selección directamente sobre árboles. También hemos propuesto algoritmos aproximados basados en el uso de árboles de un tamaño máximo limitado. La evaluación experimental muestra que el uso de árboles de probabilidad mejora la eficiencia de los algoritmos exactos y obtiene buenas aproximaciones en casos en que los algoritmos exactos sean infactibles.

- En el capítulo 5 se han combinado las ideas desarrolladas en los capítulos 3 y 4 para llevar a cabo la propagación de intervalos de probabilidad. Las conclusiones que extraemos de este capítulo se resumen a continuación:

  - Los árboles de probabilidad permiten representar los convexos asociados a las probabilidades condicionadas intervalares de forma eficiente evitando los problemas de explosión combinatoria que se producen con las tablas de probabilidad.

  - Los métodos de propagación del capítulo 2 se pueden llevar a cabo usando árboles de probabilidad. Esta representación permite obtener los puntos extremos del convexo 'a posteriori' de forma eficiente.
  - Cuando la propagación se hace con árboles de tamaño limitado se puede calcular un intervalo que incluya al intervalo exacto. Esto puede usarse para obtener un límite superior para el error cometido en la aproximación.

- Como conclusión final, hemos desarrollado métodos de propagación de conjuntos convexos de probabilidad mediante técnicas de enfriamiento simulado con árboles de probabilidad. Experimentalmente se ha visto que estos algoritmos obtienen buenos resultados, mientras que los métodos exactos son incapaces de obtener ningún resultado.

Líneas futuras de investigación

Al trabajar en la elaboración de esta memoria hemos ido descubriendo nuevas posibilidades para continuar nuestra investigación. Así, creemos que en un futuro próximo debemos abordar las siguientes líneas de trabajo:

- Mejorar los algoritmos de propagación de conjuntos convexos en los árboles de grupos tanto exactos como los de enfriamiento simulado. Para ello podemos aplicar las siguientes técnicas que estamos convencidos que pueden mejorar los resultados:
  - Adición de grupos intermedios al árbol de grupos, lo cual puede evitar tener que combinar varios potenciales que irían a un mismo grupo, creando así árboles casi completos a partir de árboles que podrían tener pocas ramas.
  - Empleo de la *propagación perezosa* [93]. Consiste en que en la propagación, los potenciales nunca se combinarían hasta que no sea estrictamente necesario, o sea, cuando tengamos que eliminar una variable. De esta forma los potenciales serían listas de árboles de probabilidad.
  - Adaptación de la operación de combinación con los algoritmos de simulación. En estos algoritmos cuando simulamos un valor para las variables transparente, éste se va propagando al resto del árbol de grupos con el envío de mensajes entre los grupos adyacentes. El cálculo de los mensajes necesita de las operaciones de combinación de árboles y finalmente de la marginalización. Pero si una variable transparente ha sido observada mediante el proceso de la simulación entonces no será necesario

combinar aquellas ramas de los árboles originales que no correspondan al valor simulado.

- Mejorar la implementación de los algoritmos genéticos de forma que no sea necesaria la propagación de individuos ya obtenidos en generaciones anteriores.
- Aplicación de otras técnicas de optimización combinatoria como puede ser la *búsqueda tabú* [59] o las *búsquedas con entorno variable* [65].
- Adaptar los algoritmos a otras formas de calcular un convexo de probabilidad condicional como las propuestas por Walley y Moral.
- Aplicar los algoritmos de propagación de intervalos construidos a problemas reales, y comprobar con expertos en el tema el buen funcionamiento del sistema.
- Usar medidas de información alternativas para que los algoritmos de propagación de intervalos con árboles de tamaño limitado optimicen los intervalos que acotan el error.
- Integrar todos los algoritmos construidos en el programa *Elvira*[2].

[2]Es un programa que está siendo implementado en Java por miembros de varias universidades españolas integradas en el proyecto "Entorno para el desarrollo de modelos gráficos probabilísticos (TIC97-1135-C04-01)" (financiado por la CICYT)

Bibliografía

[1] Aarts, E. y Korst, J., *Simulated Annealing and Boltzmann Machines*, Wiley & Sons, New York, 1989.

[2] Abramson, B. y A.J.Finizza, "Using belief networks to forecast oil prices", *International Journal of Forecasting*, tomo 7, págs. 299–316, 1991.

[3] Amarger, S., Dubois, D. y Prade, H., "Constraint Propagation with Imprecise Conditional Probabilities", en *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, B. D'Ambrosio, P. Smets y P. Bonissone, eds., págs. 26–34, Morgan & Kaufmann, 1991.

[4] Arnborg, S., Corneil, D. y Proskurowski, A., "Complexity of finding embeddings in a k-tree", *SIAM Jour. Alg. Discr. Meth.*, tomo 8, págs. 277–284, 1987.

[5] Beinlich, I., Suermondt, G., Chavez, R. y Cooper, G., "The ALARM monitoring system", en *Second European Conference on AI and Medicine*, Springer-Verlag, Berlin, 1989.

[6] Berger, J., "An Overview of Robust Bayesian Analysis (with discussion)", *Test*, tomo 3, págs. 5–124, 1994.

[7] Berler, A. y Shimony, S., "Bayes Nets for Sonar Sensor Fusion", en *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, D. Geiger y P. Shenoy, eds., págs. 14–21, Morgan & Kaufmann, 1997.

[8] Berzuini, C., Bellazzi, R., Quaglini, S. y Spiegelhalter, D., "Bayesian networks for patient monitoring", *Artificial Intelligence in Medicine*, tomo 4, págs. 243–260, 1992.

[9] Boerlage, B., *Link Strength in Bayesian Networks*, Tesis Doctoral, Department of Computer Science, University of British Columbia., Canada., 1992.

[10] Boutilier, C., Friedman, N., Goldszmidt, M. y Koller, D., "Context-Specific Independence in Bayesian Networks", en *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)*, págs. 115–123, Portland, Oregon, 1996.

[11] Breese, J. y Fertig, K., "Decision Making with Interval Influence Diagrams", en *Uncertainty in Artificial Intelligence, 6*, L. K. P.P. Bonissone, M. Henrion, ed., págs. 467–478, Elsevier, 1991.

[12] Cano, A., Cano, J. E. y Moral, S., "Simulation Algorithms for Convex Sets of Probabilities", Technical Report TR-93-2-xx, DECSAI, Universidad de Granada, 1993.

[13] Cano, A., Cano, J. E. y Moral, S., "Convex Sets of Probabilities Propagation by Simulated Annealing", en *Proceedings of the Fith International Conference IPMU'94*, págs. 4–8, Paris, 1994.

[14] Cano, A. y Moral, S., "Heuristic Algorithms for the Triangulation of Graphs", en *Advances in Intelligent Computing*, B. Bouchon-Meunier, R. Yager y L. Zadeh, eds., págs. 98–107, Springer Verlag, 1995.

[15] Cano, A. y Moral, S., "A Genetic Algorithm to Approximate Convex Sets of Probabilities", en *Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference (IPMU' 96) Vol. 2*, págs. 859–864, 1996.

[16] Cano, A. y Moral, S., "Propagación Exacta y Aproximada mediante Arboles de Probabilidad en Redes Causales", en *Actas de la VII Conferencia de la Asociación Espanola para la Inteligencia Artificial*, págs. 635–644, Málaga, 1997.

[17] Cano, A. y Moral, S., "A Review of Propagation Algorithms for Imprecise Probabilities", en *Proceedings of the First International Symposium on Imprecise Probabilities and their Applications (ISIPTA'99)*, Ghent, 1999.

[18] Cano, J. E., *Propagación de Probabilidades Inferiores y Superiores en Grafos*, Tesis Doctoral, Universidad de Granada, 1992.

[19] Cano, J. E., Delgado, M. y Moral, S., "Propagation of Uncertainty in Dependence Graphs", en *Symbolic and Quantitative Approaches to Uncertainty*, R. Kruse y P. Siegel, eds., págs. 42–47, Springer Verlag, 1991.

[20] Cano, J. E., Delgado, M. y Moral, S., "An Axiomatic System for the Propagation of Uncertainty in Directed Acyclic Networks", *International Journal of Approximate Reasoning*, tomo 8, págs. 253–280, 1993.

[21] Cano, J. E. y Moral, S., "Improoving the efficiency of upper and lower probabilities propagation", Inf. Téc. 91-2-3, DECSAI, Universidad de Granada, 1991.

[22] Cano, J. E., Moral, S. y Verdegay-López, J. F., "Combination of Upper and Lower Probabilities", en *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, B. D'Ambrosio, P. Smets y P. Bonissone, eds., págs. 61–68, Morgan & Kaufmann, 1991.

[23] Cano, J. E., Moral, S. y Verdegay-López, J. F., "Partial Inconsistency of Probability Envelopes", *Fuzzy Sets and Systems*, tomo 52, págs. 201–216, 1992.

[24] Cano, J. E., Moral, S. y Verdegay-López, J. F., "Propagation of Convex sets of Probabilities in Directed Acyclic Networks", en *Uncertainty in Intelligent Systems*, B. Bouchon-Meunier *et al.*, eds., págs. 15–26, Elsevier, 1993.

[25] Castillo, E., Gutiérrez, J. y Hadi, A., *Expert Systems and Probabilistic Network Models*, Springer Verlag, New-York, 1997.

[26] Castillo, E., Gutiérrez, J. y Hadi, A., *Sistemas Expertos y Modelos de Redes Probabilísticas*, Monografías de la Academia de Ingeniería, Madrid, 1997.

[27] Castillo, E., Gutiérrez, J. M. y Hadi, A. S., "Parametric Structure of Probabilities in Bayesian Networks", en *Lectures Notes in Artificial Intelligence, 946*, págs. 89–98, Springer-Verlag, 1995.

[28] Castillo, E., Gutiérrez, J. M. y Hadi, A. S., "Simbolic Propagation in Discrete and Continuous Bayesian Networks", en *Mathematics with Vision: First International Mathematica Symposium*, V. Keranen y P. M. P. Mitic, eds., págs. 77–84, Southampton, U.K., 1995.

[29] Chateauneuf, A. y Jaffray, J.-Y., "Some Characterizations of Lower Probabilities and other Monotone Capacities Through the Use of Möbius Inversion", *Math. Soc. Sci.*, tomo 17, págs. 263–283, 1989.

[30] Choquet, G., "Theorie of capacities", *Ann. Inst. Fourier*, tomo 5, págs. 131–292, (1953/54).

[31] Couso, I., Moral, S. y Walley, P., "Examples of Independence for Imprecise Probabilities", en *Proceedings of the First International Symposium on Imprecise Probabilities and Their Applications (ISIPTA'99)*, 1999.

[32] Cozman, F., "Robust Analysis of Bayesian Networks with Finitely Generated Convex Sets of Distributions", Technical Report CMU-RI-TR96-41, Carnegie Mellon University, 1996.

[33] Cozman, F., "Robustness Analysis of Bayesian Networks with Local Convex Sets of Distributions", en *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, Morgan & Kaufmann, San Mateo, 1997.

[34] de Campos, L. M. y Huete, J. F., "Independence Concepts in Upper and Lower Probabilties", en *Uncertainty in Intelligent Systems*, B. Bouchon-Meunier *et al.*, eds., págs. 85–96, Elsevier, 1993.

[35] de Campos, L. M., Huete, J. F. y Moral, S., "Probability Intervals: a Tool for Uncertain Reasoning", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, tomo 2, págs. 167–196, 1994.

[36] de Campos, L. M., Lamata, M. T. y Moral, S., "The Concept of Conditional Fuzyy Measure", *International Journal of Intelligent Systems*, tomo 5, págs. 237–246, 1990.

[37] de Campos, L. M. y Moral, S., "Independence Concepts for Convex Sets of Probabilities", en *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, P. Besnard y S. Hanks, eds., págs. 108–115, Morgan & Kaufmann, 1995.

[38] Dechter, R., "Bucket Elimination: A Unifying Framework for Probabilistic Inference", en *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)*, págs. 211–219, Portland, Oregon, 1996.

[39] Dempster, A. P., "Upper and Lower Probabilities Induced by a Multivalued Mapping", *Annals of Mathematical Statistics*, tomo 38, págs. 325–339, 1967.

[40] DeRobertis, L. y Hartigan, J., "Bayesian Inference Using Intervals of Measures", *Annals of Statistics*, tomo 14, págs. 461–468, 1986.

[41] Dubois, D., Godo, L., de Mántaras, R. L. y Prade, H., "Qualitative reasoning with imprecise probabilities.", *Journal of Intelligent Information Systems*, págs. 319–363, 1993.

[42] Dubois, D. y Prade, H., *Possibility Theory*, Plenum Press, New York, 1988.

[43] Dubois, D. y Prade, H., "A Survey of Belief Revision and Updating Rules in Various Uncertainty Models", *International Journal of Intelligent Systems*, tomo 9, págs. 61–100, 1994.

[44] Edelsbrunner, H., *Algorithms in Combinatorial Geometry*, Springer Verlag, Berlin, 1987.

[45] Fagin, R. y Halpern, J., "A New Approach to Updating Beliefs", en *Uncertainty in Artificial Intelligence, 6*, P. P. Bonissone, M. Henrion, L.Ñ. Kanal y J. F. Lemmer, eds., págs. 347–374, North-Holland, Amsterdam, 1991.

[46] Fagiuoli, E. y Zaffalon, M., "2U: an Exact Interval Propagation Algorithm for Polytrees with Binary Variables", *Artificial Intelligence*, tomo 106, págs. 77–107, 1998.

[47] Fertig, K. W. y Breese, J. S., "Interval Influence Diagrams", en *Uncertainty in Artificial Intelligence, 5*, M. Henrion, R. D. Shacter, L.Ñ. Kanal y J. F. Lemmer, eds., págs. 149–161, North-Holland, Amsterdam, 1990.

[48] Fertig, K. W. y Breese, J. S., "Probability Intervals over Influence Diagrams", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, tomo 15, págs. 280–286, 1993.

[49] Fine, T., *Theories of Probability*, Academic Press, New York, 1973.

[50] Fine, T., "An Argument for Comparative Probability", en *Basic Problems in Probability and Linguistics*, R. Butts y J. Hintikka, eds., págs. 105–119, D. Reidel, Dordrecht, 1977.

[51] Friedman, N., "Bayesian Network Repository", Disponible en la dirección "http://www-nt.cs.berkeley.edu/home/nir/public_html/Repository/index.htm", 1997.

[52] Friedman, N. y Goldszmidt, M., "Learning Bayesian Networks with Local Structure", en *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)*, págs. 252–262, Portland, Oregon, 1996.

[53] Friedman, N. y Russell, S., "Nonuniform Dynamic Discretization in Hybrid Networks", en *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, D. Geiger y P. Shenoy, eds., págs. 175–181, Morgan & Kaufmann, 1997.

[54] Geiger, D. y Pearl, J., "On the logic of Causal Models", en *Uncertainty in Artificial Intelligence 4*, R. Shachter, T. Levitt, L. Kanal y J. Lemmer, eds., págs. 136–147, North-Holland, 1988.

[55] Geiger, D., Verma, T., y Pearl, J., "D-separation: From Theorems to Algorithms.", en *Uncertainty in Artificial Intelligence 5*, M. Henrion, R. Shacter, L. Kanal y . Lemmer, J.F., eds., págs. 139–148, North Holland, Amsterdam, 1990.

[56] Geiger, D., Verma, T. y Pearl, J., "Identifying Independence in Bayesian Networks", *Networks*, tomo 20, págs. 507–534, 1990.

[57] Geman, S. y Geman, D., "Stochastic relaxation, Gibbs desitributions, and the Bayesian Restoration of Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, tomo 6, págs. 721–741, 1984.

[58] Gidas, B., "Nonstationary Markov chains and convergence of the annealing algorithm", *Journal of Statistical Physics*, tomo 39, págs. 73–131, 1985.

[59] Glover, F. y Laguna, M., *Tabu Search*, Kluwer AcademicKluwer Academic, Boston/Dordrecht/London, 1997.

[60] Grabisch, M., Nguyen, H. y Walker, E., *Fundamentals of Uncertainty Calculi with Applications to Fuzzy Inference*, Kluwer Academic Publishers, Dordrecht, 1995.

[61] Greene, J. y Supowit, K., "Simulated Annealing Without Rejected Moves", en *Proc. IEEE Int. Conference on Computer Design*, págs. 658–663, Port Chester, 1984.

[62] Grosof, B.Ñ., "An Inequality Paradigm for Probabilistic Knowledge", en *Uncertainty in Artificial Intelligence*, L.Ñ. Kanal y L. F. Lemmer, eds., págs. 259–275, North-Holland, Amsterdam, 1986.

[63] Hajek, B., "Cooling schedules for optimal annealing", *Mathematics of Operations Research*, tomo 13, págs. 311–329, 1988.

[64] Hansen, P. y Jaumard, B., "Probabilistic Satisfiability", Por aparecer en: J. Kohlas y S. Moral, eds., *Algorithms for Uncertain and Defeasible Reasoning*.

[65] Hansen, P. y Mladenović, N., "Variable Neighborhood Search: Principles and Applications", Inf. Téc. G-98-20, GERARD and École des Hautes Études Commerciales, 3000 chemin de la Côte-Sainte-Catherine. Montréal, Canada H3T 2A7, October 1998.

[66] Heckerman, D., A.Mamdani y Wellman, M., "Special issue on Real-World Applications of Bayesian Networks", *Communications of the ACM*, tomo 38, 1995.

[67] Heckerman, D. y Horvitz, E., "Inferring Informational Goals from Free-Text Queries: A Bayesian Approach", en *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–98)*, págs. 230–237, Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[68] Hernández, L. D., *Diseño y Validación de Nuevos Algoritmos para el Tratamiento de Grafos de Dependencias*, Tesis Doctoral, Universidad de Granada, 1995.

[69] Horvitz, E., Breese, J., Heckerman, D., Hovel, D. y Rommelse, K., "The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users", en *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–98)*, págs. 256–265, Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[70] Hrycej, T., "Gibbs Sampling in Bayesian Networks", *Artificial Intelligence*, tomo 46, págs. 351–363, 1990.

[71] Huber, P. J., *Robust Statistics*, Wiley, New York, 1981.

[72] Imbert, J. L. y Hentenryck, P. V., "Redundancy Elimination with a Lexicographic Solved Form", *Annals of Mathematics and Artificial Intelligence*, tomo 17, págs. 85–106, 1996.

[73] Jensen, A. L. y Jensen, F. V., "MIDAS – An Influence Diagram for Management of Mildew in Winter Wheat", en *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI–96)*, págs. 349–356, Morgan & Kaufmann, Portland, Oregon, 1996.

[74] Jensen, F., Andersen, S., Kjærulff, U. y Andreassen, S., "MUNIN. On the Case for Probabilities in Medical Expert Systems. A Practical Exercise", en *Lecture Notes in Medical Informatics, 33*, F. et al., ed., Springer Verlag, 1987.

[75] Jensen, F. V., *An Introduction to Bayesian Networks*, University College London Press, London, 1996.

[76] Jensen, F. V., Lauritzen, S. L. y Olesen, K. G., "Bayesian Updating in Causal Probabilistic Networks by Local Computation", *Computational Statistics Quarterly*, tomo 4, págs. 269–282, 1990.

[77] Jensen, F. V., Olesen, K. G. y Andersen, S. K., "An algebra of Bayesian belief universes for knowledge based systems", *Networks*, tomo 20, págs. 637–659, 1990.

[78] Karwan, H. H., Lofti, V., Telgen, J. y Zionts, S., *Redundancy in Mathematical Programming: a State-of-the-Art Survey*, Lecture Notes in Economics and Mathematical Systems N. 206, Springer Verlag, 1983.

[79] Kim, J. y Pearl, J., "A Computational Model for Causal and Diagnostic Reasoning in Inference Systems", en *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI-83)*, págs. 190–203, Karlsruje, 1983.

[80] Kirpatrick, S., Gelatt, C. y Vecchi, M., "Optimization by Simulated Annealing", *Science*, tomo 220, págs. 671–680, 1983.

[81] Kjærulff, U., "Triangulation of graphs - algorithms giving small total spaces.", Inf. Téc. 90-09, Department of Mathematics and Computer Sciences., Institute of Electronic Systems. Aalborg University., March 1990.

[82] Kjærulff, U., "Optimal Decomposition of Probabilistic Networks by Simulated Annealing", *Statistic and Computing*, tomo 2, págs. 7–17, 1992.

[83] Klir, G. y Folger, T., *Fuzzy Sets, Uncertainty, and Information*, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[84] Kong, A., *Multivariate belief functions and graphical models*, Tesis Doctoral, Department of Statistics, Harvard University, Cambridge, MA., 1986.

[85] Kozlov, D. y Koller, D., "Nonuniform Dynamic Discretization in Hybrid Networks", en *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, D. Geiger y P. Shenoy, eds., págs. 302–313, Morgan & Kaufmann, 1997.

[86] Kullback, S. y Leibler, R. A., "On information and sufficiency", *Annals of Mathematical Statistics*, tomo 22, págs. 76–86, 1951.

[87] Kyburg, H. E. y Pittarelli, M., "Some Problems for Convex Bayesians", en *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence*, D. Dubois *et al.*, eds., págs. 149–154, Morgan & Kaufmann, San Mateo, 1992.

[88] Lassez, C. y Lassez, J. L., "Quantifier Elimination for Conjunctions of Linear Constraints via a Convex Hull Algorithm", en *Symbolic and Numerical Computation for Artificial Intelligence*, B. Donald *et al.*, eds., págs. 103–119, Academic Press, London, 1992.

[89] Lauritzen, S. L. y Shenoy, P. P., "Computing Marginals Using Local Computation", Por aparecer en: J. Kohlas y S. Moral, eds., *Algorithms for Uncertain and Defeasible Reasoning*.

[90] Lauritzen, S. L. y Spiegelhalter, D. J., "Local Computation with Probabilities on Graphical Structures and their Application to Expert Systems", *Journal of the Royal Statistical Society, Ser. B*, tomo 50, págs. 157–224, 1988.

[91] Li, Z. y D'Ambrosio, B., "Efficient Inference in Bayes Networks as a Combinatorial Optimization Problem", *International Journal of Approximate Reasoning*, tomo 11, págs. 55–81, 1994.

[92] Lukasiewicz, T., *Precision of Probabilistic Deduction under Taxonomic Knowledge*, Tesis Doctoral, Universität Tüningen, 1996.

[93] Madsen, A. L. y Jensen, F. V., "Lazy Propagation in Junction Trees", en *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI–98)*, págs. 362–369, Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[94] Matheiss, T. H. y Rubin, D. S., "A Survey and Comparison of Methods for Finding all Vertices of Convex Polyedral Sets", *Mathematics of Operational Research*, tomo 5, págs. 167–185, 1980.

[95] Metropolis, N., Rosenbluth, M., Teller, A. y Teller, E., "Calculations by Fast Computing Machines", *Journal of Chemical Physics*, tomo 21, págs. 1087–1092, 1953.

[96] Michalewicz, Z., *Artificial Intelligence. Genetic Algorithms + Data Structures=Evolution Programs*, Springer Verlag, 1992.

[97] Moral, S., "Calculating Uncertainty Intervals from Conditional Convex Sets of Probabilities", en *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence*, D. Dubois *et al.*, eds., págs. 199–206, Morgan & Kaufmann, San Mateo, 1992.

[98] Moral, S. y de Campos, L. M., "Updating Uncertain Information", en *Uncertainty in Knowledge Bases*, B. Bouchon-Meunier *et al.*, eds., págs. 58–67, Springer Verlag, 1991.

[99] Moral, S. y del Sagrado, J., "Aggregation of Imprecise Probabilities", en *Aggregation and Fusion of Imperfect Information*, B. Bouchon-Meunier, ed., págs. 162–168, Physica-Verlag, Heidelberg, 1997.

[100] Neapolitan, R., *Probabilistic Reasoning in Expert Systems*, Wiley-Interscience, New York, 1990.

[101] Nilsson, N. J., "Probabilistic Logic", *Artificial Intelligence*, tomo 28, págs. 71–87, 1986.

[102] Olesen, K. G., Kjærulff, U., Jensen, F., Jensen, F. V., Falck, B., Andreassen, S. y Andersen, S. K., "A MUNIN Network for the Median Nerve - A Case Study in Loops", *Applied Artificial Intelligence*, tomo 3, págs. 385–404, 1989.

[103] Olmsted, S. M., *On representing and solving decision problems*, Tesis Doctoral, Department of Engineering-Economic Systems, Stanford University, Stanford, CA., 1983.

[104] P. Larrañaga, C.M. Knijpers, M. P. y Murga, R., "Decomposing Bayesian networks: triangulation of the moral graph with genetic algorithms", *Statitistics and Computing*, tomo 7, págs. 19–34, 1997.

[105] Paass, G., "Probabilistic Logic", en *Non-Standard Logics for Automated Reasoning*, P. Smets, A. Mamdani, D. Dubois y H. Prade, eds., págs. 213–251, Academic Press, London, 1988.

[106] Pearl, J., *Probabilistic Reasoning with Intelligent Systems*, Morgan & Kaufman, San Mateo, 1988.

[107] Poole, D., "Probabilistic Partial Evaluation: Exploiting Rule Structure in Probabilistic Inference", en *To appears in: Proceedings of the 15th IJCAI Conference (IJCAI' 97)*, págs. 1284–1291, Nagoya, Japan, 1997.

[108] Preparata, F. P. y Shamos, M. I., *Computational Geometry. An Introduction*, Springer Verlag, New York, 1985.

[109] Quinlan, J. R., "Inferno: a Cautious Approach to Uncertain Inference", *Computation Journal*, tomo 26, págs. 255–269, 1983.

[110] Quinlan, J. R., "Induction of Decision Trees", *Machine Learning*, tomo 1, págs. 81–105, 1986.

[111] Rasmussen., L. K., "Bayesian network for blood typing and parentage verification of cattle.", Inf. Téc. Dina research report no. 38, Department of Mathematics and Computer Science, Aalborg University, Denmark, 1995.

[112] Rasmussen., L. K., "BOBLO: an expert system based on Bayesian networks to blood group determination of cattle.", Inf. Téc. Research report 16,, Research Center Foulum, Denmark, PB 23, 8830 Tjele, Denmark., 1995.

[113] R.Fung y Favero, B., "Applying Bayesian Networks to Information Retrieval", *Communications of the ACM*, tomo 38, nº 3, págs. 42–57, 1995.

[114] Salo, A. A., "Tighter Estimates for the Posteriors of Imprecise Prior and Conditional Probabilities", *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, tomo 26, págs. 820–825, 1996.

[115] Shachter, R., "Evaluating Influence Diagrams", *Operations Research*, tomo 34, págs. 871–882, 1986.

[116] Shachter, R., "Probabilistic Inference and Influence Diagrams", *Operations Research*, tomo 36, págs. 589–604, 1988.

[117] Shafer, G., *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, 1976.

[118] Shafer, G. y Shenoy, P., "Local Computation in Hypertrees", Working Paper N. 201, School of Business, University of Kansas, Lawrence, 1988.

[119] Shenoy, P., "A Valuation-Based Language for Expert Systems", *International Journal of Approximate Reasoning*, tomo 3, págs. 383–411, 1989.

[120] Shenoy, P. P. y Shafer, G., "Axioms for Probability and Belief-Function Propagation", en *Uncertainty in Artificial Intelligence, 4*, Shachter *et al.*, eds., págs. 169–198, North-Holland, 1990.

[121] Shimony, S. E. y Santos, J. E., "Exploiting Case-Based Independence for Approximating Marginal Probabilities", *International Journal of Approximate Reasoning*, tomo 14, págs. 25–54, 1996.

[122] Skaaning, C., Jensen, F., Kjærulff, U. y Madsen, A., "Acquisition and Transformation of Likelihoods to Conditional Probabilities for Bayesian Networks", AAAI Spring Symposium, Stanford, USA.

[123] Smets, P., "Belief Functions", en *Non-Standard Logics for Automated Reasoning*, P. Smets, E. Mandani, D. Dubois y H. Prade, eds., Academic Press, London, 1988.

[124] Spirtes, P., Glymour, C. y Scheines, R., *Causation, Prediction and Search*, Springer Verlag, Berlin, 1993.

[125] Sugeno, M., *Theory of Fuzzy Integrals and its Applications*, Tesis Doctoral, Tokyo Institute of Technology, Tokyo, 1974.

[126] Tarjan, R. y Yannakakis, M., "Simple linear-time algorithm to test chordality of grahps, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs", *SIAM Journal of Computing*, tomo 13, págs. 566–579, 1984.

[127] Tessen, B., *Interval Representation of Uncertainty in Artificial Intelligence*, Tesis Doctoral, Department of Informatics, University of Bergen, Norway, 1989.

[128] Tessen, B., "Interval Probability Propagation", *International Journal of Approximate Reasoning*, tomo 7, págs. 95–120, 1992.

[129] Thöne, H., Güntzer, U. y Kießling, W., "Towards Precision of Probabilistic Bounds Propagation", en *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence*, págs. 315–322, 1992.

[130] van der Gaag, L. C., "Computing Probability Intervals under Independence Constraints", en *Uncertainty in Artificial Intelligence, 6*, P. P. Bonissone, M. Henrion, L.Ñ. Kanal y J. F. Lemmer, eds., págs. 457–466, North-Holland, Amsterdam, 1991.

[131] Verdegay-López, J., *Representaicón y Combinación de la Información con Incertidumbre mediante Convexos de Probabilidades*, Tesis Doctoral, Universidad de Granada, 1997.

[132] Verma, G. y Pearl, J., "Indentifying Independence in Bayesian Networks", *Networks*, tomo 20, págs. 507–534, 1990.

[133] Walley, P., "The Elicitation and Aggregation of Beliefs", Inf. téc., University of Warwick, 1982.

[134] Walley, P., *Statistical Reasoning with Imprecise Probabilities*, Chapman and Hall, London, 1991.

[135] Walley, P., "Inferences from multinomial data: learning about a bag of marbles (with discussion)", *Journal of the Royal Statistical Society, Series B*, tomo 58, págs. 3–57, 1996.

[136] Walley, P., "Measures of Uncertainty in Expert Systems", *Artificial Intelligence*, tomo 83, págs. 1–58, 1996.

[137] Walley, P., "A bounded derivative model for prior ignorance about a real-valued parameter", *Scandinavian Journal of Statistics*, tomo 24, págs. 463–483, 1997.

[138] Walley, P., "General Introduction to Imprecise Probabilities", http://ensmain.rug.ac.be/∼ipp/documentation/introduction/introduction.html, 1997/98.

[139] Wang, Z. y Klir, G., *Fuzzy Measure Theory*, Plenum Press, New York, 1992.

[140] Williams, P., "Indeterminate Probabilities", en *Formal Methods in the Methodology of Empirical Sciences*, M. Przelecki, K. Szaniawski y R. Wojcicki, eds., págs. 229–249, D. Reidel, Dordrecht, 1976.

[141] Zadeh, L., "Fuzzy Sets as a Basis for a Theory of Possibility", *Fuzzy Sets and Systems*, tomo 1, págs. 3–28, 1978.

[142] Zadeh, L. A., "A Theory of Approximate Reasoning", en *Machine Intelligence, 9*, J. Hayes y D. Mikulich, eds., págs. 149–194, Elsevier, Amsterdam, 1979.

[143] Zhang, N. L. y Poole, D., "Exploiting Causal Independence in Bayesian Network Inference", *International Journal of Intelligent Research*, tomo 5, págs. 301–328, 1996.