UNIVERSIDAD DE GRANADA

E.T.S. DE INGENIERÍA INFORMÁTICA

Departamento de Ciencias de la Computación
e Inteligencia Artificial

PRECOMPUTACIÓN EN GRAFOS DE DEPENDENCIAS MEDIANTE ALGORITMOS APROXIMADOS

TESIS DOCTORAL

Antonio Salmerón Cerdán

Granada, Febrero de 1998

PRECOMPUTACIÓN EN GRAFOS DE DEPENDENCIAS MEDIANTE ALGORITMOS APROXIMADOS

TESIS DOCTORAL

Antonio Salmerón Cerdán

DIRECTORES:

Serafín Moral Callejón

Luis Daniel Hernández Molinero

Febrero 1998

La memoria titulada

Precomputación en grafos de dependencias mediante algoritmos aproximados

que presenta D. Antonio Salmerón Cerdán, para optar al grado de **Doctor en Informática**, ha sido realizada en el *Departamento de Ciencias de la Computación e Inteligencia Artificial* de la *Universidad de Granada*, bajo la dirección del Dr. D. Serafín Moral Callejón y el Dr. D. Luis Daniel Hernández Molinero.

Granada, Febrero de 1998.

Fdo: D. Antonio Salmerón Cerdán.

Fdo: Dr. D. Serafín Moral Callejón. Dr. D. Luis Daniel Hernández Molinero.

A María y a mi familia.

Agradecimientos.

Quisiera expresar mi agradecimiento a todos aquellos que de algún modo me han ayudado durante la elaboración de esta memoria.

En especial, quiero dar las gracias a mis directores, los doctores Serafín Moral Callejón y Luis Daniel Hernández Molinero, por su constante apoyo, comprensión y generosidad, y por hacer del trabajo algo agradable e ilusionante.

Igualmente agradezco a los doctores Manuel Jorge Bolaños Carmona y Alfredo Martínez Almécija su fundamental ayuda en mis comienzos en la Universidad. Sin ellos hubiera sido más difícil llegar hasta aquí.

También quisiera reconocer la colaboración de Andrés Cano Utrera en el tema de árboles de probabilidad, de Finn Verner Jensen en el desarrollo del esquema HUGIN para funciones de creencia, y de Claus Skaaning Jensen, por cederme los datos sobre los grafos de pedigree para las pruebas de los algoritmos.

No puedo dejar de mencionar a mis compañeros Fernando Reche Lorite, Isabel Ortiz Rodríguez, Carmelo Rodríguez Torreblanca, José del Sagrado Martínez y demás miembros del Departamento de Estadística y Matemática Aplicada y del Grupo de Análisis de Datos de la Universidad de Almería. A ellos les agradezco su cariño y paciencia conmigo.

Agradezco el soporte económico por parte del Vicerrectorado de Investigación de la Universidad de Almería a través de una beca de investigación desde Mayo del 94 a Octubre del 96, y por parte de la CICYT a través del proyecto TIC-1135-C04-02.

Por último quiero tener un recuerdo especial para mi familia, a la que le debo todo, para todos mis amigos, y para mi novia, María, por hacer que todo tenga sentido.

Índice General

Introducción

El uso de modelos gráficos [59] para representar distintos tipos de sistemas es conocido en distintas disciplinas científicas. El origen podría situarse a principios de siglo en el campo de la Física Estadística, en el estudio de sistemas de partículas donde cada una de ellas interactúa con las que están situadas en su proximidad. Para representar de forma sencilla las relaciones de vecindad entre partículas, se utilizaron grafos no dirigidos. Se puso entonces de manifiesto que el uso de estructuras gráficas para representar problemas científicos podría llevar a una mejor comprensión y resolución de los mismos. Surgieron entonces trabajos que aplicaban ideas similares en distintas disciplinas, como la genética, donde la herencia de cualidades entre individuos se representó usando grafos dirigidos. En general, podría decirse que los *modelos gráficos* son formas de representar interacciones entre entidades de un modelo mediante grafos, lo que los convierte en una herramienta universal ampliamente utilizada.

Estos modelos han sido profundamente estudiados en Estadística desde principios de la década de los 80 [22, 57, 59], pero el mayor desarrollo de estas técnicas se ha alcanzado a raíz de su aplicación en el campo de la Inteligencia Artificial, surgiendo lo que se ha dado en llamar *grafos de dependencias*, que son estructuras gráficas que codifican las relaciones de dependencia entre las distintas variables que intervienen en un modelo.

Con el desarrollo de los primeros sistemas expertos, rápidamente se comprobó la necesidad de *representar el conocimiento*, para, de alguna forma, conseguir sistemas capaces de emular el razonamiento humano.

El siguiente paso a la hora de modelizar el conocimiento humano es el tratamiento de relaciones no deterministas. Los primeros sistemas que utilizaban este tipo de relaciones representaban el no determinismo o *incertidumbre* mediante soluciones a medida, destacando los *factores de certeza* utilizados en el sistema experto MYCIN [89], que pueden entenderse como pesos que daban más o menos importancia a las reglas de producción existentes en el sistema. Este tipo de soluciones a medida se mostraron insuficientes, debido a ciertos problemas, en el sentido de falta de expresividad o incapacidad de tratar la incertidumbre de forma global.

Es en la década de los 80 cuando los modelos gráficos comienzan a emplearse como representación del conocimiento incierto en sistemas expertos. En un principio el modelo adoptado para representar la incertidumbre fue la Teoría de la Probabilidad. La ventaja de este formalismo radica en que ofrece un marco de cálculo bien conocido. Esto dio origen a los llamados *sistemas expertos probabilísticos*, hoy en día ampliamente estudiados en la literatura [1, 14, 47, 64, 69, 86]. Los primeros trabajos en este sentido se deben a Kim y Pearl [48] y Pearl [66, 67, 69], quienes proponen mecanismos de inferencia en modelos gráficos probabilísticos donde el grafo asociado es un árbol o un poliárbol. Poco después, Shafer y Shenoy [83] y Lauritzen y Spiegelhalter [58] presentan esquemas de inferencia (propagación) para grafos dirigidos acíclicos sin la restricción de que sean árboles, basándose en una estructura asociada llamada *árbol de cliques*. Estos esquemas fueron mejorados posteriormente por Jensen y otros [44, 45] con el esquema implementado en el sistema HUGIN [2], y más recientemente por Shenoy [88] y Schmidt y Shenoy [76].

Además de la Teoría de la Probabilidad, otros formalismos se han empleado en Inteligencia Artificial para representar y manipular la incertidumbre, destacando la Teoría de la Evidencia de Dempster-Shafer, introducida por Dempster en 1967 [25] y reformulada por Shafer en 1976 [82]. Esta teoría puede verse como una forma alternativa de usar probabilidades, tal y como explican el propio Shafer [85] y Walley [93]. La aplicación de otros formalismos distintos de la probabilidad se debió, en gran medida, al estudio axiomático de las relaciones de independencia (ver, por ejemplo, [69]).

Actualmente, se están desarrollando esquemas de cálculo sobre modelos gráficos de

forma abstracta, estableciendo una serie de axiomas que debe cumplir todo formalismo susceptible de adaptarse a tal esquema de cálculo. En este sentido, deben ser destacados los trabajos de Shenoy y Shafer [87], Cano, Delgado y Moral [11], y Lauritzen y Jensen [60].

Todos los esquemas de inferencia citados anteriormente son *exactos*. Por ejemplo, en el caso de las probabilidades, se calculan los valores de la distribución de probabilidad 'a posteriori'. Sin embargo, en algunos casos un conocimiento aproximado de estos valores es suficiente y en otros, es lo único que se puede conseguir en un tiempo razonable.

El problema de la propagación mediante métodos exactos es NP-duro [18, 65], por lo que en la práctica, los métodos exactos no serán aplicables si el tamaño del problema es suficientemente grande: por ejemplo, en problemas de genética donde el grafo asociado es muy complicado [42]. Aunque el problema de la propagación mediante métodos aproximados es también NP-duro [20], la clase de problemas que se pueden tratar es más amplia.

La mayor parte de los métodos aproximados se basan en técnicas de simulación por Monte Carlo para obtener una estimación de la distribución 'a posteriori'. El problema a resolver es cómo obtener muestras a partir de una distribución de probabilidad que es difícil de manejar. Podemos distinguir tres grupos de métodos para abordar este problema: los basados en *muestreo por importancia* [12, 13, 30, 33, 35, 36, 37, 80], en *muestreo estratificado* [4, 5, 37] y en *cadenas de Markov* [42, 68]. En general, a la hora de simular valores para una variable de cara a obtener una muestra, los algoritmos de Monte Carlo utilizan sólo la información inicial disponible para esa variable, y esa información es de carácter local.

Además de los métodos de simulación, se han propuesto otras formas de algoritmos aproximados de propagación, que se basan en la idea de simplificar el problema en dos sentidos: simplificar la estructura del grafo [50, 51] o reducir el tamaño de las distribuciones de probabilidad [9, 43].

El caso de la Teoría de la Evidencia, recientemente se han desarrollado algoritmos exactos para funciones de creencia [3, 74]. Aquí existe un problema adicional: la com-

plejidad de la regla de combinación de Dempster [65]. Esto ha motivado el desarrollo de métodos aproximados, destacando los trabajos de Moral y Wilson basados en cadenas de Markov [62] y en muestreo por importancia [63].

Objetivos de la memoria

El objetivo de esta memoria se centra en el desarrollo de métodos de Monte Carlo para la propagación de incertidumbre en modelos gráficos, basados en precomputación aproximada. La idea que se presenta consiste en un algoritmo que se desarrolla en dos fases: una primera fase de precomputación aproximada en la que se obtiene una distribución de muestreo y una segunda fase de muestreo por importancia donde se hace uso del cálculo realizado en la primera fase. De esta forma, antes de simular se procesa la información disponible, con la idea de que cuando a una variable se le asigne un valor durante la simulación, se tenga en cuenta toda la información que los recursos disponibles permitan utilizar. La incertidumbre podrá venir dada en forma de probabilidades o de evidencias. Para alcanzar este objetivo hemos establecido las siguientes etapas:

1. *Estudio de las técnicas de simulación.* En este estudio han de recogerse las principales técnicas de propagación de incertidumbre basadas en simulación por Monte Carlo, especialmente las de muestreo por importancia y muestreo estratificado, que actualmente se consideran las más prometedoras.

2. *Determinación de métodos para calcular una distribución de muestreo aproximada.* El punto principal en un esquema de muestreo por importancia es la elección de la distribución de muestreo. Ésta debe ser tan próxima a la distribución exacta como sea posible. Puede ocurrir que la distribución exacta no puede calcularse en un tiempo razonable o bien sería demasiado grande como para almacenarla en un ordenador, por lo que hemos de desarrollar formas de aproximar dicha distribución. La forma de obtener la distribución exacta para cada variable consiste en la eliminación de dicha variable, esto es, combinar todas

las funciones asociadas a dicha variable y luego sumar la distribución resultante sobre ella. Hemos considerado dos formas de aproximar este paso:

(a) No combinar todas las funciones antes de marginalizar, es decir, aproximar una función como el producto de funciones más pequeñas.

(b) Encontrar una representación compacta de la función resultado de la combinación que permita limitar el tamaño de la misma. Los *árboles de probabilidad* [9] son la herramienta adecuada.

Llamaremos *precomputación aproximada* a estos pasos.

3. *Estudio de la técnica de muestreo estratificado* [4, 5] *y adaptación de ésta a nuestro esquema.* La técnica de muestreo estratificado presentaba el problema de que sólo era aplicable a redes de tamaño reducido, debido a problemas de redondeo. Por tanto, se hace necesario el desarrollo de un esquema que permita aplicarlo a redes de tamaño arbitrario. Esto se consigue con lo que llamamos *muestreo estratificado recursivo*. La forma de elegir las distribuciones de muestreo en este caso es igual que en el muestreo por importancia.

4. *Aplicación de la técnica de precomputación aproximada a la Teoría de la Evidencia.* En el caso de las evidencias, los objetivos a cubrir son los mismos. Sin embargo, aquí es necesario desarrollar una representación similar a los árboles de probabilidad pero capaces de representar funciones masa. Una herramienta adecuada son los *árboles jerárquicos* [26, 75]. Habrá que definir operaciones aproximadas sobre esta estructura para el caso de que tengamos un tamaño máximo limitado.

Organización por capítulos

La memoria comienza con un estudio de la representación de incertidumbre mediante grafos, particularizada al caso de que ésta venga dada en forma de probabilidades, en el capítulo 1. En el capítulo 2 se estudian los principales métodos conocidos para

la propagación de probabilidades, tanto de forma exacta como aproximada, haciendo especial hincapié en los métodos de Monte Carlo.

En el capítulo 3 estudiamos en profundidad la técnica de muestreo por importancia, y proponemos una nueva clase de algoritmos de este tipo basados en precomputación aproximada. Se consideran distintos casos particulares y se comprueba experimentalmente el funcionamiento de los algoritmos propuestos, comparándolos con el conocido método de ponderación por verosimilitud.

La misma forma de obtener las distribuciones de muestreo puede aplicarse al caso del muestreo estratificado. En el capítulo 4 se adaptan los algoritmos conocidos de muestreo estratificado a las nuevas distribuciones de muestreo, y se propone un esquema que amplía la validez de los mismos a redes de tamaño arbitrario. Se comprueba experimentalmente que los algoritmos estratificados siempre se benefician del uso de las distribuciones de muestreo calculadas como proponemos.

En el capítulo 5 se propone un algoritmo de muestreo por importancia en el que se representan las distribuciones mediante árboles de probabilidad. El uso de estas estructuras permite una mayor flexibilidad a la hora de la definición de criterios de aproximación de las funciones. Los resultados experimentales muestran que los árboles de probabilidad permiten propagar sobre redes de tamaño considerable. En concreto, se ha utilizado un grafo de pedigree con 441 variables, cedido por Claus S. Jensen (Universidad de Aalborg), para el cual se han obtenido excelentes resultados.

Una vez cubiertos los objetivos para el caso de probabilidades, estudiamos el caso de las evidencias en el capítulo 6. En primer lugar se estudia la inferencia exacta, proponiendo un esquema tipo HUGIN que tiene la ventaja de operar directamente con funciones masa. A continuación se propone una estructura que pretende jugar el papel de los árboles de probabilidad para la Teoría de la Evidencia. Esta estructura es la *representación por semi retículo*, y está basada en los árboles jerárquicos de Sandri [26, 75].

Utilizando las representaciones por semi retículo es posible definir un algoritmo similar al propuesto en el capítulo 5 pero para funciones masa. Este algoritmo se

presenta en el capítulo 7.

La memoria termina con las conclusiones y algunas posibilidades para continuar este trabajo en el capítulo 8.

Capítulo 1

Modelos para el Tratamiento de la Incertidumbre. Grafos de Dependencias

1.1 Introducción

Supongamos que estamos interesados en construir un sistema experto sobre un dominio acerca del cual tenemos conocimiento que conlleva incertidumbre. Podemos distinguir dos tipos de sistemas según el enfoque que adoptan para el tratamiento de la incertidumbre asociada al conocimiento:

1. Los sistemas **extensionales**, también conocidos como sistemas de producción, sistemas basados en reglas, etc. Tratan la incertidumbre como un valor de verdad generalizado asociado a las fórmulas, y calculan la incertidumbre de una fórmula como una función de la incertidumbre asociada a las subfórmulas que la componen.

2. Los sistemas **intensionales**, que asocian la incertidumbre a un subconjunto de mundos posibles o estados de conocimiento. La incertidumbre, en este caso, es

tratada de forma global. Al enfoque que utilizan se le llama enfoque declarativo o basado en modelos.

Un sistema extensional típico es el constituido por los factores de certeza de MYCIN [89], en el que el valor de verdad de una fórmula se obtiene como una función única del valor de verdad de las subfórmulas que la componen. De esta manera, la certeza de la fórmula $A \wedge B$ vendrá dada por al máximo, el producto, o algo similar, de las certezas de A y B, mientras que en la Teoría de la Probabilidad, $P(A \wedge B)$ vendrá dada por el "peso" asignado a la intersección de dos mundos posibles, aquellos en los que A es cierto y aquellos en los que B es cierto.

Los problemas que presentan los sistemas extensionales son los siguientes:

1. No manejan bien inferencias bidireccionales. Si tenemos una regla $A \rightarrow B$, no podemos decir nada acerca de A si observamos B a no ser que tengamos la regla $B \rightarrow A$, en cuyo caso deberíamos eliminar la otra regla para no formar un ciclo.

2. Tienen dificultades para retractarse de anteriores conclusiones. Esto viene dado por el carácter modular y monótono de los sistemas extensionales. Nótese que aunque se tengan valores de verdad generalizados, no por ello se elimina la monotonía, ya que si observamos el antecedente de una regla, esto nos da licencia para actuar sin tener en cuenta el resto de la base de conocimiento, lo cual nos puede hacer llegar a resultados no esperados.

3. No tratan de manera adecuada las fuentes de información dependientes. Cuando se dispara una regla, el peso que se asigna a la conclusión depende sólo del peso de las premisas, pero no se tiene en cuenta de donde vienen estas premisas.

Tanto la Teoría de la Probabilidad como la Teoría de la Evidencia de Dempster-Shafer siguen el enfoque intensional, pero presentan el problema de que a menudo los cálculos dentro de estos formalismos pueden ser computacionalmente intratables. Por ello se hace necesario el estudio de representaciones que permitan abordar los cálculos de forma más sencilla. Aún así, en muchos casos no será posible realizar esos cálculos de

forma exacta y tendremos que conformarnos con estimaciones de los resultados finales. En este capítulo nos centramos en los modelos en que la incertidumbre viene expresada mediante probabilidades. Estudiaremos los modelos de representación que se proponen en la literatura así como los algoritmos exactos y aproximados más importantes para inferencia en estos modelos.

1.1.1 Modelos gráficos y sistemas intensionales

Los sistemas intensionales resuelven los problemas citados en la sección anterior; sin embargo, presentan el problema de la eficiencia. El uso de modelos gráficos, ha permitido la resolución de sistemas intensionales en un tiempo razonable.

El uso de representaciones gráficas para representar la información probabilística se remonta a principios de nuestro siglo, pero ha sido a partir de los inicios de los años 80 cuando las posibilidades computacionales han permitido el desarrollo de un conjunto de teorías y técnicas de cálculo basadas en esas ideas.

Los *diagramas de influencia* [40] surgen en la década de los 80 como una alternativa a los árboles de decisión, de cara a la formulación y computación de problemas de decisión en ambiente de incertidumbre mediante el uso de estructuras de grafos.

También en la década de los 80 aparece una nueva representación mediante estructuras de grafos de los modelos probabilísticos en los que no intervienen variables de decisión: los *grafos de dependencias*.

En el desarrollo de los grafos de dependencias cabe destacar dos líneas de investigación, que responden a las siguientes preguntas:

1. Cómo pueden los grafos de dependencias representar la incertidumbre probabilística.

2. Cómo pueden utilizarse los grafos de dependencias para realizar inferencia probabilística.

La utilización de grafos de dependencias para representar las independencias entre las variables del modelo cuando éstas vienen caracterizadas por una distribución de probabilidad condicionada la estudiaron Darroch, Lauritzen y Speed [22] usando campos de Markov en los que se muestra la importancia de la independencia condicional.

Posteriormente, Pearl [69] caracterizó las independencias expresadas en un grafo dirigido acíclico mediante el concepto de d-separación.

El problema de la *propagación de probabilidades* en grafos de dependencias se plantea como la actualización de los valores de probabilidad de una o varias de las variables del grafo de dependencias dado que se ha observado el valor que toman ciertas variables (observaciones).

Para resolver este problema, se han desarrollado en los últimos años diversos métodos exactos [24, 44, 45, 58, 69, 83, 84, 96]. En general, se puede decir que estos métodos sacan partido de la independencia condicional entre las variables, expresada por la topología del grafo, para realizar los cálculos de forma local. Shachter y otros [81] mostraron cómo todos los algoritmos exactos pueden estudiarse dentro de un marco común basado en el concepto de árbol de grupos. Sin embargo, el problema de estos algoritmos radica en que el problema de la inferencia es NP-duro [18].

Este hecho hace necesario el uso de algoritmos aproximados si se pretende trabajar con una clase de problemas más amplia. La mayoría de estos métodos [4, 5, 12, 13, 15, 16, 30, 33, 36, 37, 42, 68, 73, 80] tratan de obtener una buena estimación de la distribución de probabilidad asociada a la red utilizando técnicas de Monte Carlo. La ventaja de estos algoritmos es que tienen una complejidad polinomial respecto al tamaño de la red. Sin embargo, cuando se requiere un cierto nivel de precisión, de nuevo el problema es NP-duro [20], aunque la clase de problemas tratables es más grande.

Otro grupo de técnicas aproximadas está compuesto por métodos que tratan de simplificar el problema modificando la red original mediante la eliminación de arcos [50, 51] o bien reduciendo el tamaño de las distribuciones de probabilidad [9, 43].

Recientemente, se han estudiado métodos híbridos que combinan cálculos exactos con estimaciones por Monte Carlo [23, 38].

1.2 Definiciones y resultados básicos

Dentro del contexto de la probabilidad, el concepto de independencia entre sucesos viene dado en términos de las funciones de densidad de las variables que representan dichos sucesos. Por ejemplo, dadas dos variables X e Y, con funciones de densidad $f(x)$ y $f(y)$, dichas variables se dicen *independientes* si se cumple que $f(x,y) = f(x) \cdot f(y)$. Sin embargo, parece claro que los seres humanos no conceptualizamos de esta manera las independencias entre sucesos; más bien, esta independencia se manifiesta de manera casi intuitiva. En el caso de que tratemos de modelizar un sistema cuya información nos viene dada por la opinión de un experto, éste seguramente nos dará las independencias expresadas de una forma cualitativa más que cuantitativa. Por ello surge la necesidad de obtener una representación del modelo donde las independencias estén representadas al margen de los datos acerca de las variables. Con esta motivación surgen los modelos de dependencias.

1.2.1 Modelos de dependencias

Para la construcción de modelos usaremos el concepto de variable proposicional, más general que el de variable aleatoria, aunque todos los resultados de esta memoria son válidos para ambos tipos.

Definición 1.1 *Sea Ω un conjunto arbitrario. Diremos que $\mathcal{A}$ es un* σ-álgebra *si verifica:*

1. $\phi \in \mathcal{A}$.
2. $A_i \in \mathcal{A}, \quad 1 \leq i \leq \infty \Rightarrow \bigcup_{i=1}^{\infty} A_i \in \mathcal{A}$.

3. $A \in \mathcal{A} \Rightarrow \Omega - A \in \mathcal{A}$.

Al par $(\Omega, \mathcal{A})$ *se le llama* espacio muestral. *A los elementos de* Ω sucesos elementales, *y a los elementos de* $\mathcal{A}$ sucesos.

Definición 1.2 (Variable aleatoria) *Sea* $(\Omega, \mathcal{A})$ *un espacio muestral. Una* variable aleatoria X *es una función*

$$X : \Omega \to I\!R$$

verificando que

$$X^{-1}(-\infty, x] \in \mathcal{A} \quad \forall x \in I\!R.$$

Una variable aleatoria se dice continua *si puede tomar un número infinito no numerable de valores, y* discreta *si puede tomar, a lo más, un número infinito numerable de valores. Si sólo puede tomar un número finito de valores, se dirá que la variable aleatoria es* discreta y finita.

Definición 1.3 (Variable proposicional) *Sea* $(\Omega, \mathcal{A})$ *un espacio muestral. Una* variable proposicional X *es una función*

$$X : \Omega \to U_X$$

con $U_X \subseteq \mathcal{A}$, *que contiene sucesos exhaustivos y mutuamente excluyentes. De este modo el conjunto de posibles valores de* X *será* $U_X = \{x_i | i \in I\}$ *para un cierto conjunto de índices* $I = \{1, \ldots, n\}$.

Diremos que X *es una variable proposicional* finita *si* $\mathcal{A}$ *contiene un número finito de sucesos, es decir, cuando* Ω *sea finito* [1].

Notación:

En el resto de esta memoria consideraremos siempre variables finitas, y las denotaremos por las últimas letras del abecedario, en mayúsculas. Hablaremos genéricamente

[1] Si Ω es finito, se suele tomar $\mathcal{A} = 2^{\Omega}$, donde 2^{Ω} denota el conjunto de las partes de Ω.

de *variables* refiriéndonos tanto a las aleatorias como a las proposicionales, en los casos en que no haya necesidad de distinguir. Dado un conjunto de variables $\{X_i\}_{i\in I}$, con $I = \{1, \ldots, n\}$, supondremos que cada variable X_i toma valores en un conjunto finito U_i y notaremos por X_I a la variable $|I|$-dimensional $X_I = (X_i)_{i\in I}$ con valores en $U_I = \prod_{i\in I} U_i$. Si $x_I \in U_I$ y $J \subset I$ notaremos por $x^{\downarrow J}$ al elemento x_J de $U_J = \prod_{j\in J} U_j$ obtenido a partir de x_I eliminando aquellas coordenadas que no están en J.

Definición 1.4 (Modelo de dependencias) *Diremos que $\mathcal{M}$ es un* modelo de dependencias *si $\mathcal{M}$ establece una regla que asigna el valor verdadero o falso al predicado*

$$I(X_I, X_K, X_J) \equiv X_I \text{ es independiente de } X_J \text{ dado } X_K$$

para cualesquiera dos conjuntos de variables X_I, X_J distintos de otro conjunto X_K en algún espacio medible $(\Omega, \mathcal{A})$. A la relación $I(,,)$ se le llama relación de independencia.

A continuación veremos cuatro propiedades de la relación de independencia que serán verificadas por todo modelo probabilístico. A un conjunto de independencias verificando estas propiedades se la llama *semigrafoide*.

Propiedades de la relación de independencia.

1. **Simetría:** Si X_I es independiente de X_J dado X_K, entonces X_J es independiente de X_I dado X_K, es decir,

$$I(X_I, X_K, X_J) \Leftrightarrow I(X_J, X_K, X_I).$$

2. **Descomposición:** Si X_I es independiente de $X_J \cup X_W$ dado X_K, entonces X_I es independiente de X_J dado X_K y X_I es independiente de X_W dado X_K, es decir,

$$I(X_I, X_K, X_J \cup X_W) \Rightarrow I(X_I, X_K, X_J) \wedge I(X_I, X_K, X_W).$$

3. **Unión débil:** Refleja el hecho de que aprender cierta información que es irrelevante (X_W) no puede hacer que otra información irrelevante (X_J) se vuelva relevante, es decir,

$$I(X_I, X_K, X_J \cup X_W) \Rightarrow I(X_I, X_K \cup X_W, X_J).$$

4. **Contracción:** Si X_W es irrelevante para X_I después de conocer alguna información irrelevante X_J, entonces X_W debe haber sido irrelevante antes de conocer X_J, es decir,

$$I(X_I, X_K, X_J) \wedge I(X_I, X_K \cup X_J, X_W) \Rightarrow I(X_I, X_K, X_J \cup X_W).$$

1.2.2 Representación de modelos de dependencias

Dado un modelo de dependencias $\mathcal{M}$, podemos representarlo mediante un grafo de dependencias $\mathcal{G}_\mathcal{M}$ en el que cada variable del modelo se corresponda con un nodo del grafo y tal que la estructura del grafo represente las relaciones entre las variables del modelo. En los siguientes ejemplos veremos cuáles son los tipos de dependencias capaces de ser representados por un grafo de dependencias.

Ejemplo 1.1 (F.V. Jensen, 1996 [47]). *El inspector Smith espera la llegada del Señor Holmes y el Dr. Watson. Ambos se retrasan y el inspector está impaciente porque tiene otra visita que atender. Se acerca a la ventana para ver si las calles están heladas, porque si es así, probablemente ambos tendrán un accidente, pues son pésimos conductores.*

De repente, la secretaria del inspector le dice que Watson ha llamado diciendo que ha tenido un accidente. Entonces Smith supone que el motivo ha sido el hielo, y que por lo tanto, muy probablemente Holmes también tendrá un accidente, así que piensa: "Bien, puedo atender la otra visita".

Pero su secretaria le aclara que no hace tanto frío como para que se hielen las carreteras, además de que han puesto sal para evitarlo. Entonces Smith rectifica su decisión y decide esperar diez minutos más a Holmes.

Obsérvese que antes de tener información sobre el estado de las carreteras, el hecho de que Watson tuviera un accidente incrementa la creencia del inspector sobre que las carreteras estén heladas, y a su vez, aumenta su creencia sobre el hecho de que Holmes también tuviera un accidente. Es decir, si no se sabe nada sobre las carreteras, existe

una dependencia entre los sucesos "Watson tiene un accidente" y "Holmes tiene un accidente". Sin embargo, una vez se sabe que las carreteras no están heladas, el hecho de que Watson haya tenido un accidente no influye en la creencia acerca de que Holmes también lo tenga. Es decir, ambos sucesos se vuelven independientes.

El grafo de dependencias que representa el modelo seguido por el inspector puede verse en la figura 1.1. En concreto, la relación de independencia reflejada por dicho grafo será

$$I(\text{Holmes,Hielo,Watson}).$$

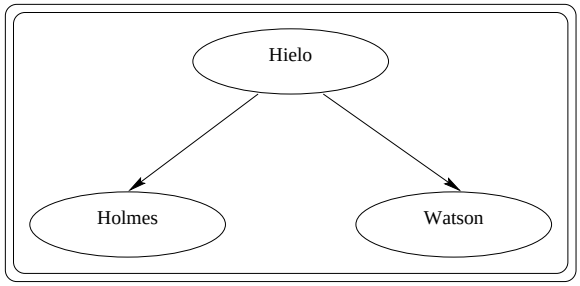

Figura 1.1: Grafo de dependencias representando el modelo del inspector Smith.

Ejemplo 1.2 (F.V. Jensen, 1996 [47]). *Ahora el señor Holmes vive en Los Ángeles. Una mañana cuando sale de casa se da cuenta de que el césped de su jardín está mojado. Su creencia sobre que haya llovido o se haya dejado el aspersor encendido aumenta.*

Luego se da cuenta de que el césped de su vecino el Dr. Watson también está húmedo. Esta observación hace que aumente su creencia sobre el hecho de que haya llovido la noche anterior. El grafo de dependencias asociado a este modelo es el de la figura 1.2, y las independencias que codifica son

$$I(\text{Watson,Lluvia,Holmes}),$$

$$\neg I(\text{Holmes,Lluvia,Aspersor}),$$

$$I(\text{Lluvia}, \emptyset, \text{Aspersor}).$$

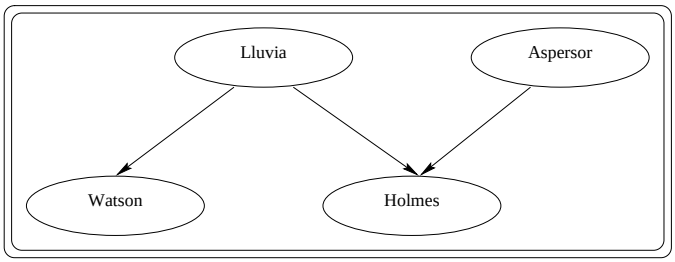

Figura 1.2: Grafo de dependencias para el modelo del césped mojado.

Ejemplo 1.3 *Durante un viaje, Holmes hace una llamada a su amigo el Dr. Watson. Holmes no tenía ninguna información acerca del estado del tiempo en Londres. Entonces Watson le dijo que, por la mañana, su coche había sufrido una parada mientras transitaba por cierta avenida.*

Inmediatamente, la creencia de Holmes sobre el hecho de que había llovido ese día aumentó, avalada por el hecho de haber sufrido una parada en el coche. Sin embargo, a continuación Watson dijo que la calzada se encontraba totalmente inundada, con lo cual el asunto del coche dejó de informar a Holmes sobre el que lloviera o no (la inundación se pudo deber a otras causas).

Este problema se puede modelizar con el grafo de la figura 1.3, y la relación de independencia que codifica es

$$I(\text{Lluvia,Avería,Charcos}).$$

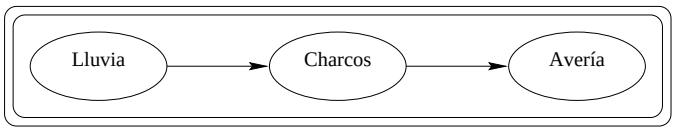

Figura 1.3: Grafo de dependencias para el modelo de la avería del coche.

La representación de las relaciones de independencia en un grafo se hace a partir del concepto de d-separación [69], que formalizaremos a continuación.

Definición 1.5 (Grafo dirigido acíclico). *Un grafo dirigido $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ se dice* acíclico *si no existe una secuencia de nodos $X_{i_1}, \ldots, X_{i_r} \in \mathcal{V}$ tales que $X_{i_1} = X_{i_r}$ y $(X_{i_j}, X_{i_{j+1}}) \in \mathcal{E}$ para todo $j = 1, \ldots, r-1$.*

Definición 1.6 (Camino bloqueado) *Sea $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un grafo dirigido acíclico. Sea $X_K \subseteq \mathcal{V}$, $X_i, X_j \in \mathcal{V} - X_K$ y α un camino entre X_i y X_j. Se dice que el camino está bloqueado por X_K si se cumple alguna de estas dos condiciones:*

1. *Hay algún vértice $X_k \in X_K$ del camino α tal que los arcos incidentes en X_k se encuentran cola con cola o cabeza con cola.*
2. *Hay algún vértice $X_k \in \mathcal{V}$ del camino α tal que los arcos incidentes en X_k se encuentran cabeza con cabeza, y es tal que ni X_k ni ninguno de sus descendientes están en X_K.*

Definido el concepto de bloqueo, se puede definir la relación de d-separabilidad, que representa una relación de independencia en el caso de grafos.

Definición 1.7 (d-separación) *Sea $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un grafo dirigido acíclico.*

1. *Sean $X_K \subsetneq \mathcal{V}$ y $X_i, X_j \in \mathcal{V} - X_K$. Se dice que X_i y X_j están* d-separados *por X_K si y sólo si cualquier camino entre X_i y X_j está bloqueado por X_K. Este hecho se denota por $< X_i | X_K | X_j >_{\mathcal{G}}$.*
2. *Sean $X_{K_1}, X_{K_2}, X_K \subsetneq \mathcal{V}$ y disjuntos. Se dice que X_{K_1} y X_{K_2} están* d-separados *por X_K, y se denota $< X_{K_1}, X_K, X_{K_2} >_{\mathcal{G}}$ si cualquier camino entre $X_i \in X_{K_1}$ y $X_j \in X_{K_2}$ está bloqueado por X_K.*

A la hora de utilizar un grafo de dependencias como representación de un modelo de dependencias, lo que en un caso ideal interesa es, por un lado, que cada variable del modelo se corresponda con un vértice del grafo, y por otro, que cada independencia del modelo se corresponda unívocamente con un bloqueo de caminos en el grafo. Esto se expresa en la siguiente definición.

Definición 1.8 (Representación perfecta) *Un grafo* $\mathcal{G}$ *es una* representación perfecta *del modelo* $\mathcal{M}$ *si y sólo si*

$$I(X_I, X_K, X_J)_{\mathcal{M}} \Leftrightarrow < X_I | X_K | X_J >_{\mathcal{G}} .$$

Sin embargo, esta condición no siempre se cumple, por lo que en muchos de los casos será necesario limitarse a las siguientes representaciones parciales de un modelo de dependencias:

Definición 1.9 (D-map) *Un grafo* $\mathcal{G}$ *es un* D-map *de* $\mathcal{M}$ *si y sólo si*

$$I(X_I, X_K, X_J)_{\mathcal{M}} \Rightarrow < X_I | X_K | X_J >_{\mathcal{G}} .$$

Es decir, las independencias del modelo quedan reflejadas en el grafo, pero puede haber dependencias en el grafo que no estén en el modelo.

Definición 1.10 (I-map) *Un grafo* $\mathcal{G}$ *es un* I-map *del modelo* $\mathcal{M}$ *si y sólo si*

$$I(X_I, X_K, X_J)_{\mathcal{M}} \Leftarrow < X_I | X_K | X_J >_{\mathcal{G}} .$$

Es decir, las independencias del grafo quedan reflejadas en el modelo, pero puede ocurrir que independencias en el modelo sean dependencias en el grafo.

Definición 1.11 *Un I-map se dice* minimal *si al eliminar un arco o arista del grafo, éste deja de ser I-map.*

Hasta ahora, hemos estudiado la parte cualitativa de los modelos de incertidumbre. Más detalles pueden encontrarse en [69].

1.2.3 El modelo probabilístico de dependencias

El concepto básico en el modelo probabilístico es el de *independencia condicional.*

Definición 1.12 (Independencia condicional) *Sea $(\Omega, \mathcal{A}, P)$ un espacio probabilístico. Dados tres sucesos $A, B, C \in \mathcal{A}$, con $P(B, C) > 0$, se dice que A y B son* condicionalmente independientes *dado C si se cumple que*

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

Teniendo en cuanta esto, pueden definirse los modelos probabilísticos de la siguiente manera:

Definición 1.13 (Modelo probabilístico) *Se dice que un modelo $\mathcal{M}$ es un* modelo probabilístico de dependencias*, si la relación de independencia $I(,,)$ viene dada por la expresión*

$$I(X, Z, Y) \Leftrightarrow P(X|Z, Y) = P(X|Z),$$

para alguna distribución de probabilidad P en algún espacio muestral.

La ventaja de este modelo es que se pueden aplicar resultados de la Teoría de la Probabilidad para expresar propiedades como las siguientes, enunciadas por Pearl [69].

Teorema 1.1 *Sean X_I, X_K, X_J y X_W, conjuntos disjuntos de variables. Todo modelo probabilístico verifica las propiedades de semigrafoide. Además, si la probabilidad P es estrictamente positiva se cumple:*

- **Intersección:**

$$I(X_I, X_K \cup X_W, X_J) \wedge I(X_I, X_K \cup X_J, X_W) \Rightarrow I(X_I, X_K, X_J \cup X_W).$$

1.2.4 Representación del modelo probabilístico

Dado un modelo probabilístico $\mathcal{M}$ sobre un espacio probabilístico $(\Omega, 2^{\Omega}, P)$ es posible construir I-maps minimales tanto sobre grafos no dirigidos (redes de Markov) como sobre grafos dirigidos (redes bayesianas). Nos centraremos en el caso de los grafos dirigidos, siguiendo los ejemplos de las figuras 1.1, 1.2 y 1.3.

Definición 1.14 (Red bayesiana o red causal) *Sea $\mathcal{M}$ un modelo probabilístico sobre un espacio probabilístico $(\Omega, 2^{\Omega}, P)$. Dada una distribución de probabilidad P sobre un conjunto de variables X, una* red bayesiana *o* red causal *es un grafo dirigido acíclico $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ en el que:*

1. *Existe una correspondencia biyectiva entre las variables de X y los vértices de $\mathcal{V}$.*
2. *$\mathcal{G}$ es un I-map minimal del modelo $\mathcal{M}$.*
3. *Cada variable X_i tiene asociada una distribución de probabilidad condicionada a las variables $X_{F(i)}$, $P(X_i|X_{F(i)})$, donde $F(i)$ denota el conjunto de índices de los padres de X_i en el grafo.*

Teniendo en cuenta la codificación de independencias en una red bayesiana, se puede demostrar la siguiente proposición.

Proposición 1.1 (Regla de la cadena) *Dada una red bayesiana sobre un conjunto de variables $X = \{X_1, \ldots, X_n\}$, se cumple que la distribución conjunta en X es el producto de las distribuciones condicionadas especificadas en la red, es decir,*

$$P(X) = \prod_{i=1}^{n} P(X_i|X_{F(i)}). \tag{1.1}$$

Esta proposición muestra una forma compacta de representar la distribución conjunta sobre las variables de la red como el producto de una serie de distribuciones condicionadas. Esto es de especial interés cuando el número de variables de la red es elevado, puesto que el espacio necesario para almacenar la distribución conjunta de probabilidad crece exponencialmente con el número de variables.

Capítulo 2

Propagación de Probabilidades en Grafos de Dependencias

2.1 Introducción

En este capítulo presentamos un estudio de las técnicas conocidas, tanto exactas como aproximadas, para la propagación en grafos de dependencias. Más concretamente, nos restringiremos al caso de las redes bayesianas, dejando el estudio de la propagación de funciones de creencia para capítulos posteriores.

La idea de la **propagación** en redes bayesianas consiste en calcular la distribución a posteriori de ciertas variables de interés dado que se conoce el valor que toman algunas otras variables.

En los últimos años se han propuesto numerosos algoritmos para tratar este problema de manera exacta [21, 45, 58, 69, 77, 78, 79, 84]. En general podemos decir que estos métodos sacan partido de la independencia condicional entre variables expresada por la propia estructura del grafo, para realizar la propagación mediante cálculos locales. La mayoría de estos métodos se basan en la construcción de un árbol de grupos de

variables a partir de una triangulación del *grafo moral* [1] asociado a la red bayesiana.

Aunque proporcionan resultados exactos, el problema en sí es NP-duro [18]. Por ello, si la red es suficientemente complicada, el tiempo de respuesta de estos algoritmos crece exponencialmente en el número de variables.

Este hecho motiva el desarrollo de algoritmos aproximados. La mayoría de ellos tratan de obtener una buena estimación de las probabilidades asociadas a la red empleando técnicas de Monte Carlo. En este caso, el problema de la propagación se enfoca como la obtención de una muestra a partir de una distribución de probabilidad que es difícil de manejar. Aunque la inferencia aproximada es también un problema NP-duro [20], la clase de problemas que se pueden abordar es más amplia que en el caso de los métodos exactos.

Recientemente, con la intención de combinar las ventajas de las técnicas exactas y de las aproximadas, se han desarrollado nuevos métodos que hibridan ambas metodologías. La idea básica consiste en dividir la red en una serie de grupos y en cada uno de ellos aplicar una técnica exacta o aproximada, según sea conveniente. En la actualidad, los mejores resultados se han dado al combinar algoritmos exactos con muestreadores de Gibbs [23], así como muestreadores por importancia con el algoritmo exacto de Shafer-Shenoy [84], idea ésta realizada por Hernández y Moral [38].

Comenzaremos estableciendo alguna notación y definiciones básicas en la sección 2.2, para describir algunas técnicas exactas en la sección 2.3 y las aproximadas en la sección 2.4.

2.2 Notación y Definiciones Básicas

En el capítulo anterior vimos que una *red bayesiana* es un grafo dirigido acíclico donde cada nodo representa a una variable y la topología del grafo codifica las relaciones de independencia entre las variables, de acuerdo con el criterio de d-separabilidad [69].

[1] Ver sección 2.3.2.

Dadas las independencias representadas en el grafo, la distribución conjunta de todas las variables de la red viene determinada especificando una distribución de probabilidad para cada nodo condicionada a sus padres en el grafo (ver proposición 1.1).

Definición 2.1 (Potencial) *Dado un conjunto de variables X_I, llamaremos* potencial *a cualquier función $f : U_I \to \mathbb{R}_0^+$.*

Obsérvese que una distribución de probabilidad es un potencial. Dado un potencial f, denotaremos por $s(f)$ al conjunto de índices de las variables para las que está definido. En el caso de la definición anterior, $s(f) = I$. Utilizaremos los términos *función* y *potencial* de forma indistinta cuando no haya lugar a confusión.

Sea $X = \{X_1, \ldots, X_n\}$ el conjunto de variables de una red bayesiana. A partir de ahora denotaremos la distribución condicionada de X_i dados sus padres en la red $F(i)$, por

$$f_i(x) = f_i(x^{\downarrow i}, x^{\downarrow F(i)}) \quad \forall i \in N \quad \forall x \in U_{s(f_i)}, \tag{2.1}$$

donde $N = \{1, \ldots, n\}$, y

$$\sum_{x_i \in U_i} f_i(x_i, x) = 1, \quad \forall x \in U_{F(i)}. \tag{2.2}$$

Entonces, la distribución conjunta para la variable aleatoria n-dimensional X se puede expresar como

$$p(x) = \prod_{i \in N} f_i(x^{\downarrow s(f_i)}) \quad \forall x \in U_N. \tag{2.3}$$

Una *observación* es el conocimiento certero acerca del valor de una variable: $X_i = e_i$. El conjunto de observaciones se denota por e y se denomina *conjunto evidencia*. Por E se entiende el conjunto de índices de las variables observadas.

Toda observación $X_i = e_i$ tiene asociada una función de Dirac definida sobre U_i como sigue:

$$\delta_{e_i}(x) = \begin{cases} 1 & \text{si} \quad e_i = x, \\ 0 & \text{si} \quad e_i \neq x. \end{cases} \tag{2.4}$$

A continuación definimos una serie de conceptos que usaremos a lo largo de la memoria.

Definición 2.2 (Marginalización) *Dado un potencial f definido sobre un conjunto de variables X_I, y dado $J \subseteq I$, se define la* marginalización *de f sobre una variable X_J (o la* eliminación *de las variables con índices en $I - J$) como un nuevo potencial $f^{\downarrow J}$ definida sobre el conjunto J dado por la expresión*

$$f^{\downarrow J}(x) = \sum_{\substack{y \in U_I \\ y^{\downarrow J} = x}} f(y), \quad \forall x \in U_J. \tag{2.5}$$

Definición 2.3 (Combinación) *Dados r potenciales $f_1, \ldots, f_r$, cada uno de ellos definidos sobre los conjuntos $I_1, \ldots, I_r$, se define su* combinación *o* producto *como un potencial definido sobre el conjunto de variables con índices en $I = \bigcup_{i=1}^{r} I_i$ y dado por la expresión*

$$f(x) = \bigotimes_{i=1}^{r} f_i(x^{\downarrow I_i}) = \prod_{i=1}^{r} f_i(x^{\downarrow I_i}), \quad \forall x \in U_I. \tag{2.6}$$

Obsérvese que el producto en la anterior expresión puede realizarse en cualquier orden. Por lo tanto, la combinación es conmutativa.

Definición 2.4 (Restricción) *Sea f un potencial definido para el conjunto de variables con índices en I, sea un conjunto de variables X_J, $J \subset I$ y un elemento fijo $x_0 \in U_J$.*

La restricción *de f a los valores x_0 es un nuevo potencial $f^{R(X_J=x_0)}$ definido sobre las variables con índices en $I-J$ de acuerdo con la siguiente expresión:*

$$f^{R(X_J=x_0)}(x) = f(y), \quad \forall x \in U_{I-J}, \tag{2.7}$$

tal que $y \in U_I$, es tal que $y^{\downarrow I-J} = x$ e $y^{\downarrow J} = x_0$.

La restricción puede simplificar el problema cuando hay observaciones sobre las variables de la red.

2.3 Técnicas Exactas de Propagación

Una forma de resolver el problema de la propagación de forma exacta sería simplemente marginalizar a partir de la distribución conjunta de todas las variables de la red. Sin embargo, dicha distribución viene dada de forma factorizada (ver ecuación (1.1)). Calcular la conjunta a partir de la factorizada plantea problemas tanto de eficiencia como de almacenamiento; por ejemplo, si todas las variables de la red son binarias (el caso más sencillo), sería necesario almacenar 2^n valores de probabilidad para una red de n variables. Esto no es viable para redes de tamaño considerable.

Una forma más eficiente de resolver el problema consiste en sacar partido de la independencia condicional entre las variables expresada implícitamente en la red, caracterizada mediante el concepto de d-separación. De esta forma se puede obtener la distribución a posteriori mediante cálculos locales, acelerando el proceso de inferencia y sin necesidad de trabajar con la distribución conjunta.

Judea Pearl [67, 69] fue el precursor de los algoritmos de propagación basados en cálculos locales. Hay que destacar su algoritmo para inferencia en poliárboles, ya que la metodología utilizada ha sido la base para el desarrollo de los demás métodos exactos.

Posteriormente surgieron algoritmos capaces de realizar la propagación en cualquier tipo de grafo dirigido acíclico (D.A.G.), destacando la técnica de eliminación de nodos

[21, 85, 96] y los llamados métodos de agrupamiento [58, 44, 45, 84]. Los métodos de agrupamiento se basan en la construcción de un árbol de grupos de variables a partir de la red original, y a partir de él desarrollan un proceso de inferencia mediante cálculos locales basado en el paso de mensajes entre los nodos, similar al usado por Pearl para poliárboles.

Todas las técnicas exactas pueden estudiarse desde un punto de vista común, basado en el concepto de árbol de grupos de variables, resultando todas ellas conceptualmente equivalentes, con la única diferencia de la forma de construir dicho árbol [81]. A continuación explicamos en detalle algunas de estas técnicas.

2.3.1 Técnica de eliminación de variables

El objetivo es obtener la distribución a posteriori $P(X_j = x_j | X_E = e_E)$ para una variable X_j dado un conjunto de variables observadas X_E. La idea básica consiste en considerar un orden de las variables donde la variable X_j sea la última. En cada paso se va eliminando una de las variables, combinando todas las funciones en las que está involucrada dicha variable y luego marginalizando sobre ella. Después de cada eliminación, las funciones resultantes ya no dependen de esa variable, pero contienen la información que ésta aportaba. Al final del proceso se obtiene un potencial para la variable objetivo. La probabilidad 'a posteriori' que se busca es proporcional a este potencial.

El algoritmo de propagación puede enunciarse como sigue:

Algoritmo de eliminación de variables

---

1. Se parte de una familia de funciones $H = \{f_1, \ldots, f_n\}$ formada por las distribuciones condicionadas asociadas a la red causal, un conjunto de observaciones $\{e_l\}_{l \in E}$ y una variable objetivo X_j.

2. Para cada función $f \in H$, restringir f a los valores $\{e_l\}_{l \in E}$ de acuerdo a la ecuación (2.7).

3. Considerar una permutación σ del conjunto $\{1, \ldots, n\}$ tal que $\sigma(n) = j$.

4. Para $i = 1$ hasta $n - 1$ hacer:

   (a) Eliminar la variable $X_{\sigma(i)}$ siguiendo el siguiente procedimiento:

      i. Combinar todas las funciones que están definidas para la variable $X_{\sigma(i)}$, obteniendo una nueva función h_i.

      ii. Eliminar $X_{\sigma(i)}$ marginalizando h_i sobre $s(h) - \{\sigma(i)\}$. Añadir el resultado al conjunto H.

      iii. Eliminar de H todas las funciones que fueron combinadas para obtener h_i.

Al final de este algoritmo, la distribución $P(X_j = x_j | X_E = e_E)$ para todo $x_j \in U_j$ se obtiene como $f(x_j) = \prod_{h \in H} h(x_j)$.

2.3.2 Métodos de agrupamiento

Estos métodos resuelven el problema de la presencia de ciclos agrupando las variables en conjuntos y formando un árbol no dirigido con dichos conjuntos para aplicar una metodología de paso de mensajes similar a la de Pearl.

En estos árboles los nodos son grupos de variables de la red bayesiana original, y se exige que las relaciones reflejen el criterio de d-separación. Nos referiremos a éstos como *árboles de grupos*. Todo árbol de grupos procedente de una red bayesiana debe verificar las siguientes condiciones:

1. Para cada par de nodos del árbol Γ y Γ' con $S = \Gamma \cap \Gamma' \neq \phi$, todos los nodos del

camino que los une contienen a S. Esta propiedad es conocida por *propiedad de unión*.

2. Cada *familia*[2] de la red se encuentra en al menos un nodo del árbol.

3. Toda distribución condicionada de la red, $f_i(x^{\downarrow i}, x^{\downarrow F(i)})$ se asigna a un nodo del árbol que contenga a la familia formada por X_i y $X_{F(i)}$.

Dado un nodo Γ_i, su *potencial asociado*, ψ_i, será el producto de las distribuciones que estén asignadas a dicho nodo. En caso de no tener ninguna, se considerará que el potencial es constantemente igual a 1.

Respecto al mantenimiento de la d-separación en el árbol puede enunciarse lo siguiente. Dado un árbol de grupos procedente de una red bayesiana, consideremos para cada par de nodos vecinos un nuevo nodo situado entre ellos dos que sea la intersección de los dos nodos anteriores. Estos nuevos grupos se denominan *separadores*. Nótese que si se añaden todos los separadores al árbol de grupos se sigue teniendo un árbol de grupos. En estas condiciones, la independencia condicional de la red bayesiana se puede representar de forma explícita a través de los separadores [81]: "Cualesquiera dos grupos en un árbol de grupos son condicionalmente independientes dado cualquier grupo o separador correspondiente a cualquier arco que intervenga en el camino entre ellos".

Un árbol de grupos de especial importancia es el *árbol de cliques*, que es aquel en el que ningún grupo es subconjunto de un grupo vecino. Para obtener un árbol de cliques a partir de un árbol de grupos, basta con incluir los grupos más pequeños en los superconjuntos que los contengan [81]. La construcción del árbol de grupos suele requerir la previa triangulación de la red causal [8, 34, 49, 56], lo que pasa por un proceso previo de *moralización* del grafo, que consiste en conectar todos los nodos de la red que tengan hijos en común e ignorando el sentido de los arcos. El resultado es un grafo no dirigido en el que se mantienen dependencias que se perderían si obviáramos la dirección de los arcos sin más. A éste grafo se le llama *grafo moral*.

[2]Por *familia* se entiende el conjunto que forman un nodo y sus padres en el grafo.

En cuanto a la determinación de las distribuciones a posteriori de una variable, la idea es la siguiente: Una vez construido el árbol de cliques se desarrolla un paso de mensajes entre los nodos adyacentes, de manera que al finalizar el algoritmo tengamos en cada nodo la distribución *a posteriori* de las variables asociadas a éstos. De este modo, la distribución 'a posteriori' sobre una variable será la marginalización de aquella distribución que esté asociada al clique al que pertenezca la variable.

A continuación describiremos el algoritmo de agrupamiento más utilizado en la actualidad. Fue desarrollado por Jensen y otros [44, 45], y es un refinamiento del método de Lauritzen y Spiegelhalter [58] que se ha implementado en la utilidad para construir sistemas expertos llamada HUGIN [2].

El algoritmo parte de un árbol de cliques construido a partir del grafo moral asociado a la red causal original. Cada familia de la red se asigna exactamente a un clique C_i, al que se le asigna un potencial ϕ_i igual al producto de las distribuciones de las familias asociadas al clique[3]. Este producto se almacena en forma conjunta:

$$\phi_{C_i}(x^{\downarrow C_i}) = \prod_{F^*(i)\subset C_i} f_i(x^{\downarrow i}, x^{\downarrow F(i)}), \tag{2.8}$$

donde $F^*(i) = \{i\} \cup F(i)$.

La operación básica del algoritmo de propagación HUGIN es la *absorción*:

Definición 2.5 (Absorción) *Dados dos cliques adyacentes C_i y C_j con separador S, se dice que C_i* absorbe *de C_j si los nuevos potenciales de los cliques involucrados pasan a ser:*

$$\phi^*_S(x^{\downarrow S}) = \sum_{x^{\downarrow C_j - S}} \phi_{C_j}(x^{\downarrow C_j}) \quad \forall x \in \Omega_{\mathcal{V}}, \tag{2.9}$$

$$\phi^*_{C_i}(x^{\downarrow C_i}) = \phi_{C_i}(x^{\downarrow C_i}) \cdot \frac{\phi^*_S(x^{\downarrow S})}{\phi_S(x^{\downarrow S})} \quad \forall x \in \Omega_{\mathcal{V}}, \tag{2.10}$$

$$\phi^*_{C_j}(x^{\downarrow C_j}) = \phi_{C_j}(x^{\downarrow C_j}) \quad \forall x \in \Omega_{\mathcal{V}}. \tag{2.11}$$

[3]Si un clique no contiene a ninguna familia completa, se le asigna potencial constante igual a 1.

El proceso de propagación se realiza a través de un paso de mensajes en dos fases: RECOLECCIÓN y DISTRIBUCIÓN, que se pueden describir como sigue:

- **Petición de recolección.** Supongamos que un clique C_i recibe un mensaje de recolección de otro clique C_j; entonces C_i manda un mensaje de recolección a todos sus vecinos excepto a C_j. Cuando un clique termina la operación de recolección, el clique que la solicitó absorbe de éste.
- **Petición de distribución.** Si un clique C_i recibe un mensaje distribución desde un clique C_j, entonces C_i absorbe de C_j y manda un mensaje de distribución a todos sus vecinos excepto C_j.

El algoritmo puede enunciarse así:

Algoritmo HUGIN

---

1. Construir un árbol de cliques.
2. Calcular los potenciales asociados a cada clique.
3. Incorporar las evidencias. Cada evidencia se incorpora a todo clique que contenga a la variable observada multiplicando su potencial por el potencial asociado a la evidencia (ver ecuación (2.4)).
4. Seleccionar un clique R como raíz.
5. Enviar un mensaje de recolección a partir de R.
6. Enviar un mensaje de distribución desde R.

Este algoritmo puede considerarse el más eficiente entre los métodos exactos, pero aun así no es aplicable a redes de gran tamaño como las que surgen en problemas de genética [42].

2.4 Técnicas aproximadas

Los algoritmos aproximados surgieron con el propósito de resolver los casos peores para los métodos exactos en un tiempo más razonable, generalmente mediante técnicas de Monte Carlo, a cambio de la pérdida de la exactitud de los cálculos. La inferencia por métodos de simulación es también un problema NP-duro cuando se requiere una precisión determinada [20]; sin embargo, el conjunto de problemas resolubles es mayor que para los métodos exactos. Podemos distinguir tres clases de algoritmos por Monte Carlo: los basados en *muestreo por importancia*, en *muestreo estratificado* [4, 5, 37] y en *cadenas de Markov*. Una exposición del funcionamiento de varios algoritmos aproximados puede encontrarse en [19]. El problema a resolver es el mismo: cómo obtener muestras de una distribución de probabilidad que es difícil de manejar.

En el *muestreo por importancia* se usa una distribución modificada a la hora de obtener las muestras independientes en el proceso de simulación. Estas muestras son ponderadas según algún criterio que refleje el "peso" o validez de la simulación. El primer algoritmo de esta clase fue el llamado *muestreo lógico probabilístico*, desarrollado por Henrion [33]. Se basa en realizar una propagación hacia delante, muestreando primero en los nodos raíz y continuando en sentido descendente hasta llegar a las hojas. Como funciones de muestreo utiliza las distribuciones condicionadas de la red, y los pesos son siempre 1 ó 0. Un peso valdrá 1 cuando la configuración obtenida es coherente con las observaciones y 0 si no lo es. Este algoritmo funciona bien cuando no hay observaciones; sin embargo, cuanto mayor es el número de ellas, peores son los resultados, al disminuir la probabilidad de obtener una configuración para las variables observadas que coincida con los valores observados.

Posteriormente, surgió el algoritmo llamado de *ponderación por verosimilitud*, que

mejora al anterior. Fue propuesto independientemente por Fung y Chang [30] y por Shachter y Peot [80]. En este caso, las variables observadas no se simulan, sino que se instancian directamente al valor observado. Luego, se pesa cada configuración con la probabilidad de obtener dicha configuración. El comportamiento es radicalmente mejor que el del algoritmo de Henrion, pero pueden darse casos sencillos en los que todos los pesos se hacen 0 ó 1 [13]; éste sería el caso en el que el valor observado para una variable tiene probabilidad 0 excepto para una sola configuración de los padres.

De especial interés dentro de este grupo de técnicas es la clase de algoritmos propuesta por Cano, Hernández y Moral [13]. Ellos refinan el proceso de selección de las funciones de muestreo, utilizando aquellas con menos entropía (las más informativas, por tanto) para simular y aquellas con más entropía (las menos informativas) para pesar la simulación. En este caso, el problema de que todos los pesos sean 0 ó 1 no se presenta.

En el próximo capítulo estudiaremos en profundidad los algoritmos basados en muestreo por importancia, limitándonos en esta sección sólo a los más sencillos, denominados también *métodos de propagación hacia delante*.

Los algoritmos de *muestreo estratificado* [4, 5, 37] difieren fundamentalmente de los de muestreo por importancia en la forma de generar las muestras.

La otra clase de algoritmos por Monte Carlo es la conocida como *Monte Carlo por Cadenas de Markov* [31]. En este caso las muestras no son independientes, sino que tienen la propiedad de Markov, esto es, cada configuración obtenida depende solamente de la configuración inmediatamente anterior. Un ejemplo clásico de esta técnica es la simulación estocástica de Pearl [68]. Un enfoque generalizado a la simulación estocástica es el propuesto por Jensen, Kong y Kjærulff [42]. En este caso la idea radica en obtener muestras con un mayor grado de independencia, a costa de aumentar el costo computacional.

Además de los métodos de simulación, se han propuesto otras formas de algoritmos aproximados de propagación, que se basan en la idea de simplificar el problema en dos sentidos:

1. Kjærulff [50, 51] propone *simplificar la estructura de la red bayesiana original* mediante la eliminación de arcos que se correspondan con dependencias débiles, de manera que el árbol de cliques resultante sea más manejable para los algoritmos de agrupamiento.

2. La otra idea consiste en *simplificar las tablas de probabilidad*. En este sentido, Jensen y Andersen [43] proponen simplificar las tablas de probabilidad eliminando aquellos valores que sean menores que cierto umbral ϵ considerándolos iguales a cero, de forma que se puede adoptar una representación más compacta de las tablas de probabilidad. Mucho más sofisticada es la metodología de Cano y Moral [9], donde estudian el uso de árboles de probabilidad para representar las distribuciones, y definen una serie de aproximaciones de esos árboles que permiten reducir drásticamente el tamaño de los potenciales. En el capítulo 5 de ésta memoria hacemos uso de estas técnicas, pero combinadas con algoritmos de simulación por importancia.

Veremos ahora en detalle los principales métodos de propagación hacia delante y por cadenas de Markov.

2.4.1 Métodos de propagación hacia delante

La idea de las técnicas de propagación hacia delante consiste en elegir un orden ancestral[4] de las variables de la red y obtener una configuración para cada variable en secuencia, muestreando según la distribución condicionada de dicha variable dados sus padres en la red. A cada configuración de las variables obtenida se le asigna un peso que, al final del proceso de simulación, y normalizando, resulta en la probabilidad a posteiori de cada variable. El primer método de este tipo es el conocido como *muestreo lógico probabilístico* debido a Henrion [33], y destaca el hecho de que todos los pesos valen 0 ó 1, dependiendo de que la configuración obtenida sea coherente con

[4]Un orden de los nodos de un grafo se dice *ancestral* si cada nodo tiene una posición en dicho orden anterior a cualquier descendiente suyo.

las observaciones o no. El algoritmo es el siguiente, para una red correspondiente a n variables $X_1, \ldots, X_n$:

Muestreo Lógico

---

1. Sea σ una permutación del conjunto $\{1, \ldots, n\}$, tal que $\{X_{\sigma(1)}, \ldots, X_{\sigma(n)}\}$ es un orden ancestral de las variables de la red.
2. Desde $i = 1$ hasta n, hacer $h_{\sigma(i)}(x) = 0$ para todo $x \in U_{\sigma(i)}$.
3. Desde $j = 1$ hasta m,
   (a) Desde $i = 1$ hasta n,
      i. Obtener un valor $x_{\sigma(i)} \in U_{\sigma(i)}$ simulando de acuerdo a la distribución $f_{\sigma(i)}$.
      ii. Si $X_{\sigma(i)}$ es una variable observada y $x_{\sigma(i)} \neq e_{\sigma(i)}$, volver al paso 3.
   (b) Desde $i = 1$ hasta n,
      i. $h_{\sigma(i)}(x_{\sigma(i)}) = h_{\sigma(i)}(x_{\sigma(i)}) + 1$.
4. Normalizar los $h_{\sigma(i)}$, $i = 1, \ldots, n$. Cada $h_{\sigma(i)}$ resultante es la distribución a posteriori de la variable $X_{\sigma(i)}$.

---

donde m es el tamaño de la muestra.

Obsérvese que el problema de este algoritmo es que si la configuración obtenida no concuerda con las observaciones, la iteración no será válida (paso 3.(a).ii. del algoritmo). Este problema no se presenta si todas las observaciones se dan en nodos raíz, dado que en ese caso se puede instanciar cada variable al valor observado y no se

simulan. Entonces, la primera variable a simular sería la primera en el orden σ que no estuviera observada, y su distribución de probabilidad estaría restringida a los valores de las variables observadas, luego no se obtendrían configuraciones contradictorias con las observaciones.

Una propuesta para mejorar este sistema es el llamado método de *integración de evidencia*, desarrollado por Chin y Cooper [15, 16]. La idea consiste en invertir los arcos de la red que involucren a las variables observadas, de forma que después de la inversión los nodos observación queden como nodos raíz. La forma de invertir los arcos es análoga a la propuesta por Shachter para diagramas de influencia [77, 78]. El problema de este método es que el proceso de inversión de arcos puede ser muy costoso.

Un método más sofisticado es el de *ponderación de la evidencia*, ideado por Fung y Chang [30]. Difiere del muestreo lógico de Henrion en dos aspectos:

1. Sólo simula las variables no observadas, es decir, X_{N-E}. Las variables X_E toman directamente el valor observado sin simular.

2. En el muestreo lógico se asignaba a cada configuración simulada un peso constante igual a 1 (paso 3.(b).i. del algoritmo). En este caso, el peso w asignado coincide con la verosimilitud de la evidencia dada la configuración de las variables ya simuladas, es decir, si $\tilde{x}$ es la configuración de las variables ya simuladas,

$$P(e|\tilde{x}) = \prod_{i \in E} f_i(e^{\downarrow i}, \tilde{x}^{\downarrow F(i)}). \tag{2.12}$$

A este método se le llama también *ponderación por verosimilitud*. En general, podría decirse que este algoritmo se comporta bien, pero obsérvese que si la probabilidad expresada en la fórmula 2.12 es cero salvo para muy pocas configuraciones del resto de variables de la red, el algoritmo tiende a ser incapaz de dar ninguna estimación de las probabilidades a posteriori, dado que todos los pesos pueden resultar nulos.

2.4.2 Métodos de propagación por cadenas de Markov

En los métodos de propagación hacia delante, las muestras obtenidas en el proceso de simulación son independientes. En los que vamos a ver a continuación, cada configuración depende de la(s) configuración(es) simuladas con anterioridad.

El primer método dentro de este grupo es el de *simulación estocástica o directa* propuesto por Pearl [68]. Las diferencias más destacadas respecto al algoritmo de ponderación por verosimilitud son:

1. En este caso, las variables no han de simularse en ningún orden en especial.
2. En lugar de simular usando la distribución condicionada de cada variable, se usa una distribución de cada variable condicionada a su envolvente de Markov en la red[5].

El algoritmo detallado queda como sigue, para una red definida sobre las variables X_N con $N = \{1, \ldots, n\}$.

Simulación estocástica

---

1. Instanciar todos los nodos de la red a uno de sus posibles valores con probabilidad mayor que cero.
2. Para cada variable no observada X_i, $i \in \{1, \ldots, n\}$, hacer $h_i(x_i) = 0$ para todo $x_i \in U_i$.
3. Desde $j = 1$ hasta m
   (a) Para cada variable X_i, $i \in \{1, \ldots, n\}$,

[5]La *envolvente de Markov* de una variable en una red bayesiana es el conjunto de los padres, hijos y padres de los hijos de dicha variable.

i. Calcular $P(X_i|W_{X_i})$, donde W_{X_i} denota la envolvente de Markov de la variable X_i, de la siguiente manera:

$$P(X_i = x_i|W_{X_i} = w_{X_i}) = \alpha f_i(x_i, x_{F(i)}) \prod_{l \in H(i)} f_l(x_l, x_{F(l)}) \quad \forall x_i \in U_i, \tag{2.13}$$

donde α es una constante de normalización, $H(i)$ denota el conjunto de índices de los hijos de la variable X_i y w_{X_i} es la configuración actual de la envolvente de Markov de la variable X_i.

ii. Simular un valor $x_i^{(j)} \in U_i$ para X_i según la distribución $P(X_i|W_{X_i})$.

iii. Actualizar h_i según una de la dos siguientes expresiones:

$$\begin{aligned} h_i(x_i^{(j)}) &= h(x_i^{(j)}) + 1, \\ h_i(x_i^{(j)}) &= h(x_i^{(j)}) + P(X_i = \tilde{x}_i|W_{X_i} = w_{X_i}). \end{aligned}$$

4. Normalizar los h_i, $i = 1, \ldots, n$. Cada h_i resultante es la distribución a posteriori de la variable X_i.

---

donde m es el tamaño de la muestra.

Este método presenta dos problemas principales. Por un lado, puede ser difícil encontrar una configuración inicial para las variables de la red que tenga probabilidad positiva. Jensen, Kong y Kjærulff [42] proponen usar inicialmente una técnica de muestreo hacia delante para encontrar la configuración inicial.

Por otro lado, cada configuración depende de la generada inmediatamente antes (ver fórmula 2.13). Por eso, puede darse el caso de que, una vez alcanzada una configuración, ésta se repita un gran número de veces, debido a que las dependencias entre las variables sean "casi" funcionales, es decir, las distribuciones generadas en la fórmula 2.13 tengan valores muy próximos a 0 o a 1. La convergencia de este método hacia la distribución exacta está asegurada, cuando todas las probabilidades son estrictamente positivas, por

resultados de la teoría de los procesos de Markov [7, 27], pero ésta puede alcanzarse muy lentamente por la razón dicha anteriormente. En el caso de tener probabilidades nulas, puede que no se de la convergencia. El siguiente ejemplo puede aclarar la situación:

Ejemplo 2.1 *Sea una red bayesiana con dos variables binarias conectadas de la forma* $X_1 \rightarrow X_2$ *y tales que* $P(X_2 = x_2|X_1 = x_1) = P(X_2 = \bar{x}_2|X_1 = \bar{x}_1) = \delta \simeq 1$. *Supongamos que* $P(X_1 = x_1) = 0.5$ *y que* $X_1 = x_1$, *entonces* $P(X_2 = x_2|W_{X_2}) = P(X_2 = x_2|X_1 = x_1) = \delta$. *Si en una simulación obtenemos* $X_2 = x_2$, *en la próxima simulación la distribución usada para simular* X_1 *será*

$$\begin{aligned} P(X_1 = x_1|W_{X_1}) &= P(X_1 = x_1|X_2 = x_2) \\ &= \alpha P(X_2 = x_2|X_1 = x_1)P(X_1 = x_1) \\ &= \alpha \times 0.5 \times \delta = \delta, \end{aligned}$$

dado que, por la regla de Bayes, $\delta = 1/P(X_2 = x_2)$, *y*

$$\begin{aligned} P(X_2 = x_2) &= P(X_2 = x_2|X_1 = x_1)P(X_1 = x_1) + P(X_2 = x_2|X_1 = \bar{x}_1)P(X_1 = \bar{x}_1) \\ &= \delta \times 0.5 + (1 - \delta) \times 0.5 = 0.5. \end{aligned}$$

Si continuamos así, obtendremos la configuración $(X_1 = x_1, X_2 = x_2)$ *con probabilidad muy próxima a 1, y, en el momento en que una de las dos variables cambiara de valor, la otra también lo haría, repitiéndose entonces muchas veces la configuración* $(X_1 = \bar{x}_1, X_2 = \bar{x}_2)$. *Obsérvese, por lo tanto, que la configuración que se obtenga en una simulación puede depender fuertemente de la obtenida en la simulación anterior.*

Tratando de resolver este problema, surgió el denominado *muestreo de Gibbs por bloques*, desarrollado por Jensen, Kong y Kjærulff [42]. Estos autores se dan cuanta de que los problemas de la simulación estocástica se deben a la dependencia entre las muestras, en el sentido de que en cada momento, sólo se cambia el valor de una variable. Esto no ocurre en el muestreo hacia delante, en el que todas las variables pueden cambiar de valor en cada muestra.

El muestreo de Gibbs por bloques es un sofisticado método que se basa en buscar un compromiso entre dependencia entre la muestras y coste computacional, partiendo de los dos casos extremos:

1. Muestrear una sola variable dada su envolvente de Markov es computacionalmente simple, pero las muestras pueden ser muy dependientes.

2. Muestrear todas las variables a la vez hace que las muestras sean independientes, pero el coste computacional puede ser intratable.

El método consiste en dividir las variables de la red en una serie de grupos de forma que todas las variables en un mismo grupo se simulan a la vez. Cuanto más grande sea cada grupo, menor será la dependencia entre las muestras, pero mayor será la complejidad de calcular la distribución conjunta que ha de usarse para simular las variables del grupo a la vez.

Capítulo 3

Algoritmos de Muestreo por Importancia

3.1 Introducción

En el capítulo anterior hemos estudiado una serie de métodos para propagar probabilidades en redes bayesianas de forma aproximada, examinando los algoritmos más destacados basados en propagación hacia delante y en cadenas de Markov.

En este capítulo proponemos una clase general de algoritmos basados en muestreo por importancia, presentada en [36, 37]. La idea básica consiste en realizar una primera propagación aproximada, basada en el concepto de eliminación de nodos [84, 96], similar a la propagación simbólica de D'Ambrosio [21, 79], consistente en ir eliminando las variables de la red en secuencia y en cada eliminación, obtener una función de muestreo para la variable eliminada. A continuación, los resultados de esta propagación se mejoran muestreando las funciones obtenidas.

En la sección 3.2 se establece la notación a seguir durante el capítulo y se enuncia el problema general. En la sección 3.3 se presentan los fundamentos del muestreo por importancia, y en la sección 3.4 se revisan los métodos de muestreo por importancia conocidos. Los nuevos algoritmos son estudiados en detalle, considerando diversas

variaciones sobre el esquema principal, en la sección 3.5, mientras que la evaluación experimental de los mismos se presenta en la sección 3.6, finalizando con las conclusiones en la sección 3.7.

3.2 Notación y Formulación del Problema

Supondremos durante este capítulo una red bayesiana definida sobre un conjunto de variables $X = \{X_1, \ldots, X_n\}$, cada una de ellas tomando valores en un conjunto finito U_i, $i = 1, \ldots, n$ y $N = \{1, \ldots, n\}$.

El objetivo que nos proponemos es calcular la distribución 'a posteriori' $p(x_k|e)$ para todo $x \in U_k$, correspondiente a cada variable X_k con $k \in N$. Esta probabilidad podría obtenerse a partir de la distribución conjunta (2.3), pero suponemos que ésta es difícil de manejar. Obsérvese que la probabilidad que queremos calcular es igual a $p(x_k, e)/p(e)$, y, dado que $p(e)$ es constante, ésta es proporcional a $p(x_k, e)$. Por lo tanto, podemos obtener la distribución 'a posteriori' si calculamos para cada $x_k \in U_k$ el valor $p(x_k, e)$ y normalizamos después. Podemos expresar $p(x_k, e)$ como sigue,

$$\begin{aligned} S = p(x_k, e) &= \sum_{\substack{x \in U_N \\ x^{\downarrow E} = e \\ x^{\downarrow k} = x_k}} \prod_{i \in N} f_i(x^{\downarrow s(f_i)}) \\ &= \sum_{x \in U_N} \left(\prod_{i \in N} f_i(x^{\downarrow s(f_i)}) \right) \left(\prod_{j \in E} \delta_{e_j}(x^{\downarrow j}) \right) \delta_{x_k}(x^{\downarrow k}) \\ &= \sum_{x \in U_N} f(x), \end{aligned} \tag{3.1}$$

donde

$$f(x) = \left(\prod_{i \in N} f_i(x^{\downarrow s(f_i)}) \right) \left(\prod_{j \in E} \delta_{e_j}(x^{\downarrow j}) \right) \delta_{x_k}(x^{\downarrow k}) \quad \forall x \in U_N$$

y

$$\delta_{x_k}(x^{\downarrow k}) = \begin{cases} 1 & \text{si} \quad x^{\downarrow k} = x_k, \\ 0 & \text{si} \quad x^{\downarrow k} \neq x_k. \end{cases}$$

Nótese que la fórmula (3.1) es también válida si se sustituye cada potencial f_i por su restricción a la evidencia e, de forma que se pueda trabajar con funciones más sencillas.

Una forma de estimar la suma en la ecuación (3.1) es mediante la técnica de muestreo por importancia.

3.3 Muestreo por Importancia

El *muestreo por importancia* surgió como una técnica de reducción de la varianza en cálculo de integrales por Monte Carlo [72]. En esta sección veremos la formulación general del muestreo por importancia y en la siguiente se estudiará su aplicación a la inferencia en redes bayesianas.

Supongamos que estamos interesados en estimar la siguiente integral,

$$I = \int g(x)dx, \quad x \in D \subseteq I\!R^n. \tag{3.2}$$

La idea básica de esta técnica consiste en concentrar la distribución de los puntos de muestreo en aquellas zonas de la región D que son más "importantes", en lugar de muestrear aleatoriamente en toda la región D. La integral (3.2) podemos expresarla como

$$I = \int \frac{g(x)}{f^*(x)} f^*(x)dx = E\left[\frac{g(X)}{f^*(X)}\right], \tag{3.3}$$

donde X es un vector aleatorio con función de densidad $f^*(x)$ tal que $f^*(x) > 0$ para

todo $x \in D \subseteq \mathbb{R}^n$. A la función $f^*(x)$ se le llama *distribución del muestreo por importancia*. Es inmediato ver que $\xi = \frac{g(X)}{f^*(X)}$ es un estimador insesgado[1] de I, con varianza

$$var(\xi) = E[\xi^2] - E[\xi]^2 = \int \frac{g^2(x)}{f^*(x)} dx - I^2. \tag{3.4}$$

De cara a obtener una estimación de la integral I, podemos tomar una muestra $\{x^{(i)}\}$, $i \in \{1, \ldots, m\}$ de valores del vector X a partir de la función $f^*(x)$. Entonces, podemos estimar la esperanza del estimador ξ, en definitiva, estimar el valor I, calculando la media muestral de los valores obtenidos,

$$\hat{I} = \frac{1}{m} \sum_{i=1}^{m} \frac{g(x^{(i)})}{f^*(x^{(i)})}. \tag{3.5}$$

Los siguientes resultados muestran cómo elegir la función del muestreo por importancia de forma que la varianza del estimador sea mínima.

Teorema 3.1 [72] *La mínima varianza del estimador ξ es igual a*

$$var(\xi_0) = \left(\int |g(x)| dx \right)^2 - I^2, \tag{3.6}$$

y se obtiene cuando la variable aleatoria X sigue la distribución

$$f^*(x) = \frac{|g(x)|}{\int |g(x)| dx}. \tag{3.7}$$

Corolario 3.1 [72] *Si $g(x) > 0$, la función óptima para el muestreo por importancia es $f^*(x) = g(x)/I$, y $var(\xi) = 0$.*

[1] Un estimador se dice *insesgado* si su esperanza coincide con el parámetro a estimar

Aunque este corolario nos dice cuál es la distribución óptima para el muestreo por importancia, en la práctica no será posible obtenerla, pues suponemos que queremos estimar la integral por Monte Carlo porque no es posible hacerlo por métodos exactos o numéricos, y para obtener la función $f^*(x)$ necesitamos el valor de I, que es precisamente lo que no conocemos. Sin embargo, la varianza se reduce sustancialmente si tomamos como distribución de muestreo una función que sea tan próxima a $g(x)$ como podamos, pero más sencilla de muestrear que ésta.

En la siguiente sección explicamos cómo aplicar la técnica de muestreo por importancia a la estimación de las probabilidades a posteriori en una red causal.

3.3.1 Muestreo por importancia en redes bayesianas

Haciendo un desarrollo análogo al de la sección anterior, obtenemos una forma de estimar la suma de la ecuación (3.1). Para ello, consideramos una distribución de probabilidad $f^* : U_N \to [0, 1]$, verificando que si $f(x) > 0$ entonces $f^*(x) > 0$. Entonces, podemos escribir la fórmula (3.1) como sigue:

$$S = \sum_{x \in U_N} \frac{f(x)}{f^*(x)} f^*(x) = E\left[\frac{f(X)}{f^*(X)}\right],$$

donde X es una v.a. con distribución f^*. Entonces,

$$\xi = \frac{f(X)}{f^*(X)}$$

es un estimador insesgado de $S = p(x_k, e)$ con varianza

$$var(\xi) = E[\xi^2] - E[\xi]^2 = \left(\sum_{x \in U_N} \frac{f^2(x)}{f^*(x)}\right) - p^2(x_k, e). \tag{3.8}$$

Como la media muestral es un estimador insesgado de la esperanza, podemos estimar $S = p(x_k, e)$ a partir de una muestra $\{x^{(i)}\}$, $i \in \{1, \ldots, m\}$ de la variable X como

$$\hat{S} = \frac{1}{m} \sum_{i=1}^{m} \frac{f(x^{(i)})}{f^*(x^{(i)})}. \tag{3.9}$$

Teorema 3.2 *La mínima varianza de ξ es igual a cero, y se obtiene cuando*

$$f^*(x) = \frac{f(x)}{\sum_{y \in U_N} f(y)} = \frac{f(x)}{p(x_k, e)} \quad \forall x \in U_N.$$

Dem: Sustituyendo $f^*(x)$ en la fórmula (3.8),

$$\begin{aligned} var(\xi) &= \left(\sum_{x \in U_N} \frac{f^2(x) \cdot p(x_k, e)}{f(x)} \right) - p^2(x_k, e) \\ &= p(x_k, e) \left(\sum_{x \in U_N} f(x) \right) - p^2(x_k, e) \\ &= p^2(x_k, e) - p^2(x_k, e) \\ &= 0. \end{aligned}$$

□

Al igual que en el caso de integrales, no será posible, en general, utilizar una distribución de muestreo proporcional a $f(x)$. Lo mejor que podremos hacer es elegir una tan próxima a $f(x)$ como sea posible, con objeto de reducir la varianza de los pesos.

El inconveniente del método desarrollado aquí es que hay que aplicarlo a cada valor de cada una de las variables de la red. Es decir, para cada valor de cada variable de la red, hay que calcular una distribución de muestreo y obtener una muestra para estimar su valor de probabilidad. Ésta es la solución adoptada en [12, 13]. Esto puede ser muy costoso si el tamaño de la red es suficientemente grande.

En este capítulo proponemos un método que se basa en una variación del desarrollo anterior, y que permite estimar de forma más rápida la probabilidad de cada uno de los valores de todas las variables de la red.

La idea consiste en realizar una simulación como si fuéramos a estimar $p(e)$. En este caso,

$$\begin{aligned} p(e) &= \sum_{\substack{x \in U_N \\ x^{\downarrow E} = e}} \prod_{i \in N} f_i(x^{\downarrow s(f_i)}) \\ &= \sum_{x \in U_N} \left(\prod_{i \in N} f_i(x^{\downarrow s(f_i)}) \right) \left(\prod_{j \in E} \delta_{e_j}(x^{\downarrow j}) \right) \\ &= \sum_{x \in U_N} f(x), \end{aligned} \tag{3.10}$$

con

$$f(x) = \left(\prod_{i \in N} f_i(x^{\downarrow s(f_i)}) \right) \left(\prod_{j \in E} \delta_{e_j}(x^{\downarrow j}) \right) \quad \forall x \in U_N.$$

Podemos construir un estimador ξ de la probabilidad $p(e)$ de la misma forma que lo hicimos antes. La varianza del estimador, en este caso, será

$$var(\xi) = \left(\sum_{x \in U_N} \frac{f^2(x)}{f^*(x)} \right) - p^2(e). \tag{3.11}$$

Utilizando la distribución f^*, se genera una muestra $x^{(i)} \in U_N$, $i = 1, \ldots, m$. Esta única muestra se usa para estimar todas las probabilidades de la siguiente forma. Para cada valor x_k de cada variable X_k con $k \in N$, se toman todas las configuraciones $x^{(j)}$, $j \in J \subseteq \{1, \ldots, m\}$ tales que $x^{(j)^{\downarrow k}} = x_k$ y se estima la probabilidad $p(x_k, e)$ como

$$\hat{p}(x_k, e) = \frac{1}{m} \sum_{j \in J} \frac{f(x^{(j)})}{f^*(x^{(j)})} = \frac{1}{m} \sum_{j=1}^{m} w_j. \tag{3.12}$$

Es decir, se estima la probabilidad de cada valor x_k como la media de los pesos de las configuraciones que forman la muestra, siendo el *peso de una configuración* $x^{(j)}$,

$$w_j = \begin{cases} \dfrac{f(x^{(j)})}{f^*(x^{(j)})} = \dfrac{\left(\prod_{i\in N} f_i(x^{(j)\downarrow s(f_i)})\right)\cdot\left(\prod_{l\in E}\delta_{e_l}(x^{(j)\downarrow l})\right)}{f^*(x^{(j)})} & \text{si } j\in J \\ 0 & \text{si } j\notin J \end{cases} \tag{3.13}$$

Obsérvese que lo que en realidad se hace es usar un estimador insesgado de $p(x_k, e)$,

$$\xi' = \frac{f(X)\delta_{x_k}(X)}{f^*(X)}, \tag{3.14}$$

con X una v.a. con f.m.p. f^*, pero en este caso, la función de muestreo f^* está construida para estimar $p(e)$ en lugar de $p(x_k, e)$. Por lo tanto, no podemos esperar que la varianza, aun en el caso de elegir f^* proporcional a f, llegue a alcanzar el valor cero. Veamos:

$$\begin{aligned} var(\xi') &= \sum_{x\in U_N}\frac{f^2(x)\delta^2_{x_k}(x^{\downarrow k})p(e)}{f(x)} - p^2(x_k,e) \\ &= p(e)\sum_{x\in U_N} f(x)\delta_{x_k}(x^{\downarrow k}) - p^2(x_k,e) \\ &= p(e)p(x_k,e) - p^2(x_k,e) \\ &= p(x_k,e)(p(e)-p(x_k,e)), \end{aligned}$$

donde se ha utilizado que $\delta^2_{x_k}(x^{\downarrow k}) = \delta_{x_k}(x^{\downarrow k})$.

De aquí y de (3.12) se sigue que

$$\begin{aligned} var(\hat{p}(x_k,e)) &= var\left(\frac{1}{m}\sum_{j=1}^{m} w_j\right) \\ &= \frac{1}{m^2}\sum_{j=1}^{m} var\,(w_j) \end{aligned}$$

$$= \frac{1}{m^2} m p(x_k, e)(p(e) - p(x_k, e))$$
$$= \frac{p(x_k, e)(p(e) - p(x_k, e))}{m}.$$

Sin embargo, no siempre es posible elegir una distribución de muestreo proporcional a f en una red bayesiana, pues suponemos que ésta es difícil de manejar, e incluso puede darse el caso de que no sea posible almacenar tal distribución en un ordenador porque su tamaño sea demasiado grande. Por lo tanto, lo mejor que podemos hacer es elegir f^* lo más próxima posible a $p(\cdot|e)$.

Una vez hemos elegido f^*, podemos estimar $p(x_k, e)$, para toda modalidad x_k de toda variable X_k, $k \in N$, como sigue (ver [13, 80] para más detalles):

Muestreo por importancia

1. Desde $i = 1$ hasta m
   (a) Generar una configuración $x^{(i)}$ a partir de f^*.
   (b) Calcular
   $$w_i = \frac{\left(\prod_{j \in N} f_j(x^{(i)\downarrow s(f_j)})\right) \cdot \left(\prod_{j \in E} \delta_{e_j}(x^{(i)\downarrow j})\right)}{f^*(x^{(i)})}. \tag{3.15}$$
2. Para cada $x_k \in U_k$, $k = \{1, \ldots, n\}$,
   (a) Estimar $p(x_k, e)$ usando la fórmula (3.12).
3. Normalizar los valores $p(x_k, e)$ para obtener $p(x_k|e)$.

donde m es el tamaño de la muestra.

3.4 Algoritmos Conocidos de Muestreo por Importancia

En esta sección describiremos los principales algoritmos de muestreo por importancia en redes bayesianas, debidos a Shachter y Peot [80] y Cano, Hernández y Moral [12, 13].

3.4.1 Algoritmo de muestreo por importancia de Shachter y Peot

Shachter y Peot fueron los primeros en relacionar el muestreo por importancia y la propagación en redes bayesianas [80]. Propusieron un algoritmo similar al enunciado en la sección anterior, y vieron cómo los métodos de muestreo lógico y ponderación de la evidencia, eran casos particulares del muestreo por importancia. Sin embargo, el algoritmo de Shachter y Peot sigue siendo de propagación hacia delante, dado que las distribuciones de muestreo, en la mayoría de los casos siguen siendo las condicionadas.

3.4.2 Algoritmo de muestreo por importancia de Cano, Hernández y Moral

Estos autores [12, 13, 35] proponen un método de propagación usando muestreo por importancia más sofisticado que el de Shachter y Peot, en el sentido de que las funciones de muestreo se seleccionan de una forma más elaborada, no simplemente tomando las condicionadas. Además, en cada paso de simulación se muestrea en más de una variable a la vez.

El proceso general es el siguiente. Partimos de una red representando a un conjunto de variables $\{X_i | i \in N\}$. Se considera el conjunto de funciones $H = \{h_j | j = 1, \ldots, k\}$, que se obtiene a partir de las distribuciones condicionadas de las variables de la red, f_i, $i \in N$, y las deltas de las observaciones, δ_{e_l}, $e_l \in U_l$, $l \in E$. Luego se elige un

orden de las funciones h y se van simulando valores para las variables para las que están definidas dichas funciones. Cada vez que se obtienen valores para unas variables a partir de una función, el resto de las funciones en H se restringen a dichos valores. El algoritmo detallado para obtener una estimación de la probabilidad $p(x_I, e)$ donde I es el conjunto de índices de las variables de interés y $e \in U_E$ es el valor de las variables observadas, es el siguiente.

Muestreo por importancia de Cano et al.

---

1. Definir el conjunto $H = \{h_j | j = 1, \ldots, k\}$ a partir de las distribuciones f_i y δ_{e_l}.
2. Hacer $\xi = 0.0$.
3. Fijar los valores x_I para los cuales se quiere estimar $p(x_I, e)$.
4. Restringir todas las funciones h_j al valor x_I.
5. Desde $i = 1$ hasta m,
   (a) Hacer $\xi_i = 1.0$.
   (b) Mientras haya variables sin simular,
      i. Elegir una función h_j de H.
      ii. Hacer $H = H - \{h_j\}$.
      iii. Hacer $\xi_i = \xi_i \cdot q(h_j)$.
      iv. Simular un valor x_{J_j} según la distribución
      $$\frac{\sum_{s(h_j)-J_j} h_j(x^{\downarrow s(h_j)})}{q(h_j)} = \frac{h_j^*(x^{\downarrow J_j})}{q(h_j)}.$$
      v. Hacer $\xi_i = \dfrac{\xi_i}{h_j^*(x^{\downarrow J_j})}$.
      vi. Restringir todas las funciones $h \in H$ al valor $x^{\downarrow J_j}$.

(c) Desde $l = 1$ hasta n, hacer $\xi_i = \xi_i \cdot f'_l(x^{\downarrow s(f'_i)})$.

(d) Hacer $\xi = \xi + \xi_i$.

6. Devolver ξ/m como la estimación de $p(x_I, e)$.

---

donde m es el tamaño de la muestra, $q(h_j)$ es la suma en todo el dominio de la función h_j,

$$q(h_j) = \sum_{x \in U_{s(h_j)}} h_j(x)$$

y

$$f'_i = \begin{cases} f_i & \text{si} \quad i \notin E \\ \\ \delta_{e_i} & \text{si} \quad i \in E \end{cases} \qquad i = 1, \ldots, n.$$

La eficiencia de este algoritmo depende de la forma en que se obtengan las funciones de selección h_j. Cano, Hernández y Moral [12, 13], proponen elegir primero aquellas que tengan menos entropía.

3.5 Muestreo por Importancia basado en Precomputación Aproximada

La decisión más importante a la hora de diseñar un algoritmo de muestreo por importancia es la elección de la distribución de muestreo: ésta debería ser tan similar a la distribución original como sea posible. En el caso particular de una red causal, la distribución original viene dada como el producto de una serie de distribuciones condicionadas y un conjunto de observaciones. Los algoritmos conocidos de muestreo por importancia [13, 30, 80] usan las funciones originales (distribuciones condicionadas u observaciones) para aproximar la distribución producto.

Aquí proponemos un nuevo enfoque para obtener las distribuciones de muestreo. La idea es usar no sólo las condicionadas y las observaciones originales, sino toda la información concerniente a cada variable. Esto es, a la hora de simular valores para una variable, usar todas las funciones de las que disponemos. Éste es el caso ideal, pero si la red es suficientemente complicada, este proceso puede ser inviable; en concreto, la complejidad de este procedimiento sería la misma que la de la propagación exacta, y eso es precisamente lo que queremos evitar. En resumen, el problema es que el coste de la combinación de todas las funciones definidas para una variable puede ser demasiado alto.

La solución que proponemos es hacer cálculo aproximado cuando el exacto sea demasiado difícil. Dado que no usamos la distribución original ($p(x|e)$) sino una aproximación (f^*), utilizamos la técnica de muestreo por importancia y obtenemos un peso para cada configuración de las variables. El punto clave aquí es cuándo y cómo hacer cálculos aproximados de tal forma que los cálculos sean rápidos y la distribución de muestreo f^* sea próxima a $p(x|e)$.

El algoritmo que proponemos comienza con una familia de funciones formada por las distribuciones condicionadas dadas en la red más las observaciones, que vienen codificadas por funciones delta de Dirac,

$$H = \{f_1, \ldots, f_n\} \cup \{\delta_{e_l}\}_{l \in E}. \tag{3.16}$$

La verdadera $p(\cdot|e)$ es proporcional al producto de todas las funciones en H (ver ecuación (3.1)).

A continuación, consideramos un orden de las variables de la red, dado por una permutación σ sobre el conjunto $\{1, \ldots, n\}$. El siguiente paso es *eliminar* las variables en secuencia, siguiendo el orden impuesto por σ. A saber, hay dos formas de realizar la *eliminación* de una variable $X_{\sigma(i)}$: exacta y aproximada.

Exacta

1. Combinar todas las funciones que están definidas para la variable $X_{\sigma(i)}$, obteniendo como resultado una función h_i.
2. Eliminar $X_{\sigma(i)}$ de la combinación, h_i, marginalizando el resultado a $s(h_i)-\{\sigma(i)\}$, es decir, calcular $h^* = h_i^{\downarrow s(h_i)-\{\sigma(i)\}}$.
3. Añadir el resultado de la marginalización, h^*, a H.
4. Eliminar de H todas las funciones que se combinaron para obtener h_i.

Si es posible repetir este proceso para todas las variables, en cada paso se obtiene una distribución de muestreo proporcional a $p(x, e)$. En realidad, el proceso es como un algoritmo de propagación exacta [84], y se verifica el siguiente teorema:

Teorema 3.3 *Supongamos que hemos realizado una eliminación exacta; entonces*

- *Si h_n es la función obtenida al eliminar $X_{\sigma(n)}$ entonces, para todo $x \in U_{\sigma(n)}$, $h_n(x)$ es proporcional a $p(x|e)$.*
- *Si h_i es la función obtenida al eliminar $X_{\sigma(i)}$ $(i < n)$, $\Sigma(i) = \{\sigma(i+1), \ldots, \sigma(n)\}$, y $x_0 \in U_{\Sigma(i)\cap s(h_i)}$, entonces, la restricción de h_i a x_0, h_i' (ver ecuación (2.7)) es proporcional a la probabilidad $p(.|e, x_0)$: $\forall x \in U_{\sigma(i)}, h_i'(x) \propto p(x|e, x_0)$.*

Dem: En primer lugar vamos a demostrar que si H es el conjunto de las funciones obtenidas antes de eliminar $X_{\sigma(i)}$ $(i = 1, \ldots, n)$, entonces

$$\forall x \in U_{\{\sigma(i)\}\cup\Sigma(i)}, \quad p(x,e) = \prod_{h\in H} h(x^{\downarrow s(h)}). \tag{3.17}$$

Esto es trivialmente cierto para $i = 1$.

A partir de (3.1) obtenemos

$$\begin{aligned} \forall x \in U_{\{1,\ldots,n\}}, \quad p(x,e) &= \prod_{i=1}^{n} f_i(x^{\downarrow s(f_i)}) \prod_{j\in E} \delta_{e_j}(x^{\downarrow j}) \\ &= \prod_{h\in H} h(x^{\downarrow s(h)}). \end{aligned}$$

Ahora, si H es el conjunto de funciones obtenidas antes de eliminar $X_{\sigma(k)}$ y H' es el mismo conjunto después de la eliminación, vamos a demostrar que si (3.17) es cierta para $i = k$ y H, entonces es también cierta para $i = k + 1$ y H'.

Si (3.17) es cierta para $i = k$ entonces

$$\forall x \in U_{\{\sigma(k)\}\cup\Sigma(k)}, \quad p(x,e) = \prod_{h\in H} h(x^{\downarrow s(h)}).$$

Si $y \in U_{\{\sigma(k+1)\}\cup\Sigma(k+1)} = U_{\Sigma(k)}$, entonces

$$p(y,e) = \sum_{x^{\downarrow\Sigma(k)}=y} p(x,e) = \sum_{x^{\downarrow\Sigma(k)}=y} \left[\prod_{h\in H} h(x^{\downarrow s(h)}) \right].$$

Sea $H(k) = \{h \in H | \sigma(k) \in s(h)\}$, entonces

$$p(y,e) = \sum_{x^{\downarrow\Sigma(k)}=y} \left(\prod_{h\in H(k)} h(x^{\downarrow s(h)}) \right) \cdot \left(\prod_{h\in H-H(k)} h(x^{\downarrow s(h)}) \right).$$

Ahora, si $h \in H - H(k)$, $x^{\downarrow s(h)} = y^{\downarrow s(h)}$ y $h(x^{\downarrow s(h)}) = h(y^{\downarrow s(h)})$ sin depender este valor de x, para un y fijo. Entonces

$$p(y,e) = \left[\prod_{h\in H-H(k)} h(y^{\downarrow s(h)})\right] \cdot \left[\sum_{x^{\downarrow\Sigma(k)}=y} \prod_{h\in H(k)} h(x^{\downarrow s(h)})\right].$$

Teniendo en cuenta que $H' = [H - H(k)] \cup \{h_k^{\downarrow\Sigma(k)}\}$ y que

$$h_k^{\downarrow\Sigma(k)}(y) = \sum_{x^{\downarrow\Sigma(k)}=y} \left(\prod_{h\in H(k)} h(x^{\downarrow s(h)})\right),$$

entonces,

$$p(y,e) = \left[\prod_{h\in H-H(k)} h(y^{\downarrow s(h)})\right] \cdot h_k^{\downarrow\Sigma(k)}(y).$$

Así, obtenemos el resultado deseado:

$$\forall y \in U_{\{\sigma(k+1)\}\cup\Sigma(k+1)}, \quad p(y,e) = \prod_{h\in H'} h(y^{\downarrow s(h)}).$$

Ahora, podemos utilizar la fórmula (3.17) para demostrar el teorema para $i = 1,\ldots,n$. Tenemos que,

$$\begin{aligned}
\forall x \in U_{\sigma(i)\cup\Sigma(i)}, \quad p(x,e) &= \prod_{h\in H} h(x^{\downarrow s(h)}) \\
&= \left[\prod_{h\in H(i)} h(x^{\downarrow s(h)})\right] \cdot \left[\prod_{h\in H-H(i)} h(x^{\downarrow s(h)})\right] \\
&= h_i(x^{\downarrow s(h_i)}) \prod_{h\in H-H(i)} h(x^{\downarrow s(h)}).
\end{aligned}$$

Sea $x = (y, x_0, z)$, donde $y \in U_{\{\sigma(i)\}}$, $z \in U_{\Sigma(i)-s(h_i)}$, $x_0 \in U_{\Sigma(i)\cap s(h_i)}$, x_0 constante. Entonces,

$$p(x,e) = p(y,x_0,z,e) = h_i((y,x_0)^{\downarrow s(h_i)}) \prod_{h\in H-H(i)} h((x_0,z)^{\downarrow s(h)}).$$

Tomando la suma en $z \in U_{\Sigma(i)-s(h_i)}$, tenemos que

$$\begin{aligned} p(y, x_0, e) &= \sum_{z \in U_{\Sigma(i)-s(h_i)}} p(y, x_0, z, e) \\ &= h_i((y, x_0)^{\downarrow s(h_i)}) \left(\sum_{z \in U_{\Sigma(i)-s(h_i)}} \left[\prod_{h \in H-H(i)} h((x_0, z)^{\downarrow s(h)}) \right] \right). \end{aligned}$$

Para x_0 constante, el segundo factor es constante y por lo tanto,

$$p(y, x_0, e) = k \cdot h_i((y, x_0)^{\downarrow s(h_i)}) = k \cdot h'_i(y),$$

y a partir de esta expresión obtenemos el resultado buscado:

$$\forall y \in U_{\sigma(i)}, \quad p(y|x_0, e) = \frac{k}{p(x_0, e)} h'_i(y) = k' \cdot h'_i(y) \propto h'_i(y).$$

□

Las dos propiedades del teorema anterior nos permiten simular un valor $x \in U_N$ con probabilidad igual a $p(x|e)$. Lo que tenemos que hacer es simular valores para las variables en el orden $X_{\sigma(n)}, \ldots, X_{\sigma(1)}$. Para obtener un valor para una variable $X_{\sigma(i)}$, muestreamos a partir de la función h_i, realizando primero la restricción de esta función a los valores x_0 obtenidos para las variables simuladas previamente ($X_{\Sigma(i)}$) y normalizando después.

En algunos casos, el tamaño[2] de h_i puede ser tan grande que su cálculo sea inviable. En este caso, la eliminación de las variables habrá de hacerse de forma aproximada. Pueden definirse numerosos criterios de aproximación, pero siempre dentro del siguiente esquema:

[2]Se define el *tamaño* de una función h como el producto del número de casos de todas las variables para las cuales h está definida.

Aproximado

1. Sea $H(i) = \{h \in H \mid \sigma(i) \in s(h)\}$, el conjunto de funciones definidas para la variable $X_{\sigma(i)}$. Eliminar $H(i)$ de H.

2. Transformar $H(i)$ mediante combinación. Para ello, repetir el siguiente proceso:

   (a) Tomar $R \subseteq H(i)$.

   (b) Combinar todas las funciones contenidas en R, es decir, calcular $f = \prod_{h \in R} h$.

   (c) Añadir el resultado de la combinación, f, a $H(i)$.

   (d) Eliminar R de $H(i)$.

3. Calcular $H^+(i)$ a partir de $H(i)$ eliminando $X_{\sigma(i)}$ en todas las funciones pertenecientes a $H(i)$.

4. Añadir $H^+(i)$ a H.

Este procedimiento coincide con el exacto si en el segundo paso se combinan todas las funciones contenidas en $H(i)$. La idea del procedimiento aproximado es combinar funciones mientras no se sobrepase cierto umbral de tamaño. Es decir, la forma de elegir los $R \subseteq H(i)$ dependerá del tamaño del resultado de combinar las funciones que lo formen. Una propiedad importante de esta aproximación de cara a su validez para obtener funciones del muestreo por importancia es que no se añaden nuevos ceros, es decir, si $x \in U_N$ es tal que $h(x^{\downarrow s(h)}) \neq 0$ para todo $h \in H$, antes de eliminar $X_{\sigma(i)}$, esta propiedad se verifica también después de eliminar la variable:

Lema 3.1 *Sean $H(i)$ y $H^+(i)$ como en el algoritmo aproximado. Sea $x \in U_N$. Se verifica que*

$$h(x^{\downarrow s(h)}) > 0 \quad \forall h \in H(i) \Rightarrow h(x^{\downarrow s(h)}) > 0 \quad \forall h \in H^+(i).$$

Dem: Si $h \in H^+(i)$, quiere decir que es el resultado de combinar algunas funciones de $H(i)$ y eliminar después la variable $X_{\sigma(i)}$, es decir,

$$h(x^{\downarrow s(h)}) = \sum_{x \in U_{\sigma(i)}} f(x^{\downarrow s(f)}) \quad \forall x \in U_N,$$

con

$$f(x^{\downarrow s(f)}) = \prod_{j \in J} h_j(x^{\downarrow s(h_j)}) \quad \forall x \in U_N,$$

donde $h_j \in H(i)$ para todo $j \in J$. Por lo tanto, si suponemos que $h_j(x^{\downarrow s(h_j)}) > 0$ para todo $h_j \in H(i)$, trivialmente $h(x^{\downarrow s(h)}) > 0$ por ser suma y producto de cantidades estrictamente positivas. □

El objetivo del algoritmo es obtener configuraciones de las variables X_N. El proceso para simular un valor para una variable $X_{\sigma(i)}$ según el algoritmo aproximado es el siguiente: si x_0 es la configuración obtenida para las variables $X_{\Sigma(i)}$, entonces

---

Simula($X_{\sigma(i)}$,$H(i)$)

---

1. Sea $H(i)$ el conjunto calculado en el paso 2 del procedimiento de eliminación aproximada.
2. Restringir cada función en $H(i)$ a x_0. Combinar todas las funciones en $H(i)$, obteniendo una nueva función h'_i definida sobre $U_{\sigma(i)}$.
3. Si $N(h'_i)$ es la normalización de h'_i, obtener un valor x_i para $X_{\sigma(i)}$ siguiendo la distribución de probabilidad $N(h'_i)$.

4. Devolver el valor x_i.

Ahora estamos en condiciones de formular un algoritmo general de muestreo por importancia para la propagación de probabilidades:

Algoritmo Principal (ALG MI)

1. Sea $H = \{f_i \mid i = 1, \ldots, n\}$.
2. Elegir un orden σ para las variables en G.
3. Incorporar las observaciones:
   (a) Restringir todas las funciones en H a las evidencias $\{e_l\}_{l \in E}$ de acuerdo con la ecuación (2.7).
   (b) Para cada variable observada X_l, $l \in E$ hacer $H = H \cup \{\delta_{e_l}\}$.
4. Desde $i = 1$ hasta n, eliminar $X_{\sigma(i)}$ mediante el procedimiento aproximado[3] y guardar el conjunto de potenciales $H(i)$.
5. Desde $j = 1$ hasta m,
   (a) $w_j = 1.0$.
   (b) Desde $i = n$ hasta 1,
      i. $x_i^{(j)}$=**Simula**($X_{\sigma(i)}$,$H(i)$).

[3]Obsérvese que el cálculo exacto es un caso particular de este paso, de manera que también es posible aquí realizar eliminación exacta.

ii. Sea $N(h'_i)$ como en el procedimiento **Simula**. Calcular

$$w_j = \frac{w_j}{N(h'_i)(x_i^{(j)})}.$$

(c) Calcular

$$w_j = w_j \cdot \left(\prod_{i=1}^{n} f_i(x^{(j)^{\downarrow s(f_i)}}) \right) \cdot \left(\prod_{l \in E} \delta_{e_l}(x^{(j)^{\downarrow l}}) \right).$$

6. Para cada $x_k \in U_k$, $k = \{1, \ldots, n\}$,

   (a) Estimar $p(x_k, e)$ usando la fórmula (3.12).

7. Normalizar los valores $p(x_k, e)$ para obtener $p(x_k|e)$.

Al final del ciclo 5.(b), el peso w_j toma el valor

$$w_j = 1/f^*(x^{(j)}),$$

donde f^* es la distribución de muestreo utilizada, que viene dada por:

$$f^*(x^{(j)}) = \prod_{i=1}^{n} N(h'_i)(x^{(j)^{\downarrow i}}). \tag{3.18}$$

Obsérvese que después del paso 5.(c) el peso resultante es el correcto para el muestreo por importancia, según la ecuación (3.15).

Nótese que la eficiencia del algoritmo depende del orden de las variables elegido en el paso 2, ya que según el orden pueden resultar funciones más o menos grandes. El problema de encontrar un orden de eliminación óptimo es similar al de encontrar una triangulación óptima del grafo moral asociado a una red bayesiana. Éste no es un problema trivial, y ha sido estudiado desde distintos enfoques por diversos autores destacando el uso de técnicas heurísticas por Cano y Moral [8], simulated annealing por

Kjærulff [49], algoritmos genéticos por Larrañaga y otros [56] y programas evolutivos por Hernández y Bolaños [34, 35].

A continuación veremos un ejemplo de cómo funciona el algoritmo de propagación por muestreo por importancia, comparando los resultados de la eliminación exacta y aproximada.

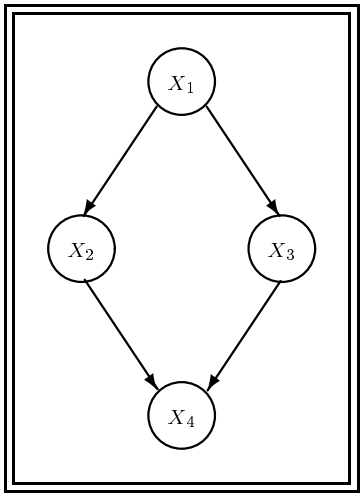

Figura 3.1: Una red causal.

Ejemplo 3.1 *Considérese la red de la figura 3.1, en la cual todas las variables toman los valores 0 ó 1. Supongamos que tenemos las siguientes tablas de probabilidad:*

1. *Probabilidad* a priori *para* X_1.

$$f_1(x_1) = P(X_1 = x_1).$$

$$f_1(0) = 0.3, \quad f_1(1) = 0.7.$$

2. *Probabilidad condicionada de* X_2 *dado* X_1.

$$f_2(x_2, x_1) = P(X_2 = x_2 | X_1 = x_1).$$

$$f_2(0,0) = 0.2, \quad f_2(1,0) = 0.8,$$

$$f_2(0,1) = 0.6, \quad f_2(1,1) = 0.4.$$

3. Probabilidad condicionada de X_3 dado X_1.

$$f_3(x_3, x_1) = P(X_3 = x_3 | X_1 = x_1).$$

$$f_3(0,0) = 0.1, \quad f_3(1,0) = 0.9,$$

$$f_3(0,1) = 0.2, \quad f_3(1,1) = 0.8.$$

4. Probabilidad condicionada de X_4 dado X_2, X_3.

$$f_4(x_4, x_2, x_3) = P(X_4 = x_4 | X_2 = x_2, X_3 = x_3).$$

$$f_4(0,0,0) = 0.3, \quad f_4(1,0,0) = 0.7, \quad f_4(0,1,0) = 0.2, \quad f_4(1,1,0) = 0.8,$$

$$f_4(0,0,1) = 0.5, \quad f_4(1,0,1) = 0.5, \quad f_4(0,1,1) = 0.4, \quad f_4(1,1,1) = 0.6.$$

Supongamos también que se ha observado que $X_4 = 1$, lo que representamos con el potencial

$$\delta_4(0) = 0 \quad , \quad \delta_4(1) = 1.$$

Tenemos que $H = \{f_1, f_2, f_3, f_4, \delta_4\}$.

Entonces, la eliminación exacta aplicada a H se realiza como sigue.

1. Eliminación de X_4. *Calcular $h_4(x_2, x_3, x_4) = f_4(x_4, x_2, x_3) \cdot \delta_4(x_4)$.*

$$h_4(0,0,0) = 0, \quad h_4(1,0,0) = 0, \quad h_4(0,1,0) = 0, \quad h_4(1,1,0) = 0,$$

$$h_4(0,0,1) = 0.7, \quad h_4(1,0,1) = 0.8, \quad h_4(0,1,1) = 0.5, \quad h_4(1,1,1) = 0.6.$$

Ahora, borramos f_4 y δ_4 de H y le añadimos $h_4^*(x_2, x_3) = h_4(x_2, x_3, x_4)^{\downarrow\{2,3\}}$,

$$h_4^*(0,0) = 0.7, \quad h_4^*(0,1) = 0.5,$$

$$h_4^*(1,0) = 0.8, \quad h_4^*(1,1) = 0.6.$$

Después de este paso, $H = \{f_1, f_2, f_3, h_4^*\}$.

2. Eliminación de X_3. *Calcular* $h_3(x_1, x_2, x_3) = f_3(x_3, x_1) \cdot h_4^*(x_2, x_3)$.

$$h_3(0,0,0) = 0.07, \quad h_3(1,0,0) = 0.14, \quad h_3(0,1,0) = 0.08, \quad h_3(1,1,0) = 0.16,$$

$$h_3(0,0,1) = 0.45, \quad h_3(1,0,1) = 0.4, \quad h_3(0,1,1) = 0.54, \quad h_3(1,1,1) = 0.48.$$

Eliminamos ahora f_3, h_4^* *de* H *y le añadimos* $h_3^*(x_1, x_2) = h_3(x_1, x_2, x_3)^{\downarrow\{1,2\}}$, *donde*

$$h_3^*(0,0) = 0.52, \quad h_3^*(0,1) = 0.62,$$

$$h_3^*(1,0) = 0.54, \quad h_3^*(1,1) = 0.64.$$

Después de este paso, $H = \{f_1, f_2, h_3^*\}$.

3. Eliminación de X_2. *Calcular* $h_2(x_1, x_2) = f_2(x_2, x_1) \cdot h_3^*(x_1, x_2)$.

$$h_2(0,0) = 0.104, \quad h_2(0,1) = 0.496,$$

$$h_2(1,0) = 0.324, \quad h_2(1,1) = 0.256.$$

Eliminar f_2, h_3^* *de* H *y añadir* $h_2^*(x_1) = h_2(x_1, x_2)^{\downarrow\{1\}}$, *donde*

$$h_2^*(0) = 0.6, \quad h_2^*(1) = 0.58.$$

Después de este paso, $H = \{f_1, h_2^*\}$.

4. Eliminación de X_1. *Calcular* $h_1(x_1) = f_1(x_1) \cdot h_2^*(x_1)$.

$$h_1(0) = 0.18, \quad h_1(1) = 0.406.$$

Borrar f_1, h_2^* *de* H *y añadir* $h_1^*(x_1) = h_1(x_1)^{\downarrow\emptyset} = 0.586$. *Obtenemos,* $H = \{h_1^*\}$.

Según el teorema 3.3, tenemos que

$$h_1'(x_i) = h_1(x_i) \propto P(X_1 = x_i|e).$$

Por lo tanto, si muestreamos un valor para X_1 *con una distribución de probabilidad* P_1^e, *proporcional a* h_1, *estamos muestreando con* $P(X_1 = x_i|e)$ *de forma exacta.*

$$P_1^e(X_1 = 0) = \frac{0.18}{0.586} = 0.31, \quad P_1^e(X_1 = 1) = \frac{0.406}{0.586} = 0.69.$$

Para un valor dado $X_1 = x_1^0$,

$$h_2'(x_i) = h_2(x_1^0, x_i) \propto P(X_2 = x_i|e, X_1 = x_1^0),$$

luego una distribución de muestreo exacta para X_2 *dado que* $X_1 = x_1^0$ *es*

$$P_2^e(X_2 = 0|X_1 = 0) = 0.17, \quad P_2^e(X_2 = 1|X_1 = 0) = 0.83,$$

$$P_2^e(X_2 = 0|X_1 = 1) = 0.56, \quad P_2^e(X_2 = 1|X_1 = 1) = 0.44.$$

Para dos valores $X_1 = x_1^0$, $X_2 = x_2^0$,

$$h_3'(x_i) = h_3(x_1^0, x_2^0, x_i) \propto P(X_3 = x_i|e, X_1 = x_1^0, X_2 = x_2^0).$$

Así, una distribución de muestreo exacta para X_3 *dado que* $X_1 = x_1^0$, $X_2 = x_2^0$ *es*

$$P_3^e(X_3 = 0|X_1 = 0, X_2 = 0) = 0.1346, \quad P_3^e(X_3 = 1|X_1 = 0, X_2 = 0) = 0.8654,$$

$$P_3^e(X_3 = 0|X_1 = 0, X_2 = 1) = 0.1290, \quad P_3^e(X_3 = 1|X_1 = 0, X_2 = 1) = 0.8710,$$

$$P_3^e(X_3 = 0|X_1 = 1, X_2 = 0) = 0.2593, \quad P_3^e(X_3 = 1|X_1 = 1, X_2 = 0) = 0.7407,$$

$$P_3^e(X_3 = 0|X_1 = 1, X_2 = 1) = 0.2500, \quad P_3^e(X_3 = 1|X_1 = 1, X_2 = 1) = 0.7500.$$

Finalmente, para tres valores $X_1 = x_1^0$, $X_2 = x_2^0$, $X_3 = x_3^0$,

$$\begin{aligned} h_4'(x_i) = h_4(x_2^0, x_3^0, x_i) &\propto P(X_4 = x_i | e, X_1 = x_1^0, X_2 = x_2^0, X_3 = x_3^0) \\ &= P(X_4 = x_i | e, X_2 = x_2^0, X_3 = x_3^0), \end{aligned}$$

y podemos calcular una distribución de muestreo exacta para X_4, *y ésta es igual a*

$$P_4^e(X_4 = 0 | X_1 = x_1^0, X_2 = x_2^0, X_3 = x_3^0) = P_4^e(X_4 = 0 | X_2 = x_2^0, X_3 = x_3^0) = 0,$$

$$P_4^e(X_4 = 1 | X_1 = x_1^0, X_2 = x_2^0, X_3 = x_3^0) = P_4^e(X_4 = 1 | X_2 = x_2^0, X_3 = x_3^0) = 1.$$

Imaginemos ahora que no todos los pasos de eliminación se han realizado de forma exacta. Por ejemplo, supongamos que la eliminación de X_4 *es igual que antes, pero que* X_3, X_2 *y* X_1 *se eliminan ahora de la siguiente forma,*

2′. Eliminación de X_3 *(Aproximada; marginalizamos sin haber combinado antes). Calcular* $f_3^* = f_3^{\downarrow\{1\}}$, $h_4^{**} = h_4^{*\downarrow\{2\}}$. *Borrar* f_3 *y* h_4^* *de* H *y añadir* f_3^*, h_4^{**}. *Nos queda* $H = \{f_1, f_2, f_3^*, h_4^{**}\}$, *donde*

$$f_3^*(0) = 1, \qquad f_3^*(1) = 1,$$

$$h_4^{**}(0) = 1.2, \quad h_4^{**}(1) = 1.1.$$

3′. Eliminación de X_2 *(Exacta). Calcular* $h_2^{**}(x_1, x_2) = f_2(x_2, x_1) \cdot h_4^{**}(x_2)$.

$$h_2^{**}(0,0) = 0.24, \quad h_2^{**}(1,0) = 0.72,$$

$$h_2^{**}(0,1) = 0.88, \quad h_2^{**}(1,1) = 0.44.$$

Borrar f_2, h_4^{**} *de* H *y añadir* $h_2^{***} = h_2^{**\downarrow\{1\}}$, *obteniéndose* $H = \{f_1, f_3^*, h_2^{***}\}$, *donde*

$$h_2^{***}(0) = 1.12, \quad h_2^{***}(1) = 1.16.$$

4′. Eliminación de X_1 *(Exacta). Calcular* $h_1^{**}(x_1) = f_1(x_1) \cdot f_3^*(x_1) \cdot h_2^{**}(x_1)$.

$$h_1^{**}(0) = 0.336, \quad h_1^{**}(1) = 0.812.$$

Calcular $h_1^{***} = h_1^{**\downarrow\emptyset} = 1.148$. *Al final,* $H = \{h_1^{***}\}$.

Ahora, las distribuciones de muestreo aproximadas pueden calcularse como sigue:

- $P_1^a(X_1 = x_i) \propto h_1^{**}(x_i)$, *es decir,*

$$P_1^a(X_1 = 0) = 0.2927, \quad P_1^a(X_1 = 1) = 0.7073.$$

- $P_2^a(X_2 = x_i | X_1 = x_1^0) \propto h_2^{**}(x_1^0, x_i)$. *Calculando,*

$$P_2^a(X_2 = 0|X_1 = 0) = 0.2143, \quad P_2^a(X_2 = 1|X_1 = 0) = 0.7857,$$

$$P_2^a(X_2 = 0|X_1 = 1) = 0.6207, \quad P_2^a(X_2 = 1|X_1 = 1) = 0.3793.$$

- $P_3^a(X_3 = x_i | X_1 = x_1^0, X_2 = x_2^0) \propto f_3(x_i, x_1^0) \cdot h_4^*(x_2^0, x_i)$. *Puede verificarse que* $P_3^a = P_3^e$.

- *De forma análoga,* $P_4^a = P_4^e$.

Este ejemplo muestra cómo es posible obtener distribuciones de muestreo aproximadas (P_i^a) *muy próximas a las exactas* (P_i^e), *aplicando un algoritmo de eliminación aproximado (y más rápido). Las distribuciones de muestreo tanto exactas como aproximadas para todas las configuraciones de las variables de la red se muestran en la tabla 3.1.*

3.5.1 Casos particulares del algoritmo principal

En esta sección estudiaremos distintas variaciones sobre el algoritmo principal. Los algoritmos particulares vendrán determinados por la forma en que se obtenga $H^+(i)$;

X_1	X_2	X_3	X_4	Exacta	Aproximada
0	0	0	0	0.0	0.0
0	0	0	1	0.007093	0.008443
0	0	1	0	0.0	0.0
0	0	1	1	0.04561	0.054283
0	1	0	0	0.0	0.0
0	1	0	1	0.033192	0.029667
0	1	1	0	0.0	0.0
0	1	1	1	0.224108	0.200308
1	0	0	0	0.0	0.0
1	0	0	1	0.100194	0.113838
1	0	1	0	0.0	0.0
1	0	1	1	0.286206	0.325183
1	1	0	0	0.0	0.0
1	1	0	1	0.0759	0.06707
1	1	1	0	0.0	0.0
1	1	1	1	0.2277	0.201209

Tabla 3.1: Funciones de muestreo para el ejemplo.

es decir, debemos definir criterios para seleccionar los subconjuntos R de $H(i)$ de las funciones que serán combinadas antes de marginalizar cuando una variable $X_{\Sigma(i)}$ va a ser eliminada.

El primer criterio en el que se podría pensar es tomar $R = H(i)$. Esto coincide con el algoritmo exacto de eliminación. Por lo tanto no tendremos en cuenta este criterio, dado que si hemos sido capaces de eliminar todas las variables de forma exacta, en una cantidad similar de tiempo podríamos calcular las probabilidades a posteriori sin necesidad de simular.

Una propiedad importante del algoritmo general que proponemos es que se puede

detectar cuándo es posible realizar propagación exacta, que sería precisamente cuando hemos eliminado todas las variables sin necesidad de aproximar. En este caso, el tiempo invertido no se pierde, y por el mismo proceso podríamos obtener la distribución a posteriori de todas las variables.

Como criterio opuesto al anterior, podríamos decidir no combinar ninguna función, es decir, $H(i)$ no se modifica en el paso 2 del esquema aproximado. Éste es el procedimiento más rápido que podemos considerar a la hora de obtener las funciones de muestreo, ya que no se invierte ningún tiempo en la combinación. sin embargo, el tiempo de simulación pude ser más alto si hay muchas funciones asociadas a cada variable. En este caso, a la hora de simular una variable, hay que evaluar todas las funciones que tiene asociadas para la configuración de las variables previamente simuladas. Además, las estimaciones obtenidas con este esquema deberían ser las peores, dado que se pierde información en cada eliminación (recuérdese que la forma correcta es primero combinar y luego marginalizar).

Estudiaremos ahora dos grupos de métodos para combinar las funciones.

3.5.1.1 Criterios de tamaño

En esta sección definimos tres criterios atendiendo al tamaño de las funciones que resultan de las combinaciones.

La primera idea consiste en combinar todas las funciones definidas para una variable dada si el tamaño de la combinación resultante no excede un cierto límite. Es decir, si no se sobrepasa cierto umbral de tamaño, fijado de antemano, se elimina la variable de forma exacta; en caso contrario, no se combina ninguna función. El umbral de tamaño puede fijarse teniendo en cuenta la cantidad de memoria disponible en el sistema.

Este método puede mejorarse si nos damos cuenta de que quizás no podamos combinar todas las funciones, pero sí algunas de ellas, de forma que la pérdida de información sea menor que si no se combina ninguna. El proceso sería seleccionar R en la eliminación aproximada incluyendo funciones mientras la combinación resultante no exceda

el límite establecido. Nótese que, en este caso, el orden en que se seleccionan las funciones es importante. Hemos considerado dos enfoques diferentes:

- combinar funciones en un orden arbitrario mientras no se supere el tamaño máximo (*criterio 1*), o
- combinar primero aquellas que maximizan el tamaño de las variables en común (*criterio 2*). De lo que se trata es de que el tamaño de la combinación en relación con los tamaños de las funciones que se combinan sea tan pequeño como sea posible. Nótese que aquellas funciones definidas sobre una sola variable pueden siempre ser combinadas sin aumentar el tamaño de la combinación, dado que no añaden ninguna variable nueva. Por lo tanto, las funciones de una variable deberían ser combinadas siempre en primer lugar.

Ejemplo 3.2 *Supongamos que vamos a eliminar la variable X_1, que tiene asociada las siguientes funciones:*

$$f_1(X_1, X_2, X_3), f_2(X_1, X_4), f_3(X_1, X_5),$$

$$f_4(X_1, X_2, X_5), f_5(X_1), f_6(X_1, X_2, X_4, X_6),$$

donde cada variable tiene tres posibles valores, y el límite de tamaño para las funciones se establece en 81 valores. Siguiendo el criterio 1, las funciones se examinan en secuencia y se combinan si es posible. Comenzamos con f_1 y f_2. El tamaño del producto de ambas es 81, que queda por debajo del límite, y, por lo tanto, ambas funciones pueden ser combinadas. Ahora buscamos otra función que pueda ser combinada con el resultado del último producto. Esa función debería ser tal que el tamaño del resultado no tuviera más de 81 valores. f_5 es la única que cumple esta condición. Ahora continuamos con f_3 y f_4. Ambas se pueden combinar, dado que el tamaño del resultado es 27. Sin embargo, f_6 no puede ser combinada sin exceder el límite de 81 valores. Por lo tanto, después de el proceso de combinación obtenemos tres funciones:

$$h_1 = f_1 \cdot f_2 \cdot f_5, \quad h_2 = f_3 \cdot f_4, \quad h_3 = f_6.$$

Siguiendo el criterio 2, comenzamos con f_1 y la combinamos con f_5, dado que ésta última es una función de una variable y no añade ninguna complejidad. Ahora buscamos aquellas funciones susceptibles de ser combinadas y que tengan mayor tamaño en común. Estas funciones son el resultado de $f_1 \cdot f_5$ y f_4. Por esa misma razón, f_3 puede ser combinada con el resultado de la última operación. Las restantes funciones, f_2 y f_6 pueden combinarse también. Por lo tanto, después de aplicar el criterio 2, las funciones resultantes son

$$h_1 = f_1 \cdot f_5 \cdot f_4 \cdot f_3, \quad h_2 = f_2 \cdot f_6.$$

Se ha comprobado experimentalmente que el criterio 2 es el mejor de este grupo (ver sección 3.6).

3.5.1.2 Criterios de entropía

Ahora estamos interesados en mejorar los resultados dados por el criterio 2. La solución podría ser hacer un mayor esfuerzo en el proceso de combinación. El enfoque que proponemos es el siguiente: supongamos que después de aplicar el criterio 2, no se han combinado todas las funciones. Una forma de refinar el proceso podría ser seleccionar, de entre la que no se han combinado, la función más "apropiada" y combinarla con aquella con la que tenga un mayor tamaño en común (como en el criterio 2). Sin embargo, nótese que en este caso estamos permitiendo que se sobrepase el umbral de tamaño predefinido, pero esto ocurre como mucho una vez para cada variable. En muchos casos, podremos asumir el incremento de costo computacional.

El punto clave aquí es definir qué entendemos por la función más "apropiada". Una posibilidad es considerar la calidad de las funciones no combinadas. A saber, la función que proporcione la mayor cantidad de información debería ser la más susceptible de ser combinada. La cantidad de información que proporciona una función puede medirse mediante la *entropía de Shannon*, que se define como sigue:

Definición 3.1 *Dada una función masa de probabilidad f, definida sobre un conjunto finito Ω, se define su* entropía *como:*

$$E(f) = -\sum_{x \in \Omega} f(x) \log f(x). \tag{3.19}$$

Cuanto mayor es la entropía de una distribución f, menos informativa es ésta [52]. El máximo ($\log |\Omega|$) se alcanza cuando f es la distribución uniforme sobre Ω, mientras que el mínimo (0) se alcanza para cualquier distribución degenerada en un punto de Ω. Así, el criterio que podemos considerar es combinar aquella función con menor entropía, con aquella que resulte en un mayor tamaño de las variables en común. Nos referiremos a éste como *criterio 3*. El algoritmo detallado es el siguiente:

---

1. Sea $X_{\sigma(i)}$ la variable a eliminar.
2. Sea $H(i)$ el conjunto de funciones resultantes del proceso de combinación según el criterio 2.
3. Seleccionar $h \in H(i)$ tal que $E(h) = \min_{f \in H(i)} E(f)$.
4. Eliminar h de $H(i)$.
5. Elegir $h^* \in H(i)$ tal que $\|s(h) \cap s(h^*)\| = \max_{f \in H(i)} \|s(h) \cap s(f)\|$, donde $\|I\|$ es el producto del número de casos de las variables cuyos índices están en I.
6. Eliminar h^* de $H(i)$.
7. $H(i)$=$H(i) \cup \{h \cdot h^*\}$.

3.5.2 Tratamiento de las observaciones

Otra cuestión importante que establecer sobre el algoritmo principal es el tratamiento de las variables observadas. En este trabajo, hemos decidido incorporarlas antes de calcular las funciones de muestreo. De esta manera, las funciones quedan restringidas a los valores observados. Por lo tanto, se reducirá el tamaño de algunas funciones. Además, no hay necesidad de distinguir entre variables observadas o no observadas a la hora de simular. Este hecho reduce la posibilidad de obtener pesos nulos. Nótese que si las funciones se reducen a los valores observados, cualquier configuración que se obtenga en el proceso de simulación será consistente con las evidencias, a menos que alguna función de muestreo se anule debido a los errores en las aproximaciones. Una de las causas para la aparición de pesos nulos en los algoritmos de simulación existentes es la discrepancia entre los valores simulados y las observaciones. Por contra, la desventaja de incorporarlas al principio es que la inclusión dinámica de nuevas evidencias obligaría a calcular de nuevo las distribuciones de muestreo.

La otra opción es no restringir las funciones a las evidencias. En este caso, las variables observadas no se simulan, sino que directamente toman el valor observado. Entonces, la probabilidad de la evidencia se usa para ponderar la simulación. Así es como funciona el algoritmo de *ponderación por verosimilitud* [30]. La ventaja es que la inclusión de nuevas evidencias es sencilla, pero, por contra, se obtienen peores aproximaciones.

3.5.3 Elección del orden de eliminación de las variables

El orden de eliminación de las variables, a la hora de calcular las funciones de muestreo, determina la eficiencia del algoritmo de propagación. Este problema es similar al de la construcción de árboles de cliques óptimos en los algoritmos de propagación exactos [45, 46, 58]. Estos algoritmos requieren una triangulación previa del grafo moral asociado a la red causal. La triangulación se realiza eliminando las variables de la red en secuencia y conectando entre sí todos los nodos adyacentes al nodo eliminado.

Una adecuada elección del orden de eliminación lleva a la obtención de cliques de menor tamaño.

En el caso del algoritmo de muestreo por importancia, un adecuado orden de eliminación debe resultar en funciones con tamaño mínimo, y, por lo tanto, cabría la posibilidad de realizar más cálculos exactos sin exceder el límite de tamaño.

Pensamos que los órdenes utilizados en los procesos de triangulación deben producir también buenos resultados en nuestros algoritmos. También, se puede considerar eliminar, en cada paso, la variable que produzca la función más pequeña como resultado de la combinación. En la experimentación explicada en este capítulo, hemos empleado un orden de eliminación de hijos a padres, empezando desde las hojas del grafo. Por lo tanto, las variables son simuladas de padres a hijos, comenzando por las raíces. Esto se ha hecho así para poder comparar los algoritmos propuestos con el de ponderación por verosimilitud en igualdad de condiciones. La siguiente proposición establece cuándo son equivalentes los nuevos algoritmos y el clásico de ponderación por verosimilitud.

Proposición 3.1 *Sea* $X = \{X_1, \ldots, X_n\}$ *el conjunto de variables de una red causal, y* $H = \{f_1, \ldots, f_n\}$ *el conjunto de distribuciones condicionadas asociadas a cada variable. Supongamos que no hay ninguna variable observada. Entonces, si* σ *es un orden ancestral de los vértices de la red, y la secuencia de eliminación es de* $\sigma(n)$ *a* $\sigma(1)$*, el algoritmo de muestreo por importancia es equivalente al de ponderación por verosimilitud.*

Dem: Basta con demostrar que para cada variable X_i, $1 \leq i \leq n$, su distribución de muestreo es f_i, es decir, la distribución condicionada de X_i dados sus padres en el grafo. Es claro que ninguna variable $X_{\sigma(i)}$ aparece nunca en funciones $f_{\sigma(j)}$, $j < i$, dado que σ es un orden ancestral.

Además, las primeras variables en ser eliminadas son aquellas que se corresponden con hojas del grafo. Ahora, supongamos que X_i es una hoja. Su única función asociada es $f_i(x^{\downarrow i}, x^{\downarrow F(i)})$, $\forall x \in U_{s(f_i)}$. Después de eliminar X_i, la función resultante es

$$\sum_{x_i \in U_i} f_i(x_i, x), \quad \forall x \in U_{F(i)},$$

y, según la ecuación (2.2), es igual a 1.

Ahora, supongamos que hemos eliminado todas las hojas. En este punto, las próximas variables a eliminar son aquellas cuyos únicos descendientes eran hojas. Por lo tanto, ahora éstas son hojas también, y sus distribuciones asociadas son simplemente las condicionadas combinadas con las resultantes de la eliminación de las antiguas hojas, que hemos visto que valen 1. Por lo tanto, la función de muestreo para cualquier variable de la red, bajo estas condiciones, es simplemente su distribución condicionada original. □

3.6 Evaluación Experimental de los Algoritmos

En esta sección presentamos los resultados del análisis empírico que hemos realizado para comprobar el funcionamiento de los nuevos algoritmos. Para las pruebas hemos utilizado un grafo con 50 variables con zonas densas y no densas. Cada variable tiene entre 2 y 4 valores posibles. Partiendo de esta red hemos diseñado tres experimentos diferentes. En el *primero* de ellos, las distribuciones de probabilidad de la red se han generado siguiendo una distribución uniforme $\mathcal{U}(0, 1)$, y no se ha considerado evidencia alguna. En el *segundo*, la única diferencia es que 7 variables han sido observadas, tomando todas ellas su primer valor. En el *tercero*, las mismas 7 variables han sido observadas, y además se han modificado las distribuciones de probabilidad de la siguiente forma: la probabilidad de obtener el valor observado se ha establecido a 0 excepto para una configuración de los padres, para la cual esa probabilidad vale 1. Hemos procedido así para reproducir las situaciones donde el método de ponderación por verosimilitud falla (ver [36]).

Hemos probado los siguientes algoritmos:

PV : Ponderación por Verosimilitud.

MI2 : Muestreo por Importancia, criterio 2.

MI3 : Muestreo por Importancia, criterio 3.

El límite de tamaño para la combinación de funciones se ha establecido en 512 valores, es decir, se permite realizar combinaciones mientras la función resultante no tenga más de 512 valores. El orden de eliminación elegido es de hijos a padres, comenzando en las hojas; por lo tanto, las variables se simulan desde las raíces hasta las hojas. Cada algoritmo se ha ejecutado 100 veces, y se ha calculado el tiempo y el error medio. El número de simulaciones para cada ejecución de los algoritmos oscila entre 1.000 y 10.000, con saltos de un millar.

Para una variable X_l, la bondad de la estimación se mide como [28]:

$$G(X_l) = \sqrt{\frac{1}{|U_l|} \sum_{a_l \in U_l} \frac{(p'(a_l|e) - p(a_l|e))^2}{p(a_l|e)(1 - p(a_l|e))}}, \tag{3.20}$$

donde $p(a_l|E)$ es la verdadera distribución *a posteriori*, $p'(a_l|E)$ es el valor estimado y $|U_l|$ es el número de casos de la variable X_l. Para un conjunto de variables $(X_i)_{i \in I}$, la bondad de la estimación es:

$$G((X_i)_{i \in I}) = \sqrt{\sum_{i \in I} G(X_i)^2}. \tag{3.21}$$

Los experimentos se han llevado a cabo en un ordenador Intel Pentium a 75 MHz, con 16MB de memoria RAM y bajo sistema operativo Linux 1.2.13. El lenguaje de programación elegido ha sido el C++.

Las tablas 3.2,3.3 y 3.4 y las figuras 3.2, 3.3 y 3.4 muestran los resultados de la experimentación. Cada tabla muestra los tiempos medios de ejecución de cada algoritmo y el error en las estimaciones. Cada figura presenta el error de las estimaciones frente

a el número de iteraciones de simulación. Hay una gráfica para cada experimento, donde se muestran simultáneamente las curvas correspondientes a los tres algoritmos comprobados.

Atendiendo a los resultados obtenidos, podemos decir lo siguiente:

- No hay diferencias significativas entre los tres algoritmos probados cuando no hay evidencias en la red (experimento 1). En este caso, el de ponderación por verosimilitud (P.V.) se muestra más rápido, pues no requiere ningún proceso previo a la propagación, a diferencia de los nuevos algoritmos que proponemos, que han de ordenar las variables, calcular las distribuciones de muestreo y además utilizan estructuras de datos más complicadas que las necesarias para el P.V.

- En el experimento 2, los algoritmos que proponemos se comportan claramente mejor, debido a la presencia de variables observadas. La razón entre tiempo de ejecución y precisión de los resultados es ampliamente favorable a los algoritmos de muestreo por importancia. Los dos criterios considerados para el M.I. no ofrecen diferencias apreciables en este caso, ni en cuanto a tiempo de ejecución ni en cuanto a precisión. La explicación a esto podemos encontrarla en el hecho de que las distribuciones condicionadas se han generado siguiendo una uniforme, por lo que no es de esperar encontrar distribuciones con valores de entropía muy diferentes.

- El caso extremo es el experimento 3, donde se han modificado las distribuciones para que la probabilidad de la evidencia sea muy baja. En esta situación, el P.V. es incapaz de dar ninguna estimación de las probabilidades de la red. Sin embargo, los algoritmos de M.I. muestran una mayor robustez al seguir ofreciendo estimaciones tan buenas como en los casos anteriores. Puede observarse que aquí sí aparecen ciertas diferencias en favor del criterio 3 (entropía), dado que los valores de entropía de las funciones sí son ahora más variados, al haber introducido distribuciones para las variables observadas con muchos valores nulos.

3.7 Conclusiones

En este capítulo hemos presentado una nueva clase de algoritmos para la propagación de probabilidades en redes bayesianas. Los nuevos algoritmos se muestran más robustos que el de ponderación por verosimilitud en el caso general (cuando hay observaciones). Cuando no hay observaciones, todos tienen un comportamiento similar, de hecho, en este caso los algoritmos son equivalente para el orden de eliminación seleccionado (ver proposición 3.1).

Una ventaja importante de nuestros procedimientos es que se detecta cuándo se puede realizar la propagación exacta (cuando todas las variables han podido ser eliminadas utilizando el procedimiento exacto). En este caso podríamos dar resultados exactos, sin llevar a cabo la simulación.

Muchos aspectos de estos algoritmos deben ser estudiados en futuros trabajos. Por ejemplo, nuevos criterios para seleccionar las funciones que van a ser combinadas.

El orden inicial de las variables es otro punto a estudiar. Por ejemplo, podríamos considerar órdenes resultantes del proceso de triangulación de la red [8, 49, 34, 35, 56] que utilizan los métodos exactos a la hora de construir el árbol de cliques [45, 46, 58].

Además, otras técnicas de simulación se pueden relacionar con las utilizadas en este capítulo [29, 31, 72].

En el siguiente capítulo presentaremos una serie de nuevos algoritmos que resultan de aplicar la técnica del muestreo estratificado [4, 5, 72] a los muestreadores que hemos desarrollado.

	PV		MI2		MI3	
Iterac. (miles)	Tiempo	Error	Tiempo	Error	Tiempo	Error
1	2.21	0.224383	3	0.223273	3	0.222592
2	4.54	0.158836	5.94	0.157194	5.76	0.158537
3	6.96	0.127586	8.65	0.129869	8.6	0.126903
4	8.77	0.110999	11.38	0.110834	11.35	0.110898
5	11.02	0.099033	14.23	0.099848	14.15	0.100548
6	13.26	0.091193	17.07	0.091004	16.93	0.090534
7	15.45	0.084254	19.92	0.084289	19.74	0.084069
8	17.48	0.078250	22.66	0.078793	22.59	0.078858
9	20.01	0.074152	25.02	0.074511	25.79	0.074105
10	22	0.070350	27.77	0.070593	28.2	0.071061

Tabla 3.2: Resultados del experimento 1

	PV		MI2		MI3	
Iterac. (miles)	Tiempo	Error	Tiempo	Error	Tiempo	Error
1	2.28	0.369696	2.66	0.142527	2.8	0.144746
2	4.55	0.262667	5.31	0.118531	5.51	0.116398
3	7.09	0.214533	8.04	0.101944	8.31	0.103525
4	9.46	0.184992	10.6	0.091004	11.04	0.092368
5	11.36	0.167224	13.29	0.081832	13.78	0.082110
6	13.44	0.154013	15.96	0.077728	16.46	0.076983
7	16.04	0.139159	18.56	0.073540	19.27	0.073808
8	18.85	0.133997	21.23	0.067369	22.09	0.068852
9	20.54	0.125444	23.84	0.064750	26	0.065158
10	24.95	0.119467	26.56	0.061767	27.56	0.062544

Tabla 3.3: Resultados del experimento 2

	PV		MI2		MI3	
Iterac. (miles)	Tiempo	Error	Tiempo	Error	Tiempo	Error
1	-	-	2.65	0.190807	2.78	0.203285
2	-	-	5.54	0.135611	5.52	0.126449
3	-	-	7.94	0.119573	8.27	0.098778
4	-	-	10.59	0.097704	11.27	0.096473
5	-	-	13.25	0.084700	13.63	0.088657
6	-	-	15.84	0.062339	16.36	0.077345
7	-	-	18.47	0.090446	19.21	0.064577
8	-	-	21.42	0.054518	21.72	0.065522
9	-	-	23.84	0.067379	24.56	0.055519
10	-	-	26.58	0.060067	27.04	0.052993

Tabla 3.4: Resultados del experimento 3

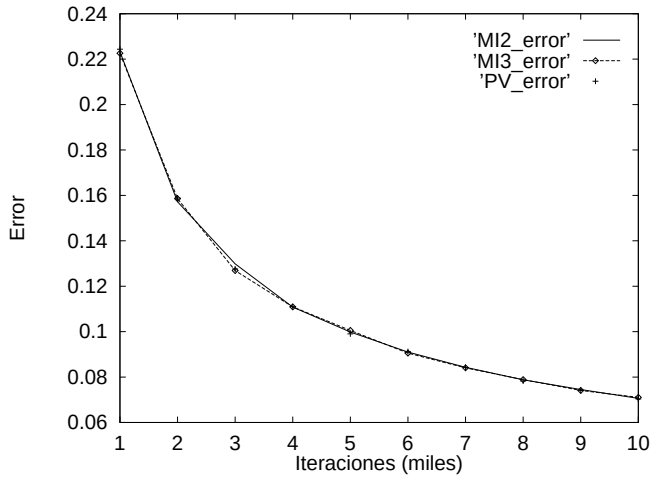

Figura 3.2: Experimento 1, Gráficas de errores.

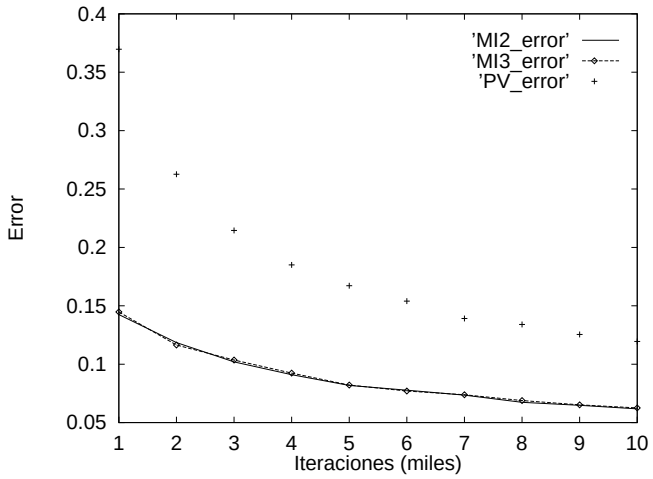

Figura 3.3: Experimento 2, Gráficas de errores.

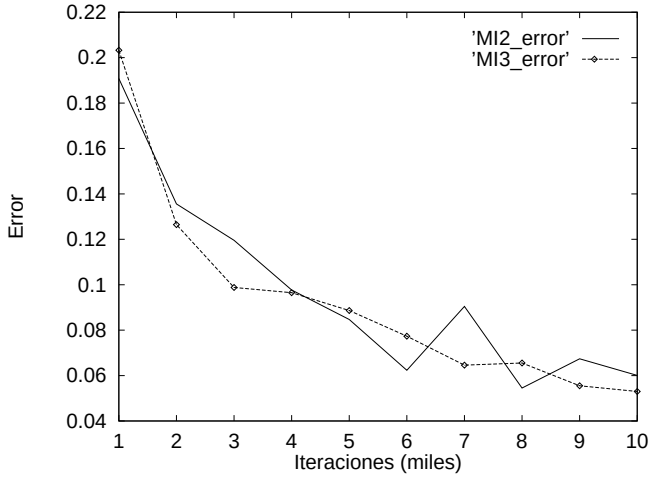

Figura 3.4: Experimento 3, Gráficas de errores.

Capítulo 4

Algoritmos de Muestreo Estratificado

4.1 Introducción

La simulación estratificada es una técnica muy conocida en estadística que conduce el proceso de simulación de forma que se eviten las muestras raras o desequilibradas. La idea básica consiste en dividir el espacio muestral en diversas regiones o estratos y elegir en cada uno de ellos un número óptimo de muestras. Esto produce una mejor representación del espacio muestral que la que resulta de las muestras aleatorias, y se pueden obtener mejores estimaciones para un tamaño determinado de la muestra o bien reducir el tamaño de la muestra para obtener la precisión requerida.

En este capítulo revisamos los trabajos donde se aplica esta técnica al problema de la propagación de probabilidades en redes bayesianas [4, 5], para incorporarlo después a nuestro sistema de simulación con precomputación.

La organización del capítulo por secciones es la siguiente. En la sección 4.2 comenzamos examinando los fundamentos teóricos del muestreo estratificado [72], que al igual que el muestreo por importancia, surge como una técnica de reducción de la varianza en el cálculo de integrales por Monte Carlo. La aplicación de esta técnica a los

algoritmos presentados en el capítulo anterior se expone en la sección 4.3. Esta técnica presenta problemas de precisión cuando las redes son de tamaño grande, debido a los errores de redondeo producidos al trabajar con números muy próximos a cero. Para resolverlo, en la sección 4.4 proponemos un método recursivo equivalente al estratificado que elimina los problemas de redondeo dividiendo el proceso de propagación en etapas y trabajando con números más alejados del cero. A continuación (sección 4.5) se detallan los experimentos realizados con los algoritmos propuestos, para finalizar el capítulo con las conclusiones en la sección 4.6.

4.2 Muestreo estratificado

El objetivo es calcular la siguiente suma:

$$I = \sum_{x \in U_N} f(x) = \sum_{x \in U_N} \frac{f(x)}{f^*(x)} \cdot f^*(x) = \sum_{x \in U_N} g(x) \cdot f^*(x),$$

donde $f^*(x)$ es una f.m.p. auxiliar que se usará para muestrear (al igual que en el muestreo por importancia) y $g(x) = f(x)/f^*(x)$. Ahora podemos dividir la región sobre la que se suma (U_N) en m estratos U_i, $i = 1, \ldots, m$, con $U_N = \bigcup_{i=1}^{m} U_i$ y $U_j \cap U_k = \phi$ si $j \neq k$, y expresar la suma como:

$$I = \sum_{i=1}^{m} \left[\sum_{x \in U_i} g(x) f^*(x) \right] = \sum_{i=1}^{m} I_i.$$

Denotamos por P_i la probabilidad del estrato i:

$$P_i = \sum_{x \in U_i} f^*(x).$$

Se cumple que

$$\sum_{i=1}^{m} P_i = 1.$$

Ahora denotamos por g_i la restricción de g al estrato i:

$$g_i(x) = \begin{cases} g(x) & \text{si} \quad x \in U_i, \\ 0 & si \quad x \notin U_i. \end{cases}$$

Entonces podemos expresar la suma de cada estrato como:

$$I_i = \sum_{x \in U_i} P_i g_i(x) \frac{f^*(x)}{P_i} = P_i E[g_i(x)].$$

La esperanza $E[g_i(x)]$ podemos aproximarla por la media muestral, obteniendo un estimador para I_i:

$$\tau_i = \frac{P_i}{N_i} \sum_{j_i=1}^{N_i} g(x_{j_i}),$$

donde N_i es el número de muestras que se toman en el estrato i, siendo el número total de muestras $N = \sum_{i=1}^{m} N_i$. Por lo tanto, un estimador de la suma I será:

$$\tau = \sum_{i=1}^{m} \tau_i = \sum_{i=1}^{m} \left[\frac{P_i}{N_i} \sum_{j_i=1}^{N_i} g(x_{j_i}) \right].$$

La elección más sencilla para P_i y N_i es tomar $P_i = 1/m$ y $N_i = NP_i$, en cuyo caso:

$$\tau = \sum_{i=1}^{m} \left[\frac{1}{N} \sum_{j_i=1}^{N/m} g(x_{j_i}) \right].$$

En general, en este método, el problema es elegir de forma óptima

- El número de muestras en cada estrato (N_i).
- La probabilidad de cada estrato, P_i, o lo que es lo mismo, la función de muestreo $f^*(x)$.

Un estudio detallado de esta técnica puede encontrarse en [72].

4.3 Aplicación del Muestreo Estratificado a los Nuevos Algoritmos

En esta sección, estudiamos cómo aplicar la técnica de muestreo estratificado [4, 5, 72] a los nuevos algoritmos estudiados en el capítulo anterior [36, 37].

4.3.1 Muestreo estratificado en redes bayesianas

Los primeros algoritmos de propagación basados en muestreo estratificado fueron desarrollados por Bouckaert [4] y Bouckaert, Castillo y Gutiérrez [5]. La idea es considerar el espacio de todas las posibles configuraciones de las variables de la red, y asignar a cada una de ellas un subintervalo de $[0, 1]$, de tal forma que las configuraciones más probables tengan asignado un subintervalo más amplio. Entonces, se selecciona un grupo de configuraciones muestreando sobre el intervalo $[0, 1]$. El procedimiento es el siguiente:

Sea un conjunto de variables $X = \{X_1, \ldots, X_n\}$, donde cada variable X_i toma valores en $U_i = \{0, 1, \ldots, r_i - 1\}$. Sean $f_i(x^{\downarrow i}, x^{\downarrow F(i)})$, $i = 1, \ldots, n$ las distribuciones condicionadas para cada variable dados sus padres en la red. En estas condiciones, podemos calcular todas las posibles configuraciones de las variables y calcular su probabilidad de ocurrencia. Podemos considerar un orden de las configuraciones de la siguiente manera [5]:

Definición 4.1 *Sean* $x = (x_1, x_2, \ldots, x_n)$ *e* $y = (y_1, y_2, \ldots, y_n)$ *dos configuraciones de* X. *Se dice que* x precede *a* y *(*$x < y$*) si:*

$$x < y \iff \exists k \text{ t.q. } \forall j < k \quad x_j = y_j \text{ y } x_k < y_k. \tag{4.1}$$

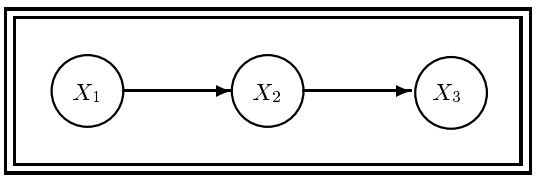

Figura 4.1: Una red bayesiana.

En base al orden definido en (4.1), se construye una tabla que representa el espacio muestral. Esta tabla se usa para obtener las configuraciones en el proceso de muestreo. Por ejemplo, sea $X = \{X_1, X_2, X_3\}$ un conjunto de variables, cada una con dos posibles

$f_1(0) = P(X_1 = 0) = 0.6$
$f_1(1) = P(X_1 = 1) = 0.4$
$f_2(0,0) = P(X_2 = 0 \mid X_1 = 0) = 0.2$
$f_2(0,1) = P(X_2 = 0 \mid X_1 = 1) = 0.5$
$f_2(1,0) = P(X_2 = 1 \mid X_1 = 0) = 0.8$
$f_2(1,1) = P(X_2 = 1 \mid X_1 = 1) = 0.5$
$f_3(0,0) = P(X_3 = 0 \mid X_2 = 0) = 0.2$
$f_3(0,1) = P(X_3 = 0 \mid X_2 = 1) = 0.3$
$f_3(1,0) = P(X_3 = 1 \mid X_2 = 0) = 0.8$
$f_3(1,1) = P(X_3 = 1 \mid X_2 = 1) = 0.7$

Tabla 4.1: Valores de probabilidad para la red anterior.

Configuración	Probabilidad	Prob. acumulada	Intervalo asociado
(0,0,0)	0.024	0.024	(0.000,0.024)
(0,0,1)	0.096	0.120	(0.024,0.120)
(0,1,0)	0.144	0.264	(0.120,0.264)
(0,1,1)	0.336	0.600	(0.264,0.600)
(1,0,0)	0.040	0.640	(0.600,0.640)
(1,0,1)	0.160	0.800	(0.640,0.800)
(1,1,0)	0.060	0.860	(0.800,0.860)
(1,1,1)	0.140	1.000	(0.860,1.000)

Tabla 4.2: Probabilidades e intervalos para las configuraciones ordenadas.

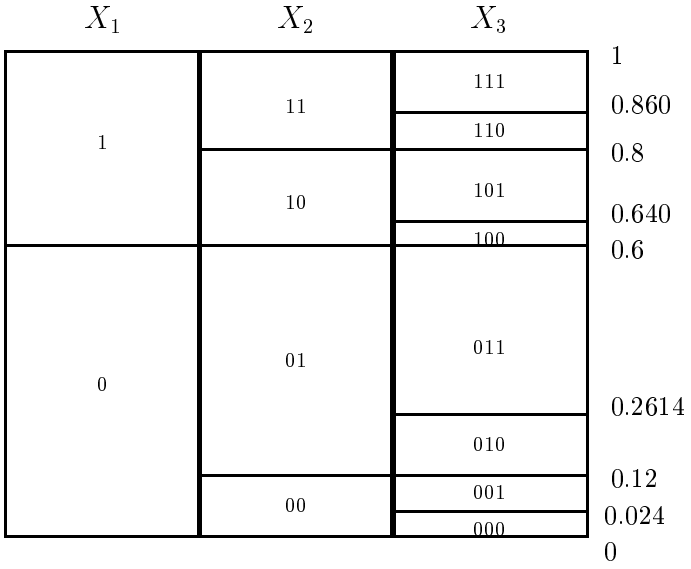

Figura 4.2: Configuraciones y sus probabilidades acumuladas.

valores. La figura 4.1 muestra la red asociada a X y la tabla 4.1 las probabilidades *a priori* para las variables de X. En la tabla 4.2 pueden verse las configuraciones ordenadas y su probabilidad de ocurrencia, probabilidad acumulada e intervalo asociado. Cada configuración x^i tiene asociado un intervalo $I_i = [l(i), h(i)) \subseteq [0, 1]$ cuyos límites se calculan a partir de las probabilidades acumuladas de acuerdo a las siguientes expresiones:

$$\begin{aligned} l(i) &= \sum_{j<i} \prod_{r=1}^{n} f_r^*(x^{j\downarrow r}), \\ h(i) &= l(i) + \prod_{r=1}^{n} f_r^*(x^{i\downarrow r}), \end{aligned} \tag{4.2}$$

donde x^j es la j-ésima configuración de la variable n-dimensional X y f_r^*, $r = 1, \ldots, n$, son las distribuciones de muestreo. La figura 4.2 muestra la división del intervalo $[0, 1]$ para la red de la figura 4.1.

Para obtener una muestra de tamaño m, se generan m números en el intervalo

$[0,1]$, y se comprueba qué configuración se corresponde con cada número generado, de acuerdo a la partición de la región (figura 4.2). A continuación, se pondera cada configuración de acuerdo con la distribución usada para calcular los intervalos (f_r^*) y la distribución original. Los m números no son aleatorios, sino que se calculan de forma determinista [5] de la siguiente manera,

$$k_i = \frac{i - 0.5}{m}, \quad i = 1, 2, \ldots, m.$$

El hecho de que estos números se generen de forma determinista motiva que este método se denomine también de *muestreo sistemático* [14].

El siguiente ejemplo explica cómo obtener una muestra a partir de una secuencia de números dada.

Ejemplo 4.1 *Considérese la red mostrada en la figura 4.1. Generando cuatro números* $k_i = (i - 0.5)/4$, $i = 1, \ldots, 4$, *obtenemos la secuencia*

$$(0.125, 0.375, 0.625, 0.875).$$

Ahora, para cada número, buscamos en el diagrama representado en la figura 4.2 las configuraciones correspondientes. Éstas son:

Número	*Configuración* (x_1, x_2, x_3)
0.125	$(0, 1, 0)$
0.375	$(0, 1, 1)$
0.625	$(1, 0, 0)$
0.875	$(1, 1, 1)$

Se puede apreciar que cuando m aumenta, la frecuencia relativa de cada configuración converge a su valor de probabilidad. El hecho de que no se utilicen números aleatorios hace que este algoritmo tenga un carácter más numérico que de simulación.

Nótese que las funciones de muestreo pueden ser cualesquiera, mientras verifiquen las condiciones de las distribuciones de muestreo por importancia [72], luego dependiendo de las que se usen, se obtendrán distintos resultados. Bouckaert, Castillo y Gutiérrez [5] usan las mismas funciones que en el algoritmo de ponderación por verosimilitud. Aquí estudiaremos el uso de las funciones de muestreo tal y como las calculamos en el capítulo anterior, es decir, mediante un proceso de precomputación aproximada [37].

4.3.2 El algoritmo de muestreo estratificado

En esta sección formalizaremos el método explicado en la sección anterior. La cuestión a resolver es cómo obtener la configuración correspondiente a un número dado $k_i \in [0, 1]$. Un procedimiento podría ser calcular las probabilidades para todas las configuraciones en secuencia, hasta que se alcance el valor k_i (es decir, aplicar la técnica de búsqueda en tablas para la generación de variables aleatorias discretas [72] a la variable n-dimensional). Sin embargo, la complejidad de este proceso es la misma que la de la obtención de las probabilidades exactas de la red, y suponemos que no hemos sido capaces de obtenerlas.

El método propuesto por Bouckaert et al. [5] comienza con una configuración de las variables, y, para un k_i dado, determina qué variables deben cambiar su valor para que la probabilidad acumulada de la configuración de todas las variables alcance a k_i. Para ello, asociado a cada variable habrá un intervalo $[l(i), h(i))$ representando las "zonas" de la región en las cuales la variable X_i cambia su valor (ver figura 4.2). Entonces, se busca la primera variable X_j tal que $k_j \in [l(j), h(j))$, comenzando con $j = n$. Cuando se encuentra ese j, los intervalos correspondientes a las variables $X_{j+1}, \ldots, X_n$ se actualizan para que contengan a k_i.

Una ventaja importante del muestreo estratificado aplicado de esta manera es que es posible saber cuántos números de la secuencia k_i se corresponden con la misma configuración, con el consiguiente ahorro de cálculos. Para un cierto $k \in [0, 1]$, esa cantidad será [5],

$$\Delta = \lfloor (h(n) - k) \cdot m \rfloor + 1, \tag{4.3}$$

donde $\lfloor x \rfloor$ es la parte entera de x para todo $x \in I\!R$.

El algoritmo general es el siguiente:

Algoritmo de Muestreo Estratificado (ALG ME)

1. Establecer un orden σ del conjunto de índices $N = \{1, \ldots, n\}$.
2. Calcular las distribuciones de muestreo f_l^*, $l \in N$ de acuerdo con la secuencia de eliminación impuesta por σ (como en el muestreo por importancia).
3. **Inicialización**.
4. Desde $j = 1$ hasta m (tamaño de la muestra),
   (a) **Generar_configuración**$(x^{(j)})$.
   (b) Calcular
   $$\Delta = \lfloor (h(n) - k_j) \cdot m \rfloor + 1.$$
   (c) Calcular
   $$w_j = \Delta \cdot \frac{\left(\prod_{i \in N} f_i(x^{(j)\downarrow s(f_i)})\right) \cdot \left(\prod_{r \in E} \delta_{e_r}(x^{(j)\downarrow r})\right)}{\prod_{l \in N} f_l^*(x^{(j)\downarrow s(f_l^*)})}.$$
   (d) $j = j + \Delta$.
5. Para cada $x_k \in U_k$, $k = \{1, \ldots, n\}$,
   (a) Estimar $p(x_k, e)$ usando la fórmula (3.12).
6. Normalizar los valores $p(x_k, e)$ para obtener $p(x_k|e)$.

Este algoritmo es bastante similar al de muestreo por importancia. Las diferencias radican en los procedimientos **Inicialización** y **Generar_configuración**. Estos procedimientos son como sigue [5]:

Inicialización

---

1. $l(0) = 0.0;\ \ h(0) = 1.0$.
2. Desde $i = 1$ hasta n,
   (a) $l(i) = 0.0$.
   (b) Si $X_{\sigma(i)} \in X_E$ hacer
   - $val(\sigma(i)) = e_{\sigma(i)}$.
   - $h(i) = h(i-1)$.

   En otro caso,
   - $val(\sigma(i)) = 0$.
   - $h(i) = h(i-1) \cdot f^*_{\sigma(i)}(0)$.

Este procedimiento calcula la primera configuración y su probabilidad. La configuración de las variables se almacena en el vector $val()$. Además, se inicializan los intervalos $[l(i), h(i))$ según la probabilidad del primer valor de cada variable X_i.

El procedimiento para generar configuraciones es el siguiente:

Generar_configuración($x^{(i)}$)

1. $k_i = \dfrac{i - 0.5}{m}$.
2. Si $l(n) \leq k_i \leq h(n)$ hacer $j = n$.
   En otro caso, seleccionar $j < n$ tal que $h(j-1) \geq k_i \geq h(j)$.
3. Mientras $j \leq n$,
   Si $X_{\sigma(j)} \in X_E$, entonces
   - $l(j) = l(j-1)$.
   - $h(j) = h(j-1)$.

   En otro caso,
   - $t = 0$.
   - $l(j) = l(j-1)$.
   - $h(j) = l(j) + (h(j-1) - l(j-1)) \cdot f^*_{\sigma(j)}(t)$.
   - Mientras $k_i > h(j)$ hacer
     - $t = t + 1$.
     - $l(j) = h(j)$.
     - $h(j) = l(j) + (h(j-1) - l(j-1)) \cdot f^*_{\sigma(j)}(t)$.
   - $val(\sigma(j)) = t$.

   $j = j + 1$.
4. Devolver la configuración $x^{(i)}$ en el vector $val()$.

En el algoritmo, k_i es el punto del intervalo $[0, 1]$ para el cual queremos saber la configuración de las variables que le corresponde. En el paso (2), se calcula la posición a partir de la cual debe actualizarse la configuración de las variables. Esta posición coincide con la de la primera variable para la cual su intervalo asociado $[l, h)$ contiene a k_i. En el paso (3) se obtiene la configuración y se actualizan los intervalos. Nótese que las variables observadas no intervienen en la construcción de los intervalos.

4.3.3 Problemas del muestreo estratificado

Aunque el algoritmo de muestreo estratificado es muy eficiente, en la práctica, surge un importante problema cuando se trata de propagar sobre redes grandes. El problema se debe a la limitada precisión de la representación los números reales en los ordenadores. Obsérvese que cuando se calculan los intervalos $[l, h)$, se usa la siguiente expresión,

$$h(j) = l(j) + (h(j-1) - l(j-1)) \cdot f^*_{\sigma(j)}(t),$$

en la cual realizamos dos sumas y un producto con números a menudo muy próximos a cero. En general, cuanto más grande es la red, más pequeños serán estos números. El resultado es que si se usan números en punto flotante para representar los límites de los intervalos, al final todos se redondean a cero, debido a la pérdida de dígitos significativos, lo que hace que el algoritmo no funcione.

Una solución podría ser incrementar el número de bits usados para representar los números en punto flotante, pero siempre podremos encontrar una red suficientemente grande como para que todos los intervalos se conviertan en cero.

Por otro lado, conforme aumenta el tamaño de la red disminuye el de los intervalos, con lo cual es menos probable que caiga más de un número dentro de cada intervalo. Esto provoca que una de las principales razones de la velocidad de este método no sea efectiva, dado que los números tenderán a caer cada uno en un intervalo distinto.

En la siguiente sección proponemos un método alternativo para realizar el muestreo

estratificado que evita los problemas de precisión [37].

4.4 Muestreo Estratificado Recursivo

Que el muestreo estratificado funcione o no, depende fuertemente del tamaño de la red. Sería interesante encontrar una forma de poder aplicar este método a una red de tamaño arbitrario. En esta sección proponemos un esquema de propagación donde el muestreo estratificado se aplica a cada variable por separado, con lo que su validez es independiente del número de variables. En cada paso, los intervalos se escalan al $[0, 1]$, con lo que no aparecen errores de redondeo tan pronunciados. Empezaremos con una primera aproximación al algoritmo, explicando con un ejemplo su funcionamiento para después formalizar el algoritmo detallado, que puede ser implementado de forma recursiva, y su equivalencia con el muestreo estratificado clásico. Un primer enfoque es el siguiente:

1. Elegir un orden $\{X_1, \ldots, X_n\}$ de las variables de la red y calcular las distribuciones de muestreo.
2. Tomar m números $k_i \in [0, 1]$ de la misma manera que en el muestreo estratificado original.
3. Desde $t = 1$ hasta $n - 1$,
   (a) Considerar X_t. (Inicialmente $t = 1$).
   (b) Para cada posible valor de X_t,
      i. Sea r la cantidad de números k_i que caen dentro del intervalo correspondiente al valor actual de X_t (ver fórmula (4.3)).
      ii. Generar r configuraciones de la variable X_{t+1}.

Cuando este procedimiento recursivo termina, cada variable ha sido simulada m veces. El siguiente ejemplo puede aclarar la idea.

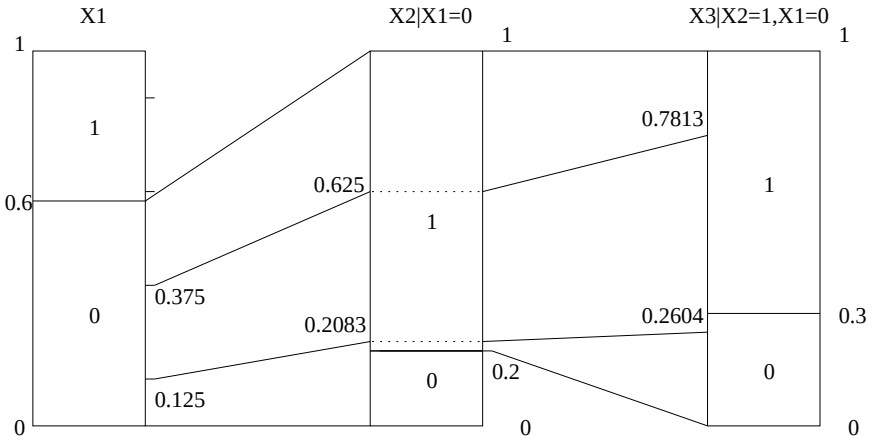

Figura 4.3: Muestreo estratificado recursivo.

Ejemplo 4.2 *Consideremos la red mostrada en la figura 4.1, y la secuencia de números* $k_i = 0.125, 0.375, 0.625, 0.875$. *Cuando generamos los cuatro números* k_i, *podemos saber cuántos de ellos caen dentro del intervalo asociado al primer caso de la variable* X_1. *Las probabilidades utilizadas para calcular los intervalos son las de la tabla 4.1. En este caso, en el intervalo* $[0, 0.6)$ *hay dos números:* $k_1 = 0.125$ *y* $k_2 = 0.375$. *Así, sabemos que en la muestra final habrá dos configuraciones* (x_1, x_2, x_3) *con* $x_1 = 0$.

Si usamos el muestreo estratificado de Bouckaert, hemos de trabajar entonces en el intervalo $[0, 0.6)$ *de cara a obtener las configuraciones de* X_2 *para los valores* k_1 *y* k_2. *Es decir, hemos de aplicar (4.2). Pero, recordemos, queremos solventar el problema del redondeo. Para ello, llevamos el intervalo* $[0, 0.6)$ *al intervalo* $[0, 1)$, *de forma que los valores* k_1 *y* k_2 *se transforman en* $k_1' = 0.2083$ *y* $k_2' = 0.625$ *(ver figura 4.3).*

A continuación, repetimos el proceso para X_2, *para la cual tenemos que generar dos valores, pero ahora trabajando en el intervalo* $[0, 1]$, *que dividimos en subintervalos de acuerdo a* $P(X_2|X_1 = 0)$. *En este caso, en el nuevo subintervalo* $[0, 0.2)$ *no hay ningún* k_i' *correspondiéndose con la primera configuración de* X_2, *y para la segunda configuración de* X_2, *y en el nuevo subintervalo* $[0.2, 1)$, *hay dos* k_i'. *Por lo tanto, obtenemos dos configuraciones* (x_1, x_2, x_3) *con* $x_2 = 1$, *que son aquellas asociadas a* k_1' *y* k_2'.

Ahora repetimos el mismo proceso para la variable X_3. Esto es, transformamos el intervalo $[0.2, 1)$ en el $[0, 1)$, y los valores k_1' y k_2' pasan a ser $k_1'' = 0.2604$ y $k_2'' = 0.7813$. El siguiente paso es identificar los valores de X_3 que se corresponden con los valores de k_1'' y k_2'' (ver figura 4.3). Estos valores son $X_3 = 0$ para $k_1'' = 0.2604$ y $X_3 = 1$ para $k_2'' = 0.7813$. Por lo tanto, obtenemos dos configuraciones (x_1, x_2, x_3), una con $x_3 = 0$ y otra con $x_3 = 1$.

Dado que no hay más variables, volvemos hasta la primera variable para reconstruir las configuraciones completas. De esta forma podemos concluir diciendo que las dos primeras configuraciones de la muestra final son $(0, 1, 0)$ y $(0, 1, 1)$.

En definitiva, se obtiene el mismo resultado que con el muestreo estratificado habitual, pero eliminando los errores de redondeo al no reducirse el tamaño de los intervalos conforme se avanza en la simulación.

Una versión detallada del algoritmo, expresada como un procedimiento recursivo, es la siguiente.

Algoritmo Estratificado Recursivo (ALG MER)

1. Seleccionar un orden σ del conjunto de índices $N = \{1, \ldots, n\}$.
2. Calcular las distribuciones de muestreo f_l^*, $l \in N$ de acuerdo con la secuencia de eliminación impuesta por σ (como en el muestreo por importancia).
3. Sea m el tamaño de la muestra.
4. Incremento: $inc = \dfrac{1}{m}$.
5. Primer número: $k = \dfrac{0.5}{m}$.
6. Límites del intervalo: $l = 0.0$, $h = 1.0$.

7. Variable a simular: $i = 1$ (la primera).
8. **SIMULA**(i,l,h,k,inc,m).
9. Para cada $x_k \in U_k$, $k = \{1, \ldots, n\}$,
   (a) Estimar $p(x_k, e)$ usando la fórmula (3.12).
10. Normalizar los valores $p(x_k, e)$ para obtener $p(x_k|e)$.

---

donde **SIMULA** es un procedimiento que funciona como el muestreo estratificado clásico pero sólo para una variable en cada momento, y que vuelve a llamarse a él mismo para simular el resto de las variables.

SIMULA(i,l,h,$k^{(i-1)}$,inc,Δ)

---

1. Si $i > n$ entonces,
   - Asignar un peso a la configuración almacenada en el vector val como en el paso 4.(c) del algoritmo **ALG ME**.
   - Terminar.
2. Si $X_{\sigma(i)} \in X_E$ entonces,
   (a) $val[\sigma(i)] = e_{\sigma(i)}$.
   (b) $k^{(i)} = k^{(i-1)}$.
   (c) **SIMULA**($i + 1$,l,h,$k^{(i)}$,inc,Δ).
   (d) Terminar.

3. $k^{(i)} = \dfrac{k^{(i-1)} - l}{h - l}, \quad inc = \dfrac{inc}{h - l}.$
4. $l = 0.0, \quad h = 0.0.$
5. $v = 0.$
6. Mientras $k^{(i)} < 1.0$ hacer
   (a) $l = h.$
   (b) $h = h + f^*_{\sigma(i)}(v).$
   (c) Mientras $h < k^{(i)}$ hacer
      - $v = v + 1.$
      - $l = h.$
      - $h = h + f^*_{\sigma(i)}(v).$
   (d) $val[\sigma(i)] = v.$
   (e) $\Delta = \left\lfloor \dfrac{h - k^{(i)}}{inc} \right\rfloor + 1.$
   (f) **SIMULA**$(i+1, l, h, k^{(i)}, inc, \Delta)$.
   (g) $k^{(i)} = k^{(i)} + \Delta \cdot inc.$
   (h) $v = v + 1.$

A continuación explicamos el funcionamiento del procedimiento **SIMULA**. Cuando se llama a este procedimiento, se comprueba si todas las variables han sido ya simuladas. Si la respuesta es afirmativa, entonces ya tenemos una configuración completa, cuyo peso puede calcularse como en el paso 4.(c) del algoritmo **ALG ME**.

En otro caso, se comprueba si la variable de entrada $X_{\sigma(i)}$ está observada o no. En el primer caso no se simula: el algoritmo toma directamente el valor observado y los

parámetros de entrada no cambian y se pasa a simular la siguiente variable. Esta tarea se realiza en el paso 2.

Pero si la variable de entrada no está observada, hay que elegir un valor v para ella. El paso 3 escala los valores de $k^{(i)}$ e *inc* al intervalo $[0,1]$. Es decir, se dividen los valores por la amplitud del intervalo correspondiente a la variable anterior. Recuérdese que $k^{(i)}$ es el número que determinará qué valor de la variable se va a seleccionar, de acuerdo con la probabilidad acumulada de los casos de la variable, y que *inc* es el incremento para alcanzar el próximo valor de $k^{(i)}$.

Los límites del intervalo han de ser escalados al $[0,1]$ también, lo que viene determinado implícitamente por la condición $k^{(i)} < 1$ en el paso 6. En este paso, se toma el menor $k^{(i)}$ y se elige un valor v para la variable $X_{\sigma(i)}$; más tarde se toma el siguiente $k^{(i)}$ y se repite el proceso, y así sucesivamente hasta que se hayan tomado todos los valores $k^{(i)}$ para el intervalo $[0,1]$ asociado a la variable $X_{\sigma(i)}$.

De cara a seleccionar un valor v para la variable, debemos fijar los intervalos

$$[l = \sum_{u<v} f^*_{\sigma(i)}(u), h = l + f^*_{\sigma(i)}(v)]$$

que contienen los valores $k^{(i)}$. Por lo tanto, dado $k^{(i)}$, debemos establecer cuál es su intervalo asociado $[l,h]$ para obtener un valor v. Éste es el objetivo de los pasos 6.(a)-6.(d).

Una vez se ha fijado el intervalo, en el paso 6.(e) se calcula el número de configuraciones, Δ, que se corresponden con el actual valor v de la variable $X_{\sigma(i)}$. Entonces, en el paso 6.(g), se toma el siguiente valor $k^{(i)}$ y se repite el proceso. Además, para el valor actual de la variable, deben simularse las demás variables también. En concreto, sabemos que habrá Δ configuraciones para el valor v. Para ello se llama recursivamente al procedimiento **SIMULA** en el paso 6.(f).

Obsérvese que todas las variables se simulan en el intervalo $[0,1]$ completo, independientemente del tamaño de la red, lo que hace que este algoritmo sea válido para redes de tamaño arbitrario.

Otra ventaja de este procedimiento respecto al muestreo estratificado clásico es que se evita buscar la configuración actual, en concreto, el paso 2 del procedimiento **Generar_configuración** del algoritmo **ALG ME** (sección 4.3.2), no es necesario.

Los siguientes resultados muestran que el algoritmo **ALG MER** es equivalente al **ALG ME** a nivel teórico, es decir, suponiendo que no está limitado el número de cifras decimales en la representación interna de los números reales.

Lema 4.1 *Sea* $X = \{X_{\sigma(1)}, \ldots, X_{\sigma(n)}\}$ *el conjunto de variables de la red, cada variable* $X_{\sigma(i)}$ *tomando valores en un conjunto* $U_{\sigma(i)}$. *Sea* $N = \{1, \ldots, n\}$, *y* $x_0 \in U_N$ *una configuración de las variables en* X. *Si* $f^*_{\sigma(j)}$, $j = 1, \ldots, n$ *son las distribuciones de muestreo para cada variable, entonces, para todo* $i = 1, \ldots, n$ *se cumple que*

$$\sum_{\substack{y < x_0^{\downarrow I} \\ y \in U_I}} \prod_{j \in I} f^*_j(y^{\downarrow j}) = \left(\prod_{j \in I'} f^*_j(x_0^{\downarrow j}) \right) \left(\sum_{\substack{y < x_0^{\downarrow \sigma(i)} \\ y \in U_{\sigma(i)}}} f^*_{\sigma(i)}(y) \right) + \sum_{\substack{y < x_0^{\downarrow I'} \\ y \in U_{I'}}} \prod_{j \in I'} f^*_j(y^{\downarrow j}), \quad (4.4)$$

donde $I = \{\sigma(1), \ldots, \sigma(i)\}$ *y* $I' = I - \{\sigma(i)\}$.

Dem: Si definimos los siguientes sucesos:

$$\begin{aligned} A &= \text{“una configuración } y \in U_I \text{ es menor que } x_0^{\downarrow I}\text{”}, \\ B &= \text{“una configuración } y \in U_{I-\{\sigma(i)\}} \text{ es igual a } x_0^{\downarrow I-\{\sigma(i)\}}\text{”}, \\ C &= \text{“una configuración } y \in U_{\{\sigma(i)\}} \text{ es menor que } x_0^{\downarrow \sigma(i)}\text{”}, \\ D &= \text{“una configuración } y \in U_{I-\{\sigma(i)\}} \text{ es menor que } x_0^{\downarrow I-\{\sigma(i)\}}\text{”}, \end{aligned}$$

entonces se verifica que $A = (B \wedge C) \vee D$ y, dado que $A = (B \wedge C)$ y D son sucesos disjuntos, y $P^*(B \wedge C) = P^*(B) \cdot P^*(C|B)$,

$$P^*(A) = P^*(B) \cdot P^*(C|B) + P^*(D),$$

donde P^* es la distribución de muestreo. Esta igualdad es trivialmente cierta y equivalente a la ecuación (4.4). □

Proposición 4.1 *Supongamos que* $k \in [0,1]$. *Sea* $k^{(i)}$ *el valor transformado de* k *cuando se simula la variable* $X_{\sigma(i)}$ *en el algoritmo* **ALG MER**. *Sean* $I = \{\sigma(1), \ldots, \sigma(i)\}$, $I' = I - \{\sigma(i)\}$, *y sea* $x_0 \in U_N$ *una configuración de todas las variables de la red. Entonces,*

$$k = \sum_{\substack{y < x_0^{\downarrow I'} \\ y \in U_{I'}}} \prod_{j \in I'} f_j^*(y^{\downarrow j}) + k^{(i)} \prod_{j \in I'} f_j^*(x_0^{\downarrow j}).$$

Dem: Nótese que las variables observadas no se simulan, y por lo tanto éstas no cambian los límites del intervalo, l y h.

Haremos la demostración por inducción.

Inicialmente, en la primera ejecución del procedimiento **SIMULA**, $l = 0.0$ y $h = 1.0$, luego $k^{(1)} = \dfrac{k-l}{h-l} = k$, y, por el paso 3 en la siguiente llamada al procedimiento **SIMULA**, con parámetros $(2, l, h, k^{(1)}, inc, \Delta)$,

$$k^{(2)} = \frac{k^{(1)} - l}{h - l} = \frac{k^{(1)} - \sum_{\substack{y < x_0^{\downarrow\sigma(1)} \\ y \in U_{\sigma(1)}}} f_{\sigma(1)}^*(y^{\downarrow\sigma(1)})}{f_{\sigma(1)}^*(x_0^{\downarrow\sigma(1)})},$$

lo que se sigue del hecho de que, para cualquier variable $X_{\sigma(i)}$, los valores de l y h son aquellos calculados antes de llamar al procedimiento **SIMULA** cuando se simula la variable anterior:

$$l = \sum_{\substack{y < x_0^{\downarrow\sigma(i-1)} \\ y \in U_{\sigma(i-1)}}} f_{\sigma(i-1)}^*(y^{\downarrow\sigma(i-1)}),$$

$$h = \sum_{\substack{y \leq x_0^{\downarrow\sigma(i-1)} \\ y \in U_{\sigma(i-1)}}} f^*_{\sigma(i-1)}(y^{\downarrow\sigma(i-1)}).$$

Por lo tanto,

$$k = \sum_{\substack{y < x_0^{\downarrow\sigma(1)} \\ y \in U_{\sigma(1)}}} f^*_{\sigma(1)}(y^{\downarrow\sigma(1)}) + k^{(2)} \cdot f^*_{\sigma(1)}(x_0^{\downarrow\sigma(1)}).$$

Luego la proposición se cumple para $i = 1$ e $i = 2$.

Ahora demostraremos que si la proposición se cumple para cualquier i, entonces también se cumple para $i + 1$:

$$k = \sum_{\substack{y < x_0^{\downarrow I'} \\ y \in U_{I'}}} \prod_{j \in I'} f^*_j(y^{\downarrow j}) + k^{(i)} \prod_{j \in I'} f^*_j(x_0^{\downarrow j}).$$

Del procedimiento **SIMULA**$(i+1, l, h, k^{(i+1)}, inc, \Delta)$, paso 3, se sigue que

$$k^{(i+1)} = \frac{k^{(i)} - \sum_{\substack{y < x_0^{\downarrow\sigma(i)} \\ y \in U_{\sigma(i)}}} f^*_{\sigma(i)}(y^{\downarrow\sigma(i)})}{f^*_{\sigma(i)}(x_0^{\downarrow\sigma(i)})},$$

luego,

$$k = \sum_{\substack{y < x_0^{\downarrow I'} \\ y \in U_{I'}}} \prod_{j \in I'} f^*_j(y^{\downarrow j}) + \left(k^{(i+1)} \cdot f^*_{\sigma(i)}(x_0^{\downarrow\sigma(i)}) + \sum_{\substack{y < x_0^{\downarrow\sigma(i)} \\ y \in U_{\sigma(i)}}} f^*_{\sigma(i)}(y^{\downarrow\sigma(i)}) \right) \prod_{j \in I'} f^*_j(x_0^{\downarrow j})$$

$$= \sum_{\substack{y<x_0^{\downarrow I'} \\ y\in U_{I'}}} \prod_{j\in I'} f_j^*(y^{\downarrow j}) + k^{(i+1)} \prod_{j\in I} f_j^*(x_0^{\downarrow j}) + \prod_{j\in I'} f_j^*(x_0^{\downarrow j}) \sum_{\substack{y<x_0^{\downarrow \sigma(i)} \\ y\in U_{\sigma(i)}}} f_{\sigma(i)}^*(y^{\downarrow \sigma(i)}),$$

y según el lema 4.1,

$$k = \sum_{\substack{y<x_0^{\downarrow I} \\ y\in U_I}} \prod_{j\in I} f_j^*(y^{\downarrow j}) + k^{(i+1)} \prod_{j\in I} f_j^*(x_0^{\downarrow j}).$$

□

Teorema 4.1 *Sea k_j, $j \in \{1,\ldots,m\}$ una secuencia de números pertenecientes al intervalo $[0,1]$. Sea $\{x^{(1)},\ldots,x^{(m)}\}$ el conjunto de configuraciones obtenidas por el algoritmo* **ALG ME** *para la secuencia anterior, y $\{y^{(1)},\ldots,y^{(m)}\}$ las obtenidas por el algoritmo* **ALG MER**. *Entonces,*

$$\forall i \in \{1,\ldots,m\}, \quad x^{(i)} = y^{(i)}.$$

Dem: Sean I y I' iguales que en la proposición 4.1. Para un $k \in [0,1]$ dado, la configuración obtenida por el algoritmo **ALG ME** es aquella configuración x_0 verificando

$$\sum_{\substack{y<x_0^{\downarrow I} \\ y\in U_I}} \prod_{j\in I} f_j^*(y^{\downarrow j}) < k \leq \sum_{\substack{y\leq x_0^{\downarrow I} \\ y\in U_I}} \prod_{j\in I} f_j^*(y^{\downarrow j}),$$

mientras que el algoritmo **ALG MER** devuelve aquella configuración verificando que, para todo $i \in \{1,\ldots,n\}$,

$$\sum_{\substack{y<x_0^{\downarrow \sigma(i)} \\ y\in U_{\sigma(i)}}} f_{\sigma(i)}^*(y) < k^{(i)} \leq \sum_{\substack{y\leq x_0^{\downarrow \sigma(i)} \\ y\in U_{\sigma(i)}}} f_{\sigma(i)}^*(y).$$

Por lo tanto, para demostrar que el teorema se cumple, es suficiente probar la siguiente equivalencia:

$$\sum_{\substack{y<x_0^{\downarrow\sigma(i)}\\ y\in U_{\sigma(i)}}} f^*_{\sigma(i)}(y) < k^{(i)} \leq \sum_{\substack{y\leq x_0^{\downarrow\sigma(i)}\\ y\in U_{\sigma(i)}}} f^*_{\sigma(i)}(y) \Leftrightarrow \sum_{\substack{y<x_0^{\downarrow I}\\ y\in U_I}} \prod_{j\in I} f^*_j(y^{\downarrow j}) < k \leq \sum_{\substack{y\leq x_0^{\downarrow I}\\ y\in U_I}} \prod_{j\in I} f^*_j(y^{\downarrow j}).$$

Por la proposición 4.1, esta última desigualdad es cierta si y sólo si

$$\sum_{\substack{y<x_0^{\downarrow I}\\ y\in U_I}} \prod_{j\in I} f^*_j(y^{\downarrow j}) < \sum_{\substack{y<x_0^{\downarrow I'}\\ y\in U_{I'}}} \prod_{j\in I'} f^*_j(y^{\downarrow j}) + k^{(i)} \prod_{j\in I'} f^*_j(x_0^{\downarrow j}) \leq \sum_{\substack{y\leq x_0^{\downarrow I}\\ y\in U_I}} \prod_{j\in I} f^*_j(y^{\downarrow j}),$$

y por el lema 4.1, esto es equivalente a

$$\sum_{\substack{y<x_0^{\downarrow\sigma(i)}\\ y\in U_{\sigma(i)}}} f^*_{\sigma(i)}(y) < k^{(i)} \leq \sum_{\substack{y\leq x_0^{\downarrow\sigma(i)}\\ y\in U_{\sigma(i)}}} f^*_{\sigma(i)}(y).$$

□

Este teorema da validez al algoritmo de muestreo estratificado recursivo, y asegura que, en todos los casos en que el algoritmo clásico sea aplicable, el algoritmo recursivo proporciona los mismos resultados. En los casos en que el algoritmo clásico no sea aplicable debido a los errores de redondeo, el algoritmo recursivo proporciona los resultados que el clásico, en teoría, debería ofrecer. Esto quiere decir que los resultados sobre convergencia que pudieran aplicarse al método de Bouckaert, Castillo y Gutiérrez [5], son también aplicables a nuestro método.

4.5 Evaluación Experimental de los Algoritmos

En esta sección presentamos los resultados de la evaluación experimental de los algoritmos propuestos en este capítulo. Se han realizado tres experimentos, todos ellos iguales a los del capítulo anterior, salvo que en este caso, dado el carácter determinista de estos algoritmos, cada prueba sólo se ejecuta una vez para cada valor fijado del número de iteraciones de simulación.

También hemos comparado experimentalmente el comportamiento de los algoritmos de muestreo estratificado frente a los de muestreo por importancia.

Las pruebas se han realizado con los siguientes algoritmos:

PV : Ponderación por Verosimilitud.

MI2 : Muestreo por importancia, criterio 2.

MI3 : Muestreo por importancia, criterio 3.

PVE : Muestreo estratificado clásico (PV Estratificado).

ME2 : Muestreo estratificado recursivo, criterio 2.

ME3 : Muestreo estratificado recursivo, criterio 3.

Los resultados de la experimentación se muestran en las tablas 4.3, 4.4, 4.5, y en las figuras 4.4 a 4.14. Cada una de las tres tablas muestra los tiempos y errores en las estimaciones para cada algoritmo en cada uno de los experimentos. Los errores en las estimaciones son también presentados en una gráfica para cada experimento. Además, se han incluido una gráficas adicionales en las que se compara cada algoritmo con su versión estratificada. Atendiendo a los resultados obtenidos, podemos decir lo siguiente:

- No hay diferencias importantes entre los algoritmos de muestreo estratificado cuando no se presentan evidencias (experimento 1). Se puede apreciar que, en

general, se obtienen mejores resultados en cada algoritmo estratificado que en su equivalente por importancia.

- En el experimento 2, se observa que los nuevos algoritmos de muestreo estratificado mejoran claramente al clásico, al igual que ocurría para el muestreo por importancia.

- En el experimento 3, ni el PV ni el PVE son capaces de dar una estimación de las probabilidades, mientras que los nuevos algoritmos mantienen un comportamiento similar al de los dos experimentos anteriores.

- En los experimentos 2 y 3, se observa que el comportamiento de los algoritmos de muestreo por importancia es menos oscilante respecto al número de iteraciones de simulación que los estratificados. Esto se debe al hecho de que cada algoritmo por importancia se ha ejecutado 100 veces, promediando los resultados, mientras que los estratificados solo se ejecutan una vez dado su carácter determinista.

- Al igual que ocurría en la experimentación del capítulo anterior, no se aprecian diferencias importantes entre los criterios 2 y 3 para obtener las funciones de muestreo. Las razones son las mismas que en el caso de los algoritmos de muestreo por importancia.

4.6 Conclusiones

En este capítulo hemos presentado una nueva clase de algoritmos de muestreo estratificado, que difieren de los existentes en las funciones de muestreo que utilizan para calcular los intervalos en los que se buscan las configuraciones. Puede decirse que el avance es similar al de los algoritmos de muestreo por importancia respecto al de ponderación por verosimilitud, es decir, los nuevos métodos se muestran mucho menos sensibles a la aparición de probabilidades muy pequeñas. Sin embargo, hemos de recordar que el problema que estamos abordando es NP-duro. Por lo tanto, siempre podremos encontrar ejemplos en los que nuestros algoritmos, que son polinomiales,

no funcionen. De cualquier manera, pensamos que la metodología aquí presentada permitirá resolver una amplia clase de problemas.

En cuanto al muestreo estratificado recursivo, hemos visto que es capaz de trabajar con redes de tamaño arbitrario, lo cual es un avance importante. Además, hemos demostrado su equivalencia teórica con el muestreo estratificado clásico; por lo tanto, verificará los mismos resultados teóricos de convergencia estudiados por Bouckaert, Castillo y Gutiérrez [5].

	PVE		ME2		ME3	
Iterac. (miles)	Tiempo	Error	Tiempo	Error	Tiempo	Error
1	4	0.216340	3	0.190459	3	0.217222
2	8	0.143627	6	0.155694	6	0.137056
3	12	0.103606	9	0.108142	8	0.115696
4	16	0.090895	12	0.079989	11	0.088431
5	20	0.096965	14	0.075941	14	0.084204
6	24	0.083591	17	0.080147	17	0.072361
7	29	0.086390	20	0.085973	20	0.081358
8	33	0.079168	23	0.076603	23	0.073819
9	36	0.067176	26	0.058812	25	0.056180
10	40	0.063141	28	0.059815	29	0.068373

Tabla 4.3: Resultados del experimento 1.

	PVE		ME2		ME3	
Iterac. (miles)	Tiempo	Error	Tiempo	Error	Tiempo	Error
1	4	0.308384	3	0.156283	3	0.134790
2	4	0.245075	6	0.128473	6	0.148865
3	11	0.167003	8	0.105013	9	0.089845
4	15	0.161373	11	0.080167	11	0.084454
5	19	0.172051	14	0.073035	14	0.076830
6	22	0.141424	16	0.072736	16	0.083851
7	26	0.117056	19	0.067206	19	0.062817
8	29	0.130670	21	0.062601	21	0.064927
9	32	0.116119	25	0.050559	24	0.049558
10	37	0.134387	27	0.051647	26	0.059121

Tabla 4.4: Resultados del experimento 2.

	PVE		ME2		ME3	
Iterac. (miles)	Tiempo	Error	Tiempo	Error	Tiempo	Error
1	-	-	2	0.162481	3	0.163088
2	-	-	5	0.102723	5	0.110056
3	-	-	8	0.102858	8	0.095401
4	-	-	10	0.067462	10	0.064753
5	-	-	13	0.064567	13	0.065017
6	-	-	15	0.070015	15	0.055274
7	-	-	17	0.056331	17	0.055452
8	-	-	20	0.070842	19	0.076249
9	-	-	23	0.056379	22	0.053025
10	-	-	25	0.042342	25	0.038829

Tabla 4.5: Resultados del experimento 3.

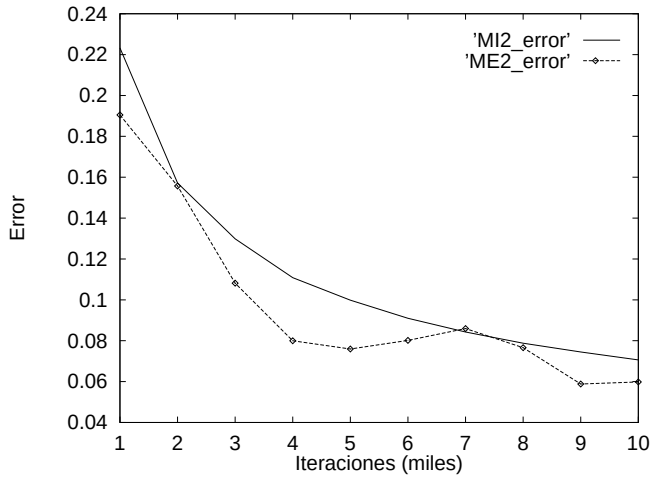

Figura 4.4: Experimento 1, MI2 frente a ME2.

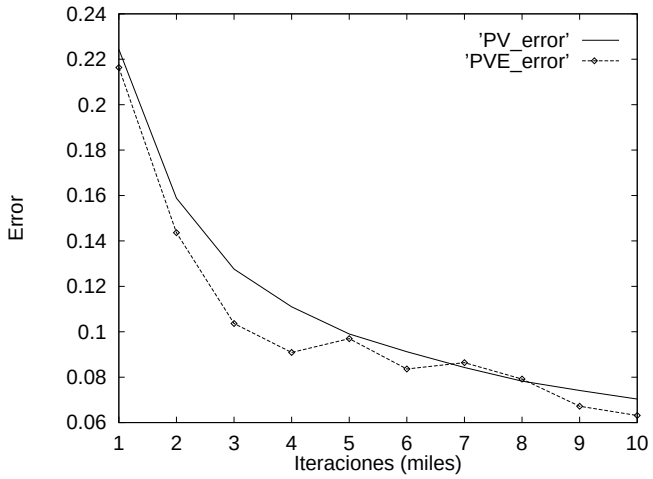

Figura 4.5: Experimento 1, PV frente a PVE.

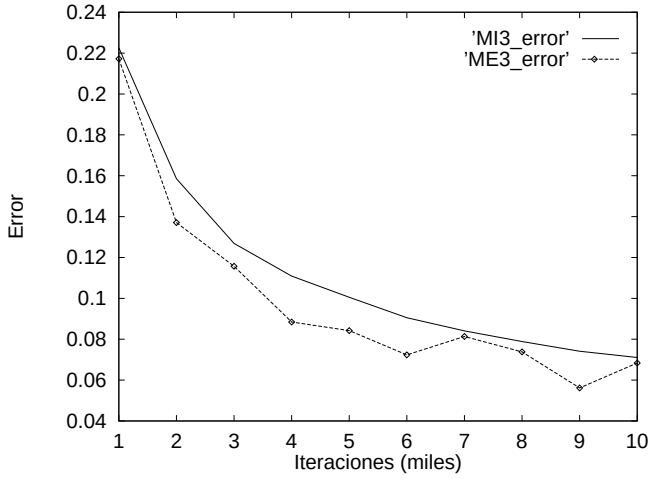

Figura 4.6: Experimento 1, MI3 frente a ME3.

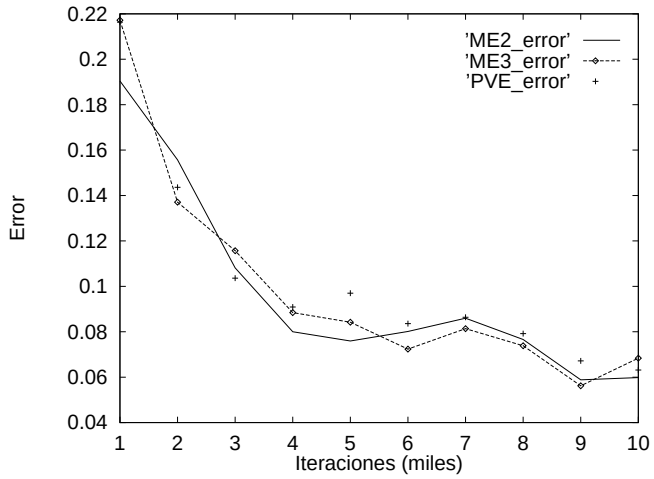

Figura 4.7: Experimento 1, Muestreo estratificado.

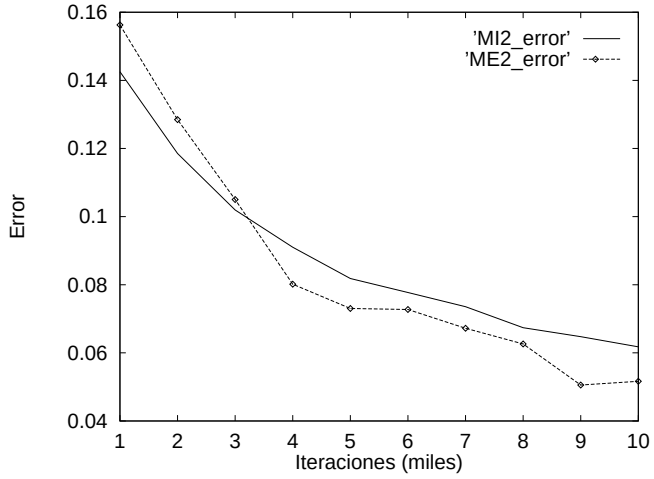

Figura 4.8: Experimento 2, MI2 frente a ME2.

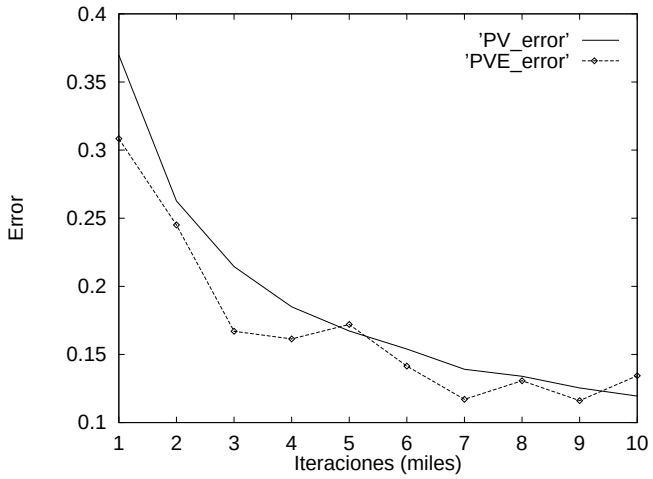

Figura 4.9: Experimento 2, PV frente a PVE.

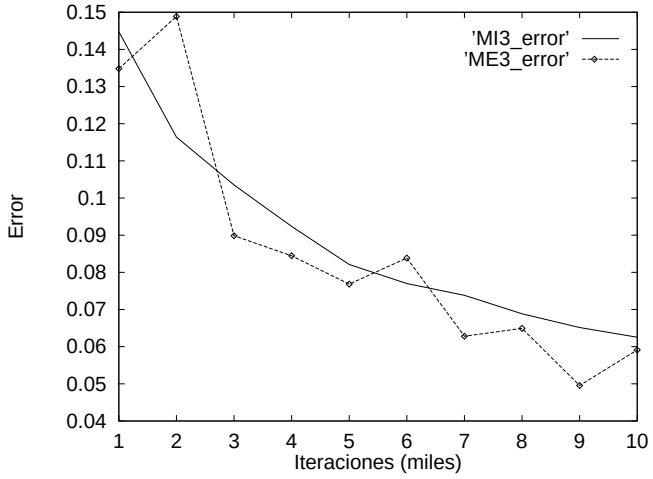

Figura 4.10: Experimento 2, MI3 frente a ME3.

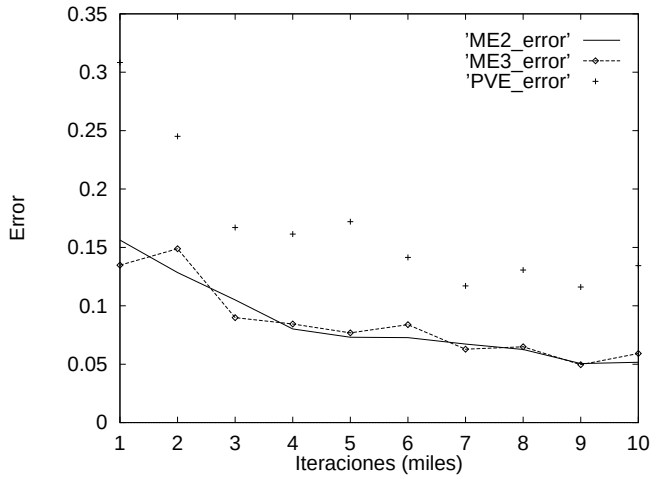

Figura 4.11: Experimento 2, Muestreo estratificado.

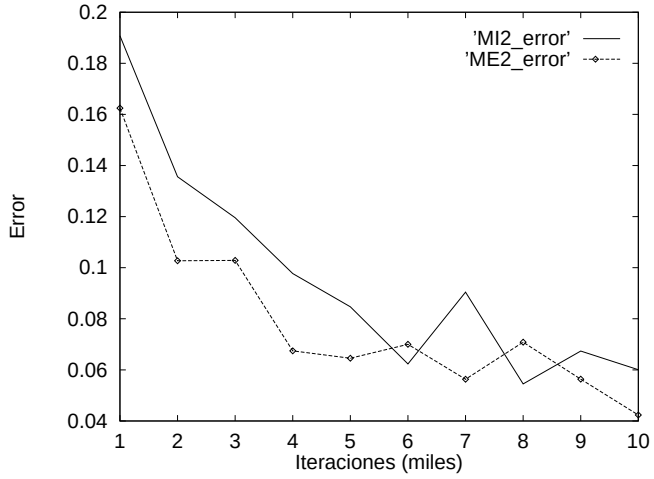

Figura 4.12: Experimento 3, MI2 frente a ME2.

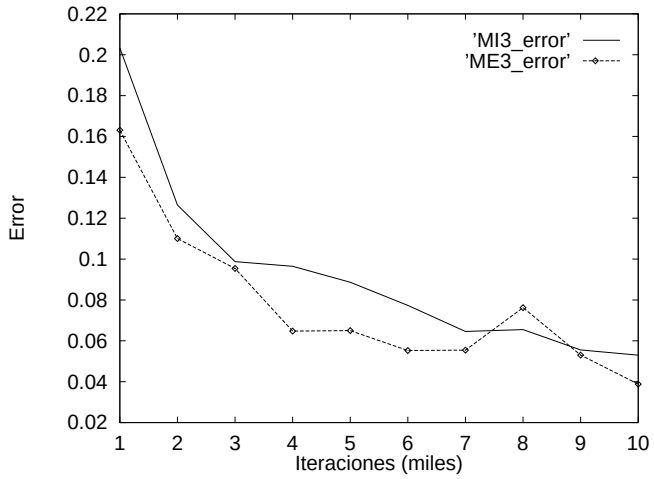

Figura 4.13: Experimento 3, MI3 frente a ME3.

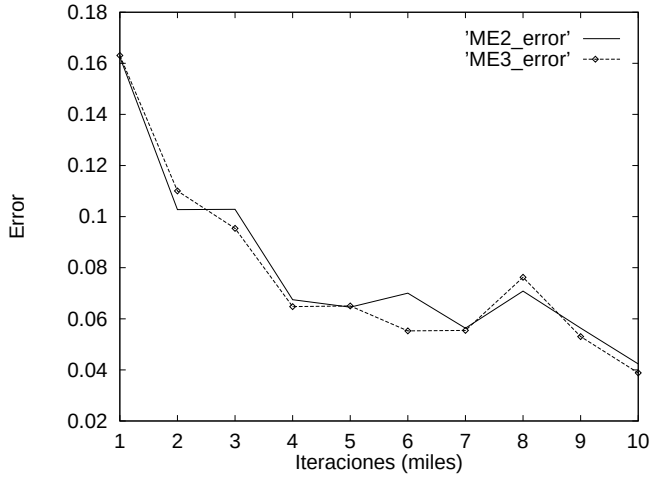

Figura 4.14: Experimento 3, Muestreo estratificado.

Capítulo 5

Algoritmos de Simulación usando Árboles de Probabilidad

5.1 Introducción

En los dos capítulos anteriores, se ha puesto de manifiesto la importancia que tiene el tamaño de los potenciales a la hora de calcular las distribuciones de muestreo. En lo visto hasta ahora, la representación que se ha utilizado para los potenciales es la de matrices de números reales, es decir, la representación clásica de las tablas de probabilidad. Una explicación clara de las tablas de probabilidad puede verse en [47].

El principal problema de las tablas es que no permiten reflejar algunas de las posibles regularidades presentes en los potenciales: en la mayoría de los casos, no permiten sacar partido de la aparición de valores repetidos. Esto sugiere el estudio de representaciones dispersas que permitan, de alguna forma, representar más información en menos espacio. Hasta el momento se han utilizado principalmente dos tipos de representaciones dispersas con este fin: bases de reglas [70, 71] y *árboles de probabilidad* [6, 9]. En general se pretende utilizar las posibles regularidades presentes en las distribuciones condicionadas de la red (*independencias basadas en el contexto* [6]) para simplificar la red de cara a facilitar los algoritmos de inferencia (ver las referencias anteriores y [96]).

Cano y Moral [9], aprovechan las cualidades de los árboles para aproximar los potenciales por otros de menor tamaño. En este capítulo propondremos una metodología similar, profundizando en algunos aspectos de los árboles que permiten mejorar significativamente la bondad de las funciones de muestreo en los algoritmos de simulación.

Comenzaremos revisando los conceptos relativos a árboles de probabilidad en la sección 5.2, estudiando la construcción de los árboles y la realización de las operaciones necesarias para la propagación haciendo uso de ellos. En la sección 5.3 estudiamos la definición de algoritmos de muestreo por importancia usando árboles en lugar de tablas, viendo qué ventajas puede aportar la nueva representación. La sección 5.4 se dedica a describir los experimentos realizados con los nuevos algoritmos y a presentar los resultados de los mismos, finalizando el capítulo con las conclusiones en la sección 5.5.

5.2 Árboles de Probabilidad

Un *árbol de probabilidad* es un árbol dirigido etiquetado en el que cada nodo interior representa una variable, y cada nodo hoja un valor de probabilidad. Cada nodo interior tendrá tantos arcos salientes como casos tenga la variable que representa. Los árboles de probabilidad se han mostrado como herramientas adecuadas para representar *independencias basadas en el contexto* (Boutilier et al. [6]):

Definición 5.1 (Independencia basada en el contexto) *Sean X, Y y C tres conjuntos disjuntos de variables. Se dice que X e Y son* independientes dado el contexto $c \in U_C$ *si $P(X|Y = y_1, C = c) = P(X|Y = y_2, C = c)$ para todo $y_1, y_2 \in U_Y$ verificando que $P(Y = y_1, C = c) > 0$ y $P(Y = y_2, C = c) > 0$.*

Ejemplo 5.1 *En la figura 5.1 se muestra una red para las variables X_1, X_2 y X_3, cada una de ellas con dos posibles valores, $X_i = 1$ ó $X_i = 2$, $i = 1, 2, 3$. La tabla representa la distribución condicionada $P(X_1|X_2, X_3)$. Puede observarse que las variables X_1 y X_3 son independientes dado el contexto $X_2 = 2$. Esta independencia es aprovechada por*

el árbol de la figura, pues representa la misma información que la tabla pero utilizando cinco valores en lugar de ocho.

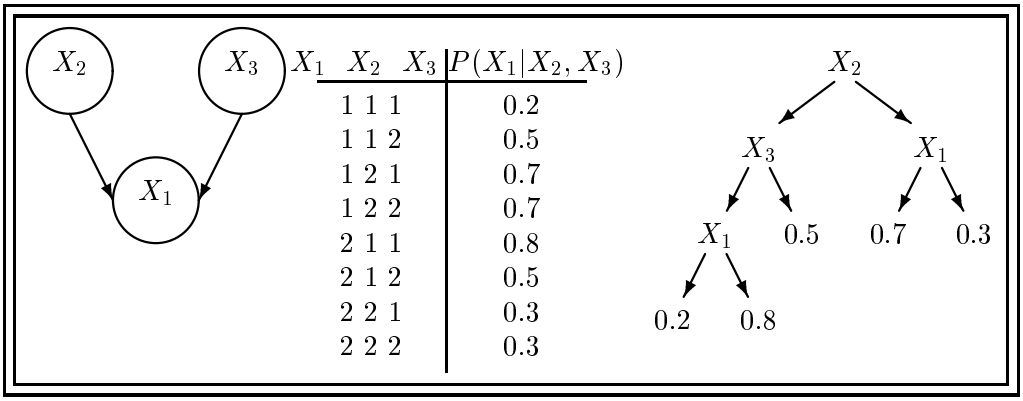

Figura 5.1: Tabla de probabilidad y su árbol asociado

Obsérvese que los árboles de probabilidad pueden utilizarse para representar cualquier potencial, y no solamente distribuciones condicionadas. Por lo tanto, podremos emplearlos para representar los potenciales que surgen en los pasos intermedios de los algoritmos de simulación, que no siempre son distribuciones de probabilidad (no están normalizados).

Definición 5.2 (Árbol asociado a un potencial) *Sea $f : U_I \to \mathbb{R}$ un potencial sobre un conjunto de variables X_I. Un árbol dirigido $\mathcal{T}_f$ se dice que es un* árbol asociado a *f si verifica las siguientes propiedades:*

1. *Cada nodo interior está etiquetado con una variable $X_i \in X_I$.*

2. *Cada nodo interior tiene tantos hijos como casos tiene la variable a la que representa, de forma que en el subárbol correspondiente al i-ésimo arco de una variable X_j representa las situaciones en que X_j toma su i-ésimo valor.*

3. *Cada nodo hoja está etiquetado con un número $r \in \mathbb{R}$ tal que $f(x_I) = r$ para todo $x_I \in U_I$ tal que las coordenadas de x_I correspondientes a las variables que*

aparecen en el camino entre la raíz del árbol y el nodo r coinciden con los valores de dichas variables indicados por los arcos del camino.

El árbol de la figura 5.1 es un árbol asociado al potencial $f(x_1, x_2, x_3) = P(X_1 = x_1 | X_2 = x_2, X_3 = x_3)$. Por ejemplo, el valor 0.2 se corresponde con $f(1, 1, 1) = P(X_1 = 1 | X_2 = 1, X_3 = 1)$, mientras que el valor 0.7 se corresponde con $f(1, 2, 1)$ y $f(1, 2, 2)$.

5.2.1 Construcción de árboles de probabilidad

Los árboles de probabilidad pueden ser de gran ayuda a la hora de la asignación de probabilidades. En general, ante un modelo en el que aparezcan variables con un alto número de padres en la red, no parece natural que el usuario tenga que establecer un valor de probabilidad para cada configuración de los padres a la hora de especificar la distribución condicionada de cierta variable. Es muy posible que muchos valores coincidan y, por lo tanto, el usuario debería tener la posibilidad de expresar solamente los valores distintos. Esto no es posible hacerlo con tablas de probabilidad, pero sí con árboles.

Por otro lado, puede ser que dispongamos de un potencial en forma de tabla y queramos representarlo mediante un árbol, o bien necesitemos representar un nuevo árbol que sea el resultado de marginalizar o combinar otros. Este es el caso en que nos centramos en esta memoria. Para abordarlo, utilizaremos la metodología de construcción y cálculo con árboles desarrollada por Cano y Moral [9]. Estos autores proponen una forma de aproximar potenciales cuando el tamaño es limitado. Esto será de gran utilidad para los algoritmos de simulación, donde, recuérdese, el tamaño de los potenciales estaba limitado.

Dado un potencial $f : U_I \rightarrow \mathbb{R}$ definido sobre un conjunto de variables X_I, el objetivo es obtener un árbol $\mathcal{T}_f$ asociado al potencial f. En el caso en que el tamaño del árbol sea mayor que el límite que tengamos fijado en el sistema, interesará obtener el árbol $\mathcal{T}$ que más se aproxime al potencial f. Si denotamos por $p_{\mathcal{T}}$ y p_f las distribuciones de probabilidad proporcionales a $\mathcal{T}$ y f respectivamente, el grado de aproximación del

árbol $\mathcal{T}$ al potencial f se mide como la entropía de Kullback-Leibler [55] de $p_{\mathcal{T}}$ a p_f:

$$D(f,\mathcal{T}) = D(p_f, p_{\mathcal{T}}) = -\sum_{x_I \in U_I} p_f(x_I) \log \frac{p_f(x_I)}{p_{\mathcal{T}}(x_I)}. \tag{5.1}$$

Definición 5.3 (Estructura) *Dos árboles $\mathcal{T}_1$ y $\mathcal{T}_2$ se dice que tienen la misma* estructura *$\mathcal{T}^*$ si contienen los mismos nodos interiores y difieren a lo más en los números situados en las hojas.*

Definición 5.4 (Restricción) *Dado un árbol $\mathcal{T}$ y un conjunto de variables X_J, se denota por $\mathcal{T}^{R(X_J=x_J)}$ a la* restricción *del árbol $\mathcal{T}$ a los valores x_J de las variables en X_J, es decir, al árbol que se obtiene sustituyendo en $\mathcal{T}$ todos los nodos correspondientes a variables $X_k, k \in J$ por los subárboles $\mathcal{T}_k$ hijos de X_k correspondientes a $X_k = x_J^{\downarrow k}$.*

Nótese que si X_J contiene todas las variables entre la raíz y una hoja, $\mathcal{T}^{R(X_J=x_J)}$ representa el valor almacenado en dicha hoja.

Proposición 5.1 *Sea $f : U_I \to \mathbb{R}$ un potencial sobre un conjunto de variables X_I. Sea $J \subseteq I$. Si $\mathcal{T}$ es un árbol que contiene el valor $\sum_{x_I^{\downarrow J} = x_J} f(x_I)/|U_{I-J}|$ en cada hoja $\mathcal{T}^{R(X_J=x_J)}$, entonces $\mathcal{T}$ minimiza la distancia de Kullback-Leibler entre todos los árboles con la misma estructura que $\mathcal{T}$ y el potencial f.*

Obsérvese que dada cualquier estructura $\mathcal{T}^*$, según la proposición anterior, el árbol $\mathcal{T}$ con dicha estructura que mejor aproxima al potencial f será aquel que en cada hoja contiene la media de los valores de f para las configuraciones de las variables que llevan a dicha hoja.

El proceso de construcción de un árbol consiste en ir eligiendo, dado un árbol $\mathcal{T}$ con estructura $\mathcal{T}^*$, qué rama se va a expandir y con qué variable. Esta elección habrá de hacerse de forma que, en cada momento, se minimice la distancia al potencial que se quiere representar. Si un nodo hoja de $\mathcal{T}^*$ está definido por los valores $X_J = x_J$,

denotamos por $\mathcal{T}^*(X_J = x_J, X_k)$ a la estructura que se obtiene de $\mathcal{T}^*$ al aumentar la hoja definida por $X_J = x_J$ con la variable X_k, y por $\mathcal{T}^*(X_J = x_J, X_k)$ al árbol correspondiente. En cada momento, elegiremos tanto la rama a expandir como la variable que minimicen la distancia a f, es decir,

$$D(\mathcal{T}(X_J = x_J, X_k), f) = \min\{D(\mathcal{T}(X_{J'} = x_{J'}, X_{k'}), f)\}, \tag{5.2}$$

donde $X_{J'} = x_{J'}$ es una hoja de $\mathcal{T}^*$ y $k' \in I - J'$.

La siguiente proposición [9] indica una forma sencilla de calcular este mínimo.

Proposición 5.2 *El par $(X_J = x_J, X_k)$ que minimiza la expresión (5.2) es aquél que maximiza la medida de información*

$$I(X_J = x_J, X_k|f) = S_{X_J=x_J} \cdot (\log|U_k| - \log S_{X_J=x_J} - E[X_k|X_J = x_J]), \tag{5.3}$$

donde

$$\begin{aligned} S_{X_J=x_J} &= \sum_{x_I^{\downarrow J}=x_J} f(x_I), \\ E[X_k|X_J = x_J] &= -\sum_{x_k \in U_k} f^{\downarrow k}(x_k|X_J = x_J) \log f^{\downarrow k}(x_k|X_J = x_J), \\ f^{\downarrow k}(x_k|X_J = x_J) &= \sum_{\substack{x_I^{\downarrow J}=x_J \\ x_I^{\downarrow k}=x_k}} f(x_I). \end{aligned}$$

La medida de información $I(X_J = x_J, X_k|f)$ mide la diferencia entre la distancia del árbol $\mathcal{T}$ al potencial f antes y después de expandir la rama $X_J = x_J$ con la variable X_k. La interpretación de la proposición anterior es que en cada momento deben elegirse hojas de las que cuelgue una suma de probabilidades alta y variables con entropía pequeña. La razón es que al expandir de esta manera se obtendrán hojas con valores de probabilidad bien diferenciados, y recordemos que el objetivo es representar las probabilidades distintas, ya que las que sean muy parecidas podrán aproximarse simplemente por un promedio.

Con esto, el algoritmo para construir un árbol de forma **exacta** consistiría en elegir los nodos uno a uno según maximicen la medida I. La construcción terminaría cuando para cada rama $X_J = x_J$ los valores del potencial f fueran uniformes, es decir, $f(x_I) = f(x'_I)$ para todo $x_I, x'_I \in U_I$ tales que $x_I^{\downarrow J} = x'^{\downarrow J}_I = x_J$. Es decir, va añadiendo nodos hasta que la inclusión de más nodos no aporte nueva información.

Para la construcción de un árbol **aproximado**, Cano y Moral [9] proponen distintas opciones. Supondremos que tenemos un número de nodos por árbol limitado. Una de las opciones consiste en ir añadiendo nodos y terminar el proceso de construcción cuando el límite prefijado se alcance. La segunda opción consiste en construir el árbol completo y después podarlo. Si $\mathcal{T}$ es el árbol en cuestión, una *poda* consiste en seleccionar un nodo tal que todos sus hijos sean hojas y sustituirlo por el promedio de sus hijos, que también son eliminados del árbol. Tenemos dos formas de realizar la poda:

1. Si consideramos un número máximo de nodos, se van eliminando nodos mientras dicho límite se exceda. En cada caso, la selección del nodo a podar vendrá determinada por la elección del par $(X_J = x_J, X_k)$ que minimice la medida $I(X_J = x_J, X_k|f)$, es decir, el par que minimice el incremento de la distancia respecto al potencial f.

2. En otro caso, se eliminan todos los nodos determinados por $(X_J = x_J, X_k|f)$ tales que
$$I(X_J = x_J, X_k|f) \leq \delta, \tag{5.4}$$
donde δ es cierto umbral para el incremento de distancia. El proceso de poda terminaría cuando ningún nodo cumpliera la condición (5.4).

5.2.2 Operaciones con árboles de probabilidad

Tres son las operaciones sobre potenciales que requieren los algoritmos de propagación: *restricción, combinación y marginalización.* Los primeros autores en estudiar cómo realizar estas operaciones con árboles fueron Cano y Moral [9], proponiendo los algoritmos que describiremos en esta sección.

La operación más sencilla es la *restricción*, que ya hemos descrito en la definición 5.4. Pasamos pues a estudiar la *combinación y marginalización*.

Dado un árbol $\mathcal{T}$ asociado a un potencial $f : U_I \to I\!R$ sobre un conjunto de variables X_I, denotamos por $\mathrm{Var}(\mathcal{T})$ al conjunto de variables que etiquetan los nodos interiores de $\mathcal{T}$. Se debe cumplir que $\mathrm{Var}(\mathcal{T}) \subseteq X_I$.

Dados dos árboles $\mathcal{T}_1$ y $\mathcal{T}_2$ asociados a los potenciales f_1 y f_2 respectivamente, el siguiente algoritmo calcula el árbol asociado al potencial $f = f_1 \cdot f_2$.

COMBINA($\mathcal{T}_1$,$\mathcal{T}_2$)

---

1. Crear un nodo $\mathcal{T}$ de árbol inicialmente sin etiqueta.
2. Seleccionar una variable $X_i \in \mathrm{Var}(\mathcal{T}_1) \cup \mathrm{Var}(\mathcal{T}_2)$.
3. Añadir a la lista $\mathcal{L}_c$ el nodo $\{\{\mathcal{T}_1, \mathcal{T}_2\}, \mathcal{T}, X_i\}$.
4. Mientras $\mathcal{L}_c$ no esté vacía,
   (a) Borrar un nodo $\{\{\mathcal{T}_i, \mathcal{T}_j\}, \mathcal{T}_r, X_k\}$ de $\mathcal{L}_c$.
   (b) Poner X_k como etiqueta de $\mathcal{T}_r$.
   (c) Para cada $x_k \in U_k$,
      i. Crear un nodo $\mathcal{T}_h$ de árbol inicialmente sin etiqueta, haciendo que sea hijo del nodo $\mathcal{T}_r$.
      ii. Sean L_i y L_j las etiquetas de los nodos $raiz(\mathcal{T}_i^{R(X_k=x_k)})$ y $raiz(\mathcal{T}_j^{R(X_k=x_k)})$.
      iii. Si L_i y L_j son números, sea $L = L_i \cdot L_j$. Poner L como etiqueta de T_h.
      iv. En caso contrario,
         A. Seleccionar una variable $X_l \in \mathrm{Var}(\mathcal{T}_i^{R(X_k=x_k)}) \cup \mathrm{Var}(\mathcal{T}_j^{R(X_k=x_k)})$.
         B. Añadir a $\mathcal{L}_c$ el nodo $\{\{\mathcal{T}_i^{R(X_k=x_k)}, \mathcal{T}_j^{R(X_k=x_k)}\}, \mathcal{T}_h, X_l\}$.

5. Devolver $\mathcal{T}$.

Dado un árbol $\mathcal{T}$ asociado a un potencial $f : U_I \to \mathbb{R}$ definido sobre un conjunto de variables X_I, el siguiente algoritmo calcula el árbol asociado al potencial $f^{\downarrow(I-\{i\})}$, con $i \in I$. Es decir, elimina la variable X_i.

MARGINALIZA($\mathcal{T}$,X_i)

---

1. Sea L la etiqueta del nodo $raiz(\mathcal{T})$.
2. Si L es un número,
   (a) Crear un nodo $\mathcal{T}_r$ de árbol con etiqueta igual al valor $L \cdot |U_i|$.
3. En caso contrario, sea $X_k = L$.
   (a) Si $X_k = X_i$, entonces $\mathcal{T}_r$ =**SUMA_HIJOS($\mathcal{T}$,X_i)**.
   (b) En caso contrario,
      i. Crear un nodo $\mathcal{T}_r$ de árbol con etiqueta igual al valor L.
      ii. Para cada $x_k \in U_k$,
         A. Poner el árbol $\mathcal{T}_h$ =**MARGINALIZA($\mathcal{T}^{R(X_k=x_k)}$,X_i)** como hijo número k de $\mathcal{T}_r$.
4. Devolver $\mathcal{T}_r$.

Este algoritmo hace uso de la función **SUMA_HIJOS($\mathcal{T}$,X_i)**, que sustituye el nodo X_i, que debe ser raíz del árbol $\mathcal{T}$, por la suma de los subárboles hijos de X_i. La función es como sigue:

SUMA_HIJOS($\mathcal{T}$,X_k)

---

1. Crear un nodo $\mathcal{T}_r$ de árbol inicialmente sin etiqueta.
2. Sean $L_1, \ldots, L_j$ las etiquetas de los nodos raíz de $\mathcal{T}^{R(X_k=x_{k_1})}, \ldots, \mathcal{T}^{R(X_k=x_{k_j})}$, con $j = |U_k|$.
3. Si $L_1, \ldots, L_j$ son todas números, poner el valor $L = L_1 + \cdots + L_j$ como etiqueta de $\mathcal{T}_r$.
4. En caso contrario
   (a) Seleccionar una variable $X_i \in \mathrm{Var}(\mathcal{T}^{R(X_k=x_{k_1})}) \cup \cdots \cup \mathrm{Var}(\mathcal{T}^{R(X_k=x_{k_j})})$.
   (b) Añadir a la lista $\mathcal{L}_m$ el nodo $\{\{\mathcal{T}^{R(X_k=x_{k_1})}, \cdots, \mathcal{T}^{R(X_k=x_{k_j})}\}, \mathcal{T}_r, X_i\}$.
   (c) Mientras $\mathcal{L}_m$ no esté vacía,
      i. Borrar un nodo $\{\{\mathcal{T}_1, \ldots, \mathcal{T}_h\}, \mathcal{T}_s, X_l\}$ de $\mathcal{L}_m$.
      ii. Poner X_l como etiqueta de $\mathcal{T}_s$.
      iii. Para cada $x_l \in U_l$,
         A. Crear un nodo $\mathcal{T}_t$ de árbol inicialmente sin etiqueta y ponerlo como hijo de $\mathcal{T}_s$.
         B. Sean $L_1, \ldots, L_h$ las etiquetas de los nodos raíz de $\mathcal{T}_1^{R(X_l=x_l)}, \ldots, \mathcal{T}_h^{R(X_l=x_l)}$.
         C. Si $L_1, \ldots, L_h$ son todas números, poner como etiqueta de $\mathcal{T}_t$ el valor $L = L_1 + \cdots + L_h$.
         D. En caso contrario

- Elegir una variable $X_m \in \text{Var}(\mathcal{T}_1^{R(X_l=x_l)}) \cup \cdots \cup \text{Var}(\mathcal{T}_h^{R(X_l=x_l)})$.
- Añadir a $\mathcal{L}_m$ el nodo $\{\{\mathcal{T}_1^{R(X_l=x_l)}, \ldots, \mathcal{T}_h^{R(X_l=x_l)}\}, \mathcal{T}_t, X_m\}$.

5. Devolver $\mathcal{T}_r$.

En los algoritmos anteriores se ha hecho uso de dos listas, $\mathcal{L}_c$ para la combinación y $\mathcal{L}_m$ para la suma de árboles en la marginalización. En el caso de $\mathcal{L}_c$, cada nodo $\{\{\mathcal{T}_1, \mathcal{T}_2\}, \mathcal{T}, X_i\}$ representa dos subárboles que deben ser combinados almacenando el resultado en $\mathcal{T}$ y poniendo como etiqueta del nodo raíz del resultado la variable X_i. De forma análoga, cada nodo $\{\{\mathcal{T}_1, \ldots, \mathcal{T}_j\}, \mathcal{T}, X_i\}$ de $\mathcal{L}_m$ contiene j subárboles que han de sumarse almacenando el resultado en el árbol $\mathcal{T}$ y etiquetando el nodo raíz del resultado con la variable X_i.

Las operaciones de combinación y marginalización descritas anteriormente, son operaciones exactas, es decir, no limitan el tamaño del árbol resultante. Cano y Moral [9] proponen también métodos para efectuar dichas operaciones de forma aproximada. La idea fundamental consiste en ir colocando en los niveles superiores del árbol resultado las variables más informativas. En cada paso, se expande el árbol por las ramas que más decrementen el error, es decir, la distancia al potencial resultado. La forma de realizar esto en los algoritmos se refleja en la variable que se inserta en las listas $\mathcal{L}_c$ y $\mathcal{L}_m$: ésta será aquella que aporte más información de entre las variables raíces de los árboles a operar. El hecho de elegir sólo entre las raíces se debe a que si los árboles están bien construidos, las raíces serán las variables más informativas. El problema está en calcular el valor de información de una cierta variable X_k en una rama $X_J = x_J$, ya que esto implicaría conocer el potencial resultado, que precisamente es lo que estamos calculando. Estos autores hacen una estimación de dicha información de la siguiente forma:

1. **Combinación.** Sean f_1 y f_2 los potenciales a combinar. Se mide la información

de una variable X_k en la rama $X_J = x_J$ como:

$$I'(X_J = x_J, X_k | f_1 \cdot f_2) = I(X_J = x_J, X_k | f_1^{R(X_J=x_J)^{\downarrow k}}) I(X_J = x_J, X_k | f_2^{R(X_J=x_J)^{\downarrow k}}). \tag{5.5}$$

2. **Marginalización.** Sea $f : U_I \to \mathbb{R}$ el potencial a marginalizar sobre $I - \{i\}$. Se mide la información de una variable X_k en la rama $X_J = x_J$ como:

$$I'(X_J = x_J, X_k | f^{\downarrow I-\{i\}}) = I(X_J = x_J, X_k | f^{R(X_J=x_J)^{\downarrow k}}). \tag{5.6}$$

Las opciones para construir, de acuerdo con lo dicho, resultados aproximados, son dos:

- Limitar el número de nodos, de forma que, una vez alcanzado el umbral, las ramas que queden por expandir (que vienen dadas por los elementos de las listas $\mathcal{L}_c$ o $\mathcal{L}_m$) se aproximan por la media de los valores correspondientes a esa rama.
- Generar el árbol completo y podarlo posteriormente. En este caso, la poda se realiza como se explicó para la construcción del árbol. Ésta es la opción adoptada en este capítulo.

5.3 Muestreo por Importancia usando Árboles

Una vez definidas las operaciones con los árboles, estamos en condiciones de formular los algoritmos de los capítulos anteriores con esta nueva representación. Los árboles son una representación de los potenciales bastante sofisticada y complicada en su funcionamiento en comparación con las tablas de probabilidad. Este incremento de la complejidad puede verse suficientemente justificado si la red es suficientemente grande y, por lo tanto, se hace necesario el aproximar los potenciales resultantes de las operaciones de combinación y marginalización por otros más pequeños que los exactos. Por lo tanto, en lo que queda de este capítulo supondremos que el objetivo es propagar sobre redes de tamaño considerable. Esto mismo motiva el hecho de que nos centremos

en el estudio de los algoritmos de muestreo por importancia, dejando los de muestreo estratificado. Como se vio en el capítulo anterior, estos últimos son menos estables que los de muestreo por importancia salvo para redes pequeñas, además de que al ser los intervalos muy pequeños, la probabilidad de que una misma configuración se corresponda con varios números generados es muy pequeña, lo que hace que la ventaja en tiempo de los algoritmos estratificados desaparezca. En cualquier caso, la forma de utilizar árboles en estos algoritmos es totalmente análoga al caso del muestreo por importancia.

Las diferencias entre usar árboles o tablas se reflejan en los algoritmos en la fase de cálculo de las distribuciones de muestreo. En concreto, cambiaría la forma del proceso de eliminación aproximada de variables descrito en el capítulo 3 sección 3.5. Aquí surgen diferentes alternativas entre las que elegir. Una primera podría ser la siguiente. A la hora de eliminar una variable, combinar todos los potenciales definidos para esa variable, limitando siempre el resultado de la combinación al límite de tamaño que se haya establecido. Formalmente, el algoritmo de eliminación de una variable $X_{\sigma(i)}$ sería:

1. Sea $H(i) = \{h \in H \mid \sigma(i) \in s(h)\}$, el conjunto de potenciales definidos para la variable $X_{\sigma(i)}$. Eliminar $H(i)$ de H.
2. Mientras haya más de un potencial en $H(i)$,
   (a) Tomar $h_1, h_2 \in H(i)$.
   (b) Calcular $h = h_1 \cdot h_2$ limitando su tamaño al umbral establecido.
   (c) Reemplazar en $H(i)$, h_1 y h_2 por h.
3. Calcular $H^+(i)$ a partir de $H(i)$ eliminando $X_{\sigma(i)}$ en el único potencial perteneciente a $H(i)$.
4. Añadir $H^+(i)$ a H.

Sin embargo, después de comprobar el funcionamiento de este método en diferentes

redes, se puso de manifiesto que las estimaciones obtenidas eran sensiblemente peores que en el caso de las tablas de probabilidad.

Por ejemplo, la tabla 5.1 compara los resultados obtenidos en el experimento 2 del capítulo 2 con los que se obtienen aplicando el algoritmo de eliminación que acabamos de describir (referenciado en la tabla como MI_arb). El tamaño tanto de las tablas como de los árboles estaba limitado también a 512 valores.

Sólo para redes muy grandes y aumentando considerablemente el tamaño de los potenciales se consiguió un nivel de exactitud similar con tablas y árboles. La tabla 5.2 muestra los errores obtenidos por el algoritmo MI_arb frente al muestreo por importancia usando tablas para una red de 441 variables con 166 observaciones (la misma que se usará en la sección 5.4). En este caso, se fijó un tamaño máximo por potencial de 10.000 valores, consiguiéndose mejorar los resultados, a costa de un mayor coste computacional. Sin embargo, los resultados tampoco llegan a ser tan destacados como para justificar un método con cálculos tan elaborados.

Esto parece ir en contra de la intuición, pero un análisis detallado del funcionamiento de los algoritmos nos da la respuesta.

El punto clave a la hora de calcular las distribuciones de muestreo radica en la aproximación del producto de los potenciales asociados a la variable a eliminar. Supongamos que queremos eliminar la variable X_i y que ésta tiene asociados los potenciales $h_1, \ldots, h_n$. La forma exacta de realizar la eliminación sería calculando $h = (h_1 \otimes \cdots \otimes h_n)^{\downarrow (s(h_1) \cup \cdots \cup s(h_n)) - \{i\}}$. Si el tamaño es demasiado grande, lo que los métodos del capítulo 3 hacen es aproximar el potencial h como $h_1^{\downarrow s(h_1) - \{i\}} \otimes \cdots \otimes h_n^{\downarrow s(h_n) - \{i\}}$, mientras que usando árboles lo que haríamos es quedarnos con una versión reducida del árbol correspondiente a $(h_1 \otimes \cdots \otimes h_n)^{\downarrow (s(h_1) \cup \cdots \cup s(h_n)) - \{i\}}$. Obsérvese que esta última aproximación puede ser peor que la anterior si la diferencia entre el tamaño del árbol exacto y el aproximado es muy grande. Ahora aproximamos con un solo potencial, mientras que antes aproximábamos con un producto de n potenciales. Los experimentos demuestran que la capacidad de aproximar de los árboles no siempre compensa la capacidad de aproximar por un producto de potenciales.

Iterac. (miles)	PV	MI2	MI3	MI_arb
1	0.369696	0.142527	0.144746	0.344398
2	0.262667	0.118531	0.116398	0.282703
3	0.214533	0.101944	0.103525	0.248798
4	0.184992	0.091004	0.092368	0.230496
5	0.167224	0.081832	0.082110	0.212632
6	0.154013	0.077728	0.076983	0.190480
7	0.139159	0.073540	0.073808	0.181474
8	0.133997	0.067369	0.068852	0.178238
9	0.125444	0.064750	0.065158	0.163637
10	0.119467	0.061767	0.062544	0.158363

Tabla 5.1: Un solo árbol (MI_arb) frente a producto de tablas (i).

Iterac. (miles)	MI2	MI_arb
1	0.472952	0.522174
2	0.323111	0.357822
3	0.283678	0.273013
4	0.255418	0.259047
5	0.218868	0.216198
6	0.199451	0.192787
7	0.187956	0.181863
8	0.180048	0.168992
9	0.170438	0.165658
10	0.148126	0.159673

Tabla 5.2: Un solo árbol (MI_arb) frente a producto de tablas (ii).

Por lo tanto, para que el uso de árboles en lugar de tablas proporcione beneficios, es necesaria una aplicación más cuidadosa de éstos. Dado que los métodos para tablas ofrecen buenos resultados, el camino a seguir debería ser funcionar de forma parecida a los algoritmos para tablas, y en el momento adecuado, sacar partido de las ventajas de los árboles.

En concreto los puntos a tener en cuenta son los siguientes:

- Dada la importancia de reducir todo lo posible el tamaño de los potenciales, cada vez que se cree uno nuevo se podará según la fórmula (5.4) para cierto δ. Esto permite eliminar hojas muy parecidas cambiándolas por su media, de forma que quedará más espacio libre para almacenar valores de probabilidad extremos. La ventaja de cara a los algoritmos de simulación es que ésta aproximación producirá pesos más uniformes.

- A la hora de calcular la distribución de muestreo para cierta variable, en lugar de combinar los árboles de forma aproximada, se usan los árboles sin combinar como se hacía en el caso de las tablas. Sin embargo, a la hora de eliminar la variable lo que hacemos es combinarlos todos, marginalizar y luego podarlos para que estén dentro del límite de tamaño.

La forma de calcular el umbral δ de información por debajo del cual se eliminará una variable, es el mismo utilizado en [9]. Dado cierto parámetro ϵ, se calcula δ como la entropía de la distribución $(0.5-\epsilon, 0.5+\epsilon)$, es decir, aquella que se obtiene "moviéndonos" en una cantidad ϵ de la uniforme para una variable binaria, con $0 < \epsilon < 0.5$. Es decir,

$$\delta = -((0.5-\epsilon)\cdot\log(0.5-\epsilon) + (0.5+\epsilon)\cdot\log(0.5+\epsilon)). \tag{5.7}$$

Supongamos ahora que T es un conjunto de árboles asociados a ciertos potenciales. En estas condiciones, el algoritmo de eliminación de una variable puede enunciarse como sigue.

ELIMINA(T,X_i,δ)

---

1. Sea $T(i) = \{\mathcal{T} \in T \mid X_i \in \text{Var}(\mathcal{T})\}$, el conjunto de árboles definidos para la variable X_i. Eliminar $T(i)$ de T. Sea $T^+(i) = T(i)$.

2. Mientras haya dos árboles $\mathcal{T}_1$ y $\mathcal{T}_2$ en $T(i)$ que puedan ser combinados sin sobrepasar el límite de tamaño,

   (a) Calcular $\mathcal{T}$ =**COMBINA**($\mathcal{T}_1$,$\mathcal{T}_2$).

   (b) Podar $\mathcal{T}$ de acuerdo al valor de información δ.

   (c) Reemplazar en $T(i)$, $\mathcal{T}_1$ y $\mathcal{T}_2$ por $\mathcal{T}$.

3. Mientras haya más de un árbol en $T^+(i)$,

   (a) Tomar $\mathcal{T}_1, \mathcal{T}_2 \in T^+(i)$.

   (b) Calcular $\mathcal{T}$ =**COMBINA**($\mathcal{T}_1$,$\mathcal{T}_2$).

   (c) Reemplazar en $T^+(i)$, $\mathcal{T}_1$ y $\mathcal{T}_2$ por $\mathcal{T}$.

4. Sea $\mathcal{T}$ el único árbol perteneciente a $T^+(i)$.

5. Calcular $\mathcal{T}'$ =**MARGINALIZA**($\mathcal{T}$,X_i).

6. Podar $\mathcal{T}'$ hasta que se encuentre dentro del límite de tamaño, eliminando, en cada momento, la rama que produzca un menor aumento de la distancia al potencial representado.

7. Añadir $T^+(i)$ a T.

Una vez establecida la forma de eliminación de las variables, podemos formular el algoritmo general de muestreo por importancia como sigue. Partimos de una red G formada por las variables $X_1, \ldots, X_n$, y un conjunto de observaciones E.

Algoritmo MI_A

1. Sea $H = \{f_i \mid i = 1, \ldots, n\}$ el conjunto de potenciales formado por las distribuciones condicionadas de la red G. Sea $T = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ el conjunto de árboles asociados a los potenciales de H.
2. Elegir un orden σ para las variables de la red.
3. Incorporar las observaciones:
   (a) Restringir todos los árboles en T a las evidencias $\{e_l\}_{l \in E}$, es decir, $\mathcal{T}_i = \mathcal{T}_i^{R(X_E = e_E)}$, $i = 1, \ldots, n$.
   (b) Para cada variable observada X_l, $l \in E$ hacer $T = T \cup \{\mathcal{T}_{\delta_{e_l}}\}$, donde $\mathcal{T}_{\delta_{e_l}}$ es el árbol asociado al potencial δ_{e_l}.
4. Desde $i = 1$ hasta n,
   (a) **ELIMINA**(T,$X_{\sigma(i)}$,δ).
5. Desde $j = 1$ hasta m (tamaño de la muestra),
   (a) $w_j = 1.0$.
   (b) Desde $i = n$ hasta 1,
      i. Obtener un valor para $X_{\sigma(i)}$, $x_i^{(j)}$, de acuerdo con $N(h_i')$, donde $N(h_i')$ es el potencial normalizado asociado al árbol $\mathcal{T}_i'$ definido sobre la variable $X_{\sigma(i)}$ y obtenido como $\mathcal{T}_i' = \otimes\{\mathcal{T}^{R(X_{\Sigma(i)} = x_0)} | \mathcal{T} \in T(i)\}$, donde $\Sigma(i) = \{\sigma(i+1), \ldots, \sigma(n)\}$ y x_0 es la configuración de las variables ya simuladas $(X_{\Sigma(i)})$.

ii. Calcular

$$w_j = \frac{w_j}{N(h'_i)(x_i^{(j)})}.$$

(c) Calcular

$$w_j = w_j \cdot \left(\prod_{i=1}^{n} f_i(x^{(j)^{\downarrow s(f_i)}}) \right) \cdot \left(\prod_{l \in E} \delta_{e_l}(x^{(j)^{\downarrow l}}) \right).$$

6. Para cada $x_k \in U_k$, $k = \{1, \ldots, n\}$,

   (a) Estimar $p(x_k, e)$ usando la fórmula (3.12).

7. Normalizar los valores $p(x_k, e)$ para obtener $p(x_k|e)$.

5.4 Evaluación Experimental

Para comprobar el funcionamiento de los algoritmos se ha realizado una experimentación similar a la de los capítulos anteriores, pero en este caso sobre una red de tamaño mucho mayor. La red en cuestión representa un pedigree, y ha sido cedida por Claus S. Jensen (Universidad de Aalborg, Dinamarca). Ésta consta de 441 variables, de las cuales 166 han sido observadas. La red se utiliza para estimar la probabilidad de que cierto individuo de entre una población de cerdos tenga cierta enfermedad PSE hereditaria. Las características generales del modelo son:

- Máximo de 13 generaciones.
- Sólo influyen dos alelos: N y n, produciendo 3 genotipos: NN, Nn y nn.
- Se da la enfermedad si el genotipo es nn.
- Se sigue el modelo genético mendeliano (probabilidad 0.5 de heredar cada uno de los genes).

La red está formada por una serie de "familias" como la mostrada en la figura 5.2. Cada variable puede tomar tres posibles valores, NN, Nn y nn. En la red distinguiremos dos tipos de variables, las que son raíces y las que no lo son. Las probabilidades asociadas a cada uno de estos tipos de variables vienen dadas en las tablas 5.3 y 5.4. Cada variable llega a tener hasta un máximo de 43 hijos, lo que hace que el cálculo exacto de las distribuciones de muestreo sea muy costoso.

Para la experimentación se ha fijado un tamaño máximo de 512 valores por potencial y un valor $\epsilon = 0.01$, lo que resulta en un valor de información para la poda de $\delta = 0.000289$. El experimento consiste en propagar realizando 1.000 simulaciones en la primera prueba, 2.000 en la segunda y así sucesivamente hasta 10.000 en la décima. Cada prueba se ha repetido 100 veces para promediar los resultados. En cada una de ellas, se ha medido el tiempo, el error y el número de simulaciones válidas. El error se ha medido usando las fórmulas (3.20) y (3.21). En promedio, el número de simulaciones válidas para el algoritmo MI_A ha sido del 99.95%, mientras que para el MI2 ha sido del 80.84%. Los tiempos que aparecen en las tablas son tiempos de simulación. El tiempo de compilación, es decir, el dedicado a calcular las distribuciones de muestreo, ha sido de 16 segundos en el caso del MI_A y de 3 segundos para el MI2.

Las pruebas se han realizado sobre una estación de trabajo Pentium 166MHz, con 32MB de RAM y sistema operativo Solaris 2.5. Los resultados pueden verse en la tabla 5.5 y en la figura 5.3.

Atendiendo a los resultados experimentales, podemos concluir que la aplicación de árboles para representar potenciales representa un importante avance a la hora de aplicar los algoritmos de muestreo por importancia a redes de tamaño considerable. De hecho, como puede apreciarse en las tablas, para un número bajo de simulaciones el error en las estimaciones del método MI2 llega a ser inadmisible. Sin embargo, el MI_A muestra un comportamiento estable y es más rápido que el anterior, a pesar de que requiera más tiempo para calcular las distribuciones de muestreo.

5.5 Conclusiones

En este capítulo hemos presentado un método de propagación aproximada capaz de tratar con redes de gran tamaño. Se comprueba que la aplicación de árboles es ventajosa frente al uso de las tablas, dado que los árboles permiten representar mejor las partes más "importantes" de una distribución, lo que a la postre beneficia al muestreo por importancia.

Existían métodos capaces de tratar con redes muy grandes, por ejemplo el muestreo de Gibbs por bloques, debido a Jensen, Kong y Kjærulff [42]. Sin embargo, este método puede llegar a ser demasiado lento en comparación con los de muestreo por importancia.

Una cualidad importante de los árboles es su flexibilidad, lo que hace al método presentado en este capítulo susceptible de ser mejorado, por ejemplo, utilizando criterios de entropía para seleccionar las funciones que han de ser combinadas. El hecho de que podamos aproximar las zonas más "uniformes" de una distribución por un promedio para dejar más espacio para los valores más informativos, abre un abanico de posibilidades que no existía con el uso de tablas.

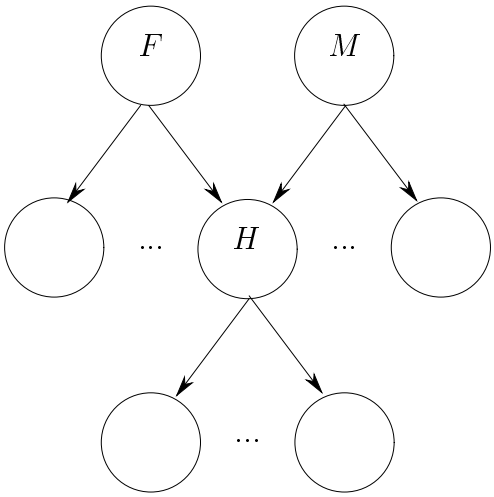

Figura 5.2: Una familia dentro de la red de pedigree.

$P(F) = P(M)$		
NN	Nn	nn
p^2	$2pq$	q^2
0.25	0.5	0.25

Tabla 5.3: Tabla de probabilidad para una variable raíz.

$P(H\|P,M)$				
		H		
P	M	NN	Nn	nn
NN	NN	1	0	0
NN	Nn	0.5	0.5	0
NN	nn	0	1	0
Nn	NN	0.5	0.5	0
Nn	Nn	0.25	0.5	0.25
Nn	nn	0	0.5	0.5
nn	NN	0	1	0
nn	Nn	0	0.5	0.5
nn	nn	0	0	1

Tabla 5.4: Tabla de probabilidad para una variable no raíz.

	ALG. MI_A			ALG MI2		
Iterac.	It. Válidas	Tiempo	Error	It. Válidas	Tiempo	Error
1000	999.59	28.43	0.480905	809.13	31.88	0.937425
2000	1998.96	59.35	0.381114	1615.74	61.15	0.629130
3000	2998.34	85.73	0.300662	2425.97	93.10	0.532136
4000	3998.10	108.36	0.252949	3231.20	125.38	0.394313
5000	4997.44	141.42	0.227796	4044.82	161.65	0.402624
6000	5996.66	172.76	0.213779	4853.02	181.44	0.361692
7000	6996.19	200.12	0.195051	5662.95	214.21	0.317355
8000	7995.31	219.12	0.171044	6464.62	257.68	0.313865
9000	8995.06	252.62	0.167659	7270.55	274.72	0.296336
10000	9994.42	285.13	0.160442	8082.92	317.00	0.279514

Tabla 5.5: Resultados experimentales.

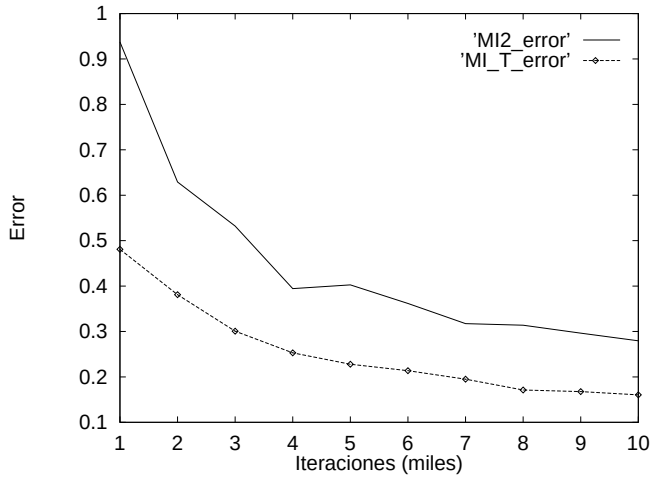

Figura 5.3: MI_A vs. MI2.

Capítulo 6

Esquema HUGIN para Propagación de Funciones de Creencia

6.1 Introducción

En los capítulos anteriores hemos estudiado la propagación de información probabilística en grafos de dependencias. Sin embargo, la probabilidad no es el único formalismo utilizado para representar la incertidumbre sobre un modelo. La Teoría de la Evidencia de Dempster-Shafer [25, 82] ofrece un marco de trabajo ampliamente utilizado en la actualidad, y puede ser interpretada como una forma más general de asignar probabilidades, haciendo estas asignaciones a conjuntos en lugar de a átomos del espacio muestral [93].

La propagación de funciones de creencia en grafos de dependencias ha sido ampliamente estudiada en la literatura, habiéndose definido axiomáticas que determinan la aptitud de cada formalismo para valerse de las técnicas de propagación local. Destacan los trabajos de Shafer y Shenoy [78, 87] y de Cano, Delgado y Moral [11]. Mejorando los esquemas citados, Jensen et al. [45] desarrollaron el esquema implementado en la herramienta de desarrollo de sistemas expertos HUGIN [2], inicialmente pensada para probabilidades. Sin embargo, no se había desarrollado una axiomática para este es-

quema que permitiera que se usase para Teoría de la Evidencia. Lauritzen y Jensen [60] demostraron que la arquitectura HUGIN es válida para las funciones de creencia de Dempster-Shafer.

En este capítulo entramos en detalle en el desarrollo de una arquitectura tipo HUGIN para funciones de creencia, en la que cabe destacar la posibilidad de realizar la propagación completa, incluyendo divisiones, sin necesidad de transformar las masas en comunalidades, con las consiguientes ventajas a nivel de eficiencia [74].

La estructura subyacente en los cálculos es la de *representación por semi retículo* (RSR), muy similar a los *árboles jerárquicos* [26, 75].

El capítulo comienza con un repaso a los conceptos básicos de funciones de creencia multivariantes y con el planteamiento del problema a resolver en la sección 6.2. Algunos métodos conocidos de propagación de funciones de creencia se describen en la sección 6.3, pasando a estudiar en detalle la arquitectura HUGIN con funciones masa en la sección 6.4. Los aspectos concernientes a la representación de funciones de creencia se tratan en la sección 6.5, estudiando en detalle una estructura basada en los árboles jerárquicos de Sandri [26, 75], y la forma de realizar las operaciones sobre esa estructura. El cálculo con funciones de creencia representadas por RSR se describe en la sección 6.6, finalizando el capítulo con las conclusiones en la sección 6.7.

6.2 Funciones de Creencia Multivariantes

En esta sección repasamos en concepto de *función de creencia multivariante*, fijamos la notación y describimos el problema a resolver en este capítulo.

Dada una variable X_i denotamos por U_i su *espacio de estados*, es decir, el conjunto de todos los posibles valores de X_i. A los elementos de un espacio de estados U_i se les llama *configuraciones* de la variable X_i. En todo este capítulo consideraremos espacios de estados finitos. Dado un conjunto de variables X_I, U_I denota el producto cartesiano de los espacios de estados de todas las variables con índice en I: $U_I = \prod_{i \in I} U_i$.

Supongamos que X_I y X_J son conjuntos de variables, con $I \subseteq J$, y x_J es una configuración de las variables de X_J. Entonces, $x_J^{\downarrow I}$ es la *proyección* de x_J a I, i.e. la configuración de U_I obtenida a partir de x_J eliminando las coordenadas que no estén en I. Si $A \subseteq U_J$, la proyección de A a I se define como $A^{\downarrow I} = \{x^{\downarrow I} | x \in A\}$. Dado $B \subseteq U_I$. La *extensión* de B a J se define como su *conjunto cilindro*: $B^{\uparrow J} = B \times U_{J-I}$.

Una exposición detallada de la Teoría de Dempster-Shafer se encuentra en [25, 82].

El concepto básico es el de *función masa* o *asignación básica de probabilidad.*

Definición 6.1 (Función masa) *Sea X_I un conjunto de variables. Una* función masa *sobre X_I es una aplicación $m : 2^{U_I} \to [0, 1]$ que verifica*

$$m(\emptyset) = 0, \tag{6.1}$$

$$\sum_{A \subseteq U_I} m(A) = 1. \tag{6.2}$$

El conjunto $\Gamma \subseteq 2^{U_I}$ tal que $m(A) \neq 0$ para todo $A \in \Gamma$ se denomina soporte *de la función de creencia representada por m. A cada elemento de Γ se le llama* elemento focal.

A partir de la función masa se pueden definir las medidas de credibilidad y plausibilidad asociadas.

Definición 6.2 (Función de credibilidad) *Sea X_I un conjunto de variables. Una* función de credibilidad *es una aplicación $Bel : 2^{U_I} \to [0, 1]$ definida como*

$$Bel(A) = \sum_{B:B \subseteq A} m(B) \quad \forall A \subseteq U_I. \tag{6.3}$$

Definición 6.3 (Función de plausibilidad) *Sea X_I un conjunto de variables. Una* función de plausibilidad *es una aplicación $Pl : 2^{U_I} \to [0, 1]$ definida como*

$$Pl(A) = \sum_{B:B \cap A \neq \emptyset} m(B) \quad \forall A \subseteq U_I. \tag{6.4}$$

Definición 6.4 (Función de comunalidad) *Sea X_I un conjunto de variables. Una* función de comunalidad *es una aplicación $Q : 2^{U_I} \to [0,1]$ definida como*

$$Q(A) = \sum_{B:B\supseteq A} m(B) \quad \forall A \subseteq U_I. \tag{6.5}$$

Cualquiera de las representaciones anteriores son equivalentes, pudiéndose, a partir de cualquiera de ellas, obtener el resto de las funciones [90]:

$$\begin{aligned} Bel(A) &= 1 - Pl(U_I - A), \quad \forall A \subseteq U_I, && (6.6)\\ m(A) &= \sum_{B:B\supseteq A} (-1)^{|B-A|} Q(B), && (6.7)\\ m(A) &= \sum_{B:B\subseteq A} (-1)^{|A-B|} Bel(B). && (6.8) \end{aligned}$$

Desde ahora hablaremos de funciones de creencia en un sentido general, que pueden venir representadas por medio de una función de comunalidad, una función masa, una función de credibilidad o una función de plausibilidad. Cualquiera de estas representaciones contiene la misma información, y es una cuestión de conveniencia cuál de ellas se emplea. Sin embargo, la representación mediante funciones masa puede ser la más compacta. El caso extremo es cuando $Q(A) = 1$ para todo $A \in \Gamma = 2^{U_I}$, que puede representarse mediante una función masa con un solo elemento focal, a saber, $\Theta = \bigcup_{A\in\Gamma} A = U_I$, para el cual $m(\Theta) = 1$.

Sean dos funciones de creencia m_1 y m_2 definidas sobre un mismo conjunto de variables X_I y con soporte Γ_1 y Γ_2 respectivamente. La información aportada por ambas funciones puede combinarse mediante la *regla de combinación de Dempster* como sigue:

$$m_{12}(A) = (m_1 \otimes m_2)(A) = K_{12}^{-1} \sum_{\substack{A_1\in\Gamma_1, A_2\in\Gamma_2\\ A_1\cap A_2 = A}} m_1(A_1)m_2(A_2) \text{ para } \emptyset \neq A \in 2^{U_I}, \tag{6.9}$$

donde K_{12} es una constante de normalización,

$$K_{12} = \sum_{\substack{A_1 \in \Gamma_1, A_2 \in \Gamma_2 \\ A_1 \cap A_2 \neq \emptyset}} m_1(A_1) m_2(A_2). \tag{6.10}$$

Si las masas a combinar no están definidas sobre el mismo conjunto de variables, antes de aplicar la fórmula anterior hay que extenderlas un mismo conjunto de variables. Por ejemplo, si m_1 está definida sobre X_{I_1}, y m_2 sobre X_{I_2}, habría que extender ambas masas al conjunto de variables X_I, con $I = I_1 \cup I_2$. La *extensión* de una función masa m sobre un conjunto de variables X_I a un conjunto X_J con $I \subseteq J$, se define como sigue:

$$m^{\uparrow J}(A) = \begin{cases} m(B) & si\ A = B^{\uparrow J}, \quad B \in 2^{U_I} \\ 0 & en\ otro\ caso \end{cases} \qquad \forall A \in 2^{U_J}. \tag{6.11}$$

Si ignoramos la constante de normalización, y denotamos por Q_m la función de comunalidad correspondiente a la función masa m, la regla de combinación de Dempster puede escribirse en términos de las comunalidades como sigue [60]:

$$(Q_{m_1} \otimes Q_{m_2})(A) = Q_{m_1}(A) \cdot Q_{m_2}(A) \text{ para } A \neq \emptyset. \tag{6.12}$$

Supongamos ahora que tenemos un conjunto de funciones de creencia $m_1, \ldots, m_n$ definidas, respectivamente, sobre los conjuntos de variables $X_{I_1}, \ldots, X_{I_n}$, y que estos conjuntos están organizados en un árbol de grupos (ver sección 2.3.2 del capítulo 2).

La función de creencia universal, m, definida sobre el conjunto X_I con $I = I_1 \cup \cdots \cup I_n$ es la combinación de las funciones de creencia anteriores, es decir, $m = m_1 \otimes \cdots \otimes m_n$. Nuestro objetivo es obtener la *marginalización* de esta función de creencia universal en cada uno de los grupos del árbol de grupos, es decir, $m^{\downarrow I_i}$, $i = 1, \ldots, n$, donde

$$m^{\downarrow I_i}(A) = \sum_{\substack{B \subseteq U_I \\ B^{\downarrow I_i} = A}} m(B) \quad \text{para todo} \quad A \subseteq U_{I_i}. \tag{6.13}$$

Al proceso de obtener las marginales de la función de creencia universal se le llama *propagación.*

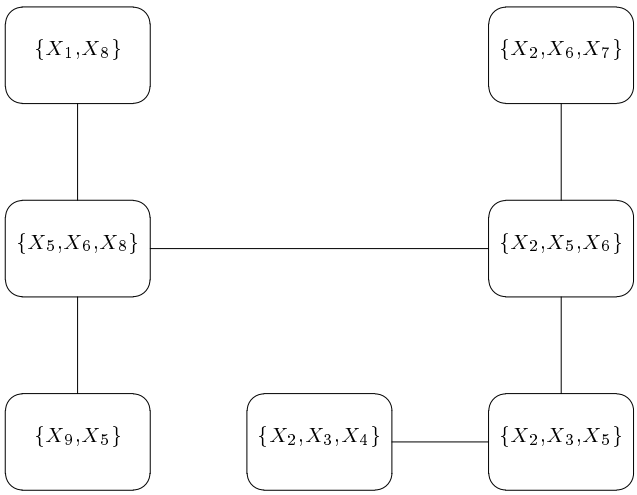

Figura 6.1: Árbol de unión para el conjunto de variables X_I, $I = \{1, 2, \ldots, 9\}$.

6.3 Métodos de Propagación Estándar

En toda esta sección trabajaremos con un árbol de grupos $\mathcal{T}$ representando a un conjunto de variables X_I con $I = I_1 \cup \cdots \cup I_n$, y donde cada X_{I_i} lleva asociada una función de creencia m_i. Un ejemplo de árbol de grupos puede verse en la figura 6.1. La función de creencia universal sobre X_I se denotará por $m = m_1 \otimes \cdots \otimes m_n$.

La tarea de calcular de forma directa la marginal de m en cada I_i presenta el problema de que el conjunto U_I puede ser muy grande, con lo que dicho cálculo puede ser difícil o prácticamente imposible, mientras que los cálculos con las funciones definidas

sobre los conjuntos X_{I_i} pueden ser mucho más asequibles. Este último caso es conocido como *cálculo local*. En esta sección describimos dos arquitecturas diferentes pero relacionadas, que permiten calcular las marginales de la función universal mediante cálculos locales. La primera de ellas es conocida como arquitectura de Shafer-Shenoy, que ha sido estudiada en un contexto muy amplio [87], y la otra es la arquitectura HUGIN, introducida inicialmente para propagación de probabilidades [45] en el sistema HUGIN [2], y que recientemente ha sido extendida a formalismos más generales por Lauritzen y Jensen [60]. Ambos esquemas conllevan la realización de sucesivas operaciones entre los nodos del árbol de unión, proceso que es conocido como *paso de mensajes*. Las diferencias entre ambas radican en la forma de los mensajes y también en la planificación de los mismos.

6.3.1 La arquitectura Shafer-Shenoy

En este esquema, se sitúan dos buzones en cada arco del árbol de unión. Dado un arco entre los nodos X_{I_i} y X_{I_j}, uno de los buzones es para los mensajes que salen de X_{I_i} y se dirigen a X_{I_j}, y el otro es para los mensajes que van en dirección contraria. Los mensajes alojados en ambos buzones serán funciones de creencia definidas sobre $X_{I_i \cap I_j}$. Inicialmente, todos los buzones están *vacíos*, y una vez que se ha introducido un mensaje en uno de ellos, se dirá que éste está *lleno*.

Un nodo X_{I_i} del árbol de grupos podrá mandar un mensaje a otro nodo vecino X_{I_j} si y sólo si todos los buzones correspondientes a mensajes entrantes a X_{I_i} están llenos excepto aquél correspondiente al mensaje de X_{I_j} a X_{I_i}. Por lo tanto, inicialmente sólo los nodos hoja pueden enviar mensajes. El mensaje de X_{I_i} a X_{I_j} se calcula como

$$\phi_{X_{I_i} \to X_{I_j}} = \left\{ m_i \otimes \left(\bigotimes_{X_{I_k} \in ve(X_{I_i}) - X_{I_j}} \phi_{X_{I_k} \to X_{I_i}} \right) \right\}^{\downarrow I_j}, \tag{6.14}$$

donde m_i es la función de creencia inicial de X_{I_i}, $\phi_{X_{I_k} \to X_{I_i}}$ son los mensajes en los buzones salientes de X_{I_k} y entrantes en X_{I_i}, y $ve(X_{I_i})$ son los nodos vecinos de X_{I_i}.

Nótese que un mensaje contiene la información procedente de un lado del árbol y es enviada al otro lado del árbol. Puede probarse [87] que siempre existe al menos un nodo que puede enviar un mensaje hasta que todos los buzones se encuentren llenos, y que cuando finaliza el paso de mensajes, para cada nodo $X_{I_i} \in \mathcal{T}$ se cumple que

$$m^{\downarrow I_i} = m_i \otimes \left(\bigotimes_{X_{I_k} \in ve(X_{I_i})} \phi_{X_{I_k} \to X_{I_i}} \right). \tag{6.15}$$

6.3.2 La arquitectura HUGIN

Este esquema ya ha sido descrito para el caso de probabilidades en la sección 2.3.2 del capítulo 2, por lo que nos limitaremos a trasladarla al caso de funciones de creencia.

En este método, en lugar de colocar buzones en los arcos del árbol de unión, entre cada par de nodos vecinos X_{I_i} y X_{I_i}, se inserta un nuevo nodo. Tal nodo es de la forma $S_{ij} = X_{I_i} \cap X_{I_j}$ y se denomina *separador*. El conjunto de todos los separadores en el árbol $\mathcal{T}$ se denota por $\mathcal{S}$. En estas condiciones, en cualquier momento la función universal sobre X_I se factoriza como

$$m = \frac{\bigotimes_{i=1}^{n} m_i}{\bigotimes_{S \in \mathcal{S}} m_S}, \tag{6.16}$$

donde m_S representa la función de creencia definida sobre el separador S. Inicialmente, las funciones de creencia asignadas a los separadores son la identidad, es decir, aquella función de creencia cuya masa es 1 para U_I y 0 en otro caso, o, lo que es lo mismo, aquella cuya comunalidad es constantemente igual a 1. La masa correspondiente a esta identidad la denotaremos por 1_I. Por tanto, en este caso la expresión (6.16) es claramente la función de creencia universal en X_I.

En el esquema HUGIN, el paso de mensajes se realiza en dos etapas. En primer lugar, un nodo ha de ser seleccionado como raíz del árbol, y a continuación, los mensajes comienzan a enviarse empezando por las hojas y en dirección a la raíz. Cuando un

nodo recibe mensajes de todos sus vecinos excepto de aquel en dirección a la raíz, éste puede mandar a su vez un mensaje hacia arriba. El proceso continúa hasta que el nodo raíz recibe mensajes de todos sus hijos. A esta fase se la llama de RECOLECCIÓN.

Después de esto, el nodo raíz envía un mensaje a cada uno de sus hijos. Luego, cada nodo que recibe un mensaje envía otro a cada uno de sus vecinos excepto a aquél del que recibió el mensaje, continuando este proceso hasta que se alcanzan los nodos hoja. A esta fase se la llama de DISTRIBUCIÓN.

A diferencia de la arquitectura de Shafer-Shenoy, en el caso de HUGIN sólo hay un tipo de mensaje, que se define como sigue. Siempre que un mensaje es enviado desde X_{I_i} a X_{I_j} con separador S_{ij}, las funciones de creencia de los nodos involucrados cambian de la siguiente forma:

$$m_i^* = m_i, \quad m_{S_{ij}}^* = m_i^{\downarrow I_i \cap I_j}, \quad m_j^* = m_j \otimes (m_{S_{ij}}^* \otimes m_{S_{ij}}^{-1})^{\uparrow I_j}, \tag{6.17}$$

donde $m_{S_{ij}}^{-1}$ es el inverso de $m_{S_{ij}}$ respecto a la operación de combinación.

Siguiendo las mismas demostraciones que en [45], puede probarse que después de las fases de RECOLECCIÓN y DISTRIBUCIÓN, cada nodo y cada separador contiene la marginal, en las variables que lo forman, de la función de creencia universal sobre X_I.

6.4 Propagación HUGIN con Funciones Masa

Obsérvese que en la expresión (6.17), a diferencia del esquema Shafer-Shenoy, hay que realizar una división; más concretamente, se combina una masa con el inverso de otra masa. Sin embargo, la operación inverso para las funciones masa no la hemos definido aún. Esta sección se dedica a buscar una expresión adecuada para tal inverso.

Una opción que podríamos elegir para definir el inverso es usar las funciones de comunalidad. La expresión (6.12) puede tomarse como base para tal definición, proporcionando, a su vez, una forma sencilla de calcularlo.

Definición 6.5 *Dado un conjunto de variables X_I y Q una función de comunalidad sobre X_I, se define el* inverso *de la función Q como*

$$Q^{-1}(A) = \begin{cases} \dfrac{1}{Q(A)} & \text{si } Q(A) \neq 0 \\ & \forall A \subseteq U_I. \\ 0 & \text{en otro caso} \end{cases} \tag{6.18}$$

Lauritzen y Jensen [60], aclaran que los inversos calculados de acuerdo a la expresión anterior pueden dar como resultado funciones que no sean funciones de creencia en el sentido de que pueden producir masas negativas. Sin embargo, las marginalizaciones siempre se realizan sobre funciones de creencia válidas, y se cumple que, después de una propagación completa, y usando el inverso definido en (6.18), las funciones que quedan tanto en los grupos como en los separadores de $\mathcal{T}$ son funciones de creencia válidas, y, además, éstas son las marginales de la función de creencia universal m. Esto asegura la validez de la propagación tipo HUGIN con funciones de creencia.

De cualquier forma, como dijimos anteriormente, las funciones masa son generalmente representaciones más compactas que las comunalidades, y por lo tanto, preferibles de cara a la representación en un ordenador. Definiremos ahora una expresión para el inverso de una función masa. A continuación, demostraremos que ambas definiciones, la realizada a partir de la comunalidad y la realizada a partir de la masa, son equivalentes.

Definición 6.6 (Clausura por intersecciones) *Dado un conjunto Γ, se define su* clausura por intersecciones, *$\tilde{\Gamma}$, como*

$$\tilde{\Gamma} = \{B | B = \bigcap_{A \in H} A, \quad H \subseteq \Gamma\}. \tag{6.19}$$

Definición 6.7 (Inverso) *Sea m una función masa sobre un conjunto de variables X_I, y con soporte $\Gamma \subseteq 2^{U_I}$. Sea $\tilde{\Gamma}$ la clausura para intersecciones de Γ. Definimos el* inverso

de m como una función $m^{-1} : 2^{U_I} \to \mathbb{R}$ con elementos focales en $\tilde{\Gamma}$ con la siguiente expresión,

$$m^{-1}(A) = \begin{cases} \dfrac{1}{\sum\limits_{\substack{B\in\Gamma \\ B\supseteq A}} m(B)} - \sum\limits_{\substack{B\in\tilde{\Gamma} \\ B\supsetneq A}} m^{-1}(B), & \text{si } A \in \tilde{\Gamma}, \\ 0, & \text{si } A \notin \tilde{\Gamma}. \end{cases} \tag{6.20}$$

Nótese que si A es maximal en $\tilde{\Gamma}$, $m^{-1}(A) = 1/m(A)$.

Los siguientes resultados demuestran que esta definición es equivalente a la anterior; es decir, (6.20) calcula realmente la masa asociada a Q^{-1} definida por (6.18). Antes, hemos de fijar alguna notación. Por $Q_{m_1 \cdot m_2}$ denotamos la función de comunalidad correspondiente a la combinación de dos masas $m_1 \otimes m_2$, i.e., el producto $Q_{m_1} \cdot Q_{m_2}$.

Proposición 6.1 *Sea m una función masa sobre un conjunto de variables X_I, con soporte Γ, y $\tilde{\Gamma}$ la clausura para intersecciones de Γ, y sea m^{-1} como en la expresión* (6.20)*. Entonces,*

$$Q_{m^{-1}}(A) = \frac{1}{Q_m(A)} \quad \forall A \subseteq U_I.$$

Dem: Distinguiremos tres casos.

- Si $A \in \tilde{\Gamma}$ es maximal:

$$\begin{aligned} Q_{m^{-1}}(A) &= \sum_{B\supseteq A} m^{-1}(B) \\ &= m^{-1}(A), \end{aligned}$$

  y dado que A es maximal, esto es igual a

$$\frac{1}{m(A)} = \frac{1}{\sum\limits_{B\supseteq A} m(B)} = \frac{1}{Q_m(A)},$$

donde $B \in \Gamma$.

- Si $A \in \tilde{\Gamma}$ no es maximal:

$$\begin{aligned} Q_{m^{-1}}(A) &= \sum_{B \supseteq A} m^{-1}(B) \\ &= m^{-1}(A) + \sum_{B \supsetneq A} m^{-1}(B), \end{aligned}$$

donde $B \in \tilde{\Gamma}$. Sustituyendo el valor de $m^{-1}(A)$ de acuerdo con la ecuación (6.20), la expresión anterior puede escribirse como sigue,

$$\begin{aligned} Q_{m^{-1}}(A) &= \frac{1}{\sum\limits_{\substack{B \supseteq A \\ B \in \Gamma}} m(B)} - \sum_{\substack{B \supsetneq A \\ B \in \tilde{\Gamma}}} m^{-1}(B) + \sum_{\substack{B \supsetneq A \\ B \in \tilde{\Gamma}}} m^{-1}(B) \\ &= \frac{1}{\sum\limits_{\substack{B \supseteq A \\ B \in \Gamma}} m(B)} \\ &= \frac{1}{Q_m(A)}. \end{aligned}$$

- Si $A \notin \tilde{\Gamma}$, hemos de tener en cuenta dos casos:
  1. Supongamos que $\nexists\, B \in \tilde{\Gamma}$ tal que $A \subseteq B$. Entonces,

$$\begin{aligned} Q_{m^{-1}}(A) &= \sum_{B \supseteq A} m^{-1}(B) \\ &= \sum_{\substack{B \supseteq A \\ B \notin \tilde{\Gamma}}} m^{-1}(B) \\ &= 0, \end{aligned}$$

dado que $m^{-1}(B) = 0$ para todo $B \notin \tilde{\Gamma}$. Además,

$$Q_m(A) = \sum_{B \supseteq A} m(B)$$

$$\begin{aligned} &= \sum_{\substack{B \supseteq A \\ B \notin \tilde{\Gamma}}} m(B) \\ &= 0, \end{aligned}$$

de lo que se sigue, usando la fórmula (6.18), que $Q_m^{-1}(A) = 0$. Por lo tanto, se cumple que $0 = Q_{m^{-1}}(A) = Q_m^{-1}(A)$.

2. Supongamos que $\exists C \in \tilde{\Gamma}$ tal que $A \subseteq C$. Sea $B = \min\{C \in \tilde{\Gamma} | A \subseteq C\}$. Tal B siempre existe, porque de otra forma podríamos encontrar dos conjuntos minimales $B_1, B_2 \in \tilde{\Gamma}$ conteniendo ambos a A y verificando que $B_1 \not\subseteq B_2$ y $B_2 \not\subseteq B_1$, lo que es equivalente a decir que A está contenido en $B_1 \cap B_2$, y esto es una contradicción con el hecho de que B_1 y B_2 son minimales y $\tilde{\Gamma}$ es cerrado para la intersección. Entonces,

$$\begin{aligned} Q_{m^{-1}}(A) &= \sum_{\substack{C \supseteq A \\ C \in \tilde{\Gamma}}} m^{-1}(C) \\ &= \sum_{\substack{C \supseteq B \\ C \in \tilde{\Gamma}}} m^{-1}(C) \\ &= Q_{m^{-1}}(B) \\ &= Q_m^{-1}(B), \end{aligned}$$

lo que es cierto dado que $m^{-1}(A) = 0$ y $B \in \Gamma$. Ahora, como $m(A) = 0$, se cumple que

$$Q_m(A) = \sum_{C \supseteq A} m(C) = \sum_{C \supseteq B} m(C) = Q_m(B),$$

lo que implica que $Q_m^{-1}(A) = Q_m^{-1}(B)$, por lo tanto, $Q_{m^{-1}}(A) = Q_m^{-1}(B) = Q_m^{-1}(A)$.

□

El siguiente teorema demuestra que m^{-1} es realmente un inverso para la combinación por la regla de Dempster.

Teorema 6.1 *Sean las condiciones de la proposición 6.1. Entonces,*

$$(m \otimes m^{-1})(A) = 1_I(A), \quad \forall A \in 2^{U_I},$$

donde 1_I *es la función de creencia identidad sobre el conjunto de variables* X_I.

Dem: De acuerdo con la proposición 6.1, para todo $A \in \tilde{\Gamma}$ se cumple que

$$\begin{aligned} Q_{m \cdot m^{-1}}(A) &= Q_m(A) \cdot Q_{m^{-1}}(A) \\ &= Q_m(A) \cdot \frac{1}{Q_m(A)} \\ &= 1 \\ &= Q_{1_I}(A), \end{aligned}$$

de lo que se sigue que

$$(m \otimes m^{-1})(A) = 1_I(A) \quad \forall A \in 2^{U_I}.$$

□

De este modo hemos definido el inverso de una función masa, que no siempre será una función masa válida. Sin embargo, como aclaramos con anterioridad, después de una propagación completa, los resultados sí son masas válidas. El avance de este enfoque es que no se necesita convertir las funciones masa en comunalidades para realizar la propagación, con el consiguiente ahorro de espacio.

6.5 Representación de Funciones de Creencia

El siguiente paso a la hora de diseñar un sistema para la propagación de funciones de creencia es la elección de una representación apropiada para las mismas. El punto clave es cómo representar los conjuntos focales y cómo organizarlos para que las operaciones se realicen de forma eficiente. En esta sección estudiamos la clásica representación mediante *cadenas de bits* (RCB) y proponemos una nueva representación *dispersa* que llamamos representación por *semi retículo* (RSR) y que está basada en los *árboles jerárquicos* [26, 75].

6.5.1 Representación por cadenas de bits (RCB)

Sea un conjunto de variables X_I con un espacio de estados finito U_I. Sea $x_1, \ldots, x_n$ un orden de las configuraciones pertenecientes a U_I. Dado un conjunto $A \subseteq U_I$, se define el *índice por cadena de bits* de A, con respecto al orden anterior, como

$$I_A = \sum_{x_i \in A} 2^i. \tag{6.21}$$

Puede demostrarse que $0 \leq I_A \leq 2^n$, donde $n = |U_I|$. El índice 0 corresponde al conjunto vacío, y el 2^n a U_I. Utilizando este esquema, una función masa puede representarse como un vector indexado por los índices de cadena de bits y conteniendo, en cada posición, la masa asociada al conjunto correspondiente al índice del vector. Esta representación tiene numerosas ventajas. Por ejemplo, las operaciones de conjunto como unión e intersección se pueden implementar con operadores de bits, que son muy rápidos, y además es sencillo transformar masas en comunalidades y viceversa (ver [90, 94, 95]).

Sin embargo, existe una importante restricción; esta representación está limitada a espacios de estados pequeños. Supongamos, por ejemplo, que X_I contiene 8 variables binarias. Entonces, $|U_I| = 256$, y el número de números reales requeridos para representar una función masa sobre X_I es 2^{256}, que es un número enorme para los ordenadores habituales. Sin embargo, el número de elementos focales de una función de creencia es habitualmente pequeño con respecto al tamaño de U_I (ver [1]), lo que sugiere pensar en otras representaciones que saquen partido de este hecho. A continuación proponemos una de estas representaciones.

6.5.2 Representación por semi retículo (RSR)

En esta sección introducimos una representación gráfica dispersa para funciones reales de conjunto, y, por lo tanto, válidas para cualquier representación de las funciones de creencia. El objetivo es encontrar un esquema computacional donde las principa-

les operaciones con funciones de creencia se puedan realizar de forma eficiente. Estas operaciones son combinación e inverso, así como otras operaciones requeridas por el esquema HUGIN, como son la extensión y marginalización. En esta sección nos limitaremos a la definición de la estructura, para, más adelante, especificar cómo han de llevarse a cabo las operaciones.

La siguiente definición está pensada esencialmente para funciones masa, sin embargo, se enuncia de forma general.

Definición 6.8 (RSR) *Sea Ω un conjunto finito, $f : 2^\Omega \to \mathbb{R}$ una función de conjunto valuada en $\mathbb{R}$. Definimos la* representación por semi retículo *(RSR) de la función f como un grafo dirigido acíclico $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, verificando las siguientes propiedades:*

1. *El conjunto de nodos $V \subseteq 2^\Omega$ es el conjunto de elementos focales de f ($V = \{v \in 2^\Omega | f(v) \neq 0\}$).*

2. *Asociado a cada nodo $X \in V$, hay un número real correspondiente a $f(X)$.*

3. *Si existe un arco $(X, Y) \in E$, significa que $X \subsetneq Y$ y no hay otro $Z \in V$ tal que $X \subsetneq Z \subsetneq Y$.*

Ejemplo 6.1 *Sea f una función definida sobre un espacio $\Omega = \{a_1, a_2, a_3, a_4\}$, y cuyos elementos focales son $X_1 = \{a_1\}$, $X_2 = \{a_4\}$, $X_3 = \{a_2, a_4\}$, $X_4 = \{a_1, a_2, a_3\}$. Supongamos que los valores de la función para esos elementos focales son $f(X_1) = 0.1$, $f(X_2) = 0.1$, $f(X_3) = 0.2$, $f(X_4) = 0.3$. La figura 6.2 muestra una RSR para f.*

Supongamos ahora que queremos añadir un nuevo conjunto focal, por ejemplo $X_6 = \{a_1, a_3, a_4\}$ con $f(X_6) = 0.3$. Comenzamos examinando los nodos raíz en la RSR para ver cuáles de ellos son subconjuntos de X_6 si es que alguno lo es. Vemos que $X_1, X_2 \subsetneq X_6$. Por lo tanto, X_6 debe ser insertado en el subgrafo formado por los descendientes de X_1 y X_2, que en este caso no los hay. Así, debemos conectar X_6 a X_1 y X_2, obteniendo el diagrama de la figura 6.3.

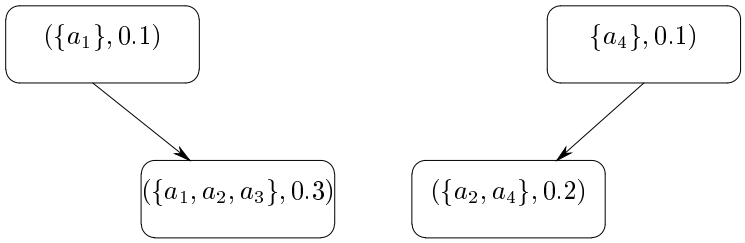

Figura 6.2: Una RSR para f.

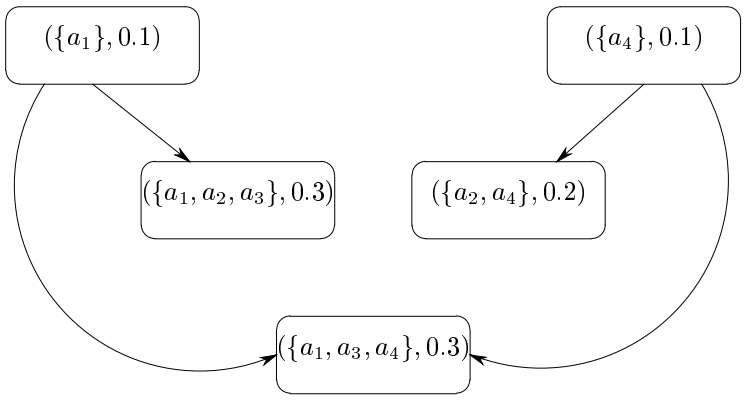

Figura 6.3: La RSR después de insertar X_6.

Cuando se construye una RSR, los conjuntos deben insertarse de tal manera que si un conjunto dado X se inserta después de otro conjunto Y, debe cumplirse que $X \not\subseteq Y$. De esta manera, se evita tener que comprobar en cada inserción cuáles deben ser los descendientes de cada nodo nuevo. Para conseguir esto, basta con establecer un orden de los elementos a insertar verificando que para cualesquiera dos conjuntos X_i, X_j, si $|X_i| < |X_j|$ entonces $i < j$.

El siguiente algoritmo describe el procedimiento de inserción, donde $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ es la RSR, X_i es el conjunto a insertar y $f(X_i)$ es el valor asociado al conjunto.

INSERTA($\mathcal{G}$,X_i,$f(X_i)$)

---

1. Sea $I = \emptyset$.
2. Para todo $X_j \in \mathcal{V}$ tal que X_j es un nodo raíz y $X_j \subseteq X_i$, hacer $I = I \cup \{j\}$.
3. Sea H el conjunto de hijos de aquellos nodos cuyos índices están en I y son subconjuntos de X_i. Borrar de I aquellos índices correspondientes a padres de los nodos de H.
4. Si $H \neq \emptyset$,
   - Para cada $X_j \in H$, hacer $I = I \cup \{j\}$.
   - Ir al paso 3.
5. Añadir X_i a $\mathcal{V}$ y conectar X_j a X_i para todo $j \in I$.
6. Asociar al nodo X_i el par $(X_i, f(X_i))$.

El conjunto I se utiliza para almacenar los índices de los nodos que han de ser conectados a X_i.

En el paso 2, el algoritmo comienza explorando los nodos raíz para ver cuáles de ellos son subconjuntos de X_i. Si encuentra alguno, actualiza el conjunto I.

A continuación, en el paso 3, la idea es buscar, de entre los descendientes de los nodos cuyos índices están en I, aquellos que son subconjuntos de X_i.

Nótese que cuando un conjunto se indexa en I, el índice de sus padres debe borrarse de I, para mantener la condición 3 de la definición 6.8.

Haciendo uso del algoritmo anterior, es fácil especificar la forma de construir una RSR correspondiente a cualquier función de conjunto valuada en $\mathbb{R}$ con soporte finito.

Sea Ω un conjunto finito, $f : 2^\Omega \to \mathbb{R}$ una función de conjunto valuada en $\mathbb{R}$. Sea $\Gamma = \{X_1, \ldots, X_n\}$ el soporte de f. Supongamos que se cumple que $|X_i| < |X_j| \Rightarrow i < j$. En estas condiciones, el siguiente algoritmo construye una RSR $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ para f.

RSR($\mathcal{G}$,f,Γ)

---

1. Sea $n = |\Gamma|$.
2. Desde $i = 1$ hasta n,
   (a) **INSERTA($\mathcal{G}$,X_i,$f(X_i)$)**

El siguiente ejemplo ilustra el proceso de construcción de una RSR.

Ejemplo 6.2 *Sea f una función real definida sobre las partes de $\Omega = \{a_1, a_2, a_3, a_4\}$ con elementos focales $X_1 = \{a_1\}$, $X_2 = \{a_4\}$, $X_3 = \{a_2, a_4\}$, $X_4 = \{a_1, a_2, a_3\}$ y $X_5 = \Omega$. Supongamos que los valores de f para dichos elementos son $f(X_1) = 0.1$, $f(X_2) = 0.1$, $f(X_3) = 0.2$, $f(X_4) = 0.3$ y $f(X_5) = 0.3$. El algoritmo comienza insertando X_1 y X_2. La inserción es directa y no hay que hacer ninguna conexión entre ellos. El siguiente conjunto a insertar es X_3. Para ello se exploran los nodos raíz, viéndose que X_2 está contenido en X_3. Luego X_3 estará conectado con X_2 (ver figura 6.4). De igual manera, se añade X_4, obteniendo la RSR de la figura 6.5. Ahora pasamos a incluir X_5. En primer lugar se examinan los nodos raíz, encontrándose que ambos son subconjuntos de X_5. Por lo tanto, la exploración continúa a partir de los hijos de X_1 y X_2, es decir, X_3 y X_4. Éstos son a su vez subconjuntos de X_5 y, dado que no hay más nodos por explorar, X_5 debe ser conectado a ambos. El resultado se muestra en la figura 6.6.*

Figura 6.4: Estado del diagrama después de insertar X_3.

Figura 6.5: Estado del diagrama después de insertar X_4.

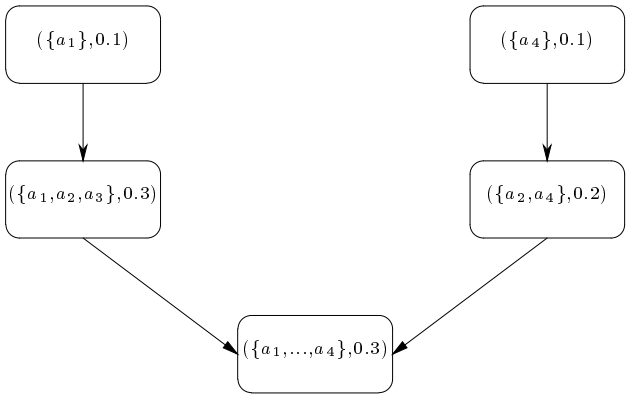

Figura 6.6: Estado del diagrama después de insertar X_5.

Ahora, la RSR debe ser extendida para manejar conjuntos de configuraciones de variables. Cada configuración puede representarse como una cadena de caracteres, reservando dos caracteres para la etiqueta de la variable y otros dos para la modalidad de la variable dentro de la configuración. Por ejemplo, supongamos que tenemos dos variables X_1 y X_2, y una configuración consistente en el primer valor de X_1 y el segundo de X_2. Esta configuración podría representarse como la cadena `X101X202`.

En lugar de repetir una configuración cada vez que ésta aparezca en un conjunto focal, todas las configuraciones podrían ser almacenadas en una tabla general, y ser referenciadas desde cualquier conjunto focal en el sistema. Esa tabla general puede implementarse como una *tabla hash*. A partir de ahora, denotaremos dicha tabla por $\mathcal{H}$.

El siguiente paso es definir cómo ha de representarse cada elemento focal de una RSR. Hay representaciones eficientes para conjuntos en general, por ejemplo, los *árboles binarios de búsqueda balanceados*, que proporcionan procedimientos eficientes de inserción de elementos o comprobación de pertenencia de elementos al conjunto por medio de funciones con complejidad $O(\log n)$ donde n es el número de elementos en el conjunto. Cada elemento, en este caso, sería simplemente un número entero referenciando a una configuración de la tabla $\mathcal{H}$. La correspondencia entre dicho número

y su configuración se define como sigue: sea N el número de celdas en la tabla $\mathcal{H}$, y x una configuración almacenada en la j-ésima posición de la i-ésima celda. Entonces, el identificador de esa configuración es el resultado de concatenar los números i y j, formando un nuevo entero ji. Obsérvese que si N es próximo al número total de configuraciones en el sistema, las operaciones de búsqueda en la tabla $\mathcal{H}$ se realizan en un tiempo constante. Una explicación detallada de este tipo de estructuras de datos puede encontrarse en [54].

6.5.2.1 Cálculo de la clausura mediante intersección de una RSR

En algunos casos, es necesario añadir una cualidad más a las RSR: deben ser cerradas para intersecciones. La razón es que el inverso de una función masa está definido sobre la clausura para intersecciones de su soporte, como se describe en la fórmula (6.20). Pasamos pues a describir cómo se calcula esta clausura para una RSR.

Teniendo en cuenta los dos hechos siguientes, la tarea puede simplificarse. En primer lugar, la intersección de un conjunto dado con cualquiera de sus subconjuntos es el propio subconjunto. Por otro lado, para cualesquiera dos conjuntos A y B, y para $C \subseteq B$, se cumple que $A \cap C \subseteq A \cap B$ y, además, $A \cap C = (A \cap B) \cap C$.

Entonces, la estrategia a seguir de cara a obtener la clausura mediante intersecciones consiste en dividir los nodos de la RSR en diferentes niveles. El primero sería aquel compuesto por los nodos hoja (aquellos que son maximales en la RSR). A continuación se calculan las intersecciones dos a dos de todos los conjuntos de ese nivel y los conjuntos que resulten se insertan en la RSR. El siguiente nivel consistirá en aquellos nodos que son antecesores directos de aquellos en el primer nivel. De nuevo se calculan las intersecciones y el proceso se repite hasta que no quedan nodos por explorar.

Hemos dicho que siempre que se calcula una nueva intersección, ésta debe insertarse en la RSR. Sin embargo, el algoritmo **INSERTA**, no es aplicable en este caso, dado que está formulado para insertar elementos que en el momento de la inserción son maximales. Por tanto, hemos de realizar algunas modificaciones para permitir la

inserción de elementos que no sean maximales. A grandes rasgos, la tarea adicional a realizar en este caso sería conectar el conjunto insertado con sus superconjuntos. El algoritmo detallado es como sigue.

Sea $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ una RSR, y $C = A \cap B$ el elemento a insertar.

INSERTA2($\mathcal{G}$,C,A,B)

---

1. Sea $I = \emptyset$.
2. Para todo $X_j \in \mathcal{V}$ tal que X_j es un nodo raíz y $X_j \subseteq C$, hacer $I = I \cup \{j\}$.
3. Sea H el conjunto de hijos de aquellos nodos cuyos índices están en I, y que son subconjuntos de C. Si alguno de ellos es igual a C, ir al paso 7. En otro caso, eliminar de I aquellos índices correspondientes a padres de los nodos de H.
4. Si $H \neq \emptyset$,
   - Para cada $X_j \in H$, hacer $I = I \cup \{j\}$.
   - Ir al paso 3.
5. Añadir C a $\mathcal{V}$.
6. Para cada X_k tal que $k \in I$, sustituir cada arco $(X_k, Y) \in \mathcal{E}$ tal que $C \subset Y$, por el par de arcos (X_k, C) y (C, Y).
7. Si A no es alcanzable desde C, añadir el arco (C, A).
8. Si B no es alcanzable desde C, añadir el arco (C, B).
9. Asociar al nodo C el valor 0.0.

Una vez redefinida la inserción, se puede especificar de forma sencilla un algoritmo para calcular la clausura para intersecciones:

CLAUSURA($\mathcal{G}$)

---

1. Sea $\mathcal{L} = \{X_1, \ldots, X_n\}$ el conjunto de hojas de $\mathcal{G}$.
2. Mientras $\mathcal{L} \neq \emptyset$,
   (a) Desde $i = 1$ hasta $n - 1$,
      i. Desde $j = i + 1$ hasta n,
         - Sea $X = X_i \cap X_j$.
         - Si $X \neq \emptyset$, **INSERTA2($\mathcal{G}$,X,X_i,X_j)**.

   (b) Sustituir $\mathcal{L}$ por el conjunto de antecesores directos de sus elementos. Sea $\mathcal{L} = \{X_1, \ldots, X_n\}$ el conjunto resultante. Ir al paso 2.

El siguiente ejemplo explica el funcionamiento de este algoritmo.

Ejemplo 6.3 *Sea la RSR de la figura 6.6. Queremos cerrarla para intersecciones utilizando el algoritmo anterior. Para ello comenzamos con las hojas, tomando* $\mathcal{L} = \{X_5\}$. *Puesto que sólo hay un elemento en* $\mathcal{L}$ *vamos al paso 2.(b), obteniendo* $\mathcal{L} = \{X_4, X_3\}$. *Ahora hemos de calcular* $X_4 \cap X_3$ *resultando* $X_6 = \{a_2\}$. *Después de insertar este conjunto según el algoritmo* **INSERTA2** *obtenemos la RSR de la figura 6.7. Ahora reemplazamos* $\mathcal{L}$ *por* $\mathcal{L} = \{X_1, X_2, X_6\}$, *comprobándose que todas las intersecciones son vacías, y, por lo tanto, no hay que insertar ningún nuevo elemento. Dado que no hay ningún antecesor de los nodos de* $\mathcal{L}$, *el algoritmo concluye y la RSR queda cerrada para intersecciones.*

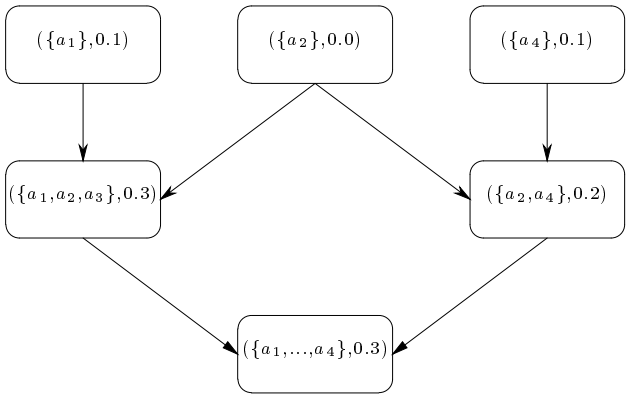

Figura 6.7: Una RSR cerrada para intersecciones.

6.6 Operaciones con Funciones de Creencia sobre la RSR

Supongamos que representamos las funciones masa mediante RSR. En esta sección describiremos cómo realizar las operaciones de la expresión (6.17) sobre esa estructura. En concreto, las operaciones son: marginalización, inverso, combinación y extensión.

Comenzaremos con la **marginalización**. Supongamos que tenemos una función masa m_I sobre un conjunto de variables X_I, representada mediante una RSR $\mathcal{G}_I$, y queremos calcular $m_J = m_I^{\downarrow J}$ con $J \subseteq I$. El proceso es sencillo: para cada elemento focal de m_I, calcular su proyección sobre J y actualizar su masa. El siguiente algoritmo calcula $\mathcal{G}_J$, una RSR para m_J.

MARGINAL($\mathcal{G}_I$,J)

---

1. Elegir un orden ancestral de los conjuntos de la RSR $\mathcal{G}_I$. Es decir, realizar un recorrido primero en anchura del grafo. Sea $\{A_1, \ldots, A_n\}$ el orden resultante.

2. Desde $i = 1$ hasta n,

   (a) **INSERTA**($\mathcal{G}_J$,$A_i^{\downarrow J}$,$m_I(A_i)$).

3. Devolver $\mathcal{G}_J$.

Obsérvese que un orden ancestral siempre verifica la propiedad requerida para aplicar el algoritmo **INSERTA**, esto es, si $A_i \subsetneq A_j$ entonces $i < j$. Si dicha propiedad se cumple en $\mathcal{G}_I$, es claro que si $A_i^{\downarrow J} \subsetneq A_j^{\downarrow J}$ entonces $i < j$, luego el orden en $\mathcal{G}_I$ es también válido para $\mathcal{G}_J$.

Un hecho debe ser considerado en este punto: si el conjunto que está siendo insertado en $\mathcal{G}_J$ es ya un elemento A_i de $\mathcal{G}_J$, entonces, en lugar de asignarle la masa $m_I(A_i^{\uparrow I})$, lo que hay que hacer es incrementar su anterior valor de masa en una cantidad $m_I(A_i^{\uparrow I})$.

La siguiente operación a describir es el **inverso**. Tenemos una masa m_I representada por una RSR $\mathcal{G}_I = (\mathcal{V}_I, \mathcal{E}_I)$, y el objetivo es calcular una RSR para m_I^{-1}. Según (6.20), si Γ es el conjunto de elementos focales de m_I, los de m_I^{-1} estarán en $\tilde{\Gamma}$. Por tanto, no es necesario calcular una nueva RSR para el inverso, pues basta con cerrar la antigua para intersecciones y actualizar los valores de masa almacenados. Pero será necesario almacenar dos valores de masa en cada nodo en lugar de uno: el valor de m_I y el de m_I^{-1}, de cara a facilitar los cálculos.

La fórmula (6.20) muestra una forma de calcular el inverso, en la cual, para cada nodo, la suma de los inversos y de las masas de todos sus superconjuntos es necesaria. Un algoritmo para hacer esto es el siguiente.

INVERSO($\mathcal{G}_I$)

---

1. **CLAUSURA**($\mathcal{G}_I$).

2. Elegir un orden ancestral de los conjuntos de $\mathcal{V}_I$. Sea $\{A_1, \ldots, A_n\}$ tal orden.

3. Desde $i = n$ hasta 1,

   (a) $a = 0.0$, $b = m_I(A_i)$.

   (b) Para cada nodo $B \in \mathcal{V}_I$ en cualquier camino desde A_i hacia arriba (i.e. $A_i \subsetneq B$),

      i. $a = a + m_I^{-1}(B)$.

      ii. $b = b + m_I(B)$.

   (c) $m_I^{-1}(A_i) = \dfrac{1}{b} - a$.

Al comenzar el proceso desde las hojas del grafo, toda la información necesaria para calcular la suma en el paso 3.(b).i está disponible.

La siguiente operación a definir es la **extensión**. Sea una RSR para una función masa m_i sobre un conjunto de variables X_{I_i}. El objetivo es obtener una RSR para la masa $m_i^{\uparrow I_j}$ sobre el conjunto de variables X_{I_j}, con $I_i \subseteq I_j$. Según la fórmula (6.11), los elementos focales de $m_i^{\uparrow I_j}$ se corresponden con los conjuntos cilindro de los elementos focales de m_i, es decir, para cada A elemento focal de m_i, su conjunto cilindro, $A^{\uparrow I_j}$, será un elemento focal de $m_i^{\uparrow I_j}$, y éstos son los únicos elementos focales de $m_i^{\uparrow I_j}$. Además, $m_i^{\uparrow I_j}(A^{\uparrow I_j}) = m_i(A)$. Por lo tanto, podemos obtener la RSR para la extensión sustituyendo cada conjunto de la RSR de m_i por su conjunto cilindro, y manteniendo el antiguo valor de masa.

Finalmente, especificaremos un algoritmo de **combinación**. Supongamos que tenemos dos funciones masa, m_1 y m_2, definidas sobre el mismo conjunto de variables X_I. Sea $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ la RSR asociada a m_1 y $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ la RSR asociada a m_2. El siguiente algoritmo calcula la RSR $\mathcal{G}_{12} = (\mathcal{V}_{12}, \mathcal{E}_{12})$ correspondiente a $m_{12} = m_1 \otimes m_2$:

COMBINA($\mathcal{G}_1$,$\mathcal{G}_2$)

---

1. Sea $\mathcal{V}_1 = \{A_1, \ldots, A_n\}$ y $\mathcal{V}_2 = \{B_1, \ldots, B_m\}$.
2. Calcular cada intersección no vacía $S = A_u \cap B_v$, $u = 1, \ldots, n$, $v = 1, \ldots, m$ y almacenarla en una lista $\mathcal{L}$ junto con el valor $f(S) = m_1(A_u) \cdot m_2(B_v)$.
3. Ordenar $\mathcal{L}$ de acuerdo al cardinal de los conjuntos que la forman. Sea $\{C_1, \ldots, C_p\}$ tal orden.
4. Desde $l = 1$ hasta p,
   (a) **INSERTA($\mathcal{G}_{12}$,C_l,$f(C_l)$)**. Si C_l ya está en la RSR $\mathcal{G}_{12}$, en lugar de asignarle un valor de masa $f(C_l)$, simplemente incrementar su antiguo valor en la cantidad $f(C_l)$.
5. Devolver $\mathcal{G}_{12}$.

Y con este algoritmo completamos las operaciones necesarias para implementar una arquitectura tipo HUGIN usando una representación dispersa. A continuación estudiaremos cómo utilizar la RSR para calcular de forma rápida valores de Bel y Q a partir de una función masa representada por una RSR.

6.6.1 Cálculo de Bel y Q sobre la RSR

En esta sección abordamos la tarea de responder a la siguiente pregunta: dado un conjunto de variables X_I, una función de creencia m_I sobre X_I y una RSR $\mathcal{G}_I$, para cada conjunto $A \subseteq U_I$, ¿Cuál es el valor de $Bel(A)$ y $Q(A)$?

La RSR proporciona una forma sencilla de responder a este tipo de consultas, simplemente realizando recorridos sobre el grafo asociado. El siguiente algoritmo calcula $Bel(A)$:

BEL($\mathcal{G}_I$,A)

---

1. Sea $\mathcal{R}$ el conjunto de nodos de $\mathcal{G}_I$ predecesores directos de A.
2. $Bel(A) = m_I(A) + \sum_{B \in \mathcal{R}} \mathbf{bPROP}(\mathcal{G}_I,B,A)$.

---

donde **bPROP**($\mathcal{G}_I$,B,A) es una función que devuelve un valor real y que está definida como

---

bPROP($\mathcal{G}_I$,B,A)

---

1. $s = m_I(B)$.
2. Marcar B como visitado.
3. Para cada nodo C padre de B en $\mathcal{G}_I$ tal que C no haya sido visitado,
   (a) $s = s$+**bPROP**($\mathcal{G}_I$,C,A).
4. Devolver s.

La forma de calcular la comunalidad es completamente análoga, pero en lugar de dirigir la propagación hacia las raíces, en este caso se dirige hacia las hojas:

Q($\mathcal{G}_I$,A)

---

1. Sea $\mathcal{L}$ el conjunto de nodos de $\mathcal{G}_I$ sucesores directos de A.
2. $Q(A) = m_I(A) \sum_{B \in \mathcal{L}}$ **qPROP**($\mathcal{G}_I$,B,A).

---

donde **qPROP**($\mathcal{G}_I$,B,A) es una función que devuelve un número real y definida como

---

qPROP($\mathcal{G}_I$,B,A)

---

1. $s = m_I(B)$.
2. Marcar B como visitado.
3. Para cada nodo C hijo de B en $\mathcal{G}_I$ tal que C no haya sido visitado,
   (a) $s = s$+**qPROP**($\mathcal{G}_I$,C,A).
4. Devolver s.

6.7 Conclusiones

En este capítulo hemos presentado una estructura sobre la cual desarrollar un esquema de propagación tipo HUGIN para funciones de creencia. Esta estructura está basada en los árboles jerárquicos [26, 75], añadiendo la característica de que esté cerrada para intersecciones en los casos necesarios (cuando se calcula el inverso de una masa).

Además, se propone una forma de calcular el inverso de una función masa, que no es necesariamente una función masa válida, pero permite realizar la propagación completa sin necesidad de transformar las masas en comunalidades para realizar las divisiones y luego volver a traducir. Se demuestra que el resultado después de una propagación completa es una función de creencia válida [60].

El esquema propuesto en este capítulo, servirá de base para los métodos aproximados del próximo capítulo.

Capítulo 7

Algoritmos Aproximados para Funciones de Creencia

7.1 Introducción

En el capítulo anterior estudiamos un esquema tipo HUGIN para propagar funciones de creencia de Dempster-Shafer. Este esquema es una forma eficiente de realizar los cálculos de forma exacta, al igual que ocurría para probabilidades, frente a otras arquitecturas como la de Shafer-Shenoy. Sin embargo, no siempre será posible hacer los cálculos de forma exacta, incluso si partimos de un árbol cluster ya construido, tal y como hacemos en el capítulo anterior. El principal problema que surge aquí, y que hace el problema de la inferencia más complicado aún que en el caso de probabilidades, es la regla de combinación de Dempster. El cálculo de la combinación de una serie de masas es de complejidad exponencial en el número de masas y el cardinal del espacio muestral [65].

En este capítulo nos proponemos el desarrollo de un método aproximado de inferencia igual al presentado en el capítulo 5 para el caso de las probabilidades. En esta ocasión, el papel que jugaban las distribuciones de probabilidad lo desempeñan las funciones masa, mientras que el correspondiente a los árboles de probabilidad es ahora

atribuido a las representaciones por semi retículo.

El capítulo comienza con un planteamiento general del problema en la sección 7.2. En la sección 7.3 se describen las nuevas operaciones que se necesitan sobre las representaciones por semi retículo, principalmente, la combinación aproximada de dos RSR. El algoritmo principal se estudia en la sección 7.4, terminando el capítulo con las conclusiones en la sección 7.5.

7.2 Planteamiento del Problema

El punto de partida ahora es algo distinto al del caso de las probabilidades. Vamos a suponer que disponemos de un conjunto de funciones masa $M = \{m_1, \ldots, m_k\}$, cada una de ellas definidas sobre un conjunto de variables X_{I_i}, $i = 1, \ldots, k$, de forma que $N = \bigcup_{i=1}^{k} I_i = \{1, \ldots, n\}$ es el conjunto de índices de todas las variables del sistema, X_N. El objetivo será obtener una estimación de la masa 'a posteriori' sobre cada una de las variables X_i, $i \in N$, es decir, la marginal $m^{\downarrow I_i}$ de la función masa universal $m = m_1 \otimes \cdots \otimes m_k$ para cada variable del sistema, de acuerdo con la expresión (6.13).

En primer lugar, necesitamos definir una operación de restricción para funciones masa. En el caso de probabilidades, la restricción es trivial, pero a la hora de definirla para funciones masa hay que ser más cuidadoso, puesto que estas últimas están definidas sobre conjuntos de configuraciones de variables en lugar de estarlo sobre configuraciones simples. La definición que proponemos es la siguiente:

Definición 7.1 (Restricción) *Sea m una función masa sobre un conjunto de variables X_I. Dado $A \subseteq U_J$ un conjunto de configuraciones para un conjunto de variables X_J, $J \subseteq I$, se define la* restricción de m a A *como*

$$m^{R(A)}(B) = \begin{cases} m(B) & \text{si} \quad B \subseteq A^{\uparrow I}, \\ \\ 0 & \text{si} \quad B \not\subseteq A^{\uparrow I}. \end{cases} \tag{7.1}$$

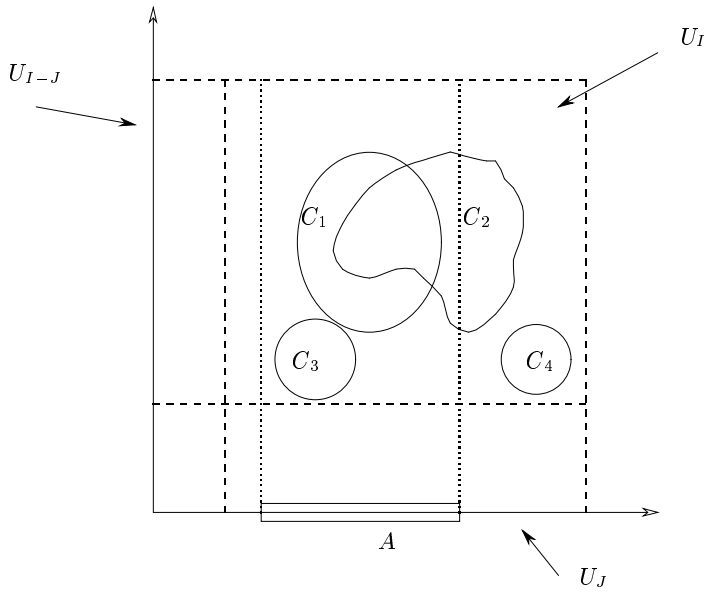

Figura 7.1: Focales para una masa sobre U_I.

Ejemplo 7.1 *Sea una masa m definida sobre un conjunto U_I con focales $C_1, \ldots, C_4$ como se muestra en la figura 7.1. Supongamos que queremos calcular la restricción $m^{R(A)}$ con $A \in U_J$. Esta nueva masa vendrá dada por aquellos focales de m que están contenidos en la extensión cilíndrica de A, es decir, $m^{R(A)}$ tendrá dos focales, C_1 y C_3, con masa $m^{R(A)}(C_1) = m(C_1)$ y $m^{R(A)}(C_3) = m(C_3)$.*

Una vez definida la restricción, es posible plantear un método similar al usado para probabilidades para estimar las masas de cada variable mediante simulación. Se pretende obtener una muestra cuyos elementos son configuraciones de conjuntos focales (un conjunto focal para cada masa) con intersección no vacía.

Lo ideal es seleccionar cada configuración $(A_1, \ldots, A_k)$ con $A_1^{\uparrow N} \cap \cdots \cap A_k^{\uparrow N} \neq \emptyset$ con una probabilidad proporcional a $m_1(A_1) \cdots m_k(A_k)$. De esta forma, podemos estimar la masa $m^{\downarrow I_i}$, $i = 1, \ldots, n$, mediante

$$\hat{m}^{\downarrow I_i}(A) = \frac{|\{(A_1, \ldots, A_k) \mid (A_1^{\uparrow N} \cap \cdots \cap A_k^{\uparrow N})^{\downarrow I_i} = A\}|}{m}, \tag{7.2}$$

donde m es el tamaño de la muestra.

Éste es un estimador insesgado de $m^{\downarrow I_i}(A)$ con varianza

$$var(\hat{m}^{\downarrow I_i}(A)) = \frac{m^{\downarrow I_i}(A)(1 - m^{\downarrow I_i}(A))}{m}. \tag{7.3}$$

En general, será difícil obtener una muestra con esta distribución de probabilidad: hay que realizar una combinación exacta. Así que el procedimiento que seguiremos será similar al que usamos en el caso de las probabilidades: calcularemos una aproximación de esta distribución mediante una precomputación rápida aunque no exacta, y usaremos dicha distribución para obtener una muestra $\{(A_1^{(i)}, \ldots, A_k^{(i)}) \mid i = 1, \ldots, M\}$. Cada elemento de la muestra, $(A_1^{(i)}, \ldots, A_k^{(i)})$, tendrá un peso asociado w_i, y la estimación habrá que realizarla de acuerdo con la media ponderada de los pesos:

$$\hat{m}^{\downarrow I_i}(A) = \frac{\sum \{w_i \mid A_1^{(i)\uparrow N} \cap \cdots \cap A_k^{(i)\uparrow N} = A\}}{\sum_{i=1}^{M} w_i}. \tag{7.4}$$

Una primera aproximación al problema viene dada por el siguiente ejemplo.

Ejemplo 7.2 *Supongamos que tenemos tres variables X_1, X_2 y X_3 y tres masas m_1, m_2 y m_3 definidas para los conjuntos de variables $\{X_1, X_2\}$, $\{X_1, X_3\}$, y $\{X_2, X_3\}$ respectivamente. Lo primero que necesitamos es un orden de eliminación de las variables, por ejemplo, X_1, X_2, X_3. En la eliminación de cada variable lo que debemos obtener es una distribución de probabilidad de muestreo que nos permita simular un focal para la masa marginal correspondiente a dicha variable. Esa distribución de muestreo debe ser consistente con la masa que queremos simular. Lo ideal sería que ambas fueran proporcionales, pero eso, en general, será inalcanzable, con lo que tendremos que quedarnos con una aproximación.*

Una vez definido el orden de eliminación de las variables, pasamos a borrarlas. El proceso de borrado consiste en tomar todas las masas definidas para la variable a eliminar y marginalizar cada función sobre las demás variables. Esto se hará habiéndolas

combinado antes o no, al igual que pasaba en el caso de las probabilidades. Veamos cómo sería:

- Eliminación de X_1. *Las funciones que están definidas para* X_1 *son* m_1 *y* m_2. *Aquí tenemos dos opciones, combinarlas o no. Esto dependerá de que el tamaño de la combinación no exceda cierto umbral que se define de antemano. Supondremos en este ejemplo que las combinamos, obteniendo una función* $m_1^* : 2^{U_{\{1,2,3\}}} \rightarrow [0,1]$ *definida como* $m_1^* = m_1 \otimes m_2$. *Ésta será la función masa que se usará para calcular la distribución de muestreo para* X_1. *Ahora hay que eliminar la variable* X_1 *de la función* m_1^* *mediante marginalización.*

- Eliminación de X_2. *Las funciones que están definidas para* X_2 *son* $m_1^{*\downarrow\{2,3\}}$ *y* m_3. *Al igual que en el paso anterior, las combinaremos. El resultado es una función* $m_2^* : 2^{U_{\{2,3\}}} \rightarrow [0,1]$ *definida como* $m_2^* = m_1^{*\downarrow\{2,3\}} \otimes m_3$. *Ahora se elimina la variable* X_2 *de la función* m_2^* *mediante marginalización.*

- Eliminación de X_3. *La única función que queda definida para* X_3 *es* $m_3^* = m_2^{*\downarrow\{3\}}$. *Por lo tanto, no tenemos que hacer nada más.*

Una vez concluido el proceso de borrado, pasamos a simular las funciones masa marginales. Lo que hacemos es ir simulando en orden inverso al de eliminación. Por lo tanto, comenzamos con X_3. *Tomamos una distribución de probabilidad* $p_3^* : 2^{U_3} \rightarrow [0,1]$ *proporcional a* m_3^*. *Usando* p_3^* *simulamos un focal* $A_3 \subseteq U_3$ *para la masa marginal de* X_3. *Ahora tenemos que simular un focal para la masa marginal de* X_2 *que sea consistente con* A_3. *Para ello, calculamos una distribución* $p_2^* : 2^{U_2} \rightarrow [0,1]$ *proporcional a* $m_2^{*R(A_3)}$ *y la usamos para obtener un focal* $A_2 \subseteq U_2$. *Ahora tenemos que simular* X_1. *Para ello tomamos una distribución de muestreo* $p_1^* : 2^{U_1} \rightarrow [0,1]$ *proporcional a* $m_1^{*R(A_2,A_3)}$ *y la usamos para obtener un focal* $A_1 \subseteq U_1$. *De esta forma, hemos simulado un focal para cada una de las masas marginales de las variables. La masa que se asignaría a la configuración de focales obtenida, vendrá dada por el peso del muestreo por importancia, es decir, el cociente entre la probabilidad de la configuración de focales y la probabilidad de muestreo de dicha configuración.*

Obsérvese que en el ejemplo anterior se ha seguido un enfoque similar al del capítulo 3, es decir, cuando se usaban tablas de probabilidad. Sin embargo, ya dijimos en el capítulo 5 que cuando el problema era de un tamaño considerable, había que adoptar soluciones más sofisticadas, como era el uso de árboles de probabilidad y el estudio de formas más cuidadosas de calcular las funciones de muestreo.

En definitiva, seguiremos un desarrollo paralelo al del capítulo 5. Por lo tanto, lo primero que necesitamos es una representación dispersa para las funciones masa, y una definición de las operaciones aproximadas sobre esa estructura. La estructura a emplear será la RSR descrita en el capítulo anterior.

7.3 Operaciones Aproximadas sobre una RSR

Las operaciones que realizaremos sobre una RSR son la *combinación*, *marginalización* y *restricción*. La marginalización nunca añade complejidad a la RSR original, por lo tanto basta con utilizar el algoritmo del capítulo anterior. Nos centraremos, pues, en la combinación aproximada y en la restricción.

7.3.1 Combinación aproximada

Por aproximar una función masa entendemos el hecho de eliminar algunos de sus focales, con objeto de conseguir una representación más compacta. Un primer enfoque a la aproximación de la combinación podría ser combinar primero y luego ir eliminando focales de la masa resultante. En este sentido, se han estudiado algunos criterios para eliminar focales de una función masa [1, 90], pero no se ha llegado a un consenso sobre cuál es mejor. Básicamente, se han seguido dos líneas a la hora de eliminar focales. La primera consiste en tomar los focales con más elementos y borrarlos, repartiendo su masa entre los demás. La segunda consiste en borrar los conjuntos con menos masa, repartiendo también la masa borrada entre los focales restantes. Como señala Almond [1], ambos enfoques plantean problemas en determinadas circunstancias.

En esta sección proponemos un enfoque diferente. La idea consiste en ir aproximando conforme se realiza la combinación. Para ello, se van seleccionando los focales maximales de cada masa y se van calculando las intersecciones y el producto de ambas masas. El conjunto resultante se va insertando en una nueva estructura junto con el valor de masa correspondiente. Si el conjunto a insertar ya existe, lo que se hace es incrementar su valor de masa. Este proceso se mantiene mientras no se exceda cierto umbral de tamaño. Cuando dicho umbral se sobrepasa, hay que reducir el tamaño de la estructura que vamos construyendo, tratando de eliminar, en cada momento, el focal que menos altere la representación de la función masa resultado de la combinación.

En concreto, lo que haremos será seleccionar parejas de conjuntos y sustituirlos por la unión de ambos, asignándole al conjunto resultante la suma de las masas de los dos que se han eliminado. De esta forma, habrá que determinar un criterio de selección de las parejas de nodos que tenga en cuenta la diferencia entre ambos conjuntos y también la masa que tiene asignada cada uno de ellos. El criterio que proponemos es elegir los nodos A_1 y A_2 que minimicen la siguiente función:

$$D(A_1, A_2) = \frac{m(A_1) \cdot |A_2 - A_1| + m(A_2) \cdot |A_1 - A_2|}{.} \max\{|A_1|, |A_2|\} \tag{7.5}$$

La razón de dividir por $\max\{|A_1|, |A_2|\}$ es para dar más importancia a la diferencia entre conjuntos pequeños. Por ejemplo, dos conjuntos que difieren en un elemento serán más "distintos" si tienen cardinal 1 que si tienen cardinal 100.

Sin embargo, la aplicación de este criterio podría llegar a ser muy costosa, dado que habría que comparar cada nodo con todos los demás. Una forma de acelerar la búsqueda de la pareja óptima es limitar el campo de búsqueda para cada nodo. Pensamos que es razonable que cada nodo sólo ha de ser comparado con los que están cerca de él en la estructura, en concreto, con su *envolvente* en el grafo asociado a la RSR, es decir, con el conjunto formado por sus padres, sus hijos y los hijos de sus padres. Para cada nodo A, denotaremos este conjunto como $env(A)$.

El cálculo de la función de coste $D()$ puede hacerse al mismo tiempo que se van

insertando elementos en el resultado de la combinación. En cada inserción, se calcula la función de coste del nodo que se inserta (X_i) respecto a cada uno de los nodos (X_j) que forman su envolvente. A continuación se inserta cada tripleta $(X_i, X_j, D(X_i, X_j))$ en una lista ordenada de mayor a menor según. De esta forma, seleccionar una pareja de nodos para reducir la RSR es simplemente tomar el primer elemento de la lista ordenada.

Aún necesitamos una cosa más para especificar un algoritmo de combinación aproximada. En concreto, necesitamos modificar el algoritmo **INSERTA2** definido en el capítulo anterior para que permita incluir elementos no maximales que no sean el resultado de la intersección de dos ya existentes. La modificación es muy sencilla. Sea $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ una RSR, y C el conjunto a insertar, con masa c.

INSERTA3($\mathcal{G}$,C,c)

---

1. Sea $I = \emptyset$.
2. Para todo $X_j \in \mathcal{V}$ tal que X_j es un nodo raíz y $X_j \subseteq C$, hacer $I = I \cup \{j\}$.
3. Sea H el conjunto de hijos de aquellos nodos cuyos índices están en I, y que son subconjuntos de C. Si alguno de ellos es igual a C, ir al paso 7. En otro caso, eliminar de I aquellos índices correspondientes a padres de los nodos de H.
4. Si $H \neq \emptyset$,
   - Para cada $X_j \in H$, hacer $I = I \cup \{j\}$.
   - Ir al paso 3.
5. Añadir C a $\mathcal{V}$.
6. Para cada X_k tal que $k \in I$, sustituir cada arco $(X_k, Y) \in \mathcal{E}$ tal que $C \subset Y$, por el par de arcos (X_k, C) y (C, Y).

7. Incrementar la masa asociada al nodo C en una cantidad c.

En estas condiciones, podemos proponer el siguiente algoritmo para la combinación aproximada de dos masas. Sean m_1 y m_2 las masas a combinar, ambas definidas sobre el mismo conjunto de variables. Si no lo están, habrá que extenderlas al conjunto unión. Sean $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ y $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ las RSR asociadas a las masas m_1 y m_2 respectivamente. El siguiente algoritmo devuelve la RSR $\mathcal{G}_{12}$ de la aproximación a la masa $m_{12} = m_1 \otimes m_2$.

COMBINA_APR($\mathcal{G}_1$,$\mathcal{G}_2$)

---

1. Sea $\mathcal{L}$ una lista, inicialmente vacía, de tripletas (A, B, c) donde A y B son conjuntos y c un valor real. Los elementos de esta lista estarán ordenados de menor a mayor según el valor de c.

2. Sean $\mathcal{V}_1 = \{A_1, \ldots, A_n\}$ y $\mathcal{V}_2 = \{B_1, \ldots, B_m\}$. Supongamos sin pérdida de generalidad que $A_u \not\subset A_v$ si $u < v$ y $B_u \not\subset B_v$ si $u < v$. Es decir, los conjuntos más grandes tienen subíndice más bajo.

3. Mientras $|\mathcal{V}_1| > 0$ y $|\mathcal{V}_2| > 0$,

   (a) Sacar el primer elemento $A \in \mathcal{V}_1$.

   (b) Para todo $B \in \mathcal{V}_2$ tal que $C = A \cap B \neq \emptyset$,

      i. **AÑADE($\mathcal{G}_{12}$,C,$m_1(A) \cdot m_2(B)$,$\mathcal{L}$)**.

   (c) Sacar el primer elemento $B \in \mathcal{V}_2$.

   (d) Para todo $A \in \mathcal{V}_1$ tal que $C = A \cap B \neq \emptyset$,

      i. **AÑADE($\mathcal{G}_{12}$,C,$m_1(A) \cdot m_2(B)$,$\mathcal{L}$)**.

4. Devolver $\mathcal{G}_{12}$.

---

donde **AÑADE** es un procedimiento que añade un conjunto a la RSR y la reduce si se excede el límite de tamaño:

AÑADE($\mathcal{G}$,A,c,$\mathcal{L}$)

---

1. **INSERTA3($\mathcal{G}$,A,c)**.
2. Para cada $B \in env(A)$,
   (a) Borrar de $\mathcal{L}$ cualquier tripleta de la forma (A, B, c) o (B, A, c).
   (b) Añadir a $\mathcal{L}$ la tripleta $(A, B, D(A, B))$.
3. Si $\mathcal{G}$ excede el límite de tamaño,
   (a) Extraer la primera tripleta (A, B, c) de la lista $\mathcal{L}$.
   (b) Sean $m(A)$ y $m(B)$ los valores de masa asociados a A y B en $\mathcal{G}$ respectivamente.
   (c) Eliminar A y B de $\mathcal{G}$.
   (d) Eliminar de $\mathcal{L}$ cualquier tripleta que contenga al conjunto A o al B.
   (e) **AÑADE($\mathcal{G}$,$A \cup B$,$m(A) + m(B)$,$\mathcal{L}$)**.

Obsérvese que el procedimiento **COMBINA_APR** realiza la combinación exacta si ésta no excede el límite de tamaño establecido.

El siguiente ejemplo ilustra el proceso de combinación aproximada.

Ejemplo 7.3 *Disponemos de dos RSR* $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ *(figura 7.2) y* $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ *(figura 7.3). Queremos obtener una aproximación de la combinación de ambas,* $\mathcal{G}_{12}$*, que no tenga más de 5 elementos. Inicialmente, tenemos que*

$$\begin{aligned} \mathcal{V}_1 &= \{\{a_1, a_2\}, \{a_1, a_3\}, \{a_1\}, \{a_2\}, \{a_4\}\}, \\ \mathcal{V}_2 &= \{\{a_1, a_2, a_3, a_4\}, \{a_2, a_3\}, \{a_3\}, \{a_4\}\}. \end{aligned}$$

Tomamos $\{a_1, a_2\}$ *de* $\mathcal{V}_1$ *y calculamos las intersecciones no vacías con elementos de* $\mathcal{V}_2$*, obteniendo*

$$\begin{aligned} &\{a_1, a_2\} \cap \{a_1, a_2, a_3, a_4\} = \{a_1, a_2\} \quad \text{con masa} \quad 0.2 \times 0.2 = 0.04, \\ &\{a_1, a_2\} \cap \{a_2, a_3\} = \{a_2\} \qquad\qquad\quad \text{con masa} \quad 0.2 \times 0.3 = 0.06. \end{aligned}$$

Añadimos estos conjuntos a la RSR $\mathcal{G}_{12}$*, obteniendo la estructura de la figura 7.4. Después de este paso, el estado de la lista* $\mathcal{L}$ *será*

$$\mathcal{L} = \{(\{a_1, a_2\}, \{a_2\}, 0.03)\},$$

puesto que

$$D(\{a_1, a_2\}, \{a_2\}) = \frac{0.04 \times 0.0 + 0.06 \times 1}{2} = 0.3.$$

Ahora tomamos $\{a_1, a_2, a_3\}$ *de* $\mathcal{V}_2$ *y, repitiendo el proceso anterior, con* $\mathcal{V}_1$*, obtenemos*

$$\begin{aligned} &\{a_1, a_2, a_3, a_4\} \cap \{a_1, a_3\} = \{a_1, a_3\} \quad \text{con masa} \quad 0.2 \times 0.3 = 0.06, \\ &\{a_1, a_2, a_3, a_4\} \cap \{a_1\} = \{a_1\} \qquad\quad \text{con masa} \quad 0.2 \times 0.1 = 0.02, \\ &\{a_1, a_2, a_3, a_4\} \cap \{a_2\} = \{a_2\} \qquad\quad \text{con masa} \quad 0.2 \times 0.2 = 0.04, \\ &\{a_1, a_2, a_3, a_4\} \cap \{a_4\} = \{a_4\} \qquad\quad \text{con masa} \quad 0.2 \times 0.1 = 0.02. \end{aligned}$$

La figura 7.5 muestra $\mathcal{G}_{12}$ *después de insertar estos conjuntos. Veamos cómo se actualiza la lista* $\mathcal{L}$*. Al insertar* $\{a_1, a_3\}$*, ésta no cambia, pues su envolvente es el conjunto*

vacío. Luego se añade $\{a_1\}$*, cuya envolvente está formada por* $\{a_1, a_3\}$*,* $\{a_1, a_2\}$ *y* $\{a_2\}$*. Insertamos las correspondientes tripletas en* $\mathcal{L}$ *y obtenemos*

$$\mathcal{L} = \{(\{a_1\}, \{a_1, a_3\}, 0.01), (\{a_1\}, \{a_1, a_2\}, 0.01), (\{a_1, a_2\}, \{a_2\}, 0.03), (\{a_1\}, \{a_2\}, 0.12)\}.$$

La inserción de $\{a_2\}$ *sólo modifica la penúltima tripleta y la de* $\{a_4\}$ *no afecta a la lista. En definitiva,*

$$\mathcal{L} = \{(\{a_1\}, \{a_1, a_3\}, 0.01), (\{a_1\}, \{a_1, a_2\}, 0.01), (\{a_1, a_2\}, \{a_2\}, 0.05), (\{a_1\}, \{a_2\}, 0.12)\}.$$

Ahora tomamos $\{a_1, a_3\}$ *de* $\mathcal{V}_1$ *y repetimos la mismo operación, obteniendo*

$$\{a_1, a_3\} \cap \{a_2, a_3\} = \{a_3\} \quad \text{con masa} \quad 0.3 \times 0.3 = 0.09,$$

$$\{a_1, a_3\} \cap \{a_3\} = \{a_3\} \qquad \text{con masa} \quad 0.3 \times 0.1 = 0.03.$$

Después de añadir dos veces el conjunto $\{a_3\}$*, obtenermos la RSR de la figura 7.6. Se observa que la envolvente de* $\{a_3\}$ *está formada por* $\{a_1, a_3\}$ *y* $\{a_1\}$*, con lo que la lista* $\mathcal{L}$ *quedará como*

$$\begin{aligned}\mathcal{L} = \; & \{(\{a_1\}, \{a_1, a_3\}, 0.01), (\{a_1\}, \{a_1, a_2\}, 0.01), (\{a_1, a_2\}, \{a_2\}, 0.05), \\ & (\{a_3\}, \{a_1, a_3\}, 0.06), (\{a_1\}, \{a_2\}, 0.12), (\{a_3\}, \{a_1\}, 0.14)\}.\end{aligned}$$

Dado que el tamaño máximo de la RSR está limitado a 5 nodos, tendremos que reducir $\mathcal{G}_{12}$*. Para ello, seleccionamos el primer elemento de la lista* $\mathcal{L}$*:* $(\{a_1\}, \{a_1, a_3\}, 0.01)$*. Eliminamos ambos conjuntos de* $\mathcal{G}_{12}$ *y borramos de* $\mathcal{L}$ *todas las tripletas en las que aparezca cualquera de ellos. A continuación, añadimos a* $\mathcal{G}_{12}$ *la unión de ambos conjuntos con masa igual a la suma de las masas de ambos. El resultado puede verse en la figura 7.7. El estado de* $\mathcal{L}$ *después de este paso es:*

$$\mathcal{L} = \{(\{a_1, a_2\}, \{a_2\}, 0.05), (\{a_1, a_3\}, \{a_3\}, 0.06)\}.$$

Los siguientes elementos a considerar son $\{a_1\}$ *de* $\mathcal{V}_1$*,* $\{a_3\}$ *de* $\mathcal{V}_2$ *y* $\{a_2\}$ *de* $\mathcal{V}_1$*, pero para todos ellos las intersecciones resultantes son vacías y no influyen, por tanto, en el resultado de la combinación.*

El siguiente es $\{a_4\}$ *de* $\mathcal{V}_2$*. Para este conjunto obtenemos*

$$\{a_4\} \cap \{a_4\} = \{a_4\} \quad \text{con masa} \quad 0.1 \times 0.4 = 0.04.$$

Añadimos este conjunto a $\mathcal{G}_{12}$ *y obtenemos la RSR de la figura 7.8. La lista* $\mathcal{L}$ *no se modifica porque la envolvente de* $\{a_4\}$ *es el conjunto vacío.*

En este momento, se tiene que $\mathcal{V}_2 = \emptyset$*, y por tanto, el algoritmo termina, dando como aproximación de la combinación la RSR de la figura 7.8.*

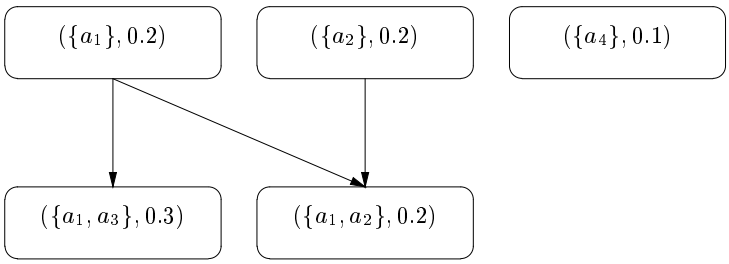

Figura 7.2: RSR $\mathcal{G}_1$.

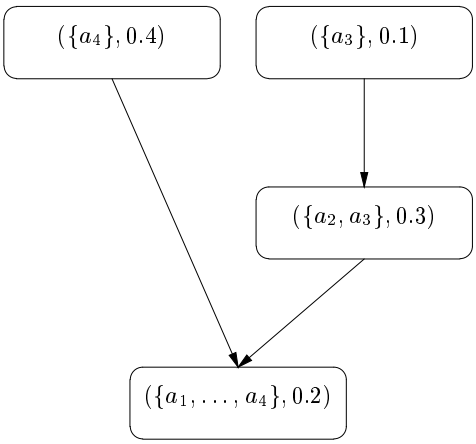

Figura 7.3: RSR $\mathcal{G}_2$.

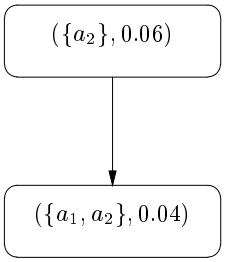

Figura 7.4: Construcción de $\mathcal{G}_{12}$ (i).

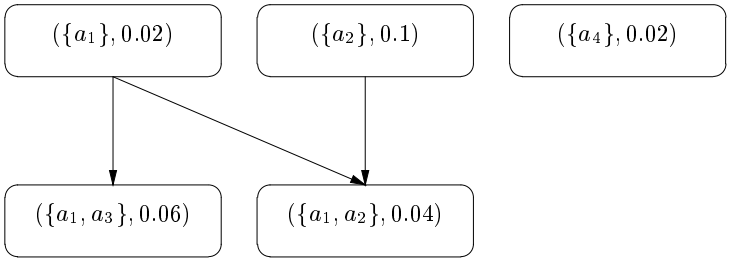

Figura 7.5: Construcción de $\mathcal{G}_{12}$ (ii).

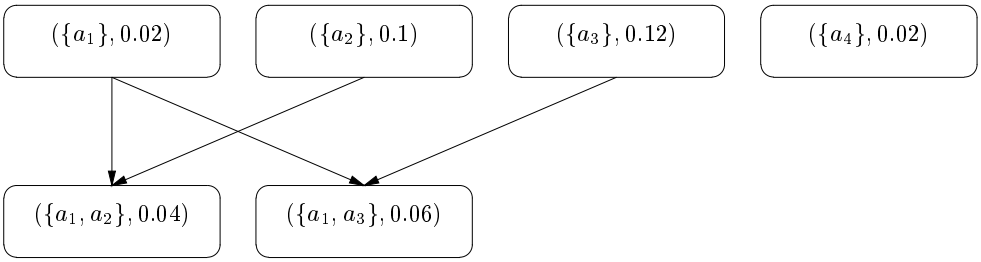

Figura 7.6: Construcción de $\mathcal{G}_{12}$ (iii).

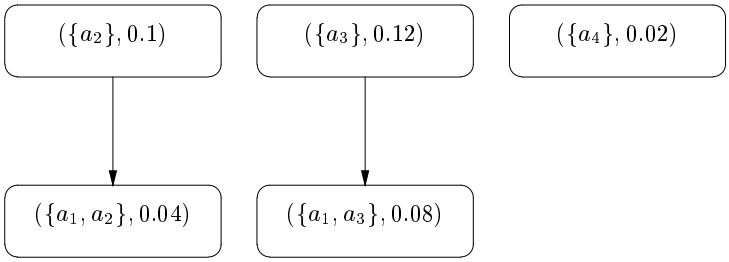

Figura 7.7: Construcción de $\mathcal{G}_{12}$ (iv).

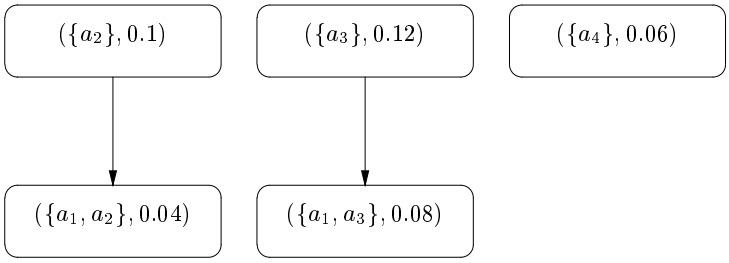

Figura 7.8: Estructura final de $\mathcal{G}_{12}$.

Una propiedad importante de la combinación aproximada es que la unión de los focales nunca disminuye cuando se reduce una RSR. De otra forma el muestreo por importancia no sería válido, como veremos más adelante.

7.3.2 Restricción

A continuación definiremos la restricción de una RSR de cierta función masa, siguiendo la definición 7.1. Obsérvese que no es necesario definir una operación de restricción aproximada, dado que el tamaño de la RSR resultante siempre va a ser menor o igual que el de la original.

El siguiente algoritmo muestra cómo realizar la operación de restricción, aprovechando que los elementos de una RSR están ordenados para la inclusión. Partimos de una masa $m : 2^{U_I} \rightarrow [0,1]$ definida para un conjunto de variables X_I, representada mediante una RSR $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ y queremos obtener una RSR $\mathcal{G}^{R(A)}$ correspondiente a la masa $m^{R(A)} : 2^{U_I} \rightarrow [0,1]$, donde $A \subseteq U_J$ es un conjunto de configuraciones de las variables X_J, con $J \subseteq I$.

RESTRIC($\mathcal{G}$,A)

---

1. Sea $\mathcal{N}$ una lista de conjuntos ordenados de menor a mayor según su cardinal.
2. Sea $\mathcal{L}$ el conjunto de nodos minimales de $\mathcal{V}$.
3. Para cada nodo $V \in \mathcal{L}$,
   (a) Si $V \not\subseteq A^{\uparrow I}$, marcar V y todo nodo antecesor suyo como visitado.
   (b) Si $V \subseteq A^{\uparrow I}$,
      i. Marcar V como visitado.
      ii. Añadir el par $\{V, m(V)\}$ a la lista $\mathcal{N}$.

4. Reemplazar $\mathcal{L}$ por todos los nodos de $\mathcal{V}$ que sean padres de algún nodo de $\mathcal{L}$ y que no hayan sido visitados.

5. Si $\mathcal{L} \neq \emptyset$, ir al paso 3.

6. Supongamos que $\mathcal{N} = \{\{V_1, m(V_1)\}, \ldots, \{V_n, m(V_n)\}\}$ es la lista de focales obtenida.

7. Desde $i = 1$ hasta n,

   (a) **INSERTA3**($\mathcal{G}^{R(A)}$,V_i,$m(V_i)$).

8. Devolver $\mathcal{G}^{R(A)}$.

Una vez definidas las operaciones necesarias sobre la RSR, estamos en condiciones de formular el algoritmo de muestreo por importancia para funciones masa.

7.4 Algoritmo de Muestreo por Importancia para Funciones Masa

En esta sección pretendemos especificar un algoritmo equivalente al presentado en la sección 5.3 del capítulo 5 para el caso de funciones masa. Hay dos tareas principales que deben ser abordadas de cara a enunciar el algoritmo general; estas tareas son:

1. Eliminación de una variable de un conjunto de funciones masa.

2. A partir de las funciones obtenidas en la eliminación de una cierta variable, y teniendo en cuenta los focales simulados para las demás variables, obtener un focal para la función masa correspondiente a la variable en cuestión.

Teniendo definidas estas tareas, el algoritmo general de simulación para funciones masa, análogo al de probabilidades, puede enunciarse como sigue.

Partimos de un conjunto de masas, $\{m_1, \ldots, m_k\}$, cada una de ellas definidas sobre un conjunto de variables X_{I_i}, $i = 1, \ldots, k$, de forma que $\bigcup_{i=1}^{k} I_i = N = \{1, \ldots, n\}$ forman el conjunto de índices de todas las variables del sistema, X_N. El siguiente algoritmo calcula una función masa a posteriori m'_i, $i = 1, \ldots, n$, para cada variable X_i del sistema.

ALG MI_DS

1. Sea $M = \{m_1, \ldots, m_k\}$.
2. Elegir un orden σ para el conjunto N.
3. Desde $i = 1$ hasta n,
   (a) $M(i)$ =**ELIMINA**(M,$X_{\sigma(i)}$).
4. Desde $j = 1$ hasta m (tamaño de la muestra)
   (a) $w_j = 1.0$.
   (b) Desde $i = n$ hasta 1,
      i. $A_i^{(j)}$ =**SIMULA**($X_{\sigma(i)}$,$M(i)$,w_j).
   (c) Desde $i = n$ hasta 1,
      i. $m'_i(A_i^{(j)}) = m'_i(A_i^{(j)}) + w_j$.
5. Normalizar los m'_i, $i = 1, \ldots, n$.

Obsérvese que en el algoritmo principal no es necesario distinguir si las masas se van a representar por semi retículos o no. Sin embargo, en el procedimiento de eliminación de variables y en la simulación sí haremos uso de esta estructura. Pasamos a definir los procedimientos **ELIMINA** y **SIMULA**.

Supongamos que M es un conjunto de RSR asociadas a ciertas funciones masa. Dada una RSR $\mathcal{G}$, denotaremos por $s(\mathcal{G})$ el conjunto de índices de las variables para las que está definida la masa asociada a $\mathcal{G}$. En estas condiciones, el algoritmo de eliminación de una variable X_i de las masas de M puede enunciarse como sigue.

ELIMINA(M,X_i)

---

1. Sea $M(i) = \{\mathcal{G} \in M \mid i \in s(\mathcal{G})\}$, el conjunto de RSR definidas para la variable X_i. Eliminar $M(i)$ de M. Sea $M^+(i) = M(i)$.
2. Mientras haya más de una RSR en $M^+(i)$,
   (a) Tomar $\mathcal{G}_1, \mathcal{G}_2 \in M^+(i)$.
   (b) Calcular $\mathcal{G}$ =**COMBINA_APR**($\mathcal{G}_1$,$\mathcal{G}_2$).
   (c) Reemplazar en $M^+(i)$, $\mathcal{G}_1$ y $\mathcal{G}_2$ por $\mathcal{G}$.
3. Sea $\mathcal{G}$ la única RSR perteneciente a $M^+(i)$.
4. Calcular $\mathcal{G}'$ =**MARGINALIZA**($\mathcal{G}$,X_i).
5. Añadir $M^+(i)$ a M.

El procedimiento **SIMULA**($X_{\sigma(i)}$,$M(i)$,w) debe devolver un focal para la masa correspondiente a una cierta variable $X_{\sigma(i)}$, donde la masa en cuestión vendrá dada por

la combinación de las funciones masa contenidas en $M(i)$. Algunas formas de realizar la combinación de una serie de funciones masa cuando éstas están definidas sobre el mismo dominio han sido estudiadas por Moral y Wilson [62, 63]. En este caso, no todas las funciones de $M(i)$ estarán definidas sobre el mismo dominio, por lo que habrá que restringirlas al conjunto de conjuntos de configuraciones de la variable $X_{\sigma(i)}$. Por otro lado, introducimos el uso de precomputación aproximada en el paso 5.(i), con la idea de obtener de forma más probable focales con intersección no vacía en el proceso de simulación.

SIMULA($X_{\sigma(i)}$,$M(i)$,w)

---

1. Sea $A_{\Sigma(i)} = \{A_{i+1}, \ldots, A_n\}$ el conjunto de los focales simulados hasta el momento.

2. Sea $M(i) = \{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$ el conjunto de RSR calculado en el paso 1 del procedimiento **ELIMINA**, y $\{m_1, \ldots, m_k\}$ las masas correspondientes.

3. Desde $j = 1$ hasta k,

   (a) Para cada $A \in A_{\Sigma(i)}$, hacer $\mathcal{G}_j$ =**RESTRIC**($\mathcal{G}_j$,A).

4. Sea m'_j la función masa representada por $\mathcal{G}_j$.

5. Desde $j = k$ hasta 1,

   (a) Si $j = k$, hacer $\mathcal{G}^{(j)} = \mathcal{G}_j$.
   En otro caso, $\mathcal{G}^{(j)}$ =**COMBINA_APR**($\mathcal{G}_j$,$\mathcal{G}^{(j+1)}$).

   (b) Calcular $Pl^{(j)}$, la función de plausibilidad asociada a la función masa $m^{(j)}$, correspondiente a $\mathcal{G}^{(j)}$, es decir,

$$Pl^{(j)}(A) = \sum_{B \cap A \neq \emptyset} m^{(j)}(B), \quad \forall A \subseteq U_{s(m^{(j)})}.$$

(c) Simular un conjunto focal A_j de m'_j tal que

$$A_j^{\uparrow s(m^{(j)})} \cap A_{j+1}^{\uparrow s(m^{(j)})} \cap \cdots \cap A_k^{\uparrow s(m^{(j)})} \neq \emptyset,$$

con probabilidad

$$P_j(A_j) \propto m'_j(A_j) \cdot Pl^{(j)}(A_j^{\uparrow s(m^{(j)})} \cap A_{j+1}^{\uparrow s(m^{(j)})} \cap \cdots \cap A_k^{\uparrow s(m^{(j)})}).$$

(d) Si m_j es una de las masas iniciales del sistema, hacer $w = w \cdot m_j(A_j)$.

(e) Hacer $w = \dfrac{w}{P_j(A_j)}$.

6. Devolver el conjunto $A = (\bigcap_{j=1}^{k} A_j^{\uparrow s(m^{(1)})})^{\downarrow \sigma(i)}$.

Obsérvese que cada vez que se combinan dos RSR en el paso 5.(a), el resultado está definido sobre la unión de los conjuntos de variables para los que están definidas las RSR que se combinan. Por lo tanto, al final del algoritmo se tiene que $m^{(1)}$ está definida para todas las variables que intervienen en cualquiera de las m_i, $i = 1, \ldots, k$. Por eso la intersección del paso 6 hay que calcularla sobre el dominio $U_{s(m^{(1)})}$.

Una propiedad que debe verificarse para que el muestreo por importancia sea válido es que debe ser posible obtener cualquier configuración que tenga probabilidad exacta mayor que cero. En el caso de las funciones masa, debemos estar seguros que cualquier focal que tenga masa mayor que cero, tenga probabilidad de muestreo también mayor que cero. Esto se verifica gracias a la propiedad de la combinación aproximada de mantener constante la unión de los focales. Esto garantiza que cualquier focal que tenga masa positiva podrá ser obtenido, pues su plausibilidad (paso 6.(b)) será positiva aunque éste haya sido eliminado de la RSR.

7.5 Conclusiones

En este capítulo hemos presentado una metodología de muestreo por importancia para funciones de creencia de Dempster-Shafer. El desarrollo ha sido análogo al seguido en el caso de probabilidades. Por un lado, se introduce la precomputación aproximada para limitar el tamaño de las funciones masa, y por otro lado se utiliza un método de Monte Carlo para estimar las masas 'a posteriori' para cada variable.

El trabajo llevado a cabo en este capítulo dan muestra de la generalidad de las ideas propuestas para propagación de probabilidades.

Una continuación inmediata de este trabajo debe ser un estudio experimental del comportamiento de los algoritmos aquí propuestos, de cara a comprobar su funcionalidad en la práctica.

Capítulo 8

Conclusiones y Líneas Abiertas

En esta memoria, hemos llevado a cabo un estudio detallado sobre métodos de propagación aproximados sobre modelos gráficos. El uso de estos métodos está justificado cuando no podemos obtener en un tiempo razonable soluciones exactas.

Dentro de los algoritmos aproximados, hemos puesto el mayor énfasis en la técnica de muestreo por importancia. En este sentido, cabe destacar la forma de obtener las funciones de muestreo, mediante una precomputación aproximada. Es importante destacar que en esta fase de precomputación, es posible detectar cuándo se pueden obtener resultados exactos, en cuyo caso no será necesario llevar a cabo la fase de simulación.

La misma idea puede aplicarse a la simulación estratificada. Éste es un método eficiente de simulación, pero su aplicabilidad se veía fuertemente limitada por el tamaño de la red. Hemos resuelto ese problema introduciendo el esquema estratificado recursivo, equivalente formalmente al anterior, pero capaz de tratar con redes de mayor tamaño. Se observa, según los resultados experimentales, que el muestreo estratificado se ve beneficiado por la obtención de las distribuciones de muestreo mediante precomputación aproximada.

En cuanto al uso de los árboles de probabilidad, los beneficios son destacables si se emplean en problemas difíciles, tales como los grafos de pedigree. Los experimentos

realizados en ese grafo muestran cómo la aplicación de árboles hace que los algoritmos sean más rápidos y que se reduzca el error a la mitad, aproximadamente, del que se produce utilizando tablas. En grafos más sencillos, la clásica representación mediante tablas puede ser más eficiente, tal y como se ha comprobado en el capítulo 5. La principal ventaja de los árboles radica en que permiten aprovechar las regularidades que pueda presentar un potencial.

Hemos realizado también un estudio sobre métodos de propagación para Teoría de la Evidencia. En primer lugar, hemos propuesto un método exacto similar al algoritmo HUGIN para probabilidades. La principal cualidad del método propuesto es que realiza todos los cálculos sobre funciones masa. Es de destacar la definición de un inverso para funciones masa (definición 6.7). Este inverso está definido de forma recursiva, por lo que desarrollamos también una estructura que permitiera calcularlo de forma eficiente. Dicha estructura, que llamamos *representación por semi retículo*, juega un papel similar al de los árboles de probabilidad a la hora de desarrollar un algoritmo de muestreo por importancia para Teoría de la Evidencia.

Con ese algoritmo aproximado para funciones masa, tratamos de expresar la generalidad de los métodos desarrollados para probabilidades.

En cuanto a las líneas a seguir en un futuro, debemos considerar que los métodos estudiados en esta memoria siempre serán susceptibles de mejora. No en vano, tenemos que recordar que el problema a resolver (la inferencia) es NP-duro, con lo que nuestros métodos, que son exponenciales, nunca resolverán todos los posibles ejemplos. Como posibles líneas de trabajo, además de las citadas en cada uno de los capítulos, podríamos señalar:

1. Generalización de los métodos de precomputación aproximada a otros formalismos. Al igual que se han extendido en esta memoria para Teoría de la Evidencia, pensamos que estos métodos, inicialmente pensados para probabilidades, pueden ser aplicables a otros formalismos como los intervalos de probabilidad, e incluso podrían generalizarse a un nivel más abstracto.

2. Estudio de nuevos criterios para aproximación de potenciales usando árboles de

probabilidad, y masas usando representaciones por semi retículo. Aquí será de gran ayuda el trabajo experimental.

3. Aplicación práctica de los métodos desarrollados. Pensamos que posibles problemas a resolver son todos aquellos que involucren un alto número de variables, de manera que los cálculos exactos sean inviables. Por ejemplo, en la Universidad de Aalborg, han aplicado técnicas de simulación para propagar en grafos de pedigree. Problemas similares podrían resolverse con nuestros métodos.

Bibliografía

[1] Almond, R. (1995). *Graphical belief modelling.* Chapman and Hall, London.

[2] Andersen, S.K., Olesen, K.G., Jensen, F.V., Jensen, F. (1989). HUGIN- A shell for building Bayesian belief universes for expert systems. En *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, 1080–1085. Morgan Kaufmann, San Mateo.

[3] Bissig, R., Kohlas, J., Lehmann, N. (1997). Fast-division architecture for Dempster-Shafer belief functions. D. Gabbay, R. Kruse, A. Nonnengart and H.J. Ohlbach (eds.), Qualitative and Quantitative Practical Reasoning, First International Joint Conference on Qualitative and Quantitative Practical Reasoning; ECSQARU–FAPR'97. *Lecture Notes in Artificial Intelligence*. Springer.

[4] Bouckaert, R.R. (1994). A stratified simulation scheme for inference in Bayesian belief networks. En *Uncertainty in Artificial Intelligence, Proceedings of the Tenth Conference*, 110–117.

[5] Bouckaert, R.R., Castillo, E., Gutiérrez, J.M. (1996). A modified simulation scheme for inference in Bayesian networks. *International Journal of Approximate Reasoning*, (14):55–80.

[6] Boutilier, J., Friedman, N., Goldszmidt, M., Koller, D. (1996). Context-specific independence in Bayesian networks. En E. Horvitz and F.V. Jensen, eds. *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, 115–123.

[7] Breiman, L. (1968). *Probability.* Addison Wesley.

[8] Cano, A., Moral, S. (1995). Heuristic algorithms for the triangulation of graphs. En *Advances in Intelligent Computing*. Eds.: B. Bouchon-Meunier, R.R. Yager and L.A. Zadeh. Springer Verlag, 98–107.

[9] Cano, A., Moral, S. (1997). Propagación exacta y aproximada con árboles de probabilidad. En *Actas de la VII Conferencia de la Asociación Española para la Inteligencia Artificial*, 635–644.

[10] Cano, J.E. (1992). Propagación de probabilidades inferiores y superiores en grafos. Tesis Doctoral. Universidad de Granada.

[11] Cano, J.E., Delgado, M., Moral, S. (1993). An axiomatic framework for the propagation of uncertainty in directed acyclic networks. *International Journal of Approximate Reasoning*, (8):253–280.

[12] Cano, J.E., Hernández, L.D., Moral, S. (1995). Propagación de probabilidades en grafos de dependencias mediante muestreo por importancia. *Actas de la VI Conferencia de la Asociación Española para la Inteligencia Artificial* (CAEPIA 95), 197–206.

[13] Cano, J.E., Hernández, L.D., Moral, S. (1996). Importance sampling algorithms for the propagation of probabilities in belief networks. *International Journal of Approximate Reasoning*, (15):77–92.

[14] Castillo, E., Gutiérrez, J.M., Hadi, A.S. (1996). *Sistemas expertos y modelos de redes probabilísticas*. Monografías de la Academia de Ingeniería.

[15] Chin, H., Cooper, G.F. (1987). Stochastic simulation of Bayesian networks. En *Proceedings of the Third Workshop on Uncertainty in Artificial Intelligence*. Seattle, Washington.

[16] Chin, H., Cooper, G.F. (1989). Bayesian belief network inference using simulation. En L.N. Kanal, T.S. Levitt and J.F. Lemmer, eds. *Uncertainty in Artificial Intelligence* 3, 129–148. Elsevier Science Publishers. North Holland.

[17] Cooper, G.F. (1989). An algorithm for computing probabilistic propositions. En L.N. Kanal, T.S. Levitt and J.F. Lemmer, eds.*Uncertainty in Artificial Intelligence* 3, 3–14. Elsevier Science Publishers. North Holland.

[18] Cooper, G.F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, (42):393–405.

[19] Cousins, S.B., Chen, W., Frisse, M.E. (1993). A tutorial introduction to stochastic simulation algorithms for belief networks. *Artificial Intelligence in Medicine*, (5):315–340.

[20] Dagum, P., Luby, M. (1993). Approximating probabilistic inference in Bayesian networks is NP-hard. *Artificial Intelligence*, (60):141–153.

[21] D'Ambrosio, B. (1989). Symbolic probabilistic inference in belief nets. Technical Report. Oregon State University.

[22] Darroch, J.N., Lauritzen, S.L., Speed, T.P. (1980). Markov fields and log-linear models for contingency tables. *Annals of Statistics*, (8):522–539.

[23] Dawid, A. P., Kjærulff, U., Lauritzen, S.L. (1995). Hybrid Propagation in junction trees. En Advances in Intelligent Computing, B. Bouchon-Meunier, R. R. Yager, L. A. Zadeh (eds), *Lecture Notes in Computer Science*, (945):87–97. Springer-Verlag.

[24] Dechter, R. (1996). Bucket elimination: a unifying framework for probabilistic inference. En E. Horvitz and F.V. Jensen, eds. *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, 211–219.

[25] Dempster, A.P. (1967). Upper and lower probabilities induced by a multivalued mapping. *Annals of Mathematical Statistics*, (38):325–339.

[26] Dugat, V., Sandri, S. (1994). Complexity of hierarchical trees in Evidence Theory. *ORSA Journal on Computing*, (6):37–49.

[27] Feller, W. (1973). *Introducción a la teoría de probabilidades y sus aplicaciones.* Limusa.

[28] Fertig, K.W., Mann, N.R. (1980). An accurate approximation to the sampling distribution of the studentized extreme-valued statistic. *Technometrics*, (22):83–90.

[29] Fishman, G.S. (1996). *Monte Carlo. Concepts, algorithms and applications.* Springer.

[30] Fung, R., Chang, K.C. (1990). Weighting and integrating evidence for stochastic simulation in Bayesian networks. En: *Uncertainty in Artificial Intelligence* 5. (M. Henrion, R.D. Shachter, L.N. Kanal, J.F. Lemmer, eds.) North-Holland (Amsterdam), 209–220.

[31] Gilks, W.R., Richardson, S., Spiegelhalter, D.J. (1996). *Markov chain Monte Carlo in practice.* Chapman and Hall.

[32] Gutiérrez, R., Martínez, A., Rodríguez, C. (1993). *Curso básico de probabilidad.* Pirámide.

[33] Henrion, M. (1988). Propagating uncertainty by logic sampling in Bayes' networks. En: *Uncertainty in Artificial Intelligence, 2* (J.F. Lemmer, L.N. Kanal, eds.) Norht-Holland (Amsterdam), 317–324.

[34] Hernández, L.D., Bolaños, M.J. (1994). Aplicación de algoritmos evolutivos para el problema de la triangulación en redes causales. En *Actas del IV Congreso Español de Tecnologías y Lógica Fuzzy*, 127–132.

[35] Hernández, L.D. (1995). Diseño y validación de nuevos algoritmos para el tratamiento de grafos de dependencias. Tesis Doctoral. Universidad de Granada.

[36] Hernández, L.D., Moral, S., Salmerón, A. (1996). Importance sampling algorithms for belief networks based on approximate computation. *Proceedings of the Sixth International Conference IPMU'96.* Vol. II, 859–864.

[37] Hernández, L.D., Moral, S., Salmerón, A. (1997). A Monte Carlo algorithm for probabilistic propagation based on importance sampling and stratified simulation techniques. Por aparecer en *International Journal of Approximate Reasoning.*

[38] Hernández, L.D., Moral, S. (1997). Inference with idempotent valuations. *Proceedings of the UAI'97 Conference.*

[39] Hernández, L.D., Moral, S. (1997). Mixing exact and importance sampling propagation algorithms in dependence graphs. *International Journal of Intelligent Systems*, (12):553–576.

[40] Howard, R.A., Matheson, J.E. (1984). Influence diagrams. En R.A. Howard, J.E. Matheson, editor, *Readings on the Principles and Applications of Decision Analysis*, (2):719–762. Strategic Decision Group, Menlo Park, California.

[41] Huang, C., Darwiche, A. (1996). Inference in belief netowrks: a procedural guide. *International Journal of Approximate Reasoning*, (15):225–263.

[42] Jensen, C.S., Kong, A., Kjærulff, U. (1995). Blocking Gibbs sampling in very large probabilistic expert systems. *International Journal of Human-Computer Studies*, (42):647–666.

[43] Jensen, F., Andersen, S.K. (1990). Approximations in Bayesian belief universes for knowledge based systems. *Proceedings of the 6th Workshop on Uncertainty in Artificial Intelligence*, Cambridge, MA.

[44] Jensen, F.V., Olesen, K.G., Andersen, S.K. (1990). An algebra of Bayesian belief universes for knowledge based systems. *Networks*, (20):637–659.

[45] Jensen, F.V., Lauritzen, S.L., Olesen, K.G. (1990). Bayesian updating in causal probabilistic networks by local computation. *Computational Statistics Quarterly*, (4):269–282.

[46] Jensen, F.V., Jensen, F. (1994). Optimal junction trees. Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence. Seattle, Washington, Mantaras, R.L and Poole, D. (eds.), Morgan Kaufmann, 360–366.

[47] Jensen, F.V. (1996). *An introduction to Bayesian networks.* UCL Press.

[48] Kim, J., Pearl, J. (1983). A computational model for causal and diagnostic reasoning in inference systems. *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, 190–193.

[49] Kjærulff, U. (1992). Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, (2):1–21.

[50] Kjærulff, U. (1993). Approximation of Bayesian networks through edge removals. Research Report IR-93-2007, Dept. of Mathematics and Computer Science, Aalborg University.

[51] Kjærulff, U. (1994). Reduction of computational complexity in Bayesian networks through removal of weak dependences. *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 374–382, Morgan Kaufmann, San Francisco, California.

[52] Klir, G.J., Folger, T.A. (1988). *Fuzzy sets, uncertainty and information.* Prentice Hall.

[53] Kozlov, D., Koller, D. (1997). Nonuniform dynamic discretization in hybrid networks. En *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, 302–313. Morgan Kaufmann.

[54] Kruse, R.L., Leung, B.P., Tondo, C.L. (1991). *Data structures and program design in C.* Prentice Hall.

[55] Kullback, S., Leibler, R. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, (22):76–86.

[56] Larrañaga, P., Kuijpers, C., Poza, M., Murga, R. (1997). Decomposing Bayesian networks: triangulation of the moral graph with genetic algorithms. *Statistics and Computing*, (7):19–34.

[57] Lauritzen, S.L, Speed, T.P., Vijayan, K. (1984). Decomposable graphs and hypergraphs. *Journal of the Australian Mathematical Society, Series A*, (36):12–29.

[58] Lauritzen, S.L., Spiegelhalter, D.J. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, (50):157–224.

[59] Lauritzen, S.L. (1996). *Graphical models*. Oxford Science Publications.

[60] Lauritzen, S.L., Jensen, F.V. (1997). Local computation with valuations from a commutative semigroup. *Annals of Mathematics and Artificial Intelligence*, (21):51–69.

[61] Monti, S., Cooper, G.F. (1996). Bounded recursive approximation: A search method for belief-network inference under limited resources. *International Journal of Approximate Reasoning*, (15):49–76.

[62] Moral, S., Wilson, N. (1994). Markov-Chain Monte-Carlo algorithms for the calculation of Dempster-Shafer Belief. *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)* Vol. 1, 269–274.

[63] Moral, S., Wilson, N. (1996). Importance sampling algorithms for the calculation of Dempster-Shafer belief. *Proceedings of the IPMU-96 Conference*, Vol. 3, 1337–1344.

[64] Neapolitan, R.E. (1990). *Probabilistic reasoning in expert systems*. John Wiley, New York.

[65] Orponen, P. (1990). Dempster's rule is #P-complete. *Artificial Intelligence*, (44):245–253.

[66] Pearl, J. (1986). A constraint propagation approach to probabilistic reasoning. En *Uncertainty in Artificial Intelligence*, (L.M. Kanal, J.F. Lemmer, eds.), 357–370. North-Holland, Amsterdam.

[67] Pearl, J. (1986). Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, (29):241–288.

[68] Pearl, J. (1987). Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, (32):247–257.

[69] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan-Kauffman. San Mateo.

[70] Poole, D. (1997). Exploiting contextual independence and approximation in belief network inference. Technical Report.

[71] Poole, D. (1997). Probabilistic partial evaluation: exploiting rule structure in probabilistic inference. En *Proc. 15th International Joint Conference on AI (IJCAI-97)*, 1284–1291.

[72] Rubinstein, R.Y. (1981). *Simulation and the Monte Carlo Method*. Wiley (New York).

[73] Salmerón, A., Bolaños, M.J. (1995). Método híbrido para inferencia en redes bayesianas. *Actas XXII Congreso Nacional de Estadística e I.O.*, 243–244.

[74] Salmerón, A., Jensen, F.V. (1997). HUGIN architecture for propagating belief functions. Technical Report #GAD9702. Data Analysis Group. Universidad de Almería.

[75] Sandri, S. (1991). Structuring bodies of evidence. *7th Conference on Uncertainty in AI*, Los Angeles.

[76] Schmidt, T., Shenoy, P.P. (1997). Some improvements to the Shenoy-Shafer and HUGIN architectures for computing marginals. School of Business Working Paper No.275. University of Kansas.

[77] Shachter, R.D. (1986). Evaluating influence diagrams. *Operations Research*, (34):871–882.

[78] Shachter, R.D. (1988). Probabilistic inference and influence diagrams. *Operations Research*, (36):589–605.

[79] Shachter, R.D., D'Ambrosio, B., Del Favero, B.A. (1990). Symbolic probabilistic inference in belief networks. *Proceedings of the AAAI'90 Conference*, Vol. 1, 126–131.

[80] Shachter, R.D., Peot, M.A. (1990). Simulation approaches to general probabilistic inference on belief networks. En: *Uncertainty in Artificial Intelligence* 5, (M. Henrion, R.D. Shachter, L.N. Kanal, J.F. Lemmer, eds.) North Holland (Amsterdam), 221–231.

[81] Shachter, R.D., Andersen, S.K., Szolovits, P. (1991). The equivalence of exact methods for probabilistic inference on belief networks. Technical Report.

[82] Shafer, G. (1976). *A mathematical theory of evidence*. Princeton University Press, Princeton, N.J.

[83] Shafer, G., Shenoy, P.P. (1988). Local computation in hypertrees. Technical Report WP-201, School of Business, University of Kansas.

[84] Shafer, G., Shenoy, P.P. (1990). Probability propagation. *Annals of Mathematical and Artificial Intelligence*, (2):327–351.

[85] Shafer, G., Pearl, J. (1990). *Readings in uncertain reasoning*. Morgan Kaufmann Publishers, San Mateo.

[86] Shafer, G. (1996). *Probabilistic expert systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

[87] Shenoy, P.P, Shafer, G.R. (1990). Axioms for probability and belief function propagation. En *Uncertainty in Artificial Intelligence* 4, (ed. R.D. Shachter, T.S. Levitt, L.N. Kanal and J.F. Lemmer), 169–198. North Holland, Amsterdam.

[88] Shenoy, P.P. (1997). Binary join trees for computing marginals in the Shenoy-Shafer architecture. *International Journal of Approximate Reasoning*, (17):239–263.

[89] Shortliffe, E.H. (1976). *Computer-based medical consultation: MYCIN.* Elsevier, New York.

[90] Thoma, H.M. (1989). Factorization of belief functions. PhD Thesis. Harvard University, Department of Statistics.

[91] Verma T., Pearl J. (1988) Causal networks: semantics and expresiveness. En *Proceedings of the 4th Workshop on Uncertainty in Artificial Intelligence*, 352–358.

[92] Verma T., Pearl J. (1990) Identifying independence in bayesian networks. *Networks*, (20):507–534.

[93] Walley, P. (1991). *Statistical reasoning with imprecise probabilities.* Chapman and Hall.

[94] Xu, H. (1991). An efficient implementation of belief function propagation. En *Uncertainty in Artificial Intelligence* 5, (ed. B.D. D'Ambrosio, Ph. Smets, and P. Bonissone), 425–432. San Mateo, CA: Morgan Kaufmann.

[95] Xu, H., Kennes, R. (1994). Steps toward efficient implementation of Dempster-Shafer theory. En *Advances in the Dempster-Shafer Theory of Evidence*, (ed. R.R. Yager et al). Wiley.

[96] Zhang, N.L., Poole, D. (1996). Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, (5):301–328.