Desarrollo de nuevos métodos de inferencia y aprendizaje en modelos gráficos probabilísticos con probabilidades imprecisas

Ofelia Retamero Pascual

Editor: Universidad de Granada. Tesis Doctorales
Autor: Ofelia Paula Retamero Pascual
ISBN: 978-84-1195-136-4
URI: https://hdl.handle.net/10481/89173

Departamento de Ciencias de la Computación e Inteligencia Artificial
Tratamiento de la Incertidumbre en Sistemas Inteligentes

UNIVERSIDAD
DE GRANADA

Desarrollo de nuevos métodos de inferencia y aprendizaje en modelos gráficos probabilísticos con probabilidades imprecisas

---

Tesis doctoral de
Ofelia Paula Retamero Pascual

Para optar al grado de
DOCTORA POR LA UNIVERSIDAD DE GRANADA
dentro del Programa de Doctorado en
TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN

Directores
Prof. Manuel Gómez Olmedo
Prof. Andrés Cano Utrera

Granada, 2023

Ofelia Retamero Pascual

Desarrollo de nuevos métodos de inferencia y aprendizaje en modelos gráficos probabilísticos con probabilidades imprecisas

Granada, 2023

Universidad de Granada

Tratamiento de la Incertidumbre en Sistemas Inteligentes

Departamento de Ciencias de la Computación e Inteligencia Artificial

Esta memoria se ha escrito en LaTeX, utilizando la plantilla de nombre *Clean Thesis* proporcionada por Ricardo Langner.

Agradecimientos

En primer lugar, quiero agradecer a mis directores de tesis, **Manuel Gómez Olmedo** y **Andrés Cano Utrera**, por darme la oportunidad de vivir esta experiencia no sólo de la mano de fantásticos profesionales sino de maravillosas personas. Vuestro apoyo, disponibilidad, paciencia y conocimientos han marcado una huella que sin duda continuará en mí el resto de mi carrera profesional. Quiero extender mi agradecimiento a todo el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada por el gran trabajo que realizan y la calidez que siempre me han mostrado. En especial, a los que fueron mis profesores en el máster, por compartir a diario su trabajo, avances y conocimientos y buscar hacerlo siempre de la manera más amable y amena posible. Deseo trasladar también mi agradecimiento a mi profesora de colegio, **Antoñita**, por enseñarme a amar las matemáticas y animarme a interesarme por ellas más allá de las horas lectivas.

La experiencia habría sido muy distinta si no fuese por los grandes compañeros y amigos que me llevo del CITIC-UGR, **Carlos, Manolo** y **Salva**, además de **Patri**, que casi lo ha vivido tan de cerca como nosotros. Durante estos años hemos compartido momentos de estudio, pero también otros muchos de ocio que serán imborrables. Nos conocimos, como si nada, hace ya casi 7 años y no me queda duda de que quedan muchos años más por disfrutar, a partir de ahora, en otra etapa diferente.

No puedo dejar pasar la oportunidad de agradecer a mis padres, **María Isabel** y **Gerardo**, por todo lo que me dieron, por enseñarme que el trabajo duro y continuo acaba por dar los frutos más bonitos, por darlo todo y todos los días por nosotros y por crear ese hogar en el que siempre somos recibidos con los brazos abiertos. Gracias también a mis hermanos, **María** y **Gerardo**, por el apoyo diario, los mejores consejos y por haceros sentir tan presentes incluso en la distancia. Por supuesto, gracias a mis niñas, **Eu, Lucía, Marta** y **Yaiza**, porque hay pocas cosas mejores que vosotras, porque la amistad tiene sentido después de conoceros y porque da igual cuantos meses pasemos sin vernos, reunirnos es el mejor momento del año. **María**, gracias por sanarme, que se dice pronto, por confiar más en mí que yo misma y por las innumerables horas de conversación y amistad, viniste para quedarte y eso me hace inmensamente feliz. **Raúl**, gracias por tu amistad, hace 8 años que coincidimos trabajando y desde luego, ¡qué suerte la mía! Hay pocas personas tan excelentes en tantos aspectos como tú. **Karin** y **Benny**, mil gracias por ser mis padres belgas,

por enseñarme y demostrarme que con amor y sin esperar nada a cambio, puede cambiarse el mundo. Espero seguir formando parte de vuestra vida y cada uno de vuestros proyectos. Además, gracias a tantos que durante estos años me han apoyado, ayudado y aportado felicidad a mis días: **Silvia, Dennis, Eva, Ula, Jarek, Emilio, Alberto, Alfonso, Dani**..., mis familias materna y paterna y todos los perritos que durante esta etapa pasaron por casa hasta encontrar su hogar definitivo.

El agradecimiento más especial va para mi compañero de vida, **Grzegorz**, por hacerme feliz a diario, por quererme incluso en los momentos difíciles y por enseñarme a apreciar y a disfrutar cada minuto. Gracias también a nuestro pequeño, **Mateo**, por llegar y poner la vida patas arriba, además de por enseñarnos lo verdaderamente importante; tus risas y tus abrazos me llenan el alma. Gracias a **Cora** y **Loki** por apoyar sin palabras y amar sin límites.

Esta tesis doctoral fue financiada mediante una beca de investigación De Formación de Personal Investigador (FPI) del Ministerio de Economía, Industria y Competitividad. El código correspondiente a esta ayuda es BES-2017-080857. Además, también ha sido financiada conjuntamente por los proyectos PID2019-106758GB-C31 y TIN2016-77902-C3-2-P del Ministerio de Educación y Ciencia de España y el Fondo Europeo de Desarrollo Regional (FEDER).

Contenidos

Lista de figuras

Lista de tablas

Introducción

Motivación

Los modelos gráficos probabilísticos (MPGs) [115, 123, 153, 187] han demostrado ser una potente herramienta para la gestión de la incertidumbre; pese a que en los comienzos de la Inteligencia Artificial no se consideraron cómo una opción útil [122]. Su aplicación ha sido imprescindible en multitud de campos relacionados con la informática [70, 110]; y otros que *a priori* no se encuentran directamente relacionados, como: medicina, agricultura, economía, meteorología... [1, 3, 5, 73, 133, 177]. Esta memoria se centra en un subconjunto de MGPs, las llamadas redes Bayesianas (RBs) [87, 103, 120, 149, 150, 156]; cuya peculiaridad es que el grafo que representa el modelo es del tipo dirigido acíclico. Sobre las RBs se busca resolver dos tipos de problemas: los centrados en encontrar la estructura de grafo y los parámetros de probabilidad relativos al problema (*i.e.* aprendizaje), y los relacionados con toma de decisiones o resolución de consultas (*i.e.* inferencia). Encontrar algoritmos que resuelvan estos problemas de un modo más efectivo, en el menor tiempo posible y que puedan ser aplicados a redes de mayor tamaño, supone un reto para la comunidad científica y es objeto de investigación continua.

Toda RB se encuentra formada por dos partes: una cualitativa (representada por el grafo) y otra cuantitativa (que son parámetros numéricos). La parte cuantitativa de las RBs, mide a través de parámetros numéricos, la dependencia o no entre las variables del problema. Almacenar estos parámetros de un modo compacto y fácilmente accesible supone una gran ventaja para posteriores tareas, como la inferencia. Tradicionalmente esta información se ha representado mediante tablas de probabilidad (condicionadas o marginales). La gran limitación de las tablas de probabilidad es ser exhaustivas, es decir, se necesita especificar cada valor para que queden totalmente definidas. A medida que el número de variables y/o estados incrementa, aumenta exponencialmente la complejidad y tamaño de la red; y con ella, su representación. Esto implica que elegir la alternativa de representación que genere una estructura más compacta y que ocupe el menor espacio posible, sea clave para que los algoritmos de inferencia requieran menos tiempo de cómputo. Por esta razón, en la literatura se han desarrollado otros modos alternativos como: árboles de probabilidad (PT - *probability tree*) y árboles de probabilidad binarios (BPT - *binary probability tree*) [29, 30, 38, 42, 82, 170], entre otros. Resaltamos dos ventajas principales de los

PTs y BPTs: (1) son capaces de sacar partido de las denominadas independencias específicas del contexto [26, 115]; y (2) permiten ser aproximados mediante operaciones de poda. Al tratar con bases de datos reales, los valores de probabilidad que representan cuantitativamente el problema presentan una gran repetición de valores, especialmente teniendo en cuenta que lo más común es considerar un pequeño número de cifras decimales (0.9, 0.75, 0.34...). Pero lo cierto es que los PTs y BPTs son capaces de sacar partido de la repetición de valores en situaciones muy concretas, por lo que las estructuras siguen almacenando valores repetidos aún en sus versiones más compactas. Con esta idea en mente, presentamos en parte de esta memoria unas estructuras alternativas, que llamamos potenciales basados en valor (VBPs - *value-based potentials*). Los VBPs buscan sacar partido de la repetición de valores, sin importar en qué orden se encuentren. La experimentación con estas nuevas representaciones muestra que, en general, consiguen expresar el conocimiento cuantitativo de un modo más compacto que las anteriores opciones. Al igual que ocurre con PTs y BPTs, también permiten ser aproximadas; presentaremos el modo de hacerlo.

El aprendizaje estructural consiste en aprender la parte cualitativa de RBs, es decir, el grafo. Este proceso puede ser llevado a cabo por un experto en la materia (opción generalmente menos deseable por requerir un gran conocimiento en el tema y porque en muchas ocasiones no se dispone de experto), o utilizando algún método automático que detecte las dependencias e independencias entre las variables partiendo de los datos. Diversos algoritmos se han centrado en este tipo de aprendizaje [92, 143, 185]. A menudo, estos métodos buscan el grafo que maximiza una determinada métrica. Otra posible perspectiva se centra en calcular la probabilidad marginal *a posteriori* de las aristas. Dentro de esta otra perspectiva podemos encontrar dos vertientes: (1) métodos de cadenas de Markov Monte Carlo (MCMC) [67, 74, 86, 131] y (2) métodos estocásticos de búsqueda [89, 107, 135]. Estos métodos presentan la limitación de no poder utilizarse en redes de tamaño considerable, por esta razón, proponemos en esta memoria un algoritmo aproximado de aprendizaje de conjuntos de redes bajo una perspectiva variacional.

Clásicamente, los MGPs se han estudiado desde un punto de vista preciso, en el que las probabilidades asociadas son valores exactos. Por otra parte, habrá problemas que pudiendo deberse a diversos factores (información insuficiente, presencia de varios expertos que proporcionen valores dispares, información vaga...) no puedan ser tratados siguiendo esta perspectiva. Un ejemplo típico de ello se ve a menudo en medicina, donde el resultado de padecer una enfermedad viene dado en términos de intervalos de probabilidad. La teoría de probabilidades imprecisas se ocupa de este tipo de problemas. Las RBs simples (precisas) de tamaño considerable presentan limitaciones a la hora de hacer inferencia sobre ellas, ya que con el incremento de tamaño, la complejidad aumenta exponencialmente y los cálculos asociados sobrepasan potencialmente los recursos computacionales. Las redes Bayesianas con probabilidades imprecisas (*i.e.* redes

credales) pueden verse como un conjunto de RBs, por lo que el desarrollo de algoritmos de inferencia más potentes, que sean capaces de trabajar con ellas, es aún más necesario que para el caso preciso. Este problema puede afrontarse de diversos modos, uno de ellos es el de utilizar algoritmos aproximados, que proporcionen una respuesta muy similar a la exacta, produciendo una pequeña pérdida de información aceptada, pero lo hagan en una fracción del tiempo. Por otro lado, la paralelización de procesos ha demostrado ser efectiva en las tareas de inferencia con RBs [174, 132], ya que permite separar las tareas para que sean realizadas simultáneamente y utilizar así los recursos computacionales de un modo inteligente. En esta memoria vamos a introducir una aproximación a un método paralelo para realizar inferencia exacta sobre redes credales (*i.e.* RBs con probabilidades imprecisas). Más concretamente, vamos a generalizar un método *branch-and-bound* ofreciendo la parelización del mismo.

Organización de la tesis

Esta memoria se distribuye en siete capítulos. El primero de ellos (Capítulo 1) supone una revisión bibliográfica de las dos grandes bases sobre las que se asienta la teoría de modelos gráficos probabilísticos: la teoría de grafos (Sección 1.1) y la teoría de la probabilidad (Sección 1.2). Los dos siguientes capítulos hablan sobre los MGPs, en primer lugar (Capítulo 2) desde un punto de vista preciso, y más tarde (Capítulo 3), impreciso. En el Capítulo 2 se definen los MGPs y sus tipos (Sección 2.1); para a continuación prestar especial atención a las redes Bayesianas, caso particular de MGP en el que el grafo es dirigido acíclico. Sobre las RBs se presenta su concepto y características principales, se introduce un criterio básico que permite detectar dependencias e independencias observando el grafo y sin necesidad de cálculo probabilístico (criterio de d-separación) (Sección 2.2), y se habla de los dos problemas principales a resolver con RBs: aprendizaje (Sección 2.3) e inferencia (Sección 2.4). Puesto que la inferencia con MGPs necesita del concepto de potencial y la definición de operaciones sobre ellos, la Sección 2.4 contendrá esta información. A lo largo del capítulo, los conceptos se ilustrarán con un conocido ejemplo, la RB *asia*. El Capítulo 3 comienza hablando sobre probabilidades imprecisas, sus conceptos principales, las diferentes aproximaciones que se han estudiado en la literatura (Sección 3.1). El resto del capítulo se desarrolla desde la perspectiva de convexos de probabilidad, por ser una alternativa sobre la que pueden realizarse las operaciones necesarias para el desarrollo de inferencia con probabilidades imprecisas. También se introducirán los llamados conjuntos credales y redes credales (*i.e.* redes Bayesianas con probabilidades imprecisas) y se expondrá cómo realizar inferencia sobre redes credales valiéndose de árboles de probabilidad (Sección 3.2).

La parte principal de esta memoria puede encontrarse en los tres capítulos sucesivos (Capítulos 4, 5 y 6). El Capítulo 4 habla sobre la representación de información cuantitativa. Al comienzo de este capítulo nos centraremos en comentar otros aspectos relevantes sobre potenciales y sus representaciones (Sección 4.1). El valor principal de este capítulo se trata de la introducción de una nueva propuesta de representación de potenciales, los potenciales basados en valor (VBPs), que buscan aprovechar al máximo la repetición de valores. Sobre VBPs, se podrá encontrar en la Sección 4.2 la definición de las cuatro alternativas propuestas, la descripción del método para acceder a un índice concreto en la propia estructura y el espacio de memoria que cada estructura ocupa. Además se evaluará empíricamente distintos aspectos de estas estructuras, utilizando dos conjuntos de redes Bayesianas (*bnlearn* [174, 175] y competiciones *UAI* [196, 197]). Las tareas para la evaluación empírica llevadas a cabo han sido: comparación de tamaños de memoria, tiempo de acceso al valor correspondiente a un índice dado y tiempo de cálculo de distribuciones *a posteriori* utilizando el algoritmo de eliminación de variables (VE). La Sección 4.3 introduce un algoritmo para aproximar estas nuevas estructuras. En concreto, se define para la alternativa VDI, por simplicidad, pero podría estudiarse de manera análoga para cualquiera de las otras. En la última parte del capítulo (Sección 4.4) podemos encontrar algunas conclusiones y comentarios sobre las estructuras VBPs y su aproximación.

El Capítulo 5 trata el aprendizaje estructural desde un punto de vista variacional. Para ello, se comienza con una sección de prerrequisitos (Sección 5.1), en los que se definen las distintas métricas Bayesianas disponibles y se introducen el aprendizaje estructural Bayesiano y el análisis variacional, sus características y funcionamiento. A continuación se presenta una primera aproximación a lo que supondría un algoritmo automático de aprendizaje estructural variacional (Sección 5.2), que parte de los datos para valerse del análisis variacional y generar una estructura que represente las relaciones efectivas entre las variables. Para ello definiremos cada uno de los pasos del análisis variacional para nuestro caso concreto. Podremos ver en detalle los pasos que el algoritmo realiza mediante los pseudocódigos correspondientes (Sección 5.3). Realizaremos también una experimentación con algunas RBs Bayesianas (Sección 5.4), que darán una idea de los beneficios que el método presenta y las cualidades de las redes sobre las que su aplicación será más útil. En la última Sección 5.5 pueden encontrarse posibles mejoras a realizar sobre el algoritmo, así como otros comentarios.

En el Capítulo 6 se establece un método paralelo, generalización del algoritmo de inferencia *branch-and-bound* sobre redes credales presentado en el trabajo de Cano *et al.* [36]. Necesitaremos primero entender los dos algoritmos presentes en el mismo trabajo de Cano *et al.* [36] (Sección 6.1) por ser la base del algoritmo paralelo. En la Sección 6.2 se introduce el algoritmo *branch-and-bound* paralelo sobre redes credales. En la última parte del capítulo (Sección 6.2.1)

se lleva a cabo una pequeña experimentación que da una idea inicial sobre el rendimiento del algoritmo. Esto se hace aplicándolo sobre dos redes credales que comparten grafo, pero difieren en el número de estados de cada variable.

Contribuciones de la tesis

Los aportes que esta memoria ofrece se engloban en tres aspectos de las redes Bayesianas: representación de la información cuantitativa, aprendizaje estructural e inferencia en redes credales.

Representación de la información cuantitativa

Como hemos comentado, tradicionalmente la información cuantitativa se ha expresado por medio de tablas de probabilidad (marginales o condicionales), aunque a lo largo de la bibliografía se han definido estructuras alternativas que se comportan mejor en ciertos aspectos como permitir sacar beneficio de los patrones de repetición de valores o la posibilidad de llevar a cabo tareas de aproximación sobre ellas, como es el caso de las otras dos representaciones que consideramos: árboles simples y podados de probabilidad. En esta memoria (Capítulo 4) proponemos unas nuevas estructuras, las que denominamos potenciales dirigidos por valor (VBPs). El objetivo de las VBPs es el de representar la información del modo más compacto posible. Esto se consigue almacenando cada valor de probabilidad de una vez y relacionando de algún modo los índices asociados a cada valor. Presentamos cuatro alternativas de representación: Los VBPs han demostrado su gran potencial sobre redes Bayesianas reales. Además, y al igual que los árboles de probabilidad, permiten ser aproximados, haciendo que las estructuras VBPs resultantes sean aún más compactas, asumiendo una pérdida de información aceptada.

Aprendizaje estructural

Sabemos que en el estudio de RBs se busca resolver dos problemas fundamentales: (1) aprendizaje (2) inferencia. Los problemas de aprendizaje se centran en definir tanto la estructura de grafo (aprendizaje estructural), como los parámetros de probabilidad asociados (aprendizaje paramétrico). Diversos algoritmos de aprendizaje estructural se han desarrollado en la bibliografía, considerando distintos enfoques. Muchos de ellos tienen la gran limitación de no poder

ser aplicados a redes de gran tamaño, ya que necesitan de unos cálculos complejos que, a menudo, exceden las capacidades computacionales. En esta tesis, se propone un método de aprendizaje estructural para RBs (aprendizaje del grafo asociado a los datos), desde un punto de vista variacional. Para ello utiliza inferencia variacional que busca resolver la probabilidad Bayesiana *a posteriori* sobre las diferentes estructuras de grafo, condicionada al conjunto de datos. Pese a que el algoritmo supone una primera aproximación y aún tiene ciertas limitaciones que podrán intentar ser resueltas en próximos trabajos, una experimentación inicial sugiere que se trata de una buena opción para ser utilizado sobre redes de tamaño considerable, con las que otros algoritmos no son capaces de trabajar.

Inferencia con redes credales

Realizar consultas probabilísticas sobre RBs es una tarea imprescindible para aumentar el conocimiento sobre un determinado problema; de esto se encarga la inferencia. El funcionamiento consiste en partir de una información (*i.e.* evidencia) dada sobre alguna de las variables, para propagarla al resto de variables y ver cómo se ven afectadas. En esta memoria vamos a estudiar el problema de inferencia sobre redes credales (*i.e.* redes Bayesianas con probabilidades imprecisas). Para ello se ha desarrollado un algoritmo de inferencia *branch-and-bound* paralelo que comienza con una solución aproximada inicial obtenida mediante *hill-climbing*, para a continuación realizar búsqueda en profundidad fijando las distintas variables transparentes. Esto permite estimar la mejor de las soluciones en cada ramificación y optar por seguir estudiando esa rama (si la mejor solución que se puede obtener siguiente esa rama es mejor que la encontrada hasta el momento) o no (en caso contrario). Como decimos, el algoritmo es paralelo, y es que utilizará un *pool* de hebras donde a cada hebra se le asignará la tarea de explorar una ramificación. Aunque es únicamente una aproximación al método y sería conveniente llevar a cabo una experimentación más extensa, la aplicación a dos redes credales muestra que las redes de más tamaño pueden potencialmente beneficiarse del algoritmo.

1 Fundamentos

Las redes Bayesianas (RBs) aúnan dos grandes teorías matemáticas, la de grafos y la de probabilidad. En esta sección asentaremos los principios básicos de ambos pilares. En primer lugar, en la Sec. 1.1 podemos ver aspectos básicos relacionados con grafos: cómo surgieron, definiciones relevantes y notación a utilizar. A continuación, la Sección 1.2 nos adentra en la teoría de la probabilidad, donde veremos los conceptos más básicos del trabajo con sucesos aleatorios y su extensión a variables aleatorias.

1.1 Teoría de grafos

Puesto que nuestro estudio se centra en las RBs, y como veremos más adelante, estas se representan de manera gráfica mediante un grafo, necesitaremos introducir algunos conceptos básicos sobre ellos. La teoría de grafos es la rama de la Matemática que estudia sistemas cuyos elementos se encuentran conectados. La gran ventaja de los grafos, es su poder de representación de un modo sencillo e intuitivo, en la que las relaciones entre los elementos que lo componen se perciben fácilmente. Veremos en esta sección los inicios de esta teoría, así como las definiciones y notación básica que nos serán útiles para comprender más tarde conceptos más complejos.

1.1.1 Los primeros grafos

El concepto de grafo fue introducido por primera vez en 1736, cuando el gran matemático Euler resolvió el conocido como "Problema de los puentes de Königsberg". Este problema habla sobre los 7 puentes que conectan las orillas del río Pregolya en la ciudad de Königsberg, conocida ahora como Kaliningrado. Y es que la población de entonces se preguntaba si era posible recorrer todos los puentes pasando una sola vez por cada uno de ellos. Euler transformó el problema en una serie de puntos unidos por líneas, y así es como dibujó el primer grafo de la historia. En la siguiente Imagen 1.1 podemos ver, el mapa actual de Kaliningrado en la parte izquierda (resaltados en color marrón tenemos los 5 puentes que aún se encuentran en pie, y en amarillo donde se encontraban

los dos restantes); y en la parte de la derecha el grafo correspondiente al problema, cuatro zonas (lo que ahora conocemos como nodos) unidas por los siete puentes (arcos):

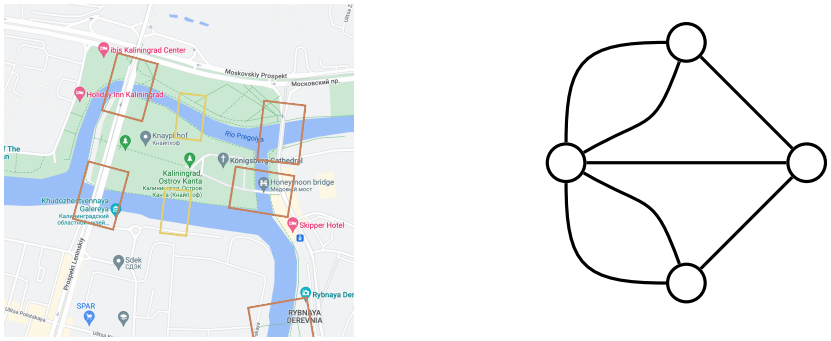


Fig. 1.1: Problema de los puentes de Königsberg. En la izquierda encontramos el mapa actual y en la derecha una representación del problema en forma de grafo.

1.1.2 Léxico, notación y definiciones relevantes

Definición 1.1.1 ***(Grafo)*** *Un* ***grafo*** *(también denominado* ***grafo simple****) queda definido por un par de conjuntos* $\mathcal{G} := \{\boldsymbol{X}, \boldsymbol{E}\}$*, donde* $\boldsymbol{X}$ *es un conjunto de nodos (o vértices),* $\{X_1, X_2, ..., X_n\}$*, y* $\boldsymbol{E}$ *un conjunto de pares de nodos de* $\boldsymbol{X}$ *(no necesariamente distinto del* $\emptyset$*),* $\{\{X_1, X_2\}, \{X_1, X_3\}, ...\}$*, indicando que existe un enlace (arco, o arista) entre ellos.*

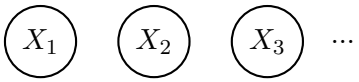
En un grafo $\mathcal{G}$, el número total de nodos, o cardinal de $\mathbf{X}$, $|\mathbf{X}|$, se denomina *orden* y el número total de aristas, o cardinal de $\mathbf{E}$, $|\mathbf{E}|$, se denomina *tamaño*.

Una arista que une los nodos X_1, X_2 se notará por simplicidad como X_1X_2. La existencia de la arista X_1X_2 convierte a los nodos X_1, X_2 en *extremos* de ella y hace que los nodos pasen a denominarse *vecinos* (o *adyacentes*). Dado un nodo X_1 y una arista e, diremos que X_1 es *incidente* con e si $\exists X_i \in \mathbf{X}$ tal que $e = X_1X_i$.

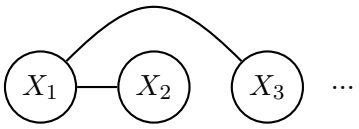
Cuando en el grafo encontramos un nodo con el que ninguno de los vértices es adyacente, decimos que se trata de un nodo *aislado*. Para un vértice, definimos su *grado* como el número de aristas incidentes con él. El *entorno* (o *vecindad*) de un nodo $X_1 \in \mathbf{X}$ es el conjunto de todos los nodos vecinos a él y se nota por $N(X_1)$, de manera rigurosa:

$$N(X_1) = \{X_i \in \mathbf{X} | \exists X_1X_i \in \mathbf{E}\}$$

En lo referente a la representación de un grafo, los nodos se verán como puntos, uno por cada uno de los nodos:



donde encontraremos que dos nodos se unen mediante una línea si y solo si entre ellos hay una arista:



Ejemplo 1 *Consideramos un sencillo ejemplo, el grafo* $\mathcal{G} = \{\boldsymbol{X}, \boldsymbol{E}\}$ *de tan sólo cuatro nodos, con algunas conexiones entre ellos:*

$\boldsymbol{X} = \{X_1, X_2, X_3, X_4\}$
$\boldsymbol{E} = \{X_1X_2, X_1X_3, X_3X_4\}$

Por lo tanto, el orden del grafo será 4 y su tamaño 3. Los entornos de cada uno de los nodos serían:

$$N(X_1) = \{X_2, X_3\}, \quad N(X_2) = \{X_1\}, \quad N(X_3) = \{X_1, X_4\}, \quad N(X_4) = \{X_3\}$$

Ninguno de los nodos del ejemplo es aislado. Y la representación de $\mathcal{G}$ *podría ser de la forma:*

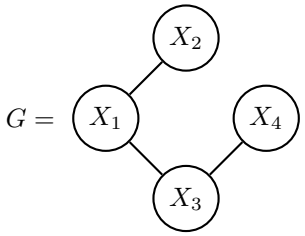


Fig. 1.2: Representación de grafo simple.

□

Dependiendo de la naturaleza de los conjuntos **X** y **E**, podemos definir los siguientes tipos de grafos:

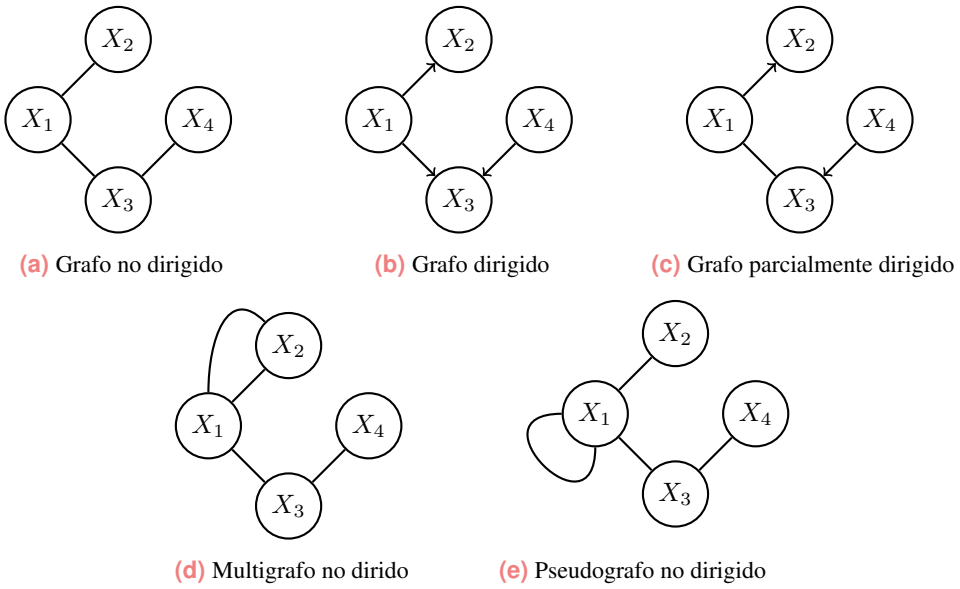
- Cuando las aristas son pares de nodos ordenados, se las denomina *aristas dirigidas* y el grafo resultante recibe el nombre de *grafo dirigido*. En caso contrario, la *arista* será *no dirigida* y el grafo será *no dirigido*. Si encontramos aristas dirigidas y no dirigidas, el grafo se denomina *parcialmente dirigido* (o *mixto*). En la representación gráfica, utilizaremos una flecha, en lugar de una línea, para indicar que la arista es dirigida; en este caso, si la arista comienza en el nodo X_1 y apunta hacia X_2, diremos que va de X_1 a X_2.

- Si se permite más de una arista uniendo el mismo par de nodos, el grafo se denominará *multigrafo*.

- En el caso particular de multigrafo, en el que existe una o varias aristas que unen un nodo consigo mismo, diremos que se presentan *lazos* y al grafo se le conocerá como *pseudografo*.

- Si las aristas conectan subconjuntos de nodos (en lugar de sólo dos), el grafo se llamará *hipergrafo*.

Ejemplo 2 *Modificando las aristas del ejemplo considerado en la Fig. 1.2, podríamos visualizar en la Figura 1.3 estos nuevos ejemplos de grafos:*



(a) Grafo no dirigido (b) Grafo dirigido (c) Grafo parcialmente dirigido

(d) Multigrafo no dirido (e) Pseudografo no dirigido

Fig. 1.3: Tipos de grafos

□

En el caso de encontrarnos con un grafo dirigido, dado un nodo $X_1 \in \mathbf{X}$, el conjunto de *padres de* X_1 se compone de los nodos cuyas aristas van de ellos a X_1 y se nota por $pa(X_1)$. Equivalentemente, el conjunto de *hijos de* X_1 se compondrá de los nodos con aristas existentes hacia ellos desde X_1, y notaremos por $ch(X_1)$ (por diminutivo de la palabra *children*, que en inglés significa "hijos").

Ejemplo 3 *Para ilustrar esta parte introducimos el siguiente grafo*

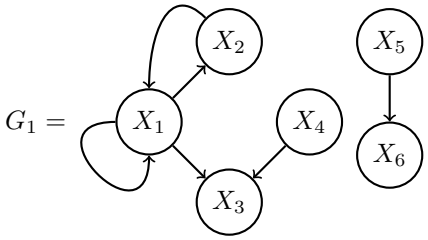


Fig. 1.4: Ejemplo de multigrafo dirigido.

Entonces, $\boldsymbol{X} = \{X_1, X_2, X_3, X_4, X_5, X_6\}$ *y* $\boldsymbol{E} = \{X_1X_1, X_1X_2, X_1X_2, X_1X_3, X_4X_3, X_5X_6\}$, *con orden* $|\boldsymbol{X}| = 6$ *y tamaño* $|\boldsymbol{E}| = 6$. *La arista* X_1X_2 *incide con los nodos* X_1 *y* X_2, *convirtiéndolos en vecinos, y la arista* X_1X_1 *se trata de un lazo.*
Además: $pa(X_1) = \{X_1, X_2\}$, $pa(X_2) = \{X_1\}$, $pa(X_3) = \{X_3, X_4\}$, $pa(X_4) = \emptyset$, $pa(X_5) = \emptyset$, $pa(X_6) = \{X_5\}$
y: $ch(X_1) = \{X_1, X_2, X_3\}$, $ch(X_2) = \{X_1\}$, $ch(X_3) = \emptyset$, $ch(X_4) = \{X_3\}$, $ch(X_5) = \{X_6\}$, $ch(X_6) = \emptyset$.

□

Un *camino* entre dos nodos de un grafo $X_0, X_n \in \mathbf{X}$ es una sucesión de nodos (no necesariamente distintos) que empieza en uno de ellos y acaba en el otro, donde cada nodo es adyacente con el nodo inmediatamente anterior y el posterior, $X_0, X_1, ..., X_{n-1}, X_n$. Además establecemos que el *camino* es *dirigido* si todas las aristas son dirigidas, y *no dirigido* si al menos una de las aristas no es dirigida. La *longitud* de un camino es el número de aristas que lo componen (contando también las aristas que se repiten). En la Fig. 1.4, el camino X_1, X_1, X_2 es dirigido, mientras que X_4, X_3, X_1, es no dirigido; ambos caminos tienen longitud 2. La *distancia* entre dos nodos de un grafo es la longitud del camino más corto que los une. Así, X_1 y X_4 se encuentran a distancia 2.

Si desde el nodo X_1 existe un camino dirigido hacia X_k, diremos que X_1 es *ascendiente* de X_k, y en este mismo caso, X_k se denominará *descendiente* de X_1. Así mismo, los nodos pueden adquirir nombres dependiendo de si existen arcos desde y hacia ellos. Diremos entonces que si un nodo no tiene padres, se denominará *nodo raíz* y si no tiene hijos recibirá el nombre de *nodo hoja*. En el Ejemplo 3, decimos que X_2 es descendiente de X_1; X_5 es nodo raíz y X_6, nodo hoja.

Si dado un camino los vértices inicial y final coinciden, entonces el camino se denomina *cerrado*. Si en el camino no aparecen aristas repetidas se llama *recorrido*. Si en un recorrido no hay vértices repetidos, se llamará *camino simple*. Si el recorrido es también camino cerrado, recibe el nombre de *circuito*. Y un circuito que es a su vez camino simple, se trata de un *ciclo*. Cuando el grafo no presenta ningún ciclo, recibe el nombre de *grafo acíclico*, en otro caso será un *grafo cíclico*. En la Fig. 1.4, tenemos que $\{X_1, X_2, X_1\}$ es un ciclo, así que nuestro grafo es cíclico.

Si para todo par de nodos del grafo $X_i, X_j \in \mathbf{X}$ existe un camino que los conecta, diremos que es un *grafo conexo*. En caso contrario, será un *grafo disconexo* y las partes del grafo que sí se encuentran conectadas se denominarán *componentes* del grafo. Definimos un *árbol* como un grafo conexo acíclico. Por lo tanto, el ejemplo tratado en la Fig. 1.4 sería un grafo disconexo de dos componentes, que podemos reconocer en la siguiente Fig. 1.5:

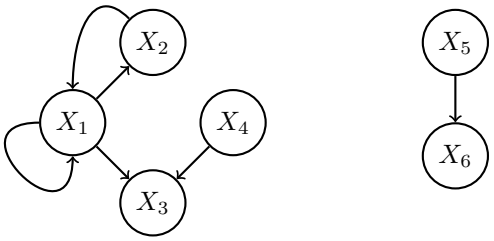


Fig. 1.5: Componentes del multigrafo pseudodirigido considerado, Fig. 1.4.

1.2 Teoría de probabilidad

En la naturaleza, podemos encontrar fenómenos determinísticos y aleatorios. Los fenómenos determinísticos son los que, bajo las mismas circunstancias, darán el mismo resultado. Así, sabemos que si soltamos un objeto nunca irá hacia arriba, siempre caerá. Por otro lado, tenemos los fenómenos aleatorios, que suponen algún grado de incertidumbre. Se podría decir que el

azar y la incertidumbre han estado siempre presentes en la mente humana. Así lo demuestran numerosos hallazgos en diversas civilizaciones, como los sumerios y asirios, que utilizaron lo que parecería el primer dado de la historia hecho de huesos de animales. Cómo medir la incertidumbre para conseguir tomar las mejores decisiones podría tratarse sencillamente desde un punto de vista lógico, pero tal y como se ha demostrado en multitud de ocasiones nos llevará a error. Así se demuestra en el trabajo de Tversky *et al.* [195], en el que se presentan varios problemas a un grupo de estudiantes y se estudian las razones por las que, en la mayoría de los casos, eligieron la opción incorrecta. Esto pone de manifiesto la necesidad de una ciencia que cuantifique esa incertidumbre, estableciendo una serie de reglas que eviten los errores sistemáticos; de lo que se encarga la teoría de la probabilidad.

En el siglo XVII se establecen los principios de teoría de probabilidad. En concreto, se fecha cuando Pascal, Fermat y el caballero de Méré [183] se enviaban correspondencia para intentar resolver un problema propuesto en 1654 por el último de ellos. El problema dice así:

"Dos jugadores, que han depositado una apuesta inicial, lanzan repetidamente una moneda, el primero gana si sale cara y el segundo si sale cruz. Han decidido que el primero que gane seis veces (consecutivas o no) se llevará el total de la apuesta. En un momento dado han salido (en cualquier orden) cinco caras y tres cruces y el juego debe ser interrumpido. ¿Cómo deben repartirse la apuesta?"

1.2.1 Probabilidad sobre sucesos aleatorios

En estadística y probabilidad, cuando hablamos de *experimento aleatorio* nos referimos a cualquier proceso del que conocemos el conjunto de los posibles resultados, pero al repetirlo bajo las mismas condiciones iniciales podemos obtener distintos resultados. Cada uno de esos posibles resultados se denomina *suceso elemental* y al conjunto de todos los posibles sucesos elementales se le denomina *espacio muestral* y se nota por Ω. Cuando el experimento aleatorio puede tomar un número infinito de resultados, lo denominamos *infinito*, y en caso contrario, será *finito*.

Según el tipo y número de sucesos elementales, podemos encontrar espacios aleatorios de dos tipos:

- Cuantitativos, donde los sucesos elementales son cantidades. Dentro de los espacios aleatorios cuantitativos encontramos una subdivisión:

  - Continuos - cuando el número de elementos es infinito no numerable. Este tipo de espacio se da especialmente cuando tenemos magnitudes físicas, como el peso, la altura, etc., por tratarse de medidas continuas.

  - Discretos - si el espacio es numerable. El lanzamiento de un dado es un ejemplo de este tipo, con sucesos elementales $\{1, 2, 3, 4, 5, 6\}$.

- Cualitativos, donde los sucesos elementales son cualidades. Como puede ser la extracción de bolas de una urna, donde los sucesos elementales serían los distintos colores de las bolas: rojo, amarillo, verde, azul.

Un *suceso aleatorio* (también denominado *suceso compuesto*, *evento*, o simplemente *suceso*) es cualquier subconjunto del espacio muestral que cumple una determinada propiedad. Diremos que dos *sucesos* son *iguales* si tienen los mismos sucesos elementales. Un *suceso* se dice *seguro* cuando se compone de todos los posibles resultados, es decir, el propio espacio muestral Ω; e *imposible* si es igual al vacío, $\emptyset$. Puesto que los sucesos son subconjuntos del espacio aleatorio, podemos utilizar el álgebra de conjuntos y representarlos mediantes diagramas de Venn.

Ejemplo 4 *Un ejemplo sencillo sería el estudio del resultado que se obtiene al lanzar un dado, donde el espacio muestral sería como comentamos* $\Omega = \{1, 2, 3, 4, 5, 6\}$*, compuesto por 6 sucesos elementales (cada uno de los valores). Un suceso aleatorio podría ser obtener un resultado par; un suceso seguro es que el número resultante será menor que 8 y un suceso imposible es que el resultado sea 7.*

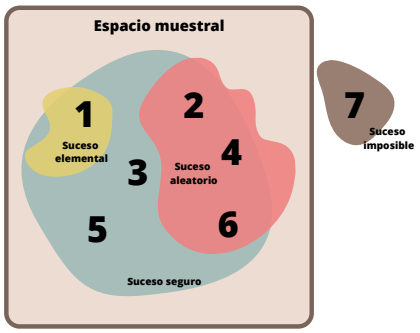


Fig. 1.6: Diagrama de Venn de posibles sucesos al lanzar un dado.

□

Considerando dos sucesos aleatorios, $S_1, S_2 \in \Omega$, podemos aplicar sobre ellos las típicas operaciones de teoría de conjuntos:

- *Contenido*: Diremos que S_1 se encuentra contenido en S_2 ($S_1 \subset S_2$), si siempre que ocurre S_1 también ocurre S_2.

- *Igualdad*: Se dice que dos sucesos son iguales ($S_1 = S_2$), si siempre que ocurre S_1 ocurre S_2 y viceversa. Equivalentemente $S_1 \subset S_2$ y $S_2 \subset S_1$.

- *Diferencia*: El suceso diferencia $S_1 - S_2$ (o S_1/S_2), se compone de todos los sucesos elementales que se encuentran en S_1 y no en S_2.
- *Unión*: La unión de los sucesos, $S_1 \cup S_2$, contiene todos los sucesos elementales que pertenecen al menos a uno de ellos.
- *Intersección*: La intersección de los sucesos, $S_1 \cap S_2$, está formada por los sucesos elementales que se encuentran contenidos en ambos sucesos aleatorios.
- *Incompatibilidad*: Dos sucesos son incompatibles si su intersección es vacía, $S_1 \cap S_2 = \emptyset$.
- *Complementariedad*: El suceso contrario (o complementario) a uno dado ($\overline{S_1}$), es el que tiene todos los sucesos elementales que no se encuentran en el primero. Se verificará entonces que $S_1 \cup \overline{S_1} = \Omega$ y $S_1 \cap \overline{S_1} = \emptyset$

Ejemplo 5 *Ilustremos estos conceptos con nuestro ejemplo del lanzamiento del dado. Consideremos los sucesos aleatorios* $S_1 = \{$*"resultado par"*$\} = \{2, 4, 6\}$,
$S_2 = \{$*"resultado menor o igual que 4"*$\} = \{1, 2, 3, 4\}$ *y*
$S_3 = \{$*"resultado impar y menor que 4"*$\} = \{1, 3\}$. *Entonces:*

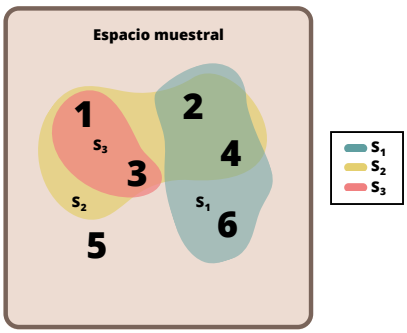


Fig. 1.7: Diagrama de Venn de tres sucesos al lanzar un dado y operaciones con ellos.

- $S_3 \subset S_2$
- $S_1 - S_2 = \{6\}$
- $S_1 \cup S_2 = \{1, 2, 3, 4, 6\}$
- $S_1 \cap S_2 = \{2, 4\}$
- S_1 *y* S_3 *son incompatibles.*
- $\overline{S_1} = \{1, 3, 5\}$

□

A lo largo de la historia, el concepto de *probabilidad* se ha definido basándose en diferentes perspectivas:

1. **Definición clásica de probabilidad o Regla de Laplace**. Fue definida por Laplace [121] y considera un experimento aleatorio en el que cualquiera de los sucesos es igualmente

posible de convertirse en resultado. Dado uno de los sucesos $S_1 \in S$ que se presenta m veces en el número total de los sucesos n, entonces se define la **probabilidad** de S_1 como:

$$P(S_1) = \frac{m}{n} \tag{1.1}$$

2. **Definición frecuentista**. Puesto que la definición clásica sólo considera dominios finitos de sucesos equiprobables, se establece en 1928 [201] una nueva definición que elimina esas restricciones.

   Digamos que al repetir un experimento aleatorio n veces, el suceso S_1 ocurre n_{S_1}, la frecuencia relativa de S_1 es $f_n(S_1) = \frac{n_{S_1}}{n}$. Considerando que el experimento se repite indefinidamente, aumentando el número de repeticiones, obtendremos distintas frecuencias relativas $f_n, f_{n+1}, ...$ Definimos la *probabilidad* de que ocurra el suceso $S_1 \in S$ como el límite de dichas frecuencias relativas:

$$P(S_1) = \lim_{n\to\infty} f_n(S_1) \tag{1.2}$$

3. **Definición axiomática de Kolmogorov**. Quizás la definición más aceptada y perfecta desde un punto de vista matemático es la efectuada por Kolmogorov en 1933 [159], generada a partir de tres axiomas que toda función que quiera considerarse probabilidad necesitará verificar y que se basa en el concepto de σ-álgebra:

   Definición 1.2.1 *(**σ-álgebra**) Sea un conjunto de sucesos medibles $S \in \wp(\Omega)$ [1] del espacio aleatorio de probabilidad Ω que cumple las siguientes propiedades:*

   - *Para todo suceso $S_i \in S$, su complementario también pertenece a S, $\overline{S_i} \in S$.*
   - *Dada una serie infinita numerable de sucesos finitos $\{S_1, S_2, ..., S_n\} \in S$, su unión también pertenece a S, $\{S_1 \cup S_2 \cup ... \cup S_n\} \in S$.*

   *Diremos entonces que S es una **σ-álgebra**.*

   Definición 1.2.2 ***(Probabilidad axiomática de Kolmogorov)** Siendo S una σ-álgebra, denominaremos **función de probabilidad** o simplemente **probabilidad** a la función $P : S \to \mathbb{R}$ que asigna a cada suceso $S_i \in S$ un valor que cuantifica su ocurrencia. P cumple necesariamente los siguientes axiomas, llamados axiomas de Kolmogorov:*

   - ***Axioma 1: (No negatividad)** P es definida positiva $0 \leq P(S_i)$ para todo $S_i \in S$*

[1] $\wp(\Omega)$ se denomina *partes de* Ω y se compone de todos los subconjuntos del espacio aleatorio

- ***Axioma 2: (Suceso seguro)*** *La probabilidad del suceso seguro es 1,* $P(\Omega) = 1$

- ***Axioma 3: (σ-aditividad)*** *Dada la colección numerable de sucesos* $\{S_i\}_{i\in\mathbb{N}} = \{S_1, S_2, ..., S_n\}$ *disjuntos dos a dos (i.e. la intersección de cada dos sucesos es vacía), la probabilidad de la unión de todos ellos es igual a la suma de las probabilidades de cada uno, es decir:*

$$P(\bigcup_{i=1}^{n} P(S_i)) = \sum_{i=1}^{n} P(S_i) \tag{1.3}$$

Por lo tanto, diremos que $P(S_i)$ *representa la probabilidad del suceso* S_i*.*

Intuitivamente, el primero de los axiomas fuerza a que ninguna probabilidad sea negativa; el segundo nos indica que dado el espacio aleatorio algo debe suceder; y el tercero establece que para cualquier colección de sucesos mutuamente excluyentes, la probabilidad de su unión siempre verificará ser la suma de las probabilidades de los sucesos.

Como consecuencia de la definición axiomática de probabilidad obtenemos las siguientes propiedades:

- La probabilidad del suceso complementario a uno dado S_1 cumple: $P(\overline{S_1}) = 1 - P(S_1)$.

- El suceso imposible tiene probabilidad nula: $P(\emptyset) = 0$.

- (Monotonía) Para cualesquiera dos sucesos S_i, S_j con $S_i \subseteq S_j$ se cumplirá: $P(S_i) \leq P(S_j)$.

- Para todo suceso S_1, $0 <= P(S_1) <= 1$.

- La probabilidad de la diferencia de dos sucesos S_1, S_2 con $S_1 \subset S_2$ verifica: $P(S_1 - S_2) = P(S_1) - P(S_2)$.

- (Regla de la adición) Dados dos sucesos, S_1 y S_2 cualesquiera, tendremos que: $P(S_1 \cup S_2) = P(S_1) + P(S_2) - P(S_1 \cap S_2)$.

  La generalización de la Regla de la adición se conoce como Principio de inclusión-exclusión y se expresa mediante la siguiente fórmula:

$$P(\bigcup_{i=1}^{n} S_i) = \sum_{i=1}^{n} P(S_i) - \sum_{i<j}^{n} P(S_i \cap S_j) + \sum_{i<j<k}^{n} P(S_i \cap S_j \cap S_k) + ... + (-1)^{(n+1)} \sum_{i<j<k...}^{n} P(S_i \cap S_j \cap S_k \cap ...)$$

Definición 1.2.3 ***(Espacio de probabilidad)*** *A la terna que se forma con el espacio muestral Ω, la σ-álgebra S y la probabilidad P, (Ω, S, P), la denominamos* ***espacio de probabilidad*** *(o* ***espacio probabilístico****).*

Dentro de la Probabilidad encontramos un concepto que nos será indispensable, la *dependencia* o *independencia* de sucesos; nos dice si la probabilidad de un suceso se ve modificada (o no) al darse el otro y viceversa. Para comprender este concepto, primero deberemos introducir otro, el de *probabilidad condicionada*. La probabilidad condicionada nos indica cómo de posible es que ocurra un determinado suceso considerando que conocemos información sobre otro (u otros) de los sucesos del espacio muestral. Esto supone que podamos calcular, por ejemplo, la probabilidad de que llueva un determinado fin de semana, conociendo la estación del año en la que se encuentra, lo que probablemente influirá en el resultado obtenido.

Definición 1.2.4 ***(Probabilidad condicionada)*** *Dado un espacio probabilístico (Ω, S, P) y dos sucesos $S_1, S_2 \in S$. Definimos entonces la* ***probabilidad de*** *S_1* ***dado*** *(o* ***condicionada a*** *) S_2, $P(S_1|S_2)$, como sigue:*

$$P(S_1|S_2) = \frac{P(S_1 \cap S_2)}{P(S_2)} \tag{1.4}$$

Donde obviamente el concepto sólo tiene sentido si $P(S_2) \neq 0$, lo que es lógico, ya que el caso contrario implicaría que $S_2 = \emptyset$ y no tendría sentido condicionar a él.

La probabilidad que acabamos de ver en la anterior Ecuación (1.4), $P(S_1 \cap S_2)$, se denomina **probabilidad conjunta** (o **compuesta**) de dos sucesos aleatorios $S_1, S_2 \in S$ y se denota como $P(S_1, S_2)$ por simplicidad. Su fórmula puede fácilmente deducirse de esta misma ecuación despejando el término:

$$P(S_1 \cap S_2) = P(S_1|S_2)P(S_2) \quad o \quad P(S_1 \cap S_2) = P(S_2|S_1)P(S_1) \tag{1.5}$$

Y puede extenderse a más de dos sucesos utilizando el siguiente teorema.

Teorema 1.2.1 ***(Teorema de la probabilidad compuesta)*** *Dado el espacio probabilístico (Ω, S, P) y sean $S_1, S_2, ..., S_{n-1} \in S$ verificando que $P(\cap_{i=1}^{n-1} S_i) > 0$. Entonces para cualquier otro suceso $S_n \in S$, se verifica:*

$$P(\cap_{i=1}^{n} S_i) = P(S_1)P(S_2|S_1)P(S_3|S_1 \cap S_2) \cdots P(S_n| \cap_{i=1}^{n-1} S_i) \tag{1.6}$$

Dada una serie de sucesos $S_1, S_2, ..., S_n \in S$, verificando que la unión de ellos resulta ser el espacio muestral, $\cup_{i=1}^{n} S_i = \Omega$, entonces ese conjunto de sucesos se denomina **sistema exhaustivo**

de sucesos. Si además son mutuamente excluyentes (o disjuntos), entonces constituyen una **partición de** Ω.

Teorema 1.2.2 ***(Teorema de la probabilidad total)*** *Dado el espacio probabilístico* (Ω, S, P) *y sean* $S_1, S_2, ..., S_n \in S$ *con* $P(S_i) > 0 \; \forall i = 1, 2, ..., n$*, partición de* Ω*. Entonces, dado otro suceso* $R \in S$*, se verifica que:*

$$P(R) = \sum_{i=1}^{n} P(R|S_i)P(S_i) \tag{1.7}$$

Teorema 1.2.3 ***(Teorema de Bayes)*** *Dado el espacio probabilístico* (Ω, S, P) *y sean* $S_1, S_2, ..., S_n \in S$ *con* $P(S_i) > 0 \; \forall i = 1, 2, ..., n$*, una partición de* Ω*. Entonces, dado otro suceso* $R \in S$*, se verifica que:*

$$P(S_i|R) = \frac{P(R|S_i)P(S_i)}{\sum_{i=1}^{n} P(R|S_i)P(S_i)} \tag{1.8}$$

El Teorema de Bayes es entonces una herramienta para conocer la denominada **probabilidad a posteriori** (o **probabilidad condicional inversa**), que cuantifica cómo de posible es que el suceso S_1 haya hecho que ocurra S_2, sabiendo que S_2 ya ha ocurrido. Esto se puede realizar, si conocemos la **probabilidad a priori** de la causa, $P(S_1)$, y la **verosimilitud** de que S_2 haya sido causado por S_1, $P(S_2|S_1)$.

Cuando un **suceso** S_1 no se ve afectado cuando sucede S_2, diremos que S_1 es **independiente** de S_2, en caso contrario se denominará **dependiente**. Así, si lanzamos un dado equilibrado dos veces, el hecho de que en la primera tirada salga 2 no afectará en el resultado que obtengamos en la segunda tirada. Esto se traduce en:

- Si S_1 y S_2 son sucesos independientes, entonces

$$P(S_1|S_2) = P(S_1) \quad y \quad P(S_2|S_1) = P(S_2) \tag{1.9}$$

- Si S_1 y S_2 son sucesos dependientes, entonces

$$P(S_1|S_2) \neq P(S_1) \quad y \quad P(S_2|S_1) \neq P(S_2) \tag{1.10}$$

Teorema 1.2.4 ***(Caracterización de independencia)*** *Dados dos sucesos* $S_1, S_2 \in S$*, con* $P(S_2) > 0$*, diremos que* S_1 *es independiente de* S_2 *sii se verifica:*

$$P(S_1 \cap S_2) = P(S_1)P(S_2) \tag{1.11}$$

Este teorema puede además extenderse a cuántos sucesos aleatorios se desee, resultando:

$$P(\bigcap_{i=1}^{n} S_i) = \prod_{i=1}^{n} P(S_i) \quad con\ S_i \in S \tag{1.12}$$

1.2.2 Extensión a variables aleatorias

En muchas ocasiones, podrá interesarnos no el suceso resultante del experimento aleatorio *per se*, sino alguna cualidad del mismo. Así, en el lanzamiento de un dado podríamos no necesitar el número obtenido, sino la paridad del resultado; o en un partido, podríamos desear saber qué equipo ha ganado. Surge entonces el concepto de variable aleatoria, término que fue usado por primera vez por Borel-Cantelli.

Una *variable aleatoria* es una función $X : \Omega \to \mathbb{R}$ que asigna un número a cada posible resultado del experimento aleatorio. Usaremos la notación usual para las variables aleatorias, las últimas letras romanas mayúsculas $X, Y, ...$ y por sus respectivas minúsculas a sus *valores* (o *estados*) $x, y, ...$ El *dominio de la variable* X se notará Ω_X, y al igual que en el estudio con sucesos, contendrá todos los posibles resultados del experimento aleatorio. El conjunto de todas las variables aleatorias del problema se notará como $\mathbf{X} = \{X_1, X_2, ..., X_N\}$, el *dominio* de $\mathbf{X}$ será $\Omega_{\mathbf{X}} = \{\Omega_{X_1} \times \Omega_{X_2} \times ... \times \Omega_{X_N}\}$ y cada uno de los elementos del dominio se conocerá como *configuración*.

Ejemplo 6 *Consideremos el experimento aleatorio en el que se lanza un dado equilibrado y la variable aleatoria X tomará el valor 1 si el resultado obtenido es impar y 2 si es par. El dominio de X será entonces $\{1, 2, 3, 4, 5, 6\}$ y la variable tendrá la forma:*

$$X(x) = \begin{cases} 1 & \text{si } x \in \{1, 3, 5\} \\ 2 & \text{si } x \in \{2, 4, 6\} \end{cases}$$

□

Las variables aleatorias se pueden clasificar en:

- Discretas, cuando a los sucesos elementales se le asignan números aislados, o lo que es lo mismo, cuando el conjunto de posibles valores es finito o infinito numerable. Es el caso del Ejemplo 6, en el que lanzamos un dado y anotamos si el resultado es par o impar.

- Continuas, si el conjunto de posibles valores es todo $\mathbb{R}$, o todo un intervalo contenido en $\mathbb{R}$. Una variable que estudia la altura de una determinada población pertenecería a esta clase.

En esta memoria consideraremos, por simplicidad, únicamente variables aleatorias discretas y cuyos estados serán mutuamente excluyentes, o lo que es lo mismo, no podrán darse dos simultáneamente.

Para cualquier variable aleatoria X existe una probabilidad inducida por ella P que conocemos como *distribución de probabilidad*. En todo caso, esta probabilidad necesariamente tendrá que cumplir que $\sum_{x_k \in \Omega_x} P(X = x_k) = 1$. Veamos la definición formal:

Definición 1.2.5 ***(Distribución de probabilidad discreta)*** *Dado un espacio probabilístico definido sobre una variable aleatoria X que toma los valores $\{x_1, x_2, ..., x_n\}$, definimos su* ***distribución de probabilidad*** *como la función:*

$$\begin{aligned} P : \Omega_X &\to [0, 1] \\ x_i &\longmapsto p_i \end{aligned}$$

donde $p_i = P(X = x_i)$ denota las probabilidades asociadas a cada valor x_i y verifican $\sum_{i=1}^{n} p_i = 1$.

Entonces, para cada x_i obtenemos lo que se conoce como su *distribución marginal*, $P(X = x_i)$, o simplemente $P(x_i)$ si es evidente por el contexto, esto nos permite expresar la distribución de probabilidad como el vector de las distribuciones marginales:

$$P(X) = \begin{bmatrix} P(x_1) \\ P(x_2) \\ \vdots \\ P(x_n) \end{bmatrix} \tag{1.13}$$

Ejemplo 7 *En la variable aleatoria X que definimos en el Ejemplo 6 y si el dado está equilibrado, es decir, la probabilidad de que salga cada cara es la misma para todas las caras, obtenemos la siguiente distribución de probabilidad P:*

$$\begin{aligned} P : \{p, i\} &\to \mathbb{R} \\ \{1\} &\to 0.5 \\ \{2\} &\to 0.5 \end{aligned}$$

□

En la gran mayoría de los casos, necesitaremos trabajar no con una variable aleatoria, sino con lo que se denomina *vector aleatorio*, que no es más que un conjunto de variables aleatorias $(X_1, X_2, ..., X_N)$. Estas variables tomarán el nombre de *componentes del vector aleatorio*. Diremos que el vector aleatorio es *discreto* si las variables que lo componen lo son, y continuo si son continuas. La distribución de probabilidad que se obtiene al considerar más de una variable aleatoria es la denominada *distribución conjunta de probabilidad*, $P(X_1, X_2, ..., X_N) = P(X_1 = x_1, X_2 = x_2, ..., X_N = x_N)$. El caso de dos variables X, Y, el vector aleatorio se puede expresar también como una matriz del siguiente modo:

$$P(X,Y) = \begin{bmatrix} P(x_1,y_1) & P(x_1,y_2) & \dots & P(x_1,y_m) \\ P(x_2,y_1) & P(x_2,y_2) & \dots & P(x_2,y_m) \\ \vdots & \vdots & \ddots & \vdots \\ P(x_n,y_1) & P(x_n,y_2) & \dots & P(x_n,y_m) \end{bmatrix} \tag{1.14}$$

Ejemplo 8 *Dado un experimento aleatorio en el que lanzamos un dado. Consideremos la variable aleatoria X definida anteriormente en el Ejemplo 6 y otra variable Y que asigna el valor 3 si el resultado al lanzar el dado es múltiplo de 3 y 0 en caso contrario, es decir:*

$$Y(y) = \begin{cases} 0 & \text{si } y \in \{1,2,4,5\} \\ 3 & \text{si } y \in \{3,6\} \end{cases}$$

con distribución de probabilidad de Y:

$$\begin{gathered} P : \{0,3\} \to \mathbb{R} \\ \{0\} \to 0.\widehat{6} \\ \{3\} \to 0.\widehat{3} \end{gathered}$$

Considerando entonces el vector aleatorio que estas dos variables definen $\boldsymbol{X} = \{X, Y\}$, con dominio $\Omega_{\boldsymbol{X}} = \Omega_X \times \Omega_Y = \{(1,0),(2,0),(1,3),(2,3)\}$ y la distribución conjunta de probabilidad tendrá la forma:

$$
\begin{array}{c}
P : \Omega_{\boldsymbol{X}} \to \mathbb{R} \\
\{(1,0)\} \to 0.\widehat{3} \\
\{(2,0)\} \to 0.\widehat{3} \\
\{(1,3)\} \to 0.1\widehat{6} \\
\{(2,3)\} \to 0.1\widehat{6}
\end{array}
\qquad
\begin{aligned}
P(X,Y) &= \begin{bmatrix} P(x_1,y_1) & P(x_1,y_2) \\ P(x_2,y_1) & P(x_2,y_2) \end{bmatrix} \\
&= \begin{bmatrix} P(1,0) & P(1,3) \\ P(2,0) & P(2,3) \end{bmatrix} \\
&= \begin{bmatrix} 0.\widehat{3} & 0.1\widehat{6} \\ 0.\widehat{3} & 0.1\widehat{6} \end{bmatrix}
\end{aligned}
$$

□

De manera análoga a como se indicó con sucesos, el Teorema de la probabilidad total se cumple en el caso de variables aleatorias.

Proposición 1.2.1 ***(Regla de la probabilidad total)*** *Dado un espacio probabilístico* $(\Omega, \boldsymbol{X}, P)$*, donde* $\boldsymbol{X} = (X, Y)$ *es el vector aleatorio,* $P(X, Y)$ *su distribución conjunta de probabilidad y* Ω_X, Ω_Y *representan los dominios disjuntos de estados mutuamente excluyentes de las variables aleatorias. Se verifica entonces que:*

$$
\begin{aligned}
\forall x \in \Omega_X \implies P(x) &= P(x,y_1) + P(x,y_2) + ... + P(x,y_m) \\
&= \sum_{y \in \Omega_Y} P(x,y)
\end{aligned}
$$

$$
\begin{aligned}
\forall y \in \Omega_Y \implies P(y) &= P(x_1,y) + P(x_2,y) + ... + P(x_n,y) \\
&= \sum_{x \in \Omega_X} P(x,y)
\end{aligned}
$$

Las fórmulas que acabamos de ver definen respectivamente las *distribuciones marginales* de X e Y. En el caso de representar el vector aleatorio como lo hicimos en (Ec. 1.14), para calcular la distribución marginal de la variable aleatoria X bastará con sumar los valores de la fila correspondiente y para la de Y, los valores de la columna.

Ejemplo 9 *Suponiendo las variables aleatorias de nuestro Ejemplo 8, tendremos que las distribuciones marginales serán:*

$$P(X = 1) = 0.\widehat{3} + 0.1\widehat{6} = 0.4\widehat{9}$$

$$P(X = 2) = 0.\widehat{3} + 0.1\widehat{6} = 0.4\widehat{9}$$

$$P(Y=0) = 0.\widehat{3} + 0.\widehat{3} = 0.\widehat{6}$$
$$P(Y=3) = 0.1\widehat{6} + 0.1\widehat{6} = 0.\widehat{3}$$

□

En el caso de variables aleatorias, al igual que con sucesos, podemos preguntarnos cómo se comporta la probabilidad de un subconjunto de ellas en el caso de que otro subconjunto ocurra, lo que conocemos como probabilidad condicionada. Veamos este concepto para el caso de vectores aleatorios de dimensión dos, el más sencillo.

Definición 1.2.6 ***(Distribución condicionada de vector discreto)*** *Dado un vector aleatorio discreto de dimensión dos, $\boldsymbol{X} = \{X, Y\}$, en un experimento aleatorio en el que conocemos el resultado de la variable X, con $P(X) \neq 0$, la **probabilidad** de Y **condicionada** a X puede calcularse como:*

$$P(Y|X) = \frac{P(X,Y)}{P(X)} \tag{1.15}$$

donde se verifica $\sum_Y P(Y|X) = 1$.

Esta definición puede ser fácilmente extendida a una dimensión mayor, donde se condiciona un subconjunto de variables $X_A \subset \mathbf{X}$, que reciben el nombre de *variables condicionadas*, a otro $X_B \subset \mathbf{X}$, que serán las variables condicionantes:

$$P(X_A|X_B) = \frac{P(X_A, X_B)}{P(X_B)} \tag{1.16}$$

donde, por supuesto, $P(X_B) \neq 0$ y $\sum_{x_A \in \Omega_{X_A}} P(X_A|X_B) = 1$.

Ejemplo 10 *Considerando de nuevo nuestro Ejemplo 8, supongamos que sabemos que el resultado es par, es decir, $X = 2$, ¿cuál es la probabilidad de que el resultado no sea múltiplo de tres, o lo que es lo mismo, $Y = 0$? ¿y de que sí lo sea, o $Y = 3$? Podremos calcularlas gracias a la fórmula (1.15):*

$$P(Y=0|X=2) = \frac{P(X=2,Y=0)}{P(Y=0)} = \frac{0.\widehat{3}}{0.\widehat{6}} = 0.4\widehat{9}$$

$$P(Y=3|X=2) = \frac{P(X=2,Y=3)}{P(Y=3)} = \frac{0.1\widehat{6}}{0.\widehat{3}} = 0.4\widehat{9}$$

□

Independencia entre variables aleatorias

Ya definimos anteriormente ese concepto tan esencial en el cálculo de probabilidades, la independencia de dos sucesos aleatorios cualquiera del espacio aleatorio y vimos que de un modo intuitivo, dos sucesos eran independientes si la ocurrencia de uno no afectaba al otro. Por extensión, dos variables aleatorias serán independientes [128] si lo son los eventos que generan. Procedamos a estudiar la independencia entre subconjuntos de variables aleatorias de **X** de un modo más riguroso.

Definición 1.2.7 ***(Independencia de variables aleatorias discretas)*** *Sea* ***X*** *un vector aleatorio discreto cuyas variables aleatorias están definidas sobre el mismo espacio de probabilidad. Diremos que el subconjunto de* ***variables aleatorias*** $X_A \subset \boldsymbol{X}$ *es* ***independiente*** *(o* ***marginalmente independiente****, o* ***mutuamente independiente****) del subconjunto* $X_B \subset \boldsymbol{X}$ *si se cumple que:*

$$P(X_A|X_B) = P(X_A) \tag{1.17}$$

o equivalentemente:

$$P(X_B|X_A) = P(X_B) \tag{1.18}$$

Notaremos entonces $I(X_A \perp X_B)$*. En caso de no cumplirse, concluiremos que las variables son dependientes.*

Teorema 1.2.5 ***(Caracterización de independencia de variables aleatorias discretas)*** *En las mismas condiciones que en la definición 1.2.7, decimos que:*

$$X_A, X_B \; independientes \Leftrightarrow P(X_A, X_B) = P(X_A)P(X_B) \tag{1.19}$$

Ejemplo 11 *Volvamos una vez más a nuestro vector aleatorio definido en el Ejemplo 8, ¿serán las variables aleatorias* X *e* Y *independientes? Comprobémoslo:*

$$\begin{aligned} P(X=1,Y=0) &= P(X=1)\cdot P(Y=0)? \\ 0.\widehat{3} &= 0.4\widehat{9}\cdot 0.\widehat{6} \\ 0.\widehat{3} &= 0.\widehat{3} \end{aligned}$$

$$\begin{aligned} P(X=2,Y=0) &= P(X=2)\cdot P(Y=0)? \\ 0.\widehat{3} &= 0.4\widehat{9}\cdot 0.\widehat{6} \\ 0.\widehat{3} &= 0.\widehat{3} \end{aligned}$$

$$
\begin{aligned}
P(X=1,Y=3) &= P(X=1)\cdot P(Y=3)? \\
0.1\widehat{6} &= 0.4\widehat{9}\cdot 0.\widehat{3} \\
0.1\widehat{6} &= 0.1\widehat{6}
\end{aligned}
$$

$$
\begin{aligned}
P(X=2,Y=3) &= P(X=2)\cdot P(Y=3)? \\
0.1\widehat{6} &= 0.4\widehat{9}\cdot 0.\widehat{3} \\
0.1\widehat{6} &= 0.1\widehat{6}
\end{aligned}
$$

Podemos por tanto asegurar que X y Y son variables independientes.

□

Definición 1.2.8 ***(Independencia dos a dos de variables aleatorias discretas)*** *Sea* $\boldsymbol{X} = \{X_1, X_2, ..., X_N\}$ *un vector aleatorio discreto cuyas variables aleatorias están definidas sobre el mismo espacio de probabilidad. Decimos que estas* ***variables*** *son* ***independientes dos a dos*** *si* $\forall i, j = 1, 2, ..., N$ *con* $i \neq j$*,* X_i *y* X_j *son independientes.*

Proposición 1.2.2 *Dadas las mismas condiciones que en la Definición 1.2.7, si las variables son independientes* $\implies$ *las variables son independientes dos a dos. En general, el contrario no es cierto.*

Para el desarrollo de esta memoria necesitaremos introducir el concepto de otro tipo de independencia, la llamada independencia condicional de dos subconjuntos de variables dado otro.

Definición 1.2.9 ***(Independencia condicional de variables aleatorias discretas)*** *Sea* $\boldsymbol{X}$ *un vector aleatorio discreto cuyas variables aleatorias están definidas sobre el mismo espacio de probabilidad. Diremos que el subconjunto de* ***variables aleatorias*** $X_A \subset \boldsymbol{X}$ *es* ***independiente condicionalmente*** *del subconjunto* $X_B \subset \boldsymbol{X}$ *dado el subconjunto* $C \subset \boldsymbol{X}$ *si* $P(X_B, C) \neq 0$ *y se cumple que:*

$$P(X_A|X_B, C) = P(X_A, C) \tag{1.20}$$

o equivalentemente si $P(X_A, C) \neq 0$ *y*

$$P(X_B|X_A, C) = P(X_B, C)$$

Notaremos entonces $I(X_A \perp X_B, C)$*.*

Es claro, que en el caso en que $C = \emptyset$, se daría independencia marginal entre los subconjuntos de variables.

Teorema 1.2.6 ***(Caracterización de independencia condicional de variables aleatorias discretas)*** *En las mismas condiciones que en la definición 1.2.9, decimos que:*

$$X_A, X_B \text{ son independientes dado } X_C \Leftrightarrow P(X_A, X_B|X_C) = P(X_A|X_C)P(X_B|X_C) \tag{1.21}$$

Los tres tipos de independencia que acabamos de estudiar implican que la independencia de las variables se cumple sea cual sea el valor dado de las variables aleatorias, pero hay tipos de independencia más generales. El primero que vamos a tratar, la *independencia específica del contexto* [26, 115] (CSI, por sus siglas en inglés - *context-specific indenpendence*), considera que subconjuntos de variables aleatorias pueden ser independientes para un sólo valor de la variable dada.

Definición 1.2.10 ***(Independencia específica del contexto)*** *Sea* ***X*** *un vector aleatorio discreto cuyas variables aleatorias están definidas sobre el mismo espacio de probabilidad. Dados los subconjuntos de variables aleatorias X_A, X_B, X_C, C. Diremos que X_A y X_B son* ***independientes específicos dado*** *X_C y el* ***contexto*** *$C = c$, si se cumple que:*

$$P(X_A|X_B, X_C, c) = P(X_A|X_C, c) \tag{1.22}$$

siempre y cuando $P(X_B, X_C, c) \neq 0$. Notaremos entonces $I(X_A \perp X_B, X_C, c)$.

Dos subconjuntos de variables serán *parcialmente independientes* si lo son para un subconjunto de valores del dominio de uno de los subconjuntos de variables.

Definición 1.2.11 ***(Independencia parcial)*** *Sea* ***X*** *un vector aleatorio discreto cuyas variables aleatorias están definidas sobre el mismo espacio de probabilidad. Dados los subconjuntos disjuntos de variables aleatorias X_A, X_B, diremos que se trata de subconjuntos* ***parcialmente independientes*** *si se cumple que:*

$$P(X_A|X_B = x_B^*) = P(X_A) \tag{1.23}$$

con $x_B^* \in \Omega_{X_B^*}$, donde $\Omega_{X_B^*} \subset \Omega_{X_B}$ y $P(X_B = x_B^*) \neq 0$

Dado el caso en el que los subconjuntos de variables aleatorias X_A y X_B son parcialmente independientes dado el contexto, entonces se dice que son *parcialmente independientes específicas del contexto* (PCI, por sus siglas en inglés - *parcial conditional independence*) [129, 155].

2 Modelos gráficos probabilísticos

Puesto que en el anterior Capítulo 1 quedaron asentadas las bases de la teoría de grafos (ver Sec. 1.1) y probabilidad (ver Sec. 1.2), podemos adentrarnos en los conocimientos más específicos a nuestra memoria. En este capítulo veremos la teoría de los llamados modelos gráficos probabilísticos (MGPs) en general (Sec. 2.1) y más particularmente de un subconjunto de ellos, las redes Bayesianas (RBs). Sobre las RBs veremos su concepto (ver Sec. 2.2), para después tratar los principales problemas a resolver con ellas: aprendizaje (ver Sec. 2.3) e inferencia (ver Sec. 2.4). Además, ilustraremos los nuevos conceptos con un conocido ejemplo de red Bayesiana, la red *Asia* [124].

2.1 Introducción a los modelos gráficos probabilísticos

Los modelos gráficos probabilísticos (MGPs) [115, 123, 153, 187] permiten modelizar eficientemente problemas que presentan incertidumbre. La idea básica es que los MGPs sólo considerarán las relaciones de dependencia relevantes en un determinado problema y esas serán las que se incluirán en el modelo probabilístico, lo que supone reducir complejidad y permite que los cálculos probabilísticos necesiten menos recursos computacionales.

Todo MGP queda definido por sus dos componentes:

- **Componente cualitativa**: representada por un grafo. Estos grafos representan visualmente las variables aleatorias (en forma de nodos) y las relaciones (condicionales) de dependencia e independencia existentes entre ellas (por medio de aristas que unen los nodos, o la ausencia de aristas).
- **Componente cuantitativa**: formada por una serie de parámetros numéricos que cuantifican el grado de dependencia e independencia entre las variables aleatorias del problema que se observaba en el grafo.

De un modo más riguroso, definimos un MGP como una terna, $(\mathbf{X}, P, \mathcal{G})$, donde $\mathbf{X}$ nota el conjunto de variables aleatorias involucradas en el problema, y P y $\mathcal{G}$ son respectivamente la

distribución de probabilidad y el grafo asociados a ese mismo problema. P y $\mathcal{G}$ están relacionados, ya que cada relación de dependencia que quede definida en $\mathcal{G}$, necesariamente se corresponderá con una distribución condicional en P, y viceversa.

Podemos clasificar los MGPs atendiendo a si el grafo que lo representa es:

- Dirigido, donde se define una relación de dependencia padre-hijo entre las variables. El MGP más utilizado y conocido en esta categoría son las redes Bayesianas [87, 103, 120, 149, 150, 156]; aunque también se encuentran otros como los diagramas de influencia [31, 98, 112, 147, 178, 200] o las cadenas de Markov [4, 145].
- No dirigido, donde las relaciones entre las variables se entenderán simétricas. Esto quiere decir que se sabe que ambas variables se encuentran relacionadas, pero no que una de ellas sea padre y la otra hijo. Las redes de Markov son MGPs no dirigidos [15, 20, 168, 188].

2.2 Redes Bayesianas

Las redes Bayesianas (RBs) [87, 149, 150, 156], también conocidas como *redes de creencia*, son un tipo de modelo gráfico probabilístico, cuya peculiaridad es que el grafo que describe el problema es un grafo dirigido acíclico (DAG, por sus siglas en inglés - *directed acyclic graph*). Así, si encontramos un arco que va dirigido de la variable X a la Y, sabremos que X causa o influye sobre Y. Esto sugiere cómo las variables se relacionan entre ellas, ya sea de manera directa o indirecta (mediante otra u otras variables). Podremos ver que cada variable del problema tendrá asociada una tabla condicional de probabilidad (CPT, por sus siglas en inglés - *conditional probability table*), que representa la distribución de probabilidad condicional de la variable para cada combinación de valores de sus padres en el grafo. En el caso de que la variable aleatoria no presente padres, esa tabla de probabilidad será simplemente la probabilidad marginal de la variable.

Para una mejor ilustración y comprensión de los conocimientos que en esta sección trataremos, introducimos un clásico ejemplo de red Bayesiana. Este ejemplo fue descrito por primera vez por Lauritzen y Spiegelhalter en 1988 [124] y se encuentra en el repositorio de *bnlearn* [10], donde definen el problema como sigue:

Ejemplo 12 ***(Red Bayesiana Asia)*** *La dificultad para respirar (disnea) puede deberse a tuberculosis, cáncer de pulmón o bronquitis, o a ninguna de ellas, o a más de una. Una visita reciente a Asia aumenta las posibilidades de tuberculosis, mientras que se sabe que fumar es un factor*

de riesgo tanto para el cáncer de pulmón como para la bronquitis. Los resultados de una sola radiografía de tórax no discriminan entre cáncer de pulmón y tuberculosis, como tampoco la presencia o ausencia de disnea.

Tenemos entonces 8 variables involucradas en el problema, todas discretas y binarias, cuyos posibles estados son, para todas ellas, Sí o No; indicando la ocurrencia o no. Entre paréntesis podemos ver la letra con la que denotaremos cada variable: Disnea (D), Tuberculosis (T), Cáncer de pulmón (L), Bronquitis (B), Visita a Asia (A), Fumador (S), Radiografía de tórax (X), Tuberculosis o cáncer de pulmón (E).

En la siguiente Fig. 2.1, podemos observar el grafo. Junto a cada una de las variables del problema, encontramos su tabla de probabilidad (condicionada o marginal) asociada. Queda, por tanto, completamente especificada la red Bayesiana Asia.

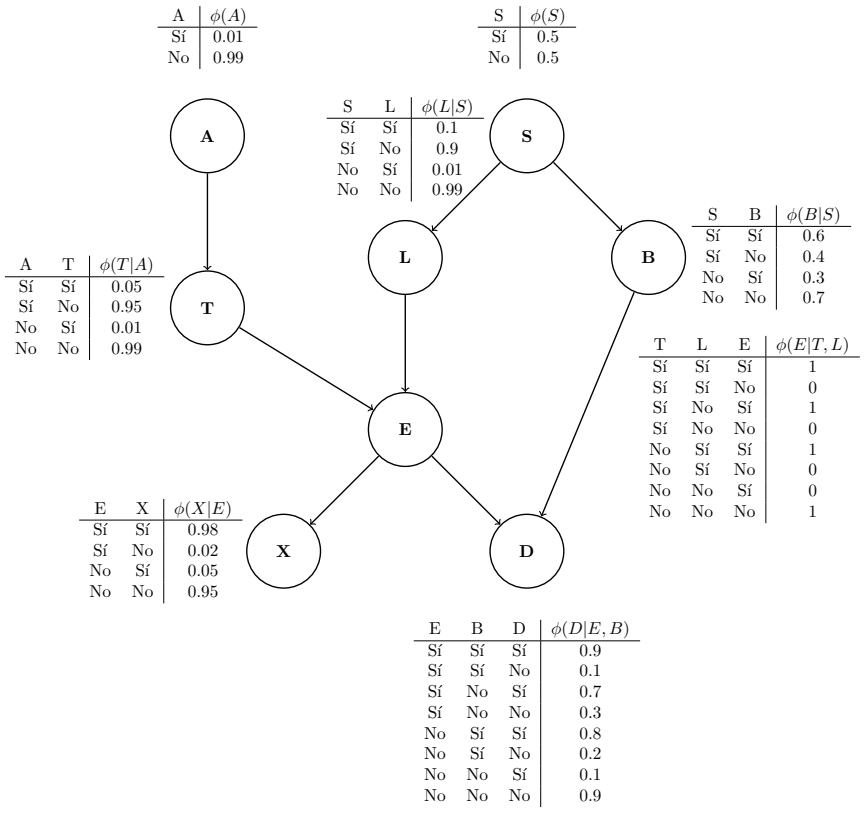


Fig. 2.1: Red Bayesiana *Asia*.

□

Como ya sabemos, toda red Bayesiana queda gráficamente representada por un DAG y analíticamente expresada por sus tablas de probabilidad marginales o condicionales. En la Sección 1.2, que dedicamos a asentar las bases sobre probabilidad, vimos cómo estudiar las dependencias entre variables de manera analítica y nos queda fijar cómo será posible estudiar esas mismas independencias sólo observando el DAG. En el estudio de RBs, será indispensable conocer el llamado criterio de *d-separación*, que permite estudiar relaciones de dependencia-independencia entre variables basándose en el DAG y sin recurrir a ningún cálculo probabilístico.

En general, establecemos que en un momento dado, cada una de las variables aleatorias toma uno solo de sus estados. El conocimiento del estado de una o varias variables es lo que se conoce como *evidencia*. La evidencia se denomina *fuerte* si las variables toman valores exactos y *suave* en caso contrario. Cuando tenemos evidencia sobre una variable, decimos que se encuentra *inicializada* (u *observada* o *instanciada*). Para establecer relaciones entre variables donde un cambio en una de ellas afecta a las que se encuentran relacionadas, utilizaremos el criterio gráfico de *separación directa* [78, 153], abreviado como *d-separación*, por ser el más extendido y por ser capaz de detectar todas las independencias entre nodos sin llevar a cabo ningún cálculo de probabilidades. Antes de introducir dicho criterio y puesto que estamos centrados en grafos dirigidos, veamos todas las posibles formas en que dos nodos cualesquiera del grafo $X, Y \in \mathbf{X}$ pueden estar conectados a través de un tercero $Z \in \mathbf{X}$:

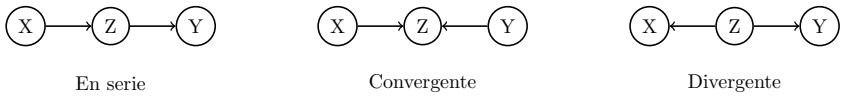


Fig. 2.2: Posibles conexiones entre tres nodos de un grafo.

Consideremos que queremos saber si dos nodos X, Y están d-separados. Para ello necesitaremos comprobar si todos los caminos no dirigidos que los unen están bloqueados, o lo que es lo mismo, necesitaremos verificar si alguna de las conexiones del camino se encuentran bloqueadas. Dadas los tres tipos de uniones comentados en Fig. 2.2:

- Conexión en serie / conexión cabeza - cola: si Z no se encuentra observado, X e Y se influirán mutuamente, la información puede fluir y la conexión no se encuentra bloqueada. El caso contrario ocurre cuando sí hay evidencia sobre Z, el camino se bloquea.

- Conexión divergente / conexión cola - cola: nos encontramos en este caso con un padre (Z) y dos hijos (X e Y). De igual modo que con la conexión en serie, si no hay evidencia sobre Z la conexión no se bloquea, pero sí lo hace en caso de haber evidencia sobre Z.

- Conexión convergente / conexión cabeza - cabeza: ocurre lo contrario que en los dos casos anteriores, la información sólo fluye cuando Z o alguno de sus descendientes se encuentran observados.

Definición 2.2.1 ***(D-separación)*** *Dados los conjuntos de nodos $X, Y, Z \subset \boldsymbol{X}$ disjuntos dos a dos, diremos que X e Y se encuentran **d-separados** dado Z sii en cada camino indirecto que los unen existe un nodo $A \in \boldsymbol{X}$ que verifica una de las siguientes opciones:*

1. *La conexión es en serie o divergente y $A \in Z$.*

2. *La conexión es convergente y A ni ninguno de sus descendientes pertenecen a Z.*

*En el caso opuesto, diremos que X, Y son **d-conexos** dado Z.*

Ejemplo 13 *Visualicemos el criterio de d-separación, acudiendo al grafo de la red Asia propuesto en el Ejemplo 12. Recordemos el grafo correspondiente, que puede observarse en la Figura 2.3. Y centrémonos en dos subconjuntos de variables, los de las Figuras 2.4(a) y 2.4(b) que recogen las relaciones entre las variables S, B, D y T, L, E respectivamente.*

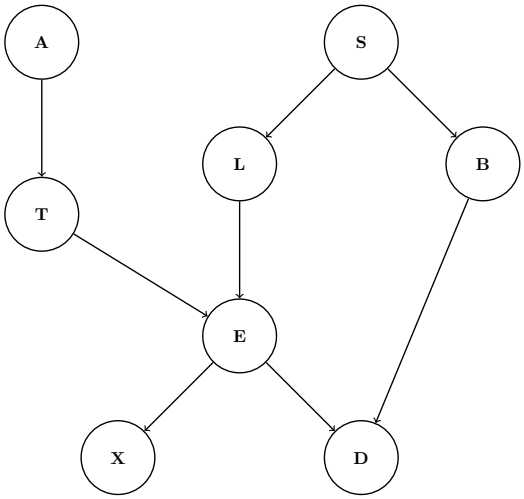
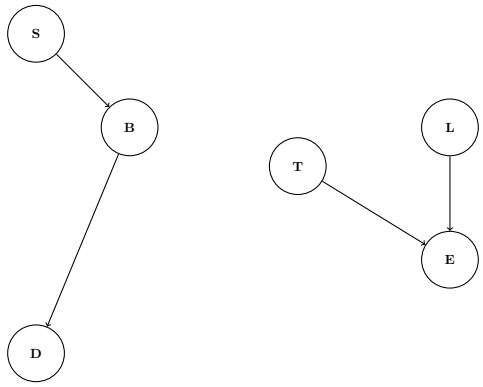


Fig. 2.3: Grafo correspondiente a la red Bayesiana *Asia*.



(a) (b)

Fig. 2.4: Subconjuntos de variables de la red *Asia*.

□

Puesto que en la figura de la izquierda Fig. 2.4(a) observamos que se trata de una conexión en serie, si tenemos conocimiento de que el paciente presenta bronquitis, el hecho de ser o no fumador no afectará a la presencia de disnea. Mientras que en la figura de la derecha Fig. 2.4(b), si la variable E se encuentra observada, es decir, el conocimiento de que el paciente tiene cáncer de pulmón o tuberculosis, sabremos que modificará ambas variables T y L.

Los problemas que se tratan desde un punto de vista probabilístico se centran en el cálculo y uso de la distribución conjunta de probabilidad. Por desgracia, el número de parámetros necesarios para especificarla crece exponencialmente con el número de variables. De hecho, en un problema en el que tenemos N variables binarias, necesitaríamos 2^N parámetros para representar la distribución conjunta de las variables. Por suerte, en el trabajo con RBs se utiliza la regla de la cadena [101], que simplifica los cálculos atendiendo a las relaciones de dependencia que se producen entre las variables.

Teorema 2.2.1 ***(Regla de la cadena para RBs)*** *Dada una RB definida sobre las variables aleatorias* $\boldsymbol{X} = \{X_1, X_2, ..., X_N\}$. *Esa RB determina entonces de manera unívoca una distribución conjunta de probabilidad de las variables,* $P(\boldsymbol{X})$*, dada como producto de todas las tablas de probabilidad condicional de la RB:*

$$P(\boldsymbol{X}) = \prod_{i=1}^{N} P(X_i|pa(X_i)) \tag{2.1}$$

Esta ecuación reduce considerablemente el número de parámetros necesarios para generar la distribución conjunta de probabilidad de las variables. Veamos como aprovecharíamos la regla de la cadena en la red Bayesiana *Asia* definida en Ejemplo 12.

Ejemplo 14 *Puesto que la red Asia se compone de 8 variables bivariadas, para calcular su distribución conjunta de probabilidad en teoría necesitaríamos* 2^8*, es decir, 256 valores. Por el contrario, una vez definidas las distribuciones de probabilidad condicionadas, veamos que calculando el número de parámetros necesarios para especificar cada una de esas distribuciones condicionadas y aplicando la regla de la cadena, el número de parámetros disminuye.*

1. *Para especificar el tamaño de la distribución conjunta de probabilidad necesitaremos calcular el número de parámetros de las distribuciones de probabilidad marginales* $P(A)$ *y* $P(S)$ *(por tratarse de variables sin padres) y de las distribuciones de probabilidad de las variables condicionadas a sus padres:* $P(T|A), P(L|S)$*,* $P(B|S), P(E|T,L), P(X|E)$ *y* $P(D|X)$*.*

2. *La regla de la cadena (Ecuación (2.1)) se vería como sigue, donde encima de cada probabilidad podemos observar el número de parámetros correspondientes a cada una de las distribuciones marginales y condicionadas:*

$$\begin{aligned} P(\{A,S,T,L,B,E,X,D\}) &= \overset{2}{P(A)} \cdot \overset{2}{P(S)} \cdot \overset{4}{P(T|A)} \cdot \overset{4}{P(L|S)} \cdot \overset{4}{P(B|S)} \cdot \\ &\quad \cdot \overset{8}{P(E|T,L)} \cdot \overset{4}{P(X|E)} \cdot \overset{8}{P(D|E,B)} \end{aligned}$$

3. *Así que es posible calcular la distribución conjunta de probabilidad especificando únicamente 36* $(= 2+2+4+4+4+8+4+8)$ *parámetros, en lugar de los 256* $(= 2^8)$ *que la especificación completa requeriría.*

□

2.3 Aprendizaje de redes Bayesianas

En este punto nos centraremos en estudiar cómo construir una RB, así como algunos de los métodos más conocidos para ello. De manera general, para especificar una RB deberán seguirse los siguientes tres pasos:

- Definición de las variables aleatorias y sus estados: será necesario establecer todas las variables que van a formar parte de la red (que serán los nodos del DAG) y conocer los estados que pueden tomar.
- Definición de la estructura: el segundo paso será establecer qué relaciones de dependencia se producen entre esas variables (o lo que es lo mismo, fijar los arcos del DAG).
- Definición de los parámetros: una vez el grafo está definido, se procede a detallar todos los valores de probabilidad de cada una de las tablas de probabilidad (marginales o condicionadas) que tienen sentido atendiendo a las relaciones entre las variables que se definieron en el paso anterior.

El proceso que acabamos de describir puede realizarse completamente a mano por un experto en la materia, pero lo cierto es que a menudo esto es muy costoso o no es posible, ya que el proceso es bastante largo y tedioso. Incluso en los casos de RBs de tamaño moderado, disponer de un experto puede resultar costoso o su conocimiento puede ser insuficiente. Es por eso, que suele ser más habitual aprender la RB inducida por los datos, en ocasiones con ayuda de conocimiento experto.

Las técnicas de aprendizaje automático disponibles para representar una RB partiendo de los datos se centran en aprender la estructura (aprendizaje estructural) y los parámetros (aprendizaje paramétrico) de la RB.

2.3.1 Aprendizaje estructural de RBs

Como hemos comentado en la introducción de esta sección, el aprendizaje estructural de una RB consiste en generar la estructura (*i.e.* el grafo) relativo al problema en cuestión. Generar métodos que partan de los datos y aprendan automáticamente la estructura gráfica que mejor los representa ha sido y es objeto de investigación [92, 143, 185]; especialmente para el caso de redes que crecen en tamaño. Conceptualmente, cualquier algoritmo comprendido en el aprendizaje estructural de RBs a partir de los datos pertenece a alguna de estas dos categorías:

- **Métodos basados en restricciones -** La característica principal de estos métodos es que realizan pruebas de independencia condicional repetidamente para establecer las relaciones entre las variables del problema. En general, comienzan con el grafo completo, es decir, consideran que todas las variables están relacionadas, para a continuación ir eliminando sucesivamente las aristas del DAG que no tienen sentido al realizar los test de independencias. Un conocido método basado en restricciones es el algoritmo PC [185],

que parte del grafo completo y en primer lugar comprueba las dependencias de nivel 0, es decir, las independencias marginales entre pares de variables, a continuación considera las de nivel 1, donde se estudia la independencia entre pares de variables dada una tercera y proceder sucesivamente aumentando complejidad.

- **Métodos basados en puntuaciones -** Estos métodos se basan en funciones, como por ejemplo la *log-likelihood*, que miden el grado de ajuste del grafo a los datos. Así, el método busca los posibles DAGs para quedarse con el que maximiza la función elegida. Algunas de las funciones más usadas incluyen métricas Bayesianas de Dirichlet [50, 92], criterio de información Bayesiano [137], longitud mínima de descripción [25, 119] y entropía [49]. Uno de los algoritmos basados en puntuación clásico más conocido es el de ascensión de colinas (HC, por sus siglas en inglés - *hill-climbing*) [169]. El algoritmo K2 [49], se trataría de otra popular opción. Y también los métodos Monte Carlo [69, 144].

En los últimos años, se han propuesto **métodos híbridos**, que combinan las dos aproximaciones anteriores. En primer lugar, se utilizan métodos basados en restricción para acotar el espacio de posibles soluciones; para después aprender la estructura del DAG que más se asemeja mediante métodos basados en puntuaciones. Por ejemplo, los trabajos de Abellán *et al.* [2] y Gámez *et al.* [76] utilizan *scores* Bayesianos para llevar a cabo los test estadísticos en un algoritmo tipo PC. Otro ejemplo pueden ser los trabajos de Cano *et al.* [37] y Tsamardinos *et al.*, [193]; cuyas versiones del algoritmo PC utilizan un procedimiento codicioso para definir las aristas, similar a K2 y para después realizar tests de independencia únicamente para las aristas creadas. Un potente método de este tipo es el *max-min hill climbing* [75, 158, 194].

2.3.2 Aprendizaje paramétrico de RBs

Cuando las variables del problema así como las relaciones entre ellas ya se han definido, ya sea a través de un experto o aprendiendo de los datos con algún método de los anteriormente comentados, es el momento de llevar a cabo el aprendizaje paramétrico, es decir, establecer las probabilidades *a priori* para los nodos raíz y las condicionales para el resto de nodos. Tal y como ocurre con el aprendizaje estructural, el paramétrico puede también realizarse por expertos, siendo esta tarea mucho más compleja que la anterior y resultando imposible para el caso de redes complejas.

El aprendizaje de los parámetros de la red se enfoca desde dos alternativas dependidendo de la completitud de los datos:

- **Datos completos.** En el caso de tener suficientes datos y tratarse de datos completos (*i.e.* sin datos perdidos), el método a seguir es bastante directo. Las tablas de probabilidades condicionadas tendrán los valores de los parámetros que maximizan la *función de verosimilitud*; es por eso que a este algoritmo se le conoce como *estimador de máxima verosimilitud* o por sus siglas MLE (en inglés *máximum likelihood estimate*). Supongamos, por ejemplo, que queremos calcular la CPT de la variable X, cuyos padres son Y y Z, entonces las probabilidades siguen la fórmula:

$$P(x_i|y_j, z_k) \sim n_{i,j,k}/n_{j,k}$$

donde $n_{i,j,k}$ es el número de veces en que $X_i = x_i, Y_j = y_j$ y $Z_k = z_k$; y $n_{j,k}$ es el número de casos para los que $Y_j = y_j$ y $Z_k = z_k$.

Cuando los parámetros se estiman a través de los datos, podemos encontrarnos casos en los que uno de los estados de alguna variable no se da para ninguna de las instancias; y como resultado, la probabilidad para este caso quedaría definida como cero. En muchas ocasiones esto sucede no porque sea un suceso realmente imposible, sino porque el número de datos es insuficiente para estimar correctamente los parámetros; por lo que es recomendable utilizar una técnica que contemple esta casuística. Esto se puede llevar a cabo aplicando alguna técnica de *suavizado*, que elimine probabilidades nulas. Aunque existen diferentes técnicas, la más común y conocida es la corrección de *Laplace*. En el suavizado de *Laplace* se comienza asumiendo que las probabilidades se distribuyen según una distribución uniforme; o lo que es lo mismo, si se tiene que la variable X_i toma k_i estados, entonces $P(x_i) = 1/k$ para todo $x_i \in X_i$. Después, se actualizan los valores de probabilidad a:

$$P(x_i) = \frac{1 + m}{k_i + M} \tag{2.2}$$

donde m es el número de veces que x_i ocurre y M el número total de instancias.

- **Datos incompletos.** Esta situación puede darse por dos razones:

  1. Presencia de datos perdidos, es decir, en algunas de las instancias no conocemos el estado que toma alguna (o algunas) de las variables. Existen distintas formas de paliar este problema: (1) eliminar las instancias con valores perdidos; (2) asignar un nuevo (el mismo) estado a todas esas instancias de la variable de las que no hay registro; (3) sustituir por la moda de esa variable en concreto; (4) estimar esos valores perdidos por algún método basado en los datos. Se debe escoger la alternativa que mejor resultado vaya a dar considerando las características de los datos. En el caso de que la cantidad de datos no sea grande, (1) y (2) estarían suponiendo una importante

pérdida de información. Y (3) es una aproximación bastante "bruta", ya que no tiene en cuenta los estados que las instancias toman para el resto de variables. En general, la mejor opción suele ser optar por completar los datos aprendiendo de los datos; para ello existen muchos métodos ya definidos [12, 90, 154].

2. Nodos ocultos, que se traduce en que una o más variables no contienen ningún dato. El procedimiento para tratar estos nodos ocultos se basa en el algoritmo *esperanza-maximización* (EM - por sus siglas en inglés *expectation-maximization*). El algoritmo comienza dando valores aleatorios a los datos perdidos, para a continuación repetir dos fases de manera iterativa: paso (E), en el que los valores perdidos se estiman basándose en los parámetros actuales; paso (M), donde los parámetros se actualizan basándose en los datos estimados. El procedimiento EM para calcular los parámetros de la red seguirá los siguientes pasos:

   a) Usando el estimador de máxima verosimilitud, se asignan los parámetros de las variables para las que ni ellas ni sus padres contienen valores perdidos.

   b) Se asignan valores aleatorios a los parámetros desconocidos.

   c) Usando inferencia y los parámetros aleatorios, se calculan los valores de los nodos ocultos basándose en las variables conocidas.

   d) Se actualizan los parámetros que acaban de estimarse.

   e) Se vuelven a estimar los parámetros de los nodos ocultos partiendo de los obtenidos hasta ahora.

   f) Se repite este procedimiento hasta que se produzca convergencia; es decir, hasta que no ocurran cambios significativos de una iteración a otra.

2.4 Inferencia sobre redes Bayesianas

En el estudio de RBs, muy a menudo, se plantearán cuestiones similares a estas que planteamos sobre la red *Asia*: sabiendo que un paciente tiene disnea, (1) ¿qué probabilidad hay de que se deba a una bronquitis?, o, (2) ¿cómo de posible es que sea causado por un cáncer de pulmón? Así, las anteriores cuestiones se corresponderían con calcular: (1) $P(B|D)$, (2) $P(L|D)$. Esto es lo que se conoce como hacer inferencia, y consiste entonces en propagar una información

conocida a través de la red y estudiar qué efectos produce sobre otras variables (o lo que es lo mismo, calcular probabilidades *a posteriori* de una o varias variables conocida la evidencia).

Los algoritmos de inferencia [111, 124, 153, 178] se utilizan para calcular probabilidades *a posteriori*. Estos algoritmos buscan valerse de las independencias entre las variables para obtener una factorización de la distribución conjunta de probabilidad. Asumiremos en esta memoria la aproximación de Shafer y Shenoy [180, 182] quienes hablan de los algoritmos desde un punto de vista de *valuaciones*. Por valuaciones entenderemos a cualquier representación de una información, que podrían ser distribuciones de probabilidad, convexos de probabilidad, etc. Podemos decir entonces que lo que Shafer y Shenoy llaman valuaciones es lo que en esta memoria nombramos como *potenciales*.

2.4.1 Potenciales probabilísticos y operaciones con ellos

Los potenciales probabilísticos son funciones reales ligadas a las distribuciones de probabilidad, pero que, al contrario que estas, no se encuentran necesariamente normalizadas, es decir, el conjunto de sus valores no necesitarán sumar 1. De manera rigurosa, definimos un potencial como:

Definición 2.4.1 ***(Potencial)*** *Un* ***potencial*** *es una función definida sobre un conjunto de variables finitas* $\boldsymbol{X}$ *de la forma:*

$$\phi : \Omega_{\boldsymbol{X}} \rightarrow \mathbb{R}_0^+$$

donde, cómo venimos considerando en esta memoria, $\Omega_{\boldsymbol{X}}$ *representa el dominio de la variable* $\boldsymbol{X}$*.*

Diremos que si $\mathbf{X}= \{X_1, X_2, ..., X_N\}$ es un conjunto de variables que toma valores en $\Omega_{\mathbf{X}} = \{\Omega_{X_1} \times \Omega_{X_2} \times ... \times \Omega_{X_N}\}$, entonces para cada $I \subseteq \{1, 2, ..., N\}$ existe un conjunto de potenciales ϕ_I definidos sobre Ω_{X_I}, donde X_I es un subconjunto de las variables $\{X_1, X_2, ..., X_N\}$.

Decimos que el *tamaño* del potencial es el número de parámetros necesarios para que quede totalmente definido. Este número corresponde con el producto del número de estados de cada variable. Es decir, en el caso en el que el potencial represente a las variables X_1, X_2, X_3 donde $|\Omega_{X_1}| = 2, |\Omega_{X_2}| = 3$ y $|\Omega_{X_3}| = 2$, necesitaremos $2 \cdot 3 \cdot 2 = 12$ valores (*i.e.* el tamaño del potencial es 12).

Clásicamente, los potenciales se representan mediante tablas de probabilidad o tablas de probabilidad condicionadas. Notaremos como $T^{\phi}(X_A)$ a la tabla de probabilidad que representa al potencial $\phi(X_A)$.

Ejemplo 15 *Consideremos un conjunto de variables aleatorias* $\boldsymbol{X} = \{X_1, X_2, X_3, X_4\}$*, tomando 2, 3, 2 y 2 estados respectivamente. Y supongamos un potencial definido sobre las tres primeras variables* $\{X_1, X_2, X_3\}$*,* $\phi_1(X_1, X_2, X_3)$*, que toma los valores que se muestran en la tabla de probabilidad de Fig. 2.5.*

$indice$	x_1	x_2	x_3	$\phi_1(x_1, x_2, x_3)$
0	0	0	0	0.1
1	0	0	1	0.9
2	0	1	0	0.5
3	0	1	1	0.5
4	0	2	0	0.0
5	0	2	1	1
6	1	0	0	0.8
7	1	0	1	0.2
8	1	1	0	0.2
9	1	1	1	0.8
10	1	2	0	0.9
11	1	2	1	0.1

Fig. 2.5: Representación del potencial $\phi_1(X_1, X_2, X_3)$ como función que asigna un número a cada posible configuración de valores de las variables $\{X_1, X_2, X_3\}$.

En la Fig. 2.5, la columna de la izquierda indica lo que denominaremos como el índice de la configuración y que definiremos a continuación; las tres columnas centrales presentan todas las posibles combinaciones de estados de las tres variables X_1, X_2, X_3*, y en la columna de la derecha obtenemos el valor del potencial asociado a cada una de esas configuraciones.*

□

Hemos visto en la tabla de Fig. 2.5 del Ejemplo 15 un nuevo concepto, el *índice de la configuración*, que aparece en la columna situada en la izquierda. Este no es más que un identificador numérico único que representa a cada una de las posibles configuraciones del dominio. Consideraremos que los índices empezarán en 0 y acabarán en $|\Omega_{\mathbf{X}} - 1|$. En este caso, el índice 0 se corresponde con la configuración $\{0, 0, 0\}$, el índice 1 corresponde con $\{0, 0, 1\}$, y así sucesivamente hasta la última de las configuraciones $\{1, 2, 1\}$ a la que se adjudica el índice 11.

Es posible establecer una conexión entre los índices y las configuraciones basada en el concepto de *peso*.

Definición 2.4.2 ***(Peso de variable)*** *Supuesto un conjunto de variables* $\boldsymbol{X} := \{X_1, X_2, \ldots, X_N\}$. *Cada variable* X_i *tiene un* ***peso*** w_i *que puede computarse de forma recursiva de la siguiente manera:*

$$w_i = \begin{cases} 1 & si \;\; i = N \\ |\Omega_{X_{i+1}}| \cdot w_{i+1} & en\ otro\ caso \end{cases} \tag{2.3}$$

Para el potencial considerado en Ejemplo 15, los valores de los pesos serán: $w_3 = 1, w_2 = 2, w_1 = 6$. Podemos visualizar en la Tabla 2.1 un resumen de las cardinalidades y pesos de las variables X_1, X_2, X_3 del Ejemplo 15.

Variable	X_1	X_2	X_3
Carnalidad	2	3	2
Peso	6	2	1

Tab. 2.1: Tabla resumen de la cardinalidad y peso de las variables del Ejemplo 15

Conocidos los pesos, podemos deducir el índice de una configuración $\mathbf{x} := \{x_1, x_2, \ldots, x_N\}$ aplicando la fórmula:

$$indice(\mathbf{x}) = \sum_{i=1}^{N} x_i \cdot w_i \tag{2.4}$$

Ejemplo 16 *Considerado el potencial* $\phi_1(X_1, X_2, X_3)$ *dado en Ejemplo 15, y los pesos de las variables* $w_1 = 6$, $w_2 = 2$, *y* $w_3 = 1$. *Podemos calcular los índices para las configuraciones utilizando la ecuación (2.4):*

$$
\begin{aligned}
indice(\{0,0,0\}) &= 0 \cdot 6 + 0 \cdot 2 + 0 \cdot 1 = 0 \\
indice(\{0,0,1\}) &= 0 \cdot 6 + 0 \cdot 2 + 1 \cdot 1 = 1 \\
indice(\{0,1,0\}) &= 0 \cdot 6 + 1 \cdot 2 + 0 \cdot 1 = 2 \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
indice(\{1,2,1\}) &= 1 \cdot 6 + 2 \cdot 2 + 1 \cdot 1 = 11
\end{aligned}
$$

□

Dado un índice k, su *configuración asociada* se denota por $\mathbf{x}^{(k)}$ y satisface $indice(\mathbf{x}^{(k)}) = k$. Si tenemos un índice k, el valor asignado a cada variable X_i puede computarse como:

$$x_i = (k//w_i) \,\%\, |\Omega_{\mathbf{X}_i}| \tag{2.5}$$

donde $//$ denota la división entera y $\%$ el módulo de la división. Esta operación será necesaria para permitir una conversión rápida entre configuraciones e índices.

Notamos que la asociación entre índices y configuraciones necesita que las variables se encuentren ordenadas. Cualquier orden es válido, pero en esta memoria consideraremos por defecto el orden en el que las variables se encuentren escritas en el potencial (es decir, para un potencial $\phi(X, Y, Z)$, la primera variable será X y la última Z). Además, asumiremos que la primera variable será la que tenga el mayor peso, aunque considerar el opuesto sería igualmente válido.

Shafer y Shenoy consideran en sus trabajos las dos operaciones básicas (marginalización y combinación) sobre potenciales e hicieron una gran aportación, definiendo unos axiomas que se debían cumplir para poder usar los algoritmos de propagación sobre estructuras gráficas. Veamos cómo se llevarían a cabo ambas operaciones sobre tablas de probabilidad.

Ejemplo 17 *Siguiendo con el potencial del anterior Ejemplo 15, podemos marginalizar sobre cualquiera de las variables X_1, X_2 o X_3, cabiendo también la posibilidad de hacerlo simultáneamente para un par de variables. Podemos ver en la Tabla 2.2 dos tablas, la de la izquierda representa el potencial ϕ_1 original y en la segunda se observa cómo sería en este caso la marginalización de ese mismo potencial ϕ_1 sobre la variable X_1.*

índice	x_1	x_2	x_3	$\phi_1(x_1, x_2, x_3)$	$\Longrightarrow$	índice	x_2	x_3	$\phi_1^{\downarrow X_1}(x_2, x_3)$
0	0	0	0	0.1		0	0	0	0.9
1	0	0	1	0.9		1	0	1	1.1
2	0	1	0	0.5		2	1	0	0.7
3	0	1	1	0.5		3	1	1	1.3
4	0	2	0	0.0		4	2	0	0.9
5	0	2	1	1		5	2	1	1.1
6	1	0	0	0.8					
7	1	0	1	0.2					
8	1	1	0	0.2					
9	1	1	1	0.8					
10	1	2	0	0.9					
11	1	2	1	0.1					

Tab. 2.2: Tabla de probabilidad correspondiente al potencial ϕ_1 de las variables $\{X_1, X_2, X_3\}$ definido en el Ejemplo 15, marginalizado sobre la variable X_1.

Los valores de probabilidad de la segunda tabla se han generado sumando los valores de la tabla inicial, para los que las variables X_2 y X_3 tienen los mismos valores de configuración. Esto quiere decir que el valor correspondiente al índice 0 de la tabla del potencial marginalizado (0.9) es suma de los valores de los índices 0 y 6 de la tabla inicial; el siguiente, índice 1, es suma de los índices 1 y 7; y así sucesivamente. Las flechas rojas indican qué dos elementos de la tabla original se combinan para formar el de la tabla marginalizada.

□

Ejemplo 18 *Consideremos el mismo conjunto de variables que definimos en Ejemplo 15, y definamos sobre las variables $\{X_1, X_4\}$, un nuevo potencial $\phi_2(X_1, X_4)$, que podemos visualizar como la tabla que se encuentra en el centro de la Tabla 2.3. Podemos combinar este potencial junto con el de las variables $\{X_1, X_2, X_3\}$, $\phi_1(X_1, X_2, X_3)$, obteniendo entonces el potencial correspondiente a las variables $\{X_1, X_2, X_3, X_4\}$. Esta combinación de potenciales puede observarse en la tabla de la derecha de la figura de la Tabla 2.3:*

índice	x_1	x_2	x_3	$\phi_1(x_1, x_2, x_3)$
0	0	0	0	0.1
1	0	0	1	0.9
2	0	1	0	0.5
3	0	1	1	0.5
4	0	2	0	0.0
5	0	2	1	1
6	1	0	0	0.8
7	1	0	1	0.2
8	1	1	0	0.2
9	1	1	1	0.8
10	1	2	0	0.9
11	1	2	1	0.1

$\times$

índice	x_1	x_4	$\phi_2(x_1, x_4)$
0	0	0	0.2
1	0	1	0.8
2	1	0	0.5
3	1	1	0.5

$=$

índice	x_1	x_2	x_3	x_4	$\phi_C(x_1, x_2, x_3, x4)$
0	0	0	0	0	0.02
1	0	0	0	1	0.08
2	0	0	1	0	0.18
3	0	0	0	1	0.72
4	0	1	0	0	0.01
5	0	1	0	1	0.04
6	0	1	1	0	0.01
7	0	1	0	1	0.04
8	0	2	0	0	0.0
9	0	2	0	1	0.0
10	0	2	1	0	0.2
11	0	2	0	1	0.8
12	1	0	0	0	0.4
13	1	0	0	1	0.4
14	1	0	1	0	0.1
15	1	0	0	1	0.1
16	1	1	0	0	0.1
17	1	1	0	1	0.1
18	1	1	1	0	0.4
19	1	1	0	1	0.4
20	1	2	0	0	0.45
21	1	2	0	1	0.45
22	1	2	1	0	0.05
23	1	2	0	1	0.05

Tab. 2.3: Tabla de probabilidad correspondiente a la combinación de los potenciales $\phi_1(X_1, X_2, X_3)$ y $\phi_2(X_1, X_4)$.

La tabla del potencial combinado $\phi_C(x_1, x_2, x_3, x_4)$ se genera de la siguiente manera: puesto que ambos potenciales ϕ_1 y ϕ_2 comparten la variable X_1, el conjunto de configuraciones de la nueva tabla se forma combinando las configuraciones de ambas tablas cuyo valor para la variable X_1 coincide; los valores de probabilidad se calcularán como la multiplicación de los valores de las configuraciones de las dos tablas. Así, la combinación de ambos índices 0 de las tablas de ϕ_1 y ϕ_2 genera el índice 0 de la tabla de la derecha; índice 0 e índice 1, relativos respectivamente a ϕ_1 y ϕ_2 combinan para dar lugar a índice 1 en ϕ_C; y así sucesivamente.

□

Además de las operaciones básicas de combinación y marginalización necesarias para realizar inferencia, procedemos a definir otras funciones igualmente importantes y que necesitaremos usar en el desarrollo de la memoria. Dado un potencial definido sobre un conjunto de variables, podemos restringirlo a una configuración quedándonos con los valores del potencial que sean consistentes con ella.

Ejemplo 19 *Consideremos una vez más el potencial ϕ_1 de las variables $\{X_1, X_2, X_3\}$ disponible en Ejemplo 15. Podríamos, por ejemplo, restringirlo a que la variable X_2 tome uno de sus estados. Veamos el potencial resultante al restringir $X_2 = 2$ en la tabla posicionada en la derecha de Tab. 2.4:*

índice	x_1	x_2	x_3	$\phi_1(x_1, x_2, x_3)$		
0	0	0	0	0.1		
1	0	0	1	0.9		
2	0	1	0	0.5		
3	0	1	1	0.5		
4	0	2	0	0.0 ←		
5	0	2	1	1 ←		
6	1	0	0	0.8		
7	1	0	1	0.2		
8	1	1	0	0.2		
9	1	1	1	0.8		
10	1	2	0	0.9 ←		
11	1	2	1	0.1 ←		

índice	x_1	x_3	$\phi_1^{R(X_2=2)}(x_1, x_2, x_3)$
0	0	0	0.0
1	0	1	1
2	1	0	0.9
3	1	1	0.1

Tab. 2.4: Tabla de probabilidad correspondiente al potencial que se genera al restringir el potencial $\phi_1(X_1, X_2, X_3)$ del Ejemplo 15 a que $X_2 = 2$.

En la tabla del potencial original ϕ, presentado en la parte derecha de la Figura 2.4, las flechas rojas señalizan las configuraciones que formarán parte del potencial restringido a $X_2 = 2$.

□

Todo potencial puede ser convertido en una distribución de probabilidad operando para que sus valores sumen 1, es el proceso que se conoce como *normalización*. Simplemente, se sumarán todos los valores del potencial y se dividirá cada uno de los valores por esa suma.

Ejemplo 20 *Para el caso del potencial ϕ_1 definido sobre las variables X_1, X_2, X_3 que podemos observar en la tabla de la derecha de Fig. 2.5, normalizaríamos siguiendo el procedimiento:*

$$\begin{aligned} sum(\phi_1(X_1, X_2, X_3)) &= 0.1 + 0.9 + 0.5 + 0.5 + 0.0 + 1 + 0.8 + 0.2 + 0.2 + 0.8 + 0.9 + 0.1 \\ &= 6 \end{aligned}$$

Dividiendo entonces cada valor por $sum(\phi_1(X_1, X_2, X_3))$, tenemos la distribución conjunta de probabilidad para las variables $\{X_1, X_2, X_3\}$, que podemos observar en la tabla de la derecha de la Tabla 2.5:

índice	x_1	x_2	x_3	$\phi_1(x_1, x_2, x_3)$	$\Longrightarrow$	índice	x_1	x_2	x_3	$\phi_N(x_1, x_2, x_3)$
0	0	0	0	0.1		0	0	0	0	0.1 / 6 = $0.01\widehat{6}$
1	0	0	1	0.9		1	0	0	1	0.9 / 6 = 0.15
2	0	1	0	0.5		2	0	1	0	0.5 / 6 = $0.08\widehat{3}$
3	0	1	1	0.5		3	0	1	1	0.5 / 6 = $0.08\widehat{3}$
4	0	2	0	0.0		4	0	2	0	0.0 / 6 = 0.0
5	0	2	1	1		5	0	2	1	1 / 6 = $0.1\widehat{6}$
6	1	0	0	0.8		6	1	0	0	0.8 / 6 = $0.1\widehat{3}$
7	1	0	1	0.2		7	1	0	1	0.2 / 6 = $0.0\widehat{3}$
8	1	1	0	0.2		8	1	1	0	0.2 / 6 = $0.0\widehat{3}$
9	1	1	1	0.8		9	1	1	1	0.8 / 6 = $0.1\widehat{3}$
10	1	2	0	0.9		10	1	2	0	0.9 / 6 = 0.15
11	1	2	1	0.1		11	1	2	1	0.1 / 6 = $0.01\widehat{6}$

Tab. 2.5: Tabla de probabilidad correspondiente a la normalización del potencial $\phi(X_1, X_2, X_3, X_4)$ del Ejemplo 15.

Donde podemos fácilmente comprobar que el total de los valores del nuevo potencial normalizado suman 1.

$$
\begin{aligned}
sum(\phi_N(X_1, X_2, X_3)) &= 0.01\widehat{6} + 0.15 + 0.08\widehat{3} + 0.08\widehat{3} + 0.0 + 0.1\widehat{6} + 0.1\widehat{3} + 0.0\widehat{3} + 0.0\widehat{3} + \\
&\quad + 0.1\widehat{3} + 0.15 + 0.01\widehat{6} \\
&= 1
\end{aligned}
$$

Lo cierto es que en el trabajo con modelos gráficos probabilísticos, el potencial que represente una tabla de probabilidad condicionada de una variable dados los padres necesitará que los valores sumen uno para cada configuración de los padres, como ocurre en la tabla correspondiente al potencial $\phi_1(X_1, X_2, X_3)$, que ya comentamos que representa la distribución $P(X_3|X_1, X_2)$. Podemos observar este hecho en la Tabla 2.6, donde se agrupan los valores que deben sumar 1:

índice	x_1	x_2	x_3	$\phi_1(x_1, x_2, x_3)$
0	0	0	0	0.1
1	0	0	1	0.9
2	0	1	0	0.5
3	0	1	1	0.5
4	0	2	0	0.0
5	0	2	1	1
6	1	0	0	0.8
7	1	0	1	0.2
8	1	1	0	0.2
9	1	1	1	0.8
10	1	2	0	0.9
11	1	2	1	0.1

Tab. 2.6: Tabla de probabilidad condicionada de la variable X_3 dados su padres $\{X_1, X_2\}$ del Ejemplo 15, donde podemos observar como los valores para cada configuración de los padres $\{X_1, X_2\}$ suman 1.

□

Ya comentamos que para definir por completo un potencial que representa las variables $\{X_1, X_2, ..., X_n\}$ se necesitan $|\Omega_{X_1}| \cdot |\Omega_{X_2}| \cdot ... \cdot |\Omega_{X_n}|$ valores. Esto implicará que la representación de potenciales de gran tamaño se complique, ya que a medida que el tamaño crece, la complejidad aumenta exponencialmente. Esto puede facilitarse reduciendo el tamaño de los potenciales. Una forma

de realizar esta reducción se consigue mediante la *factorización*, permitiendo definir grandes potenciales mediante factorización multiplicativa de potenciales de tamaño menor. En caso de que el potencial pueda factorizarse en dos más pequeños, obtendremos que el tamaño será la suma de los tamaños de los potenciales.

Definición 2.4.3 ***(Factorización de potenciales)*** *Un potencial $\phi(\boldsymbol{X})$ se dice* ***factorizable*** *en los factores $\phi_1(X_A)$ y $\phi_2(X_B)$ si*

$$\forall x \in \boldsymbol{X} \Rightarrow \phi(\boldsymbol{X}) = \phi_1(X_A) \cdot \phi_2(X_B)$$

donde $\boldsymbol{X} = X_A \cup X_B$.

En el caso particular, en el que $X_A \cap X_B = \emptyset$, decimos que $\phi(\boldsymbol{X})$ se descompone en los factores $\phi_1(X_A)$ y $\phi_2(X_B)$.

2.4.2 Problemas de inferencia y algoritmos

El problema de inferencia puede presentarse en tres variantes:

- **Actualización de creencia**: se trata de la actualización de la información sobre una determinada variable a la luz de alguna información nueva (evidencia: información no disponible al modelizar el problema). Llevado a probabilidades, esto se traduce en calcular la probabilidad *a posteriori* de una variable X (o varias variables), cuando un subconjunto de variables X_E toma valores conocidos, es decir, $P(X|X_E)$. En el ejemplo de la red *Asia*, podríamos estar interesados en calcular cuál es la probabilidad de que un paciente sea fumador a la vista de que padece disnea. En definitiva, se trata de ver la forma en que tener información sobre la ocurrencia de disnea modifica mi creencia en la condición de fumador de un paciente.

- **Búsqueda de máxima distribución *a posteriori*** (MAP - en inglés *maximum a posteriori probability*) dada una cierta evidencia. Se trata de encontrar la configuración de valores de las variables no observadas (no incluidas en la evidencia) que maximizan la probabilidad de ocurrencia. En nuestro ejemplo, imaginemos que se observa en un cierto paciente que ha visitado Asia y padece disnea. El objetivo es ahora buscar qué valor han de tomar el resto de variables para maximizar la ocurrencia de dicho evento.

- **Búsqueda de la explicación más probable** (MPE - en inglés *most probable explanation*). Caso particular de MAP en el que se buscan los valores que deben tomar algunas variables

no evidenciales (en lugar de todas las variables que no forman parte de la evidencia, como en MAP) que maximizan la probabilidad de ocurrencia del evento correspondiente.

Cuando la información disponible sobre una variable se actualiza, esta pasa a los nodos vecinos y actualiza su información, y a su vez estos pasan la información nueva y anterior a sus vecinos no modificados para ser también actualizados. Tratar de estudiar este paso de mensajes a "fuerza bruta", consistiría en obtener la distribución conjunta de probabilidad para posteriormente marginalizar sobre la variable o variables en cuestión. Este es un problema realmente arduo y que gana complejidad exponencialmente, a medida que crece el número de variables. Es por esta razón que el estudio de nuevos métodos de inferencia computacionalmente más eficientes se encuentra en continuo desarrollo y ha dado lugar a grandes avances en los últimos años. Conceptualmente, podemos dividir los algoritmos en: (1) exactos [52, 53, 59]; y aproximados[38, 42, 57, 142], que dan una respuesta muy similar a la exacta a excepción de un error aceptado, y la dan idealmente en un menor tiempo y utilizando menos recursos computacionales.

Algoritmos exactos

Los algoritmos de inferencia exactos son aquellos que tras aplicarse dan el valor exacto de probabilidad, es decir, en las probabilidades calculadas no existe error. De manera general, podemos clasificar los algoritmos en [96, 187]:

- **Algoritmo de Pearl** - También conocido como *propagación de creencias* [152], este algoritmo supone la base para otros muchos algoritmos de inferencia definidos con posterioridad. Presenta la desventaja de poder aplicarse únicamente para el caso de grafos individualmente conexos [1] (árboles y poliárboles [2]). La idea principal de este algoritmo es que parte de una evidencia dada, para propagarla sobre el resto de nodos de la red y ver cómo esta afecta al resto de variables.

- **Eliminación de variables** [179, 182, 210, 211] - El algoritmo parte de la distribución conjunta de probabilidad y va sacando provecho de las independencias entre variables, para realizar operaciones e ir eliminando variables, de manera que la probabilidad que se busca pueda seguir siendo calculada; y además se calcule de un modo mucho más simple y eficiente. Los diferentes métodos basados en eliminación de variables se diferencian por cómo definen la secuencia de eliminación [127].

[1] Grafos en los que cada par de nodos se encuentra conectado, como máximo, por un camino simple.
[2] Grafo individualmente conexo para el que algunos nodos presentan más de un padre.

- **Métodos basados en condicionamiento** [153] - Transforman el grafo en un poliárbol para posteriormente poder aplicar el algoritmo de Pearl. Esto se consigue buscando las variables que d-separan las dependencias que no permiten aplicar propagación de creencias, para entonces inicializarlas.
- **Métodos basados en árboles de cliques** [102, 104, 124] - Generan un nuevo grafo dirigido acíclico donde las variables se agrupan convenientemente formando "supernodos", de manera que pueda aplicarse sobre esta nueva red en el algoritmo de Pearl.

Algoritmos aproximados

Como comentamos, resolver problemas de inferencia exacta se complica exponencialmente a medida que el tamaño y número de variables crece; de hecho, en general, se trata de un problema *NP-completo* [48]. Por esta razón, surge la necesidad de crear algoritmos aproximados que den una respuesta casi tan buena como la que darían algoritmos exactos, pero obtenida en un menor tiempo y utilizando menores recursos computacionales. En contraposición, la respuesta aproximada implicará un error aceptado en el cálculo de probabilidades. En los casos en los que la red sea demasiado compleja, la inferencia aproximada será la única opción disponible, ya que al usar métodos exactos se excederán los recursos computacionales. El problema de inferencia aproximada cuando se requiere una precisión determinada también es un problema *NP-completo* [58], pero permiten poder ser utilizados sobre un mayor número de redes, incluyendo algunas sobre los que la inferencia exacta no es posible.

Existen muchos algoritmos cuyo objetivo es el de realizar inferencia de un modo aproximado [23, 24, 38, 42, 47, 83, 105, 100, 130, 151, 171, 198]. Un grupo de estos algoritmos son los conocidos como determinísticos, que buscan simplificar el problema de alguna forma para después aplicar algoritmos exactos. Un tipo de simplificación sería el de recurrir a estructuras alternativas [24, 38, 42, 83, 171] que representen un conocimiento muy similar al original, asumiendo una pequeña pérdida de información. La operación de poda de árboles pertenecería a este conjunto. En ese proceso se parte de un árbol de probabilidad para combinar varios valores (o conjuntos de valores) de probabilidad en uno solo, produciendo un árbol de menor tamaño. Otro ejemplo sería el que presentamos en esta memoria (Capítulo 4), unas estructuras alternativas [24, 83] para representación de potenciales que utilizan la repetición de valores de probabilidad para ahorrar espacio; además, presentamos también un método para aproximar dichas estructuras, convirtiéndolas en una opción incluso más compacta. Existen otros tipos de simplificación, como eliminación de arcos [23, 47, 198] que buscan la manera de quitar de la RB arcos que de algún modo son "poco relevantes" o "débiles" atendiendo a la particular consulta que se quiere resolver;

y aproximación de tablas de probabilidad condicionada [105, 130]. Por otro lado, se encuentran los que se conocen como métodos Monte Carlo y se basan en simulación, como los basados en el muestreo de Gibbs [100, 151] o en muestreo por importancia [57, 95, 141, 170, 179]. Para el desarrollo de nuestra memoria, será importante que conozcamos en qué consiste la inferencia variacional [21, 108, 202], hablaremos sobre ello en el Capítulo 5.

3 Modelos gráficos con probabilidades imprecisas

En el anterior Capítulo 2 introdujimos los modelos gráficos probabilísticos y más en detalle las redes Bayesianas. En general, estos modelos se expresan en términos de probabilidades precisas, tal y como vimos en el anterior capítulo. En este capítulo vamos a estudiar una generalización a los MGPs, los modelos gráficos probabilísticos con probabilidades imprecisas. Primeramente, en la Sección 3.1 podremos ver una introducción a las probabilidades imprecisas, poniendo hincapié en una de las aproximaciones disponibles: los conjuntos convexos de probabilidad. En Sec. 3.2 trataremos con conjuntos credales y la generalización de redes Bayesianas con probabilidades imprecisas, llamadas redes credales. En esta sección nos detendremos para comentar cómo se lleva a cabo la inferencia con este tipo de modelos.

3.1 Teoría de probabilidades imprecisas

Como hemos comentado, clásicamente, la probabilidad se ha tratado desde un punto de vista preciso, en el que para cada evento S se asigna un valor único de probabilidad $P(S)$, que normalmente satisface los axiomas de Kolmogorov (1.2.2). Podríamos encontrarnos, por contra, con problemas para los que la cantidad de información es limitada, poco específica o contradictoria. Supongamos que se nos presenta una urna y se nos asegura que en su interior hay un total de 10 bolas de dos colores, azul y rojo, pero se desconoce la proporción de cada color. Asignar valores de probabilidad a los distintos sucesos de este problema sería complicado y se incurriría muy probablemente en errores. Podríamos también pensar en casi cualquier problema médico [18, 94, 160], donde es muy común expresar la probabilidad de padecer una determinada enfermedad conocidos los síntomas mediante un intervalo de probabilidad, digamos entre el 2% y el 4%, en lugar de que sea el 2.67%. Estos ejemplos y muchos otros son la prueba de que en ocasiones resulta más sencillo y natural recurrir a una teoría de probabilidad que contemple valores no precisos.

La teoría de probabilidades imprecisas [139] es entonces una generalización de la teoría de probabilidad 'clásica' (precisa) y una muy útil herramienta para resolver, entre otros, el tipo de problemas que acaban de comentarse. Hay varias razones por las que considerar probabilidades no precisas es buena idea [43, 176]:

- Permiten modelizar problemas para los que no existe la suficiente información o esta no es lo suficientemente específica [64, 113].

- Son capaces de representar un problema en el que varios expertos definan simultáneamente los parámetros del modelo y se produzca discordancia entre sus opiniones [125]. Un modelo clásico preciso no podrá englobar estas opiniones y se deberá optar por algún tipo de restricción, como considerar sólo uno de ellos, o quedarse con el punto medio del intervalo de probabilidades, con la pérdida de información que ello genera.

- La teoría de probabilidades imprecisa tiene una sólida base axiomática. Así queda presente, por ejemplo, en el trabajo de Girón *et al.* [81] con su teoría de modelos Quasi-Bayesianos o en el de Walley [205] con la teoría de las previsiones inferiores coherentes.

- Posibilitan representar desconocimiento sobre la distribución *a priori* en problemas de robustez estadística [17, 66].

Las probabilidades imprecisas no son un concepto novedoso, y de hecho llevan siendo estudiadas más de 100 años (ver [88] para una revisión histórica). Aunque los conceptos eran por entonces algo difusos, de hecho, existían términos distintos que se utilizaban para denominar el mismo problema o se notaba con el mismo nombre a problemas diferentes.

En la literatura encontramos distintas aproximaciones para tratar con valores imprecisos; en el trabajo de Cano *et al.* [40] podemos encontrar una recopilación de ellas. Algunas de estas aproximaciones son [205]: capacidades de orden-2 [46, 99], intervalos de probabilidad [60, 208], medidas difusas [84, 206]... El caso de intervalos de probabilidad es realmente intuitivo y ha sido ampliamente utilizado en la literatura. Aunque en esta memoria nos centramos en la representación desde el punto de vista de conjuntos convexos de probabilidades [33, 34, 39, 61, 65, 126, 199], por resultar ser una opción más genérica y por ser posible realizar operaciones básicas de combinación y restricción con ellos, también hablaremos de intervalos de probabilidad.

3.1.1 Conjuntos convexos de probabilidad: notación y definiciones básicas

Tal y como venimos considerando, $\mathbf{X} = \{X_1, X_2, ..., X_N\}$ será el conjunto de todas variables aleatorias, donde cada X toma valores en Ω_X. Recordamos, que una distribución de probabilidad es una función del tipo $p : \Omega_X \rightarrow [0, 1]$ que verifica $\sum_{u \in \Omega_X} p(u) = 1$.

Definición 3.1.1 *(**Conjunto convexo de probabilidad**) Dado un **conjunto de distribuciones de probabilidad** H, diremos que este es **convexo** si $\forall p_1, p_2 \in H$ y $\forall \lambda \in [0, 1]$ se verifica:*

$$\lambda p_1 + (1 - \lambda)p_2 \in H$$

Los conjuntos convexos de probabilidad pueden ser representados alternativamente como: (1) **restricciones lineales** [199, 205] (siendo los intervalos de probabilidad [27, 72, 118] un caso particular de esta representación); y (2) por medio de sus **puntos extremos** (*i.e.* **vértices**).

La teoría de intervalos de probabilidad [27, 72, 118], se basa en un sistema de apuestas en el que se define la cantidad máxima que estás dispuesto a perder en caso de que el evento no se cumpla (definiendo la probabilidad inferior, $\underline{P}$) y la ganancia mínima a cobrar en caso de que el evento ocurra (definiendo la probabilidad superior, $\overline{P}$). De este modo, para cada evento queda definido un intervalo de probabilidad $[\underline{P}, \overline{P}]$. Para una variable X queda entonces definido un par de funciones:

$$\underline{P}, \overline{P} : 2^{\Omega} \rightarrow [0, 1]$$

donde $\underline{P}(S) \leq \overline{P}(S),\ \forall S \subseteq \Omega$.

Ejemplo 21 *Supongamos un partido de fútbol entre nuestro equipo y un equipo visitante. Los posibles resultados son: ganamos (G), perdemos (P) o empatamos (E). Supongamos que inicialmente disponemos de los siguientes intervalos:*

$\underline{P}(\emptyset) = 0$	$\overline{P}(\emptyset) = 0$	$\underline{P}(\{G\}) = 0.3$	$\overline{P}(\{G\}) = 0.7$
$\underline{P}(\{P\}) = 0.1$	$\overline{P}(\{P\}) = 0.8$	$\underline{P}(\{E\}) = 0.1$	$\overline{P}(\{E\}) = 0.7$
$\underline{P}(\{G, P\}) = 0.3$	$\overline{P}(\{G, P\}) = 0.9$	$\underline{P}(\{G, E\}) = 0.5$	$\overline{P}(\{G, E\}) = 1$
$\underline{P}(\{P, E\}) = 0.1$	$\overline{P}(\{P, E\}) = 0.8$	$\underline{P}(\{G, P, E\}) = 0.1$	$\overline{P}(\{G, P, E\}) = 1$

Se observa que ambas probabilidades superior e inferior se definen para algunos de los subconjuntos del espacio, dando como resultado un intervalo de probabilidad por cada subconjunto.

□

Un convexo puede ser también expresado por medio de un número finito de puntos extremos. A partir de este momento, dado un conjunto convexo de distribuciones de probabilidad H, notaremos como $Ext(H)$ al conjunto de puntos extremos que lo representa. Partiendo de los intervalos de probabilidad, podemos deducir las distribuciones extremas. En el trabajo de Verdegay [199] se realizó una revisión de los distintos algoritmos que permiten realizar esta tarea.

Ejemplo 22 *De los intervalos de probabilidades del anterior Ejemplo 21 pueden deducirse las siguientes distribuciones de probabilidad que forman un conjunto convexo mediante puntos extremos. En el trabajo de De Campos et al. [60] puede hallarse más información sobre un procedimiento eficiente que permite calcular estas distribuciones.*

$$\begin{array}{lll} p_1(G) = 0.7, & p_1(P) = 0.2, & p_1(E) = 0.1 \\ p_2(G) = 0.7, & p_2(P) = 0.1, & p_2(E) = 0.2 \\ p_3(G) = 0.4, & p_3(P) = 0.5, & p_3(E) = 0.1 \\ p_4(G) = 0.3, & p_4(P) = 0.5, & p_4(E) = 0.2 \\ p_5(G) = 0.3, & p_5(P) = 0.1, & p_5(E) = 0.6 \end{array}$$

Comprobamos que las 5 funciones p_i tienen tres componentes, una por cada suceso elemental del espacio, y que los valores de ellas suman 1; como debe ser por tratarse de distribuciones de probabilidad.

Otra forma de expresar esta misma información sería en forma de tabla:

	G	P	E
p_1	0.7	0.2	0.1
p_2	0.7	0.1	0.2
p_3	0.4	0.5	0.1
p_4	0.3	0.5	0.2
p_5	0.3	0.1	0.6

□

De un modo gráfico e ilustrativo, cualquier conjunto convexo de probabilidad puede representarse dentro un equilátero de lado 1; donde las distribuciones de probabilidad de un convexo corresponderían con las esquinas de un conjunto convexo. De este modo, cada lado se corresponde con un elemento de Ω y la distribución de probabilidad da información sobre la distancia al lado en cuestión. Esto quiere decir, que en el ejemplo del partido de fútbol que venimos considerando,

puesto que $p_1(G) = 0.7,\ \ p_1(P) = 0.2,\ \ p_1(E) = 0.1$, tendremos que el punto p_1 se sitúa a una distancia de 0.7 de G, a 0.2 de P y a 0.1 E. Así, podemos observar el convexo de probabilidad como en la Figura 3.1 que presentamos a continuación.

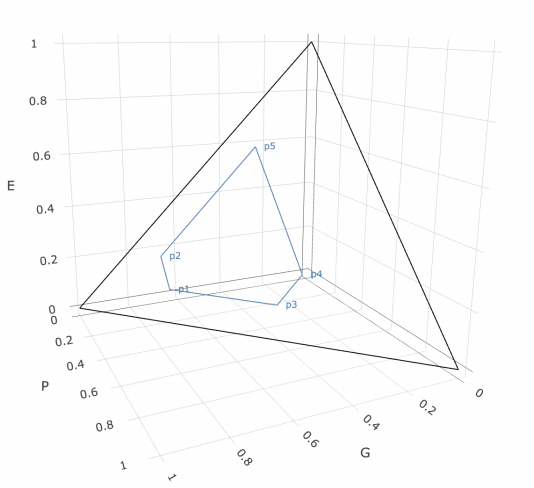


Fig. 3.1: Representación del convexo de probabilidad dentro de un rectángulo equilátero de lado 1 referente al ejemplo del partido de fútbol.

Operaciones con convexos

En general, podemos decir que toda operación probabilística podría extenderse al cálculo con convexos simplemente repitiendo la operación sobre las distintas probabilidades, pero esta suposición es bastante simplista. Veamos la definición de las operaciones básicas:

- **Marginalización**: Sea H un conjunto convexo de distribuciones de probabilidad definidas sobre un conjunto de variables aleatorias X_I, con dominio en Ω_I, y cuyos puntos extremos son $Ext(H) = \{h_1, h_2, ..., h_k\}$. Sea $J \leq I$, entonces la marginalización de H a las variables X_J se obtiene marginalizando cada una de las funciones de H a las variables X_J; lo que se corresponde con el convexo dado por:

$$H^{\downarrow J} = CH\{h_1^{\downarrow J}, h_2^{\downarrow J}, ..., h_k^{\downarrow J}\} \tag{3.1}$$

  donde CH representa el operador de cláusula convexa [1].

[1] En geometría, entendemos que la cláusula convexa (*i.e.* envolvente convexa o envoltura convexa) es el menor convexo que contiene el objeto, en nuestro caso, el conjunto de distribuciones de probabilidad.

Además, se cumple que $Ext(H^{\downarrow J}) \subseteq \{h_1^{\downarrow J}, h_2^{\downarrow J}, ..., h_k^{\downarrow J}\}$, y en general el recíproco no es cierto.

- **Intersección**: Dados dos conjuntos convexos H y H' definidos en Ω_{X_J} y Ω_{X_I} respectivamente. Decimos entonces que la intersección de ambos convexos sigue la forma:

$$H \cap H' = \{p \in \mathcal{P}_{I \cup J} : p^{\downarrow I} \in H, p^{\downarrow J} \in H'\} \tag{3.2}$$

  Puesto que H y H' son cerrados y convexos definidos por un número finito de puntos extremos, su intersección también será un cerrado y convexo definido por un número finito de puntos extremos.

- **Combinación**: Dados dos conjuntos convexos H y H' definidos en Ω_{X_J} y Ω_{X_I} respectivamente; y cuyos conjuntos de puntos extremos son $Ext(H) = \{h_1, h_2, ..., h_k\}$ y $Ext(H') = \{h'_1, h'_2, ..., h'_l\}$. Definimos entonces la combinación de ambos conjuntos convexos como:

$$H \otimes H' = CH\{h_1 \cdot h'_1, ..., h_1 \cdot h'_l, ..., h_k \cdot h'_1, ..., h_k \cdot h'_l\} \tag{3.3}$$

  donde es necesario recurrir a la operación de cláusula convexa para asegurar que el resultado es un convexo.

  De igual modo que en el caso de la marginalización, $Ext(H \otimes H') \subseteq \{h1 \cdot h'_1, ..., h1 \cdot h'_l, ..., h_k \cdot h'_1, ..., h_k \cdot h'_l\}$.

Información marginal y condicional

Una parte de nuestro estudio se centra en la inferencia de redes Bayesianas con probabilidades imprecisas. Y como pudimos estudiar en la Sección 2.2, las redes Bayesianas se valen de distribuciones marginales y condicionales para expresar las distribuciones de probabilidad de las variables aleatorias. En la Sección 1.2 definimos estos conceptos tan indispensables para el caso de probabilidades precisas, pasemos ahora a generalizar para el caso de probabilidades imprecisas.

Definición 3.1.2 ***(Información condicional para conjuntos convexos)*** *Dadas las variables aleatorias X e Y, la información condicional de Y dado X será un conjunto convexo, $H^{Y|X}$, de funciones $h : \Omega_X \times \Omega_Y \to [0,1]$ que verifican:*

$$\sum_{\omega_Y \in \Omega_Y} h(\omega_X, \omega_Y) = 1, \ \ \forall \omega_X \in \Omega_X \tag{3.4}$$

Pese a que podría resultar lógico que la información condicional de Y dado X no fuese más que la definición de un conjunto convexo de probabilidades para cada posible valor de X, la definición anterior es más general. En Cano *et al.* [146] puede observarse un ejemplo que corrobora esta afirmación.

Recordamos, que en probabilidades precisas, si disponemos de la distribución marginal de una variable X y de la distribución condicionada de Y a X, podemos calcular la distribución conjunta de probabilidad de ambas variables mediante el Teorema de Bayes. En el caso de probabilidades imprecisas, podemos calcular la información conjunta para dos variables si disponemos de las informaciones marginal de una y condicionada de la otra respecto de esta, y se hará del siguiente modo:

Proposición 3.1.1 *Sea una información marginal, H^X, sobre la variable X cuyos puntos extremos son $\{p_1, p_2, ..., p_k\}$ y otra condicionada, $H^{Y|X}$, cuyos puntos extremos son $\{h_1, h_2, ..., h_l\}$. Podemos entonces definir una información global sobre el par de variables (X, Y), como el convexo, $H^{X,Y}$, de distribuciones de probabilidad p sobre Ω_X, Ω_Y generado por los puntos:*

$$H^{X,Y} = CH\{p_1 \cdot h_1, ..., p_1 \cdot h_l, ..., p_2 \cdot h_1, ..., p_2 \cdot h_l, ..., p_k \cdot h_1, ..., p_k \cdot h_l\} \tag{3.5}$$

Sobre la anterior proposición podemos comentar:

- De manera general, sólo algunos de los puntos de la forma $p_i \cdot h_j$ resultarán ser puntos extremos del convexo $H^{Y,X}$. Existen varios algoritmos, como los que se estudian en Preparata *et al.* [161] y en Edelsbrunner [68] que permiten calcular la representación mínima del conjunto de puntos extremos, $Ext(H^{Y,X})$.
- $H^{Y,X} \neq \{p \cdot h : p \in H^X, h \in H^{Y|X}\}$. De hecho, este segundo conjunto es en general no convexo. Pero $H^{Y,X}$ sí se corresponde con la cláusula convexa de $\{p \cdot h : p \in H^X, h \in H^{Y|X}\}$.

Sea de un conjunto convexo de distribuciones de probabilidad sobre $\Omega_X \times \Omega_Y$, $H^{Y,X}$, donde las variables X e Y, toman respectivamente valores sobre los conjuntos Ω_X y Ω_Y y sea

$Ext(H^{Y,X}) = \{h_1, ..., h_n\}$. Podemos marginalizar este conjunto convexo por medio de la Ecuación (3.1) para obtener así cualquiera de los convexos H^X o $H^{Y|X}$. Así, $H^X = H^{\downarrow \Omega_X}$, y será generado por los puntos $p_i : \Omega_X \rightarrow [0, 1]$, con $i = 1, ..., n$ y donde:

$$p_i(y) = \sum_{y \in \Omega_Y} h_i(y, x)$$

De igual manera, el convexo marginalizado H^X será:

$$H^X = CH\{h_1^{\downarrow \Omega_X}, ..., h_n^{\downarrow \Omega_X}\} \tag{3.6}$$

De un modo similar podríamos marginalizar el convexo H para obtener $H^{Y|X}$. Por el contrario, teniendo los conjuntos convexos de probabilidad H^X y $H^{Y|X}$ no podremos, en general, recuperar H como combinación de ellos. Sea $H' = H^Y \otimes H^{Y|X}$, entonces $H \subseteq H'$.

Independencia con conjuntos convexos de probabilidad

El concepto de independencia es clave para el desarrollo de algoritmos de inferencia, ya que permite descomponer un problema complejo en varios subproblemas más sencillos, de manera que estos nos permitan construir más fácilmente el modelo global. Ya estudiamos varios tipos de independencia para el caso de probabilidad clásica (precisa) en la Sección 1.2.2, pero para conjuntos convexos de probabilidad este concepto no está nada establecido. En los trabajos de De Campos *et al.* [61] y Walley [205] se pueden encontrar varias definiciones distintas y no equivalentes para la independencia en conjuntos convexos de probabilidad. El trabajo de Couso *et al.* [51] muestra y ejemplifica la necesidad de uso de diferentes definiciones de independencia por poder ser aplicadas a distintos tipos de problemas.

Definición 3.1.3 ***(Independencia marginal condicionada de convexos)*** *Dado un conjunto convexo de probabilidad definido sobre tres variables aleatorias $\{X, Y, Z\}$ y sabiendo que estas toman valores respectivamente sobre los dominios $\{\Omega_X, \Omega_Y, \Omega_Z\}$. Decimos que hay* ***independencia marginal condicionada*** *de X e Y dada Z si se verifica:*

$$H = H^{X,Z} \cap H^{Y,Z} = \{p : p^{\downarrow \Omega_X, \Omega_Z} \in H^{X,Z}, p^{\downarrow \Omega_Y, \Omega_Z} \in H^{Y,Z}\} \tag{3.7}$$

donde $H^{X,Z}$ y $H^{Y,Z}$ son respectivamente los conjuntos marginales de H en $\Omega_X \times \Omega_Z$ y $\Omega_Y \times \Omega_Z$.

El concepto de independencia marginal se conoce también como **interacción desconocida** y se trata de un tipo de independencia bastante débil, ya que sólo implica que conocido el valor de probabilidad de Z los valores de probabilidad de X e Y no se influyen mutuamente.

A continuación vamos a presentar el concepto de *independencia fuerte* [140]. Este tipo de independencia es una generalización de la independencia probabilística y será el más apropiado para utilizar en los algoritmos de propagación, ya que descompone el convexo.

Definición 3.1.4 ***(Independencia fuerte)*** *Dado un convexo de probabilidades H sobre las variables $\{X, Y\}$ y que estas variables aleatorias toman valores respectivamente sobre los dominios Ω_X y Ω_Y. Decimos que X e Y son* ***fuertemente independientes*** *si existen dos convexos H^X H^Y que verifican:*

$$H = H^X \otimes H^Y = CH\{p_1 \cdot p_2 : p_1 \in H^X, p_2 \in H^Y\} \tag{3.8}$$

donde H^X e H^Y surgen de marginalizar el convexo H a Ω_X y Ω_Y, respectivamente.

Definición 3.1.5 ***(Independencia condicional fuerte)*** *Sea un convexo de probabilidad H definido sobre tres variables aleatorias $\{X, Y, Z\}$ y que estas toman valores respectivamente sobre los dominios Ω_X, Ω_Y y Ω_Z. Decimos que hay* ***independencia condicional fuerte*** *de X e Y dada Z si se verifica:*

$$H = H_1 \otimes H_2 \tag{3.9}$$

donde H_1 y H_2 son conjuntos convexos definidos respectivamente sobre $\Omega_X \times \Omega_Z$ y $\Omega_Y \times \Omega_Z$. Notamos que H_1 y H_2 no tienen que ser marginalizaciones de H.

3.2 Conjuntos credales y redes credales

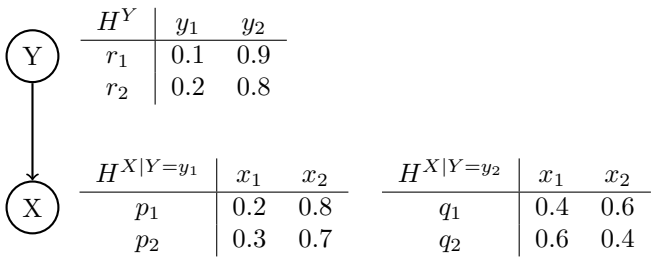
Definición 3.2.1 ***(Conjunto credal)*** *Un* ***conjunto credal*** *para una variable X es un conjunto convexo y cerrado (i.e. que contiene a su frontera) de distribuciones de probabilidad, H^X.*

Como venimos asumiendo, todo convexo será finito, en este caso generado por un número finito de puntos extremos.

Una *red credal* [54] es un modelo gráfico probabilístico (en concreto una generalización de red Bayesiana) que trata con probabilidades imprecisas. A cada variable se le asociará un conjunto credal condicional (en lugar de una única distribución de probabilidad, como ocurría en el caso preciso), $H^{X_i|pa(X_i)}$; donde, como ya vimos en teoría de grafos (Sec. 1.1), $pa(X_i)$ será el

conjunto de variables padres de X_i. En el caso de que la variable sea nodo raíz, se le asignará un conjunto credal marginal. Si cada red Bayesiana definía una distribución conjunta de probabilidad, una red credal definirá un *conjunto credal conjunto*. Asumiremos una hipótesis bastante común que fue establecida por Rocha *et al.* [163], la de conjuntos credales especificados por separado. Esto significa que para dos configuraciones de las variables padres de una variable dada, las probabilidades no guardan relación. Esto implica que para cada configuración de los padres $y_j \in \Omega_{pa(X_i)}$ se dará un conjunto credal local $H^{X_i|pa(X_i)=y_j}$. De un conjunto credal especificado por separado podemos obtener un conjunto credal condicional extenso.

Ejemplo 23 *Proponemos un sencillo ejemplo de red credal. En él tenemos únicamente dos variables X e Y con dos estados cada una.*



H^Y	y_1	y_2
r_1	0.1	0.9
r_2	0.2	0.8

$H^{X\|Y=y_1}$	x_1	x_2
p_1	0.2	0.8
p_2	0.3	0.7

$H^{X\|Y=y_2}$	x_1	x_2
q_1	0.4	0.6
q_2	0.6	0.4

Fig. 3.2: Ejemplo sencillo de red credal de dos variables bivariadas.

Vemos en la Figura 3.2 cómo la variable Y tiene asignada una tabla de probabilidad marginal que representa el conjunto credal asociado. Por otro lado, puesto que X depende de Y, a X le corresponden dos conjuntos credales especificados por separado, uno por cada estado de la variable padre $Y = \{y_1, y_2\}$. En la Tabla 3.1 que presentamos a continuación puede observarse como sería el conjunto credal condicional extenso relativo a X.

$H^{X\|Y}$	x_1, y_1	x_2, y_1	x_1, y_2	x_2, y_2
p_1, q_1	0.2	0.8	0.4	0.6
p_1, q_2	0.2	0.8	0.6	0.4
p_2, q_1	0.3	0.7	0.4	0.6
p_2, q_2	0.3	0.7	0.6	0.4

Tab. 3.1: Conjunto credal condicional extenso asociado a la variable X obtenido de los conjuntos credales especificados por separado $H^{X|Y=y_1}$ y $H^{X|Y=y_2}$.

□

Tal y como ocurre con las redes Bayesianas, mediante el DAG de la red credal pueden estudiarse relaciones de independencia y dependencia entre variables mediante el criterio de d-separación. Por otra parte, en la anterior Sección 3.1.1 estudiamos las distintas definiciones de independencia con convexos de probabilidad desde un punto de vista de la componente cuantitativa de la red, lo que será indispensable, ya que la definición de la red credal dependerá de la noción de independencia elegida. En el caso de independencia fuerte, que como comentamos será el más propio y el que consideraremos en esta memoria, al conjunto credal conjunto se le conocerá como *extensión fuerte*. La extensión fuerte de una red credal es el *conjunto credal conjunto* que contiene todas las combinaciones posibles de vértices para todos los conjuntos credales en la red [56]; es decir, cada vértice de una extensión fuerte se factoriza de la siguiente manera:

$$p(X_1, X_2, ..., X_N) = \prod_i p(X_i|pa(X_i)) \tag{3.10}$$

3.2.1 Inferencia sobre redes credales

En el capítulo anterior, Sección 2.4 podemos ver la base de los algoritmos de inferencia con probabilidades precisas. Recordamos que partimos de cierta evidencia dada y la idea es la de calcular probabilidades *a posteriori*, sacando partido de las independencias presentes entre las variables, de manera que el problema se resuelva sin tener que recurrir al cálculo de la distribución conjunta de probabilidad de todas las variables, para a continuación marginalizar sobre las variables pertinentes. Para Los potenciales que se darán en redes credales serán potenciales condicionales; que son conjuntos convexos de distribuciones de probabilidad condicionales (*i.e.* conjuntos credales condicionales extensos). Las redes credales dadas por intervalos de probabilidad, pueden ser interpretadas como un conjunto infinito de redes Bayesianas, donde todas ellas comparten el mismo grafo y como valores tienen los distintos valores contenidos en esos intervalos. De esta manera, realizar inferencia sobre la red credal equivaldría a ejecutar cualquier algoritmo de inferencia sobre cada una de las redes Bayesianas del conjunto infinito. Esta manera de afrontar el problema es, como era de esperar, tremendamente ineficiente. En esta memoria, consideraremos que calcular inferencia sobre una red credal supondrá calcular las probabilidades superior e inferior para cada estado de la variable sobre la que se hace la consulta, dada evidencia.

Puesto que, como hemos visto y definido, es posible llevar a cabo operaciones de marginalización y combinación sobre conjuntos convexos de probabilidad; podremos llevar a cabo los mismos

algoritmos de inferencia existentes para probabilidades precisas. En la literatura, diferentes algoritmos para inferencia con redes credales han sido desarrollados [9, 32, 163], aunque encontrar métodos que trabajen de un modo eficiente aún es un desafío, sobre todo para el caso de redes credales complejas. Y es que a medida que el número de variables y de estados de las variables incrementa, el tamaño de las distribuciones conjuntas de probabilidad aumenta exponencialmente. Así se muestra en el ejemplo propuesto por Rocha *et al.* [163]. En él se introduce una red credal formada por sólo tres nodos (X, Y, X) de cuatro estados cada una y donde cada conjunto credal tiene cuatro vértices. El DAG de esta red sigue la forma $X \longrightarrow Y \longleftarrow Z$. En este caso, hay 4^{18} (= 68.719.476.736) vértices potenciales para la extensión fuerte; o lo que es lo mismo, 4^{18} distribuciones conjuntas distintas que factorizan (ver Ecuación 3.10). En este mismo trabajo de Rocha *et al.* [163] se propone el uso de algoritmos de inferencia *branch-and-bound*, por ser capaces de trabajar con este tipo de problemas.

Como método para representar potenciales vamos a considerar, en este caso, árboles de probabilidad (PTs), que introduciremos a continuación y cuya definición será también imprescindible en el siguiente Capítulo 4. Los elegimos por tratarse de una representación más compacta que el caso de tablas de probabilidad y además permitir llevar a cabo sobre ellos operaciones básicas de restricción, marginalización y combinación.

Árboles de probabilidad

En el anterior Capítulo 2 quedo definido el concepto de potencial de probabilidad, así como las operaciones básicas sobre ellos para realizar inferencia. Recordamos que de manera clásica, los potenciales se han representado por medio de tablas de probabilidad, aunque dependiendo del problema, puede ser más recomendable el uso de representaciones alternativas. Una conocida alternativa son los *árboles de probabilidad* (PT, por sus siglas en inglés - *probability tree*), también conocidos como *árboles numéricos*. Los PTs a menudo se utilizan para almacenar potenciales y operar con ellos de manera exacta y aproximada [38, 42, 170].

Definición 3.2.2 ***(Árbol de probabilidad)*** *Un* ***árbol de probabilidad*** *que almacena un potencial ϕ, $\mathcal{T}_\phi$, es un árbol dirigido con dos tipos de nodos: (1) nodos internos que representan variables y (2) nodos hojas que serán valores reales no negativos y que representan valores del potencial. De los nodos internos salen arcos (uno por cada estado del potencial).*

Notaremos como $\mathcal{T}$ a un árbol de probabilidad sobre un conjunto de variables $\mathbf{X}$. El *tamaño* de un árbol de probabilidad $\mathcal{T}$, $size(\mathcal{T})$, se define como el número de nodos que contiene.

Ejemplo 24 *Un ejemplo sencillo puede verse en la siguiente Figura 3.3*

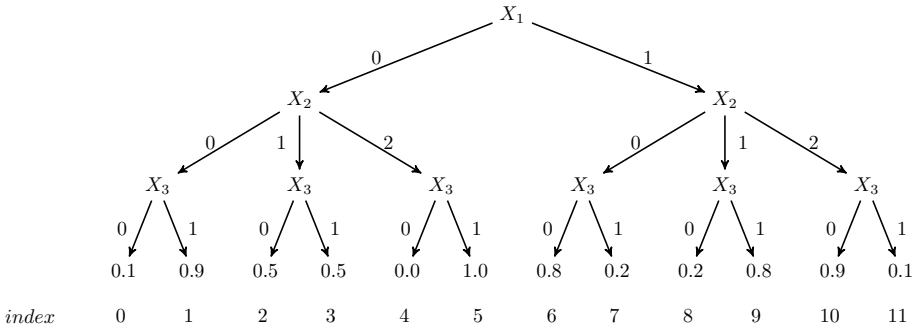


Fig. 3.3: Ejemplo de árbol de probabilidad.

□

Los PTs pueden ser restringidos a una o varias de sus variables. La restricción de $\mathcal{T}$ a un subconjunto de variables X_A será notado como $\mathcal{T}^{R(x_A)}$ y consistirá en la pare del árbol que sea consistente con los valores de las configuraciones x_A.

Ejemplo 25 *En la siguiente Figura 3.4 podemos ver un ejemplo de PT (parte izquierda) y el correspondiente PT restringido a que la variable X_1 tome el estado x_1^1.*

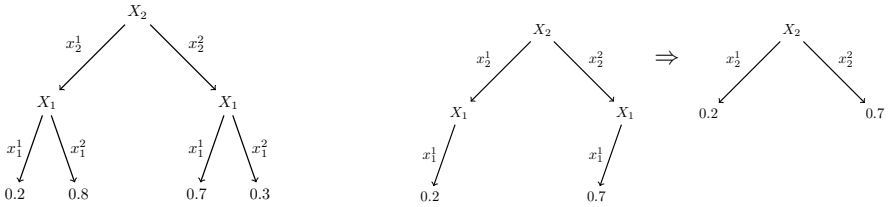


Fig. 3.4: Ejemplo de árbol de probabilidad (PT) 3.4(a) y su restricción a $X_1 = x_1^1$ 3.4(b).

□

Es posible realizar tareas de inferencia utilizando árboles de probabilidad porque las operaciones básicas de combinación y marginalización pueden llevarse a cabo con ellos. Para obtener más información al respecto puede acudirse a [42].

Ejemplo 26 *Vemos a continuación mediante un ejemplo de cómo se llevaría a cabo la combinación de dos PTs. En la parte superior izquierda de la Figura 3.5 tenemos dos árboles de probabilidad, el situado más a la izquierda es un nuevo árbol definido únicamente sobre la variable X_1 (llamémoslo $\mathcal{T}_1$) y el de la derecha es el mismo árbol que vimos en el anterior Ejemplo 25 (llamémoslo $\mathcal{T}_2$). En la figura superior derecha vemos cómo el árbol ha cambiado, ahora el nodo raíz es la variable X_1 (aunque podría haberse presentado de manera análoga con la raíz X_2); el paso importante aquí es que los valores de probabilidad de $\mathcal{T}_1$ se multiplican a las variables X_2. En las dos figuras de la parte inferior simplemente se multiplican los valores de un modo lógico y se calculan los valores resultantes.*

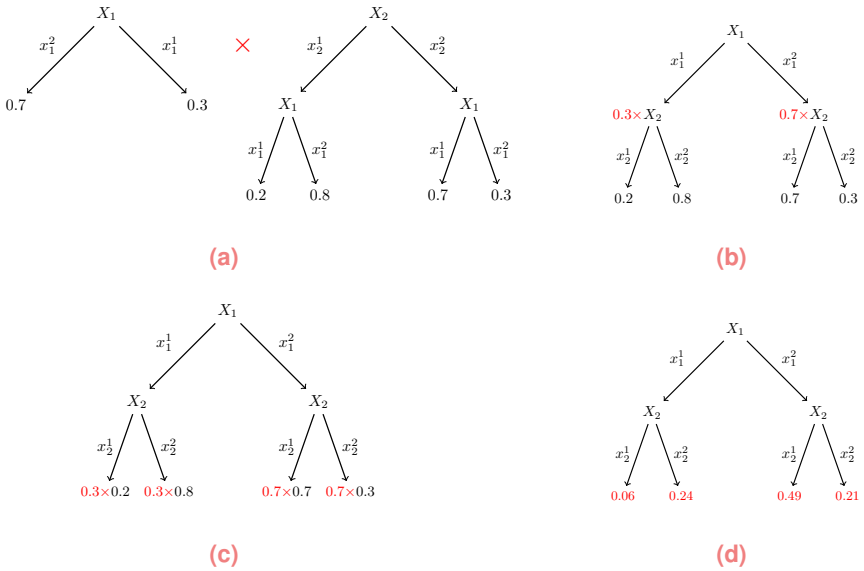


Fig. 3.5: Ejemplo de combinación de dos árboles de probabilidad (PTs).

□

Ejemplo 27 *Consideremos un PT definido sobre tres variables X_1, X_2, X_3, todas bivariadas, tal y como podemos observar en la parte superior izquierda de la Figura 3.6. Y supongamos que queremos marginalizar dicho PT para la variable X_3. El proceso a efectuar puede verse en las partes restantes de la misma Figura 3.6, en las que vemos como desaparece la variable X_3 dejando dos subárboles cuyos valores se suman convenientemente.*

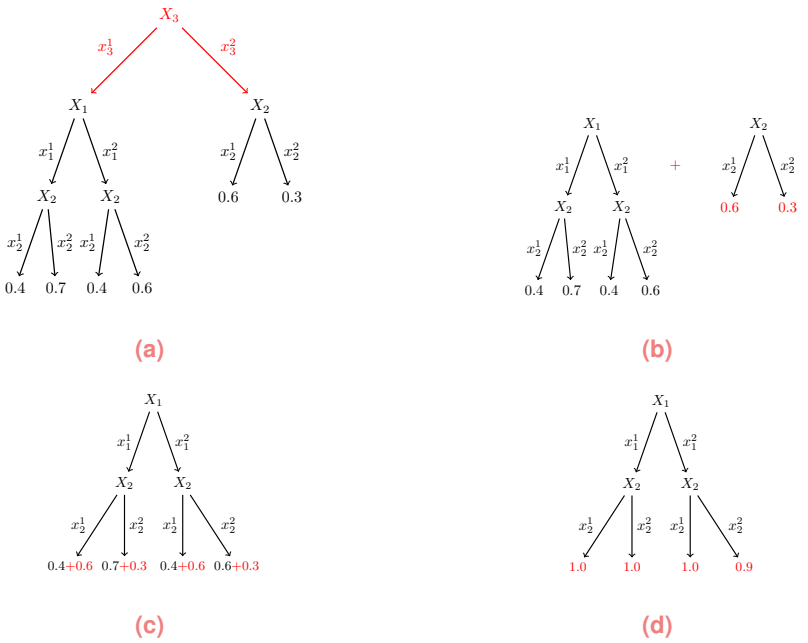


Fig. 3.6: Ejemplo de marginalización de un árbol de probabilidad (PT) a una de sus variables X_3.

□

Existen varios algoritmos para a llevar a cabo inferencia sobre redes credales con árboles de probabilidad [35, 41]. Sabemos que para cada una de las variables X_i del problema tendremos m conjuntos credales locales de la forma $\{H^{X_i|Y=y_1}, ..., H^{X_i|Y=y_m}\}$; por ser m el número de configuraciones de Y. Para permitir que un árbol de probabilidad pueda representar el problema completo, estos algoritmos se ayudan de la creación de nuevas variables, una para cada $y_j \in \Omega_Y$, llamadas *variables transparentes* y notadas como T_{y_j}. Al conjunto de todas las variables transparentes lo notaremos como $\mathbf{T}$. Una variable transparente T_{y_j} tendrá tantos estados como vértices tenga el conjunto credal local $H^{X_i|Y=y_j}$. Y un vértice del conjunto credal condicional extenso $H^{X_i|Y}$ podrá obtenerse fijando todas las variables transparentes a uno de sus valores.

Ejemplo 28 *En este ejemplo vamos a ver como un árbol de probabilidad puede representar un conjunto credal global $H^{X|Y}$ asociado a los dos conjuntos credales $H^{X|Y=y_1}$ y $H^{X|Y=y_2}$; definidos sobre las variables X e Y, ambas bivariadas. Supongamos los conjuntos credales locales del Ejemplo 23:*

$H^{X\|Y=y_1}$	x_1	x_2
p_1	0.2	0.8
p_2	0.3	0.7

$H^{X\|Y=y_2}$	x_1	x_2
q_1	0.4	0.6
q_2	0.6	0.4

Tab. 3.2: Ejemplo de conjuntos credales locales.

Como sabemos, el conjunto credal condicional extenso en forma de tabla se ve como mostramos en la Figura 3.2:

$H^{X\|Y}$	x_1, y_1	x_2, y_1	x_1, y_2	x_2, y_2
$r_1 = (p_1, q_1)$	0.2	0.8	0.4	0.6
$r_2 = (p_1, q_2)$	0.2	0.8	0.6	0.4
$r_3 = (p_2, q_1)$	0.3	0.7	0.4	0.6
$r_4 = (p_2, q_2)$	0.3	0.7	0.6	0.4

Tab. 3.3: Conjunto credal condicional extenso asociado a la variable X obtenido de los conjuntos credales especificados por separado $H^{X|Y=y_1}$ y $H^{X|Y=y_2}$.

Ahora bien, podemos expresar el conocimiento de este conjunto credal condicional extenso mediante un PT, recurriendo al uso de las variables transparentes T_{y_1} y T_{y_2}. Mostramos cómo quedaría el árbol de probabilidad en la Figura 3.7:

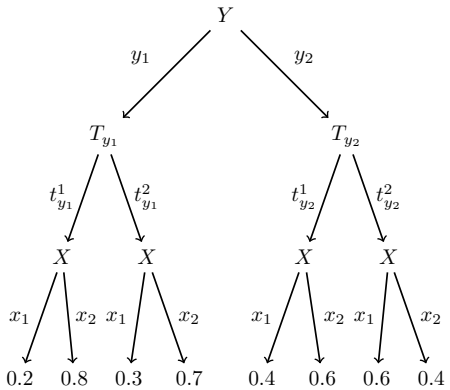


Fig. 3.7: Ejemplo conjunto credal condicional extenso asociado a los dos conjuntos credales especificados por separado $H^{X|Y=y_1}$ y $H^{X|Y=y_2}$, representado por su árbol de probabilidad.

Los puntos extremos pueden ser obtenidos al fijar las variables transparentes a uno de sus estados. En nuestro caso, por ejemplo, si consideramos $T_{y_1} = t^1_{y_1}$ *y* $T_{y_2} = t^2_{y_2}$*, obtenemos el punto extremo* r_2*.*

□

Los árboles de probabilidad tienen una gran ventaja frente a la representación de tabla, pueden ser reducidos en tamaño mediante la conocida operación de poda. Esta poda puede llevarse a cabo partiendo del PT original, de manera que la nueva representación saca provecho de la repetición de valores. Otra manera de obtener un PT podado más compacto es mediante algoritmos de aproximación. Estos PT aproximados dan como resultado un árbol con una pequeña pérdida de información aceptada, pero mejoran tiempos de cómputo y espacio en memoria.

Un sencillo algoritmo aproximado para la inferencia de conjuntos credales con árboles de probabilidad se basa en el algoritmo de eliminación de variables (VE) [62, 182, 211] (Sec. 4.2.6). En este algoritmo se eliminan, en primer lugar, las variables no imprescindibles, que son las relacionadas con la consulta y las transparentes. El algoritmo aplica poda para reducir el tamaño de PTs que representan las distribuciones condicionales iniciales y los potenciales resultado de operaciones de combinación y marginalización tras realizar propagación. Esa poda se realiza seleccionando iterativamente un árbol terminal y reemplazándolo por el valor promedio de los nodos hoja. El criterio para seleccionar un árbol terminal y aplicar poda sobre él, es que la distancia de Kullback-Leibler [117] entre el potencial inicial y el que se genera tras la poda no sea mayor que un umbral establecido σ. Este procedimiento se aplicará hasta que no exista ningún árbol terminal que cumpla el criterio anterior.

4 Representación de información cuantitativa en modelos gráficos probabilísticos

Tal y como vimos en el Capítulo 2, los modelos gráficos probabilísticos se componen de una parte cualitativa y otra cuantitativa. En esta parte de la memoria vamos a centrar nuestro estudio en la representación de la parte cuantitativa. En este caso, la información cuantitativa son potenciales. Recordamos que los potenciales probabilísticos son funciones reales no necesariamente normalizadas, ligadas a las distribuciones de probabilidad. Su representación se ha llevado a cabo tradicionalmente mediante tablas de probabilidad o tablas de probabilidad condicionadas (ver Sección 2.4.1 para más información). Este tipo de representación será la base con la que compararemos todas las demás. En el Capítulo 3, Sección 3.2.1 introdujimos los árboles de probabilidad, representación alternativa a las tablas de probabilidad ampliamente utilizada. En la Sección 4.1 vamos a hablar sobre las estructuras alternativas para representar potenciales. En la Sección 4.2 expondremos una de las aportaciones de esta memoria, unas nuevas estructuras (cuatro variantes) que nombramos *potenciales basados en valor* (VBPs - por sus siglas en inglés *value-based potentials*) y que representan potenciales de un modo compacto, sacando partido de la repetición de valores. Además, de manera similar a como se hace con árboles de probabilidad, definiremos un algoritmo para aproximar estas representaciones VBPs, dando así una herramienta para convertirlas en estructuras aún más compactas (ver Sección 4.3).

4.1 Representaciones alternativas previas para potenciales: árboles de probabilidad

Decimos que la representación de potenciales probabilísticos mediante tablas es exhaustiva, ya que es necesario especificar el valor correspondiente para cada configuración del potencial para que la representación se considere completa. Esto significa que el *tamaño de una tabla de*

probabilidad se corresponde con el tamaño del potencial y podrá calcularse como el producto del número de elementos de los dominios de todas las variables del potencial, es decir:

$$size(T^{\phi}(X_A)) = \prod_{X_i \in X_A} |\Omega_{X_i}|$$

A medida que el potencial a representar crece, el tamaño de la tabla aumenta exponencialmente. Por esta razón, trabajar y operar con dichos potenciales puede ser una tarea realmente tediosa y en muchas ocasiones computacionalmente imposible. Además, este tipo de representaciones no saca partido de la aparición de las llamadas independencias específicas del contexto [26], cuyo concepto ya estudiamos en la Sección 1.2.2, y que se traduce en valores repetidos en las tablas de probabilidad y el consecuente uso de espacio de memoria que podría evitarse. Es por eso que en la literatura se han desarrollado formas alternativas de representación tales como los árboles de probabilidad, árboles binarios, etc. [29, 30, 38, 42, 82, 170]. Pese a los inconvenientes que hemos presentado, en nuestro estudio consideraremos la representación de tabla como base para compararla con las estructuras alternativas, de manera que podamos observar la mejora que suponen.

Puesto que, en esencia, un potencial es un objeto multidimensional con una dimensión dada para cada variable, una manera común de almacenarlo en la memoria del ordenador es mediante un vector unidimensional (1DA - por sus siglas en inglés *one dimensional array*) [115].

Definición 4.1.1 ***(Vector unidimensional)*** *Dado un potencial ϕ definido sobre un conjunto de N variables, $X_1, X_2, ..., X_N$, puede representarse como un* ***vector unidimensional*** *(1DA) $\mathcal{A}_\phi$ como sigue:*

$$\mathcal{A}_\phi := \Big[\phi(0,0,\ldots,0), \phi(0,0,\ldots,1), \ldots, \phi(|\Omega_{X_1}|-1, |\Omega_{X_2}|-1, \ldots, |\Omega_{X_N}|-1)\Big] \quad (4.1)$$

La ventaja principal de la representación de potenciales mediante 1DAs consiste en que la posición en la que se almacena cada valor coincide con el índice correspondiente de la configuración, por lo que el acceso mediante índices es muy eficiente. Estos índices de la configuración coinciden con los de la representación en forma de tabla. El tamaño de un 1DA, $size(\mathcal{A}_\phi)$, es el número de entradas, que será igual al número de configuraciones en el potencial y que coincide con el tamaño de la representación de tabla.

Ejemplo 29 *Dado el potencial $\phi_1(X_1, X_2, X_3)$ que definimos en el Ejemplo 15, podemos observar en la Figura 4.1 su representación como 1DA de tamaño 12. Los números de la parte superior corresponden con los índices de las configuraciones y los inferiores los valores del potencial asociados.*

0	1	2	3	4	5	6	7	8	9	10	11
0.1	0.9	0.5	0.5	0.0	1.0	0.8	0.2	0.2	0.8	0.9	0.1

Fig. 4.1: Potencial $\phi_1(X_1, X_2, X_3)$ en su representación del tipo 1DA

□

Como introdujimos en el anterior capítulo 3, los *árboles de probabilidad* (PT), son estructuras alternativas que se utilizan para almacenar potenciales y operar con ellos de manera exacta y aproximada [38, 42, 170].

Ejemplo 30 *El mismo potencial que consideramos en el Ejemplo 15 se ve como árbol de probabilidad tal y como mostramos en la Figura 4.2. Este PT tiene un tamaño de 21 nodos (9 nodos internos y 12 hojas).*

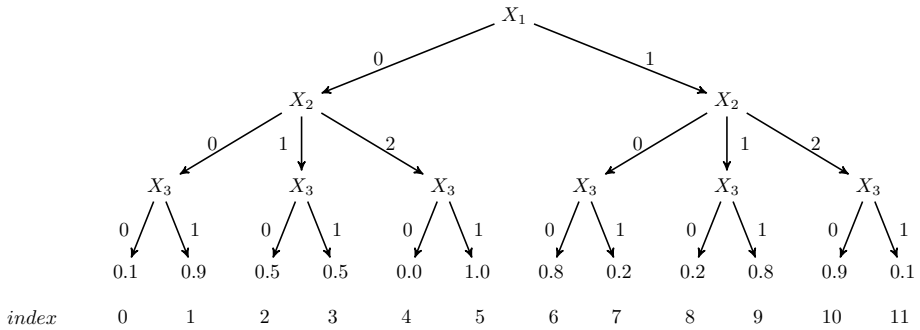


Fig. 4.2: Potencial $\phi_1(X_1, X_2, X_3)$ en su representación como PT

La representación de ese árbol de probabilidad comienza por la variable raíz X_1, que ramifica en dos arcos, una por cada estado de la variable (0, 1); a continuación se procede del mismo modo con la siguiente variable X_2, que tiene 3 estados (0, 1, 2); para terminar con X_3, con 2 estados (0, 1). Los nodos hoja son los valores de probabilidad para cada configuración de los valores de la variable. En la parte inferior se observan los índices de las configuraciones.

□

Al trabajar con PTs, el modo más eficiente de acceder es mediante configuraciones, recorriendo el árbol de raíz a hojas y seleccionando la rama correspondiente para cada variable hasta llegar al valor del nodo hoja.

Una gran ventaja de los PTs es que pueden beneficiarse de independencias específicas del contexto [26], permitiendo que valores iguales sean agrupados en uno solo. Esta operación se denomina *poda* y supone un ahorro en espacio de memoria, en ocasiones, muy considerable. Un árbol al que se le ha aplicado poda recibe el nombre de *árbol de probabilidad podado*, que denominaremos PPT (por sus siglas en inglés - *pruned probability tree*).

Ejemplo 31 *El potencial ϕ_1 que definimos en el Ejemplo 15 y hemos considerado en los ejemplos anteriores presenta independencias específicas del contexto, ya que para $X_1 = 0, X_2 = 1$ el valor del potencial es 0.5, sea cual sea el valor de X_3. En la Figura 4.3 presentamos el PPT correspondiente al potencial citado.*

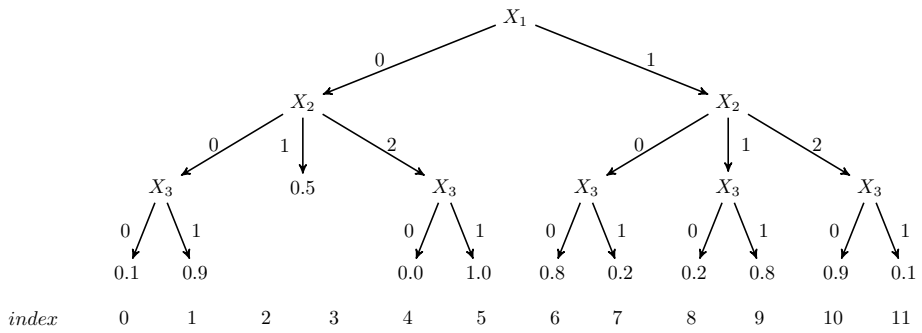


Fig. 4.3: Potencial $\phi_1(X_1, X_2, X_3)$ representado como PPT

□

Además, los PTs permiten ser aproximados, utilizando la misma operación de poda. Esto se realiza uniendo un número de nodos hoja asociados a la misma configuración de los padres en uno, de manera que este contenga el valor medio de ambos. En general, y con el objetivo de perder el mínimo de información posible, se suelen agrupar los valores que se encuentran más próximos. En la siguiente Figura 4.4 pueden observarse dos árboles, el de la izquierda sería el PT original y el de la derecha el aproximado. En este árbol podado, los valores 0.24 y 0.26 (que aparecían en el árbol original en los índices 3 y 4) se encuentran bastante próximos y se ha decidido unirlos en un sólo valor (0.25).

Por desgracia, los PTs no pueden beneficiarse del ahorro de espacio en memoria que supondría agrupar otros tipos de repetición de valores. Así, si observamos el PT definido en la Figura 4.2, podremos ver que los valores para los índices 7 y 8 son ambos 0.2, y además son consecutivos, pero realizar poda no será posible, ya que corresponden a configuraciones en que X_2 y X_3 varían.

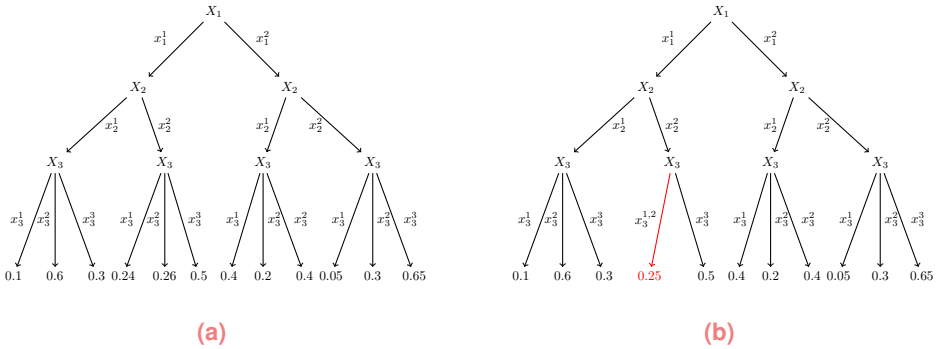


Fig. 4.4: Ejemplo de árbol de probabilidad (PT) y su árbol aproximado resultado de unir los valores 0.24 y 0.26 de los índices 3 y 4.

Otra variante de los PT son los árboles de probabilidad binarios [29, 30, 38] (abreviado como BPT por sus siglas en inglés - *binary probability trees*). Los BPTs son capaces de dividir el dominio de cada variable en dos subconjuntos de estados. Así, en el PT que presentamos a la izquierda de la Figura 4.5, los valores 0.4 en la configuración c (a la izquierda) no puede ser podado. Por el contrario, con BPs podemos agrupar los estados 0 y 2 de X_k para representar el valor 0.4 como un único nodo hoja (tal y como se aprecia en la parte de la derecha de Fig. 4.5). Esto permite que se reduzca el espacio de memoria para el contexto c, pero requerirá más nodos para representar el fragmento de árbol del lado derecho (contexto c'). Por esta razón, en esta memoria los BPTs no tomarán parte en nuestra comparación de estructuras.

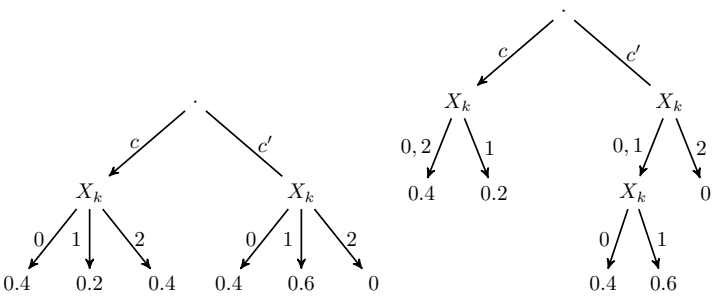


Fig. 4.5: Ejemplo de árbol de probabilidad binario (BPT)

4.2 Potenciales basados en valor

Hemos visto que la representación clásica en forma de tabla de probabilidad es exhaustiva, es decir, necesitamos especificar cada valor para que quede totalmente representada. Esta característica

de diseño de las CPTs hace que no puedan beneficiarse de las denominadas independencias específicas del contexto [26], cuyo concepto vimos en la Sec. 1.2; además, las CPTs crecen exponencialmente en tamaño cuando el número de variables se incrementa, lo que supone un gran problema para la representación de potenciales de gran volumen. En la literatura se ha tratado de lidiar con esta desventaja definiendo nuevas estructuras, como comentados árboles de probabilidad (PTs) y árboles de probabilidad binarios (BPTs) [29, 30, 38, 42, 82, 170]. Estas estructuras alternativas, por el contrario, sí pueden ahorrar espacio evitando la repetición de valores. Para el caso de PTs y BPTs es también posible realizar tareas de aproximación mediante la operación de poda, lo que conlleva más ahorro de espacio a cambio de una pérdida de información; y permite el tratamiento de potenciales aún más grandes. Por otro lado, sólo pueden beneficiarse de las independencias específicas del contexto en casos concretos, lo que aún podría mejorarse.

4.2.1 VBPs: definición

En este capítulo trataremos una de las aportaciones de esta memoria, una representación alternativa para potenciales que se beneficia de la repetición de valores independientemente del orden en el que aparezcan. Hemos denominado a estas representaciones *potenciales basados en valor* y abreviado como VBPs por sus siglas en inglés *Value-Based Potentials*. De manera similar a la operación de poda para PTs, los VBPs podrán también aproximarse. Las principales ventajas de los VBP sobre otras estructuras de datos relacionadas son las siguientes:

- En primer lugar, los requerimientos de memoria se reducen notablemente para algunas redes (como veremos en el estudio empírico, para muchas de ellas) debido a una mejor capacidad de explotar patrones de regularidad de valores. Veremos que en ocasiones, los resultados son incluso mejores que los obtenidos para la clásica 1DA, que es bastante eficiente.

- En segundo lugar, un VBP representa un solo potencial, independientemente de los otros parámetros del modelo. Esto permite adaptar fácilmente muchos algoritmos de inferencia para trabajar con VBP, simplemente ajustando las operaciones básicas sobre potenciales (por ejemplo, combinación y marginación).

- Además, la aproximación de VBPs es posible, permitiendo el trabajo con potenciales más complejos asumiendo una pérdida de información estimada *a priori*.

Los VBP, por otro lado, no incluyen ninguna información sobre dependencias de variables y se basan únicamente en los valores, por lo que no se pueden usar para representar modelos completos.

Uno de los objetivos principales será el de comparar estas nuevas estructuras VBPs con las estructuras más conocidas (1DA, PT, PPT), prestando especial interés al ahorro de espacio en memoria que suponen. Para ello y como prerrequisito, necesitaremos asentar las bases y conceptos para hacer dicha comparación. Esto se llevará a cabo en la Sección 4.2.1. Pasaremos entonces a introducir las cuatro alternativas de VPB que proponemos en la Sec. 4.2.1, donde especificaremos no sólo su definición y características, sino también algoritmos que permiten el acceso a un índice requerido y ecuaciones que indican el espacio que cada estructura requiere para ser almacenada. La Sección 4.2.3 de este capítulo nos proporciona los algoritmos para realizar operaciones básicas con potenciales dirigidos por valor (VBPs). En la Sec. 4.2.3 podremos ver el comportamiento de RBs reales donde los potenciales se expresan en cada una de las formas consideradas. En la Sección 4.3 determinamos cómo se realiza la aproximación de VBP, presentando así un mecanismo más para obtener potenciales más compactos. En la Sec. 4.3.5 evaluaremos empíricamente este algoritmo de aproximación.

Análisis de requisitos de memoria

El tamaño de la representación de un potencial ya nos da una idea aproximada de cómo de complejo es. Pero para llevar a cabo una comparación de complejidad exhaustiva y real entre las distintas estructuras, necesitaremos ejecutar un análisis más preciso de los requisitos de memoria. Cualquier representación de la información cuantitativa de un modelo necesitará de elementos adicionales (e.g. punteros, meta-información, etc.) para ser especificada y usará diferentes tipos de datos.

Consideramos un potencial ϕ definido sobre un conjunto de N variables aleatorias $\mathbf{X} = \{X_1, X_2, \dots X_N\}$. Definiremos los siguientes conceptos y notación relacionados con los distintos tamaños de memoria:

- s_f es la memoria requerida para almacenar un valor flotante.
- s_i es la memoria requerida para almacenar el índice correspondiente a una configuración específica del dominio Ω. Asumimos que se trata de un entero largo.
- s_r denota el tamaño de referencia de un objeto.

- s_v representa el espacio de memoria necesario para almacenar información sobre una variable: nombre, cardinalidad y nombres de sus estados. Puesto que depende de los nombres de las variables y sus estados, vamos a asumir un valor fijo para todas las variables (igualmente, este espacio de memoria es realmente pequeño al comparar con el que se usa para guardar un potencial, por lo que no será relevante). Incluso podemos definir una forma estándar de codificar variables con identificadores numéricos y seguir la misma estrategia para sus estados.

- s_s denota el espacio de memoria de la estructura de datos utilizada para guardar información: vector, lista, conjunto o diccionario, donde cada tipo se notará como s_{arr}, s_{list}, s_{set} y s_{dict} respectivamente.

Vista la notación, pasemos a estudiar el espacio de memoria que ocupa cada tipo de representación. Como ya hemos comentado, la representación como 1DA tiene una gran ventaja, ya que los valores de la configuración se almacenan consecutivamente. Es decir, el valor en la posición k se corresponde con la k-ésima configuración. Por lo tanto, no necesitaremos almacenar información sobre los índices. La codificación de la representación asume una cantidad de memoria dada por el número de valores a almacenar, el tamaño de la estructura de datos del vector y la memoria requerida para sus variables.

Proposición 4.2.1 ***(Espacio de memoria de un vector que representa un potencial)***

Sea $\mathcal{A}_\phi$ el vector unidimensional, 1DA, que representa al potencial $\phi(\boldsymbol{X})$, donde $|\boldsymbol{X}| = N$. La cantidad de memoria requerida viene dada por la siguiente expresión:

$$memory(\mathcal{A}_\phi) = N \cdot s_v + m \cdot s_f + s_s \tag{4.2}$$

donde $m = |\Omega_{\boldsymbol{X}}|$ es el número de elementos del vector.

Ejemplo 32 *Si consideramos entonces el potencial $\phi_1(X_1, X_2, X_3)$ expresado en forma de vector que vimos en el Ejemplo 29, el tamaño de memoria estimado sería:*

$$memory(\mathcal{A}_{\phi_1}) = 3s_v + 12s_f + s_s$$

□

La representación de árbol (PT y PPT) suele ser menos eficiente en términos de requisitos de memoria, ya que la estructura completa del árbol necesitará ser guardada. Además, la cantidad de

memoria depende del número de nodos internos, n_I, y del número de hojas n_L. Notamos que $n_I + n_L = size(\mathcal{T})$. Los nodos internos almacenan enlaces (o referencias) a subárboles (uno por cada estado de la variable aleatoria correspondiente). Estos enlaces se almacenan en un vector. Es entonces relevante considerar el número de arcos salientes. Denotamos como $n_I^{(j)}$ al número total de nodos internos para variables con j estados.

Proposición 4.2.2 ***(Espacio de memoria de un árbol de probabilidad que representa un potencial)*** *Sea $\mathcal{T}_\phi$ el árbol de probabilidad, PT, que representa al potencial $\phi(\boldsymbol{X})$, con $|\boldsymbol{X}| = N$. Entonces, la cantidad de espacio en memoria necesario se estima mediante la siguiente ecuación:*

$$memory(\mathcal{T}_\phi) = N \cdot s_v + n_L \cdot s_f + \sum_{j=2}^{K} n_I^{(j)} \cdot (s_v + s_s + j \cdot s_r) \tag{4.3}$$

donde $K = \max\{|\Omega_{X_i}| : X_i \in \boldsymbol{X}\}$ es la máxima cardinalidad de entre las variables del potencial.

En el caso de árboles no podados, el número de nodos hojas coincidirá con el número de configuraciones en el potencial. Por lo tanto, los dos primeros términos en la Ecuación 4.3 serán iguales. Así, los principales factores que determinan el tamaño de un árbol son la estructura de datos empleada para almacenar enlaces a subárboles y la información sobre repetición de variables.

Ejemplo 33 *Sea $\mathcal{T}_{\phi_1}$ el PT del Ejemplo 30 (Fig. 4.2) que contiene 7 nodos internos de variables binarias, 2 nodos internos para variables de tres estados, y 12 nodos hoja. De un modo similar, sea $\mathcal{T}'_{\phi_1}$ el PPT del Ejemplo 31 (Fig. 4.3) con 6 nodos internos para variables binarias, 2 nodos internos de tres estados, y 11 nodos hoja. Para computar su coste de memoria, usaremos la siguiente expresión:*

$$memory(\mathcal{T}_{\phi_1}) = 3s_v + 12s_f + 7(s_v + s_s + 2s_r) + 2(s_v + s_s + 3s_r)$$
$$memory(\mathcal{T}'_{\phi_1}) = 3s_v + 11s_f + 6(s_v + s_s + 2s_r) + 2(s_v + s_s + 3s_r)$$

Notemos que tras realizar la operación de poda, el número de nodos internos y nodos hoja ha disminuido. Por otro lado, el coste en ambos casos es mayor que el tamaño de la representación de tabla. En la práctica, para que se ahorre espacio con la representación de árbol podado, será necesario que el número de valores repetidos en posiciones susceptibles de ser podados sea considerablemente grande.

□

Para cuantificar el análisis, supongamos los siguientes tamaños de tipos de datos (en realidad los tamaños pueden variar según la arquitectura de la máquina utilizada). Por otro lado, los tamaños reales no son relevantes siempre y cuando se utilicen los mismos tamaños para todas las comparaciones):

- entero: 4 bytes (para índices).
- flotante: 8 bytes (para valores reales).
- puntero o referencia: 8 bytes (direcciones de memoria).
- variable: 50 bytes (incluyendo el espacio para almacenar nombre, nombres de estados, etc.). Lo que se traduce en $s_v = 50$.
- el valor de s_s dependerá de la estructura de datos elegida:
  - vector y lista (s_{arr} y s_{list} respectivamente): 16 bytes.
  - conjunto (s_{set}): 32 bytes.
  - diccionario (s_{dict}): 64 bytes.

Para estos tamaños, el coste estimado de memoria de las representaciones $\mathcal{A}_{\phi_1}$, $\mathcal{T}_{\phi_1}$ y $\mathcal{T}'_{\phi_1}$ es 262, 1000 y 910 respectivamente.

Motivación

Llegados a este punto, está claro que existe una necesidad de definir mecanismos eficientes capaces de manejar información cuantitativa para las tareas de representación, inferencia y aprendizaje de MGPs. Ya hemos visto que los PT permiten capturar algunos patrones de repetición en situaciones muy concretas. La estructura que definimos en esta memoria, los potenciales basados en valor ,VBPs (Value-Based Potentials), surge con la idea de dejar que los valores guíen el proceso de representación y ahorrar tanto espacio de repeticiones como sea posible. Por lo tanto, los VBPs se diseñaron con los siguientes objetivos en mente:

- Ser capaces de sacar partido de los patrones de repetición de valores, independientemente del orden en que aparezcan.

- Permitir un acceso eficiente a los valores y proveer de las operaciones necesarias para realizar tareas de inferencia. En esta memoria se proporcionan también implementaciones básicas para las operaciones de combinación y marginalización para dar una estimación inicial de cómo será el comportamiento de VBPs en tareas de inferencia.
- Facilitar métodos de aproximación de las estructuras de VBPs.

Idea intuitiva

Una manera realmente simple de visualizar el conjunto de valores podría ser mediante algo similar a un diagrama de Venn; tendríamos un círculo exterior representando el dominio y en el que observamos cada uno de los valores del potencial dispuestos convenientemente (es decir, cada valor se almacena cerca de sus índices correspondientes). Así, el ejemplo que venimos considerando podría verse como se presenta en la siguiente Fig. 4.6. Observamos que el índice 4, relacionado con el valor 0.0 (valor predeterminado por defecto), no se almacena. Esto sería similar a almacenar los valores usando una estructura 1DA.

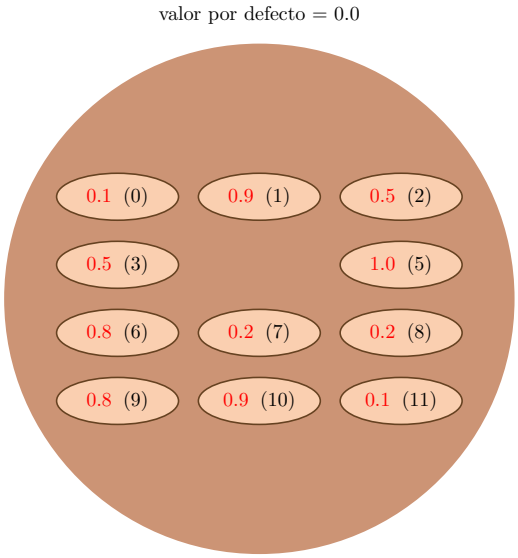


Fig. 4.6: Idea visual del potencial $\phi_1(X_1, X_2, X_3)$ representado la relación entre valores e índices.

Sin embargo, el objetivo de nuestro estudio es el de evitar almacenar valores duplicados. Vemos como los valores $0.1, 0.9, 0.5, 0.8$ y 0.2 aparecen asociados a distintos índices; por ejemplo, 0.1 se asocia con los índices 0 y 11. Las estructuras VBPs, buscan representar cada valor de probabilidad una sola vez, de manera que los índices en los que ese valor aparece se encuentren

vinculados con él. En la Fig. 4.7 se pueden observar las agrupaciones que se llevarían a cabo. Destacamos que en este caso sólo 6 valores se almacenan, frente a los 11 que se almacenarían en la representación presente en la Figura 4.6. Así, el propósito de los VBPs es hacer tantos grupos como valores diferentes de probabilidad, para evitar así el almacenamiento innecesario de repeticiones y asociar los índices de configuraciones relacionados.

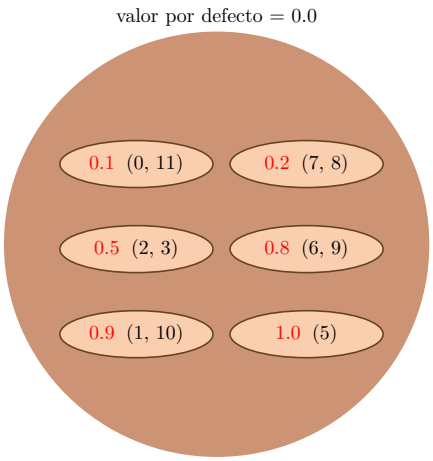


Fig. 4.7: Idea visual del potencial $\phi_1(X_1, X_2, X_3)$ agrupando valores de probabilidad.

Proponemos cuatro representaciones VBPs alternativas, que pueden ser clasificadas en dos grupos dependiendo de cómo se hacen las consultas:

- Enfoques *dirigidos por los valores*, basados en el uso de diccionarios en los que las claves serán los valores. En este grupo encontramos dos de las representaciones: **potenciales dirigidos por valor con granos** (*Value-Driven with Grains*, VDG) y **potenciales dirigidos por valor con índices** (*Value-Driven with Indices*, VDI).

- Enfoques *dirigidos por índices*, donde las claves son los índices. En este grupo encontramos las alternativas: **potenciales dirigidos por índices con índices** (*Index-Driven with Indices*, IDP) y **potenciales dirigidos por índices con mapas** (Index-Driven with Map, IDM). Ambas utilizan un vector de valores, V. IDP utiliza también otro vector, L, que almacena los índices y la información requerida para vincular índices y valores. IDM utiliza un mapa, M, con los índices como claves y la información para enlazar con V como valores.

Presentaremos en detalle estas cuatro alternativas, pero antes queremos poner de manifiesto que las cuatro comparten una característica común, y es que para manejar la información será necesario ejecutar una búsqueda. En esta búsqueda, lo que se requiere es obtener el valor de probabilidad correspondiente a un índice conocido de la configuración. Cómo realizar este procedimiento para cada VBP será descrito en forma de algoritmos de acceso en cada apartado correspondiente a la estructura. Con el objetivo de reducir espacio en memoria aún más, podemos aprovechar esa búsqueda definiendo un valor por defecto, de manera que si la búsqueda falla sabremos que el valor es el que definimos por defecto. En esta memoria consideraremos que ese valor es 0, aunque podría considerarse igualmente cualquier otro; siendo la opción más interesante desde el punto de ahorro de espacio de memoria la de elegir el valor que más se repita.

Para cada una de las cuatro estructuras VBPs (VDG, VDI, IDP, IDM), veremos su definición de manera rigurosa, el algoritmo de acceso a un índice dado y la ecuación que determina el espacio en memoria que esa representación requiere para un potencial. Ilustraremos todos estos conceptos usando nuestro ejemplo de potencial, ϕ_1, que definimos en el Ejemplo 15.

Potenciales dirigidos por valores con granos

En los potenciales, a menudo aparecen valores idénticos en configuraciones con índices consecutivos. Si observamos el potencial dado en el Ejemplo 29, el valor 0.5 aparece en las posiciones 2 y 3. De manera similar ocurre con el valor 0.2 en las posiciones 7 y 8. Los conjuntos de configuraciones asociados a un mismo valor se pueden definir de forma compacta mediante el uso de intervalos (*i.e.* granos). Formalmente, un grano se puede definir de la siguiente manera:

Definición 4.2.1 ***(Grano)*** *Sea* $\boldsymbol{X}$ *un conjunto de variables y sean* i *y* j *índices de configuraciones válidas de* $\Omega_{\boldsymbol{X}}$*, con* $i \leq j$*. Un* ***grano*** $g(i, j)$ *define una secuencia de índices consecutivos que comienza en* i *y acaba en* j*,* $i, i+1, \ldots j$*.*

Para simplificar, notaremos al grano $g(i, j)$ como se hace normalmente con intervalos, (i, j). Utilizaremos granos para representar secuencias de valores repetidos en VBP, siendo su uso especialmente ventajoso en potenciales de buen tamaño que presenten el mismo valor de probabilidad en un gran número de índices consecutivos. En un VDG que representa un potencial, cada valor distinto de cero (nuestro valor por defecto) se asociará con un grano, definiendo todos los índices para los que el potencial tiene este valor. Notaremos que en el caso en el que el grano sea de la forma (i, i), entenderemos que ese valor de probabilidad aparece en el índice i, pero no en

el consecutivo $i+1$. Definimos de manera rigurosa un potencial en su expresión dirigida por valores con granos (VDG) como:

Definición 4.2.2 ***(Potencial dirigido por valores con granos)*** *Sea ϕ un potencial definido sobre $\boldsymbol{X}$. Entonces, un* ***potencial dirigido por valores con granos*** *para ϕ, VDG_ϕ, es un diccionario D cuyas entradas se definen como $< v, L_v >$, donde $v \in \phi$ (clave) es un valor distinto del seleccionado por defecto y L_v es una lista de granos que almacenan los índices asociados a ese valor v. Por lo tanto, para cada grano $g(i,j) \in L_v$, todos los índices se corresponden con v (es decir, para todo $k = i, i+1, \ldots, j$ entonces $\phi(\boldsymbol{x}_k) = v$).*

Ejemplo 34 *El potencial $\phi_1(X_1, X_2, X_3)$ que hemos usado en los ejemplos anteriores y que se definió en el Ejemplo 15 se representará como VDG como mostramos en la Fig. 4.8.*

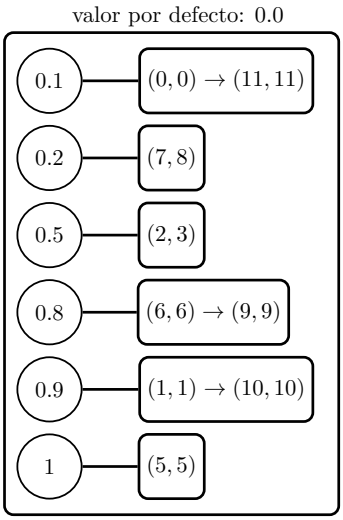


Fig. 4.8: Potencial $\phi_1(X_1, X_2, X_3)$ representado como VDG.

El rectángulo situado en el exterior con las esquinas redondeadas representa el diccionario. Los nodos circulares representan cada una de las claves y las listas de granos asociadas a cada clave se encuentran en la parte derecha. Podemos observar que cada lista de índices consecutivos viene representada como un grano y la separación entre granos es una flecha.

En la Fig. 4.8 observamos que cada valor de probabilidad se almacena únicamente una vez. Algunos de los valores de ϕ_1 aparecen sólo una vez en todo el potencial, como es el caso de 1.0, cuyos extremos inicial y final del grano coinciden y son 5. El resto de los valores sí se encuentran repetidos, 0.1, por ejemplo, es el valor correspondiente a los índices 0 y 11. Puesto que estos

índices no son consecutivos, necesitarán ser almacenados en dos granos distintos. Por otro lado, los valores 0.2 *y* 0.5 *sí aparecen en índices consecutivos y sus granos correspondientes hacen visibles las secuencias de repeticiones.*

□

Algoritmo 1 (Acceso a un índice en VDG) *Dado un potencial* ϕ *expresado en su versión VDG (VDG*$_\phi$*), el algoritmo para acceder al valor correspondiente a un índice* l *se describe en el siguiente Algoritmo 1.*

Algoritmo 1 Acceso al índice l en VDG$_\phi$

```
1:  funcion acceso
2:      resultado = 0.0                          ▷ define el valor por defecto como "result"
3:      para cada v (clave) en el conjunto de claves D hacer: ▷ bucle sobre las entradas del diccionario
4:          L_v ← D(v)                           ▷ lista de granos para v
5:          para cada g (grano) en L_v hacer:
6:              si l ∈ g entonces:               ▷ l se incluye en g
7:                  resultado = v y terminar iteración
8:              fin condicion
9:          fin bucle
10:     fin bucle
11:     devolver resultado
12: fin funcion
```

Asumamos que el índice que buscamos en nuestro potencial ϕ_1 es el 6. El proceso de búsqueda entre las entradas del diccionario continúa hasta encontrar la clave 0.8. El bucle interno (línea 5) itera sobre los granos para este valor. Como el primer grano que contiene el índice 6 es el valor 0.8, entonces la búsqueda finaliza y el resultado de la búsqueda es 0.8. Si la búsqueda falla y no se encuentra ningún índice, sabremos que 0.0 (el valor por defecto) será el que se devuelva, como ocurriría para el índice 4.

Proposición 4.2.3 ***(Espacio de memoria para un VDG que representa un potencial)*** *Sea VDG*$_\phi$ *la representación de* $\phi(\boldsymbol{X})$*, con* $|\boldsymbol{X}| = N$ *y* d *el número de valores diferentes en el potencial (sin contar el valor por defecto). El número de granos para cada valor se denota como* $g_1, g_2, \ldots, g_d$*. Por tanto, la cantidad de memoria estimada sigue la siguiente ecuación:*

$$memory(VDG_\phi) = N \cdot s_v + s_f + s_{dict} + d \cdot (s_f + s_{list}) + \sum_{j=1}^{d} 2 \cdot g_j \cdot s_i \tag{4.4}$$

Los términos de la Ecuación 4.4 consideran los tamaños de: variables, almacenamiento del valor por defecto, diccionario, valores, listas y granos con dos entradas para cada grano. El número de granos para cada valor dependerán de las secuencias de repetición, que serán más largas si las secuencias son mayores. Por lo tanto, el punto crítico de esta representación es el número de granos necesarios, que viene dado por $\sum_{j=1}^{d} g_j$.

Ejemplo 35 *Considerando el potencial $\phi_1(X_1, X_2, X_3)$ del Ejemplo 15 dado por su expresión en VDG, VDG$_{\phi_1}$. Puesto que contiene 6 valores diferentes a almacenar en el diccionario y su valor por defecto es 0.0, el diccionario tendrá 6 entradas. Dada la secuencia de repetición, serán necesarios 9 granos. El coste de memoria puede entonces calcularse mediante la siguiente expresión:*

$$memory(\textit{VDG}_{\phi_1}) = 3s_v + s_f + s_{dict} + 6 \cdot (s_f + s_{list}) + 9 \cdot 2 \cdot s_i$$

Utilizando los tamaños de memoria que describimos en la Sección 4.2.1, la cantidad de memoria total es 438 bytes. Recordamos que los tamaños de memoria para las representaciones 1DA, PT y PPT son 262, 1000 y 910 bytes respectivamente.

□

Potenciales dirigidos por valor con índices

Si bien la estructura anterior con granos es una representación compacta para potenciales, podría codificar información innecesaria cuando los valores repetidos no están en posiciones consecutivas. Este es el caso de 0.1 en la Fig. 4.8, cuya entrada incluye 2 granos de longitud 1: $(0, 0)$ y $(11, 11)$. Esta repetición puede evitarse, por ejemplo, asociando valores a la lista completa de índices en los que aparecen. Con esta idea en mente se define la alternativa VDI, en la que sustituiremos los granos por listas. En nuestro ejemplo, el valor 0.1 estará relacionado con la lista $(0 \rightarrow 11)$. Esta alternativa será especialmente ventajosa si se produce repetición de valores, pero la secuencia de valores no contiene grandes series de repeticiones.

Definición 4.2.3 ***(Potencial dirigido por valores con índices)*** *Sea ϕ un potencial definido sobre el conjunto de variables aleatorias $\boldsymbol{X}$. Un **potencial dirigido por valores con índices** para ϕ, VDI$_\phi$, es un diccionario D en el que cada entrada $< v, L_v >$ contiene un valor v (como clave) y una lista de índices L_v, tal que $\phi(\boldsymbol{x}_i) = v$ para todo $l \in L_v$.*

Ejemplo 36 *El potencial $\phi_1(X_1, X_2, X_3)$ usado anteriormente y descrito en el Ejemplo 15 se representará en su forma VDI como se observa en la Fig. 4.9.*

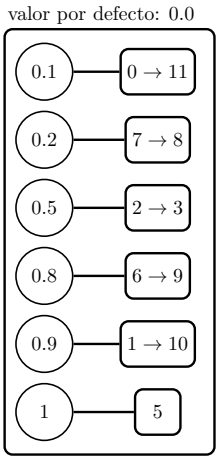


Fig. 4.9: Potencial $\phi_1(X_1, X_2, X_3)$ representado como VDI.

El rectángulo exterior representa el diccionario, donde las claves de las entradas son los valores de probabilidad y se dibujan como círculos. Las claves dan acceso a listas de índices, que vienen representados como los rectángulos con esquinas redondas que podemos localizar en el interior del diccionario. Todos los índices aparecen y se separan mediante una flecha.

□

Algoritmo 2 (Acceso a índice en VDI) *Dado un VDI$_\phi$, el algoritmo para acceder al valor correspondiente a un índice dado l se describe en el siguiente Algoritmo 2.*

Algoritmo 2 Acceso al índice l en VDI$_\phi$

```
1:  funcion acceso
2:      resultado = 0.0                          ▷ define el valor por defecto como "resultado"
3:      para cada v (clave) en el conjunto de claves D hacer: ▷ bucle sobre las entradas del diccionario
4:          L_v ← D(v)                           ▷ lista de índices para v
5:          para cada p en L_V hacer:            ▷ bucle sobre la lista de índices
6:              si p == l entonces:              ▷ l se incluye en L_v
7:                  resultado = v y parar iteracion
8:              fin condicion
9:          fin bucle
10:     fin bucle
11:     devolver resultado
12: fin funcion
```

Asumamos que 6 es el índice de ϕ_1 buscado. El proceso de búsqueda a través de las entradas del diccionario continuará hasta encontrar la clave 0.8. El bucle interno (línea 5) itera sobre la lista. Como se encuentra el índice 6, el resultado será 0.8. En el caso de que la búsqueda falle, sabremos que el valor por defecto 0.0 será el resultado, como en el caso del índice 4.

Proposición 4.2.4 ***(Espacio de memoria para un VDI que representa un potencial)*** *Sea VDI_ϕ la representación dirigida por valor con valores del potencial $\phi(\boldsymbol{X})$, donde $|\boldsymbol{X}| = N$. Asumamos que d representa el número de valores diferentes en el potencial (sin contar con el valor por defecto). El número de índices para cada valor se denota como $i_1, i_2, \ldots, i_d$. Entonces, la memoria necesaria estimada se puede calcular mediante la Ecuación 4.5:*

$$memory(VDI_\phi) = N \cdot s_v + s_f + s_{dict} + d \cdot (s_f + s_{list}) + \sum_{j=1}^{d} i_j \cdot s_i \tag{4.5}$$

Los términos de la Ecuación 4.5 consideran tamaños para: variables, el valor por defecto, diccionario, valores, listas e índices. En VDI, el recurso de ahorro de memoria mayor se produce al evitar el almacenamiento de valores repetidos, ya que todos los índices en los que los valores distintos del valor por defecto aparecen deberán almacenarse de manera explícita.

Ejemplo 37 *Sea VDI_{ϕ_1} el potencial VDI del Ejemplo 36, con 6 valores diferentes que almacenar en el diccionario y sea 0.0 el valor por defecto. Entonces, el diccionario almacena 6 pares y 6 listas con 11 índices. El coste de memoria puede calcularse usando la siguiente expresión:*

$$memory(VDI_{\phi_1}) = 3s_v + s_f + s_{dict} + 6 \cdot (s_f + s_{list}) + 11 \cdot s_i$$

Usando los tamaños de memoria que describimos en la Sección 4.2.1, la cantidad de memoria completa será 410 bytes, algo menor a la alternativa VDG. Recordamos que los tamaños de memoria para las representaciones 1DA, PT, PPT y VDG son 262, 1000, 910 y 438 bytes respectivamente.

□

Potenciales dirigidos por índices con pares

El problema para acceder a las estructuras dirigidas por valor es la necesidad de realizar una iteración doble. El proceso de buscar un índice objetivo, l, requiere iterar sobre la lista de entradas

y sobre las listas asociadas (de granos o índices). Para evitar esta doble iteración y hacer la búsqueda más eficiente, se introducen nuevas estructuras cuya búsqueda se basa en los propios índices. Este es el caso de las alternativas IDP e IDM.

Definición 4.2.4 ***(Potencial dirigido por índices con pares)*** *Sea ϕ un potencial definido sobre el conjunto de variables aleatorias **X**. Un **potencial dirigido por índices con pares** para ϕ, IDP_ϕ, es un par de vectores: V y L. Los valores no repetidos en ϕ (excluyendo el valor por defecto 0.0) se almacenan en $V := \{v_0, v_1, \ldots, v_{d-1}\}$. Sea nd_ϕ la representación del conjunto de índices que almacenan valores distintos al definido por defecto. El vector L se define como sigue:*

$$L := \{(i, j) : \phi(\boldsymbol{x}_i) = v_j, \quad i \in nd_\phi\} \tag{4.6}$$

Por lo tanto, IDP se basa en dos componentes. En primer lugar, un vector que almacena los valores (excluyendo el valor por defecto). Y en segundo lugar, un vector de pares (índice en el potencial - índice en el vector de valores), donde el segundo índice del par es el que relaciona cada índice con su valor.

Ejemplo 38 *La representación como IDP del potencial $\phi_1(X_1, X_2, X_3)$ definido en la Fig. 2.5 se muestra en la Figura 4.10.*

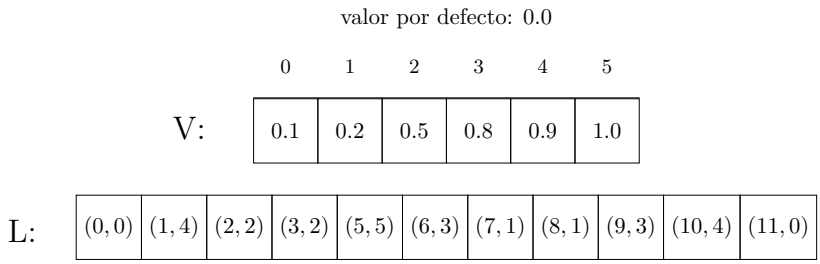


Fig. 4.10: Potencial $\phi_1(X_1, X_2, X_3)$ representado como IDP.

Como hemos explicado, IDP utiliza dos vectores: V almacena los valores no repetidos (a excepción del definido por defecto, que en nuestro caso es 0.0); y L, que contiene los pares de índices. Fijémonos, por ejemplo, en el valor 0.5 en $\phi_1(X_1, X_2, X_3)$, presente en los índices 2 y 3. Entonces, el vector de pares, L (parte inferior de Fig. 4.10), requiere dos pares para esta relación: $(2, 2)$ y $(3, 2)$. Ambos indican que los índices del potencial 2 y 3 almacenan el valor disponible en $V(2)$.

□

Algoritmo 3 ***(Acceso a índice en IDP)*** *Dado un potencial IDP$_\phi$, el algoritmo para acceder al valor correspondiente a un índice dado l se describe en el siguiente Algoritmo 3.*

Algoritmo 3 Acceso a índice l en IDP$_\phi$

```
1: funcion acceso
2:     resultado = 0.0                          ▷ define el valor por defecto como "resultado"
3:     para cada par t = (i_φ, i_V) en L hacer:   ▷ bucle sobre el vector de pares L
4:         si i_φ == l entonces:                 ▷ l es encontrado; se almacena el valor en V(i_V)
5:             resultado = V(i_V) y parar iteración
6:         fin condicion
7:     fin bucle
8:     devolver resultado
9: fin funcion
```

Asumamos que 6 es el índice de ϕ_1 que buscamos. Se busca el valor en L hasta llegar a la posición sexta (par $(6, 3)$). El valor a devolver se guarda en $V(3)$ y es 0.8. Si se busca el índice 4, se devuelve 0.0 (por no encontrarse el índice).

Proposición 4.2.5 ***(Espacio de memoria para un IDP que representa un potencial)*** *Sea IDP$_\phi$ la representación dirigida por índices con pares del potencial $\phi(\boldsymbol{X})$, con $|\boldsymbol{X}| = N$. Asumamos que d representa el número de valores diferentes en el potencial (sin tener en cuenta el fijado por defecto). El número de índices correspondientes con un valor distinto al de por defecto es p. La cantidad de memoria necesaria para almacenarla sigue la siguiente expresión:*

$$memory(\textit{IDP}_\phi) = N \cdot s_v + s_f + 2 \cdot s_{arr} + d \cdot s_f + 2 \cdot s_i \cdot p \tag{4.7}$$

Los términos en la Ecuación 4.7 consideran los tamaños de: variables, valor por defecto, ambos vectores, valores y pares de índices. Esta representación intenta usar estructuras simples y favorece la búsqueda directa sobre índices más que de valores.

Ejemplo 39 *Sea IDP$_{\phi_1}$ el potencial IDP del Ejemplo 38 con 6 valores diferentes que almacenar y con 0.0 como valor por defecto. Entonces, V almacena 6 valores y L 11 pares. El coste de memoria puede computarse utilizando la siguiente expresión:*

$$memory(\textit{IDP}_\phi) = 3s_v + s_f + 2 \cdot s_{arr} + 6 \cdot s_f + 11 \cdot 2 \cdot s_i$$

Utilizando los tamaños de memoria descritos en la Sección 4.2.1, la cantidad de memoria total será de 326 bytes. Recordamos que los tamaños de memoria para las representaciones 1DA, PT, PPT, VDG y VDI son 262, 1000, 910, 438 y 410 bytes respectivamente.

□

Potenciales dirigidos por índices con mapas

Esta estructura pretende dar un paso más para facilitar el acceso a la misma y lo hace utilizando un diccionario cuyas claves son los índices.

Definición 4.2.5 ***(Potencial dirigido por índices con mapas)*** *Sea ϕ un potencial definido sobre $\boldsymbol{X}$. Entonces, un* ***potencial dirigido por índices con mapas*** *para ϕ, IDM_ϕ, consiste en un diccionario D y un vector V. Los valores sin repetir de ϕ se almacenan en $V := \{v_0, v_1, \ldots, v_{d-1}\}$. Las entradas de D, $< i, j >$, están formadas por índices de ϕ (claves) y los índices de V. Sea nd_ϕ la representación del conjunto de índices que almacenan valores distintos del seleccionado por defecto. Dada una entrada $< i, j >$, entonces: $\phi(\boldsymbol{x}_i) = v_j$ y $i \in nd_\phi$.*

Ejemplo 40 *En la Figura 4.11 podemos observar la representación como IDM del potencial $\phi_1(X_1, X_2, X_3)$ definido en el Ejemplo 15.*

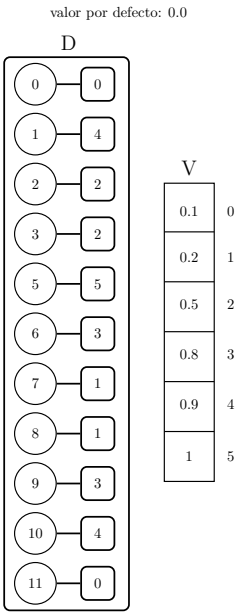


Fig. 4.11: Potencial $\phi_1(X_1, X_2, X_3)$ como IDM.

El diccionario D se representa en la parte de la izquierda de Fig. 4.11 y el vector de valores V se encuentra en la parte de la derecha. Las claves en D se dibujan como círculos y dan acceso a los índices de V (cajas asociadas a las claves).

□

Algoritmo 4 (Acceso a índice en IDM) *Dado un potencial IDM$_\phi$, el algoritmo para acceder al valor correspondiente a un índice dado l se describe en el siguiente Algoritmo 4.*

Algoritmo 4 Acceso a índice l en IDM$_\phi$

```
1: funcion acceso
2:     resultado = 0.0                    ▷ defince el valor por defecto como "resultado"
3:     entrada(< l, j >) ← D(l)           ▷ busca l en el diccionario
4:     si entrada != nulo entonces:       ▷ el diccionario contiene l como clave
5:         resultado ← V(j)
6:     fin condicion
7:     devolver resultado
8: fin funcion
```

Asumamos que el índice que buscamos en ϕ_1 es el 6, puesto que es una clave válida, la entrada $< 6, 3 >$ será la que se recupere. El valor a devolver se almacena en $V(3) = 0.8$. Si se busca el índice 4, entonces se devolverá 0.0 (ya que es el índice que no está presente en D).

Proposición 4.2.6 ***(Espacio de memoria para un IDM que representa un potencial)*** *Sea IDM$_\phi$ la representación dirigida por índices con mapas del potencial $\phi(\boldsymbol{X}), con|\boldsymbol{X}| = N$. Asumamos que d representa el número de valores diferentes en el potencial (sin contar el valor seleccionado por defecto) y p es el número de índices que almacenan valores distintos del seleccionado por defecto. La cantidad de memoria requerida puede estimarse mediante la siguiente ecuación:*

$$memory(\boldsymbol{IDM}_\phi) = N \cdot s_v + s_f + s_{dict} + s_{arr} + d \cdot s_f + 2 \cdot p \cdot s_i \tag{4.8}$$

Los términos de la Ecuación 4.8 representan tamaños de: variables, valor por defecto, diccionario, vector de valores, valores distintos e índices en las entradas del diccionario.

Ejemplo 41 *Sea IDM$_{\phi_1}$ el potencial IDM del Ejemplo 40 con 6 valores distintos que almacenar y 0.0 de valor por defecto. Por tanto, el vector de valores contiene 6 elementos y el diccionario contiene 11 entradas. El costo computacional puede calcularse como:*

$$memory(\mathit{IDM}_\phi) = 3s_v + s_f + s_{dict} + s_{arr} + 6 \cdot s_f + 2 \cdot 11 \cdot s_i$$

Considerando los tamaños específicos detallados en la Sección 4.2.1, el tamaño de memoria total es de 374 bytes. Recordamos que los tamaños de memoria para las representaciones 1DA, PT, PPT, VDG, VDI e IDP son 262, 1000, 910, 438, 410 y 326 bytes respectivamente.

□

4.2.2 Experimentación sobre ahorro de espacio

Ejemplo de caso extremo

Consideremos un ejemplo de uso de las estructuras que representan un potencial con características extremas, ya que sólo contiene tres valores de probabilidad distintos (0.0, 0.5, y 1.0) y muchas repeticiones de uno de esos valores (0.0, el elegido por defecto con una ocurrencia del 70%). Este potencial tiene 1024 valores posibles, con 5 variables aleatorias en su dominio de 4 estados cada una. Los valores se generan aleatoriamente. Hemos considerado 10 potenciales aleatorios con estas características para obtener una idea más fiable del comportamiento de las representaciones. Compararemos los espacios de memoria para las tres representaciones alternativas de potenciales 1DA, PT y PPT; y para las cuatro que acabamos de definir en esta memoria: VDG, VDI, IDP e IDM. Las representaciones $1DA$ y PT son independientes de los valores específicos y necesitan tamaños de memoria de 8574 y 53816 respectivamente. Los resultados obtenidos para el resto de representaciones se muestran en la Tabla 4.1; pese a que los tamaños de 1DA y PT se mantienen constantes, hemos decidido añadirlos para facilitar la comparación visual. El valor de la representación menor para cada potencial (que resulta ser el de VDI en todos los casos) se podrá observar en color rojo.

En todos los potenciales aleatorios, el número de valores distintos del cero (el fijado por defecto) varían de 156 a 222 (observable en la última fila de la Tabla 4.1) y la mayor secuencia de valores repetidos es de 2. Pese a que podrían obtenerse mejores resultados en los casos en los que la secuencia de repetición fuese mayor (al menos con la representación VDG), los resultados muestran grandes ahorros de espacio en memoria en comparación con las representaciones PT y PPT, para los que la opción VDI supone un ahorro de entre 96% y 98%. Como dijimos, la representación 1DA es bastante eficiente, aún así, todas las estructuras VBPs requieren un menor espacio en memoria, proporcionando ahorros de entre 84% y 87% con respecto a 1DA.

iteración	1	2	3	4	5	6	7	8	9	10
$1DA$	8574	8574	8574	8574	8574	8574	8574	8574	8574	8574
PT	53816	53816	53816	53816	53816	53816	53816	53816	53816	53816
PPT	39090	38332	37658	38396	40946	38910	42610	38120	42706	38544
VDG	1822	1814	1670	1774	1934	1854	1990	1734	2030	1694
VDI	1206	1210	1110	1190	1310	1218	1346	1166	1374	1194
IPD	1862	1870	1670	1830	2070	1886	2142	1782	2198	1838
IDM	1910	1918	1718	1878	2118	1934	2190	1830	2246	1886
valores	180	166	156	176	206	183	215	170	222	177

Tab. 4.1: Tamaños de memoria para 10 representaciones de potenciales aleatorios con valores extremos.

4.2.3 Operaciones con VBPs

Ya vimos en la introducción a potenciales (Sec. 2.4.1) las definiciones de las operaciones de marginalización y combinación. Recordemos ambos conceptos.

Si ϕ es un potencial definido sobre un conjunto de variables aleatorias $\mathbf{X}$; y dado $\mathbf{Z} \subset \mathbf{X}$, entones la marginalización de ϕ a $\mathbf{Z}$ se computa mediante la siguiente fórmula:

$$\phi^{\downarrow \mathbf{Z}}(\mathbf{x}) = \sum_{\mathbf{x}^{\downarrow \mathbf{Z}} = \mathbf{z}} \phi(\mathbf{z}), \quad \forall \mathbf{z} \in \Omega_{\mathbf{Z}} \tag{4.9}$$

donde $\mathbf{x}^{\downarrow \mathbf{Z}}$ denota la proyección de configuraciones $\mathbf{x}$ sobre $\mathbf{Z}$.

Dados dos potenciales $\phi_1(\mathbf{X})$ y $\phi_2(\mathbf{Y})$, la combinación de ambos es el potencial denotado como $\phi_1 \otimes \phi_2$, que queda definido sobre $\mathbf{Z} = \mathbf{X} \cup \mathbf{Y}$ y se calcula mediante la ecuación:

$$\phi_1 \otimes \phi_2(\mathbf{z}) = \phi_1(\mathbf{z}^{\downarrow \mathbf{X}}) \cdot \phi_2(\mathbf{z}^{\downarrow \mathbf{Y}}), \quad \forall \mathbf{z} \in \mathbf{X} \cup \mathbf{Y} \tag{4.10}$$

Sería ideal disponer de operaciones que saquen provecho de las ventajas que las nuevas representaciones VBPs ofrecen, ya que el objetivo de parte de nuestro estudio consiste no sólo en definir estructuras más compactas, sino investigar cómo se comportan en las tareas de inferencia con RBs. Procedemos a introducir una primera aproximación de cómo podrían realizarse las operaciones básicas de marginalización y combinación sobre las estructuras VBPs que acabamos de definir. Hemos desarrollado algoritmos simples y directos para ambas operaciones (marginalización Alg. 5 y combinación Alg. 6). Estos algoritmos se basan en el acceso a los valores de los

potenciales (operaciones definidas en los Algoritmos 1, 2, 3 y 4), por lo que la operación de acceso es extremadamente importante. Puesto que la correspondencia entre índices y configuraciones es uno a uno, podemos referirnos indiferentemente a ellos (e.g., el índice l en $\mathbf{Z}$ corresponde con la configuración $\mathbf{z}_l$).

Algoritmo 5 (Marginalización en VBPs) *Sea $VBP_\phi(\boldsymbol{X})$, un potencial expresado en su alternativa VBP definido sobre el conjunto de variables $\boldsymbol{X}$, y sea $Y \in \boldsymbol{X}$. El método a seguir para marginalizar Y de VBP_ϕ se describe en el Algoritmo 5. Notamos que el mismo algoritmo puede usarse para todas las alternativas VBP.*

Algoritmo 5 Marginalización de Y de $VBP_\phi(\mathbf{X})$

```
1:  funcion marginalizar
2:      Z = X \ Y                                  ▷ crea el dominio del resultado
3:      crea VBP_{φ_r}(Z)                          ▷ crea potencial resultante vacío
4:      para cada l = {0...k}, k = |Ω_Z| − 1 hacer:   ▷ bucle sobre los índices VBP_φ(Z)
5:          para cada y ∈ Ω_Y hacer:
6:              x_{ly} ← z_l^{↑Y=y}                ▷ obtenemos configuración x_{ly} compatible con z_l
7:              v_{ly} ← VBP_φ(x_{ly})             ▷ valor en VBP_φ(X) para x_{ly}
8:          fin bucle
9:          VBP_{φ_r}(z_l) ← Σ_{y∈Ω_y} v_{ly}
10:     fin bucle
11:     devolver VBP_{φ_r}(Z)
12: fin funcion
```

El Algoritmo 5 elimina la variable Y del potencial $VBP_\phi(\mathbf{X})$. En las líneas 2 y 3, el dominio del potencial final $\mathbf{Z} = \mathbf{X} \setminus Y$ se usa para crear el potencial final, que será inicialmente vacío. La línea 4 itera sobre los índices $VBP_{\phi_r}(\mathbf{Z})$. Asumamos que l se corresponde con una configuración específica de variables en $\mathbf{Z}$ (denotada por $\mathbf{z}_l$). Los índices compatibles $\mathbf{x}_l$ se refieren a las configuraciones producidas al completar $\mathbf{z}_l$ con los valores posibles de Y. Esta operación es la que vemos en la descripción del Algoritmo 5 como $\mathbf{z}_l^{\uparrow Y}$. El bucle interno (líneas 5 a 8) iteran sobre los valores de Y. La suma de $\mathbf{v}_{ly}$ valores se asigna al potencial resultante (línea 9).

Algoritmo 6 (Combinación de VBPs) *Dados dos potenciales definidos sobre los conjuntos de variables $\boldsymbol{X}$ e $\boldsymbol{Y}$ respectivamente, expresados en sus alternativas VBP, $VBP_{\phi_1}(\boldsymbol{X})$ y $VBP_{\phi_2}(\boldsymbol{Y})$, el método para combinar ambos potenciales se presenta en el Algoritmo 6. De la misma manera que ocurre con el Algoritmo 5, el método que definimos a continuación puede aplicarse para cualquier alternativa VBP.*

Algoritmo 6 Combinación de $VBP_{\phi_1}(\mathbf{X})$ y $VBP_{\phi_2}(\mathbf{Y})$

1: **funcion** combinar
2: $\quad\mathbf{Z} = \mathbf{X} \cup \mathbf{Y}$ ▷ crea el dominio del resultado
3: $\quad$crear $VBP_{\phi_r}(\mathbf{Z})$ ▷ crea potencial resultante vacío
4: $\quad$**para** cada $l = \{0 \dots k\}, k = |\Omega_{\mathbf{Z}}| - 1$ **hacer:** ▷ bucle sobre los índices $VBP_{\phi}(\mathbf{Z})$
5: $\quad\quad v_l \leftarrow 0.0$
6: $\quad\quad \mathbf{x}_l \leftarrow \mathbf{z}_l^{\downarrow \mathbf{X}}$ ▷ proyección del índice $\mathbf{z}_l$ sobre $\mathbf{X}$
7: $\quad\quad v_1 \leftarrow VBP_{\phi_1}(\mathbf{x}_l)$ ▷ valor en $VBP_{\phi_1}(\mathbf{X})$ para $\mathbf{x}_l$
8: $\quad\quad$**si** $v_1 \neq 0.0$ (valor por defecto) **entonces:**
9: $\quad\quad\quad \mathbf{y}_l \leftarrow \mathbf{z}_l^{\downarrow \mathbf{Y}}$ ▷ proyección del índice $\mathbf{z}_l$ sobre $\mathbf{Y}$
10: $\quad\quad\quad v_2 \leftarrow VBP_{\phi_2}(\mathbf{y}_l)$ ▷ valor en $VBP_{\phi_2}(\mathbf{Y})$ para $\mathbf{y}_l$
11: $\quad\quad\quad$**si** $v_2 \neq 0.0$ **entonces:**
12: $\quad\quad\quad\quad v_l = v_1 \cdot v_2$
13: $\quad\quad\quad$**fin condicion**
14: $\quad\quad$**fin condicion**
15: $\quad\quad VBP_{\phi_r}(\mathbf{z}_l) \leftarrow v_l$
16: $\quad$**fin bucle**
17: $\quad$devolver $VBP_{\phi_r}(\mathbf{Z})$
18: **fin funcion**

El Algoritmo 6 combina dos potenciales, $VBP_{\phi_1}(\mathbf{X})$ y $VBP_{\phi_2}(\mathbf{Y})$. La línea 2 genera el dominio $\mathbf{Z}$ como $\mathbf{Z} = \mathbf{X} \cup \mathbf{Y}$. Este es el dominio del potencial $VBP_{\phi_r}(\mathbf{Z})$ que deberá ser devuelto. La línea 4 itera sobre los índices $VBP_{\phi_r}(\mathbf{Z})$. Sea l el índice a considerar (que se corresponde con una configuración dada $\mathbf{z}_l$). El valor que se asignará a l se inicializa a 0.0 (línea 5). La configuración $\mathbf{z}_l$ necesitará ser proyectada en $VBP_{\phi_1}(\mathbf{X})$ (línea 6) y $VBP_{\phi_2}(\mathbf{Y})$ (línea 9). Estas operaciones se notan por $\mathbf{z}_l^{\downarrow \mathbf{X}}$ y $\mathbf{z}_l^{\downarrow \mathbf{Y}}$, y comprenden borrar de $\mathbf{z}_l$ aquellos valores de las variables que no pertenecen a $\mathbf{X}$ e $\mathbf{Y}$ respectivamente. El índice de la configuración $\mathbf{x}_l$ se usa para obtener v_1 (línea 7). Si v_1 es 0.0, entonces es seguro que $v_l = 0.0$, por tanto, no se necesitan más operaciones. En otro caso, será también necesario acceder a $VBP_{\phi_2}(\mathbf{y}_l)$ (línea 10). Finalmente, v_l se asigna a $VBP_{\phi_r}(\mathbf{Z})$ en la línea 15.

Evaluación empírica

Para evaluar las capacidades de las representaciones VBPs en comparación con las anteriores alternativas de representación de potenciales (1DA, PT y PPT), hemos utilizado dos conjuntos de

redes Bayesianas. El primero de los conjuntos es parte del repositorio de *bnlearn* [174, 175] y el segundo de las competiciones *UAI* ([196, 197]). Los experimentos se organizan en tres bloques diferentes: comparación de tamaños de memoria (Sec. 4.2.4), tiempo de acceso (Sec. 4.2.5) y tiempo de cálculo de distribuciones posteriores utilizando el algoritmo de eliminación de variables (VE) [62, 182, 211] (Sec. 4.2.6).

Las representaciones que se han comparado en los experimentos son:

- 1DA, PT, PPT, VDG, VDI, IDP e IDM para tamaños de memoria y comparaciones en tiempo de acceso sobre las redes de ambos recursos *bnlearn* y *UAI*.
- 1DA, PT, VDI e IDM para cálculos *a posteriori* con las redes de *UAI*.

4.2.4 Estudio de las características de las redes Bayesianas

En las siguientes Tabla 4.2 y Tabla 4.3 hemos recopilado alguna información básica sobre las redes Bayesianas usadas en la experimentación, provenientes de *bnlearn* y de *UAI* respectivamente, comprendiendo: nombre de la red, número de nodos, número de arcos, números mínimo, medio y máximo de estados de las variables, y número completo de parámetros para cuantificar la incertidumbre en las redes. Las redes se han ordenado en cada tabla de manera ascendente por número de parámetros. Podemos observar que las redes del conjunto *UAI* requieren más parámetros que las de *bnlearn*.

Comparación de tamaños de memoria

Vamos a proceder a comparar los tamaños de memoria necesarios para representar las RBs que acabamos de introducir. Para ello, convertimos los potenciales de las RBs a las representaciones alternativas VDG, VDI, IDP e IDM, y hacemos la comparación de espacio con respecto a 1DA, PT y PPT. El tamaño de memoria que utiliza 1DA se toma como referencia y no aparece en la tabla. Dada una determinada red, sea m_{1DA} el espacio de memoria para la representación 1DA y m_{rep} el tamaño de memoria para una representación diferente. El valor s incluido en las celdas de la tabla y que representa el porcentaje de ganancia (o pérdida) de rep es calculado como:

$$s = \frac{m_{rep} * 100}{m_{1DA}} - 100 \tag{4.11}$$

red	nodos	arcos	mín. est.	med. est.	máx. est.	parámetros
cancer	5	4	2	2	2	10
asia	8	8	2	2	2	18
survey	6	6	2	2.33	3	21
sachs	11	17	3	3	3	178
child	20	25	2	3	6	230
alarm	37	46	2	2.83	4	509
win95pts	76	112	2	2	2	574
insurance	27	52	2	3.29	5	1008
andes	223	338	2	2	2	1157
hepar2	70	123	2	2.31	4	1453
hailfinder	56	66	2	3.98	11	2656
pigs	441	592	3	3	3	5618
water	32	66	3	3.625	4	10083
link	724	1125	2	2.53	4	14211
munin1	186	273	2	5.33	21	15622
munin2	1003	1244	2	5.36	21	69431
munin3	1041	1306	2	5.38	21	71059
pathfinder	109	195	2	4.11	63	72079
munin4	1038	1388	2	5.44	21	80352
munin	1041	1397	2	5.43	21	80592
barley	48	84	2	8.77	67	114005
diabetes	413	602	3	11.34	21	429409
mildew	35	46	3	17.6	100	540150

Tab. 4.2: Características de las redes Bayesianas estudiadas del repositorio *bnlearn*

red	nodos	arcos	mín. est.	med. est.	máx. est.	parámetros
BN_76	2155	3686	2	7.01	36	627298
BN_87	422	867	2	2	2	933776
BN_29	24	30	10	10	10	1132080
BN_125	50	375	2	2	2	2117680
BN_115	50	375	2	2	2	2285616
BN_119	50	375	2	2	2	2410544
BN_121	50	375	2	2	2	2564144
BN_123	50	375	2	2	2	3249200
BN_27	3025	7040	3	6	10	3698565
BN_117	50	375	2	2	2	4003888
BN_22	2425	4239	2	18.743	91	4073904
BN_111	50	375	2	2	2	4238512
BN_109	50	375	2	2	2	4581936
BN_113	50	375	2	2	2	4669488
BN_20	2483	5272	2	18.92	91	5009364
BN_107	50	375	2	2	2	5154864
BN_105	50	375	2	2	2	6431792

Tab. 4.3: Características de las redes Bayesianas estudiadas de las competiciones *UAI*.

Esto quiere decir que un valor negativo para s indica que rep requiere menos espacio de memoria que 1DA. Por el contrario, los valores positivos indican un mayor consumo de memoria.

Tamaños de memoria para las redes Bayesianas de *bnlearn*

Los resultados para este conjunto de redes se presentan en la Fig. 4.12.

Sobre los resultados podemos comentar:

- Las representaciones PTs y PPTs siempre van a requerir un mayor espacio de memoria que su respectiva 1DA. Ambas representaciones ocupan tamaños de memoria similares, a excepción de la red **win95pts**. Esta red en concreto presenta muchos potenciales con valores repetidos para los que la operación de poda reduce el número de nodos hojas de manera substancial, y como consecuencia el espacio de memoria necesario disminuye.
- Para la mayoría de las redes, la representación más competitiva es IDP. Para el caso de redes con un número de parámetros menor a 8427 (desde la red **cancer** hasta la **hailfinder**), IDP necesita más espacio de memoria que 1DA, pero para el resto de casos, IDP ofrece un rango de ahorro de memoria entre el -6.31% y el -90.25%. La red **barley** es una

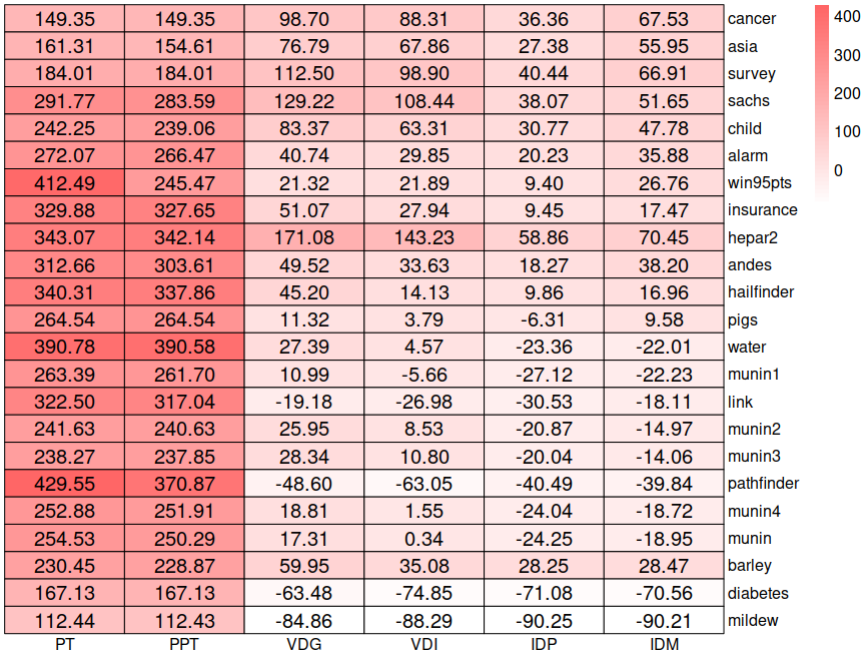


Fig. 4.12: Tamaños de memoria para las redes de *bnlearn*.

excepción, ya que los potenciales en esta red tienen secuencia de repetición de pocos índices. Por ejemplo, el potencial de la variable **jordn** tiene 4752 valores posibles, pero sólo 4 son diferentes, por lo que podría parecer lógico que se produjera ahorro de energía. Sin embargo, las secuencias se producen de manera que PPT no puede sacar provecho. Por esta razón, VDG necesitará un gran número de granos para quedar determinada. Además, los potenciales tienen pocos ceros, por lo que hay muchos índices que almacenar. En estos casos, sería apropiado modificar el valor por defecto para que fuese el más repetido, en lugar de considerar 0.0 sistemáticamente, pero esto complicaría las operaciones de combinación y marginalización.

- Los ahorros más importantes se producen en las redes **diabetes** y **mildew**. En el caso de estas dos redes hay muchos potenciales con alto índice de repetición de valores. En **diabetes** hay 25 potenciales con 7056 parámetros, pero sólo 44 valores distintos. Ocurre de manera similar en **mildew**, donde por ejemplo encontramos un potencial con 39040 valores de probabilidad con 1756 de ellos distintos; y otro potencial con 201300 parámetros y 4508 valores distintos. Para estos potenciales, los valores repetidos no pueden unirse en las representaciones PPTs.

- VDI es la mejor representación para **pathfinder** y **diabetes**. Esto puede explicarse por el bajo número de valores distintos en comparación con el número de parámetros. En **pathfinder**, un potencial con 8064 parámetros necesita sólo 29 valores distintos.

Tamaños de memoria para las redes Bayesianas de las competiciones *UAI*

Los resultados para este conjunto de redes se presentan en la Fig. 4.13.

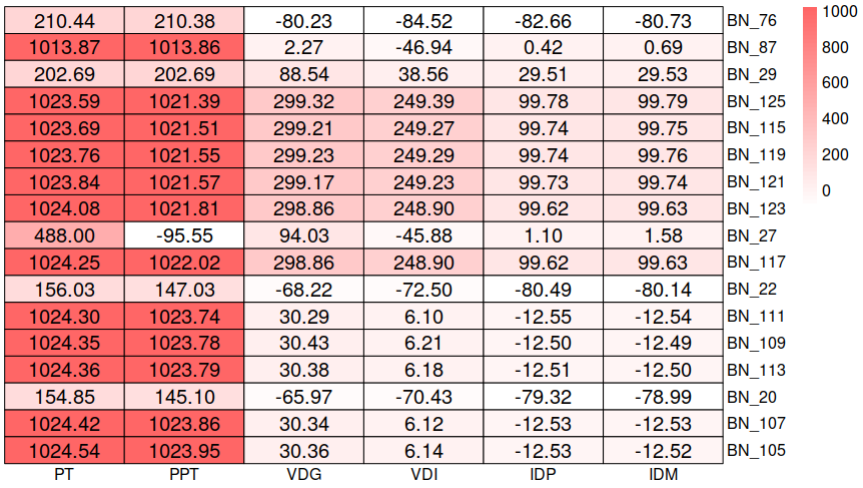


Fig. 4.13: Tamaños de memoria para las redes *UAI*.

Sobre los resultados podemos comentar:

- Los porcentajes para las representaciones alternativas PTs y PPTs están siempre por encima del 200%, a excepción de **BN_27**. Esta red presenta una diferencia importante entre PT (488%) y PPT (-95.55%). De hecho, PPT es la representación que mejor resultado produce, pero VDG ofrece un ahorro de memoria similar. Esta red tiene 1005 potenciales con 3645 parámetros, pero sólo un valor (de hecho, PPTs sólo tiene un nodo hoja).

- IDP es la mejor representación para la mayoría de las redes, con ahorros significativos para **BN_76**, **BN_22**, y **BN_20**; y ahorros moderados para **BN_111**, **BN_109**, **BN_113**, **BN_107**, y **BN_106**. Para el resto de redes, se presentan como la mejor alternativa, tras 1DA.

- Todas las representaciones que hemos presentado ofrecen ser una alternativa competitiva frente a PTs y PPTs. En general, los ahorros más significativos se producen cuando el

número de parámetros es alto, donde las representaciones eficientes son primordiales para aplicar algoritmos de inferencia.

4.2.5 Tiempo de acceso

Al tratar con modelos complejos, se asume que los modelos alternativos implicarán un mayor tiempo de cómputo (en compensación por el ahorro de memoria). Puesto que las representaciones que necesitan largos tiempos de cómputo son poco prácticas, la eficiencia de acceso a los valores de los potenciales es un requisito indispensable. Por esta razón, hemos basamos parte de nuestra experimentación en probar qué tal se comportan las alternativas VBP en este aspecto. Hemos seleccionado de manera aleatoria 10000 pares (potencial, índice), utilizando el mismo conjunto para todas las representaciones de cada red. En las Tablas 4.14 y 4.15 se muestran los resultados con tiempos en milisegundos. Al final de cada columna podemos ver con qué representación (1DA, PT, ...) se corresponde.

Para hacer una estimación fiable de los tiempos de acceso, hemos utilizado la biblioteca **Scalameter** [172]. Esta herramienta permite configurar los experimentos de medida de tiempo asegurando que la máquina alcance un estado estacionario (después de una fase de calentamiento); después de eso, repite varias veces el procedimiento de interés y finalmente informa del tiempo promedio.

Tiempos de acceso en las redes de *bnlearn*

En la siguiente Fig. 4.14 presentamos los tiempos de acceso obtenidos para las redes pertenecientes al repositorio *bnlearn*. Estos resultados muestran que la alternativa IDM es la más competitiva, con tiempos similares a 1DA. Los tiempos de acceso para el resto de representaciones son en la mayoría de los casos menores que los requeridos para las representaciones PT y PPT, a excepción de **barley** y **mildew**. Para estas redes, el ahorro de espacio de memoria implica una estructura más compleja que requiere tiempos de acceso más prolongados.

Tiempos de acceso en las redes de *UAI*

En la Fig. 4.15 podremos observar los tiempos de acceso obtenidos para las redes *UAI*. En este conjunto, IDM vuelve a ser la representación más ventajosa, de nuevo con tiempos similares a 1DA. En estas redes complejas, para las que VBPs ofrecen reducciones significativas en el

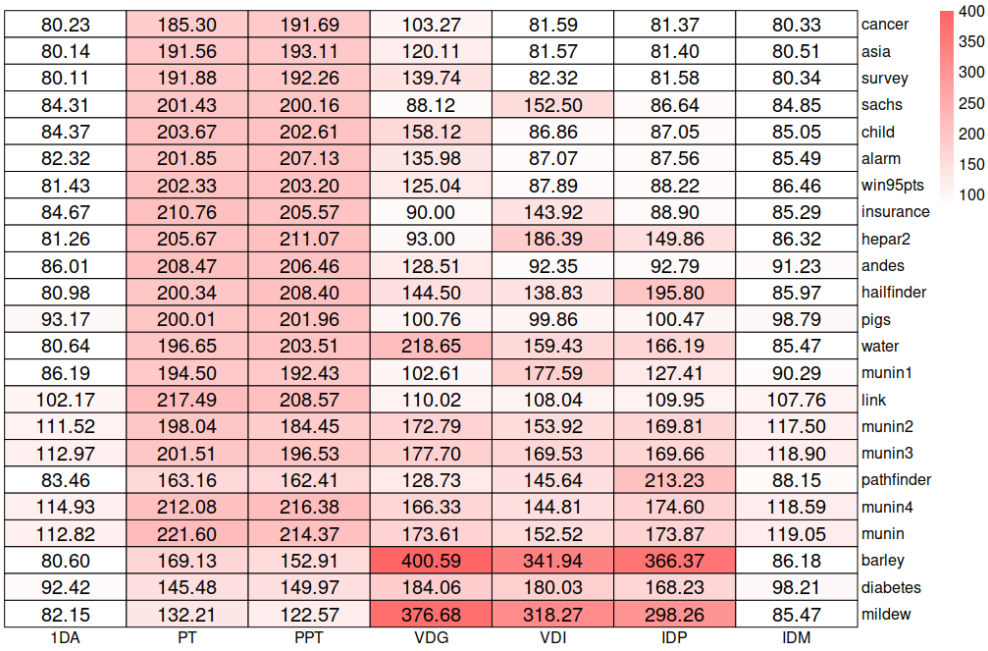


Fig. 4.14: Tiempos de acceso para las redes de *bnlearn*.

espacio de memoria, las estructuras VDG, VDI e IDP necesitan un tiempo de acceso mayor. Este hecho es especialmente relevante para las alternativas cuya búsqueda es dirigida por valor (VDG y VDI).

4.2.6 Tiempo de cálculo de distribuciones *a posteriori* utilizando el algoritmo VE

A continuación, estudiaremos los tiempos de cómputo requeridos para obtener la distribución *a posteriori* de 10 variables seleccionadas aleatoriamente (usando el algoritmo de eliminación de variables (VE)) de cada red *bnlearn*, y utilizando los algoritmos de marginación y combinación descritos anteriormente. Los experimentos se han limitado a las redes *bnlearn* porque para la mayoría de las redes *UAI*, los cálculos con 1DA, PT y PPT exceden la capacidad de memoria de la computadora utilizada para este trabajo experimental.

En cualquier caso, es importante destacar que el objetivo de estos experimentos es tener una idea global del comportamiento de las estructuras VBP. Una línea de trabajo futura será la realización de implementaciones de las operaciones de marginación y combinación, teniendo en cuenta las propiedades especiales de cada representación.

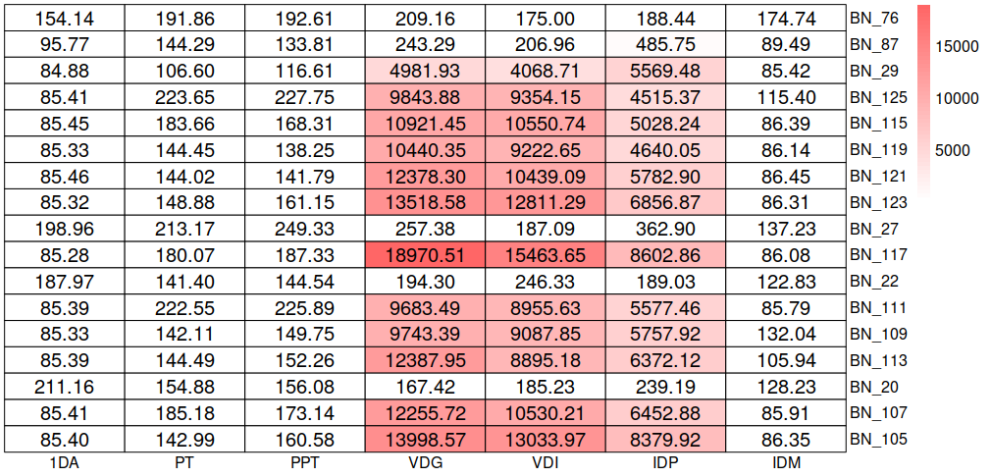


Fig. 4.15: Tiempos de acceso para las redes de *UAI*.

Hemos seleccionado una alternativa para cada categoría: VDI para la aproximación basada en valores e IDM para el enfoque basado en índices. Estas dos representaciones muestran la mejor compensación entre el uso de la memoria y los tiempos de acceso dentro de su categoría. Se comparan con 1DA y PT (cuando se usan árboles, no hay mucha diferencia entre PT y PPT en general). Hemos utilizado la biblioteca **Scalameter** para medir los tiempos de cálculo. Los resultados de esta sección se presentan en la Fig. 4.16, donde en la última fila de cada columna observamos el tipo de representación a la que se asocian los resultados.

A la vista de estos resultados, se observa que la inferencia con PT muestra la mayor eficiencia. Esto se debe a las implementaciones específicas de las operaciones de marginación y combinación en PTs [170], y el hecho de que se trata de una implementación recursiva. Cabe señalar que para algunas de las redes *UAI*, la ejecución del algoritmo produce un error de desbordamiento de pila al generar PTs con muchas variables. En estos casos, la evaluación con 1DA también falla produciendo errores de falta de memoria.

También se observa que, en la mayoría de los casos, los tiempos para IDM son similares a los de 1DA. Esto es notable y sugiere que implementaciones más refinadas de las operaciones de marginación y combinación, ajustadas a su estructura, mejorarán los tiempos de cálculo actuales. Las implementaciones más eficientes de estas operaciones deberían tratar de evitar iterar sobre todos los índices del potencial resultante, utilizando en su lugar sólo aquellos que están almacenados. Para algunas redes, esto puede ofrecer reducciones de tiempo significativas.

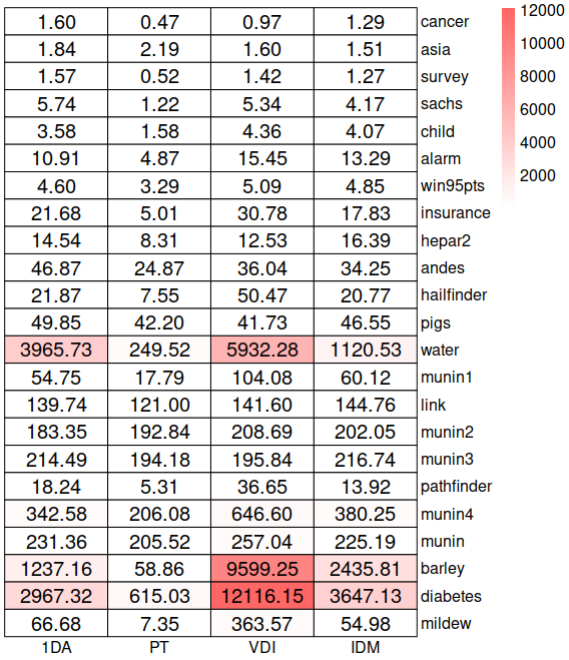


Fig. 4.16: Tiempos de EV para las redes de *bnlearn*

4.3 Aproximación de VBPs

Una vez definidas las estructuras VBPs y a la vista de la evaluación empírica realizada, nos preguntamos cómo sería posible modificar estas estructuras de manera que permitan evaluar modelos más complejos. En esta sección presentamos el siguiente paso, el de la aproximación de las mismas. Ya vimos en la Sección 4.1, cómo se realizaba la operación de poda (*pruning*) en árboles de probabilidad (PTs), dando lugar a los llamados árboles de probabilidad podados (PPT); así como las ventajas que esta operación ofrecía en cuanto a ahorro de espacio de memoria. Por otro lado, la operación de aproximación de PTs es computacionalmente muy costosa, ya que es necesario determinar el grado de información de cada variable para asegurar que las más informativas aparezcan lo más cerca posible de la raíz. Esto asegura que los valores almacenados en las hojas del árbol sean los más similares y, por lo tanto, la pérdida de información sea menor cuando se reemplazan varios valores por su valor promedio.

En esta sección vamos a definir un procedimiento muy sencillo para aproximar VBPs, junto con su justificación teórica. De las cuatro alternativas VBPs que hemos presentado (VDG, VDI, IDP e IDM), en esta parte de la memoria nos centraremos en potenciales dirigidos por valores con índices (*value-driven with indices* - VDI) y definiremos el algoritmo sobre ellos, aunque de manera análoga podría definirse sobre cualquiera de las otras tres alternativas. Además de

la definición del algoritmo, posteriormente procederemos a aplicarlo a un conjunto de redes Bayesianas, para estimar así el comportamiento del mismo y dar una idea mediante ejemplos reales de cómo otros podrían beneficiarse de las VBPs de manera similar.

4.3.1 Idea intuitiva

Recordamos, que intuitivamente podemos visualizar el conjunto de valores como algo similar a un conjunto de Venn, en el que los valores de probabilidad aparecen convenientemente ordenados y situados junto con los índices relacionados con dicho valor, tal y como ya presentamos en la Fig. 4.7. De este modo, al aproximar las estructuras VBPs, uniremos esos valores de probabilidad para conseguir una estructura de menor tamaño.

El proceso de aproximación será iterativo, donde en cada iteración se unirán los dos valores que supongan la menor pérdida de información posible. Así, en cada paso, obtendremos un nuevo VBP aproximado que no contendrá los dos valores de probabilidad a aproximar y en su lugar aparecerá un único valor "unión". Este nodo aproximado contendrá como valor de probabilidad a la media ponderada de los dos iniciales; y como índices asociados, a la unión de los dos conjuntos. Mostramos en la siguiente Fig. 4.17 cómo sucedería con la primera iteración para el ejemplo de potencial VBP que venimos estudiando. En la figura de la izquierda tenemos el diagrama que correspondería al potencial original y en la parte derecha podemos ver cómo se combinan los valores 0.9 y 1.0 para dar lugar a uno nuevo $0.9\widehat{3}$ (resultado de la media ponderada de los anteriores $0.9\widehat{3} = \frac{0.9\times2+1.0\times1}{2+1}$), al que se le asignan los índices de los dos nodos unidos $(1, 5, 10)$. El resto de la estructura permanece como se encontraba al inicio de la iteración.

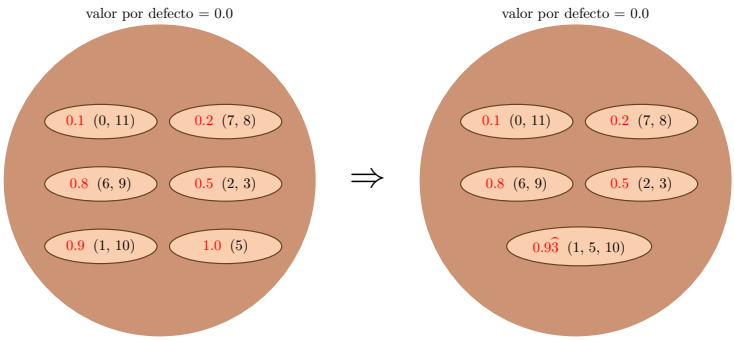


Fig. 4.17: Idea visual del potencial $\phi_1(X_1, X_2, X_3)$ agrupando valores de probabilidad, junto con lo que resultaría en la primera de las iteraciones al aplicar el algoritmo de aproximación sobre el mismo potencial.

4.3.2 Algoritmo

El método para aproximar un potencial dado $\phi : \Omega_X \to \mathbb{R}_0^+$ va a ser explicado considerando potenciales representados como VDI, por ser una estructura más simple, pero la aplicación práctica seguiría la misma idea para cualquier otra alternativa (como IDP, por ejemplo). Denotamos el potencial en su forma VBP para ser aproximado como V. Supongamos que almacena n valores diferentes, indicados como $v_1, v_2, \ldots, v_n$ en orden creciente de valor. Como mencionamos en nuestra descripción de VBPs, cada valor diferente v_i sólo se almacena una vez, pero la información sobre el conjunto de índices (o configuraciones) correspondientes a cada v_i queda almacenada. Notamos como $\mathbf{S}_i$ al conjunto de índices de v_i, con $n_i = |\mathbf{S}_i|$ (lo que significa que n_i es el número de configuraciones en $\mathbf{S}_i$).

Definición 4.3.1 ***(Operación de reducción)*** *El sistema básico de aproximación consiste en reducir el número de valores aplicando una operación de* ***reducción****. Esta operación reemplaza dos valores consecutivos, v_i y v_{i+1}, por su media ponderada. La reducción generalmente se puede describir usando la siguiente notación:*

- *v_a y v_b serán los valores a reducir, donde $\boldsymbol{S}_a$ y $\boldsymbol{S}_b$ son respectivamente los conjuntos de índices asociados. Notamos además $n_a = |\boldsymbol{S}_a|$ y $n_b = |\boldsymbol{S}_b|$.*
- *v_r es el nuevo valor que reemplaza a v_a y v_b, con $\boldsymbol{S}_r = \boldsymbol{S}_a \cup \boldsymbol{S}_b$ y $n_r = |\boldsymbol{S}_r|$. Este valor puede computarse como:*
$$v_r = \frac{n_a \cdot v_a + n_b \cdot v_b}{n_a + n_b}.$$
- *Es importante observar que esta operación no modifica la suma total de los valores de los potenciales. De hecho, si ϕ es el potencial original y V es el resultado de las reducciones repetitivas, entonces $sum(\phi) = sum(V)$.*

Como consecuencia, al final de esta operación, se reduce el número de valores. El algoritmo completo empleado para aproximar V_ϕ se puede describir intuitivamente de la siguiente manera:

1 Hay varias alternativas de aproximación diferentes, lo que se traduce en estructuras candidatas de las que una se convertirá en la elegida. Como hay n valores diferentes, habrá $n-1$ estructuras candidatas, que se generan al reducir cada par de valores consecutivos. Iterando desde $i = 1$ a $i = n - 1$:

1.1 Consideramos dos valores sucesivos: v_i y v_{i+1}. Se obtiene la estructura candidata reduciendo ambos valores como se explicó anteriormente en la Definición 4.3.1. El resultado de esta operación será V_i.

1.2 Calculamos la distancia de Kullback–Leibler entre el potencial original V y V_i. Este valor se denota por $D(V, V_i)$.

2 Se selecciona la estructura candidata $V_m = \arg\min_{j=1...n-1} D(V, V_j)$.

3 Se repiten los pasos anteriores hasta que se cumple el criterio de parada seleccionado.

Antes de presentar una descripción detallada del algoritmo, queremos incluir una serie de consideraciones:

- Es evidente que no es necesario construir las estructuras candidatas, sino sólo evaluar la pérdida de información de las operaciones de *reducción* correspondientes.

- La distancia de Kullback-Leibler entre una estructura candidata V_i y la original se pueden calcular teniendo en cuenta sólo aquellos valores e índices involucrados en la *reducción*, y la medida a calcular es de hecho la pérdida de información producida por esta operación. La forma de calcular esta medida se explicará a continuación.

- Un posible criterio de parada (este es el utilizado en el trabajo experimental, aunque se podrían considerar otros) consiste en establecer un umbral de pérdida de información global, t_l. Por tanto, el procedimiento de reducción de valores consecutivos continuará mientras la suma de pérdidas de información no alcance el umbral t_l.

4.3.3 Marco teórico

Sea $\phi(\mathbf{X})$ un potencial dado donde $V(\mathbf{X})$ representa su versión VBP. El grado de aproximación entre ambas representaciones se medirá mediante la *distancia de Kullback–Leibler* [117] calculada sobre los potenciales normalizados correspondientes ($\overline{\phi}$ y $\overline{V}$):

$$D(\phi, V) = \sum_{\mathbf{x}\in\Omega_X} \overline{\phi(\mathbf{x})} log \frac{\overline{\phi(\mathbf{x})}}{\overline{V(\mathbf{x})}}. \tag{4.12}$$

La distancia es un número real no negativo que sólo sería igual a cero si V proporciona una representación exacta de ϕ. Como explicamos anteriormente, la operación clave para aproximar ϕ representado como V es la de *reducción*, que se describió en la Definición 4.3.1.

Algoritmo 7 Aproximación de un potencial ϕ representado como V (VBP).

funcion aproximar ▷ t_l: umbral global de pérdida
  $perdida \leftarrow 0$
  mientras $perdida < t_l$ **hacer:** ▷ no se alcanza el umbral de pérdida
    $n \leftarrow$ número de valores en ϕ
    para $i \in \{1, ..., n-1\}$ **hacer:**
      considerar la reducción de v_i y v_{i+1}
      // computar la pérdida de información V causada por la reducción
      computar $I(V, \mathbf{S}_i, \mathbf{S}_{i+1})$
    fin bucle
    elegir V_m que minimiza $I(V, \mathbf{S}_i, \mathbf{S}_{i+1}), i = 1 \ldots n-1$
    $perdida \leftarrow perdida + I(V, \mathbf{S}_m, \mathbf{S}_{m+1})$
    $V \leftarrow V_m$ ▷ si es posible, se sigue reduciendo V
  fin bucle
  devolver V ▷ devolver V cuando se llegue al límite de pérdida
fin funcion

Definición 4.3.2 ***(Pérdida de información)*** *Denotemos como V_j la estructura VBP aproximada que se obtiene en la iteración número j del algoritmo, y supongamos que nos encontramos en la iteración $j+1$. La nueva reducción a considerar se describirá como hemos definido anteriormente. Entonces, la* ***pérdida de información*** *producida en esta reducción se define como:*

$$I(V_j, \boldsymbol{S}_a, \boldsymbol{S}_b) = D(\phi, V_j) - D(\phi, V_{j+1}). \tag{4.13}$$

La selección del par de valores que minimizan la pérdida de información llevarán como consecuencia al mínimo valor de la distancia de Kullback–Leibler entre el potencial original y el aproximado.

Proposición 4.3.1 *La pérdida de información que se obtiene al reducir v_a y v_b en V puede computarse como:*

$$I(V, \boldsymbol{S}_a, \boldsymbol{S}_b) = \frac{1}{sum(V)} \left[log(v_r) sum(V^{\downarrow \boldsymbol{S}_r}) - log(v_a) sum(V^{\downarrow \boldsymbol{S}_a}) - log(v_b) sum(V^{\downarrow \boldsymbol{S}_b}) \right]. \tag{4.14}$$

donde $sum(V)$ denota la suma de cada valor de V y $V^{\downarrow \boldsymbol{S}}$ representa el potencial V restringido a las configuraciones incluidas en $\boldsymbol{S}$ y todas las demás se descartan. Si consideramos que ϕ es el potencial original y V su representación como VBP (quizás, tras aplicar operaciones de reducción repetidamente), entonces $sum(\phi) = sum(V)$ y la ecuación anterior puede reescribirse como:

$$I(V, \boldsymbol{S}_a, \boldsymbol{S}_b) = \frac{1}{sum(\phi)} \left[log(v_r) sum(\phi^{\downarrow \boldsymbol{S}_r}) - log(v_a) sum(\phi^{\downarrow \boldsymbol{S}_a}) - log(v_b) sum(\phi^{\downarrow \boldsymbol{S}_b}) \right]. \tag{4.15}$$

Demostración - *Dado ϕ un potencial representado por V (como VBP) y sea V_j el potencial que se obtiene como resultado de aplicar el algoritmo de aproximación sobre V en la iteración número j. De acuerdo con la Definición 4.3.2:*

$$I(V_j, \mathbf{S}_a, \mathbf{S}_b) = D(\phi, V_j) - D(\phi, V_{j+1}).$$

Esta diferencia puede ser calculada separando las configuraciones definidas en $\boldsymbol{X}$ en tres subconjuntos distintos: $\Omega_X = \{\Omega_{\boldsymbol{X}} \setminus \boldsymbol{S}_r\} \cup \boldsymbol{S}_a \cup \boldsymbol{S}_b$:

$$\begin{aligned} I(V_j, \mathbf{S}_a, \mathbf{S}_b) = D(\phi, V_j) - D(\phi, V_{j+1}) = \\ \sum_{\mathbf{x} \in \{\Omega_{\mathbf{X}} \setminus \mathbf{S}_r\}} \left[\overline{\phi}(\mathbf{x}) log(\frac{\overline{\phi}(\mathbf{x})}{\overline{V}_j(\mathbf{x})}) - \overline{\phi}(\mathbf{x}) log(\frac{\overline{\phi}(\mathbf{x})}{\overline{V}_{j+1}(\mathbf{x})}) \right] + \\ \sum_{\mathbf{x} \in \mathbf{S}_a} \left[\overline{\phi}(\mathbf{x}) log(\frac{\overline{\phi}(\mathbf{x})}{v_a / sum(\phi))} - \overline{\phi}(\mathbf{x}) log(\frac{\overline{\phi}(\mathbf{x})}{v_r / sum(\phi)}) \right] + \\ \sum_{\mathbf{x} \in \mathbf{S}_b} \left[\overline{\phi}(\mathbf{x}) log(\frac{\overline{\phi}(\mathbf{x})}{v_b / sum(\phi))} - \overline{\phi}(\mathbf{x}) log(\frac{\overline{\phi}(\mathbf{x})}{v_r / sum(\phi)}) \right]. \end{aligned}$$

Notamos que la primera parte de la suma es igual a 0, ya que los valores de las configuraciones que no intervienen en la reducción ($\boldsymbol{x} \in \Omega_{\boldsymbol{X} \setminus \boldsymbol{S}_r}$) son idénticos en V_j y V_{j+1}. Además, al tener en cuenta las propiedades de los logaritmos, la ecuación anterior puede expresarse como:

$$\sum_{\mathbf{x}\in\mathbf{S}_a} \overline{\phi}(\mathbf{x}) \left[log(\overline{\phi}(\mathbf{x})) - log(\frac{v_a}{sum(\phi)}) - log(\overline{\phi}(\mathbf{x})) + log(\frac{v_r}{sum(\phi)}) \right] +$$
$$\sum_{\mathbf{x}\in\mathbf{S}_b} \overline{\phi}(\mathbf{x}) \left[log(\overline{\phi}(\mathbf{x}) - log(\frac{v_b}{sum(\phi)}) - log(\overline{\phi}(\mathbf{x})) + log(\frac{v_r}{sum(\phi)}) \right] =$$
$$\sum_{\mathbf{x}\in\mathbf{S}_a} \overline{\phi}(\mathbf{x}) log(\frac{v_r}{v_a}) + \sum_{\mathbf{x}\in\mathbf{S}_b} \overline{\phi}(\mathbf{x}) log(\frac{v_r}{v_b}).$$

Puesto que $\overline{\phi}(\mathbf{x}) = \dfrac{\phi(\mathbf{x})}{sum(\phi)}$ *y si eliminamos el logaritmo de la suma, ya que no depende de las configuraciones, la anterior ecuación puede reescribirse como:*

$$log(\frac{v_r}{v_a}) \sum_{\mathbf{x}\in\mathbf{S}_a} \frac{\phi(\mathbf{x})}{sum(\phi)} + log(\frac{v_r}{v_b}) \sum_{\mathbf{x}\in\mathbf{S}_b} \frac{\phi(\mathbf{x})}{sum(\phi)} =$$
$$\frac{1}{sum(\phi)} \left[log(v_r) sum(\phi^{\downarrow\mathbf{S}_a}) - log(v_a) sum(\phi^{\downarrow\mathbf{S}_a}) + log(v_r) sum(\phi^{\downarrow\mathbf{S}_b}) - log(v_b) sum(\phi^{\downarrow\mathbf{S}_b}) \right] =$$
$$\frac{1}{sum(\phi)} \left[log(v_r) sum(\phi^{\downarrow\mathbf{S}_r}) - log(v_a) sum(\phi^{\downarrow\mathbf{S}_a}) - log(v_b) sum(\phi^{\downarrow\mathbf{S}_b}) \right].$$

□

4.3.4 Ejemplo

Para mostrar el funcionamiento del algoritmo de aproximación de VBPs vamos a proceder a aplicarlo sobre el ejemplo que hemos venido utilizando hasta ahora, el que fue definido en el Ejemplo 2.5. Podemos visualizar su representación como VDI en la Fig. 4.18. En la implementación práctica del algoritmo se busca simplificar el cómputo el máximo posible. La suma total de los valores del potencial en la Ecuación 4.14 puede obviarse, ya que no es necesaria para determinar la estructura candidata que produce la pérdida de información mínima.

Puesto que el potencial almacena 6 valores de probabilidad diferentes, la iteración inicial deberá considerar 5 estructuras candidatas, o lo que es lo mismo, 5 operaciones de reducción diferentes. Después se calcularán las respectivas pérdidas de información. Esta información puede observarse en la Tabla 4.4. Cada fila considera los valores a reducir (v_a y v_r) y el nuevo valor ponderado (v_r),

además de sus correspondientes conjuntos de índices ($\mathbf{S}_a$, $\mathbf{S}_b$ y $\mathbf{S}_r$). La alternativa que presenta la menos pérdida de información aparece señalada en negrita.

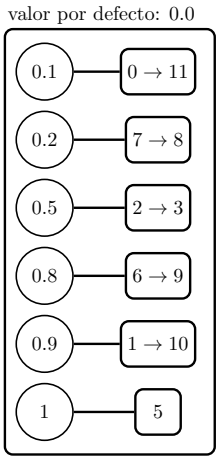


Fig. 4.18: Potencial a aproximar.

$v_a - \mathbf{S}_a$	$v_b - \mathbf{S}_b$	$v_r - \mathbf{S}_r$	$I(V, \mathbf{S}_a, \mathbf{S}_b)$
$0.1 - \{0, 11\}$	$0.2 - \{7, 8\}$	$0.15 - \{0, 7, 8, 11\}$	0.014757
$0.2 - \{7, 8\}$	$0.5 - \{2, 3\}$	$0.35 - \{2, 3, 7, 8\}$	0.057686
$0.5 - \{2, 3\}$	$0.8 - \{6, 9\}$	$0.65 - \{2, 3, 6, 9\}$	0.030339
$0.8 - \{6, 9\}$	$0.9 - \{1, 10\}$	$0.85 - \{1, 6, 9, 10\}$	0.002556
$\mathbf{0.9 - \{1, 10\}}$	$\mathbf{1 - \{5\}}$	$\mathbf{0.93333 - \{1, 5, 10\}}$	$\mathbf{0.001533}$

Tab. 4.4: Estructuras candidatas para la primera iteración. En negrita encontramos la estructura candidata a elegir en esta iteración.

Diremos entonces que la estructura candidata elegida será la que une los últimos dos valores de probabilidad (0.9 y 1.0), la que se corresponde con la última de las filas de la Tabla 4.4. La estructura VDI aproximada correspondiente puede verse en la Fig. 4.19 y la pérdida total es 0.001533.

Para la segunda iteración hay 4 estructuras candidatas a considerar, como se muestra en la Tabla 4.5.

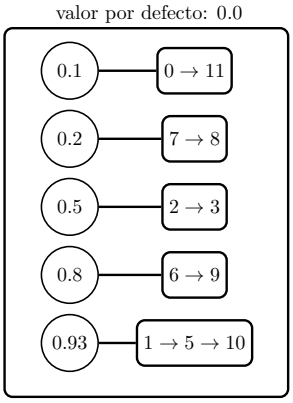


Fig. 4.19: Aproximación de ϕ obtenida en la primera iteración.

$v_a - \mathbf{S}_a$	$v_b - \mathbf{S}_b$	$v_r - \mathbf{S}_r$	$I(V, \mathbf{S}_a, \mathbf{S}_b)$
$0.1 - \{0, 11\}$	$0.2 - \{7, 8\}$	$0.15 - \{0, 7, 8, 11\}$	0.014757
$0.2 - \{7, 8\}$	$0.5 - \{2, 3\}$	$0.35 - \{2, 3, 7, 8\}$	0.057686
$0.5 - \{2, 3\}$	$0.8 - \{6, 9\}$	$0.65 - \{2, 3, 6, 9\}$	0.030339
$\mathbf{0.8 - \{6, 9\}}$	$\mathbf{0.93333 - \{1, 5, 10\}}$	$\mathbf{0.88 - \{1, 5, 6, 9, 10\}}$	**0.005323**

Tab. 4.5: Estructuras candidatas para la segunda iteración. En negrita se observa el mejor resultado de la reducción.

Si combinamos los valores 0.1 y 0.2, obtendremos la nueva estructura aproximada del potencial, visible en la Fig. 4.20.

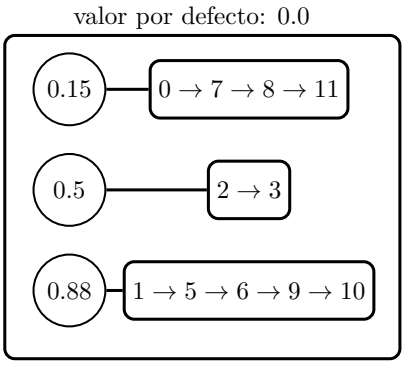


Fig. 4.20: Aproximación final de ϕ obtenida en la tercera iteración.

La pérdida de información en comparación con el potencial original es ahora 0.021613 (correspondiente a la pérdida acumulada de las iteraciones anteriores). La siguiente iteración deberá seleccionar de entre dos reducciones, como vemos en la Tabla 4.6.

$v_a - \mathbf{S}_a$	$v_b - \mathbf{S}_b$	$v_r - \mathbf{S}_r$	$I(V, \mathbf{S}_a, \mathbf{S}_b)$
$0.15 - \{0, 7, 8, 11\}$	$0.5 - \{2, 3\}$	$0.5333 - \{0, 2, 3, 7, 8, 11\}$	0.123074
$0.5 - \{2, 3\}$	$0.88 - \{1, 5, 6, 9, 10\}$	$0.65 - \{2, 3, 6, 9\}$	0.063298

Tab. 4.6: Estructuras candidatas para la cuarta iteración.

Si el límite de pérdida de información fijado es 0.05, entonces no se llevarán a cabo más reducciones y el proceso termina con el potencial aproximado que aparece en la Fig. 4.20.

Como último comentario sobre el funcionamiento del algoritmo, diremos que a la vista de las tablas, es evidente que los valores de pérdida de información para cada reducción permanecen constantes, independientemente de los otros valores. Esto facilita la consideración de estructuras alternativas almacenando los valores de pérdida ya calculados, evitando así cálculos repetidos.

4.3.5 Evaluación empírica del algoritmo de aproximación

La aplicación del algoritmo de aproximación se evaluará considerando varias *BNs*, todas desarrolladas para modelar problemas médicos. Todas ellas están incluidas en el repositorio de *bnlearn* (ver [174, 175]) y están categorizadas según su tamaño como grandes (*hepar2*) o muy grandes (*diabetes*, *munin*, y *pathfinder*). Veamos las principales características de estas redes:

- La red *hepar2* se definió por A. Onisko en su tesis doctoral [148] como parte del proyecto *HEPAR II*. Este proyecto fue inspirado por el sistema *HEPAR* [22], cuyo objetivo es el de dar soporte a la diagnosis de trastornos hepáticos y del tracto biliar.

- *pathfinder* ([93]) es un sistema que usado en combinación con conocimiento experto de patólogos quirúrgicos, puede ayudar en el diagnóstico de enfermedades de los ganglios linfáticos. Como resultado de este estudio, se definió la red discreta *pathfinder*.

- La red *munin* se definió al crear el sistema experto identificado por el mismo acrónimo *MUNIN* (MUscle and Nerve Inference Network) [8]. El objetivo de este sistema era ayudar a la electromiografía (EMC), que está diseñada para localizar y caracterizar lesiones del sistema neuromuscular, desde un enfoque fisiopatológico combinado con conocimiento experto.

- La BN *diabetes* [7] representa un modelo que trata de ajustar la terapia de insulina para personas que sufren de diabetes. El modelo considera el estado de los pacientes midiendo la glucosa en sangre, insulina biológicamente activa y cantidad de carbohidratos no digeridos en un lapso de una hora, además de otras variables conocidas involucradas en el proceso del metabolismo de la glucosa.

La siguiente Tabla 4.7 presenta un resumen de la información de cada red: número de nodos, número de arcos, número de parámetros (np), tamaño promedio del manto de Markov (M.B.), grado promedio y grado máximo de entrada. Las redes están ordenadas según el número de parámetros porque esta es la característica más relevante en términos del trabajo experimental.

Red	**Nodos**	**Arcos**	**np**	**Tam. medio M.B.**	**Grado med.**	**Grado máx. entr.**
hepar2	70	123	2139	4.51	3.51	6
pathfinder	223	338	97,851	5.61	3.03	6
munin	1041	1397	98,423	3.54	2.68	3
diabetes	413	606	461,069	3.97	2.92	2

Tab. 4.7: Características de las RBs.

Esta sección experimental se organiza del siguiente modo:

1. Análisis del espacio de memoria necesario para almacenar las redes completas utilizando las diferentes alternativas con el objetivo de compararlo con la representación 1DA, que consideramos la base (ver la Sección 4.3.5).

2. Análisis de las principales características de los potenciales específicos de algunas variables que luego se utilizarán para realizar la inferencia, así como el espacio de memoria necesario para su representación con las diferentes estructuras consideradas (ver Sección 4.3.5).

3. Examen del espacio de memoria necesario para almacenar cada red tras el proceso de aproximación (ver Sección 4.3.5). Se determina la relación con el espacio de memoria que requiere la representación base, así como la reducción que se produce en lo que se refiere a las representaciones alternativas pero sin aproximación alguna. En este caso, los resultados se presentan mediante una tabla específica para cada red con el fin de recoger la información sobre los valores umbral considerados.

4. Errores de propagación producidos por la aproximación, tanto locales (sólo se aproxima el potencial de la variable objetivo) como globales (se aproximan todos los potenciales) (ver Sección 4.3.5). Se presenta una tabla para cada red que recoge los resultados para las variables seleccionadas y para el conjunto de umbrales utilizados.

5. Para obtener más información sobre el efecto de la aproximación, también se incluyen algunos gráficos que muestran cómo la aproximación afecta al orden de las probabilidades de las distribuciones marginales obtenidas como resultado de la propagación. Si estas distribuciones se utilizan para tomar decisiones, es importante que las alternativas se mantengan en el mismo orden (según su valor de probabilidad) en el que aparecen en el resultado exacto, sin aproximación (ver Sección 4.3.5)

Análisis del tamaño de memoria global

Esta sección analiza las redes para determinar el tamaño de memoria necesario para almacenar cada forma de representación considerada: 1DA, PT, PPT, VDI e IPD. Esta parte dará información sobre la conveniencia de utilizar representaciones alternativas de tipo VBP en redes que modelan problemas médicos del mundo real. De esta forma, se dispone de un tamaño de memoria base que permitirá comprobar posteriormente el efecto de la aproximación sobre los espacios de memoria necesarios para las representaciones de tipo VBP. La Tabla 4.8 incluye la siguiente información:

- *Red*: nombre de la BN;
- *1DA*: espacio de memoria indispensable para almacenar el conjunto de potenciales 1DA;
- *PT*: espacio de memoria indispensable para almacenar la representación PT y si se ahorra o incrementa en referencia a la versión 1DA. Este segundo valor se muestra en la segunda línea y se calcula como:

$$\frac{as * 100}{bs} - 100, \tag{4.16}$$

donde as se refiere al espacio de memoria de la representación alternativa; y bs al de la representación 1DA;

- *PPT, VDI* e *IDP*: equivalentemente, se repite el punto anterior para las representaciones PPT, VDI e IDP.

En la Tabla 4.8, los resultados que ahorran el mayor espacio se muestran en negrita. A la vista de estos resultados diremos que las estructuras VBP se comportan mejor que los PT y PPT para cada BN, aunque en algunas no se produce ahorro de espacio al comparar con 1DA. Podemos también comentar que:

- En *hepar2*, PPT ofrece una pequeña mejoría en comparación con PT, lo que indica que en realidad hay pocos valores repetidos que pueden beneficiarse de la operación de poda. La estructura VDI aporta un ahorro de aproximadamente el 44% comparado con PT; mientras que IDP aporta un 62%.
- En el caso de *pathfinder*, se producen notables ahorros de espacio en memoria en comparación con 1DA y muy importantes si comparamos con PT y PPT. El mayor ahorro se produce al aplicar la estructura VDI.
- Sobre la red *munin*, VDI necesita casi el mismo espacio de memoria que 1DA y se ahorra cerca del 23% con IDP (de hecho, se ven ahorros significativos frente a PT y PPT).
- Por último, para *diabetes*, ambas estructuras VBP reducen bastante el espacio de memoria, siendo VDI algo mejor en este aspecto.

En definitiva, y a la vista de estos datos; estas estructuras presentan gran capacidad para ofrecer mecanismos eficientes de representación de información cuantitativa y, como se verá a continuación, permiten el uso de la operación de aproximación con la posibilidad de lograr un ahorro adicional de memoria.

Análisis del tamaño de memoria local

Este experimento recoge información sobre variables seleccionadas de cada BN, con el objetivo de determinar sus características y verificar la relación representación - espacio de memoria. Estas variables serán las utilizadas más tarde como variables objetivo al utilizar el algoritmo VE ([62, 182, 211]) para calcular sus distribuciones marginales. Las columnas incluidas en la Tabla 4.9 son:

Red	1DA	PT	PPT	VDI	IDP
hepar2	32,530	132,026 305.8592	131,756 305.0292	74,070 127.6975	49,602 **52.4808**
pathfinder	806,982	4,249,768 426.6249	3,779,470 368.3463	301,602 **−62.6259**	482,438 −40.2170
munin	994,672	3,393,878 241.2057	3,353,900 237.1865	997,864 0.3209	766,072 **−22.9825**
diabetes	3,773,200	10,044,948 166.2183	10,044,810 166.2146	964,380 **−74.4413**	1,105,728 −70.6952

Tab. 4.8: Análisis del espacio de memoria global. En negrita se observan las estructuras que arrojan los mejores resultados (o con el menor incremento en el porcentaje.)

- *Red*: nombre de la BN;
- *variable*: nombre de la variable a examinar;
- *np*: número global de parámetros del potencial de la variable objetivo;
- *nd*: número de valores diferentes en el potencial (que serán los números realmente almacenados en la estructura);
- *1DA*: espacio de memoria para la representación 1DA;
- *PT*: espacio de memoria para la representación PT y ahorro o pérdida en función de 1DA. Este segundo valor se incluye en la segunda línea y calculado como anteriormente hemos hecho;
- *PPT*, *VDI*, e *IDP*: tal y como hemos especificado en el caso de PT, para las estructuras PPT, VDI e IDP.

En este experimento, hemos seleccionado las variables con el mayor número de parámetros: *hepar2* (*ggtp*, *ast*, *alt*, y *bilirubin*); *pathfinder* (*F39*, *F74*, y *F40*); *munin* (*v1* (*L_LNLPC5_DELT_MUSIZE*), *v2* (*L_LNLE_ADM_MUSIZE*), y *v3* (*L_MED_ALLCV_EW*)); y *diabetes* (*cho_0*, *cho_1*, y *cho_2*). Los mejores valores de ahorro se muestran en negrita. A la vista de los resultados mostrados en la Tabla 4.9, podemos comentar:

- En las variables de *hepar2*, el número de valores de probabilidad diferentes es algo menor que el número de parámetros. Esto justifica el hecho de que el espacio de memoria requerido no se reduce comparado con 1DA, aunque ofrecen ahorros significativos con respecto a PT y PPT.

- Las variables seleccionadas de la red *pathfinder* presentan un alto índice de repetición, de modo que el número de valores distintos es bastante menor que el número de parámetros. Esto produce que los ahorros de memoria en comparación con 1DA sean muy significativos, que son mayores con la alternativa VDI.

- Para las dos primeras variables consideradas de la BN *munin*, encontramos sólo 12 valores distintos pero 600 parámetros. Esto justifica el ahorro de espacio en memoria que se produce para las estructuras VBPs. En el caso de la tercera variable, encontramos más valores distintos (133), pero también supone un gran grado de repetición si comparamos con los 600 parámetros.

- Las variables de *Diabetes* tienen características similares: solamente 45 valores diferentes (y 7056 valores posibles). Como consecuencia, el ahorro de espacio de memoria es muy notable y mejor en el caso de VDI.

Tamaño de memoria global con aproximaciones

Este experimento considera el efecto de aproximar cada potencial de las BNs desde el punto de vista del espacio en memoria necesario para almacenar las nuevas estructuras aproximadas, tras aproximar eligiendo distintos criterios de parada. Esto determina el grado en que la aproximación permite una reducción en el tamaño de memoria para almacenar las redes. Los resultados de esta sección se dividen en varias tablas, una para cada red (Tablas 4.10–4.13), y todas tienen una estructura similar.

- La primera columna contiene los distintos valores del criterio de parada.

- La segunda presenta datos relativos a la estructura VDI: tamaño de memoria tras aproximar, ahorro frente a 1DA y ahorro frente a la estructura VDI exacta.

- La tercera columna es idéntica a la segunda pero considerando los datos de la estructura IDP.

Tab. 4.9: Análisis de tamaño de memoria local. Los números en negrita representan los mejores porcentajes de ahorro (o con el menor porcentaje de incremento)

Red	Variable	np	nd	1DA	PT	PPT	VDI	IDP
hepar2	ggtp	384	334	3454	19,452 463.1731	19,452 463.1731	9990 189.2299	6150 **78,0544**
	ast	288	231	2636	13,648 417.7542	13,648 417.7542	7084 168.7405	4508 **71,0167**
	alt	288	249	2636	13,648 417.7542	13,648 417.7542	7516 185.1290	4652 **76,4795**
	bilirubin	288	244	2636	13,426 409.3323	13,426 409.3323	7396 180.5766	4612 **74.9621**
pathfinder	F39	8064	30	64,794	376,442 480.9828	359,850 455.3755	15,114 **−76.6738**	28,698 −55.7089
	F74	7560	111	60,712	293,736 383.8187	152,072 150.4810	28,676 **−52.7672**	52,632 −13.3087
	F40	4032	43	32,488	116,076 257.2888	114,588 252.7087	5640 **−82.6397**	9280 −71.4356
munin	v1	600	12	5032	19,132 280.2067	19,132 280.2067	1112 **−77.9014**	1464 −70.9062
	v2	600	12	5032	19,132 280.2067	19,132 280.2067	1112 **−77.9014**	1464 −70.9062
	v3	600	133	4982	15,012 201.3248	15,012 201.3248	4066 −18,3862	2582 **−48,1734**
diabetes	cho_0	7056	45	56,630	139,546 146.4171	139,546 146.4171	9454 **−83.3057**	16,878 −70.1960
	cho_1	7056	45	56,630	139,546 146.4171	139,546 146.4171	9454 **−83.3057**	16,878 −70.1960
	cho_2	7056	45	56,630	139,546 146.4171	139,546 146.4171	9454 **−83.3057**	16,878 −70.1960

Red *hepar2*

Tab. 4.10: hepar2—Análisis del tamaño de memoria global tras la aproximación.

Crit. parada	VDI	IDP
0.00001	63,846 (96.2681/−13.8032)	46,194 (42.0043/−6.8707)
0.00005	58,062 (78.4875/−21.6120)	44,266 (36.0775/−10.7576)
0.00010	55,302 (70.0031/−25.3382)	43,346 (33.2493/−12.6124)
0.00050	48,558 (49.2714/−34.4431)	41,098 (26.3388/−17.1445)
0.00100	45,678 (40.4181/−38.3313)	40,138 (23.3876/−19.0799)
0.00500	39,798 (22.3425/−46.2697)	38,178 (17.3624/−23.0313)
0.01000	37,518 (15.3335/−49.3479)	37,418 (15.0261/−24.5635)
0.05000	33,798 (3.8979/−54.3702)	36,178 (11.2143/−27.0634)
0.10000	32,550 (0.0615/−56.0551)	35,762 (9.9354/−27.9021)

Podemos hacer algunos comentarios sobre la Tabla 4.10:

- Para VDI, es evidente que hay un aumento muy notable en el ahorro de espacio de memoria a medida que el umbral utilizado para la aproximación se hace mayor, alcanzando tamaños muy similares a los de *1DA* para el umbral 0.1. Para cada umbral, hay una reducción en relación con la estructura VDI exacta (sin el uso de aproximación).

- Con la estructura IDP el comportamiento es similar, aunque las reducciones no son tan notables como con VDI.

Red *pathfinder*

Esta red contiene $97,851$ parámetros, y los tamaños de memoria para 1DA; PT y PPT son $806,982$, $4,249,768$ y $3,779,470$ respectivamente. Los tamaños de memoria para varios grados de aproximación se presentan en la Tabla 4.11.

Tab. 4.11: pathfinder—Análisis del tamaño de memoria global tras la aproximación.

Crit. parada	VDI	IDP
0.00001	293,178 (−63.6698/−2.7931)	479,630 (−40.5650/−0.5820)
0.00005	290,682 (−63.9791/−3.6207)	478,798 (−40.6681/−0.7545)
0.00010	289,266 (−64.1546/−4.0902)	478,326 (−40.7266/−0.8523)
0.00050	285,450 (−64.6275/−5.3554)	477,054 (−40.8842/−1.1160)
0.00100	283,170 (−64.9100/−6.1114)	476,294 (−40.9784/−1.2735)
0.00500	277,002 (−65.6743/−8.1564)	474,238 (−41.2331/−1.6997)
0.01000	274,050 (−66.0401/−9.1352)	473,254 (−41.3551/−1.9037)
0.05000	267,090 (−66.9026/−11.4429)	470,934 (−41.6426/−2.3846)
0.10000	264,450 (−67.2298/−12.3182)	470,054 (−41.7516/−2.5670)

Merece la pena recordar, que para esta red la alternativa VDI sin aproximación ya presentaba un ahorro de en torno al 62.7% con respecto a 1DA, que crece cuando el criterio de parada se hace mayor. Para el mayor límite de pérdida de información, el ahorro es de 12.3% al compararlo con la estructura VDI exacta. Con la estructura IDP se ven resultados similares.

Red *munin*

Esta red necesita $994,672$ parámetros para almacenar la información cuantitativa y el espacio de memoria que se requiere para representaciones alternativas con la operación de aproximación son $994,672$, $3,393,878$ y $3,353,900$ para 1DA, PT y PPT respectivamente. El efecto de aproximar puede observarse en la Tabla 4.12.

En esta red, el ahorro de espacio es realmente notable con respecto a 1DA y las representaciones exactas de VDI e IDP, aunque la opción VDI da mejores resultados.

Red *diabetes*

En la red *diabetes*, el número de parámetros es $461,069$ y el espacio de memoria para las representaciones 1DA, PT y PPT son $3,773,200$, $10,044,948$ y $10,044,810$ respectivamente. El efecto de aproximar en cuanto al espacio de memoria puede observarse en la Tabla 4.13. Al

Tab. 4.12: munin—Análisis del tamaño de memoria global tras la aproximación.

Crit. parada	VDI	IDP
0.00001	880,744 (−11.4538/−11.7371)	727,032 (−26.9074/−5.0961)
0.00005	829,096 (−16.6463/−16.9129)	709,816 (−28.6382/−7.3434)
0.00010	800,584 (−19.5128/−19.7702)	700,312 (−29.5937/−8.5840)
0.00050	725,440 (−27.0674/−27.3007)	675,264 (−32.1119/−11.8537)
0.00100	692,296 (−30.3996/−30.6222)	664,216 (−33.2226/−13.2959)
0.00500	615,976 (−38.0725/−38.2705)	638,776 (−35.7802/−16.6167)
0.01000	587,848 (−40.9003/−41.0894)	629,400 (−36.7229/−17.8406)
0.05000	533,728 (−46.3413/−46.5130)	611,360 (−38.5365/−20.1955)
0.10000	518,272 (−47.8952/−48.0619)	606,208 (−39.0545/−20.8680)

Tab. 4.13: diabetes–Análisis del tamaño de memoria global tras la aproximación.

Crit. parada	VDI	IDP
0.00001	843,180 (−77.6535/−12.5677)	1,065,328 (−71.7659/−3.6537)
0.00005	799,932 (−78.7996/−17.0522)	1,050,912 (−72.1480/−4.9575)
0.00010	776,868 (−79.4109/−19.4438)	1,043,224 (−72.3517/−5.6527)
0.00050	719,148 (−80.9406/−25.4290)	1,023,984 (−72.8617/−7.3928)
0.00100	694,908 (−81.5831/−27.9425)	1,015,904 (−73.0758/−8.1235)
0.00500	644,076 (−82.9302/−33.2135)	998,960 (−73.5249/−9.6559)
0.01000	626,172 (−83.4047/−35.0700)	992,992 (−73.6830/−10.1956)
0.05000	594,324 (−84.2488/−38.3724)	982,376 (−73.9644/−11.1557)
0.10000	585,636 (−84.4791/−39.2733)	979,480 (−74.0411/−11.4176)

igual que con la red *munin*, se producen importantes reducciones de espacio de memoria, siendo más relevantes para VDI.

Propagación de errores en la aproximación

El objetivo de esta parte es comprobar el efecto de la aproximación sobre los errores de propagación utilizando el algoritmo *VE* sobre el conjunto de variables seleccionadas. Para cada variable objetivo, se presentan dos valores diferentes: el error cuando la aproximación se limita al potencial de la variable objetivo y cuando la aproximación se aplica a todo el conjunto de potenciales. Como las aproximaciones de VDI e IDP producirán los mismos potenciales, este experimento se realizará exclusivamente en la representación de VDI. Los pasos seguidos para producir los resultados son:

1. Realización de una propagación de *VE* en cada variable objetivo para almacenar el resultado marginal obtenido como el resultado fundamental V_g.

2. Modificación de la red aproximando el potencial de la variable objetivo, los restantes potenciales se mantienen como se definieron en la especificación de la red.

3. Realización de una segunda propagación de *VE* en la red modificada configurando la variable objetivo seleccionada. El resultado se denomina V_{la}.

4. Aplicación de la aproximación sobre todo el conjunto de potenciales.

5. Cálculo de una tercera propagación para la variable seleccionada, produciendo V_{ga}.

6. Cálculo de las distancias entre el resultado base (exacto) y los aproximados: $D(V_g, V_{la})$ y $D(V_g, V_{ga})$.

Con el objetivo de introducir los resultados que se han obtenido, se presentan las Tablas 4.14–4.17, que se organizan como sigue: en la primera columna se observan los valores límites de pérdida de información; para cada variable, las columnas *"local"* contienen los errores de propagación con aproximación en las variables objetivo ($D(V_g, V_{la})$) y las columnas *"global"* muestran los errores de propagación cuando la aproximación se aplica a todos los potenciales ($D(V_g, V_{ga})$). Notamos que los valores que presentamos se han redondeado para incluir únicamente tres decimales. El valor de la distancia $d_l = 0.001$ se denominará valor límite.

Red *hepar2*

Al observar la Tabla 4.14 es evidente que si el valor del límite de pérdida de información está por debajo de 0.005, los errores permanecen por debajo de d_l. Los errores por encima de este valor de pérdida de información sólo aparecen para los últimos tres valores de umbral, e incluso para estos valores hay variables en las que tanto la aproximación global como la local permanecen por debajo de d_l. El valor de error más grande es 0.005 (bastante pequeño) para la variable *ggtp* en el caso de aproximación global con un valor umbral de 0.1

Tab. 4.14: hepar2—Análisis local y global de la propagación del error de aproximación.

Crit. parada	ggtp		ast		alt		bilirubin	
	Local	Global	Local	Global	Local	Global	Local	Global
0.00001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.00005	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.00010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.00050	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.00100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.00500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.01000	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.001
0.05000	0.002	0.004	0.001	0.001	0.000	0.000	0.000	0.001
0.10000	0.002	0.005	0.003	0.004	0.001	0.002	0.001	0.001

Red *pathfinder*

Los resultados de esta red son similares a los obtenidos para *hepar2* y pueden observarse en la Tabla 4.15. Todos los errores se encuentran por debajo de d_l para valores límites de pérdida de información de entre 0.00001 y 0.01. Incluso por encima de esos valores, los errores producidos por la aproximación local son menores que d_l en cada variable. El mayor error ocurre para el criterio de parada 0.1 y aproximación global para la variable *F40*.

Tab. 4.15: pathfinder—Análisis local y global de la propagación del error de aproximación.

Crit. parada	F39		F74		F40	
	Local	Global	Local	Global	Local	Global
0.00001	0.000	0.000	0.000	0.000	0.000	0.000
0.00005	0.000	0.000	0.000	0.000	0.000	0.000
0.00010	0.000	0.000	0.000	0.000	0.000	0.000
0.00050	0.000	0.000	0.000	0.000	0.000	0.000
0.00100	0.000	0.000	0.000	0.000	0.000	0.000
0.00500	0.000	0.000	0.000	0.000	0.000	0.000
0.01000	0.000	0.000	0.000	0.000	0.000	0.000
0.05000	0.000	0.004	0.000	0.000	0.000	0.005
0.10000	0.000	0.005	0.000	0.001	0.000	0.006

Red *munin*

Para la red *munin* sólo hay errores significativos para la aproximación global con criterios de parada altos (0.0096 para un máximo de pérdida de información de 0.05 y 0.093 para 0.1 y variable *v3*), tal y como se muestra en la Tabla 4.16. La aproximación local produce errores menores que d_l.

Red *diabetes*

Podemos ver en la Tabla 4.17 que para la red *diabetes*, la variable *cho_0* ofrece los peores resultados, con errores iguales a d_l en el caso de aproximaciones locales y valores límites de pérdida de información de más de 0.0005. En cualquier caso, todos los valores de error son muy pequeños, incluso para la aproximación global y altos valores como criterio de parada.

Tab. 4.16: munin—Análisis local y global de la propagación del error de aproximación.

Crit. parada	v1		v2		v3	
	Local	Global	Local	Global	Local	Global
0.00001	0.000	0.000	0.000	0.000	0.000	0.000
0.00005	0.000	0.000	0.000	0.000	0.000	0.000
0.00010	0.000	0.000	0.000	0.000	0.000	0.000
0.00050	0.000	0.000	0.000	0.000	0.000	0.001
0.00100	0.000	0.000	0.000	0.000	0.000	0.001
0.00500	0.000	0.001	0.000	0.001	0.000	0.005
0.01000	0.000	0.001	0.000	0.001	0.000	0.006
0.05000	0.000	0.002	0.000	0.003	0.000	0.096
0.10000	0.000	0.002	0.000	0.003	0.000	0.093

Tab. 4.17: diabetes—Análisis local y global de la propagación del error de aproximación.

Crit. parada	cho_0		cho_1		cho_2	
	Local	Global	Local	Global	Local	Global
0.00001	0.000	0.000	0.000	0.000	0.000	0.000
0.00005	0.000	0.000	0.000	0.000	0.000	0.000
0.00010	0.000	0.000	0.000	0.000	0.000	0.000
0.00050	0.001	0.001	0.000	0.001	0.000	0.000
0.00100	0.001	0.001	0.000	0.001	0.000	0.000
0.00500	0.001	0.001	0.000	0.000	0.000	0.000
0.01000	0.001	0.001	0.000	0.000	0.000	0.000
0.05000	0.000	0.000	0.000	0.001	0.000	0.000
0.10000	0.001	0.001	0.001	0.002	0.000	0.002

Orden de preferencias

Como se indicó anteriormente, los resultados de la propagación se pueden utilizar para ayudar en un proceso de toma de decisiones. Por lo tanto, es importante que los errores se mantengan bajos (como lo demuestran los experimentos anteriores), pero que también se mantenga el orden entre los valores de probabilidad de los estados de las variables sobre las que se realiza la propagación. En los gráficos incluidos en esta experimentación (ver las Tablas 4.18–4.22), los posibles estados de las variables se denotan como si. Imaginemos una variable con los tres estados: $s1, s2, s3$. Supongamos también que la propagación exacta indica que el orden de los estados según su probabilidad, de mayor a menor, es $s2, s1, s3$. Este será, por tanto, el orden de preferencias que se deberá mantener para que la decisión no cambie como consecuencia de los errores producidos por la aproximación. De esta forma, la situación ideal será aquella en la que el orden de las preferencias no se altere a pesar de la aproximación que se haga, ya sea global o local. Los colores de las tablas también representan las diferencias entre los valores de probabilidad obtenidos para cada alternativa (con respecto a los valores de probabilidad obtenidos en la propagación exacta), con colores que van desde el verde para las diferencias más bajas hasta el rojo para los valores más altos y donde los intermedios se encuentran en diferentes tonos de amarillo.

Red *hepar2*

Para esta red, el orden de preferencias es:

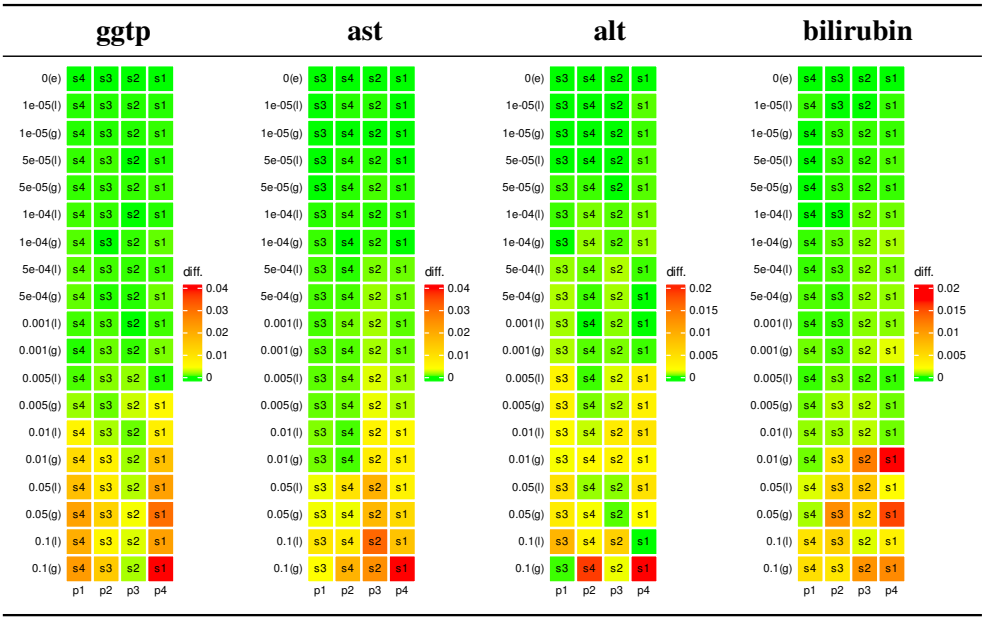
- $s3, s3, s2, s1$ para *ggtp* y *bilirubin*;
- y $s3, s4, s2, s1$ para *ast* y *alt*.

Este orden se mantiene para ambas aproximaciones local y global, además de para todos los valores límite. De manera similar, puede verse que las diferencias entre probabilidades se mantienen bajas, con valores máximos de 0.04 para el caso de las variables *ggtp* y *ast*. (Resultados observables en la Tabla 4.18.)

Red *pathfinder*

Para las tres variables consideradas en esta red, se observa el mismo comportamiento que en la red anterior: se mantienen los órdenes de preferencia para cada valor límite y ambas formas de aproximación. La mayor diferencia de probabilidad es 0.06 para la variable $F40$, que ocurre

Tab. 4.18: Preferencias para las variables de la red *hepar2*.



en la aproximación global con los mayores valores límite. (Resultados observables en la Tabla 4.19.)

Red *munin*

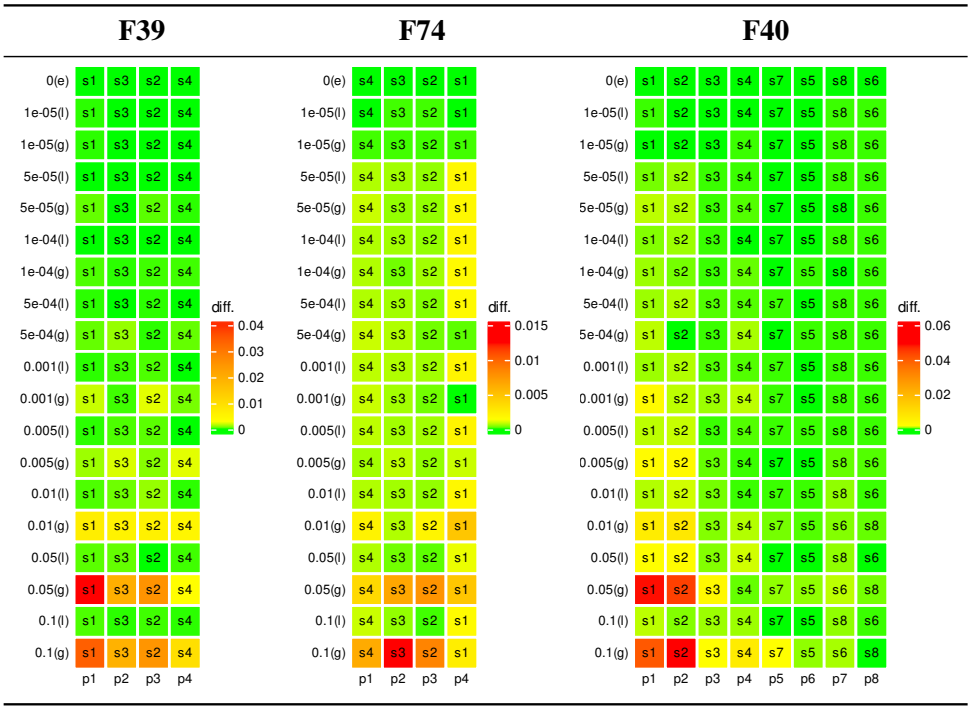
Para esta red, hay cambios en el orden de preferencias, aunque estos no afectan las alternativas más probables. Para las tres variables, los cambios aparecen con valores límites comenzando en 0.0005 y siempre en aproximación global. Los mayores valores de diferencias de probabilidad ocurren para la tercera variable, alcanzando 0.15 con límites altos y aproximación global. (Resultados observables en la Tabla 4.20.)

Red *diabetes*

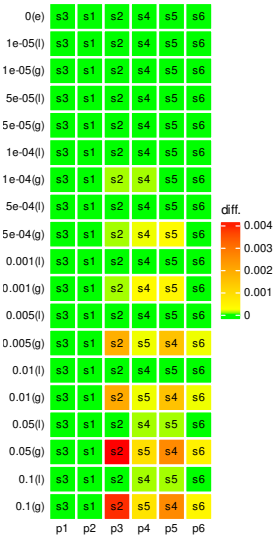
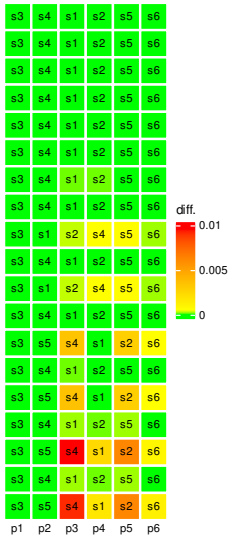
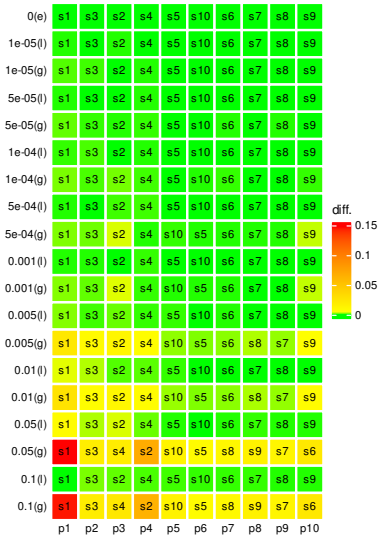
Los resultados para este caso se han dividido en dos tablas, ya que el número de estados de las variables consideradas es alto (21 en total).

Para las variables de la primera tabla (resultados observables en la Tabla 4.21), hay cambios en los órdenes de preferencias para valores límite desde 0.005 para ambos tipos de aproximación.

Tab. 4.19: Preferencias para las variables de la red *pathfinder*.



Tab. 4.20: Preferencias para las variables de la red *munin*.

L_LNLPC5_DELT_MUSIZE	L_LNLE_ADM_MUSIZE	L_MED_ALLCV_EW
		

Debemos notar que las diferencias entre valores de probabilidad son muy pequeñas (como máximo 0.02) y que los cambios no afectan las primeras preferencias (3 primeras para el caso de *cho_0* y 6 primeras para *cho_1*).

Para esta última variable, no hay cambios en el orden de preferencia y la diferencia entre los valores de probabilidad es muy pequeña, incluso en el caso de valores límite altos y aproximación global (el valor máximo es 0.02). (Resultados observables en la Tabla 4.22).

4.4 Comentarios y conclusiones sobre VBPs

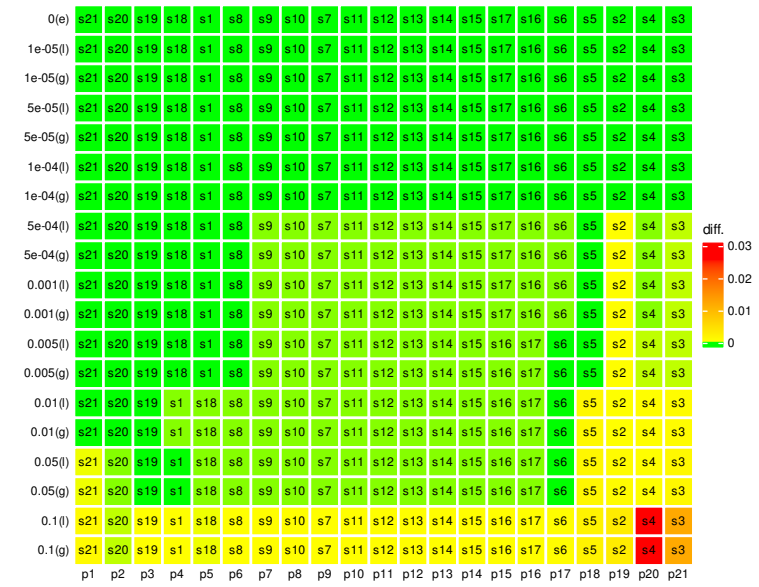
Sobre las estructuras de representación de la información cuantitativa VBPs, su aproximación y experimentos realizados podemos hacer los siguientes comentarios:

- En cuanto al uso del espacio de memoria que presentan las cuatro alternativas, se observa que todas ellas ofrecen resultados competitivos frente a 1DA, PT y PPT. Para la mayoría de las redes, se destacan las alternativas VDI e *IDP*. En cuanto a tiempos de acceso, la mejor alternativa es IDM. Por ello, se seleccionó esta representación como alternativa para las pruebas de VE.

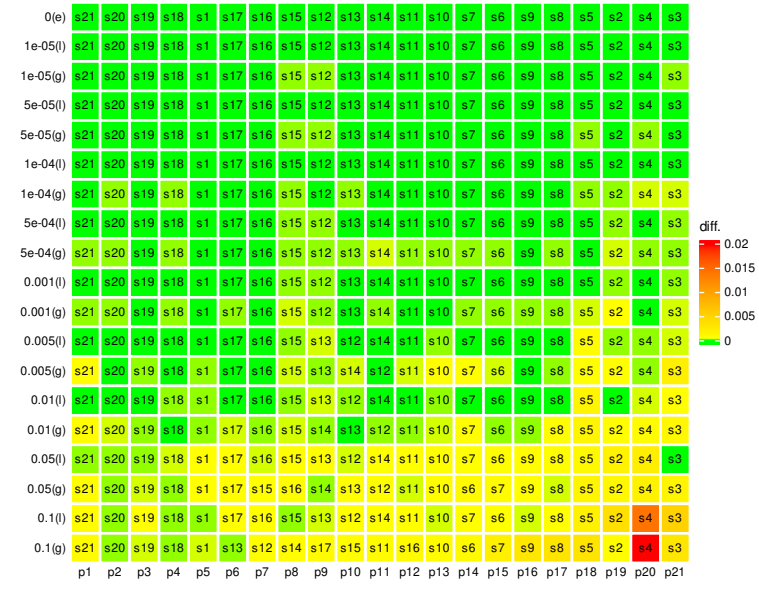
Tab. 4.21: Preferencias para las variables *cho_0* y *cho_1* de la red *diabetes*.

.

cho_0



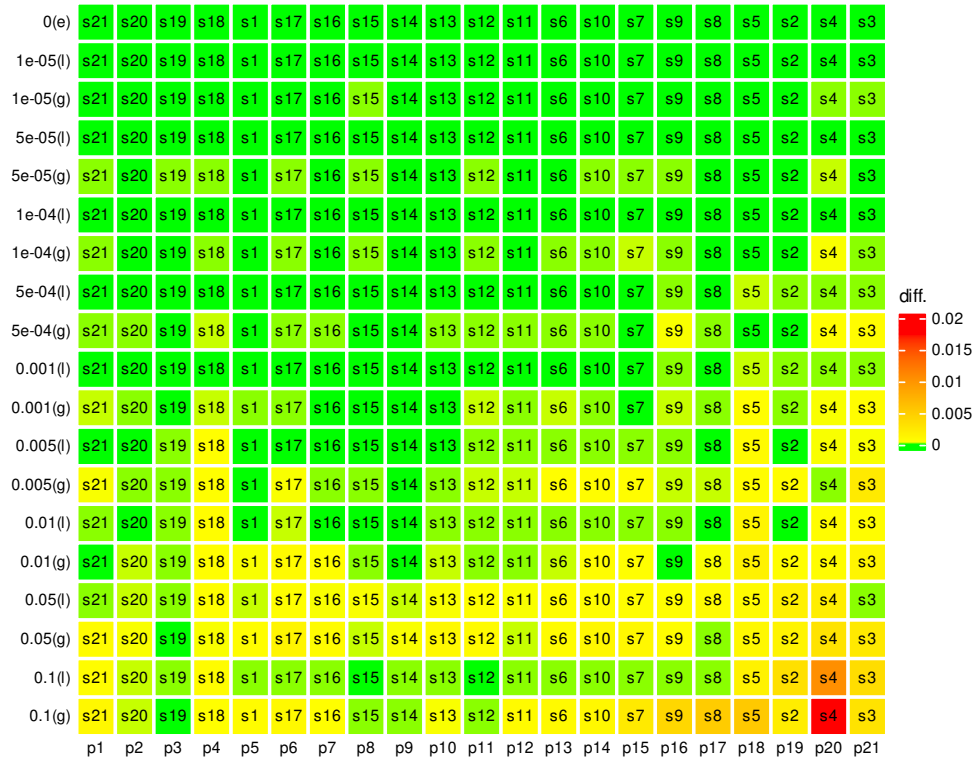
cho_1



Tab. 4.22: Preferencias para la variable *cho_2* de la red *diabetes*

.

cho_2



- Las versiones básicas de marginación y combinación permiten observar que VDI e IDM también ofrecen tiempos de ejecución razonables, similares a los necesarios para 1DA. En nuestra opinión, estos resultados son prometedores. También pensamos que implementaciones más eficientes producirán mejores resultados. Esta tarea podrá ser abordada en futuros trabajos.

- Puesto que los índices en las cuatro versiones VBPs se encuentran ordenados, utilizar búsqueda binaria podría suponer una mejora en cuanto al acceso a un índice determinado se refiere. La idea detrás de la búsqueda binaria de un valor (notemos v en un conjunto se basa en comparar el valor con el valor que ocupe la posición de en medio (notemos v_m), pueden encontrarse tres situaciones:

  - Si $v = v_m$, entonces la búsqueda termina.
  - Si $v < v_m$, se descarta la mitad superior de los valores.
  - En el caso contrario al anterior ($v > v_m$), se descarta la mitad inferior.

  El proceso se repite recursivamente, reduciendo en cada paso a la mitad de los valores posibles.

- En cuanto al trabajo realizado relacionado con la aproximación de las estructuras VBP, para el que seleccionamos un subconjunto de *BNs* que modelan problemas médicos reales, se observa que las distribuciones de probabilidad que cuantifican la incertidumbre de los problemas tienen varias características comunes, entre ellas el hecho de que existen muchos eventos imposibles y que algunos valores de probabilidad tienden a aparecer varias veces. Por ejemplo, al analizar la red *pathfinder* (ver Tabla 4.9), se puede ver que aunque la distribución de probabilidad para la variable *F39* tiene 8064 parámetros, solo hay 30 diferentes valores para estos. Sin embargo, estas repeticiones no siempre aparecen de manera que puedan ser utilizadas por estructuras en forma de árbol como *PPT* o *BPT*. Esto justifica el uso de las estructuras *VBP* consideradas, que en algunos casos permiten un ahorro considerable de espacio de memoria en relación con otras posibles estructuras de representación.

- La necesidad de lidiar con problemas cada vez más complejos puede conducir posteriormente a situaciones en las que los modelos no pueden evaluarse mediante algoritmos de inferencia exactos, como el algoritmo *VE*. En tales casos, nuestro trabajo considera la posibilidad de aproximar las estructuras de *VBP*, lo que supone un ahorro adicional de espacio a costa de perder una cantidad de información previamente aceptada. El sencillo método para aproximar VBPs demuestra experimentalmente que los errores que induce

en los resultados de los algoritmos de inferencia son pequeños y que en muchos casos no alteran los órdenes de preferencia entre alternativas variables. De esta forma, el proceso de toma de decisiones basado en los resultados aproximados coincidiría con el realizado si se pudiera calcular la propagación exacta.

- El software utilizado en este documento se implementó en **Scala**. El código está disponible en `https://github.com/mgomez-olmedo/VBPots` y los materiales también incluyen la información necesaria para reproducir los experimentos. El paradigma de programación funcional combinado con la orientación a objetos que ofrece **Scala** se puede usar para paralelizar operaciones bien definidas en CPU multinúcleo cuando sea posible. Algunos de estos beneficios fueron estudiados en [134].

5

Aprendizaje de conjuntos de redes Bayesianas con aproximación variacional

Con anterioridad (Sección 2.3.1), pudimos ver las distintas etapas del problema de aprendizaje sobre redes Bayesianas. En esta parte de la memoria nos centraremos en el aprendizaje estructural, que como ya vimos, trata de aprender el grafo de la RB que mejor se ajusta a los datos. Este proceso puede llevarse a cabo por expertos en la materia, aunque a medida que el tamaño de la red crece, se convierte en un problema exponencialmente más complejo. Por esta razón, aprender el grafo a partir del conjunto de datos y utilizando algún método automático, suele ser una mejor opción. Vimos también que los algoritmos pertenecen a una de tres categorías: basados en restricciones, basados en puntuaciones o híbridos. Hay otro modo de realizar aprendizaje estructural que en lugar de buscar el grafo que maximiza una determinada métrica, se centra en calcular la probabilidad marginal *a posteriori* de las aristas. Existen dos vertientes: (1) métodos de cadenas de Markov Monte Carlo (MCMC) [67, 74, 86, 131], cuyo objetivo es el de muestrear diferentes estructuras del grafo, de acuerdo con la distribución estacionaria de la cadena de Markov definida; y (2) métodos estocásticos de búsqueda [89, 107, 135] que no buscan converger a la distribución estacionaria (como MCMC), sino puntuar los modelos cuya métrica haya sido mayor en el proceso de búsqueda. La limitación de estos modelos es no poder ser aplicadas a grandes RBs, lo que puede solucionarse acudiendo a modelos que produzcan una respuesta aproximada. Este es el caso del nuevo método de aprendizaje estructural que proponemos en esta memoria, basado en análisis variacional.

En la Sección 5.1 establecemos los prerrequisitos y notación necesarios para comprender el método, incluyendo una introducción a lo que se conoce como análisis variacional. A continuación, en la Sec.5.2 podremos ver la notación y formulación utilizadas. En la Sec. 5.3 nos adentramos en el método de aprendizaje estructural variacional, comprendiendo la explicación y los algoritmos propuestos. Por último, presentamos en la Sec. 5.4 los resultados de aplicar el algoritmo a cuatro redes Bayesianas del repositorio *bnlearn* [174, 175].

5.1 Prerrequisitos

Supongamos que buscamos aprender una red Bayesiana dada por un conjunto de datos D que está formado por N variables. Estas variables serán notadas como hemos venido haciendo en el resto de la memoria, $\mathbf{X} = \{X_1, X_2, ..., X_N\}$; donde cada X_i toma valores en un dominio Ω_{X_i}. Supongamos además que $\mathcal{G}$ representa el grafo correspondiente a la RB, que determina las relaciones de dependencia existentes entre las variables. Este grafo $\mathcal{G}$ se especificará por medio de un vector de conjuntos de padres, $\mathcal{G} := (pa(X_1), pa(X_2), ..., pa(X_N))$, donde cada $pa(X_i)$ se corresponderá con el conjunto de padres del nodo X_i. Recordamos que un nodo pertenece al conjunto de padres del nodo X, $pa(X)$, cuando existe una arista dirigida que va de ese nodo a X (ver Sec 1.1 para más información).

Sabemos y vimos en el Capítulo 2.2 destinado a MGPs, que una RB quedará completamente definido al especificar el grafo y el conjunto de distribuciones de probabilidad. En este sentido, el vector θ contendrá los parámetros de las distribuciones de probabilidad marginales o condicionadas, donde θ_{ij} será el vector de dimensión $|\Omega_{X_i}|$ asociado a $P(X_i|pa(X_i) = j)$ y $pa(X_i) = j$ es la *j*-ésima asignación de valores de las variables en $pa(X_i)$.

5.1.1 Métrica Bayesiana

El problema del aprendizaje estructural en redes Bayesianas consiste, como hemos comentado, en encontrar el grafo dirigido acíclico que mejor representa al conjunto de datos D. Para cada conjunto de datos podremos encontrar varias RBs que los representen; es por eso que necesitamos establecer algún modo de cuantificar el ajuste red-datos, que permita seleccionar el mejor grafo. De esto se encargan las métricas Bayesianas. Existen diversas métricas Bayesianas cuyo uso está bastante extendido: BD (Bayesian Dirichlet) [92], K2 [50], BDe (Bayesian Dirichlet + "e" - verosimilitud equivalente) [92] y Bdeu (Bayesian Dirichlet verosimilitud equivalente + "u" - distribución conjunta uniforme) [28]. En este trabajo utilizaremos la última de estas métricas, BDeu. En el trabajo de Heckerman *et al.* [92], donde se define la métrica BD (base para BDe y BDeu), se establecen las cuatro suposiciones usadas para calcular la probabilidad marginal de los datos dada la estructura del grafo,

$$p(D|\mathcal{G}) = \int P(D|\mathcal{G}, \theta)\mathcal{P}(\theta|\mathcal{G}) \cdot \mathrm{d}\theta \tag{5.1}$$

Antes de presentar las suposiciones, fijaremos la siguiente notación:

- $\theta_{\mathcal{G}} = \{\theta_i\}_{i=1,...,n}$ nota los parámetros de una RB, $\mathcal{B}$, cuyo grafo correspondiente es $\mathcal{G}$.
- $\theta_i = \{\theta_{ij}\}_{j=1,...,q_i}$ nota los parámetros relacionados sólo con la variable $X_i \in \mathbf{X}$ en $\mathcal{B}$.
- $\theta_{ij} = \{\theta_{ijk}\}_{k=1,...,r_i}$ nota los parámetros de la variable $X_i \in \mathbf{X}$ en $\mathcal{B}$, dado que sus padres toman la configuración j-ésima.

Las suposiciones son las siguientes:

- **Suposición 1 (Muestra multinomial) -** considerado el conjunto de datos $D = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_M\}$, para cualquier instancia $\mathbf{y}_d \in D$ se cumple:

  $$P(\mathbf{y}_{di} = x_{ik} | \mathbf{y}_{dpa(X_i)} = w_{ij}, D_d) = P(X_i = x_{ik} | pa(X_i) = w_{ij}) = \theta_{ijk} \tag{5.2}$$

  donde $D_d = \{\mathbf{y}_1, ..., \mathbf{y}_{d-1}\}$.

- **Suposición 2 (Dirichlet) -** dado el grafo acíclico $\mathcal{G}$ tal que $P(\mathcal{G} > 0)$, entonces para cada parámetro $\theta_{ij} \in \theta_{\mathcal{G}}$ queda definida una distribución de Dirichlet *a priori* de parámetro α_{ij}.

- **Suposición 3 (Independencia de parámetros -)** dado el grafo acíclico $\mathcal{G}$ tal que $P(\mathcal{G} > 0)$, entonces:

  - **Independencia de parámetros global -** $score(\theta_{\mathcal{G}}|\mathcal{G}) = \prod_{i=1}^{n} score(\theta_i|\mathcal{G})$
  - **Independencia de parámetros local -** $score(\theta_i|\mathcal{G}) = \prod_{j=1} {q_i}^s core(\theta_{ij}|\mathcal{G})$ para todo $i = 1, ..., n$.

  donde $score$ nota a la métrica.

- **Suposición 4 (Modularidad de parámetros) -** dados dos grafos dirigidos acíclicos, $\mathcal{G}$ y $\mathcal{G}'$, tal que $P(\mathcal{G} > 0)$ y $P(\mathcal{G}' > 0)$, si X_i tiene los mismos padres en ambos grafos $\mathcal{G}$ y $\mathcal{G}'$, entonces:

  $$score(\theta_{ij}|\mathcal{G}) = score(\theta_{ij}|\mathcal{G}') \tag{5.3}$$

  para todo $j = 1, ..., q_i$.

Estas suposiciones de independencia nos permiten descomponer la probabilidad marginal de los datos dada la estructura del grafo como el siguiente producto de probabilidades:

$$P(D|\mathcal{G}) = \prod_{i}^{n} P(D_i | D_{pa(X_i)}) \tag{5.4}$$

donde por D_i y $D_{pa(X_i)}$ entenderemos respectivamente como los conjuntos de observaciones de X_i y de sus padres en $\mathcal{G}$. Los términos del producto que acabamos de definir en 5.4 pueden descomponerse como:

$$P(D_i|D_{pa(X_i)}) = \prod_{j=1}^{|\Omega_{X_i}|} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+Nij)} \prod_{k=1}^{|\Omega_{X_i}|} \frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})} \tag{5.5}$$

donde N_{ijk} es el número de instancias de D que tienen sentido para la asignación j-ésima de $pa(X_i)$ y con $X_i = k$; $N_{ij} = \sum_k N_{ijk}$ y $\alpha_{ijk} = \sum_k \alpha_{ijk}$.

Para el caso que nos compete, la métrica BDeu, tenemos que:

$$\alpha_{ijk} = \frac{1}{|\Omega_{pa(X_i)}||\Omega_{X_i}|} \tag{5.6}$$

La métrica BDe introducida por Heckerman *et al.* [92] añade una nueva suposición:

- **Suposición 5 (Verosimilitud equivalente) -** dados dos grafos dirigidos acíclicos $\mathcal{G}$ y $\mathcal{G}'$, tal que $P(\mathcal{G} > 0)$ y $P(\mathcal{G}' > 0)$, si $\mathcal{G}$ y $\mathcal{G}'$ son equivalentes (o lo que es lo mismo, ofrecen las mismas independencias condicionales) entonces:

$$score(\theta_D|\mathcal{G}) = score(\theta_D|\mathcal{G}') \tag{5.7}$$

La métrica BDeu se tratará de un caso particular de la métrica BDe, por lo que partirá también de las cuatro suposiciones de BD y de la quinta suposición de BDeu; y además considerará que cada estado del espacio conjunto, condicionado a la estructura de grafo, es igualmente probable Esto se traduce en:

$$P(X_i = x_{ik}, pa(X_i) = w_{ij}|\mathcal{G}) = \frac{1}{r_i q_i} \tag{5.8}$$

Con la definición de una distribución *a priori* de las estructuras de grafos, $P(\mathcal{G})$, especificamos completamente la métrica Bayesiana de una estructura de grafo: $score(\mathcal{G}|D) = lnP(D|\mathcal{G}) + lnP(\mathcal{G})$. De hecho, si la distribución *a priori* puede descomponerse localmente ($P(\mathcal{G}) = \prod_i P(pa(X_i))$, entonces la métrica puede también descomponerse localmente:

$$score(\mathcal{G}|D) = \sum_i score(X_i, pa(X_i)|D) \tag{5.9}$$

donde $score(X_i, pa(X_i)|D) = lnP(D_i|D_{pa(X_i)}) + lnP(pa(X_i))$.

Una manera sencilla de generar la distribución *a priori* sobre un grafo, $P(\mathcal{G})$ es considerar que se trata de una distribución uniforme. Diversos trabajos [16, 44, 173] han llegado a la conclusión de que no se trata del modo más óptimo de actuar, especialmente porque no tiene en cuenta el problema de corrección de multiplicidad. Este problema dice que si el número de candidatos a ser padre de una variable X_i crece, debe decrecerse la probabilidad de inclusión de arcos, para controlar el número de arcos falsos positivos. Se utiliza entonces, en lugar de la distribución uniforme: $P(\mathcal{G}) \propto \prod_i \binom{i}{|pa(X_i)|}^{-1}$.

5.1.2 Aprendizaje estructural Bayesiano de redes Bayesianas

El aprendizaje estructural Bayesiano de RBs se encarga de inferir una distribución sobre la estructura del gráfico acíclico dirigido (DAG) de las redes, a partir de los datos [63]. Muchos algoritmos de aprendizaje estructural devuelven un sólo DAG, lo que en ciertos casos puede llevar a predicciones no demasiado buenas [131], especialmente cuando el conjunto de datos no es muy grande. Pero en otros casos, los algoritmos devuelven un conjunto de soluciones. Las opciones son entonces: (1) seleccionar un único modelo, o (2) combinar las soluciones y utilizar así el modelo Bayesiano completo. En esta memoria, optaremos por la segunda de las opciones. Para llevar a cabo este proceso, en primer lugar necesitaremos calcular la probabilidad Bayesiana *a posteriori* sobre las diferentes estructuras de grafo, condicionada al conjunto de datos, es decir:

$$p(\mathcal{G}|D) = \frac{P(D|\mathcal{G})P(\mathcal{G})}{\sum_{G'} P(D|\mathcal{G}')P(\mathcal{G}')} \tag{5.10}$$

Una vez se tiene la probabilidad Bayesiana *a posteriori*, es posible comprobar si existe un arco dirigido, f, entre dos variables X_i y X_j. Puede hacerse calculando la probabilidad *a posteriori* como:

$$E[f|D] = \sum_{\mathcal{G}} f(\mathcal{G})P(\mathcal{G}|D) \tag{5.11}$$

donde:

$$f(\mathcal{G}) = \begin{cases} 1 & si \quad \mathcal{G}\ contiene\ un\ arco\ que\ une\ X_i\ y\ X_j \\ 0 & en\ otro\ caso \end{cases} \tag{5.12}$$

También, conocer la probabilidad $P(\mathcal{G}|D)$ permite calcular lo que se conoce como la distribución predictiva *a posteriori* de una variable latente X condicionada a los datos observados. La fórmula correspondiente es:

$$P(X|D) = \sum_{\mathcal{G}} p(X|\mathcal{G})P(\mathcal{G}|D) \tag{5.13}$$

esto hace posible calcular la incertidumbre que hay cuando más de un DAG se ajustan de manera similar a los datos.

El problema surge cuando el tamaño de variables crece, ya que estos cálculos de los sumatorios que acabamos de presentar en las Fórmulas 5.11 y 5.13 superarán muy probablemente la capacidad de la máquina. Tal y como pasa en otros procesos, puede optarse por la obtención de una solución aproximada que exprese el problema casi como la solución exacta, asumiendo una pequeña pérdida de información. Para poder resolver estos cálculos de manera aproximada, a menudo se han optado por algoritmos MCMC [74, 86, 131]; pero estos algoritmos tampoco pueden ser aplicados sobre conjuntos de datos de tamaño considerable. Más recientemente, se ha recurrido a métodos que utilizan análisis variacional y que sí son capaces de operar con redes de mayor tamaño; hablaremos de ello a continuación, en la siguiente Sección 5.1.3.

5.1.3 Análisis variacional

En la bibliografía, hablar de análisis variacional en el contexto de las redes Bayesianas, se refiere fundamentalmente a la inferencia variacional (VI) [21, 108, 202]. Se trata de un método para aproximar distribuciones de probabilidad, ampliamente usado en especial para la aproximación de distribuciones *a posteriori*. La inferencia variacional es una alternativa más rápida y escalable que las cadenas de Markov Monte Carlo (MCMC) (opción ampliamente utilizada), aunque produce una pequeña pérdida de información. En el trabajo de Blei *et al.* [21] se puede encontrar un buen resumen sobre la inferencia variacional, además de una guía para conocer sobre qué casos es más conveniente utilizar MCMC y en cuáles inferencia variacional.

Supongamos que nos encontramos ante un típico problema de inferencia, queremos calcular la probabilidad *a posteriori*. Sabemos y estudiamos en 1.2.6 que la fórmula para calcular la distribución condicionada de un conjunto de variables latentes $\mathbf{Z} = \{Z_1, Z_2, ...Z_m\}$ a otro de variables observadas $\mathbf{X} = \{X_1, X_2, ..., X_n\}$, $(P(\mathbf{Z}, \mathbf{X}))$, es:

$$p(\mathbf{Z}|\mathbf{X}) = \frac{p(\mathbf{Z}, \mathbf{X})}{p(\mathbf{X})} \tag{5.14}$$

donde el denominador de la fracción $P(\mathbf{X})$ es la distribución marginal de las observaciones, y puede calcularse marginalizando las variables latentes de la distribución conjunta de probabilidad.

$$p(\mathbf{X}) = \int p(\mathbf{X}, \mathbf{Z})\, dz \tag{5.15}$$

En muchos casos la marginalización no es posible, lo que complica las tareas de inferencia. Es por eso, que obtener una aproximación se presenta como una gran opción; y una manera de conseguirlo es mediante VI.

La idea básica detrás de la VI es la de generar una familia de funciones de distribución aproximadas sobre las variables latentes, que se nota como $\mathcal{Q}$; para posteriormente encontrar el miembro de esa familia que minimiza la distancia de Kullback–Leibler a la distribución *a posteriori* exacta,

$$q^*(\mathbf{Z}) = argmin_{q(\mathbf{Z})\in\mathcal{Q}} KL(q(\mathbf{Z}), p(\mathbf{Z}|\mathbf{X})) \tag{5.16}$$

donde KL es la distancia de Kullback–Leibler [117]. Así, aproximamos la probabilidad *a posteriori* mediante la optimización del miembro de la familia elegido q^*.

Pese a que nos hemos centrado en la optimización utilizando la distancia de Kullback–Leibler (inferencia variacional Kullback–Leibler [13]); Wainwright *et al.* [202] indican que puede utilizarse equivalentemente cualquier otro método que optimice para calcular la aproximación de una densidad, como podrían ser propagación de creencias [209] y propagación de expectativas [138], entre otros.

La distancia de Kullback–Leibler es una medida que sirve para comparar cómo de próximas se encuentran dos densidades. La distancia KL será 0 cuando ambas densidades sean idénticas, por lo que el objetivo será que esa distancia sea lo más pequeña posible. Su fórmula es la siguiente:

$$KL(q(\mathbf{Z}), p(\mathbf{Z}|\mathbf{X})) = E[log\ q(\mathbf{Z})] - E[log\ p(\mathbf{Z}|\mathbf{X})] \tag{5.17}$$

Por lo que, expandiendo la probabilidad condicional en la anterior Fórmula 5.17 quedaría:

$$KL(q(\mathbf{Z}), p(\mathbf{Z}|\mathbf{X})) = E[log\ q(\mathbf{Z})] - E[log(\mathbf{Z}, \mathbf{X})] + log\ p(\mathbf{X}) \tag{5.18}$$

Esto quiere decir que la distancia de Kullback–Leibler depende de $log\ p(\mathbf{X})$, lo que lo hace difícil de computar. Es por eso que se recurre a optimizar una fórmula equivalente salvo por una

constante; lo que se concoe como la *función límite inferior de la evidencia* (ELBO - por sus siglas en inglés *evidence lower bound*). Esta queda expresada como:

$$ELBO(q) = E[log\ p(\mathbf{Z}, \mathbf{X})] - E[log\ q(\mathbf{Z})] \tag{5.19}$$

Buscamos entonces maximizar la función ELBO, lo que es equivalente a minimizar la distancia de KL.

Otra propiedad de la función ELBO es que para todo $q(\mathbf{Z})$ se cumple que: $log\ p(\mathbf{X}) \geq ELBO(q)$. Esto es consecuencia directa de las Fórmula 5.18 y 5.19; y del hecho de que la distancia de Kullback–Leibler siempre es positiva [117]:

$$log\ p(\mathbf{X}) = KL(q(\mathbf{Z}), p(\mathbf{Z}|\mathbf{X}) + ELBO(q)) \tag{5.20}$$

Ya que acabamos de definir la función objetivo para el proceso de optimización (ELBO 5.19). Pasamos a especificar la familia variacional $\mathcal{Q}$. La aproximación más utilizada para aplicar VI es usando la conocida como *aproximación de campo medio* (*i.e.* familia variacional de campo medio), donde las variables latentes son mutuamente independientes. Cada miembro de la familia sigue la forma:

$$q^*(\mathbf{Z}) = \prod_i q(\mathbf{Z}_i) \tag{5.21}$$

Uno de los métodos más usados para resolver el problema de optimización, partiendo de la función ELBO y la familia variacional de campo medio, se conoce como *inferencia variacional de actualización de coordenadas* [19].

Dada una variable latente $Z_j \in \mathbf{Z}$, su condicional completa es su densidad condicional dadas el resto de variables latentes y las observadas, $p(Z_j|Z_{i\leq j}, \mathbf{X})$. La inferencia variacional de actualización de coordenadas se basa en optimizar iterativamente cada factor de la familia variacional de campo medio, manteniendo los otros fijos. Esto permite que podamos reescribir:

$$q(\mathbf{Z}) = \frac{e^{E_{i\leq j}[log\ p(D,\mathbf{Z})]}}{C} \tag{5.22}$$

donde C es la constante de normalización y $E_{i\leq j}$ nota la esperanza de las variables $\mathbf{Z} = \{Z_1, ..., Z_n\}$ tal que $j \leq i$, que utiliza la distribución definida por $\prod_j^{j\leq i} q(Z_j)$.

Puesto que se asume que las variables latentes son independientes, las esperanzas de la parte derecha de la Fórmula 5.22 no involucran al factor variacional j-ésimo, por lo que se trata de

una actualización de coordenadas válida. Estas ecuaciones de actualización de coordenadas 5.22 garantizan que la función ELBO crece de manera monótona con cada iteración; y llega, en algún momento, a un óptimo local.

5.2 Introducción al problema de aprendizaje estructural variacional

5.2.1 Familia variacional de campo medio

Para nuestro problema de aprendizaje estructural variacional, presentamos el grafo sobre el conjunto de variables aleatorias $\mathbf{X} = \{X_1, ..., X_N\}$ como una lista de la forma:

$$\mathcal{G} = \{X_i \leftrightarrows X_j, \ \ donde \ \ i, j = 1, ..., n \ \ con \ \ i < j\}$$

donde cada elemento $X_i \leftrightarrows X_j$ nos da información sobre los arcos entre las variables X_i e X_j.

Esta estructura de la red $\mathcal{G}$ podrá ser representada como una matriz de dimensión $N \times N$, donde cada elemento $\mathcal{G}_{ij}$ (donde i se asocia con el número de fila y j con el de columna) representa la relación entre las variables X_i y X_j. Cada uno de estos elementos $\mathcal{G}_{ij}$ podrá tomar uno de los siguientes valores:

- $\mathcal{G}_{ij} = +1$: si existe un arco que va de X_i a X_j ($X_i \rightarrow X_j$);
- $\mathcal{G}_{ij} = -1$: si el arco lleva el sentido contrario al anterior caso, es decir, que va de X_j a X_i ($X_j \rightarrow X_i$);
- $\mathcal{G}_{ij} = 0$: si las variables son independientes, o lo que es lo mismo, si no existe arco que une a X_i y X_j ($X_i \perp X_j$).

De este modo podemos generar una matriz con la particularidad de ser antisimétrica y cuya diagonal principal estará compuesta por ceros. Por esta razón, estaremos interesados únicamente en los valores que se encuentren por encima de la diagonal principal.

Ejemplo 42 *A modo ilustrativo, veamos la matriz que quedaría generada con un ejemplo. Consideraremos en esta sección una de las RBs que utilizamos en la parte de evaluación empírica del trabajo con potenciales basados en valor (Capítulo 4). Se trata de la red cancer, que puede encontrarse en el repositorio bnlearn y se definió por primera vez en el trabajo de Korb et al.*

[116]. Las variables involucradas en el problema son: Pollution (P), Smoker (S), Cancer (C), Xray (X) y Dyspnoea (D). Recordamos que sus características principales son las siguientes:

- *Número de nodos: 5.*
- *Número de arcos: 4.*
- *Número mínimo de estados: 2.*
- *Número medio de estados: 2.*
- *Número máximo de estados: 2*
- *Número de parámetros: 10.*

El grafo correspondiente a esta RB se define en el mismo trabajo de Korb et al. [116] como puede apreciarse en la Figura 5.1. Presentamos a continuación la matriz que representa la estructura de la red:

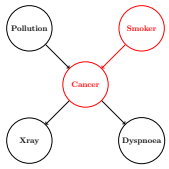


Fig. 5.1: Grafo de la red Bayesiana *cancer*.

$$G = \begin{array}{c} \begin{matrix} P & S & C & X & D \end{matrix} \\ \begin{pmatrix} \boldsymbol{0} & 0 & 1 & 0 & 0 \\ 0 & \boldsymbol{0} & 1 & 0 & 0 \\ -1 & -1 & \boldsymbol{0} & 1 & 1 \\ 0 & 0 & -1 & \boldsymbol{0} & 0 \\ 0 & 0 & -1 & 0 & \boldsymbol{0} \end{pmatrix} \end{array} \begin{matrix} P \\ S \\ C \\ X \\ D \end{matrix}$$

□

En rojo, podemos observar en el grafo el arco que va de la variable $Smoker$ a $Cancer$ y los dos elementos de la matriz que se corresponden con esta relación.

□

Además, para utilizar inferencia variacional sobre nuestro problema de aprendizaje estructural aproximando $P(\mathcal{G}|D)$, necesitaremos definir la familia variacional $\mathcal{Q}$. Esta familia $\mathcal{Q}$ se basa, como hemos comentado, en el método más común, la aproximación de campo medio y será la de presencia/ausencia de arcos entre las distintas variables aleatorias, es decir:

$$q(\mathcal{G}|\lambda) = \prod_{i,j}^{i<j} q(X_i \leftrightarrows X_j|\lambda_{ij}) \tag{5.23}$$

donde $q(X_i \leftrightarrows X_j|\lambda_{ij})$ se trata de una variable aleatoria multinomial que puede tomar tres posibles estados y que quedará parametrizada por el vector λ_{ij}:

- $X_i \perp X_j$ - que indica la no existencia de arco entre las variables X_i y X_j, con probabilidad λ_{ij_0}.
- $X_i \rightarrow X_j$ - indicando que existe arco que va de X_i a X_j, con probabilidad $\lambda_{ij_{-1}}$.
- $X_i \leftarrow X_j$ - indicando que existe arco que va de X_j a X_i, con probabilidad $\lambda_{ij_{+1}}$.

Alternativamente, podemos expresar la Fórmula 5.23 de la familia variacional como:

$$q(\mathcal{G}|\lambda) = \prod_i q(pa(X_i)|\lambda_i) \tag{5.24}$$

donde $q(pa(X_i)|\lambda_i)$ es la distribución variacional para los padres del nodo X_i.; λ_i denotará la colección de parámetros λ_{ij} para los que $i > j$. La distribución variacional de padres de X_i puede también expresarse como el producto de probabilidades de los arcos que definen el conjunto de padres:

$$q(pa(X_i)|\lambda_i) = \prod_{X_j \in pa(X_i)} q(X_i \leftarrow X_j|\lambda_{ij}) \prod_{X_k \notin pa(X_i)} (1 - q(X_i \leftarrow X_k|\lambda_{ik})) \tag{5.25}$$

Este conocimiento sobre la probabilidad del grafo $\mathcal{G}$ dado el conjunto de datos D quedará plasmado en otra matriz, $P(\mathcal{G}|D)$; donde cada elemento será una lista de dimensión 3 que contendrá esos tres valores de probabilidad $(\lambda_{ij_{+1}}, \lambda_{ij_0}, \lambda_{ij_{-1}})$. Esta matriz tendrá la misma dimensión que la anterior $(N \times N)$ y será notada como $Q(\mathcal{G}|\lambda)$. En esta segunda matriz, igualmente sólo estaremos interesados en los valores que se encuentren por encima de la diagonal principal; ya que los que se encuentren en la posición simétrica darán la misma información. Suponiendo un problema de dos variables X_1, X_2, la matriz seguiría el siguiente patrón:

$$Q(G|\lambda) = \begin{array}{c} \\ X_1 \\ X_2 \end{array} \begin{array}{c} \begin{array}{cc} X_1 & X_2 \end{array} \\ \begin{pmatrix} (P(Q_{1,1[-1]}), P(Q_{1,1[0]}), P(Q_{1,1[+1]})) & (P(Q_{1,2[-1]}), P(Q_{1,2[0]}), P(Q_{1,2[+1]})) \\ (P(Q_{2,1[-1]}), P(Q_{2,1[0]}), P(Q_{2,1[+1]})) & (P(Q_{2,2[-1]}), P(Q_{2,2[0]}), P(Q_{2,2[+1]})) \end{pmatrix} \end{array}$$

Ejemplo 43 *Suponiendo la misma red cancer del Ejemplo 42. La matriz de probabilidades generada tras ejecutar el algoritmo variacional para esta red cancer, tiene la siguiente forma:*

$$Q = \begin{array}{c} \\ P \\ S \\ C \\ X \\ D \end{array} \begin{array}{c} \begin{array}{ccccc} P & S & C & X & D \end{array} \\ \begin{pmatrix} & (0.16, \boldsymbol{0.66}, 0.4) & (0.38, 0.11, \boldsymbol{0.51}) & (0.16, \boldsymbol{0.68}, 0.16) & (0.4, \boldsymbol{0.65}, 0.17) \\ & & (\boldsymbol{0.51}, 0.07, 0.42) & (0.03, \boldsymbol{0.9}, 0.07) & (0.25, \boldsymbol{0.49}, 0.26) \\ & & & (\boldsymbol{0.57}, 0.14, 0.29) & (0.29, 0.08, \boldsymbol{0.63}) \\ & & & & (0.1, \boldsymbol{0.85}, 0.05) \\ & & & & \end{pmatrix} \end{array}$$

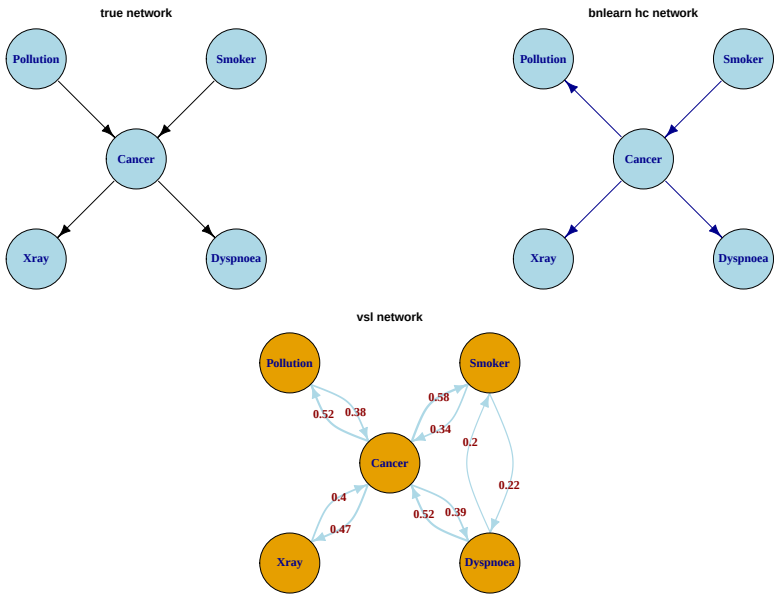


Fig. 5.2: Grafo de la red Bayesiana *cancer*.

En negrita podremos observar destacado el mayor valor de probabilidad de los tres presentes en cada elemento de la matriz. Sólo presentaremos los valores que se encuentren por encima de la diagonal principal, ya que, como comentamos, los valores de posiciones simétricas darán la misma información.

Partiendo de estos valores de probabilidad ofrecidos por el algoritmo de aprendizaje estructural variacional, será posible definir el grafo cuyas relaciones pueden deducirse de dichos valores. En la siguiente Figura5.2 podemos observar tres grafos, por un lado, el que se encuentra en la esquina superior izquierda, corresponde con el grafo original presente en el trabajo de Korb et al. [116] y que ya pudimos ver en el Ejemplo 42; el de la esquina superior derecha se corresponde con el que se aprende utilizando el método propuesto en bnlearn; y el de la parte inferior es el que se obtendría siguiendo la aproximación variacional que proponemos en esta memoria. Para este último grafo, mostramos también los valores de probabilidad (sin contar con el de independencia entre las dos variables, por ser de los arcos entre nodos, considerando sólo los valores por encima de 0.2. A la vista de los resultados numéricos presentes en la matriz de probabilidades, podemos comentar que las relaciones de dependencia e independencia entre las variables se encuentran claras y corresponden con las del grafo original. Por otro lado, el sentido de los arcos varía en función del método de aprendizaje usado y presenta discordancia con el original, lo que pone de manifiesto que los datos permiten y sugieren grafos diferentes.

□

5.2.2 Ecuaciones variacionales de actualización de coordenadas

En la anterior Sec 5.2.1 hemos visto cómo sería la familia variacional de aproximación Q. Y tal y cómo explicamos en 5.1, el objetivo es el de determinar el conjunto de parámetros λ que minimiza la Ecuación 5.16, que recordamos que tiene la forma:

$$argmin_{\lambda} KL(q(\mathcal{G}|\lambda), P(\mathcal{G}|D))$$

Y este problema puede resolverse, como comentamos, mediante un conjunto de ecuaciones de actualización de coordenadas observables en Ec.5.22 y que en nuestro caso se expresarán, por resultar más conveniente, en término de logaritmos, resultando la Ecuación–:

$$lnq(X_i \leftrightarrows X_j = k) = E_{\mathcal{G} \sim q(\mathcal{G} \setminus \{X_i \leftrightarrows X_j\})}[lnp(D, \mathcal{G} \cup \{X_i \leftrightarrows X_j = k\})] - lnC \qquad (5.26)$$

donde C es la constante de normalización, $k \in -1, 0, 1$, $q(\mathcal{G} \setminus \{X_i \leftrightarrows X_j\})$ se refiere a la distribución variacional sobre todos los arcos a excepción del que hay entre las variables X_i y X_j, y $\mathcal{G} \cup \{X_i \leftrightarrows X_j = k\}$ nota un grafo para el que el arco entre X_i y X_j se define determinísticamente por k.

La fórmula que acabamos de presentar (Ecuación 5.26) define un conjunto de ecuaciones de actualización, una por cada par de nodos del grafo, donde cada una tendrá que calcularse de manera iterativa hasta lograr convergencia. EL mayor problema de la anterior fórmula es realizar los cálculos asociados a la esperanza, ya que el cálculo exacto es inviable. La alternativa es usar métodos Monte Carlo para aproximar las ecuaciones de actualización por muestreo de las distribuciones variacionales. Este método tiene el problema, y es que $q(\mathcal{G})$ define una distribución multinomial sobre un número de estados más que exponencial. Por lo que para obtener una estimación Monte Carlo razonable, serán necesarias un gran número de muestras Monte Carlo. La manera de obtener un conjunto manejable de ecuaciones de actualización será mediante la factorización proporcionada por la regla de la cadena, que pudimos ver en la Ecuación 5.4.

5.2.3 Límite inferior de la evidencia

Como comentamos, una de las ventajas que presenta la inferencia variacional frente a los métodos Monte Carlo, es que optimizan una función, llamada *función límite inferior de la evidencia* (ELBO 5.19), cuyo valor puede usarse para evaluar la convergencia del método.

En nuestro caso, la función ELBO puede obtenerse mediante la Fórmula 5.27:

$$ELBO(q) = \sum_i E_{pa(X_i)\sim q(pa(X_i)|\lambda_i)}[score(X_i, pa(X_i)|D)] + H(q(\mathcal{G}|\lambda)) \tag{5.27}$$

donde el primer término de la suma se computa utilizando muestras Monte Carlo de $q(pa(X_i)|\lambda_i)$ y el segundo nota la entropía de la distribución variacional, que es la suma de la entropía de las distribuciones multinomiales individuales que definen la presencia o ausencia de arcos.

Finalmente, notaremos como $q(\mathcal{G}|\lambda^*)$ a la distribución variacional obtenida por las ecuaciones de actualización, una vez que se haya llegado a convergencia de acuerdo con la función ELBO.

5.3 Algoritmos de aprendizaje estructural variacional

El problema será, como venimos comentando, el de aprender el grafo a partir de los datos y lo enfocaremos desde un punto de vista variacional. El objetivo será entonces el de encontrar el conjunto de parámetros λ que verifican la expresión:

$$argmin_\lambda KL(Q(\mathcal{G}|\lambda)|P(\mathcal{G}|D)) \tag{5.28}$$

donde KL es la distancia de Kullback–Leibler [117] que definimos en el Capítulo 4.

Estamos entonces ante un problema de optimización que se resolverá mediante la estimación de gradientes de Monte Carlo y ascenso de coordenadas. El procedimiento general puede definirse como expresamos en el Algoritmo 8:

Algoritmo 8 Aprendizaje estructural variacional

```
1: funcion aprender(D)          ▷ donde D es el conjunto de datos a aprender
2:     inicializar Q(G|λ)                                          ▷ Algoritmo 9
3:     mientras no se cumpla criterio parada hacer:
4:         actualizar Q(Q(G|λ))                                    ▷ Algoritmo 10
5:         evaluar criterio parada
6:     fin bucle
7: fin funcion
```

5.3.1 Inicialización

La matriz que representa $Q(\mathcal{G}|D)$ puede ser inicializada de distintas formas, como considerando que los valores se distribuyen siguiendo una distribución uniforme o mediante otro algoritmo previo. En este estudio será inicializada de manera aleatoria tal y como describimos en el Algoritmo 9:

Algoritmo 9 Inicialización de la matriz $Q(\mathcal{G}|D)$

funcion inicializar($Q(\mathcal{G}|\lambda)$) ▷ Donde Q es una matriz $N \times N$
  para $i = 0$ hasta N **hacer:**
    para $j = i + 1$ hasta N **hacer:**
      generar un valor aleatorio $U(0, 1)$ para $G_{ij[+1]}$
      generar un valor aleatorio $U(0, 1)$ para $G_{ij[0]}$
      generar un valor aleatorio $U(0, 1)$ para $G_{ij[-1]}$
      normalizar valores de probabilidad
    fin bucle
  fin bucle
fin funcion

5.3.2 Optimización

El procedimiento para actualizar los valores Q_{ij} es el descrito en el Algoritmo 10.

Algoritmo 10 Actualización de matriz $Q(\mathcal{G}|\lambda)$

funcion actualizar($Q(\mathcal{G}|\lambda)$) ▷ Donde Q es una matriz $N \times N$; y m el número de muestras
  para $i = 0$ hasta N **hacer:**
    para $j = i + 1$ to N **hacer:**
      // generar el conjunto de padres para actualizar Q_{ij}: $pa(X_i), pa(X_j)$
      $dfpa_{X_i}$=generarConjuntoPadres($i, m, Q(\mathcal{G}|\lambda)$)
      $dfpa_{X_j}$=generarConjuntoPadres($j, m, Q(\mathcal{G}|\lambda)$)
      // actualizar valores de probabilidad
      actualizarQDistribucion($dfpa_{X_i}, dfpa_{X_j}, i, j, m, Q(\mathcal{G}|\lambda)$)
    fin bucle
  fin bucle
fin funcion

La actualización de Q_{ij} requiere muestrear conjuntos de padres para X_i y X_j y calcular la puntuación para cada situación posible: $X_i \rightarrow X_j$, $X_i \leftarrow X_j$ y $X_i \perp X_j$. El procedimiento para el muestreo de padres se describe en el Algoritmo 11. Al muestrear los padres de X_i es necesario considerar la distribución Q_{ij} para cada par $(X_i, X_j), j > i$ y $(X_j, X_i), j < i$. Las distribuciones para $Q_{ij}, j > i$ producirán j como padre si el número aleatorio generado conduce al estado $Q_{ij[-1]}$ (en este caso la arista será $X_i \leftarrow X_j$). Las distribuciones para $Q_{ji}, j < i$ añadirán j como padre si el número aleatorio conduce a la selección del estado $Q_{ij[+1]}$ (la arista va de X_j a X_i: $X_j \rightarrow X_i$).

Algoritmo 11 Generación de conjuntos de padres para X_i

funcion generarConjuntos($i, m, Q(\mathcal{G}|\lambda)$)
    crear una estructura de datos ($dfpa_{X_i}$) con m filas (número de muestras a obtener) y N columnas ($X_1, \ldots X_N$)
    para $s = 1$ hasta m **hacer:**
        para $j = i + 1$ hasta N **hacer:**
            generar un número aleatorio $r = U(0, 1)$
            seleccionar X_j como padre si r se corresponde con $Q_{ij[-1]}$, $dfpa_{X_i}(s, j) = 1$
        fin bucle
        para $j = 1$ hasta $i - 1$ **hacer:**
            generar un número aleatorio $r = U(0, 1)$
            seleccionar X_j como padre si r se corresponde con $Q_{ji[+1]}$, $dfpa_{X_i}(s, j) = 1$
        fin bucle
    fin bucle
    devolver $dfpa_{X_i}$
fin funcion

Las estructuras de datos $dfpa_{X_i}$ y $dfpa_{X_j}$ se usan para actualizar la distribución Q_{ij}. Este paso se describe en el Algoritmo 12. Cuando este procedimiento termina, habrá nuevas estimaciones para las probabilidades que describen los estados para la arista entre X_i y X_j. el método auxiliar llamado actualizarProb se introduce por conveniencia (ver Algoritmo 13).

La actualización de cada valor de probabilidad se explica en el Algoritmo 13 y requiere el cálculo de los parámetros.

Algoritmo 12 Actualización de la distribución Q_{ij}

funcion actualizarDistribucionQ($dfpa_{X_i}, dfpa_{X_j}, i, j, m, Q(\mathcal{G}|\lambda)$)
    // completa conjunto de padres para calcular parámetros
    s_{i1} = $dfpa_{X_i}(X_j = 1)$
    s_{i0} = $dfpa_{X_i}(X_j = 0)$
    s_{j0} = $dfpa_{X_j}(X_i = 0)$
    s_{j1} = $dfpa_{X_j}(X_i = 1)$

    // actualiza el valor de probabilidad para $Q_{ij[-1]}$
    $Q_{ij[-1]} = actualizarProb(m, s_{i1}, s_{j0})$

    // actualiza el valor de probabilidad para $Q_{ij[+1]}$
    $Q_{ij[+1]} = actualizarProb(m, s_{i0}, s_{j1})$

    // actualiza el valor de probabilidad para $Q_{ij[0]}$
    $Q_{ij[0]} = actualizarProb(m, s_{i0}, s_{j0})$
fin funcion

Algoritmo 13 Actualización de $Q_{ij[s]}$

funcion actualizarProb(m, s_i, s_j) ▷ s_i, s_j: los conjuntos de padres para X_i y X_j
    sum=0
    para k = 1 hasta m **hacer:**
        conjuntos $pa_{X_i} = X_l$, donde $s_i(k, l) = 1$
        sum=sum+$calcularScore(X_i, pa(X_i))$
    fin bucle
    para k = 1 hasta m **hacer:**
        conjuntos $pa_{X_j} = X_l$, donde $s_j(k, l) = 1$
        sum=sum+$calcularScore(X_j, pa(X_j))$
    fin bucle
    devolver sum/n
fin funcion

La última operación a llevar a cabo es el cálculo de la puntuación de una variable, X_i, dado un cierto conjunto de padres, $pa(X_i)$. Esta operación se describe en el Algoritmo 14.

Algoritmo 14 Cálculo de puntuación

funcion calcularScore(X_i, $pa(X_i)$)
  $q_i = \prod_{x_j \in pa(X_i)} r_j$, ($r_j$: número de estados de X_j)
  $n_{ik} = 0$, para todos los estados k de X_i
  $s = 1$, (tamaño de muestra equivalente)
  para j=1 hasta $q_i =$ **hacer:** ▷ obtiene estadísticos del conjunto de datos
    $n_{ij} = 0$
    para k=1 hasta r_i (número de estados de X_i) **hacer:**
      n_{ijk}: número de instancias con $X_i = k$, $pa(X_i) = q_j$
      $n_{ij} = n_{ij} + n_{ijk}$
    fin bucle
  fin bucle
  sum=0 ▷ calcula puntuación
  para j=1 hasta $q_i =$ **hacer:**
    val1= $log\left(\frac{\Gamma(s/q_i)}{\Gamma(n_{ij}+s/q_i)}\right)$
    val2=0
    para k=1 hasta r_i **hacer:**
      val2=val2+$log\left(\frac{\Gamma(n_{ijk}+s/r_iq_i)}{\Gamma(s/r_iq_i)}\right)$
    fin bucle
    sum=sum+val1+val2
  fin bucle
  penalizacion=$ln(\binom{N}{|\Pi_i|})$ ▷ $|\Pi_i|$: número de padres de X_i
  devolver $sum - penalizacion$
fin funcion

5.3.3 Evolución del algoritmo

La evolución del algoritmo puede analizarse calculando el límite inferior de la probabilidad conjunta representada por Q:

$$\mathcal{L}(Q) = \sum_{i=1}^{N} \frac{1}{N} \sum_{\Pi_k \sim Q(pa(X_i))} calcularScore(X_i|\Pi_k) \tag{5.29}$$

El procedimiento para obtener este límite se describe en el Algoritmo 15.

Algoritmo 15 Cálculo de límite inferior

funcion calculoLimiteInferior($Q(\mathcal{G}|\lambda)$)
  sumaGlobal=0
  para i = 1 hasta N **hacer:**
    $dfpa_{X_i} = generarConjuntoPadres(i, m, Q(\mathcal{G}|\lambda)$
    sum=0
    para k = 1 hasta m **hacer:**
      $pa(X_i) = dfpa_{X_i}(k, X_j = 1)$ ▷ selecciona X_j en fila k con $dfpa_{X_i}(i, j) = 1$
      sum=sum+$calcularScore(X_i, pa(X_i))$
    fin bucle
    sumaGlobal=sumaGlobal+(sum/m)
  fin bucle
  devolver $globalSum$
fin funcion

5.4 Experimentación con el algoritmo

Para evaluar cómo se comporta el procedimiento de aprendizaje variacional que acabamos de introducir hemos considerado el estudio sobre un total de 4 RBs, todas disponibles en el repositorio *bnlearn* [174, 175]: *alarm, asia, hepar2, win95ps*. Algunos detalles sobre estas redes se dieron en el anterior Capítulo 4 (ver Sec. 4.2.4). Veamos un poco de información sobre estas cuatro RBs:

- La red *alarm* se define en el trabajo de Beinlich et al. [14], donde buscan avisar a pacientes de diagnósticos peligrosos. La red se compone de 8 nodos relativos a diagnósticos, 16 recomendaciones y 13 variables intermedias.

- *asia* [124] es la red que hemos venido considerando como ejemplo en varias secciones de la memoria y que podemos ver en 2.2. Se compone de 8 variables, cada una con dos posibles estados.

- Ya comentamos en 4.3.5 que la RB *hepar2* se definió por A. Onisko en su tesis doctoral [148] como parte del proyecto *HEPAR II*. Este proyecto fue inspirado por el sistema *HEPAR* [22], cuyo objetivo es el de dar soporte a la diagnosis de trastornos hepáticos y del tracto biliar.

- *win95ps*, definida en el trabajo de Heckerman et al. [91], representa el conocimiento para solucionar problemas de impresoras en Windows 95.

Procedemos a especificar en la Tabla 5.1 los relevantes para este estudio. En esta tabla podemos encontrar las siguientes columnas:

- **red-** que contiene en cada fila el nombre de la red a la que se refiere.
- **nodos-** número de variables del grafo.
- **arcos-** número de conexiones entre los nodos.
- **grado med.-** grado medio de los nodos. Recordamos que el grado un nodo es el número de arcos que salen de él; por tanto, el grado medio será la media de los grados de todos los nodos.
- **grado máx.-** grado máximo de los nodos de la red.
- **tam. train-** será el número elegido de instancias que formarán parte del conjunto de entrenamiento.
- **tam. test-** de manera análoga al anterior, será el tamaño del conjunto de test.
- **iter-** se refiere al número de iteraciones que el algoritmo realizará antes de parar el proceso.
- **np-** serán los cuatro tamaños del conjunto de padres, elegidos teniendo en cuenta la naturaleza de las RBs para probar el algoritmo. Darán una idea aproximada o rango del número de padres que debe tener el grafo a aprender.

Los experimentos se realizaron 5 veces para cada RB, donde para cada una de esas repeticiones se utilizaban un conjunto de test y otro aleatorio de entrenamiento, manteniendo el tamaño y número de iteraciones.

red	nodos	arcos	parámetros	grado med.	grado máx.	tam. train	tam. test	iter	np
alarm	37	46	509	2.49	4	3500	10000	3500	$\{30, 50, 70, 90\}$
asia	8	8	18	2	2	100	10000	105	$\{10, 20, 30, 40\}$
hepar2	70	123	1453	3.51	6	3000	10000	10350	$\{25, 50, 75, 100\}$
win95pts	76	112	574	2.95	7	2500	10000	14250	$\{25, 50, 75, 100\}$

Tab. 5.1: Características de las redes Bayesianas estudiadas del repositorio *bnlearn* para evaluar el algoritmo de aprendizaje variacional de redes.

Los resultados obtenidos mediante la experimentación se presentan mediante una tabla resumen para cada RB. Esta tabla tendrá 5 filas, en la primera podemos observar el valor de referencia de la métrica BDeu para *test* y *train* proporcionado por *bnlearn*, y el resto de filas serán las distintas métricas obtenidas al aplicar el algoritmo de aprendizaje estructural variacional; y 6 columnas, donde las 5 últimas presentan los resultados de cada repetición. Además, después de cada tabla podremos ver varias gráficas que evalúan el *score* del modelo a medida que el número de iteraciones aumenta. Para cada repetición del experimento con distintos conjuntos de *train* y *test* tendremos 4 gráficas, una por cada tamaño del conjunto de padres. En cada gráfica encontraremos:

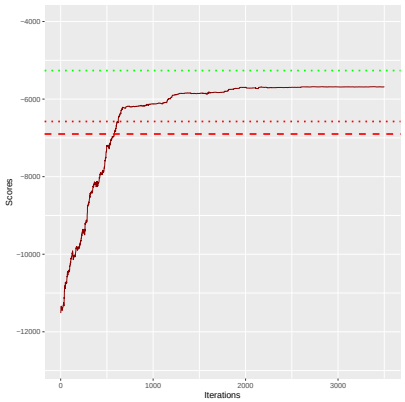
- Línea en color verde: indicando el resultado del modelo variacional sobre el conjunto de test.
- Línea discontínua en color rojo: indicando el resultado dado por *bnlearn* sobre el conjunto de entrenamiento.
- Línea de puntos en color rojo: resultado dado por *bnlearn* sobre el conjunto de test.
- Línea continua en color rojo: muestra la evolución del *score* variacional.

5.4.1 Red *alarm*

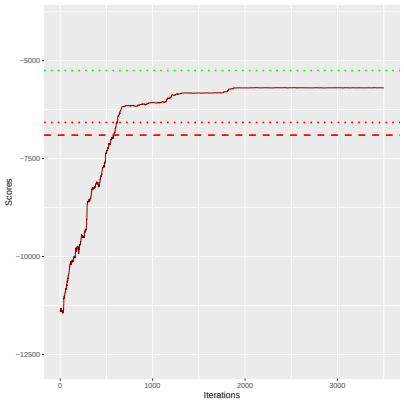
Sobre los resultados que se presentan en la Tabla 5.2 y las gráficas presentes en las Figuras 5.3, 5.4, 5.5, 5.6 y 5.7, referentes a la red *alarm*, podemos comentar que en todos los casos se mejora utilizando el algoritmo de aprendizaje variacional propuesto, con respecto del valor de *bnlearn*. Siempre se produce convergencia entre las 600-700 iteraciones, independientemente del número de padres seleccionado. Puesto que al observar las gráficas con distinto número de padres no se produce gran mejora, podremos concluir que el tamaño de padres adecuado podría ser cercano a 30.

	V1		V2		V3		V4		V5	
	train	test	train	test	train	test	train	test	train	test
bnlearn	-6900	-6577	-7042	-6738	-7026	-6712	-7080	-6730	-7188	-6651
np = 30	**-5685**	-5265	-5882	-5345	-5759	**-5220**	**-5783**	**-5192**	**-5918**	**-5284**
np = 50	-5693	**-5251**	-5977	-5394	-5732	-5229	-5867	-5301	-6017	-5359
np = 70	-5713	**-5251**	**-5870**	**-5317**	-5769	-5286	-5843	-5239	-5933	-5311
np = 90	-5697	-5257	-5950	-5364	**-5727**	-5247	-5878	-5287	-6000	-5367

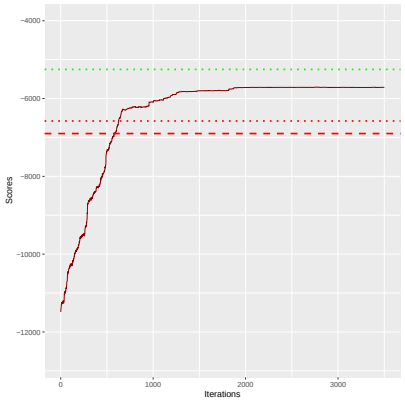
Tab. 5.2: Tabla resumen de los resultados obtenidos para la red *alarm* sobre las 5 distintas repeticiones del algoritmo de aprendizaje variacional, considerando distintos tamaños del conjunto de padres.



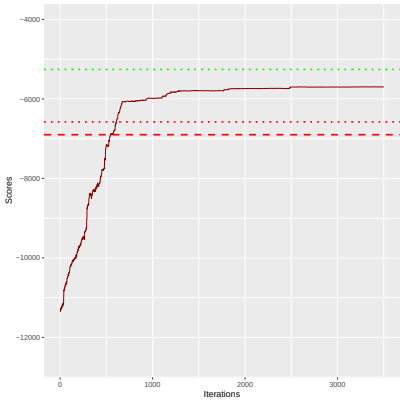
(a) Tamaño del conjunto de padres = 30.



(b) Tamaño del conjunto de padres = 50.

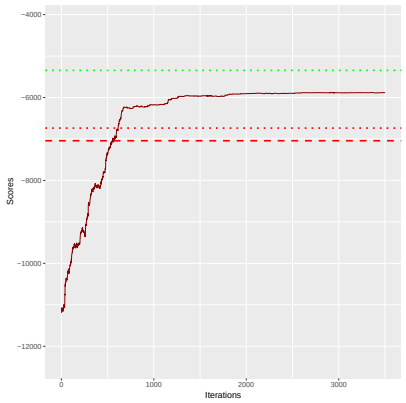


(c) Tamaño del conjunto de padres = 70.

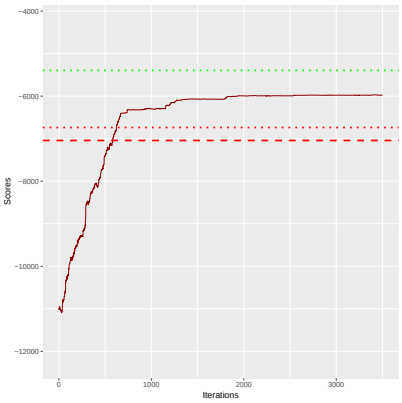


(d) Tamaño del conjunto de padres = 90.

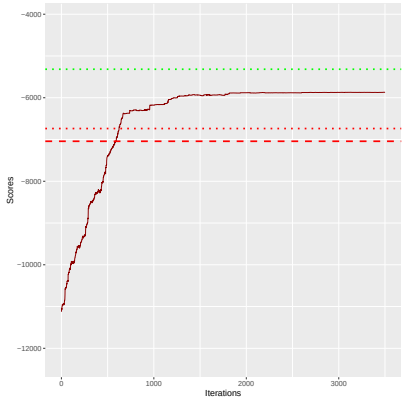
Fig. 5.3: Gráficas de evolución de la métrica según el número de iteraciones, primer experimento red *alarm* (V1).



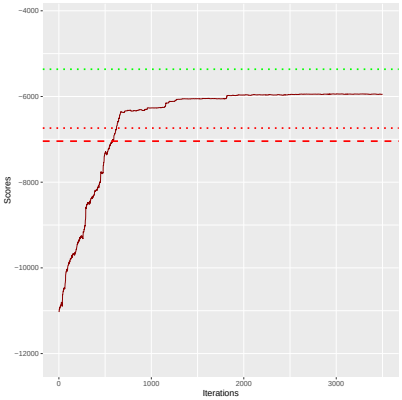
(a) Tamaño del conjunto de padres = 30.



(b) Tamaño del conjunto de padres = 50.

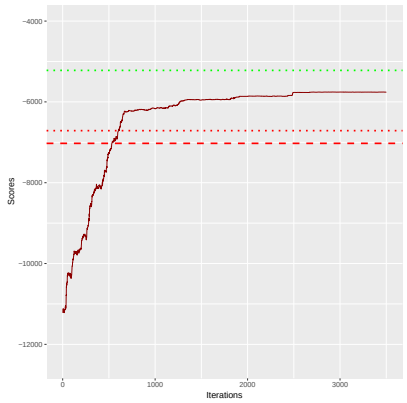


(c) Tamaño del conjunto de padres = 70.

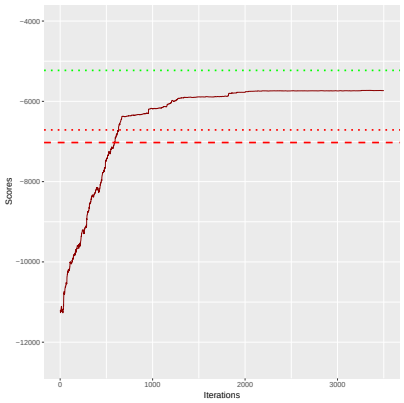


(d) Tamaño del conjunto de padres = 90.

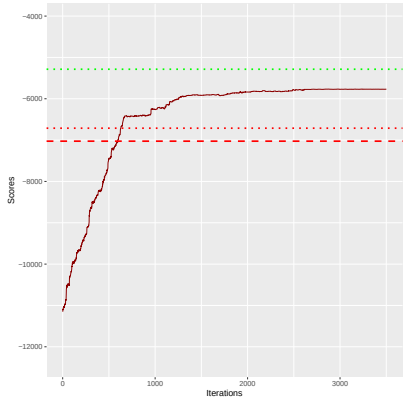
Fig. 5.4: Gráficas de evolución de la métrica según el número de iteraciones, segundo experimento red *alarm* (V2).



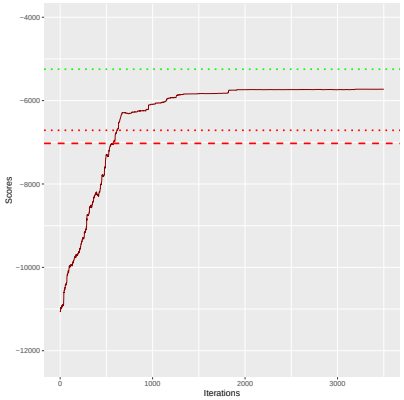
(a) Tamaño del conjunto de padres = 30.



(b) Tamaño del conjunto de padres = 50.

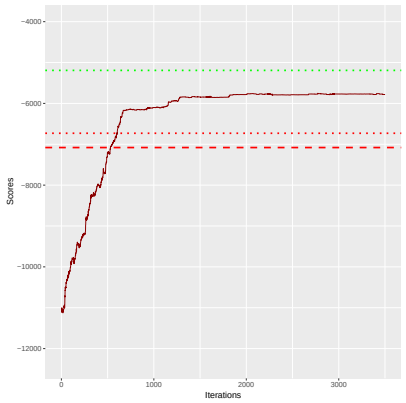


(c) Tamaño del conjunto de padres = 70.

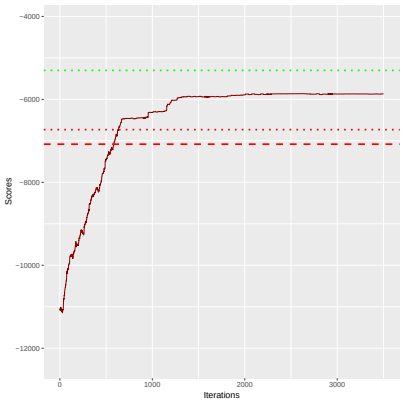


(d) Tamaño del conjunto de padres = 90.

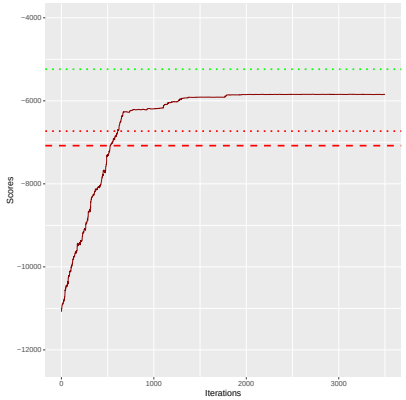
Fig. 5.5: Gráficas de evolución de la métrica según el número de iteraciones, tercer experimento red *alarm* (V3).



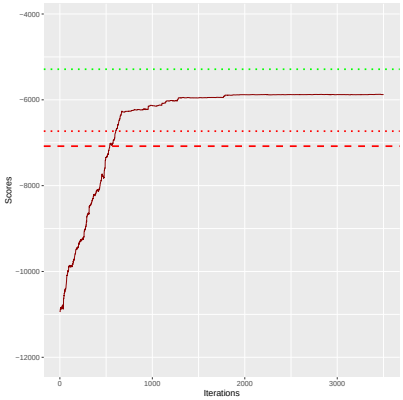
(a) Tamaño del conjunto de padres = 30.



(b) Tamaño del conjunto de padres = 50.

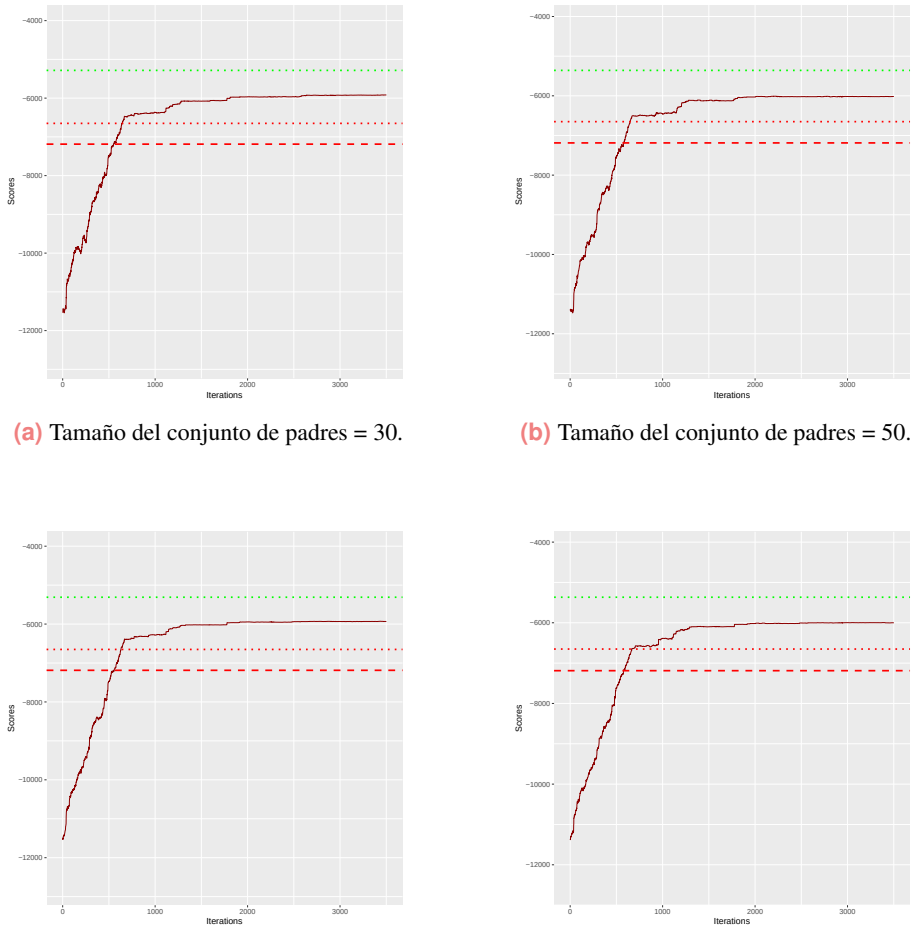


(c) Tamaño del conjunto de padres = 70.



(d) Tamaño del conjunto de padres = 90.

Fig. 5.6: Gráficas de evolución de la métrica según el número de iteraciones, cuarto experimento red *alarm* (V4).



(a) Tamaño del conjunto de padres = 30.

(b) Tamaño del conjunto de padres = 50.

(c) Tamaño del conjunto de padres = 70.

(d) Tamaño del conjunto de padres = 90.

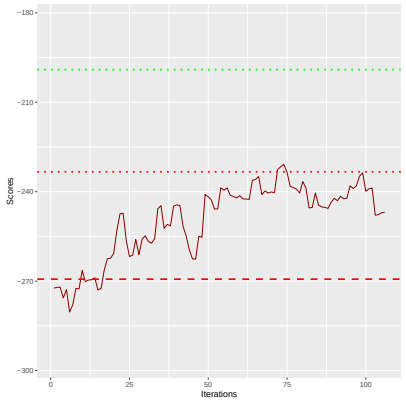
Fig. 5.7: Gráficas de evolución de la métrica según el número de iteraciones, quinto experimento red *alarm* (V5).

5.4.2 Red *asia*

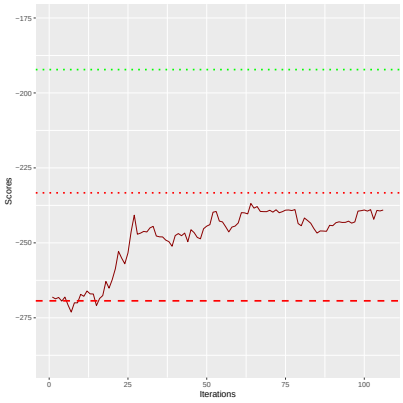
A la vista de la Tabla 5.3 y Figuras 5.8, 5.9, 5.10, 5.11 y 5.12 sobre la red *asia*, podemos decir que los resultados mejoran. En este caso también se produce convergencia, aunque *a priori* parezca que se fluctúa, lo cierto es que se debe a que el rango del eje y es menor que el de las gráficas de la RB *alarm* y por eso se aprecian visualmente los cambios de valor aunque sean pequeños.

	V1		V2		V3		V4		V5	
	train	test	train	test	train	test	train	test	train	test
bnlearn	-269.3	-233.3	-247.5	-234.0	-240.7	-229.3	-274.9	-232.0	-271.5	-229.2
np = 10	-246.9	-199.0	-229.8	-202.9	-220.7	-198.9	-239.1	-197.2	-246.9	-199.0
np = 20	-239.1	**-192.2**	-226.4	-200.9	-216.3	**-195.5**	-237.5	-195.4	-239.1	**-192.2**
np = 30	-244.7	-200.5	**-226.0**	-200.9	-216.7	-198.7	-240.5	**-195.0**	-244.7	-200.5
np = 40	**-238.1**	-199.8	-226.6	**-200.8**	**-214.7**	-198.1	**-235.9**	-198.0	**-238.1**	-199.8

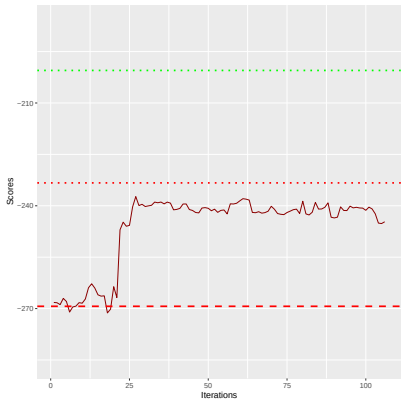
Tab. 5.3: Tabla resumen de los resultados obtenidos para la red *asia* sobre las 5 distintas repeticiones del algoritmo de aprendizaje variacional, considerando distintos tamaños del conjunto de padres.



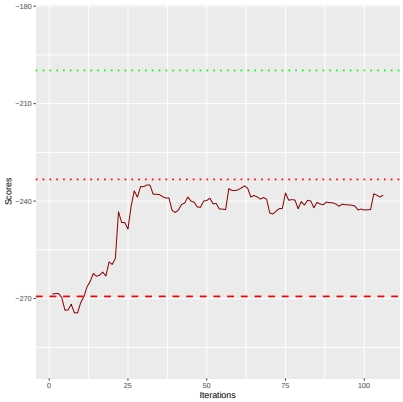
(a) Tamaño del conjunto de padres = 10.



(b) Tamaño del conjunto de padres = 20.

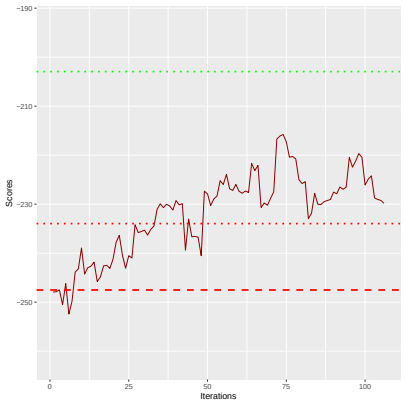


(c) Tamaño del conjunto de padres = 30.

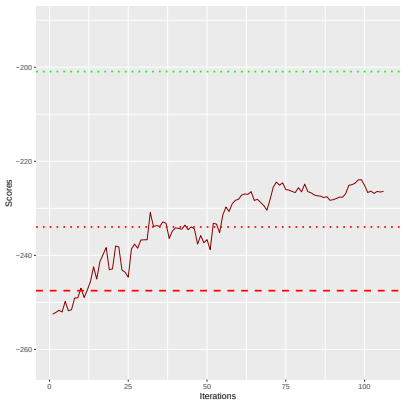


(d) Tamaño del conjunto de padres = 40.

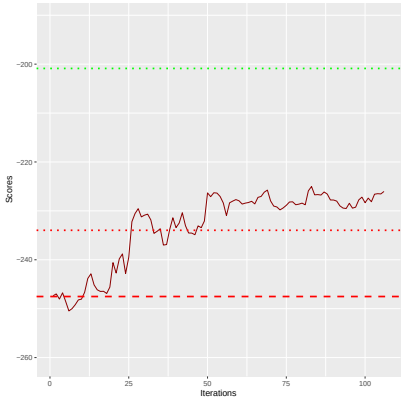
Fig. 5.8: Gráficas de evolución de la métrica según el número de iteraciones, primer experimento red *asia* (V1).



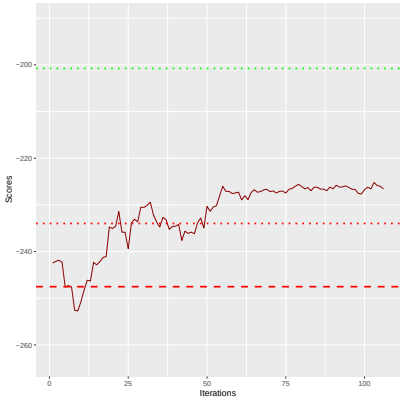
(a) Tamaño del conjunto de padres = 10.



(b) Tamaño del conjunto de padres = 20.

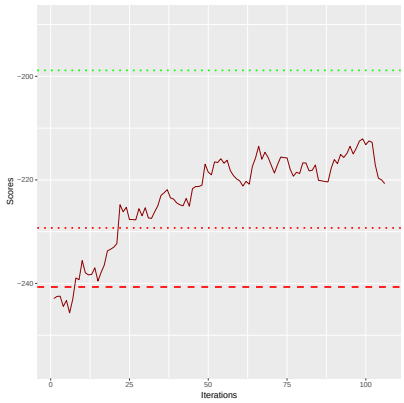


(c) Tamaño del conjunto de padres = 30.

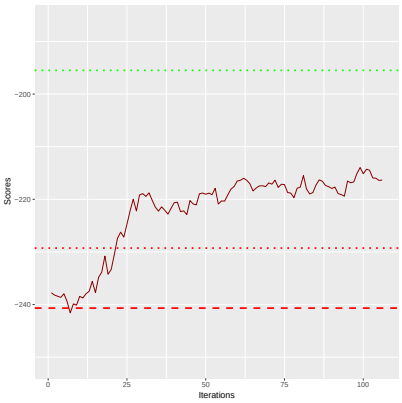


(d) Tamaño del conjunto de padres = 40.

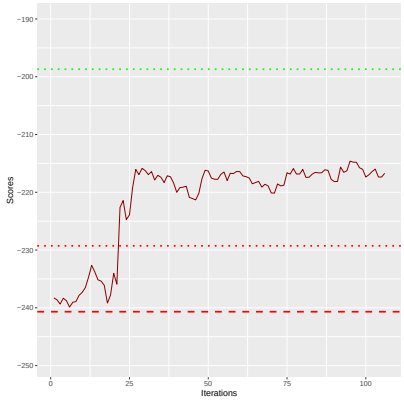
Fig. 5.9: Gráficas de evolución de la métrica según el número de iteraciones, segundo experimento red *asia* (V2).



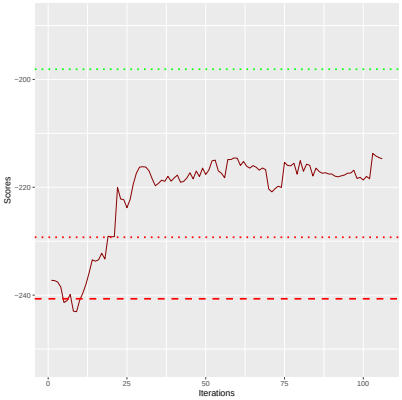
(a) Tamaño del conjunto de padres = 10.



(b) Tamaño del conjunto de padres = 20.

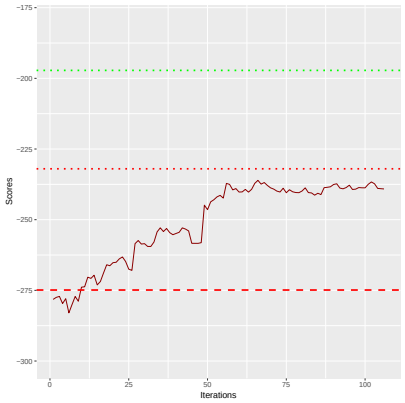


(c) Tamaño del conjunto de padres = 30.

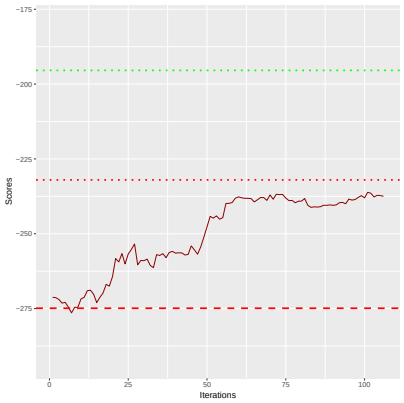


(d) Tamaño del conjunto de padres = 40.

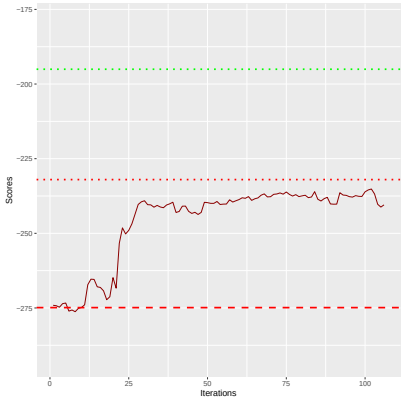
Fig. 5.10: Gráficas de evolución de la métrica según el número de iteraciones, tercer experimento red *asia* (V3).



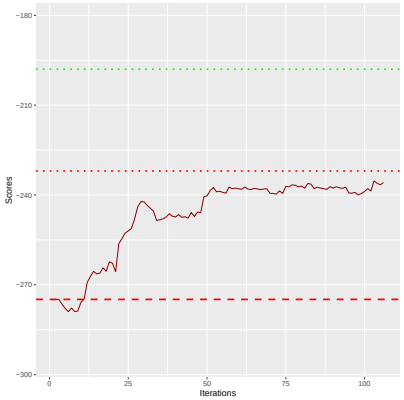
(a) Tamaño del conjunto de padres = 10.



(b) Tamaño del conjunto de padres = 20.

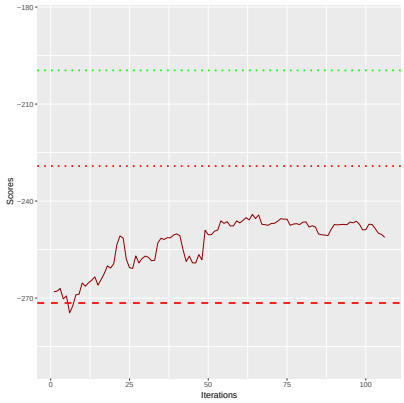


(c) Tamaño del conjunto de padres = 30.

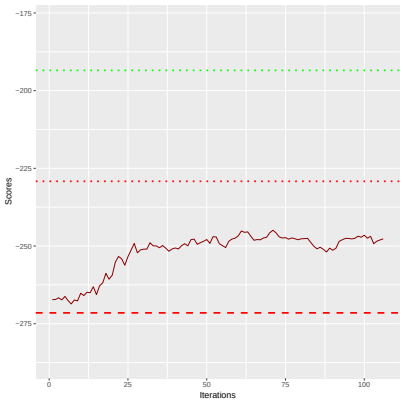


(d) Tamaño del conjunto de padres = 40.

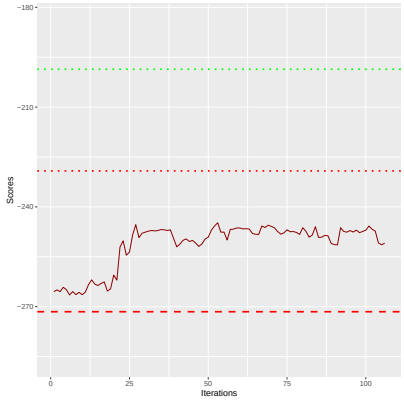
Fig. 5.11: Gráficas de evolución de la métrica según el número de iteraciones, cuarto experimento red *asia* (V4).



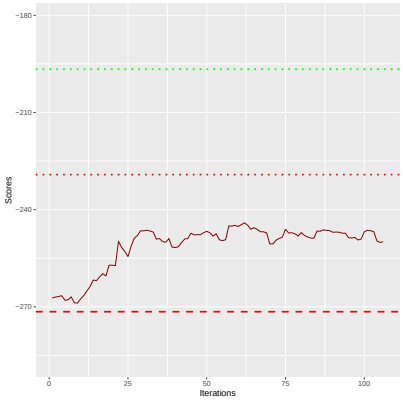
(a) Tamaño del conjunto de padres = 10.



(b) Tamaño del conjunto de padres = 20.



(c) Tamaño del conjunto de padres = 30.



(d) Tamaño del conjunto de padres = 40.

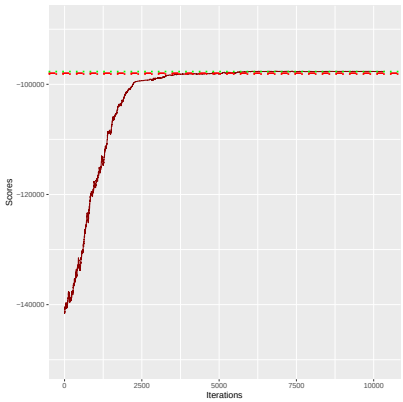
Fig. 5.12: Gráficas de evolución de la métrica según el número de iteraciones, quinto experimento red *asia* (V5).

5.4.3 Red *hepar2*

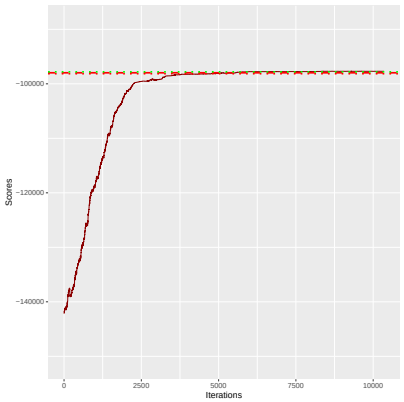
Sobre el comportamiento de la red *hepar2* al aplicar el algoritmo, que podemos observar en la Tabla 5.4 y las Figuras 5.13, 5.14, 5.15, 5.16 y 5.17, podemos comentar que al igual que para el caso de *alarm*, se observa una rápida convergencia en pocas iteraciones y todos los resultados mejoran. Además, no se observa diferencia significativa entre las gráficas según el número de padres, por lo que podrá optarse por elegir el número menor.

	V1		V2		V3		V4		V5	
	train	test	train	test	train	test	train	test	train	test
bnlearn	-97973	-98139	-98361	-98190	-98506	-98115	-98079	-98117	-98614	-98113
np = 25	**-97650**	**-97692**	**-97968**	**-97727**	-98232	-97793	**-97607**	**-97648**	-98258	-97864
np = 50	-97705	-97800	-98054	-97794	**-98097**	**-97704**	-97839	-97858	-98276	**-97697**
np = 75	-97662	-97718	-97998	-97741	-98177	-97776	-97715	-97720	-98319	-97764
np = 100	-97723	-97789	-98046	-97775	-98199	-97815	-97652	-97657	**-98233**	-97790

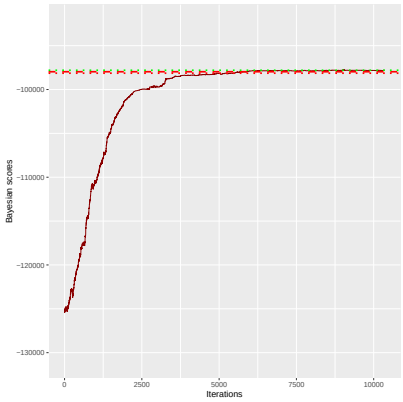
Tab. 5.4: Tabla resumen de los resultados obtenidos para la red *hepar2* sobre las 5 distintas repeticiones del algoritmo de aprendizaje variacional, considerando distintos tamaños del conjunto de padres.



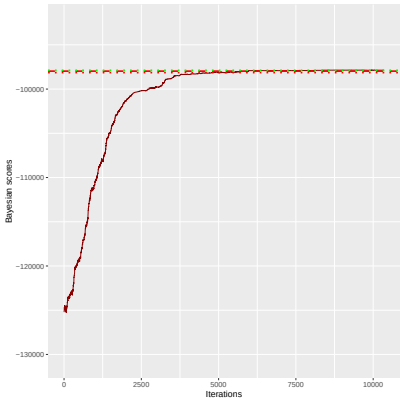
(a) Tamaño del conjunto de padres = 25.



(b) Tamaño del conjunto de padres = 50.

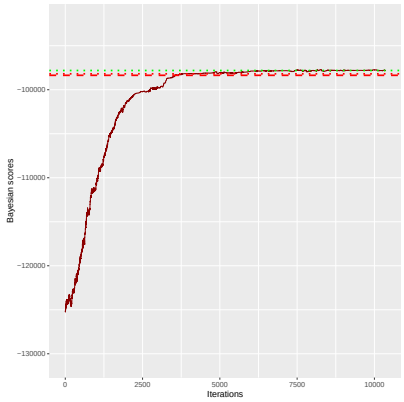


(c) Tamaño del conjunto de padres = 75.

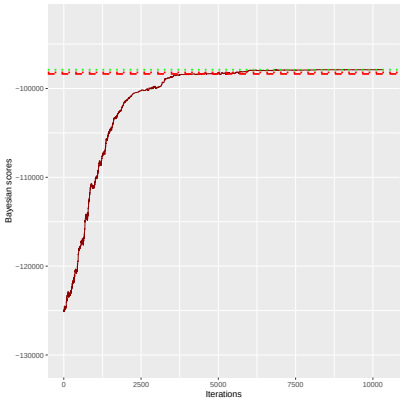


(d) Tamaño del conjunto de padres = 100.

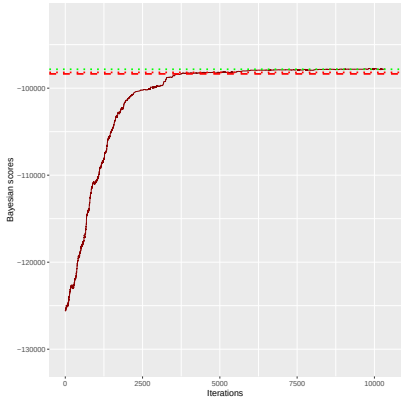
Fig. 5.13: Gráficas de evolución de la métrica según el número de iteraciones, primer experimento red *hepar2* (V1).



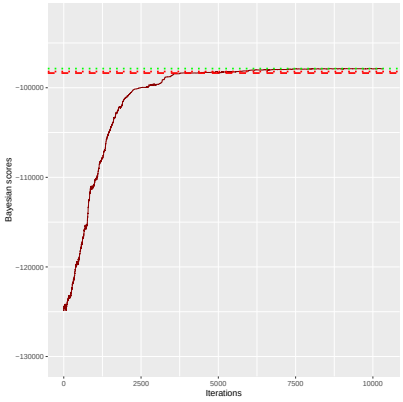
(a) Tamaño del conjunto de padres = 25.



(b) Tamaño del conjunto de padres = 50.

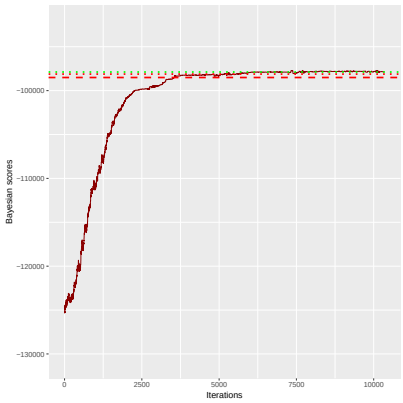


(c) Tamaño del conjunto de padres = 75.

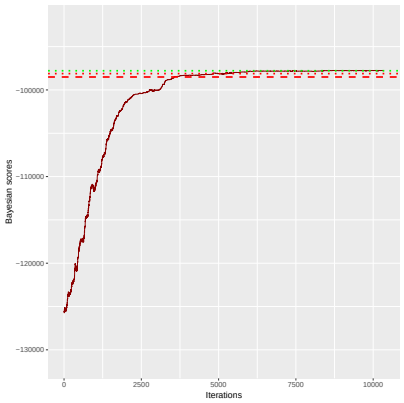


(d) Tamaño del conjunto de padres = 100.

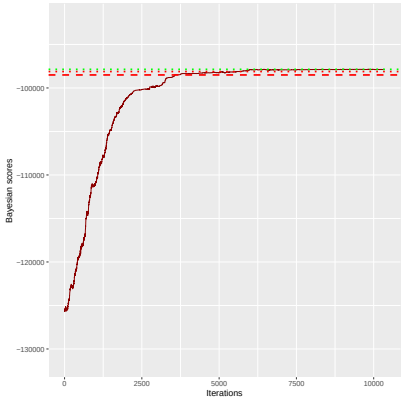
Fig. 5.14: Gráficas de evolución de la métrica según el número de iteraciones, segundo experimento red *hepar2* (V2).



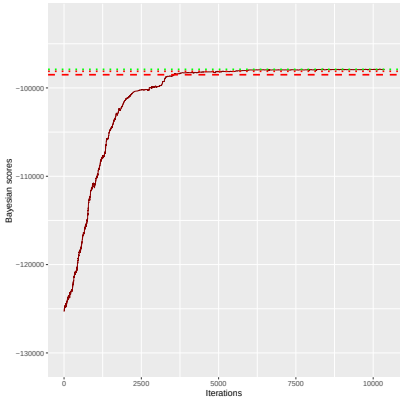
(a) Tamaño del conjunto de padres = 25.



(b) Tamaño del conjunto de padres = 50.

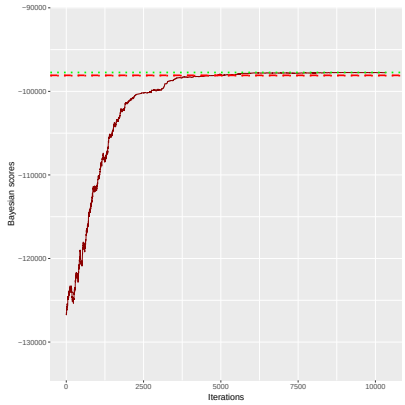


(c) Tamaño del conjunto de padres = 75.

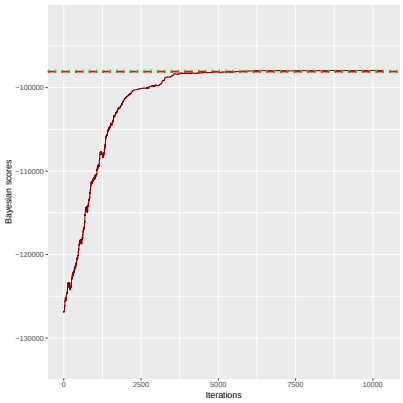


(d) Tamaño del conjunto de padres = 100.

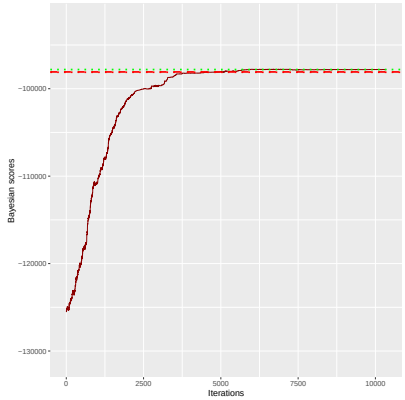
Fig. 5.15: Gráficas de evolución de la métrica según el número de iteraciones, tercer experimento red *hepar2* (V3).



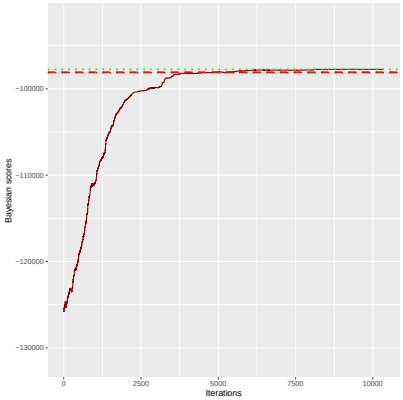
(a) Tamaño del conjunto de padres = 25.



(b) Tamaño del conjunto de padres = 50.

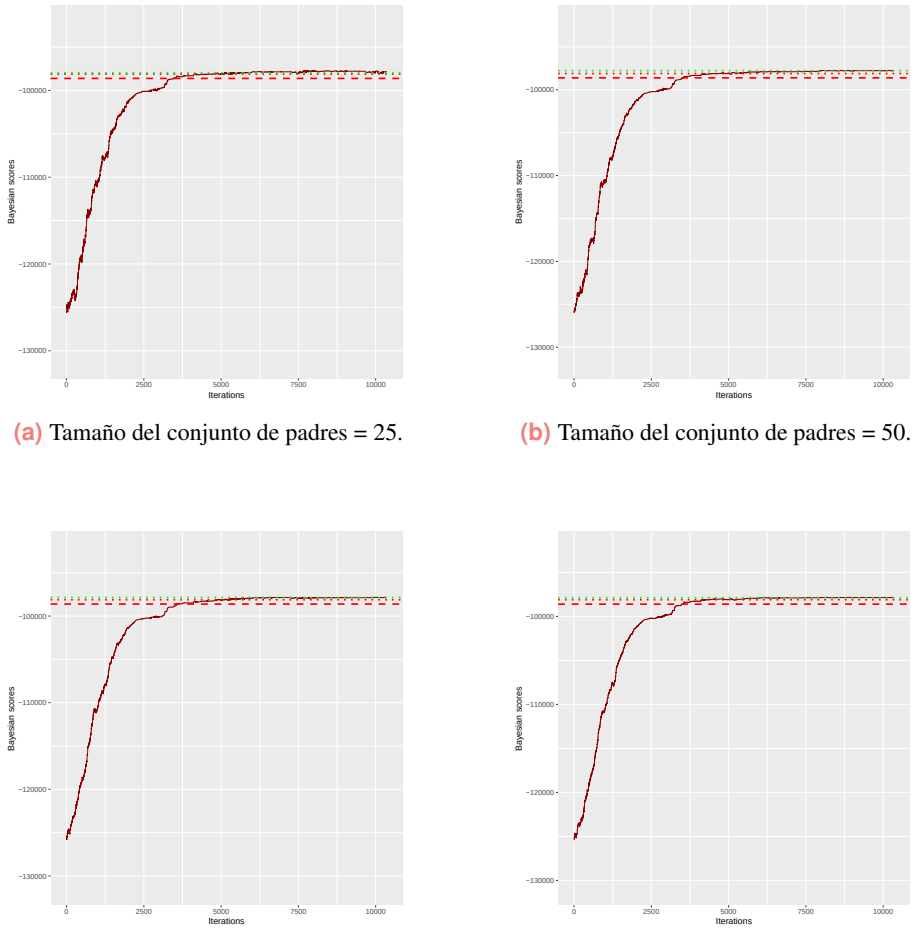


(c) Tamaño del conjunto de padres = 75.



(d) Tamaño del conjunto de padres = 100.

Fig. 5.16: Gráficas de evolución de la métrica según el número de iteraciones, cuarto experimento red *hepar2* (V4).



(a) Tamaño del conjunto de padres = 25.

(b) Tamaño del conjunto de padres = 50.

(c) Tamaño del conjunto de padres = 75.

(d) Tamaño del conjunto de padres = 100.

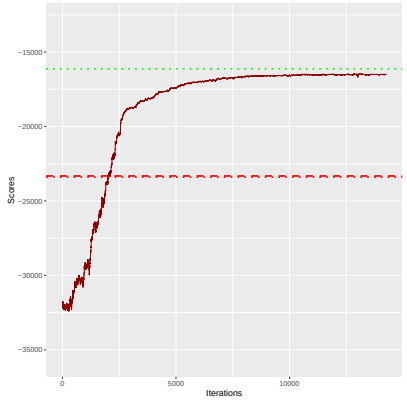
Fig. 5.17: Gráficas de evolución de la métrica según el número de iteraciones, quinto experimento red *hepar2* (V5).

5.4.4 Red *win95pts*

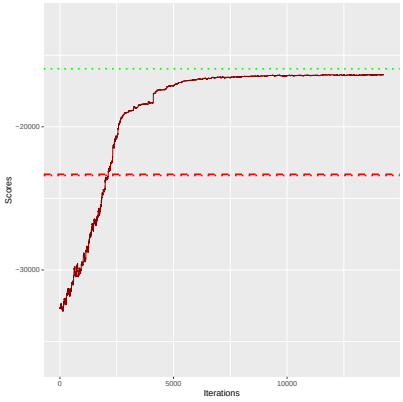
De manera similar a como se observa para las redes *alarm* y *hepar2*, en la Tabla 5.5 y Figuras 5.18, 5.19, 5.20, 5.21 y 5.22, que muestran los resultados obtenidos al aplicar el algoritmo de aprendizaje estructural variacional propuesto, obtenemos convergencia en pocas iteraciones y mejores resultados al aplicar el algoritmo de aprendizaje variacional. Y el cambio de número de padres no parece afectar, por lo que se podrá seleccionar el menor.

	V1		V2		V3		V4		V5	
	train	test	train	test	train	test	train	test	train	test
bnlearn	-23322	-23398	-23413	-23240	-23371	-23498	-23570	-23390	-23506	-23570
np = 25	-16501	-16125	-16470	**-15197**	**-16276**	**-15889**	-16493	-16047	**-16489**	**-15994**
np = 50	-16366	**-15950**	-16472	-15970	-16421	-16038	-16491	-15926	-16497	-16006
np = 75	**-16309**	-16008	-16481	-15921	-16317	-15963	**-16437**	**-15802**	-16519	-16006
np = 100	-16310	-16004	**-16433**	-15852	-16363	-16023	-16733	-16152	-16516	-16051

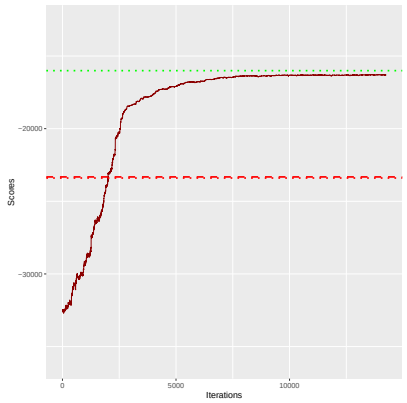
Tab. 5.5: Tabla resumen de los resultados obtenidos para la red *win95pts* sobre las 5 distintas repeticiones del algoritmo de aprendizaje variacional, considerando distintos tamaños del conjunto de padres.



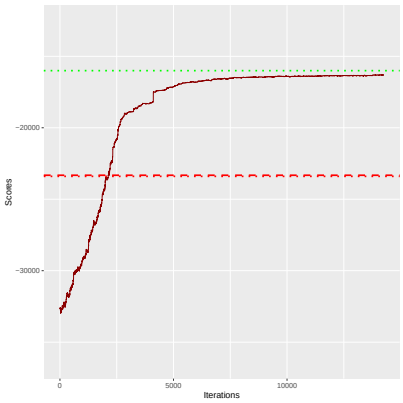
(a) Tamaño del conjunto de padres = 25.



(b) Tamaño del conjunto de padres = 50.

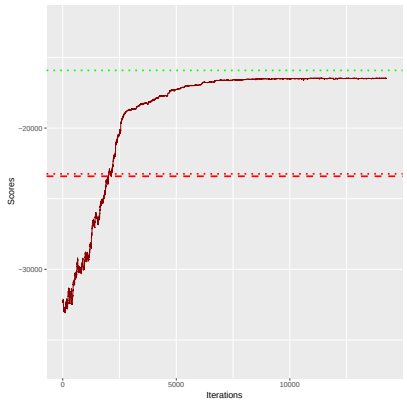


(c) Tamaño del conjunto de padres = 75.

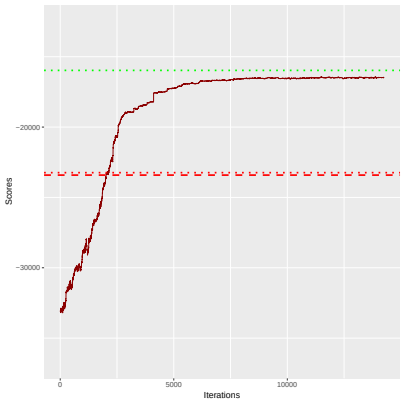


(d) Tamaño del conjunto de padres = 100.

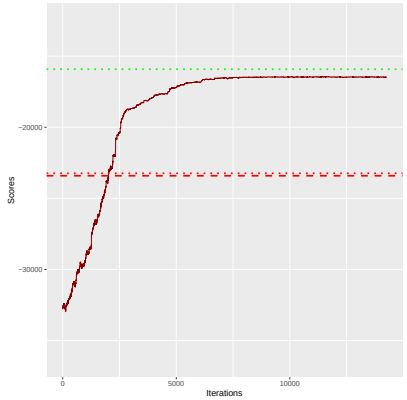
Fig. 5.18: Gráficas de evolución de la métrica según el número de iteraciones, primer experimento red *win95pts* (V1).



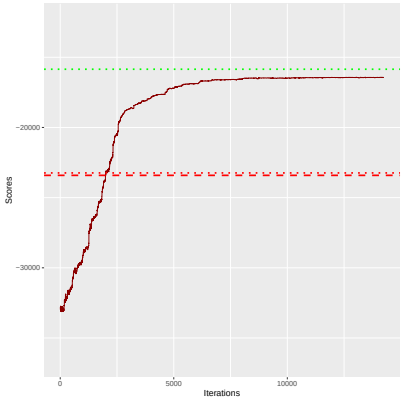
(a) Tamaño del conjunto de padres = 25.



(b) Tamaño del conjunto de padres = 50.

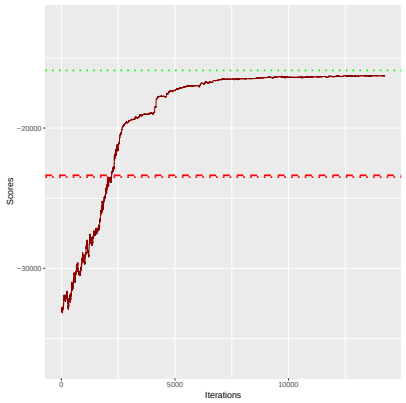


(c) Tamaño del conjunto de padres = 75.

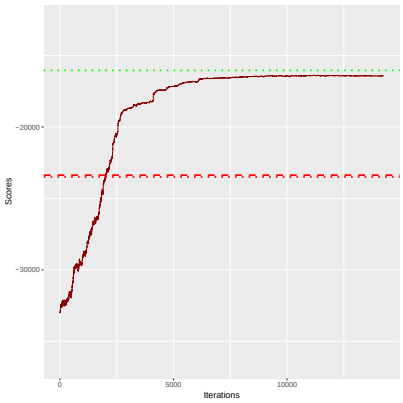


(d) Tamaño del conjunto de padres = 100.

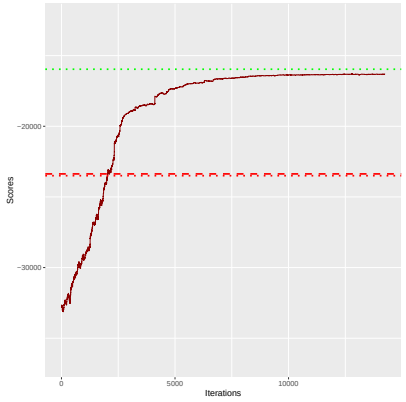
Fig. 5.19: Gráficas de evolución de la métrica según el número de iteraciones, segundo experimento red *win95pts* (V2).



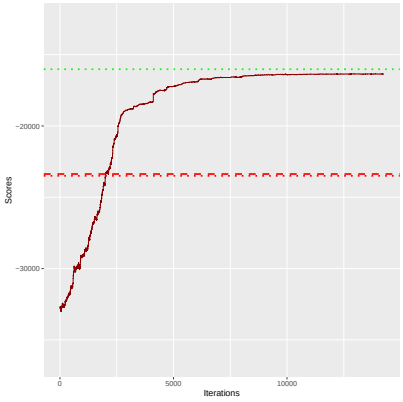
(a) Tamaño del conjunto de padres = 25.



(b) Tamaño del conjunto de padres = 50.

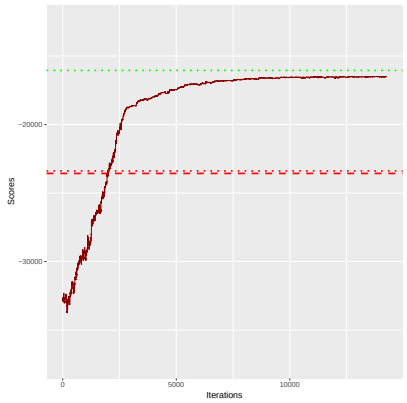


(c) Tamaño del conjunto de padres = 75.

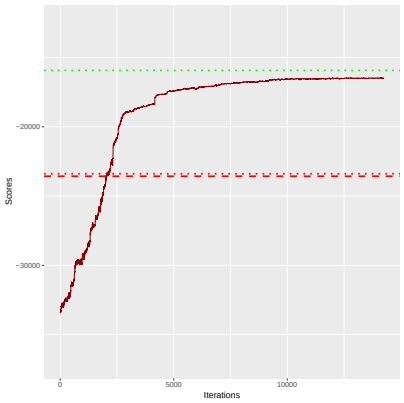


(d) Tamaño del conjunto de padres = 100.

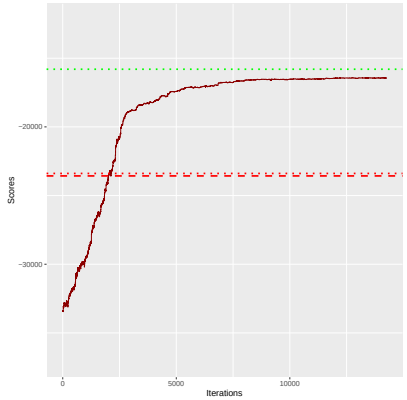
Fig. 5.20: Gráficas de evolución de la métrica según el número de iteraciones, tercer experimento red *win95pts* (V3).



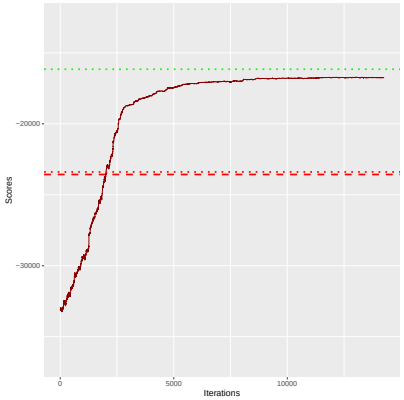
(a) Tamaño del conjunto de padres = 25.



(b) Tamaño del conjunto de padres = 50.

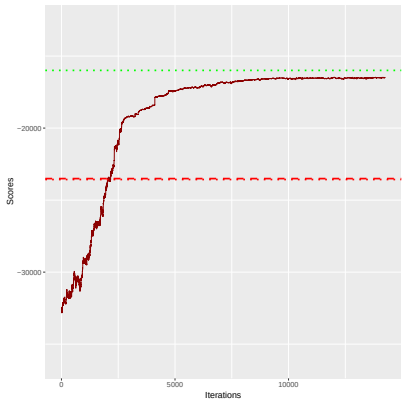


(c) Tamaño del conjunto de padres = 75.

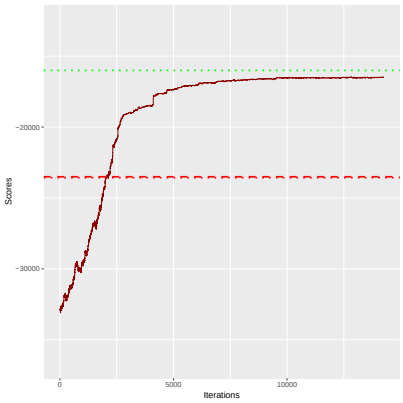


(d) Tamaño del conjunto de padres = 100.

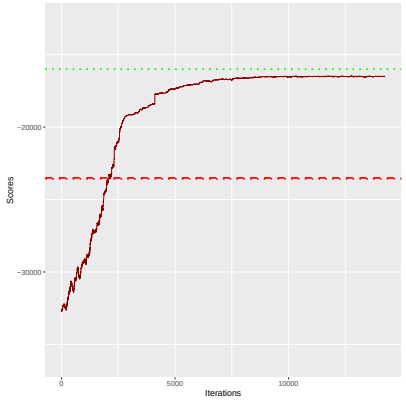
Fig. 5.21: Gráficas de evolución de la métrica según el número de iteraciones, cuarto experimento red *win95pts* (V4).



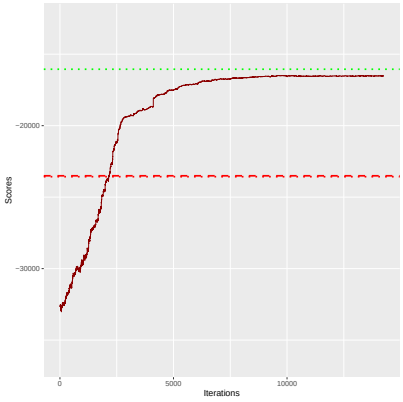
(a) Tamaño del conjunto de padres = 25.



(b) Tamaño del conjunto de padres = 50.



(c) Tamaño del conjunto de padres = 75.



(d) Tamaño del conjunto de padres = 100.

Fig. 5.22: Gráficas de evolución de la métrica según el número de iteraciones, quinto experimento red *win95pts* (V5).

5.5 Conclusiones y comentarios sobre el algoritmo de aprendizaje variacional de RBs

- El método de aprendizaje variacional que hemos presentado supone, de momento, una aproximación. Para llegar a un resultado satisfactorio y que el algoritmo sea capaz de detectar las relaciones condicionales existentes, será necesario que el tamaño del conjunto de datos de entrenamiento sea lo suficientemente grande. En caso de qué así sea, existirá

un grafo dirigido acíclico $\mathcal{G}^*$ que se trate de un óptimo local para la función ELBO 5.27 y una vez que las ecuaciones de actualización alcancen $\mathcal{G}^*$ se quedarán estáticas.

- En cuanto a la experimentación, comentamos que para las cuatro redes Bayesianas estudiadas los resultados han mejorado con respecto a los ofrecidos por *bnlearn*.
- Además, el algoritmo está programado para sacar ventaja de diferentes tareas de paralelización. Las opciones disponibles de paralelización pueden aplicarse a las operaciones indicadas a continuación:

    1. Generación de los conjuntos de padres.
    2. Cálculo de métrica para todas las variables.
    3. Cálculo de entropía.
    4. Cálculo de métrica para una variable.
    5. Generación de los conjuntos de padres + cálculo de entropía.
    6. Cálculo de métrica para todas las variables + cálculo de entropía.
    7. Generación de los conjuntos de padres + cálculo de entropía + cálculo de métrica para una variable.
    8. Generación de los conjuntos de padres + cálculo de métrica para una variable.
    9. Cálculo de entropía + cálculo de métrica para una variable.

    Dentro de estas 9 alternativas de paralelización, la que mejor resultados dio en un estudio inicial fue la *(4) Cálculo de métrica para una variable*, por lo que fue la que se utilizó para todos los experimentos de todas las redes.

6 Algoritmos para inferencia en MGPs con probabilidades imprecisas

En los anteriores Capítulos 2 y 3 se pudo encontrar una introducción a la inferencia con MGPs precisos e imprecisos. Como comentamos, las tareas de inferencia son fundamentales en el trabajo de MGPs en general, y RBs en particular. Con ello, pueden resolverse consultas sobre una o varias variables del problema, a la vista de una evidencia. En este capítulo de la memoria queremos centrarnos en la inferencia sobre redes credales (*i.e.* redes Bayesianas con probabilidades imprecisas), cuya definición podemos ver en la Sección 3.2. En concreto, esta parte de la memoria se basa en el trabajo realizado por Cano *et al.* [36] en el que se introdujeron dos algoritmos para realizar inferencia exacta y aproximada sobre redes credales. Nuestra aportación en esta memoria consistirá en la elaboración de un algoritmo *branch-and-bound* paralelo para realizar inferencia, extensión del descrito en [36].

La distribución del capítulo se distribuye como sigue: la Sección 6.1 explica los dos algoritmos de inferencia aproximada y exacta que se introdujeron en el trabajo de Cano *et al.* [36]. De esta manera se deja asentada la base para el desarrollo del algoritmo *branch-and-bound* paralelo, cuya introducción se lleva a cabo en la Sección 6.2.

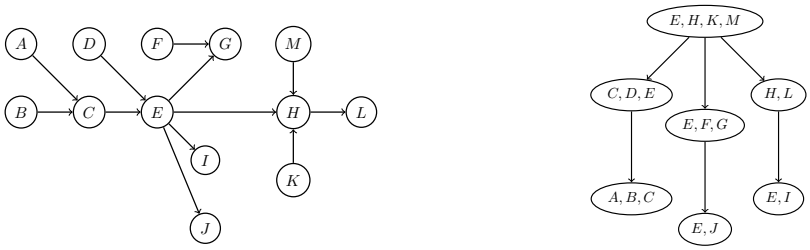
6.1 Algoritmos de partida: *hill-climbing* y *branch-and-bound*

El trabajo de Cano *et al.* [36] considera como paradigma de representación de probabilidades imprecisas los conjuntos convexos de probabilidad, que estudiamos en el Capítulo 3, Sección 3.1.1. Asume, además, una típica hipótesis, la de conjuntos credales separadamente especificados. Esto implica que para cada configuración de los padres, existirá un conjunto credal diferente [55]. En el mismo Capítulo 3 sobre MGPs con probabilidades imprecisas, se recoge información sobre la inferencia con probabilidades imprecisas. Algunos de los algoritmos de inferencia operan

directamente sobre las redes credales [6, 71, 189, 190], aunque los intervalos resultantes son, en general, mayores que los que se obtendrían al realizar cálculos globales con los conjuntos convexos asociados. Por otra parte, realizar cálculos para obtener límites exactos con los conjuntos convexos puede ser bastante ineficiente, ya que equivaldría a propagar todas las probabilidades conjuntas que se obtendrían eligiendo una probabilidad condicional exacta en cada conjunto convexo.

6.1.1 Algoritmo de propagación de Shenoy y Shafer

Puesto que el algoritmo *hill-climbing* de Cano *et al.* [36] se basa en la arquitectura de propagación de algoritmos para redes Bayesianas propuesta por Shenoy y Shafer [182], vamos a proceder a introducirlo. Este algoritmo propaga sobre los denominados *árboles de cliques*. Los árboles de cliques son árboles de probabilidad que modifican el grafo original de manera que los nodos se agrupan en supernodos (o cliques), manteniendo las relaciones de dependencia entre las variables. En la parte derecha de la siguiente Figura 6.1 podemos ver un posible árbol de cliques que representa el grafo de la parte izquierda.



(a) (b)

Fig. 6.1: Ejemplo de grafo dirigido acíclico 6.1(a) y una de sus posibles conversiones en árbol de cliques 6.1(b).

Aunque existen varios algoritmos que inferencia sobre árboles de cliques [114, 124, 181, 182], en la propuesta de Shenoy y Shafer [182] a cada clique C_i se le asigna un potencial, Ψ_{C_i}, que inicialmente es la función identidad. Para cada par de cliques adyacentes C_i, C_j existirán dos mensajes: (1) $M_{C_i \to C_j}$: que será el mensaje que el nodo C_i manda al C_j; y (2) $M_{C_j \to C_i}$: con la dirección opuesta. Las distribuciones condicionadas de probabilidad p_i y las observaciones δ_i, serán asignadas al nodo que contiene todas las variables relacionadas. Y cada nodo contendrá un conjunto de distribuciones condicionadas de probabilidad (pudiendo ser vacío) y de funciones de Dirac (asociadas a las observaciones), que deberán ser combinadas en el potencial Ψ_{C_i}.

El algoritmo se lleva a cabo recorriendo el árbol de cliques en dos direcciones (de nodos hojas a nodos raíz y viceversa). En el proceso los mensajes se irán actualizando siguiendo la Ecuación 6.1

$$M_{C_i \to C_j} = \left(\Psi_{C_i} \cdot \left(\prod_{C_k \in Ady(C_i, C_j)} M_{C_k \to C_i} \right) \right)^{\downarrow C_i \cap C_j} \tag{6.1}$$

donde $Ady(C_i, C_j)$ representa el conjunto de nodos adyacentes a C_i distintos de C_j y Ψ_{C_i} es la combinación de todos los potenciales asociados a C_i.

Cuando la propagación se haya efectuado, puede procederse a calcular una probabilidad *a posteriori* deseada. Supongamos que buscamos calcular $p(X_q|x_E)$, donde x_E son los valores observados de las variables evidencia X_E. Entonces, para calcular la probabilidad habrá que buscar un clique C_i que contenga a la variable X_q y normalizar la expresión:

$$p(X_q, x_E) = \left(\Psi_{C_i} \cdot \left(\prod_{C_j \in Ady(C_i)} M_{C_j \to C_i} \right) \right)^{\downarrow X_q} \tag{6.2}$$

donde por $Ady(C_i)$ entendemos el conjunto de nodos adyacentes a C_i. El factor de normalización es la probabilidad de la evidencia dada y puede ser calculada de la evaluación de la Ecuación (6.2) usando:

$$p(x_E) = \sum_{x_q^i} p(x_q^i, x_E) \tag{6.3}$$

De hecho, podemos calcular la probabilidad de la evidencia, $p(\mathbf{x}_E)$, eligiendo cualquier clique C_i y combinando todos los mensajes que vayan hacia C_i con el potencial Ψ_{C_i} y sumando todas las variables en C_i.

6.1.2 Algoritmo *hill-climbing*

En general, los algoritmos *hill-climbing* (*i.e.* de escalada simple, o ascenso de colinas) son técnicas de optimización matemática de la familia de algoritmos de búsqueda local. Suponen un proceso iterativo donde se parte de una solución inicial, para a continuación modificar los elementos uno a uno y comprobar si la solución al problema mejora todas las que se encontraron hasta el momento. El proceso se repite hasta que se cumple el criterio de parada (que no puedan encontrarse mejoras, ejecutar un número de veces, límite en el tiempo de cómputo, etc.).

El objetivo del algoritmo *hill-climbing* de Cano *et al.* [36] es el de calcular las probabilidades inferior o superior *a posteriori* para el estado de una variable, supongamos x_q^i de la variable

X_q, dada una evidencia. Esto puede resolverse seleccionando la configuración de las variables transparentes que produzca los valores menor o mayor para $p(x_q^i|x_E)$. Notando como p_{t_s} a la distribución de probabilidad global determinada por la configuración de variables transparentes t_s, el problema se traduce en encontrar $max_{t_s} p_{t_s}(X_q = x_q^i|x_E)$ o $min_{t_s} p_{t_s}(X_q = x_q^i|x_E)$. Y el resultado de ejecutar el método será el límite interior (superior o inferior) para $P(X_q = x_q^i)$ en la extensión fuerte de una red credal. Encontrar un límite interior implica que si buscamos el límite superior, el valor resultado será menor que el valor real, y mayor en el caso de buscar el inferior.

En primer lugar, será necesario transformar el DAG en un árbol de cliques, al igual que en el algoritmo de inferencia de Shafer y Shenoy [182] que describimos en el apartado anterior (Sec. 6.1.1) y como el que se puede observar en la Figura 6.1. Se necesitará un sistema de paso de mensajes doble. De manera que para cada par nodos conectados, C_i y C_j, tendremos dos mensajes que irán de C_i a C_j (que notamos como $M^1_{C_i \to C_j}$ $M^2_{C_i \to C_j}$) y otros dos en la dirección opuesta (notados como $M^1_{C_j \to C_i}$ y $M^2_{C_j \to C_i}$). Estos mensajes se calcularán de la siguiente manera:

- $M^1_{C_j \to C_i}$ se calcularán según se indica en la Fórmula 6.1.
- $M^2_{C_j \to C_i}$ se calcularán según se indica en la Fórmula 6.1, pero asumiendo que $X_q = x_q^i$ se añade a la evidencia x_E.

Una vez obtenido el árbol de cliques, cada conjunto credal $H^{X_i|Y}$ (representado como un árbol de probabilidad similar al que vimos en el ejemplo 28) se asigna a un clique.

Dada una observación $X_i = x_i^j$, su potencial ϕ_{X_i} se combina con el potencial Ψ_{C_i} que contiene a X_i. Puede obtenerse el mismo resultado al restringir Ψ_{C_i} a $X_i = x_i^j$ si $X_i \in C_i$. Entonces, el potencial Ψ_{C_i} para cada C_i se obtiene como combinación de todos los potenciales asignados a C_i para, a continuación, restringir esta combinación a X_E.

El árbol que representa la red credal contendrá un conjunto de variables transparentes, de manera similar a como puede observarse en la Figura 3.7. Es decir, que cada conjunto $H^{X_i|Y}$ usa un conjunto de variables transparentes (una para cada configuración de Y). De esta forma, el algoritmo *hill-climbing* busca seleccionar la configuración de variables transparentes de la red credal, tal que esa configuración hace que se maximice o minimice $P_t(X_q = x_q^i|x_E)$. Suponemos que en un momento concreto del proceso de inferencia por ascensión de colinas, las variables transparentes se encuentran en una configuración t. Esta configuración t se irá modificando en el paso *hill-climbing* para optimizar $p_t(X_q = x_q^i|x_E)$. Se selecciona una configuración aleatoria inicial t_0 para el conjunto de variables transparentes T. Esta configuración inicial de T se agrega al conjunto de evidencia X_E. Cada árbol de probabilidad Ψ_{C_i} se restringe por tanto de acuerdo

al nuevo conjunto de observaciones. Por ejemplo, en 3.7, si la configuración inicial de T contiene $T_{y_i} = t^1_{y_1}$ y $T_{y_2} = t^2_{y_2}$, entonces el potencial Ψ_{C_i} que contiene $H^{X_i|Y}$ se restringirá, dando como resultado el árbol que podemos observar en la figura 6.2:

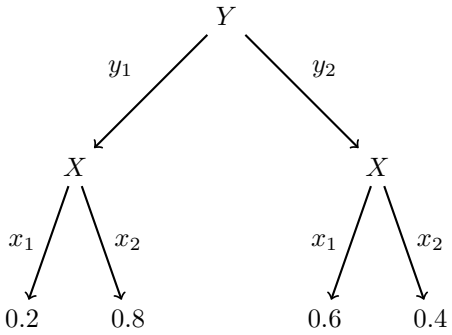


Fig. 6.2: Árbol restringido para $H^{X|Y}$ dado que $T_{y_i} = t^1_{y_1}$ y $T_{y_2} = t^2_{y_2}$.

Con esto se consigue que el potencial Ψ_{C_i} no contenga ninguna variable transparente, todas se encontrarán observadas y desaparecerán de los árboles de probabilidad al restringir a los valores observados.

El algoritmo hace una propagación inicial usando el sistema de mensajes doble y enviando mensajes de los nodos hojas al nodo raíz. Los mensajes se calculan como comentamos anteriormente.

Tras la propagación inicial, comenzamos el paso *hill-climbing*. Se recorre el árbol de cliques de los nodos hojas al nodo raíz y viceversa. Este paso terminará tras ejecutar un número de repeticiones o hasta que el algoritmo no mejore la mejor solución que ya se haya encontrado. Cada vez que pasamos por un nodo C_i, maximizamos localmente $p(X_q = x^i_q|x_E)$ seleccionando un estado para una de las variables transparentes en C_i. Las variables transparentes se mejoran una a una. El proceso termina con ese nodo cuando no hay más variables transparentes que mejorar.

Veamos el pseudocódigo del algoritmo:

Algoritmo 16 *Hill-climbing* sobre redes credales.

Entrada Una red credal $\mathcal{N}_0$

Salida El valor de $\overline{p} = maxp(x_q^i|x_E)$

1: Representar cada conjunto credal condicional $H^{X_i|Y}$ utilizando un árbol de probabilidad $\mathcal{T}$.
2: Construir un árbol de cliques que represente la red credal.
3: Asociar cada conjunto credal condicional $H^{X_i|Y}$ (un árbol de probabilidad) con un nodo C_i del árbol de cliques.
4: **para** cada árbol de probabilidad $\mathcal{T}$ disponible **hacer:**
5: Incorporar la evidencia x_E usando la operación de restricción sobre $\mathcal{T} : \mathcal{T}^{R(x_E)}$.
6: **fin bucle**
7: **para** cada C_i en el árbol de cliques **hacer:**
8: Calcular Ψ_{C_i} combinando todos los árboles de probabilidad asociados al nodo C_i.
9: **fin bucle**
10: Elegir una configuración inicial aleatoria t_0 para el conjunto de variables transparentes T.
11: Incorporar la evidencia t_0 a todos los árboles de probabilidad en el árbol de cliques usando la restricción de operación.
12: Llevar a cabo una propagación inicial en el árbol de cliques utilizando un sistema de mensajes doble:

- Mensajes $M^1_{C_i \to C_j}$ se utilizan para propagar la evidencia dada x_E y la configuración actual para T: permitiendo calcular $p(x_E)$.
- Mensajes $M^2_{C_i \to C_j}$ propagan x_E, la configuración actual para T y $X_q = x_q^i$: permiten calcular $p(x_q^i, x_E)$.

13: **para** cada paso de 1 a m **hacer:**
14: Recorrer el árbol de cliques de raíz a hojas y viceversa.
15: **para** cada nodo visitado C_i y variable transparente T_j en Ψ_{C_i} **hacer:**

- Calcular $p(x_E)$ y $p(x_q^i, x_E)$ para cada $t_j^i \in \Omega_{T_j}$.
  1. Descartar la observación actual para T_j.
  2. Calcular valores para $p(x_E)$ y $p(x_q^i, x_E)$ para cada $t_j \in \Omega_{T_j}$:

$$p(x_E, T_j) = \left(\Psi_{C_i} \cdot \left(\prod_{C_j \in Ady(C_i)} M^1_{C_j \to C_i} \right) \right)^{\downarrow T_j}$$

$$p(x_q^i, x_E, T_j) = \left(\Psi_{C_i} \cdot \left(\prod_{C_j \in Ady(C_i)} M^2_{C_j \to C_i} \right) \right)^{\downarrow T_j}$$

- De los vectores $p(x_E, T_j)$ y $p(x_q^i, x_E, T_j)$ calcular un nuevo vector:

$$p(x_q^i|x_E, T_j) = \frac{p(x_q^i, x_E, T_j)}{p(x_E, T_j)}$$

- Elegir el caso $t_j^i \in \Omega_{T_j}$ que maximice $p(x_q^i|x_E, T_j)$.
- $T_j = t_j^i$ es añadido a la configuración de variables transparentes T.

16: **fin bucle**
17: **fin bucle**
18: Devolver $\overline{p}$ como el mayor $p(x_q^i|x_E, T_j)$ encontrado en los m pasos.

6.1.3 Algoritmo *branch-and-bound*

El algoritmo *branch-and-bound* busca resolver problemas de optimización. A grandes rasgos, este algoritmo se puede visualizar como un árbol de búsqueda en el que en cada nodo hoja representa una posible solución al problema. El método siempre parte de una solución inicial. Al acceder a una nueva rama se realiza una estimación de la mejor solución que puede obtenerse por ese camino, entonces se pueden producir dos situaciones posibles:

- Si la estimación no es mejor que la mejor solución encontrada hasta el momento, se poda la rama y el conjunto de soluciones que allí se encuentren no se considerarán como solución definitiva. Esto provoca que no sea necesario explorar el árbol de búsqueda completo.

- En caso contrario, se continúa explorando esta rama. De manera que en la siguiente ramificación se vuelve a estimar la mejor solución y se desecha o elige una de las ramas.

Recorrer una rama por completo significará que esa solución es la mejor que se ha encontrado hasta ese punto del algoritmo, por lo que la variable que almacena el mejor valor será actualizada y se continuará con el resto de ramas no exploradas hasta que se hayan estudiado todas. La solución al algoritmo será, por tanto, la última.

En el trabajo de Rocha y Cozman [55], se introduce un algoritmo para realizar inferencia exacta sobre poliárboles; el algoritmo *branch-and-bound* de Cano *et al.* [36] será similar a este, pero puede ser utilizado para todo tipo de grafos. Este algoritmo parte de una solución inicial que se halla mediante el método *hill-climbing* que acabamos de describir (Sec. 6.1.2), lo que reduce el tiempo de cómputo.

Al igual que en el método *hill-climbing* que acabamos de explicar (Sec. 6.1.2), al ejecutar este algoritmo hallaremos las probabilidades inferior o superior *a posteriori* para el estado x_q^i de la variable X_q; pero esta vez la solución generada no es aproximada, sino exacta. Este algoritmo parte de una red credal $\mathcal{N}_0$ que se obtiene eliminando las variables de $\mathcal{N}$ que no se usarán para la inferencia. El algoritmo *branch-and-bound* requiere establecer una cota r para la solución. De este modo, si buscamos obtener $p(x_q^i|x_E)$, entonces r debe producir una estimación tal que $r(\mathcal{N}) \geq max(p(x_q^i|x_E))$. Para ello, Rocha *et al.* [55] utilizan el algoritmo A/R de Tessem [189], aplicable a poliárboles. Los pasos básicos del algoritmo son los expresados en el Algoritmo 17.

Algoritmo 17 *Branch-and-bound* sobre redes credales representadas por poliárboles.

Entrada Una red credal $\mathcal{N}_0$

Salida El valor de $\overline{p} = max\ p(x_q^i|x_E)$

Inicializar $\widehat{p}$ con $-\infty$.
si la red credal $\mathcal{N}_0$ contiene un solo vértice **entonces:**
    Actualizar $\widehat{p}$ con $p(x_q^i|x_E)$ si $p(x_q^i|x_E) > \widehat{p}$.
sino
    Usando las k posibilidades de uno de los conjuntos credales $\mathcal{N}_0$, obtener una lista de k redes credales $\mathcal{N}_{0_1}, ..., \mathcal{N}_{0_k}$ de $\mathcal{N}_0$.
    para cada $\mathcal{N}_{0_h}$ **hacer:**
        si $r(\mathcal{N}_{0_h}) > \widehat{p}$ **entonces:**
            Llamar de manera recursiva primero profundidad sobre $\mathcal{N}_{0_h}$.
        fin condicion
    fin bucle
fin condicion
Establecer el último $\widehat{p}$ como $\overline{p}$.

Al igual que el algoritmo *hill-climbing*, este algoritmo usa variables transparentes en cada conjunto credal condicionado $H^{X_i|Y}$; y usa de nuevo una variable transparente por cada configuración de las variables Y. El algoritmo *branch-and-bound* llevará, por tanto, la búsqueda de la configuración de transparentes t que maximiza o minimiza $P(x_q|x_E)$.

El árbol de búsqueda relativo tendría como nodos raíz e interiores las distintas variables transparentes; donde de cada una de ellas saldrán tantos arcos como estados tenga la variable. Cada nodo hoja define una RB que se obtiene al fijar cada una de las variables transparentes a uno de sus estados.

Ejemplo 44 *Supongamos que la red credal necesita tres variables transparentes (T_1, T_2 y T_3), y que cada una de ellas puede tomar dos estados. Entonces el árbol de búsqueda sería algo como lo que presentamos en la siguiente Figura 6.3.*

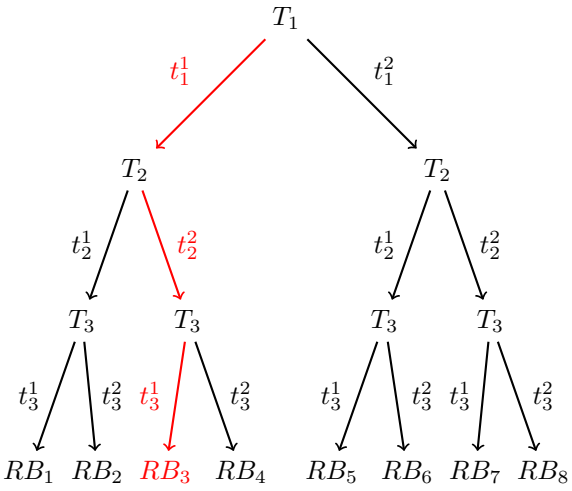


Fig. 6.3: Árbol de búsqueda.

En rojo, podemos ver por ejemplo que si fijamos $T_1 = t_1^1$, $T_2 = t_2^2$ *y* $T_3 = t_3^1$, *obtenemos entonces una de las redes Bayesianas que hemos notado como* $RB3$.

□

Una vez que se tiene el árbol de búsqueda, se hace una búsqueda en profundidad, de manera que se explora la mejor solución que puede obtenerse para las ramificaciones y se podan aquellas ramas cuya solución no va a mejorar a la que hay definida hasta el momento. Para realizar esta poda se calcula una cota $r(N_o)$ mediante algún método aproximado que genere soluciones rápidas. El algoritmo de Rocha *et al.* [55] realiza inferencia utilizando el método de A/R de Tessem [189] sobre la RB que se obtiene al fijar las variables transparentes.

Cano *et al.* [36] presentan un algoritmo *branch-and-bound* muy similar al que acabamos de introducir pero con algunas diferencias:

- La inicialización de $\widehat{p}$ se hace mediante el algoritmo *hill-climbing* que se introduce en el mismo trabajo de Cano *et al.* [36]. Esto produce un ahorro en tiempo, ya que la solución inicial se acerca más a la real.

- Utiliza el algoritmo de eliminación de variables para hacer la inferencia sobre nodos hoja, en su versión para árboles de probabilidad [41].

- Utiliza también una versión (la explicada en la Sección 3.2.1) sobre árboles de probabilidad del algoritmo de eliminación de variables para calcular las cotas aproximadas $r(\mathcal{N}_0)$ en nodos internos. Este método también se usa sobre las operaciones de marginalización y combinación para hacer que los árboles de probabilidad se mantenga en un tamaño que permita trabajar con ellos.

6.2 Algoritmo *branch-and-bound* paralelo

Una vez que hemos explicado el algoritmo *branch-and-bound* que se definió en el trabajo de Cano *et al.* [36], procedemos a partir de él para añadir una mejora. En esta parte de la memoria pretendemos presentar una introducción a cómo se realizaría este mismo algoritmo de un modo paralelo, de manera que los recursos computacionales se utilicen de un modo más efectivo y esto suponga un ahorro en tiempo de cómputo para las tareas de inferencia sobre redes credales. Para ello se recurre a un *pool* de hebras, donde cada hebra se ocupará de explorar parte del árbol de búsqueda.

El *pool* de hebras recibe un conjunto de tareas en paralelo de la aplicación y es capaz de organizar las distintas hebras; primero en la cola de tareas, para después asignar una nueva tarea cuando una de las hebras haya terminado con la anterior. En un momento concreto, cada hebra del *pool* se encuentra ejecutando una tarea pendiente. Cuando una tarea finaliza su ejecución, la hebra que la ejecuta queda libre para ejecutar otra de las que se encontraban pendientes. En la Figura 6.4 esquematizamos el funcionamiento del *pool* de hebras. Notamos a las hebras como h_i.

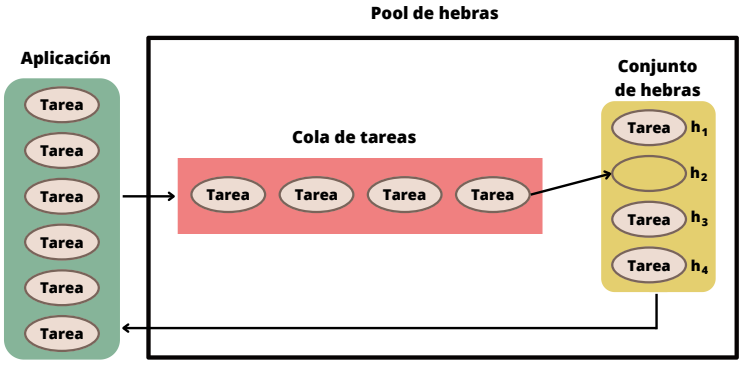


Fig. 6.4: Funcionamiento de *pool* de hebras.

En este algoritmo *branch-and-bound* paralelo buscamos, al igual que en los algoritmos *hill-climbing* y **branch-and-bound** no paralelo, calcular una cota superior o inferior para $P(X_q = x_q^i|x_E)$. Usaremos un *pool* de hebras con un determinado número de hebras k, lo que permite ejecutar hasta k tareas en paralelo. Supongamos el mismo árbol de búsqueda que tenemos en Fig. 6.3. Cada hebra (tarea) explorará una parte del árbol de búsqueda. Las hebras se distribuirían de manera similar a como podemos ver en la Figura 6.5.

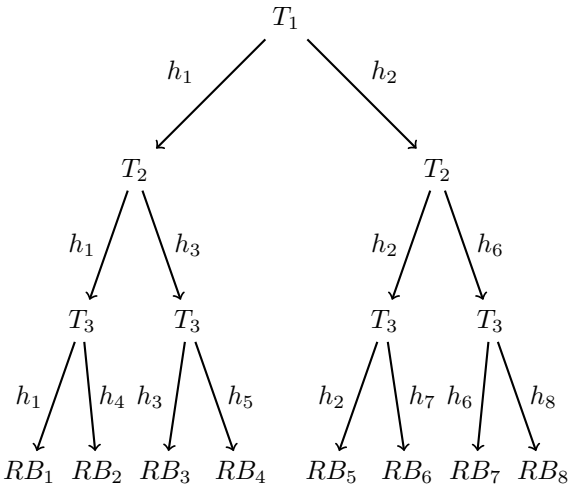


Fig. 6.5: Árbol de búsqueda con las distintas hebras.

Ejemplo 45 *Hemos supuesto que hay suficientes hebras (8 en este caso) y cada una puede ocuparse de realizar las tareas de inferencia sobre una RB. En el caso de haber menos, las hebras se distribuirán del mismo modo que en Fig. 6.5 y cuando se alcance el número máximo, el proceso se quedará esperando hasta que una hebra se encuentre disponible para seguir por ella.*

Se comenzará utilizando una de las hebras (h_1), que parte de una de las variables transparentes (T_1, por simplicidad). Esta misma hebra continuará trabajando fijando T_1 a su primer estado. La siguiente hebra (h_2) partirá de fijar T_1 en su segundo estado. A continuación, la hebra h_1 considerará el estado t_1^1 para T_1 y t_2^1 para T_2, mientras que h_3 considerará que T_2 se fija en el segundo de sus estados. Se sigue este proceso sucesivamente hasta que no haya más hebras disponibles o hasta que todas ellas hayan fijado cada variable transparente y, por tanto, determinen una red Bayesiana. Las hebras trabajarán en paralelo y compartirán una variable común ($\overline{p}$), que al comienzo del algoritmo será la solución inicial dada por *hill-climbing* y guardará la mejor solución encontrada hasta el momento. Al comienzo de la tarea, la hebra calculará la mejor

solución que puede obtenerse en esa ramificación mediante el mismo algoritmo de eliminación de variables que en la versión no paralela de Cano *et al.* [36]. Cuando una hebra termine de ejecutarse comprobará si la mejor solución obtenida es mayor que $\overline{p}$, en este caso actualizará $\overline{p}$ a esa solución. A continuación presentamos este método *branch-and-bound* paralelo de inferencia exacta sobre redes credales. Tres algoritmos formar parte del método que definiremos mediante refinamiento progresivo, de manera que se introduce primero la función más general (*propagate*) que utiliza la siguiente, para continuar por la segunda (*branchAndBound*) que requiere de la tercera (*branchAndBoundRecur*.

Algoritmo 18 Propagación de evidencia sobre red credal

1: **funcion** propagar($\mathcal{N}$) ▷ Donde $\mathcal{N}$ es la red credal.
2: Calcular valor aproximado para $P(x_q^1|x_E)$ mediante *hill-climbing*.
3: Inicializar $\widehat{p}$ con el valor dado por *hill-climbing*. ▷ Donde $\widehat{p}$ es la variable común que almacena la mejor solución encontrada hasta el momento.
4: Llamar a branchAndBound($\mathcal{N}_0$) ▷ Donde $\mathcal{N}_0$ es la red credal que se obtiene eliminando de $\mathcal{N}$ las variables no relacionadas.
5: El valor calculado para $P(x_q^1|x_E)$ será el último valor que tenga la variable compartida $\widehat{p}$.
6: **fin funcion**

Algoritmo 19 *Branch-and-bound*

1: **funcion** branchAndBound($\mathcal{N}_0$)
2: Crear una tarea *task* que ejecutará el método *branch-and-bound* sobre $\mathcal{N}_0$ al enviarla al *pool* de hebras.
3: Crear *pool* de hebras y ejecutar *task*.
4: Esperar a que acaben todas las hebras del *pool* de hebras.
5: **fin funcion**

6.2.1 Experimentación *branch-and-bound* paralelo

Con el objetivo de cuantificar la utilidad del método *branch-and-bound* en su versión paralela que acabamos de definir en el anterior apartado Sec.6.2, vamos a aplicar el algoritmo a dos redes credales (*cozman2s.elv27* y *cozmanConvex3st2ex0.3.elv*). Estas dos redes se introdujeron en el trabajo de Cozman *et al.* [55] y se utilizaron para evaluar el algoritmo *branch-and-bound* que quedó definido en el mismo trabajo. Ambas redes comparten el mismo grafo, que presentamos con anterioridad en la Fig. 6.1(a) y que recordamos en Fig. 6.6.

Algoritmo 20 *Branch-and-bound paralelo*

funcion branchAndBoundRecur($\mathcal{N}_0$)
    Calcular solución $r(\mathcal{N}_0)$ con el algoritmo de eliminación con arboles de probabilidad con cotas (mín y máx) para redes credales.
    si $r(\mathcal{N}_{0_h}) > \widehat{p}$ **entonces:**
        si entonces:N_0 tiene más de un vértice
            Usando las k posibilidades de uno de los conjuntos credales $\mathcal{N}_0$, obtener una lista de k redes credales $\mathcal{N}_{0_1}, ..., \mathcal{N}_{0_k}$ de $\mathcal{N}_0$.
            para cada $\mathcal{N}_{0_h}$ $(h > 1)$ **hacer:**
                Ejecutar *branch-and-bound* sobre N_{0h} en paralelo en otra hebra disponible del *pool* de hebras.
            fin bucle
            Ejecutar de manera recursiva branchAndBound ($\mathcal{N}_{0_1}$) en la hebra actual.
        sino
            ctualizar $\widehat{p}$ con $r(\mathcal{N}_0$.
        fin condicion
    fin condicion
fin funcion

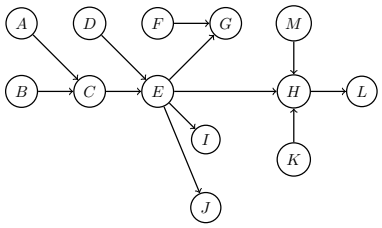


Fig. 6.6: Grafo de las redes credales utilizadas para la experimentación del algoritmo *branch-and-bound* paralelo.

Se ha realizado inferencia sobre la variable E sin considerar evidencia alguna. Y se ha hecho variar el número de hebras de 2 a 8, con el objetivo de estudiar si aumentar el número de hebras produce una reducción en el tiempo de cómputo. La diferencia entre las redes credales se da en el número de estados de cada variable, una tiene dos y la otra tres. Todas las pruebas se realizan utilizando dos vértices (puntos extremos) para cada conjunto credal condicional $H^{X_i|Y=y_j}$. Eso hace que las redes credales necesiten respectivamente 2048 (*cozman2s.elv27*) y 2097152 (*cozmanConvex3st2ex0.3.elv*) propagaciones con un algoritmo de inferencia en redes bayesianas para obtener el resultado exacto.

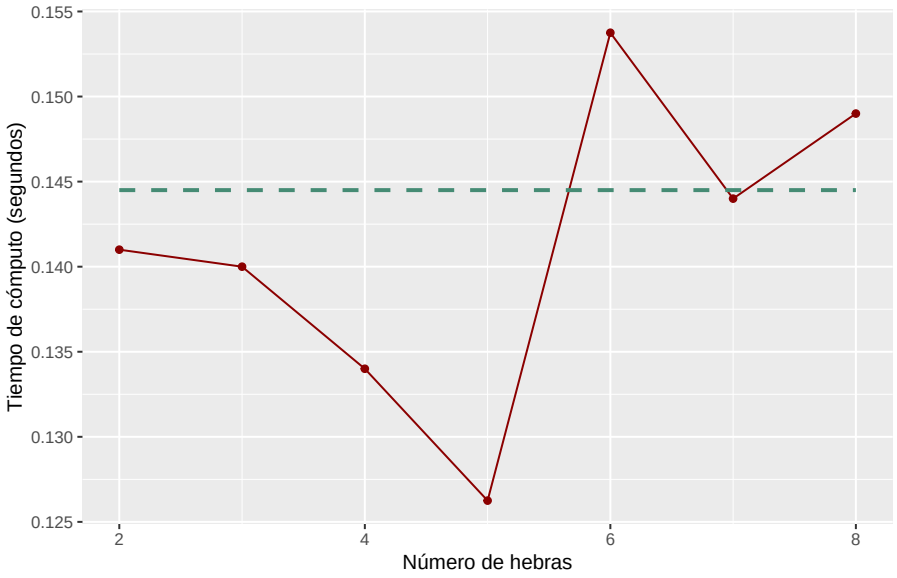


Fig. 6.7: Resultados de tiempos de cómputo al aplicar el algoritmo de *branch-and-bound* paralelo sobre la red credal *cozman2s.elv27*, variando el número de hebras entre 2 y 8.

Los resultados de los experimentos pueden observarse en las Figuras 6.7 y 6.8. En ellas, el eje x representa los distintos números de hebras y el tiempo de cómputo de cada número de hebras. En color granate se representa la distribución de los valores medios de los tiempos de cómputo (de cada estado y cada límite inferior o superior) del algoritmo *branch-and-bound* paralelo; y en verde el valor, también medio (de cada estado y cada límite inferior o superior) del algoritmo en su versión no paralela.

A la vista de estos experimentos, podemos comentar que para la red *cozman2s.elv27* el uso del algoritmo paralelo no parece presentar mejoría en cuanto a tiempo de cómputo. Por otro lado, el gráfico sí muestra que en el caso de la red *cozmanConvex3st2ex0.3.elv* el tiempo de cómputo se ha reducido en algo más de un 77% del tiempo con la utilización del algoritmo *branch-and-bound* paralelo. Además, en este caso, el aumento del número de hebras reduce el tiempo de cómputo, aunque se estabiliza con 6 hebras.

Este algoritmo propuesto en la memoria supone una primera aproximación y muy posiblemente pueda ser mejorado en trabajos futuros. Los experimentos realizados para probar el rendimiento de dicho algoritmo han sido limitados, pero hacen intuir que el método pueda posiblemente aportar mejoras en tiempo de cómputo en el caso de redes más complejas. Como trabajo futuro, será necesario aplicar el método a un mayor conjunto de redes credales.

ser bastante ineficiente

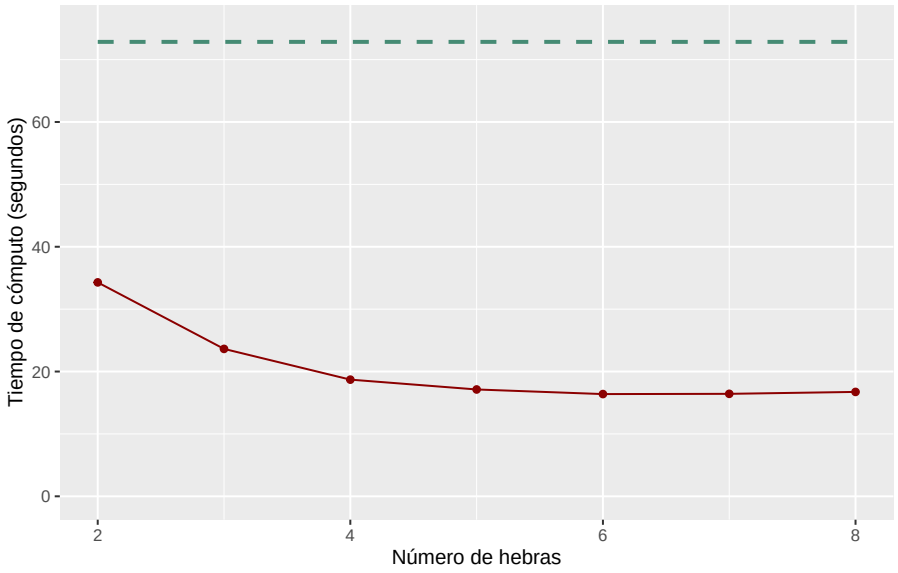


Fig. 6.8: Resultados de tiempos de cómputo al aplicar el algoritmo de *branch-and-bound* paralelo sobre la red credal *cozmanConvex3st2ex0.3.elv*, variando el número de hebras entre 2 y 8.

La computadora que ha realizado los experimentos tiene 8 *cores* (Intel(R) Core(TM) i7-7700 CPU @ 3.60GH con Linux Fedora 32). Posiblemente, en un ordenador con un mayor número de *cores*, el tiempo de computo disminuirá hasta utilizar un número de hebras cercano al número de *cores*.

Conclusiones y trabajo futuro

6.3 Conclusiones

- El trabajo de investigación continuo en MGP permite que los avances proporcionados pueden aplicarse al tratamiento de modelos más complejos. Estos avances pueden estar enmarcados en distintas tareas. En esta memoria se han ofrecido tres opciones: estructuras que permiten representar el conocimiento cuantitativo de un modo compacto, sacando provecho de la repetición de valores, independientemente del orden en que aparezcan; (2) algoritmo aproximado de aprendizaje estructural desde una perspectiva variacional, que puede ser utilizado sobre redes más complejas;(3) algoritmo exacto *branch-and-bound* paralelo de inferencia con redes credales, que permite utilizar los recursos computacionales más eficientemente valiéndose de un *pool* de hebras.

- La aportación sobre tratamiento de información cuantitativa indica que en los problemas reales aparecen patrones y repeticiones en los valores numéricos que cuantifican probabilidades (y pensamos que también ocurrirá lo mismo con las utilidades en los problemas de toma de decisiones). El aprovechamiento de estos patrones permite reducir el espacio de memoria necesario para almacenar y operar con los potenciales de probabilidad y, por tanto, brinda la posibilidad de abordar problemas más complejos.

- Además, en las estructuras propuestas, se permite aplicar una operación de aproximación muy efectiva y de escaso coste computacional. Esta operación puede ser usada en modelos en que la evaluación exacta no es factible.

- La aproximación sobre el aprendizaje variacional ha permitido investigar las posibilidades de esta técnica de optimización al problema de aprendizaje de redes. En este método, la relación entre las variables se trata como una distribución y en él, se generan conjuntos de redes que potencialmente representan efectivamente a los datos, en lugar de un único modelo; permite entonces combinar las soluciones y utilizar así el modelo Bayesiano completo.

- La inferencia con probabilidades imprecisas también ofrece muchas posibilidades de mejora. En esta tesis se ha abordado la paralelización de un algoritmo *branch-and-bound* ya existente. Las posibilidades de los procesadores actuales permiten que muchas tareas puedan ser ejecutadas de forma concurrente de un modo sencillo, por lo que utilizar este hecho para poder tratar modelos en tiempo razonable, es muy beneficioso.

Trabajo futuro

- En este trabajo se introdujo una primera aproximación a cómo se realizarían operaciones de combinación y marginalización sobre las estructuras de representación de la información que hemos presentado, pero disponer de operaciones específicas que saquen provecho de las ventajas que los potenciales basados en valor ofrecen, sería un futuro trabajo a realizar muy importante. Pensamos que esto implicaría una mejora el tiempo de realización de estas operaciones de forma sustancial, y por extensión, en las tareas de inferencia.
- La implementación del algoritmo de aprendizaje puede ser mejorada, de forma que sea más rápido y permita tratar conjuntos de datos con más variables. La versión actual está implementada en R y también permite paralelizar varias tareas. Está pendiente la realización de una experimentación más extensa con las alternativas de paralelización para aprovechar al máximo las capacidades del procesador usado.
- Además, el método presenta también limitaciones: (1) por un lado, no podemos asegurar que la distribución *a posteriori* que se define en las Ecuaciones 5.24 y 5.25 sea capaz de caracterizar que el grafo no puede contener ciclos. En caso de que $\mathcal{G}$ fuese cíclico, entonces $q(\mathcal{G}|\lambda^*)$ debería ser nulo. (2) Por otro lado, la aproximación variacional no es capaz de detectar dependencias entre los arcos.
- Por otro lado, pensamos mejorar el paquete R implementado para esta tarea, de forma que pueda presentarse a la comunidad científica y optar a su inclusión en el repositorio de paquetes oficiales de R.
- En cuanto al algoritmo de propagación con probabilidades imprecisas, también existe la posibilidad de pensar en otros algoritmos de búsqueda que permitan paralelización. También pensamos investigar la posibilidad de paralizar algunas tareas básicas, lo que agregaría un nivel de paralización de más bajo nivel.

Bibliografía

[1]Ahmed M Abdel-Zaher and Ayman M Eldeib. “Breast cancer classification using deep belief networks”. In: *Expert Systems with Applications* 46 (2016), pp. 139–144 (cit. on p. 1).

[2]Joaquín Abellán, Manuel Gómez-Olmedo, Serafín Moral, et al. “Some Variations on the PC Algorithm.” In: *Probabilistic Graphical Models*. Citeseer. 2006, pp. 1–8 (cit. on p. 37).

[3]Bruce Abramson and Anthony Finizza. “Using belief networks to forecast oil prices”. In: *International Journal of Forecasting* 7.3 (1991), pp. 299–315 (cit. on p. 1).

[4]Feri Afrinaldi. “Exploring product lifecycle using Markov chain”. In: *Procedia Manufacturing* 43 (2020), pp. 391–398 (cit. on p. 30).

[5]Pedro Aguilera Aguilera, Antonio Fernández, Rosa Fernández, Rafael Rumí, and Antonio Salmerón. “Bayesian networks in environmental modelling”. In: *Environmental Modelling & Software* 26.12 (2011), pp. 1376–1388 (cit. on p. 1).

[6]Stéphane Amarger, Didier Dubois, and Henri Prade. “Constraint propagation with imprecise conditional probabilities”. In: *Uncertainty Proceedings 1991*. Elsevier, 1991, pp. 26–34 (cit. on p. 180).

[7]Steen Andreassen, Roman Hovorka, Jonathan Benn, Kristian G Olesen, and Ewart R Carson. “A model-based approach to insulin adjustment”. In: *AIME 91: Proceedings of the Third Conference on Artificial Intelligence in Medicine, Maastricht, June 24–27, 1991*. Springer. 1991, pp. 239–248 (cit. on p. 115).

[8]Steen Andreassen, Finn V Jensen, Stig Kjær Andersen, et al. “MUNIN: an expert EMG Assistant”. In: *Computer-aided electromyography and expert systems*. Pergamon Press, 1989, pp. 255–277 (cit. on p. 115).

[9]Alessandro Antonucci, Yi Sun, Cassio P De Campos, and Marco Zaffalon. “Generalized loopy 2U: a new algorithm for approximate inference in credal networks”. In: *International Journal of Approximate Reasoning* 51.5 (2010), pp. 474–484 (cit. on p. 64).

[10]*Asia (synthetic) data set by Lauritzen and Spiegelhalter*. `https://www.bnlearn.com/documentation/man/asia.html`. "bnlearn - an R package for Bayesian network learning and inference" (cit. on p. 30).

[11]Louis JM Aslett, Frank Coolen, and Jasper De Bock. *Uncertainty in Engineering: Introduction to Methods and Applications*. Springer Nature, 2022.

[12]Marco Aste, Massimo Boninsegna, Antonino Freno, and Edmondo Trentin. “Techniques for dealing with incomplete data: a tutorial and survey”. In: *Pattern Analysis and Applications* 18 (2015), pp. 1–29 (cit. on p. 39).

[13]David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012 (cit. on p. 143).

[14]Ingo A Beinlich, Henri Jacques Suermondt, R Martin Chavez, and Gregory F Cooper. "The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks". In: *AIME 89: Second European Conference on Artificial Intelligence in Medicine, London, August 29th–31st 1989. Proceedings*. Springer. 1989, pp. 247–256 (cit. on p. 155).

[15]Albert Benveniste, Eric Fabre, and Stefan Haar. "Markov nets: probabilistic models for distributed and concurrent systems". In: *IEEE Transactions on Automatic Control* 48.11 (2003), pp. 1936–1950 (cit. on p. 30).

[16]James O Berger, Jose M Bernardo, and Dongchu Sun. "Objective priors for discrete parameter spaces". In: *Journal of the American Statistical Association* 107.498 (2012), pp. 636–648 (cit. on p. 141).

[17]James O Berger, Elías Moreno, Luis Raul Pericchi, et al. "An overview of robust Bayesian analysis". In: *Test* 3.1 (1994), pp. 5–124 (cit. on p. 54).

[18]Corrado Betterle, Chiara Dal Pra, Franco Mantero, and Renato Zanchetta. "Autoimmune adrenal insufficiency and autoimmune polyendocrine syndromes: autoantibodies, autoantigens, and their applicability in diagnosis and disease prediction". In: *Endocrine reviews* 23.3 (2002), pp. 327–364 (cit. on p. 53).

[19]Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006 (cit. on p. 144).

[20]Andrew Blake, Pushmeet Kohli, and Carsten Rother. *Markov random fields for vision and image processing*. MIT press, 2011 (cit. on p. 30).

[21]David M Blei, Alp Kucukelbir, and Jon D McAuliffe. "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877 (cit. on pp. 52, 142).

[22]Leon Bobrowski. "HEPAR: Computer system for diagnosis support and data analysis". In: *Prace Ibib* 31 (1992), pp. 27–48 (cit. on pp. 114, 155).

[23]Brent Boerlage. "Link strength in bayesian networks". PhD thesis. University of British Columbia, 1992 (cit. on p. 51).

[24]Pedro Bonilla-Nadal, Andrés Cano, Manuel Gómez-Olmedo, Serafín Moral, and Ofelia Paula Retamero. "Using Value-Based Potentials for Making Approximate Inference on Probabilistic Graphical Models". In: *Mathematics* 10.14 (2022), p. 2542 (cit. on p. 51).

[25]Remco R Bouckaert. "Probabilistic network construction using the minimum description length principle". In: *Symbolic and Quantitative Approaches to Reasoning and Uncertainty: European Conference ECSQARU'93 Granada, Spain, November 8–10, 1993 Proceedings 2*. Springer. 1993, pp. 41–48 (cit. on p. 37).

[26]Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. "Context-specific independence in Bayesian networks". In: *arXiv preprint arXiv:1302.3562* (2013) (cit. on pp. 2, 27, 72, 74, 76).

[27]John S Breese and Kenneth W Fertig. "Decision making with interval influence diagrams". In: *arXiv preprint arXiv:1304.1096* (2013) (cit. on p. 55).

[28]Wray Buntine. "Theory refinement on Bayesian networks". In: *Uncertainty proceedings 1991*. Elsevier, 1991, pp. 52–60 (cit. on p. 138).

[29]Rafael Cabañas, Manuel Gomez-Olmedo, and Andres Cano. "Using binary trees for the evaluation of influence diagrams". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 24.01 (2016), pp. 59–89 (cit. on pp. 1, 72, 75, 76).

[30]Rafael Cabañas, Manuel Gómez-Olmedo, and Andrés Cano. "Approximate inference in influence diagrams using binary trees". In: *Proceedings of the 6th European Workshop on Probabilistic Graphical Models (PGM)*. 2012, pp. 43–50 (cit. on pp. 1, 72, 75, 76).

[31]Rafael Cabañas de Paz. "New methods and data structures for evaluating influence diagrams". PhD thesis. 2017 (cit. on p. 30).

[32]Cassio P de Campos and Fabio G Cozman. "Inference in credal networks using multilinear programming". In: *Proceedings of the Second Starting AI Researcher Symposium*. 2004, pp. 50–61 (cit. on p. 64).

[33]A Cano, JE Cano, and S Moral. "Simulation algorithms for convex sets of probabilities". In: *Technical Report* (1993) (cit. on p. 54).

[34]Andrés Cano, José E Cano, and Serafín Moral. "Convex sets of probabilities propagation by simulated annealing". In: *Proceedings of the Fith International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU)*. 1994, pp. 4–8 (cit. on p. 54).

[35]Andrés Cano, Juan M Fernández-Luna, and Serafín Moral. "Computing probability intervals with simulated annealing and probability trees". In: *Journal of Applied Non-Classical Logics* 12.2 (2002), pp. 151–171 (cit. on p. 67).

[36]Andrés Cano, Manuel Gómez, Serafín Moral, and Joaquín Abellán. "Hill-climbing and branch-and-bound algorithms for exact and approximate inference in credal networks". In: *International journal of approximate reasoning* 44.3 (2007), pp. 261–280 (cit. on pp. 4, 179–181, 185, 187, 188, 190).

[37]Andrés Cano, Manuel Gómez-Olmedo, and Serafín Moral. "A score based ranking of the edges for the PC algorithm". In: *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models*. 2008, pp. 41–48 (cit. on p. 37).

[38]Andrés Cano, Manuel Gómez-Olmedo, and Serafín Moral. "Approximate inference in Bayesian networks using binary probability trees". In: *International Journal of Approximate Reasoning* 52.1 (2011), pp. 49–62 (cit. on pp. 1, 50, 51, 64, 72, 73, 75, 76).

[39]Andrés Cano and Serafín Moral. "A review of propagation algorithms for imprecise probabilities". In: (1999) (cit. on p. 54).

[40]Andrés Cano and Serafín Moral. "Algorithms for imprecise probabilities". In: *Handbook of Defeasible Reasoning and Uncertainty Management Systems: Algorithms for Uncertainty and Defeasible Reasoning* (2000), pp. 369–420 (cit. on p. 54).

[41]Andrés Cano and Serafin Moral. "Using probability trees to compute marginals with imprecise probabilities". In: *International Journal of Approximate Reasoning* 29.1 (2002), pp. 1–46 (cit. on pp. 67, 187).

[42]Andrés Cano, Serafín Moral, and Antonio Salmerón. "Penniless propagation in join trees". In: *International Journal of Intelligent Systems* 15.11 (2000), pp. 1027–1059 (cit. on pp. 1, 50, 51, 64, 65, 72, 73, 76).

[43]Andrés Cano Utrera. "Propagación aproximada de intervalos de probabilidad en grafos de dependencias". PhD thesis. 1999 (cit. on p. 54).

[44]Carlos M Carvalho and James G Scott. "Objective Bayesian model selection in Gaussian graphical models". In: *Biometrika* 96.3 (2009), pp. 497–512 (cit. on p. 141).

[45]Enrique Castillo, José Manuel Gutiérrez, and Ali S Hadi. "Sistemas expertos y modelos de redes probabilísticas". In: *Academia de Ingeniería* (1997), pp. 7–9.

[46]Alain Chateauneuf and Jean-Yves Jaffray. "Some characterizations of lower probabilities and other monotone capacities through the use of Möbius inversion". In: *Mathematical social sciences* 17.3 (1989), pp. 263–283 (cit. on p. 54).

[47]Arthur Choi, Hei Chan, and Adnan Darwiche. "On Bayesian network approximation by edge deletion". In: *arXiv preprint arXiv:1207.1370* (2012) (cit. on p. 51).

[48]Gregory F Cooper. "The computational complexity of probabilistic inference using Bayesian belief networks". In: *Artificial intelligence* 42.2-3 (1990), pp. 393–405 (cit. on p. 51).

[49]Gregory F Cooper and Edward Herskovits. "A Bayesian method for constructing Bayesian belief networks from databases". In: *Uncertainty Proceedings 1991*. Elsevier, 1991, pp. 86–94 (cit. on p. 37).

[50]Gregory F Cooper and Edward Herskovits. "A Bayesian method for the induction of probabilistic networks from data". In: *Machine learning* 9 (1992), pp. 309–347 (cit. on pp. 37, 138).

[51]I Couso, S Moral, and P Walley. "Examples of independence for imprecise probabilities. In de Cooman, G., editor". In: *Proceedings of the First Symposium on Imprecise Probabilities and Their Applications (ISIPTA), Ghent, Belgium*. 1999 (cit. on p. 60).

[52]Robert Cowell. "Introduction to inference for Bayesian networks". In: *Learning in graphical models* (1998), pp. 9–26 (cit. on p. 50).

[53]Robert G Cowell, Philip Dawid, Steffen L Lauritzen, and David J Spiegelhalter. *Probabilistic networks and expert systems: Exact computational methods for Bayesian networks*. Springer Science & Business Media, 2007 (cit. on p. 50).

[54]Fabio G Cozman. "Credal networks". In: *Artificial intelligence* 120.2 (2000), pp. 199–233 (cit. on p. 61).

[55]FABIO GAGLIARDI COZMAN. “Inference in Credal Networks with Branch-and-Bound AlgorithmsÝ”. In: *ISIPTA’03* (), p. 480 (cit. on pp. 179, 185, 187, 190).

[56]Fabio Gagliardi Cozman. “Separation properties of sets of probability measures”. In: *arXiv preprint arXiv:1301.3845* (2013) (cit. on p. 63).

[57]Paul Dagum and Michael Luby. “An optimal approximation algorithm for Bayesian inference”. In: *Artificial Intelligence* 93.1-2 (1997), pp. 1–27 (cit. on pp. 50, 52).

[58]Paul Dagum and Michael Luby. “Approximating probabilistic inference in Bayesian belief networks is NP-hard”. In: *Artificial intelligence* 60.1 (1993), pp. 141–153 (cit. on p. 51).

[59]Adnan Darwiche. “A differential approach to inference in Bayesian networks”. In: *Journal of the ACM (JACM)* 50.3 (2003), pp. 280–305 (cit. on p. 50).

[60]Luis M De Campos, Juan F Huete, and Serafin Moral. “Probability intervals: a tool for uncertain reasoning”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 2.02 (1994), pp. 167–196 (cit. on pp. 54, 56).

[61]Luis M De Campos and Serafin Moral. “Independence concepts for convex sets of probabilities”. In: *arXiv preprint arXiv:1302.4940* (2013) (cit. on pp. 54, 60).

[62]Rina Dechter et al. “Bucket elimination: A unifying framework for probabilistic inference.” In: *Learning in graphical models* 89 (1998), pp. 75–104 (cit. on pp. 69, 97, 117).

[63]Tristan Deleu, António Góis, Chris Emezue, et al. “Bayesian structure learning with generative flow networks”. In: *Uncertainty in Artificial Intelligence*. PMLR. 2022, pp. 518–528 (cit. on p. 141).

[64]A Dempster. “Upper and lower probabilities induced by multivalued mapping, A. of Mathematical Statistics, Ed”. In: *AMS-38* (1967) (cit. on p. 54).

[65]Arthur P Dempster et al. “Upper and lower probabilities induced by a multivalued mapping.” In: *Classic works of the Dempster-Shafer theory of belief functions* 219.2 (2008), pp. 57–72 (cit. on p. 54).

[66]Lorraine DeRoberts and JA Hartigan. “Bayesian inference using intervals of measures”. In: *The Annals of Statistics* (1981), pp. 235–244 (cit. on p. 54).

[67]Daniel Eaton and Kevin Murphy. “Bayesian structure learning using dynamic programming and MCMC”. In: *arXiv preprint arXiv:1206.5247* (2012) (cit. on pp. 2, 137).

[68]Herbert Edelsbrunner. *Algorithms in combinatorial geometry*. Vol. 10. Springer Science & Business Media, 1987 (cit. on p. 59).

[69]Byron Ellis and Wing Hung Wong. “Learning causal Bayesian network structures from experimental data”. In: *Journal of the American Statistical Association* 103.482 (2008), pp. 778–789 (cit. on p. 37).

[70]Alireza Farasat, Alexander Nikolaev, Sargur N Srihari, and Rachael Hageman Blair. “Probabilistic graphical models in modern social network analysis”. In: *Social Network Analysis and Mining* 5.1 (2015), pp. 1–18 (cit. on p. 1).

[71]Kenneth W Fertig and John S Breese. “Interval influence diagrams”. In: *Machine Intelligence and Pattern Recognition*. Vol. 10. Elsevier, 1990, pp. 149–161 (cit. on p. 180).

[72]Terrence L Fine. “Lower probability models for uncertainty and nondeterministic processes”. In: *Journal of statistical Planning and inference* 20.3 (1988), pp. 389–411 (cit. on p. 55).

[73]Nir Friedman. “Inferring cellular networks using probabilistic graphical models”. In: *Science* 303.5659 (2004), pp. 799–805 (cit. on p. 1).

[74]Nir Friedman and Daphne Koller. “Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks”. In: *Machine learning* 50 (2003), pp. 95–125 (cit. on pp. 2, 137, 142).

[75]José A Gámez, Juan L Mateo, and José M Puerta. “Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood”. In: *Data Mining and Knowledge Discovery* 22 (2011), pp. 106–148 (cit. on p. 37).

[76]José A Gámez and J Miguel Puerta. “Constrained score+ (local) search methods for learning Bayesian networks”. In: *Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 8th European Conference, ECSQARU 2005, Barcelona, Spain, July 6-8, 2005. Proceedings 8*. Springer. 2005, pp. 161–173 (cit. on p. 37).

[77]Francisco Javier García Castellano. “Modelos bayesianos para la clasificación supervisada: aplicaciones al análisis de datos de expresión genética”. PhD thesis. 2009.

[78]Dan Geiger, Thomas Verma, and Judea Pearl. “Identifying independence in Bayesian networks”. In: *Networks* 20.5 (1990), pp. 507–534 (cit. on p. 32).

[79]Carlos J. Gil Bellosta. *Introducción a la probabilidad y la estadística para científicos de datos*. `https://datanalytics.com/libro_estadistica/`. 2022.

[80]Donald Gillies. *Philosophical theories of probability*. Routledge, 2012.

[81]Francisco Javier Girón and Sixto Ríos. “Quasi-Bayesian behaviour: A more realistic approach to decision making?” In: *Trabajos de estadística y de investigación operativa* 31 (1980), pp. 17–38 (cit. on p. 54).

[82]Manuel Gómez and Andrés Cano. “Applying numerical trees to evaluate asymmetric decision problems”. In: *Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 7th European Conference, ECSQARU 2003 Aalborg, Denmark, July 2-5, 2003 Proceedings 7*. Springer. 2003, pp. 196–207 (cit. on pp. 1, 72, 76).

[83]Manuel Gómez-Olmedo, Rafael Cabañas, Andrés Cano, Serafín Moral, and Ofelia P Retamero. “Value-based potentials: Exploiting quantitative information regularity patterns in probabilistic graphical models”. In: *International Journal of Intelligent Systems* 36.11 (2021), pp. 6913–6943 (cit. on p. 51).

[84]Michel Grabisch, Hung T. Nguyen, and Elbert A. Walker. *Fundamentals of uncertainty calculi with applications to fuzzy inference*. Vol. 30. Springer Science & Business Media, 2013 (cit. on p. 54).

[85]Jonathan L Gross and Jay Yellen. *Handbook of graph theory*. CRC press, 2003.

[86]Marco Grzegorczyk and Dirk Husmeier. “Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move”. In: *Machine Learning* 71.2-3 (2008), p. 265 (cit. on pp. 2, 137, 142).

[87]Elias Gyftodimos and Peter A Flach. “Hierarchical bayesian networks: A probabilistic reasoning model for structured domains”. In: *Proceedings of the ICML-2002 Workshop on Development of Representations*. Citeseer. 2002, pp. 23–30 (cit. on pp. 1, 30).

[88]T. Hailperin. *Boole's Logic and Probability*. Studies in Logic and the Foundations of Mathematics, 1986 (cit. on p. 54).

[89]Chris Hans, Adrian Dobra, and Mike West. “Shotgun stochastic search for “large p” regression”. In: *Journal of the American Statistical Association* 102.478 (2007), pp. 507–516 (cit. on pp. 2, 137).

[90]Herman O Hartley. “Maximum likelihood estimation from incomplete data”. In: *Biometrics* 14.2 (1958), pp. 174–194 (cit. on p. 39).

[91]David Heckerman and John S Breese. “Causal independence for probability assessment and inference using Bayesian networks”. In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 26.6 (1996), pp. 826–831 (cit. on p. 156).

[92]David Heckerman, Dan Geiger, and David M Chickering. “Learning Bayesian networks: The combination of knowledge and statistical data”. In: *Machine learning* 20 (1995), pp. 197–243 (cit. on pp. 2, 36, 37, 138, 140).

[93]David Earl Heckerman, Eric J Horvitz, and Bharat N Nathwani. “Toward normative expert systems: Part i the pathfinder project”. In: *Methods of information in medicine* 31.02 (1992), pp. 90–105 (cit. on p. 114).

[94]J Aranda Hernández, AL Aguilar-Shea, and J Walsh. “Cáncer colorrectal de intervalo y criterios de calidad de colonoscopia: a propósito de un caso”. In: *SEMERGEN-Medicina de Familia* 39.1 (2013), pp. 52–55 (cit. on p. 53).

[95]LD Hernández, S Moral, and A Salmerón. “Importance sampling algorithms for belief networks based on approximate computation”. In: *Proceedings of the Sixth International Conference IPMU*. Vol. 96. 1996, pp. 859–864 (cit. on p. 52).

[96]Luis Daniel Hernández Molinero. “Algoritmos de Propagación I: Métodos Exactos”. In: (1998) (cit. on p. 50).

[97]Seyedmohsen Hosseini. “A decision support system based on machined learned Bayesian network for predicting successful direct sales marketing”. In: *Journal of Management Analytics* 8.2 (2021), pp. 295–315.

[98]Ronald A Howard and James E Matheson. “Influence diagram retrospective”. In: *Decision Analysis* 2.3 (2005), pp. 144–147 (cit. on p. 30).

[99]Peter J Huber. “Robust statistics”. In: *International encyclopedia of statistical science*. Springer, 2011, pp. 1248–1251 (cit. on p. 54).

[100]Claus S Jensen, Uffe Kjærulff, and Augustine Kong. "Blocking Gibbs sampling in very large probabilistic expert systems". In: *International Journal of Human-Computer Studies* 42.6 (1995), pp. 647–666 (cit. on pp. 51, 52).

[101]Finn V Jensen. "Bayesian networks". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 1.3 (2009), pp. 307–315 (cit. on p. 34).

[102]Finn V Jensen. "Bayesian updating in causal probabilistic networks by local computations". In: *Comput. Stat. Quarterly* 4 (1990), pp. 269–282 (cit. on p. 51).

[103]Finn V Jensen and Thomas Dyhre Nielsen. *Bayesian networks and decision graphs*. Vol. 2. Springer, 2007 (cit. on pp. 1, 30).

[104]Finn Verner Jensen, Kristian G Olesen, and Stig Kjaer Andersen. "An algebra of Bayesian belief universes for knowledge-based systems". In: *Networks* 20.5 (1990), pp. 637–659 (cit. on p. 51).

[105]Frank Jensen and SK Anderson. "Approximations in bayesian belief universe for knowledge based systems". In: *arXiv preprint arXiv:1304.1101* (2013) (cit. on pp. 51, 52).

[106]Zhiwei Ji, Qibiao Xia, and Guanmin Meng. "A review of parameter learning methods in Bayesian network". In: *Advanced Intelligent Computing Theories and Applications: 11th International Conference, ICIC 2015, Fuzhou, China, August 20-23, 2015. Proceedings, Part III 11*. Springer. 2015, pp. 3–12.

[107]Beatrix Jones, Carlos Carvalho, Adrian Dobra, et al. "Experiments in stochastic computation for high-dimensional graphical models". In: (2005) (cit. on pp. 2, 137).

[108]Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. "An introduction to variational methods for graphical models". In: *Machine learning* 37 (1999), pp. 183–233 (cit. on pp. 52, 142).

[109]Michael I Jordan and Yair Weiss. "Probabilistic inference in graphical models". In: *Handbook of neural networks and brain theory* (2002).

[110]Michael G Kapteyn, Jacob VR Pretorius, and Karen E Willcox. "A probabilistic graphical model foundation for enabling predictive digital twins at scale". In: *Nature Computational Science* 1.5 (2021), pp. 337–347 (cit. on p. 1).

[111]JinHyung Kim and Judea Pearl. "A computational model for causal and diagnostic reasoning in inference systems". In: *International Joint Conference on Artificial Intelligence*. 1983 (cit. on p. 40).

[112]Uffe B Kjaerulff and Anders L Madsen. "Bayesian networks and influence diagrams". In: *Springer Science+ Business Media* 200 (2008), p. 114 (cit. on p. 30).

[113]George J Klir and Tina A Folger. *Fuzzy sets, uncertainty, and information*. Prentice-Hall, Inc., 1987 (cit. on p. 54).

[114]Jürg Kohlas. *Information algebras: Generic structures for inference*. Springer Science & Business Media, 2012 (cit. on p. 180).

[115]Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009 (cit. on pp. 1, 2, 27, 29, 72).

[116]Kevin B Korb and Ann E Nicholson. *Bayesian artificial intelligence*. CRC press, 2010 (cit. on pp. 146, 148).

[117]Solomon Kullback and Richard A Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86 (cit. on pp. 69, 108, 143, 144, 150).

[118]Henry E Kyburg Jr. "Bayesian and non-Bayesian evidential updating". In: *Artificial intelligence* 31.3 (1987), pp. 271–293 (cit. on p. 55).

[119]Wai Lam and Fahiem Bacchus. "Using new data to refine a Bayesian network". In: *Uncertainty Proceedings 1994*. Elsevier, 1994, pp. 383–390 (cit. on p. 37).

[120]Pat Langley, Wayne Iba, and Kevin Thompson. "An analysis of Bayesian classifiers". In: *Aaai*. Vol. 90. Citeseer. 1992, pp. 223–228 (cit. on pp. 1, 30).

[121]Pierre-Simon de Laplace. *Théorie analytique des probabilités*. Mme Ve Courcier, imprimeur-libraire pour les mathématiques et la marine, 1812 (cit. on p. 15).

[122]Pedro Larrañaga and Serafín Moral. "Probabilistic graphical models in artificial intelligence". In: *Applied soft computing* 11.2 (2011), pp. 1511–1528 (cit. on p. 1).

[123]Steffen L Lauritzen. *Graphical models*. Vol. 17. Clarendon Press, 1996 (cit. on pp. 1, 29).

[124]Steffen L Lauritzen and David J Spiegelhalter. "Local computations with probabilities on graphical structures and their application to expert systems". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 50.2 (1988), pp. 157–194 (cit. on pp. 29, 30, 40, 51, 155, 180).

[125]Isaac Levi. *Hard choices: Decision making under unresolved conflict*. Cambridge University Press, 1990 (cit. on p. 54).

[126]Isaac Levi. "Imprecision and indeterminacy in probability judgment". In: *Philosophy of Science* 52.3 (1985), pp. 390–409 (cit. on p. 54).

[127]Zhaoyu Li and Bruce d'Ambrosio. "Efficient inference in Bayes networks as a combinatorial optimization problem". In: *International Journal of Approximate Reasoning* 11.1 (1994), pp. 55–81 (cit. on p. 50).

[128]Yuv Linnik. "INDEPENDENT VARIABLES". In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 4. Univ of California Press. 1961, p. 289 (cit. on p. 25).

[129]Jarno Lintusaari. "PCSI-labeled directed acyclic graphs". In: (2014) (cit. on p. 27).

[130]Chao-Lin Liu and Michael P Wellman. "Evaluation of Bayesian networks with flexible state-space abstraction methods". In: *International Journal of Approximate Reasoning* 30.1 (2002), pp. 1–39 (cit. on pp. 51, 52).

[131]David Madigan, Jeremy York, and Denis Allard. “Bayesian graphical models for discrete data”. In: *International Statistical Review/Revue Internationale de Statistique* (1995), pp. 215–232 (cit. on pp. 2, 137, 141, 142).

[132]Anders L Madsen and Finn V Jensen. *Parallelization of inference in Bayesian networks*. Citeseer, 1999 (cit. on p. 3).

[133]Ilias Maglogiannis, Elias Zafiropoulos, A Platis, and Costas Lambrinoudakis. “Risk analysis of a patient monitoring system using Bayesian Network modeling”. In: *Journal of Biomedical informatics* 39.6 (2006), pp. 637–647 (cit. on p. 1).

[134]Andres R Masegosa, Ana M Martinez, and Hanen Borchani. “Probabilistic graphical models on multi-core CPUs using Java 8”. In: *IEEE Computational Intelligence Magazine* 11.2 (2016), pp. 41–54 (cit. on p. 135).

[135]Andrés R Masegosa and Serafín Moral. “New skeleton-based approaches for Bayesian structure learning of Bayesian networks”. In: *Applied Soft Computing* 13.2 (2013), pp. 1110–1120 (cit. on pp. 2, 137).

[136]Andrés Ramón Masegosa Arredondo. “Models of Supervised Classification. Applications to Genomics and Information Retrieval”. PhD thesis. University of Granada, 2009.

[137]David Maxwell Chickering and David Heckerman. “Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables”. In: *Machine learning* 29 (1997), pp. 181–212 (cit. on p. 37).

[138]Thomas P Minka. “Expectation propagation for approximate Bayesian inference”. In: *arXiv preprint arXiv:1301.2294* (2013) (cit. on p. 143).

[139]Serafín Moral. “09 Modelos Gráficos para Probabilidades Imprecisas”. In: (1998) (cit. on p. 54).

[140]Serafín Moral and Andrés Cano. “Strong conditional independence for credal sets”. In: *Annals of Mathematics and Artificial Intelligence* 35 (2002), pp. 295–321 (cit. on p. 61).

[141]Serafín Moral and Antonio Salmerón. “Dynamic importance sampling in Bayesian networks based on probability trees”. In: *International Journal of Approximate Reasoning* 38.3 (2005), pp. 245–261 (cit. on p. 52).

[142]Kevin Murphy, Yair Weiss, and Michael I Jordan. “Loopy belief propagation for approximate inference: An empirical study”. In: *arXiv preprint arXiv:1301.6725* (2013) (cit. on p. 50).

[143]Richard E Neapolitan et al. *Learning bayesian networks*. Vol. 38. Pearson Prentice Hall Upper Saddle River, 2004 (cit. on pp. 2, 36).

[144]Teppo Niinimäki, Pekka Parviainen, and Mikko Koivisto. “Structure discovery in Bayesian networks by sampling partial orders”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2002–2048 (cit. on p. 37).

[145]James R Norris and James Robert Norris. *Markov chains*. 2. Cambridge university press, 1998 (cit. on p. 30).

[146]José Enrique Cano Ocaña. “Propagación de probabilidades inferiores y superiores en grafos”. PhD thesis. Universidad de Granada, 1993 (cit. on p. 59).

[147]Scott Mostyn Olmsted. *On representing and solving decision problems*. Stanford University, 1984 (cit. on p. 30).

[148]Agnieszka Onisko. “Probabilistic causal models in medicine: Application to diagnosis of liver disorders”. In: *Ph. D. dissertation, Inst. Biocybern. Biomed. Eng., Polish Academy Sci., Warsaw, Poland*. 2003 (cit. on pp. 114, 155).

[149]J udea Pearl and Stuart Russell. “Bayesian networks”. In: (2000) (cit. on pp. 1, 30).

[150]Judea Pearl. “Bayesian networks: A model of self-activated memory for evidential reasoning”. In: *Proceedings of the 7th conference of the Cognitive Science Society, University of California, Irvine, CA, USA*. 1985, pp. 15–17 (cit. on pp. 1, 30).

[151]Judea Pearl. “Evidential reasoning using stochastic simulation of causal models”. In: *Artificial intelligence* 32.2 (1987), pp. 245–257 (cit. on pp. 51, 52).

[152]Judea Pearl. “Fusion, propagation, and structuring in belief networks”. In: *Artificial intelligence* 29.3 (1986), pp. 241–288 (cit. on p. 50).

[153]Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988 (cit. on pp. 1, 29, 32, 40, 51).

[154]Ronald K Pearson. *Mining imperfect data: Dealing with contamination and incomplete records*. SIAM, 2005 (cit. on p. 39).

[155]Johan Pensar, Henrik Nyman, Jarno Lintusaari, and Jukka Corander. “The role of local partial independence in learning of Bayesian networks”. In: *International journal of approximate reasoning* 69 (2016), pp. 91–105 (cit. on p. 27).

[156]Aritz Pérez, Pedro Larrañaga, and Iñaki Inza. “Bayesian classifiers based on kernel density estimation: Flexible classifiers”. In: *International Journal of Approximate Reasoning* 50.2 (2009), pp. 341–362 (cit. on pp. 1, 30).

[157]Cora Beatriz Pérez Ariza. “New data structures and algorithms for uncertainty treatment with probabilistic graphical models”. PhD thesis. 2014.

[158]Eric Perrier, Seiya Imoto, and Satoru Miyano. “Finding Optimal Bayesian Network Given a Super-Structure.” In: *Journal of Machine Learning Research* 9.10 (2008) (cit. on p. 37).

[159]Jan von Plato. “AN Kolmogorov, Grundbegriffe der wahrscheinlichkeitsrechnung (1933)”. In: *Landmark Writings in Western Mathematics 1640-1940*. Elsevier, 2005, pp. 960–969 (cit. on p. 16).

[160]Katherine E Poruk, Matthew A Firpo, Douglas G Adler, and Sean J Mulvihill. “Screening for pancreatic cancer: why, how, and who?” In: *Annals of surgery* 257.1 (2013), p. 17 (cit. on p. 53).

[161]Franco P Preparata and Michael I Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012 (cit. on p. 59).

[162]Luis Rincón. *Curso elemental de probabilidad y estadística*. Universidad Nacional Autónoma de México, 2007.

[163]José Carlos Ferreira da Rocha, Fabio Gagliardi Cozman, and Cassio Polpo de Campos. "Inference in polytrees with sets of probabilities". In: *arXiv preprint arXiv:1212.2458* (2012) (cit. on pp. 62, 64).

[164]Patricia Román Román. *Tema 1: Variables aleatorias multidimensionales*. Universidad de Granada, 2013.

[165]Patricia Román Román. *Tema 3: Espacios de probabilidad: Definición axiomática y propiedadades básicas de la probabilidad*. Universidad de Granada, 2013.

[166]Patricia Román Román. *Tema 4: Probabilidad condicionada: teoremas básicos. Independencia de sucesos*. Universidad de Granada, 2013.

[167]Patricia Román Román. *Tema 5: Variables aleatorias: distribuciones de probabilidad y características.* Universidad de Granada, 2013.

[168]Havard Rue and Leonhard Held. *Gaussian Markov random fields: theory and applications*. Chapman and Hall/CRC, 2005 (cit. on p. 30).

[169]Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010 (cit. on p. 37).

[170]Antonio Salmerón, Andrés Cano, and Serafın Moral. "Importance sampling in Bayesian networks using probability trees". In: *Computational statistics & data analysis* 34.4 (2000), pp. 387–413 (cit. on pp. 1, 52, 64, 72, 73, 76, 104).

[171]Sumit Sarkar. "Using tree-decomposable structures to approximate belief networks". In: *Uncertainty in Artificial Intelligence*. Elsevier. 1993, pp. 376–382 (cit. on p. 51).

[172]*Scalameter: automate your performance testing today*. `http://https://scalameter.github.io/`. 2021 (cit. on p. 102).

[173]James G Scott and James O Berger. "An exploration of aspects of Bayesian multiple testing". In: *Journal of statistical planning and inference* 136.7 (2006), pp. 2144–2162 (cit. on p. 141).

[174]Marco Scutari. "Bayesian network constraint-based structure learning algorithms: Parallel and optimised implementations in the bnlearn R package". In: *arXiv preprint arXiv:1406.7648* (2014) (cit. on pp. 3, 4, 97, 114, 137, 155).

[175]Marco Scutari. "Learning Bayesian networks with the bnlearn R package". In: *arXiv preprint arXiv:0908.3817* (2009) (cit. on pp. 4, 97, 114, 137, 155).

[176]Teddy Seidenfeld and Larry Wasserman. "Dilation for sets of probabilities". In: *The Annals of Statistics* 21.3 (1993), pp. 1139–1154 (cit. on p. 54).

[177]Volkan Sevinç. "Determining the Flat Sales Prices by Flat Characteristics Using Bayesian Network Models". In: *Computational Economics* 59.2 (2022), pp. 549–577 (cit. on p. 1).

[178]Ross D Shachter. “Evaluating influence diagrams”. In: *Operations research* 34.6 (1986), pp. 871–882 (cit. on pp. 30, 40).

[179]Ross D Shachter and Mark A Peot. “Simulation approaches to general probabilistic inference on belief networks”. In: *Machine intelligence and pattern recognition*. Vol. 10. Elsevier, 1990, pp. 221–231 (cit. on pp. 50, 52).

[180]Glenn R Shafer and Prakash P Shenoy. *Local computation in hypertrees*. 1991 (cit. on p. 40).

[181]Prakash P Shenoy. “Binary join trees”. In: *arXiv preprint arXiv:1302.3604* (2013) (cit. on p. 180).

[182]Prakash P Shenoy and Glenn Shafer. “Axioms for probability and belief-function propagation”. In: *Machine intelligence and pattern recognition*. Vol. 9. Elsevier, 1990, pp. 169–198 (cit. on pp. 40, 50, 69, 97, 117, 180, 182).

[183]Oscar B Sheynin. “Early history of the theory of probability”. In: *Archive for History of Exact Sciences* 17.3 (1977), pp. 201–259 (cit. on p. 13).

[184]David J Spiegelhalter, A Philip Dawid, Steffen L Lauritzen, and Robert G Cowell. “Bayesian analysis in expert systems”. In: *Statistical science* (1993), pp. 219–247.

[185]Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2000 (cit. on pp. 2, 36).

[186]Todd Andrew Stephenson. *An introduction to Bayesian network theory and usage*. Tech. rep. Idiap, 2000.

[187]Luis Enrique Sucar. “Probabilistic graphical models”. In: *Advances in Computer Vision and Pattern Recognition. London: Springer London. doi* 10.978 (2015), p. 1 (cit. on pp. 1, 29, 50).

[188]Ben Taskar, Carlos Guestrin, and Daphne Koller. “Max-margin Markov networks”. In: *Advances in neural information processing systems* 16 (2003) (cit. on p. 30).

[189]Bjørnar Tessem. “Interval probability propagation”. In: *International Journal of Approximate Reasoning* 7.3-4 (1992), pp. 95–120 (cit. on pp. 180, 185, 187).

[190]Helmut Thöne, Ulrich Güntzer, and Werner Kießling. “Towards precision of probabilistic bounds propagation”. In: *Uncertainty in Artificial Intelligence*. Elsevier. 1992, pp. 315–322 (cit. on p. 180).

[191]Manuel Tovar Díaz and Miguel Ángel Montero Alonso. *Tema 1: Introducción al cálculo de probabilidades*. Universidad de Granada, 2002.

[192]Manuel Tovar Díaz and Miguel Ángel Montero Alonso. *Tema 2: Variable aleatoria y función de distribución*. Universidad de Granada, 2002.

[193]Ioannis Tsamardinos, Constantin F Aliferis, Alexander R Statnikov, and Er Statnikov. “Algorithms for large scale Markov blanket discovery.” In: *FLAIRS conference*. Vol. 2. St. Augustine, FL. 2003, pp. 376–380 (cit. on p. 37).

[194]Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. "The max-min hill-climbing Bayesian network structure learning algorithm". In: *Machine learning* 65 (2006), pp. 31–78 (cit. on p. 37).

[195]Amos Tversky and Daniel Kahneman. *Judgment under Uncertainty: Heuristics and Biases: Biases in judgments reveal some heuristics of thinking under uncertainty.* Vol. 185. 4157. American association for the advancement of science, 1974, pp. 1124–1131 (cit. on p. 13).

[196]UAI. *Inference Competition*. `http://www.hlt.utdallas.edu/~vgogate/uai14-competition/index.html`. 2014 (cit. on pp. 4, 97).

[197]UAI. *Inference Competition*. `http://www.hlt.utdallas.edu/~vgogate/uai16-evaluation/`. 2016 (cit. on pp. 4, 97).

[198]Robert A Van Engelen. "Approximating Bayesian belief networks by arc removal". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.8 (1997), pp. 916–920 (cit. on p. 51).

[199]J Verdegay-López. "Representación y Combinación de la Información con Incertidumbre mediante Convexos de Probabilidades". PhD thesis. 1997 (cit. on pp. 54–56).

[200]Kai Virtanen, Janne Karelahti, and Tuomas Raivio. "Modeling air combat by a moving horizon influence diagram game". In: *Journal of guidance, control, and dynamics* 29.5 (2006), pp. 1080–1091 (cit. on p. 30).

[201]Richard Von Mises. *Statistik und wahrheit*. Vol. 20. Springer, 1928 (cit. on p. 16).

[202]Martin J Wainwright, Michael I Jordan, et al. "Graphical models, exponential families, and variational inference". In: *Foundations and Trends® in Machine Learning* 1.1–2 (2008), pp. 1–305 (cit. on pp. 52, 142, 143).

[203]Peter Walley. "A Bounded Derivative Model for Prior Ignorance about a Real-valued Parameter". In: *Scandinavian Journal of Statistics* 24.4 (1997), pp. 463–483.

[204]Peter Walley. "Inferences from multinomial data: learning about a bag of marbles". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 3–34.

[205]Peter Walley. *Statistical reasoning with imprecise probabilities*. Vol. 42. Springer, 1991 (cit. on pp. 54, 55, 60).

[206]Zhenyuan Wang and George J. Klir. *Fuzzy Measure Theory*. Kluwer Academic Publishers, 1992 (cit. on p. 54).

[207]Gregory Wheeler. "A gentle approach to imprecise probability". In: *Reflections on the Foundations of Probability and Statistics: Essays in Honor of Teddy Seidenfeld*. Springer, 2022, pp. 37–67.

[208]Peter M Williams. "Indeterminate probabilities". In: *Formal Methods in the Methodology of Empirical Sciences: Proceedings of the Conference for Formal Methods in the Methodology of Empirical Sciences, Warsaw, June 17–21, 1974*. Springer. 1976, pp. 229–246 (cit. on p. 54).

[209]Jonathan S Yedidia, William T Freeman, and Yair Weiss. “Bethe free energy, Kikuchi approximations, and belief propagation algorithms”. In: *Advances in neural information processing systems* 13 (2001), p. 689 (cit. on p. 143).

[210]Nevin L Zhang and David Poole. “A simple approach to Bayesian network computations”. In: *Proc. of the Tenth Canadian Conference on Artificial Intelligence*. 1994 (cit. on p. 50).

[211]Nevin Lianwen Zhang and David Poole. “Exploiting causal independence in Bayesian network inference”. In: *Journal of Artificial Intelligence Research* 5 (1996), pp. 301–328 (cit. on pp. 50, 69, 97, 117).