UNIVERSIDAD DE GRANADA

E.T.S. DE INGENIERÍA INFORMÁTICA

Departamento de Ciencias de la Computación e Inteligencia Artificial

MÉTODOS LOCALES Y DISTRIBUIDOS PARA LA CONSTRUCCIÓN DE REDES DE CREENCIA ESTÁTICAS Y DINÁMICAS

TESIS DOCTORAL

José Miguel Puerta Callejón

Granada, Septiembre de 2001

MÉTODOS LOCALES Y DISTRIBUIDOS PARA LA CONSTRUCCIÓN DE REDES DE CREENCIA ESTÁTICAS Y DINÁMICAS

MEMORIA QUE PRESENTA

JOSÉ MIGUEL PUERTA CALLEJÓN

PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

SEPTIEMBRE 2001

DIRECTOR

LUIS MIGUEL DE CAMPOS IBÁÑEZ

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
E INTELIGENCIA ARTIFICIAL

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

La memoria titulada **Métodos Locales y Distribuidos para la Construcción de Redes de Creencia estáticas y dinámicas**, que presenta D. José Miguel Puerta Callejón para optar al grado de DOCTOR, ha sido realizada en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección del Doctor D. Luis Miguel de Campos Ibáñez.

Granada, Septiembre de 2001

El doctorando

El director

José Miguel Puerta Callejón

Luis Miguel de Campos Ibáñez

AGRADECIMIENTOS

En primer lugar he de mostrar mi más sincero agradecimiento al doctor Luis Miguel de Campos Ibáñez, director de esta memoria, por el apoyo, los consejos, el estímulo que he recibido en todo momento y sobre todo por su amistad, forjada a lo largo de estos años de colaboración. Tengo clarísimo que sin su esfuerzo, ayuda y dedicación este trabajo nunca hubiese visto la luz.

También quiero dar las gracias a S. Acid, A. Cano, J. M. Fernández, J. Huete, S. Moral y al resto de miembros del grupo de investigación UTAI del departamento de Ciencias de la Computación e I.A. de la Universidad de Granada su interés y su apoyo a lo largo de estos años. Especialmente quiero agradecerle a Juan Huete su enorme interés, su inestimable apoyo y su gran humanidad, gracias Juan por aguantarme todos estos años. Quiero extender este agradecimiento al resto de miembros del Departamento de Ciencias de la Computación e I.A. que de alguna manera se interesaron por la evolución de este trabajo.

Cómo no, agradecer a mis compañeros de departamento F. Cuartero, P. Cuenca, J. A. Gámez, A. Garrido, R. García, P. González, F. J. Gómez, J. A. Guerrero, M. López, T. Olivares, F. J. Quiles, J.L. Sánchez, V. Valero y F. J. Vigo su apoyo, comprensión e interés que mostraron a lo largo del desarrollo de esta memoria. No obstante, me gustaría dar las gracias especialmente a J. A. Guerrero por la desinteresada ayuda que siempre me ha ofrecido; a F. Cuartero, A. Garrido, F. J. Quiles y V. Valero por la enorme confianza que siempre han depositado en mí y por sus consejos, siempre sabios.

Sin embargo, no tengo por más que destacar a José Antonio Gámez, compañero y sobre todo amigo, amigo con mayúsculas; con compañeros como José Antonio de verdad que da gusto tanto trabajar como ir a tomar alguna cosa en algún lugar. Gracias Jose, por haber compartido todos estos años conmigo y esperemos que sigamos compartiendo por muchos años más. También he de hacer mención aparte a Teresa Olivares, mi cuñada, por estar siempre ahí, nunca poner mala cara y siempre estar dispuesta a echar una mano.

Quiero hacer extensiva mi gratitud al resto de mis compañeros del departamento de Informática de la Universidad de Castilla-La Mancha.

Por último, y muy especialmente, quiero dar las gracias a mi esposa, Elena, por su comprensión infinita, apoyo infatigable y enorme calidad humana. Espero compensarte algún día todo este tiempo. A mi hija Ana, gracias por tus ganas de vivir y por recordarnos lo que es más importante. A mis padres, por confiar en su “primogénito”, a mis abuelos, hermanos, cuñados y sobrinos; y al resto de mi familia por ser todos como sois, no cambiéis nunca.

A Elena y Ana.
A mi familia.

MÉTODOS LOCALES Y DISTRIBUIDOS PARA LA CONSTRUCCIÓN DE REDES DE CREENCIA ESTÁTICAS Y DINÁMICAS

JOSÉ MIGUEL PUERTA CALLEJÓN

Índice General

Índice de Figuras

Índice de Tablas

Introducción

La incertidumbre está presente en toda la información que manejamos en nuestra vida diaria, la cual puede provenir de que los datos que manejamos sean vagos, imprecisos, errores de medidas, etc. Así pues, el manejo de la incertidumbre se hace imprescindible en cualquier tarea que pretendamos abordar en cualquier campo de la informática, especialmente en aquellos sistemas donde se pretenda un comportamiento inteligente tal como el aprendizaje, el razonamiento, la planificación, la toma de decisiones y muchos otros en donde se incorporen datos del mundo real.

Las *Redes de Creencia* [85, 111] son una herramienta que en los últimos años ha demostrado su potencialidad como modelo de representación del conocimiento con incertidumbre en Inteligencia Artificial. El éxito de numerosas aplicaciones en campos tan variados como la medicina, recuperación de la información, visión artificial, fusión de información, agricultura, etc. avalan esta afirmación [86, 109, 1, 16, 119, 74, 83, 62, 14, 77, 80, 11, 58, 56]

Las redes de creencia representan mediante un grafo dirigido acíclico el conocimiento cualitativo del modelo, este conocimiento se articula en la definición de relaciones de independencia/dependencia entre las variables que componen el modelo. Estas relaciones abarcan desde una independencia completa hasta una dependencia funcional entre variables del modelo. El hecho de utilizar una representación gráfica para la especificación cualitativa del modelo hace de las redes de creencia una herramienta muy atractiva en su uso, ya que esta forma de representación es muy cercana a la forma de representar el conocimiento por parte de un humano.

El conocimiento cuantitativo viene determinado por un conjunto de parámetros que ponderan de alguna forma la fuerza de la relaciones expresadas en la parte cualitativa. Si estos parámetros son especificados mediante distribuciones de probabilidad (marginales y condicionales), entonces las redes de creencia se suelen denominar *Redes Bayesianas*. Sin embargo, estos parámetros se pueden especificar utilizando otras medidas de incertidumbre y no necesariamente probabilidades.

En los últimos años se ha producido un estudio profundo y por consiguiente un avance espectacular en el formalismo de las redes de creencia. Una de las dificultades más importantes asociadas a la construcción de sistemas expertos probabilísti-

cos, la mayoría de los cuales se basan en las redes de creencia, está en su elicitación [125]. Es por ello que se han estudiado formas de extraer el conocimiento a partir de datos en forma de redes de creencia. Estos métodos de aprendizaje automático de redes de creencia a partir de datos pueden ser una herramienta por sí sola y por tanto aportar una solución al problema "definitiva". O bien, estos métodos de aprendizaje se pueden ver como una herramienta para la toma de decisiones; como un modelo de partida en el que se base posteriormente un experto en el dominio en cuestión. Este ciclo de colaboración se puede iterar hasta alcanzar un modelo satisfactorio para el problema abordado.

El problema básico que resuelve el aprendizaje es el de poder encontrar un modelo de red a partir de datos ejemplo que mejor se adapte a éstos. De esta forma representamos el conocimiento aportado por los datos de una forma más útil y compacta, así como más comprensible para un humano. A partir de esta representación podremos realizar tareas de razonamiento válidas para situaciones o casos aún no observados.

Hemos comentado que las redes de creencia constan de dos componentes diferentes aunque estrechamente relacionados; los algoritmos de aprendizaje automático para las redes necesariamente tienen que realizar dos tareas bien diferenciadas, aunque no independientes entre sí:

- El aprendizaje de la estructura gráfica. Un grafo dirigido acíclico.
- La estimación de las distribuciones asociadas en la red. Distribuciones marginales y condicionales en redes Bayesianas.

La tarea del aprendizaje estructural depende de la estimación paramétrica que realicemos y el aprendizaje paramétrico depende de la estructura gráfica empleada. Así que, de una forma u otra, el aprendizaje estructural necesita de la estimación de los parámetros para confirmar o desmentir la presencia o no de una relación de dependencia/independencia entre variables del modelo.

En esta memoria fundamentalmente nos vamos a dedicar al problema del aprendizaje automático a partir de datos de la estructura de una red de creencia.

Por otra parte, para poder modelar procesos dinámicos, aunque éstos podrían ser modelados, bajo ciertas condiciones, con redes de creencia, ha aparecido una extensión de las mismas. Estas redes, denominadas *Redes de Creencia Dinámicas,*[1] son capaces de modelar sistemas dinámicos y, por tanto, tratar de forma explícita el tiempo. Un caso especial de redes de creencia dinámicas son aquellas que modelan sistemas dinámicos *markovianos*. Estas redes tienen la característica fundamental de que las predicciones sobre un tiempo futuro sólo dependen del tiempo presente,

[1] Cuando comparemos ambos modelos, denominaremos a las redes de creencia, *redes de creencia estáticas* en contraposición a las *redes de creencia dinámicas.*

esto es, el tiempo futuro es condicionalmente independiente del tiempo pasado conocido el tiempo presente. Si a esta última propiedad le añadimos la propiedad de que el sistema dinámico que queremos modelar es *estacionario*, entonces da lugar a que las distribuciones de probabilidad condicional que describen el modelo son independientes del tiempo, esto es, se repiten las mismas distribuciones en todos y cada uno de los instantes de tiempo considerados en el modelo.

A partir de estas propiedades se puede establecer un modelo gráfico para poder describir estos sistemas. Fundamentalmente consistirá en un conjunto de variables (nodos) indexados en el tiempo, y unas relaciones entre las variables, temporales y no temporales. Las primeras conectarán variables de distinto índice en el tiempo y con una orientación desde el índice menor hacia el índice mayor; y las segundas conectarán variables en el propio índice temporal. Si se cumplen las propiedades descritas en el párrafo anterior, entonces las relaciones temporales sólo pueden unir variables de índices de tiempo consecutivos, y todas las relaciones, tanto temporales como no temporales, son las mismas independientemente del índice de tiempo considerado. Así, una red de creencia dinámica markoviana y estacionaria se puede ver como:

- Una red inicial, para el primer periodo (índice) de tiempo, que especifica una distribución para el estado inicial del sistema dinámico.
- Un modelo de transición definido sobre una pareja de periodos de tiempo consecutivos, que especifica las distribuciones de transición del sistema dinámico.

En esta memoria nos dedicaremos también a estudiar estos modelos y a proponer métodos de aprendizaje estructural automático a partir de datos bajo ciertas premisas.

Objetivos

El objetivo básico de esta memoria es proponer algoritmos de aprendizaje estructural para redes de creencia estáticas y dinámicas. En el primer caso nos hemos centrado en el estudio de algoritmos locales de búsqueda para la optimización de una métrica o función de ajuste dada. Uno de los objetivos será desarrollar nuevos métodos de búsqueda, así como la adaptación de otras metaheurísticas, basadas en búsquedas locales, al problema del aprendizaje estructural. Por otra parte se desarrollarán métodos distribuidos de búsqueda para poder ser aplicados al mismo problema.

En el caso de las redes de creencia dinámicas vamos a proponer una nueva metodología en el proceso de aprendizaje. El objetivo básico es el de desarrollar nuevos

métodos de aprendizaje automático, tanto de las relaciones temporales como de las redes de creencia dinámicas completas.

Los subobjetivos para conseguir los objetivos anteriormente descritos son:

- Estudio de las redes de creencia como herramienta para la representación del conocimiento con incertidumbre, así como el estudio de las redes de creencia dinámicas como mecanismo de representación de sistemas dinámicos.

- Estudio de las técnicas de aprendizaje automático existentes para la construcción de redes de creencia estáticas y dinámicas a partir de datos. Estos métodos básicamente se pueden dividir en: técnicas basadas en tests de independencia condicional y técnicas basadas en optimización más una función de ajuste. Las primeras de ellas tratan de representar en una red las dependencias e independencias encontradas entre las variables del problema a partir de los datos. Los segundos tratan de encontrar una buena aproximación de la distribución subyacente en los datos. Para ello, habitualmente, se recorre el espacio de búsqueda de los grafos dirigidos acíclicos evaluando cada uno de ellos mediante la métrica utilizada, quedándose con el grafo que mejor puntuación obtenga como salida. En este caso particular estamos interesados en los métodos de búsqueda local por su eficiencia y por su relativa eficacia en los dominios previamente comprobados. También estamos interesados en estudiar métodos de búsqueda distribuidos para explorar más regiones del espacio de búsqueda de forma simultánea con la esperanza de aumentar la probabilidad de encontrar un buen óptimo.

- Desarrollo de nuevos métodos de búsqueda local para el aprendizaje de redes de creencia estáticas, así como la aplicación de una nueva metaheurística de reciente aparición, basada en búsquedas locales. Además de proponer nuevos métodos de búsqueda local en el espacio de los grafos dirigidos acíclicos, estudiaremos cómo realizar una búsqueda local en el espacio de órdenes entre las variables. En este caso deberemos definir una vecindad apropiada a este espacio y una forma de evaluar eficientemente una secuencia de ordenación a partir de una vecina. Es de destacar que varios trabajos previos de diferentes autores, ponen de manifiesto que el espacio de búsqueda en órdenes es mas homogéneo que el espacio de grafos dirigidos acíclicos. Adaptaremos la metaheurística, comentada previamente, a la búsqueda en los dos espacios comentados.

- Desarrollo de nuevos métodos de búsqueda local distribuida basada en la anterior metaheurística. También aplicaremos otra nueva metaheurística distribuida y colaborativa para la resolución del problema del aprendizaje estructural.

- Desarrollo de una metodología general de aprendizaje de redes de creencia dinámicas. Desarrollo, dentro de esta metodología, de nuevos algoritmos tanto basados en búsquedas locales y métricas como en tests de independencia condicional para el aprendizaje del modelo de transición de un sistema dinámico markoviano y estacionario.

- Adaptación general de los algoritmos de aprendizaje estructural de redes de creencia estáticas para el aprendizaje de redes de creencia dinámicas.

Contenido de la Tesis

El capítulo primero contiene el estudio de los conceptos preliminares necesarios para el resto de la memoria. Se articula en cuatro secciones; en la primera sección se realiza una introducción al resto del capítulo. La segunda sección realiza una descripción de los conceptos básicos sobre redes de creencia. La siguiente sección se centra en el problema del aprendizaje de redes de creencia estáticas, estudiando los métodos basados en tests de independencia condicional así como los basados en una métrica más una búsqueda local, particularizando en los algoritmos más conocidos en la literatura específica del tema y en los que nos basaremos en gran medida a lo largo de esta memoria. La última sección se dedica a estudiar el modelo de red de creencia dinámica, así como unos breves apuntes sobre el aprendizaje estructural en estos modelos.

El segundo capítulo se dedica a estudiar tanto nuevos métodos de búsqueda local, como la adaptación de una nueva heurística local para el aprendizaje de redes de creencia estáticas. Este capítulo se articula en seis secciones dedicando la primera a introducir el resto del capítulo. La segunda sección se dedica al estudio de una nueva metaheurística (VNS, Búsquedas en Entornos Variables) basada en múltiples reinicios y posteriores búsquedas locales en entornos de vecindad cada vez superiores si la búsqueda actual no ha tenido éxito (mejora el óptimo actual). La siguiente sección se dedica a estudiar tanto nuevos métodos de búsqueda local en el espacio de grafos dirigidos acíclicos, como la adaptación del método VNS al problema del aprendizaje en este mismo espacio de búsqueda. La sección cuarta se dedica a desarrollar nuevos métodos de aprendizaje basados en búsquedas locales pero esta vez en el espacio de secuencias de ordenación entre las variables; desarrollaremos un algoritmo de ascensión de colinas en este espacio de búsqueda y posteriormente adaptaremos de nuevo la heurística VNS a este nuevo espacio de búsqueda. En la sección siguiente evaluaremos experimentalmente todos los algoritmos desarrollados previamente. Para finalizar este capítulo, estudiaremos una nueva definición de vecindad en el espacio de grafos dirigidos acíclicos y lo validaremos de forma experimental.

El tercer capítulo se dedica al estudio de nuevos métodos de aprendizaje distribuido de redes de creencia estáticas. Nuestro enfoque, en este caso, será el de distribuir la búsqueda de tal forma que se mejora la eficacia de los algoritmos desarrollados. Este capítulo se divide en cinco secciones, siendo la primera una introducción al capítulo correspondiente. La segunda sección se dedica a estudiar cómo se puede distribuir la búsqueda VNS cuando disponemos de n procesadores y de memoria compartida en un computador. Esta suposición estará presente en todos los métodos desarrollados durante este capítulo. En la siguiente sección estudiamos una nueva metaheurística de reciente aparición como es el Sistema de Colonias de Hormigas. Esta metaheurística se puede ver como una método de búsqueda distribuido y cooperativo entre diferentes búsquedas, habitualmente locales. Durante esta sección adaptaremos y desarrollaremos todos los elementos necesarios para poder aplicar esta metaheurística al problema del aprendizaje de redes de creencia estáticas, tanto en el espacio de grafos dirigidos acíclicos como en el espacio de secuencias de ordenación entre las variables del problema. En la sección cuarta se presenta una nueva metodología de búsqueda híbrida entre las dos primeras búsquedas estudiadas, esto es, el método VNS y el Sistema de Colonias de Hormigas. Para finalizar el capítulo, evaluaremos de forma experimental los algoritmos desarrollados a lo largo de éste.

El capítulo cuarto se dedica al estudio del aprendizaje de la relaciones temporales de una red de creencia dinámica, conocida la estructura "estática" de la misma, esto es, conocidas las relaciones de dependencia/independencia no temporales. La primera sección se dedica a introducir el problema que se quiere resolver. En la sección segunda plantearemos de una manera más formal el problema así como las premisas para su resolución. La tercera sección se dedica al estudio de métodos de búsqueda local basados en una función de ajuste para el aprendizaje de las relaciones temporales de un modelo dinámico. En la cuarta sección se desarrollan algoritmos basados en tests de independencia condicional para solucionar el mismo problema; validaremos de forma teórica cada uno de los algoritmos planteados a lo largo de esta sección. En la siguiente sección se plantean algoritmos para refinar los resultados ofrecidos por los anteriores. En la sección sexta definiremos un modelo de red de creencia, que hemos denominado *simple*, que de forma habitual se presenta en casos prácticos; daremos una definición formal del modelo así como un algoritmo basado en tests de independencia condicional para su aprendizaje. Para finalizar el capítulo, realizaremos un estudio experimental de los algoritmos descritos durante el mismo.

En el capítulo quinto nos dedicamos a estudiar cómo se puede realizar de forma general el aprendizaje de redes de creencia dinámicas a partir de los algoritmos descritos en los capítulos anteriores. Tras la primera sección, de introducción, las secciones segunda y tercera introducen una metodología general, que será la de utilizar un algoritmo de redes de creencia estática, para aprender el modelo "estático"

de las redes de creencia dinámicas y utilizar posteriormente algoritmos de aprendizaje de relaciones temporales para completar el modelo de transición de las redes de creencia dinámicas. En la sección segunda este proceso se lleva a cabo bajo la suposición de que el instante inicial en que comienza el proceso temporal es conocido, mientras que en la sección tercera no se asume esta restrición. Para finalizar el capítulo, en la sección cuarta estudiaremos cómo poder adaptar los algoritmos de aprendizaje de redes de creencia estáticas, tanto basados en tests de independencia condicional como lo basados en búsquedas locales más una función de ajuste, al aprendizaje directo (sin distinguir entre relaciones temporales y no temporales) de las redes de creencia dinámicas.

Por último, presentamos las conclusiones sobre el trabajo realizado en esta memoria y algunas posibles líneas para continuar nuestra investigación.

Capítulo 1

Redes de Creencia, Algoritmos de Aprendizaje y Redes de Creencia Dinámicas

1.1 Introducción

Las *Redes de Creencia (RC)* son una representación apropiada para manejar conocimiento incierto y es por ello que son utilizadas habitualmente como núcleo de sistemas expertos capaces de manejar incertidumbre. Una red de creencia es una estructura gráfica (grafo dirigido acíclico) que de forma explícita representa un conjunto de variables y las relaciones de independencia y dependencia entre éstas. Cuando estas relaciones entre las variables se interpretan como causas y efectos, a estas redes se le suelen denominar *Redes Causales*.

Este tipo de redes, además de su representación cualitativa correspondiente a la topología de la red, nos permite representar un conocimiento cuantitativo entre las variables. Este conocimiento viene expresado habitualmente mediante distribuciones de probabilidad condicionales que miden la fuerza de las relaciones anteriormente planteadas. Cuando esto es así se suelen denominar *Redes Bayesianas*.

El primer problema que se plantea a la hora de utilizar las redes de creencia es el de la construcción del modelo para un determinado problema a resolver. Una posible solución será un conjunto de técnicas que nos permita elicitar el conocimiento aportado por un experto. Sin embargo, es bien conocido que la adquisición del conocimiento a partir de expertos produce un *cuello de botella* a la hora de desarrollar

estos modelos. Para paliar este problema, se han desarrollado un conjunto de herramientas que permiten el aprendizaje automático a partir de casos ejemplo de este tipo de estructuras.

En este capítulo se presenta una introducción sobre este formalismo y estudiaremos algunos algoritmos de aprendizaje automático de estos modelos. Finalmente presentaremos el problema que se nos plantea cuando tratamos de utilizar el anterior formalismo para modelar sistemas temporales, y por tanto justificaremos e introduciremos el formalismo de las *Redes de Creencia Dinámicas (RCD)* para tratar con sistemas dinámicos.

1.2 Redes de Creencia

Una red de creencia nos va a permitir representar nuestro conocimiento sobre un determinado problema a través de estructuras gráficas, grafos dirigidos acíclicos, donde los nodos representan las variables del problema y los arcos representan relaciones de dependencia, relevancia o causalidad. Observando la estructura gráfica de una red, notaremos que tenemos una representación explícita de un conjunto de dependencias e independencias entre las variables de nuestro problema. Esta observación nos lleva a obtener una representación semántica de la red, esto es, para un determinado problema, podemos extraer relaciones de relevancia o causalidad entre las variables. Una relación de dependencia entre dos variables x e y, implica una modificación en la creencia de x, dado que conocemos el valor que toma la variable y. Análogamente, una relación de independencia entre estas variables se interpreta como una no ganancia de información (no se modifica nuestra creencia) al conocer el valor de una de ellas.

Veremos ahora un ejemplo ilustrativo de las relaciones de independencia representadas en una red de creencia. Supongamos que tenemos la red de creencia, denominada Asia, de la figura 1.1. La red Asia fue presentada por [100] Lauritzen y Spiegelhalter a modo de ilustración de su método de propagación de evidencias. En esta red tenemos representadas ocho variables discretas. Por ejemplo la variable *Fumador* representa una variable bivaluada que nos indica si el paciente fuma o no, la variable *Visita a Asia* indica si el paciente a visitado Asia o no previamente.

Veamos las relaciones que se pueden deducir del ejemplo anterior: Fijémonos primero en el subgrafo $Fumador \rightarrow Bronquitis \rightarrow Disnea$. La evidencia de saber que un paciente es fumador nos hace aumentar la creencia de que va a padecer Bronquitis y ésta a su vez nos hace aumentar la creencia de que sufrirá Disnea. Sin embargo, si conocemos que el paciente padece Bronquitis, entonces el conocer si es o no fumador no me aportará una mayor creencia de que padecerá Disnea. Esto nos dirá que las variables Fumador y Disnea son Condicionalmente Independientes conocida la variable Bronquitis. Fijémonos ahora en el subgrafo

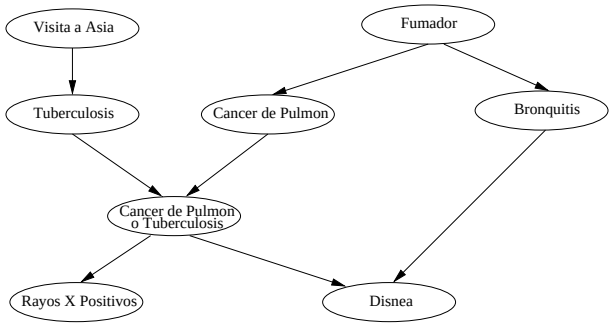


Figura 1.1: Red de creencia Asia.

Tuberculosis → *Cáncer de Pulmón o Tuberculosis* ← *Cáncer de Pulmón.* Sin ninguna evidencia las variables Tuberculosis y Cáncer de pulmón son independientes, esto es, el conocer que un paciente tenga o no Tuberculosis no me aportará ninguna creencia nueva de que padezca Cáncer de pulmón o no, sin tener conocido nada más. Ahora bien, si tenemos la evidencia de que el paciente tiene Cáncer de Pulmón o Tuberculosis, esto es, la variable Cáncer de Pulmón o Tuberculosis es positiva, entonces el conocer que el paciente tiene Cáncer de pulmón evidentemente me aportará una menor creencia sobre la variable Tuberculosis. En este tipo de situaciones, se dice que las variables Tuberculosis y Cáncer de pulmón son Marginalmente Independientes y son Condicionalmente Dependientes conocido Cáncer de Pulmón o Tuberculosis.

En la siguiente sección realizaremos un estudio del concepto de independencia, presentando el conjunto de axiomas que parece sensato exigirle al concepto intuitivo de independencia.

1.2.1 Axiomática de Independencia

Existen situaciones en donde somos capaces de establecer de forma clara que entre un conjunto de variables existe una relación de independencia, sin que por ello seamos capaces de establecer de forma cuantitativa estas relaciones. Es por ello que deberíamos pensar en el concepto de independencia más allá que como un concepto meramente cuantitativo, por consiguiente es necesario establecer un entorno abstracto para capturar el concepto de independencia.

Notaciones. Sea U el conjunto de variables del sistema (Universo del discurso), denotaremos con letras minúsculas a los elementos individuales de U, esto es, x, y ,..., mientras que los conjuntos de variables se denotarán mediante letras mayúsculas X, Y ,... s_x representa el conjunto de estados que puede tomar la variable discreta x,

así s_X, representará el conjunto de configuraciones, a partir de sus estados, que puede tomar el conjunto de variables discretas X. Una configuración de X, se denotará como **x**.

Definición 1.1 (Modelo de Dependencias) *Un modelo de Dependencias [111] se define como un par* $M = (U, I(\cdot))$*, donde* $I(\cdot)$ *es un conjunto de reglas que asignan valores de verdad al predicado "*X *es independiente de* Y*, dado que conocemos los valores que toman las variables de* Z*" y lo notaremos como* $I(X,Y|Z)$.

Por ejemplo, en entornos probabilísticos, una distribución de probabilidad P, puede ser considerada un modelo de dependencias utilizando la siguiente relación [35, 130, 112, 99]:

$$I(X,Y|Z) \Longleftrightarrow P(X|YZ) = P(X|Z) \text{ siempre y cuando } P(YZ) > 0 \qquad (1.1)$$

para toda instanciación de los subconjuntos X, Y y Z. En cualquier caso, un modelo de dependencias puede aplicarse con cualquier otro formalismo no probabilístico [38, 46, 39, 45, 136, 81]. Un estudio de las relaciones de independencia en la teoría de la probabilidad y en la teoría de Bases de Datos [55], proporciona un conjunto de propiedades que parece razonable exigir a toda relación que intente capturar el concepto intuitivo de independencia. Estas propiedades se pueden axiomatizar como sigue [111]:

A0 Independencia Trivial: $I(X,\emptyset|Z)$

En cualquier estado de conocimiento, una información nula no modifica la información que tenemos sobre X.

A1 Simetría: $I(X,Y|Z) \Rightarrow I(Y,X|Z)$

Dado un estado de conocimiento Z, si el conocer Y no aporta ninguna información sobre el valor de X, entonces el conocer X no aportará información sobre el valor que pueda tomar Y.

A2 Descomposición: $I(X,Y \cup W|Z) \Rightarrow I(X,Y|Z)$

Si dos componentes de información Y y W conjuntamente son consideradas irrelevantes sobre el valor que pueda tomar X, entonces cada una por separado también debe ser considerada irrelevante para X.

A3 Unión Débil: $I(X,Y \cup W|Z) \Rightarrow I(X,W|Y \cup Z)$

Este axioma, establece que al conocer la información Y considerada irrelevante para X, entonces esta información no puede ayudar a que otra información irrelevante W se transforme en relevante para X.

A4 Contracción: $I(X,Y|Z)\&I(X,W|Z\cup Y) \Rightarrow I(X,Y\cup W|Z)$

Si se considera que W es una información irrelevante para X después de conocer información irrelevante Y, entonces W también debería ser irrelevante para X antes de conocer Y.

A5 Intersección: $I(X,Y|Z\cup W)\&I(X,W|Z\cup Y) \Rightarrow I(X,Y\cup W|Z)$

Si dos elementos combinados de información, Y y W son relevantes para X, entonces al menos uno de ellos deber ser relevante para X, cuando el otro es conocido junto con Z.

Cualquier modelo de dependencias que satisface los axiomas anteriores menos el A5 se denomina semigrafoide, si además satisface el axioma A5 de intersección se llama grafoide [113].

Este conjunto de axiomas permite representar la esencia del concepto de independencia. Por tanto, proporcionan una herramienta adecuada para poder comparar las propiedades de una relación de independencia considerando diferentes formalismos. Además, el conjunto de axiomas puede considerarse como una regla general de inferencia, capaz de derivar nuevas relaciones de independencia a partir de un conjunto inicial de relaciones.

1.2.2 Redes de Creencia como Modelos de Dependencias

El objetivo de esta sección será considerar la red de creencia como una representación gráfica de un modelo de dependencias y realizar un análisis de las distintas propiedades que se presentan. En este caso, debe de existir una correspondencia directa entre el conjunto de variables en el modelo y el conjunto de nodos en un grafo, donde mediante la estructura gráfica de la red se representen un conjunto de propiedades de independencia del modelo.

Definición 1.2 (Grafo de Dependencias) *Un grafo $G = (V,E)$, donde V es el conjunto de nodos y E el conjunto de enlaces, es un grafo de dependencias para el modelo $M = (U,I(\cdot))$ si existe una correspondencia directa entre los elementos de U y V, tal que la topología del grafo refleje algunas propiedades de $I(\cdot)$.*

Si bien los grafos de dependencias pueden ser no dirigidos o dirigidos, estos últimos son los más usados debido a su mayor expresividad. Así, la existencia de

un arco $x \rightarrow y$ indica dependencia directa entre las variables x e y. Además, algunas veces muestra relación de causalidad entre ellas: x es causa de y. No obstante, esta afirmación no siempre es cierta, sino que depende del campo de conocimiento en que nos movamos.

Una interpretación semántica de un grafo de dependencias, necesita algún criterio gráfico que de forma precisa determine qué propiedades de independencia son reflejadas por la estructura del grafo. Este mismo criterio debe ser utilizado al hacer el análisis del grafo como una representación de un modelo de dependencias. Antes de considerar el criterio, debemos establecer una serie de definiciones.

Definición 1.3 (Conceptos básicos sobre grafos dirigidos) *Dado un grafo $G = (V,E)$ dirigido, denominaremos:*

- **camino**, *a una secuencia ordenada de nodos $(x_1,\ldots,x_r)$, tal que $\forall j = 1\ldots r-1$ la arista $x_j \rightarrow x_{j+1} \in E$ o $x_{j+1} \rightarrow x_j \in E$.*
- **camino dirigido**, *a una secuencia ordenada de nodos $(x_1,\ldots,x_r)$, tal que $\forall j = 1\ldots r-1$ la arista $x_j \rightarrow x_{j+1} \in E$.*
- **ciclo**, *a un camino $(x_1,\ldots,x_r)$, en el que $x_1 = x_r$.*
- **ciclo dirigido**, *a un camino dirigido $(x_1,\ldots,x_r)$, en el que $x_1 = x_r$.*
- **padres**, *de x_i, al conjunto de nodos:*

$$\pi(x_i) = \{x_j \in V \mid x_j \rightarrow x_i \in E\}$$

- **hijos**, *de de x_i, al conjunto de nodos:*

$$hij(x_i) = \{x_j \in V \mid x_i \rightarrow x_j \in E\}$$

- **ancestros**, *de x_i, al conjunto de nodos:*

$$Anc(x_i) = \{x_j \in V \mid \exists \text{ un camino dirigido } (x_j,\ldots,x_i)\}$$

- **descendientes**, *de x_i, al conjunto de nodos:*

$$Desc(x_i) = \{x_j \in V \mid \exists \text{ un camino dirigido } (x_i,\ldots,x_j)\}$$

- **nodos adyacentes**, *de x_i, al conjunto de nodos:*

$$Adyac(x_i) = \{x_j \in V \mid \exists\ x_i \rightarrow x_j \in E \text{ ó } x_j \rightarrow x_i \in E\}$$

- **grafo dirigido acíclico (GDA)**, *a un grafo dirigido que no contiene ningún ciclo dirigido.*

Definición 1.4 *El esqueleto de un grafo dirigido acíclico (GDA) G es el grafo no dirigido que se forma al eliminar de G todas las direcciones en los arcos. Un enlace cabeza-cabeza en un nodo es un camino que tiene la forma $x \rightarrow z \leftarrow y$, el nodo z es un nodo cabeza-cabeza en el camino, se dice que no es acoplado cuando x no es adyacente a y. Un camino C se dice activo por un conjunto de variables Z si se satisface que:*

- *Todo nodo del camino C con arcos cabeza-cabeza está en Z o tiene algún descendiente dentro de Z.*
- *Cualquier otro nodo en el camino no pertenece a Z.*

Si no se satisface esta relación se dice que el camino está bloqueado por Z.

Vistas estas definiciones el criterio gráfico de independencia en una red de creencia, llamado *d-separación* [111, 106, 134], puede expresarse como:

Definición 1.5 (d-separación). *Si X, Y y Z son tres subconjuntos de nodos disjuntos en un GDA G, entonces Z se dice que* **d-separa** *X de Y, o lo que es lo mismo X e Y son gráficamente independientes dado Z y lo notaremos como $\langle X, Y|Z\rangle$, si todos los caminos entre cualquier nodo de X y cualquier nodo de Y están bloqueados por Z.*

Utilizando el anterior criterio, cualquier GDA G sobre un conjunto de variables U, se puede considerar como un Modelo de Dependencias, $M(U, \langle\cdot\rangle)$ [111]. En este caso, además tenemos que el modelo de dependencias es un grafoide, esto es, satisface el conjunto de axiomas anterior.

Dado un modelo de dependencias M, no siempre es posible construir un GDA que satisfaga todas las relaciones de independencia en el modelo. Si nos plantemos la posible relación existente entre el modelos de dependencias M y su representación gráfica G, podremos encontrarnos los siguientes casos.

Definición 1.6 (I-Map) *[111]. Un GDA G se dice que es un* **I-Map** *de un modelo de dependencias M si toda relación de d-separación en G corresponde a una relación de independencia válida en el modelo M, es decir, si dados X, Y y Z conjuntos disjuntos de variables tenemos que:*

$$\langle X, Y|Z\rangle \Longrightarrow I(X, Y|Z) \tag{1.2}$$

Dado un GDA G, que es un I-Map de un modelo de dependencias M, decimos que es un **I-Map Minimal** de M si al borrar alguno de sus arcos, G deja de ser un I-Map del modelo.

Definición 1.7 (D-Map) *[111]. Un GDA G se dice que es un* **D-Map** *de un modelo de dependencias M si toda relación de independencia en el modelo M se corresponde con una relación de d-separación en G, es decir, si dados X, Y y Z conjuntos disjuntos de variables tenemos que:*

$$\langle X,Y|Z\rangle \Longleftarrow I(X,Y|Z) \tag{1.3}$$

Un I-Map garantiza que los nodos que están d-separados corresponden a variables independientes, pero no garantiza que para aquellos nodos que están d-conectados (no d-separados), sus variables sean dependientes. Recíprocamente, en un D-Map se puede asegurar que todos los nodos que están d-conectados son dependientes en el modelo, pero no toda relación de d-separación corresponde a variables independientes en el modelo.

Definición 1.8 (Perfect-Map) *[111]. Un GDA G se dice que es un* **Perfect-Map** *de un modelo M, si es un I-Map y un D-Map simultáneamente, esto es:*

$$\langle X,Y|Z\rangle \Longleftrightarrow I(X,Y|Z) \tag{1.4}$$

Si un grafo G es un Perfect-Map de un modelo de dependencias M, diremos que los modelos son **Isomorfos**, pudiéndose hablar indistintamente, entonces, de relaciones de independencia y de relaciones de d-separación.

Dado un modelo de dependencias, pueden existir distintas representaciones gráficas reflejando las mismas relaciones de independencia que el modelo. En este caso decimos que las representaciones son Isomorfas, y lo notaremos como $\equiv$. Por ejemplo, las siguientes relaciones reflejan el hecho de que x y z son marginalmente dependientes, pero conocida y se hacen condicionalmente independientes:

$$x \leftarrow y \leftarrow z \equiv x \rightarrow y \rightarrow z \equiv x \leftarrow y \rightarrow z$$

El siguiente teorema [111] nos da un conjunto de propiedades necesarias para que un GDA sea considerado isomorfo a un modelo de dependencias.

Teorema 1.1 *Una condición necesaria para un modelo de dependencias M sea isomorfo a un GDA G es que toda relación $I(X,Y|Z)$ satisfaga el siguiente conjunto de axiomas:*

- *Simetría:*

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

- *Composición / Descomposición:*

  $$I(X,Y \cup W|Z) \Leftrightarrow I(X,Y|Z) \& I(X,W|Z)$$

- *Unión Débil:*

$$I(X,Y \cup W|Z) \Rightarrow I(X,W|Y \cup Z)$$

- *Contracción:*

$$I(X,Y|Z)\&I(X,W|Z \cup Y) \Rightarrow I(X,Y \cup W|Z)$$

- *Intersección:*

$$I(X,Y|Z \cup W)\&I(X,W|Z \cup Y) \Rightarrow I(X,Y \cup W|Z)$$

- *Transitividad Débil:*

$$I(X,Y|Z)\&I(X,Y|Z \cup w) \Rightarrow I(X,w|Z) \text{ ó } I(w,Y|Z)$$

- *Cordalidad:*

$$I(x,w|y \cup z)\&I(y,z|x \cup w) \Rightarrow I(x,w|y) \text{ ó } I(x,w|z)$$

Hasta ahora nos hemos referido únicamente a la parte cualitativa del modelo que es la que nos permite representar las (in)dependencias entre las variables del sistema. A continuación vamos a añadir a éste la parte cuantitativa del modelo.

El formalismo que vamos a utilizar a lo largo de esta memoria para representar la parte cuantitativa del modelo es la *teoría de la probabilidad*. En entornos probabilísticos, una distribución de probabilidad P , puede ser considerada un modelo de dependencias utilizando la siguiente relación [35, 130, 112, 99]:

$$I(X,Y|Z) \Longleftrightarrow P(X|YZ) = P(X|Z) \text{ siempre y cuando } P(YZ) > 0 \qquad (1.5)$$

para toda instanciación de los subconjuntos X, Y y Z.

En un entorno probabilístico el conocimiento cuantitativo viene expresado por una distribución de probabilidad conjunta definida sobre las variables del sistema $U = \{x_1, x_2, \ldots, x_n\}$. Esto plantea un problema de representación, ya que si suponemos por ejemplo que todas las variables tienen dos estados posibles y que tenemos n variables, el número de entradas necesarias para representar la distribución de probabilidad conjunta es 2^n.

Vamos a ver ahora que si tenemos representada la parte cualitativa del conocimiento como un grafo de dependencias G, entonces la parte cuantitativa puede representarse de forma eficiente. Sea P una distribución de probabilidad conjunta

sobre U. La regla de la cadena nos permite representar la distribución de probabilidad conjunta $P(x_1, x_2, \ldots, x_n)$, como:

$$P(x_1, x_2, \ldots, x_n) = P(x_n|x_{n-1}, \ldots, x_1) \ldots P(x_3|x_2 x_1) P(x_2|x_1) P(x_1) \tag{1.6}$$

Supongamos ahora que tenemos definido un orden σ sobre las variables de U, tal que si $x_i \rightarrow x_j \in G$, entonces $x_i <_\sigma x_j$ (orden topológico, al ser un grafo dirigido acíclico siempre se puede encontrar este orden). Tomemos ahora un factor cualquiera $P(x_i|x_{i-1}, \ldots, x_1)$ de la expresión anterior. Si la secuencia de variables está ordenada según σ respecto de G, es claro que el conjunto $\{x_{i-1}, \ldots, x_1\}$ contiene al conjunto $\pi(x_i)$ y no contiene ningún descendiente de x_i, cumpliéndose entonces que:

$$\langle x_i, \{x_{i-1}, \ldots, x_1\} | \pi(x_i) \rangle_G$$

Si consideramos que G es un I-map, atendiendo al conocimiento cualitativo expresado en la red, entonces una variable x_i es condicionalmente independiente conocidos sus padres del resto de variables excepto sus descendientes, entonces la distribución de probabilidad P anterior se puede expresar como sigue:

$$P(x_1, x_2, \ldots, x_n) = P(x_n|\pi(x_n)) \ldots P(x_2|\pi(x_2)) P(x_1|\pi(x_1)) \tag{1.7}$$

con $\pi(x_i)$ representado el conjunto de padres del nodo x_i en la red. Por tanto la distribución de probabilidad conjunta se puede recuperar a través de la siguiente expresión:

$$P(x_1, x_2, \ldots, x_n) = \prod_i P(x_i|\pi(x_i)) \tag{1.8}$$

Por tanto, para recuperar la distribución de probabilidad sólo tendremos que almacenar para cada nodo una distribución de probabilidad condicional. Con esta representación se consigue, en general, un ahorro considerable en espacio requerido para almacenar la distribución de probabilidad conjunta.

Una vez visto cómo representar tanto la parte cuantitativa como cualitativa de un modelo, vamos a dar la definición formal de *Red de Creencia Bayasiana*.

Definición 1.9 (Red de Creencia Bayesiana) *Dada una distribución de probabilidad P sobre un conjunto de variables U, se define una red de creencia bayesiana (RC) como un grafo dirigido acíclico G tal que:*

1. *Cada nodo del grafo representa a una variable de U.*
2. *G es un I-map minimal de P.*

3. *Toda $x_i \in U$ lleva asociada la distribución de probabilidad condicional:*

$$P(x_i|\pi(x_i))$$

1.3 Algoritmos de Aprendizaje

Una vez que hemos definido las redes de creencia, se pueden destacar dos problemas fundamentales para trabajar con este formalismo. Por un lado se trata de utilizar éstas para realizar procesos eficientes de razonamiento una vez que tengamos especificado el modelo completo. Para este tipo de tareas han sido muchos los trabajos realizados en la literatura [122, 64, 10, 111, 51, 26, 100]. La existencia de estos algoritmos cada vez más eficientes, hacen que aparezcan cada vez más aplicaciones prácticas [11, 13, 17] que utilizan las redes de creencia como motor de inferencia en sus sistemas.

En los anteriores casos se considera que el modelo completo de red esta construido de antemano. El problema que se plantea ahora es cómo construir la red. Una posibilidad es elegir a un conjunto de expertos en el dominio en cuestión para que construyan el modelo, sin embargo es de enorme interés proporcionarles a estos expertos herramientas que adquieran este tipo de conocimiento de forma automática a partir de ejemplos de muestra, para que de esta forma tengan una herramienta de soporte para la decisión.

En esta sección revisaremos algunos de los algoritmos de Aprendizaje de Redes de Creencia más conocidos en la literatura especializada del tema, así como sus fundamentos teóricos en cada caso.

Podemos realizar una clasificación de los algoritmos de aprendizaje basándonos en las técnicas que utilizan en el proceso de adquisición del conocimiento.

- Métodos que utilizan relaciones de independencia condicional.
- Métodos que utilizan técnicas de optimización en base a una medida de ajuste o métrica.

Esta clasificación no es estricta ni exhaustiva, existen métodos que utilizan una combinación de ambos métodos, por ejemplo en [124, 5, 7, 34] o utilizan otro tipo de técnicas [12].

1.3.1 Algoritmos basados en relaciones de independencia condicional

En esta sección, vamos a estudiar algoritmos de aprendizaje basados en criterios de independencia condicional entre variables del modelo. Como comentábamos, estos

algoritmos en cierta medida son independientes del modelo de representación cuantitativa expresada en una red, por lo que se pueden considerar más "abstractos". En este sentido, su objetivo no es obtener una red donde la distribución de probabilidad que representa se "parezca" a la original, sino que hacen un estudio cualitativo de las propiedades del modelo y a partir de ellas intentan recuperar una red que represente "mejor" estas propiedades. Estos algoritmos toman como entrada un conjunto de relaciones de independencia condicional entre las variables o conjuntos de variables en el modelo. La salida será una red de creencia o red Bayesiana donde se satisfagan estas propiedades, es por ello que a este tipo de algoritmos también se lo suelen denominar como algoritmos de satisfacción de restricciones.

Para construir una red Bayesiana bastaría con estimar las distintas distribuciones de probabilidad condicional y efectuar algún tipo de test de hipótesis como test de independencia condicional y de esta forma conseguir los predicados de independencia condicional presentes en el modelo. En nuestro caso vamos a utilizar un test de hipótesis basado en la distribución χ^2; sabiendo que el estadístico $2 \cdot N \cdot Dep(X,Y|Z)$, siendo $Dep(\cdot)$ la entropía cruzada de Kullback-Leibler (expresión 1.18), sigue una distribución χ^2 con $[s_Z \cdot [(s_X - 1) \cdot (s_Y - 1)]]$ grados de libertad, con s_X representando el número de configuraciones posibles para el conjunto X, considerando X un conjunto de variables discretas. Y siendo N el número de casos sobre los datos que realizamos el test de independencia condicional.

Podemos hacer una abstracción del modelo original y considerarlo como un Modelo de Dependencias M. Hay que indicar que una distribución de probabilidad o una red de creencia (utilizando como test de independencia condicional el criterio de d-separación), pueden ser consideradas como modelos de dependencias.

Con objeto de recuperar la red, supondremos que los resultados de los tests de independencia condicional realizados se corresponden con las relaciones de independencia en el modelo. Además, se asume que se observan todas las variables relevantes en el problema y, que cuando partimos de una base de datos de casos, todos éstos siguen la misma relación. Con este tipo de algoritmos, se independiza el método para construir la red del formalismo que se utiliza para representar, de forma cuantitativa , el conocimiento sobre el problema. Para ello, los algoritmos se basan en el estudio de las propiedades estructurales del modelo. Como resultado de estas propiedades, tenemos que cuando el modelo es representable por un grafo dirigido acíclico, en general se encuentra la mejor representación del modelo. Entre las desventajas que tiene el uso de este tipo de algoritmos podemos destacar que: a) Cuando se parte de una base de datos, se necesitan una gran cantidad de observaciones para que los resultados de los tests de independencia sean fiables; b) No es posible asignar a priori probabilidades sobre los arcos, aunque si se podría permitir el uso del conocimiento dado por un experto sobre la existencia de una determinada relación o no, algún orden entre las variables [129, 44], etc.; c) Finalmente, propor-

cionan como salida un único modelo, sin cuantificar la verosimilitud con respecto a otras estructuras. Además en este último aspecto, en general, una equivocación en la estimación de una determinada relación de independencia suele ser más negativo en este tipo de algoritmos que en los basados en métricas.

A continuación estudiaremos un algoritmo de este tipo para ilustrar su comportamiento y su filosofía general, que de alguna forma comparten todos estos algoritmos. Este algoritmo es el algoritmo PC, el algoritmo más conocido sin lugar a dudas inspirado en este tipo de técnicas. Sin embargo existen una gran cantidad de algoritmos que de alguna u otra forma utilizan este mismo criterio. En los primeros de ellos se imponen restricciones en cuanto a su estructura [37, 40, 41, 67, 68] para mejorar la eficiencia de éstos. Posteriormente se diseñan algoritmos para recuperar GDAs tal como en los trabajos [44, 134, 126, 135].

1.3.1.1 Algoritmo PC

Para los algoritmos que estudiaremos a continuación se supone que el modelo M es representable por un grafo dirigido acíclico G, esto es, el modelo es isomorfo a G.

El primer algoritmo que consideraremos, dado por Spirtes, Glymour y Scheines [127], recupera de forma única un grafo que, salvo isomorfismos, representa el modelo. El algoritmo se basa en la siguiente propiedad:

Proposición 1.1 *Sea M un modelo isomorfo a un grafo dirigido acíclico G. Entonces M es isomorfo a G si y solo si:*

1. *Para cada par de variables x e y en G, x e y son adyacentes si y solo si x e y son condicionalmente dependientes dado todo subconjunto de nodos en G que no incluye a x ni a y.*

2. *Para cada terna de variables (x,y,z) tal que el par (x,y) es adyacente en G y el par (z,y) también es adyacente en G, pero no el par (x,z), entonces $x \rightarrow y \leftarrow z$ es un subgrafo de G si y solo si x y z son dependientes dado todo conjunto de variables que contiene a y pero no a z ni x*

Si utilizamos directamente la proposición 1.1, tendremos el algoritmo denominado SGS, el cual tiene un coste computacional exponencial, ya que por fuerza bruta debemos de buscar un subconjunto de $U \setminus \{x,y\}$ que haga independientes x de y. El algoritmo SGS puede observarse en la figura 1.2.

Por consiguiente en el peor de los casos el algoritmo SGS requiere un número de tests de d-separación que se incrementa exponencialmente con el número de vértices. Dicho algoritmo es muy ineficiente ya que para cada arco en el grafo verdadero G el peor caso es el caso esperado. Para cualquier arco no dirigido que está en el grafo G, el número de tests de d-separación que debe realizarse en el paso segundo

Algoritmo SGS

1. *Formar un grafo completo no dirigido H con el conjunto de vértices U.*
2. *Para cada par de variables x e y, si existe un subconjunto S de $U \setminus \{x,y\}$ tal que $I(x,y|S)$, eliminar la arista $x-y$ de H.*
3. *Sea K el grafo no dirigido que se obtiene como resultado del paso anterior. Entonces para cada tripleta $x-y-z$ en H donde $z-x$ no esté en H, si no existe un subconjunto S de $U \setminus \{x,y\}$, tal que $I(x,z|S \cup \{y\})$, entonces orientar la tripleta como $x \rightarrow y \leftarrow z$.*
4. *Repetir*
   (a) *Si $x \rightarrow y-z$ está en H, con x y z dos nodos no adyacentes, orientar $y-z$ como $y \rightarrow z$.*
   (b) *Si existe un camino dirigido desde x hasta y, y existe la conexión $x-y$, entonces orientar el arco como $x \rightarrow y$.*

   Hasta que no puedan ser orientados más arcos.

Figura 1.2: Algoritmo SGS

del algoritmo no es afectado de ningún modo por las adyacencias en el grafo verdadero. Por otra parte, el algoritmo tiene un problema cuando trabajamos con datos, la determinación de relaciones de independencia condicional de alto orden [1] a partir de datos es generalmente menos eficiente y fiable que si realizamos tests de relaciones de independencia condicional de orden bajo.

Por tanto sería de desear un algoritmo que tomara en cuenta estas consideraciones y no realizara tests de orden alto, atendiendo a las relaciones de adyacencia del grafo G. El siguiente algoritmo hace precisamente esto, empieza con el grafo no dirigido completo y va comprobando en primer lugar las relaciones de orden cero, posteriormente las de orden uno y así sucesivamente. El conjunto de variables sobre los que se va a condicionar sólo necesita ser un subconjunto del conjunto de variables adyacentes a una u otra de las variables condicionadas en el grafo formado en la etapa donde realiza tal comprobación.

Sea $Adyac(G,x)$ el conjunto de vértices adyacentes a x en el grafo G. Este conjunto se va modificando, en el algoritmo que veremos, conforme el algoritmo va progresando partiendo del grafo completo. El algoritmo PC puede observarse en la figura 1.3.

La complejidad del algoritmo depende del número de adyacencias que tengan los nodos en el grafo. Sea k el mayor número de adyacencias para un nodo en el grafo G, y sea n el número de vértices en el grafo. Entonces el número de tests de independencia condicional necesarios para el algoritmo está acotado por:

$$2\binom{n}{2}\sum_{i=0}^{k}\binom{n-1}{i}$$

Esta cantidad esta acotada por:

$$\frac{n^2(n-1)^{k-1}}{(k-1)!}$$

Para hacer el análisis en el peor caso, se asume que todo par de variables está d-separado por un subconjunto con cardinalidad k. En el caso general, el número de tests de independencia condicional requeridos por grafos con una cardinalidad máxima k será mucho menor. De todas formas, los requerimientos computacionales crecen exponencialmente con k.

El algoritmo PC es eficiente y fiable, pero realiza tests innecesarios. Así, para determinar cuándo se elimina un arco entre x e y, el procedimiento debe comprobar todo subconjunto S de $Adyac(C,x)\setminus\{y\}$ o de $Adyac(C,y)\setminus\{x\}$, pero las relaciones de independencia o dependencia entre muchos de estos subconjuntos de variables

[1] Se denomina así a las relaciones de independencia condicional en donde el conjunto condicionante tiene un número elevado de variables. Si por el contrario este conjunto contiene un número reducido de variables se le denominan relaciones de independencia de orden bajo.

Algoritmo PC

1. *Formar el grafo C completo no dirigido sobre el conjunto de vértices U.*
2. $n = 0$

   Repetir

   - *Repetir*

     - Seleccionar un par ordenado de variables x e y que sean adyacentes en C, tal que $Adyac(C,x) \setminus \{y\}$ tiene un cardinal mayor o igual a n, y un subconjunto S de $Adyac(C,x) \setminus \{y\}$ de cardinal n. Si x e y están d-separadas dado S, borrar el arco $x - y$ de C y guardar S en $SEPSET(x,y)$ y $SEPSET(y,x)$.

     Hasta

     - Todos los pares ordenados de variables adyacentes de x e y, tal que $Adyac(C,x) \setminus \{y\}$ de cardinalidad n han sido ya comprobados anteriormente.

     $n = n + 1$

   Hasta

   - Para cada par ordenado de variables adyacentes a x e y. $Adyac(C,x) \setminus \{y\}$ tiene una cardinalidad menor que n.

3. *Sea C' el grafo resultado de la etapa anterior. Para cada tripleta de vértices (x,y,z) tal que: el par (x,y) y el par (y,z), son adyacentes en C', pero el par (x,z) no es adyacente en C', orientar $x - y - z$ como $x \rightarrow y \leftarrow z$ si y solo si y no esta en $SEPSET(x,z)$.*
4. *Repetir*

   - Si $a \rightarrow b$, $b - c$ y a y c no son adyacentes y no hay cabeza de flecha en b, entonces orienta $b - c$ como $b \rightarrow c$.

   - Si hay un camino dirigido desde a hasta b y un arco entre a y b, entonces orienta $a \rightarrow b$.

   Hasta que que ningún arco más se pueda orientar.

Figura 1.3: Algoritmo PC

pueden ser irrelevantes para establecer una relación entre x e y. Si, para un modelo isomorfo a un grafo dirigido acíclico, las variables x e y son condicionalmente independientes dado los padres de x o los padres de y, entonces lo son dado un subconjunto de padres de x o de padres de y que contiene sólo los vértices que se encuentran en un camino no dirigido entre x e y. Por tanto, es suficiente con realizar los tests de independencia condicionados a subconjuntos de variables adyacentes a x y subconjuntos de variables adyacentes a y que están en caminos no dirigidos entre x e y. Esta idea se encuentra en [127] con una versión del algoritmo denominada PC*. En cualquier caso, el número de caminos posibles entre dos nodos es lo suficientemente grande como para que, por requerimientos de memoria, este algoritmo sólo tenga una aplicación práctica con un conjunto pequeño de variables. Cuando el número de variables es grande se deberá utilizar el algoritmo PC.

En el primer punto del segundo paso del algoritmo, se selecciona un par de variables y un subconjunto S para determinar una relación de independencia en el modelo. La búsqueda que realiza será más rápida si se seleccionan en primer lugar aquellas variables con más probabilidad de ser condicionalmente independientes dado S. Este problema se puede abordar utilizando distintas heurísticas de búsqueda:

H1 Comprobar los pares de variables y subconjuntos S en orden lexicográfico.

H2 Comprobar primero aquellos pares de variables que sean menos dependientes. Los subconjuntos S se seleccionan en orden lexicográfico.

H3 Para una variable determinada x, comprobar primero aquellas variables y que son probabilísticamente menos dependientes con x, condicionando sobre aquellos subconjuntos que son probabilísticamente más dependientes con x.

En [127] podemos encontrar una descripción del comportamiento de estas tres heurísticas ante un conjunto de ejemplos.

1.3.2 Algoritmos basados en optimización de una medida

En esta sección realizaremos un repaso de algunos algoritmos de aprendizaje que utilizan algún criterio de bondad en el ajuste como base para recuperar la red. El problema se puede enfocar en cómo podemos construir, a partir de datos, un grafo dirigido acíclico que, con el menor número de arcos, sea una buena representación de la base de datos.

Los algoritmos que se enmarcan en esta clase incorporan, implícita o explícitamente, los siguientes tres elementos:

- Una medida de calidad f que nos permite seleccionar la mejor estructura entre un conjunto de ellas.

- Una heurística de búsqueda para seleccionar, de entre todo el conjunto de posibles estructuras un subconjunto "bueno" de ellas, ya que es bien conocido que el espacio de búsqueda de los grafos dirigidos acíclicos es super-exponencial [29, 121].
- Un método para obtener la información cuantitativa (distribuciones de probabilidad) de la estructura resultante.

1.3.2.1 Medidas de Calidad de una Red de Creencia

Una medida de calidad $f(G:D,\phi)$, es un criterio mediante el cual podemos ordenar de alguna forma el conjunto de redes de creencia posibles, donde G es un red de creencia, ϕ es una información a priori[2], y D es un conjunto de datos. Por ello, dada la información a priori y/o un conjunto de datos D, el objetivo perseguido es obtener una red de creencia de alta calidad. Una medida de calidad debe satisfacer ciertas propiedades deseables. Por ejemplo, sería deseable que una media de calidad f asignara la misma medida a dos redes G_1 y G_2 que fuesen isomorfas, esto es, representaran el mismo modelo de dependencias. Otras propiedades deseables para una medida f serían:

- A redes recomendadas por los expertos se les debe asignar calidades más altas que a las rechazadas por ellos.
- Las representaciones perfectas (Perfect-Map) deben recibir calidades mayores que las que no lo son.
- Los I-Map minimales deber recibir calidades mayores que los no minimales.
- Las redes de reducido número de parámetros a igualdad de condiciones deben recibir mayor medida de calidad.
- Las redes que confirmen la información contenida en los datos D deben obtener una mayor medida de calidad que las que los contradigan.

Para ampliar conocimientos sobre este tema y otras propiedades se puede consultar [19, 20, 21, 75, 66, 23, 25, 24].

Las medidas de calidad, en nuestro caso, deben manejar la incertidumbre en dos parámetros, esto es, la estructura de la red de creencia y los parámetros asociados a ésta. En este caso se puede disponer de la información a priori ϕ y el conjunto de datos D, y el objetivo es encontrar la mejor red de creencia usando alguna medida de calidad $f(G:D,\phi)$. En este caso hemos de notar que en G tenemos representado tanto la estructura gráfica como el conjunto de parámetros de la red. Si estamos

[2] es habitual obviar este término en la función f ya que normalmente se especifica esta información a priori, de esta forma es normal encontrarnos la función como $f(G:D)$

interesados solamente en el aprendizaje estructural, entonces hemos de maximizar esta medida con respecto a sus parámetros.

Algunas medidas de calidad se definen como la suma de tres componentes [27]: $f = p$(información a priori)+ g(datos disponibles)+ h(complejidad), donde las anteriores funciones son conocidas. El significado de estas funciones son los siguientes:

- La función p asigna una probabilidad alta a las redes que han sido indicadas como altamente probables por la información a priori y una probabilidad baja a las que han sido indicadas como poco probables, este término se va haciendo despreciable conforme el conjunto de datos del que disponemos aumenta. Si no hay conocimiento previo este término se sustituye por cero lo que equivale a que se supone que la distribución de probabilidad asignada es la uniforme.

- La función g es un término de bondad en el ajuste que mide lo bien o mal que una red de creencia reproduce los datos disponibles D. Da una medida alta a las redes que estén de acuerdo con los datos y baja a las que los contradicen. Normalmente la contribución de este término aumenta conforme aumentan el número de arcos que introducimos en la red. En este caso se tiene más parámetros o grados de libertad y, normalmente, se puede obtener un mejor ajuste a los datos. Las elecciones típicas para este término son las siguientes:

  – La log-verosimilitud, esto es, el logaritmo de la función verosimilitud de los datos con respecto a la red: $\log P(D|G(\theta))$, donde θ representa el conjunto de parámetros de la red.

  – El logaritmo de la probabilidad a posteriori de los parámetros θ dada la estructura G y los datos D: $\log P(\theta|G,D)$.

- La función complejidad h penaliza aquellas redes con una estructura compleja (en parámetros o en arcos). Por esto, este término favorece a redes simples con un número reducido de arcos y parámetros.

  Para medir la complejidad de una red es necesario conocer su dimensión y normalmente se define como el número de parámetros necesarios para especificar su distribución conjunta.

En la literatura especializada existente se han propuesto varias medidas de calidad para redes de creencia. Estas las podemos clasificar en los tipos siguientes:

- Medidas Bayesianas
- Medidas de Mínima Longitud de Descripción (MDL)
- Medidas de Información

Estos tres tipos de medidas de calidad se discuten en las secciones siguientes.

1.3.2.2 Medidas Bayesianas

Las medidas de calidad Bayesianas se basan en la estadística Bayesiana (teorema de Bayes y las familias conjugadas en particular). En definitiva sería calcular la probabilidad a posteriori:

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)} \tag{1.9}$$

La idea básica de estas medidas Bayesianas consiste en asignar a toda red una medida que es una función de su probabilidad a posteriori (ver [66, 31]).

Medida de Cooper-Herskovits K2. Cooper y Herskovits [31] proponen la medida siguiente:

$$f(G:D) = \log P(G) + \sum_{i=1}^{n}\left[\sum_{k=1}^{s_i}\left[\log\frac{\Gamma(r_i)}{\Gamma(N_{ik}+r_i)} + \sum_{j=1}^{r_i}\log\Gamma(N_{ijk}+1)\right]\right] \tag{1.10}$$

donde N_{ijk} es la frecuencia de las configuraciones encontradas en los datos D de las variables x_i, donde n es el número de variables, tomando su j-ésimo valor y sus padres en G, $\pi_G(x_i)$, tomando su k-ésima configuración, donde s_i es el número de configuraciones posibles del conjunto de padres y r_i es el número de valores que puede tomar la variable x_i. $N_{ik} = \sum_{j=1}^{r_i} N_{ijk}$. Donde $\Gamma()$ es la distribución Gamma; en el caso discreto $\Gamma(n) = (n-1)!$.

Medida de Geiger y Heckerman BDe. Geiger y Heckerman [66] proponen la siguiente medida:

$$f(G:D) = \log P(G) + \sum_{i=1}^{n}\left[\sum_{k=1}^{s_i}\left[\log\frac{\Gamma(\eta_{ik})}{\Gamma(N_{ik}+\eta_{ik})} + \sum_{j=1}^{r_i}\log\frac{\Gamma(N_{ijk}+\eta_{ijk})}{\Gamma(\eta_{ijk})}\right]\right] \tag{1.11}$$

donde η_{ijk} son hiperparámetros y $\eta_{ik} = \sum_{j=1}^{r_i}\eta_{ijk}$. Existen diferentes formas de calcular estos hiperparámetros, bajo ciertas condiciones, a partir de un parámetro η que puede interpretarse como el "tamaño muestral equivalente". Un caso particular es cuando $\eta_{ijk} = 1$, en este caso esta medida coincide con la anterior (K2). Otro caso particular es cuando $\eta_{ijk} = \eta/(r_i \cdot s_i)$, en este caso se le suele denominar BDeu (uniforme).

1.3.2.3 Medidas de Mínima Longitud de Descripción

En las medidas Bayesianas es necesario especificar la información a priori. Esta información no tiene porque ser conocida y es por ello que en la mayoría de las

situaciones se suponga uniforme (desconocimiento absoluto, máxima entropía). Las medidas de mínima longitud de descripción [94, 131, 120] son una forma alternativa de establecer una métrica basada en conceptos de la teoría de la codificación, en donde la codificación se basa en reducir lo más posible el número de elementos necesarios para representar un mensaje atendiendo a su probabilidad de ocurrencia, esto es, los mensajes más frecuentes tienen códigos cortos y los mensajes menos frecuentes tendrán códigos largos. El principio de mínima longitud de descripción selecciona la codificación de conduce a una mínima longitud en la codificación de los mensajes.

En el caso de redes de creencia, la longitud de descripción incluye: La longitud requerida para almacenar los parámetros θ de la red. Es conocido que la longitud media necesaria para almacenar un número en el rango 0 a N es $1/2\log N$. Por esta razón, para almacenar los parámetros de la función de probabilidad conjunta, se necesita una longitud de $1/2C(G)\log N$, donde N es el tamaño de la muestra y $C(G)$ es la complejidad de la red G definida como:

$$C(G) = \sum_{i=1}^{n} (r_i - 1)s_i \tag{1.12}$$

es decir, el número de parámetros libres asociados a la función de probabilidad conjunta, siempre que almacenemos éstas como tablas de probabilidad; esta función puede verse modificada si utilizamos otras representaciones como árboles de probabilidad [59, 26].

Por otra parte hemos de considerar la longitud del conjunto de datos D codificados usando la distribución de probabilidad asociada a G, que en caso de redes de creencia discretas resulta ser:

$$\sum_{i=1}^{n} \sum_{j=1}^{r_i} \sum_{k=1}^{s_i} N_{ijk} \log \frac{N_{ijk}}{N_{ik}} \tag{1.13}$$

Entonces la medida quedaría de la siguiente forma:

$$f(G:D) = \sum_{i=1}^{n} \sum_{j=1}^{r_i} \sum_{k=1}^{s_i} N_{ijk} \log \frac{N_{ijk}}{N_{ik}} - 1/2C(G)\log N \tag{1.14}$$

1.3.2.4 Medidas de Información

Otra forma de medir la calidad de una red de creencia es mediante las medidas de información. Estas medidas están muy relacionadas con las anteriores. La idea básica consiste en seleccionar la estructura de red que mejor se ajusta a los datos, penalizada por el número de parámetros necesarios para especificar su correspondiente función de probabilidad conjunta. Esto conduce a una generalización de la medida de la ecuación 1.14:

$$f(G:D) = \sum_{i=1}^{n} \sum_{j=1}^{r_i} \sum_{k=1}^{s_i} N_{ijk} \log \frac{N_{ijk}}{N_{ik}} - C(G) f(N) \tag{1.15}$$

donde $f(N)$ es una función de penalización no negativa. Si esta función $f(N) = 1$, tendremos la medida del criterio de información de Akaike (AIC), si $f(N) = \log N/2$, entonces tendremos la medida del criterio de información de Schwarz (BIC) dando lugar a la misma medida de la ecuación 1.14 y si $f(N) = 0$, tendremos una medida (Bayesiana) del criterio de máxima verosimilitud.

Otras medidas de información bastante utilizadas están basadas en la medida de entropía cruzada de Kullback-Leibler [92], por ejemplo en el trabajo [30] utilizan esta medida para recuperar árboles. Posteriormente en el trabajo [118] generalizan el anterior algoritmo para recuperar poliárboles. Otros algoritmos que se pueden considerar como generalización del anterior son [3, 65]. Otro trabajo en donde se utiliza una medida de entropía es [79].

1.3.2.5 Algoritmos de Búsqueda

El problema del aprendizaje se podría plantear de la siguiente forma: Tenemos una base de datos de casos $D = \{\mathbf{v}^1, \ldots, \mathbf{v}^m\}$, donde $\mathbf{v}^i$ es un caso de la base de datos. Asumimos que tenemos definida una función de medida para un grafo dirigido acíclico dados los datos de la base de datos de casos $f(G:D)$. Sea $\mathscr{G}_n$ la familia de todos los grafos dirigidos acíclicos G con n nodos. Entonces el problema que queremos resolver será:

$$\text{Encontrar} \quad G^* = \arg\max_{G \in \mathscr{G}_n} f(G:D) \tag{1.16}$$

Una propiedad deseable para f y muy importante para el problema del aprendizaje es que la métrica sea descomponible en presencia de datos completos, esto es, la métrica f puede ser descompuesta del siguiente modo:

$$f(G:D) = \sum_{i=1}^{n} f(x_i, \pi_G(x_i) : N_{x_i, \pi_G(x_i)}) \tag{1.17}$$

donde $N_{x_i,\pi_G(x_i)}$ son los estadísticos (frecuencias) del valor de la variable x_i y el conjunto de sus padres en G, $\pi_G(x_i)$, en D, esto es, el número de instancias en D que concuerdan con cada posible configuración de x_i y $\pi_G(x_i)$ (en adelante omitiremos el término $N_{x_i,\pi_G(x_i)}$) . Esta propiedad es muy importante para el aprendizaje de redes de creencia: Un procedimiento local que cambia un solo arco para definir su vecindad, puede eficientemente evaluar el valor de la función f para cada uno de los vecinos de G. Estos procedimientos puede hacer uso intensivo de los cálculos realizados en etapas anteriores.

Es bien conocido (ver [29]) que el problema de aprendizaje de redes de creencia es no polinomial (NP-duro), es por ello que se ha planteado como un problema de optimización combinatoria en donde se han aplicado diferentes técnicas heurísticas de búsqueda, como pueden ser métodos locales [23, 31, 21, 60]...; Algoritmos de optimización global [97, 98, 42, 105]...

A continuación vamos a detallar dos de los algoritmos más conocidos en la literatura y sobre los que nos basaremos en gran medida a lo largo de esta memoria. Estos dos algoritmos utilizan como método de búsqueda una ascensión de colinas, heurística greedy. Estos dos algoritmos son el algoritmo K2 y el algoritmo B.

El algoritmo K2 fue propuesto por Cooper y Herskovits en [31] y fue una extensión de otro algoritmo propuesto por ellos mismos en [79]. Este algoritmo (figura 1.4) comienza por la red vacía de enlaces y supone que tenemos un orden causal establecido entre los nodos de la red. Para cada variable x_i, el algoritmo añade a su conjunto de padres $\pi(x_i)$, el nodo, menor que x_i en el orden, que conduce a un máximo incremento en la medida utilizada f. El proceso se repite hasta que, o bien no se incremente el valor de la medida, o se llegue a una cota u en el número de padres. En [9] se presenta una evaluación de este algoritmo de aprendizaje utilizando conjuntos de datos simulados.

Algoritmo K2

1. *Para* $i = 1$ *hasta n hacer:*
   (a) $\pi_i = \emptyset$; $OK = True$.
   (b) $P_{old} = f(x_i, \pi_i)$.
   (c) *Mientras OK y* $|\pi_i| < u$ *hacer:*
      i. *Sea z el nodo en el conjunto de predecesores de* x_i *que no pertenecen a* π_i*, que maximiza a* $f(x_i, \pi_i \cup \{z\})$.
      ii. $P_{new} = f(x_i, \pi_i \cup \{z\})$.
      iii. *Si* $P_{new} > P_{old}$ *Entonces* $\{P_{old} = P_{new}; \pi_i \cup \{z\}\}$. *En caso contrario* $OK = false$.
   (d) *Los padres del nodo* x_i *son* π_i.

Figura 1.4: Algoritmo K2

El algoritmo B (figura 1.5) fue propuesto por Buntine en [23]. El algoritmo comienza con conjuntos de padres vacíos como el algoritmo K2. En cada etapa

Algoritmo B

1. *Etapa de Inicialización:*
   (a) *Para* $i = 1$ *hasta n hacer:* $\pi(x_i) = \emptyset$
   (b) *Para* $i = 1$ *y* $j = 1$ *hasta n hacer:*
   Si $(i \neq j)$ *Entonces* $A[i,j] = f(x_i, x_j) - f(x_i, \emptyset)$
2. *Etapa Iterativa:*
   (a) *Repetir hasta que* $(A[i,j] \leq 0$ *ó* $A[i,j] = -\infty \ \forall i,j)$.
      i. *Seleccionar un par de índices* $(i,j) = \max_{i,j} A[i,j]$
      ii. *Si* $(A[i,j] > 0)$ *Entonces* $\pi(x_i) = \pi(x_i) \cup \{x_j\}$
      iii. *Para* $x_a \in Ancestros(x_i)$ *y* $x_b \in Descendientes(x_i) \cup \{x_i\}$ *hacer:*
      $A[a,b] = -\infty$.
      iv. $A[i,j] = -\infty$
      v. *Para* $k = 1$ *hasta n hacer:*
      Si $(A[i,k] > -\infty)$ *Entonces* $A[i,k] = f(x_i, \pi(x_i) \cup \{x_k\}) - f(x_i, \pi(x_i))$

Figura 1.5: Algoritmo B

se añade una nueva arista que no de lugar a ciclos dirigidos y que maximice el incremento de la medida f utilizada. El proceso se repite hasta que no se consigue incrementar más la medida utilizada o se obtiene una red completa.

1.3.3 Un algoritmo Híbrido basado en conjuntos mínimos d-separadores. BENEDICT.

BENEDICT [5], acrónimo compuesto por las palabras BElief NEtworks DIscovery using Cut-set Techniques, es una metodología híbrida para el aprendizaje de redes de creencia: utiliza una métrica específica y un método de búsqueda, pero también emplea explícitamente las relaciones de independencia condicional representadas en la red para definir la métrica, y utiliza tests de independencia para limitar el proceso de búsqueda.

La idea básica de los algoritmos de este tipo es cuantificar la discrepancia entre cualquier red candidata y la base de datos, midiendo para ello las discrepancias entre las independencias condicionales representadas en la red (a través del concepto de d-separación, separación direccional o independencia gráfica [111]) con las correspondientes independencias condicionales que puedan deducirse de la base de datos. La agregación de todas estas discrepancias será la métrica que utilicen los algoritmos. En cuanto al proceso de búsqueda, BENEDICT emplea una técnica greedy: inicialmente se parte de un grafo completamente inconexo, y en cada iteración se prueba a insertar cada uno de los arcos posibles, eligiendo aquél que produce una mayor disminución de la discrepancia, e incluyéndolo en el grafo de forma permanente. Se continua con este proceso hasta que finalmente se satisface una condición de parada.

La versión de BENEDICT que comentaremos aquí determina la estructura de la red bajo la suposición de que se dispone de una ordenación total de las variables (como sucede con otros algoritmos de aprendizaje [31, 79, 134]).

Puesto que la idea básica del algoritmo es medir las discrepancias entre las independencias condicionales representadas en cualquier red candidata y aquéllas que reflejan los datos, lo primero que hay que plantear es qué independencias representa una red. Esta cuestión tiene, en principio, una respuesta muy clara: todas las relaciones de independencia que pueden deducirse del grafo mediante el criterio de d-separación. Sin embargo, el número de asertos de d-separación representados en un grafo puede ser muy alto (crece exponencialmente con el tamaño del mismo), y por razones de eficiencia y fiabilidad interesa excluir gran parte de ellos y utilizar sólo un subconjunto "representativo" de todas las d-separaciones presentes. Una opción muy razonable se basa en utilizar el hecho de que en un grafo dirigido acíclico G, cualquier nodo x_j que no sea un descendiente de x_i está d-separado de x_i mediante el conjunto de padres de x_i en el grafo. Por tanto, se puede emplear como conjunto de independencias el formado por las sentencias de la forma $I(x_i, x_j | \pi_G(x_i))$, para

cada par de variables no adyacentes x_i y x_j; se supone que $x_j < x_i$ en el orden dado.

Sin embargo también importa el número de variables implicadas en esas independencias: cada uno de los asertos de independencia extraídos del grafo ha de ser contrastado con los datos mediante una medida de discrepancia, *Dep*. Así pues, interesa reducir lo más posible el tamaño de los conjuntos d-separadores: dados dos nodos x_i y x_j, tal que $x_j < x_i$, en lugar de utilizar el conjunto $\pi_G(x_i)$, BENEDICT usa un conjunto de tamaño mínimo que consiga d-separar x_i de x_j. Encontrar este conjunto supondrá un esfuerzo adicional, pero se verá compensado con un decrecimiento en la computación de la medida de discrepancia; también se obtendrán unos resultados más fiables, ya que se necesitan menos datos para estimar fiablemente una medida de orden menor.

Por tanto, dada una red candidata G, las relaciones de independencia cuya discrepancia con los datos se va a calcular son: $I(x_i, x_j | S_d(x_i, x_j)_G)$, para cualquier par de nodos no adyacentes x_i, x_j en G, tal que $x_j < x_i$, donde $S_d(x_i, x_j)_G$ es el mínimo conjunto d-separador de x_i y x_j. El método empleado para encontrar los conjuntos $S_d(x_i, x_j)_G$ está basado en una modificación del conocido algoritmo de Ford-Fulkerson para problemas de máximo flujo en redes [4]. En el algoritmo BENEDICT, el cálculo de los conjuntos d-separadores mínimos se lleva a cabo mediante la función *Mínimo-Corte*.

En cuanto a la forma de medir la discrepancia entre cualquier sentencia gráfica de independencia condicional representada en el grafo y los datos, se emplea la entropía cruzada de Kullback-Leibler [92], que mide el grado de dependencia entre x e y dado que conocemos el valor de z:

$$Dep(x, y|z) = \sum_{x,y,z} P(x, y, z) \log \frac{P(x, y|z)}{P(x|z)P(y|z)} \tag{1.18}$$

donde P representa la distribución de probabilidad estimada a partir de la base de datos. Esta medida toma el valor 0 cuando x e y son realmente independientes dado z, y es tanto mayor cuanto más dependientes entre sí son x e y dado z.

En lo que se refiere a la medida de discrepancia global entre el grafo G y la base de datos D, $f(G : D)$, que emplea el algoritmo para puntuar los méritos relativos de cada red candidata seleccionada por el proceso de búsqueda, se define de la siguiente manera:

$$f(G : D) = \sum_{x_j < x_i, x_j \notin \pi_G(x_i)} Dep(x_i, x_j | S_d(x_i, x_j)_G)$$

El algoritmo BENEDICT se describe en la figura 1.6.

En la descripción del algoritmo anterior no se ha especificado la forma concreta en que se detiene el proceso de aprendizaje. Se utilizan tests de independencia para ir eliminando arcos del conjunto de arcos candidatos, y detener el proceso de forma natural cuando dicho conjunto llegue a ser vacío [7] (se eliminan arcos candidatos

Algoritmo BENEDICT

1. *Comenzar con un grafo G sin arcos* ($G = \emptyset$)
2. *Se fija* $L = \{x_j \rightarrow x_i | x_j < x_i\}$; $f = 0$
3. *Para cada par de nodos* $x_s < x_t$ *hacer* $f = f + Dep(x_t, x_s|\emptyset)$
4. $min = f$
5. *Hasta detenerse hacer*
   (a) *Para cada enlace* $x_j \rightarrow x_k \in L$ *hacer*
      i. $G' = G \cup \{x_j \rightarrow x_k\}$; $f = 0$
      ii. *Para cada nodo* x_t *hacer*
         Para cada nodo $x_s < x_t$ *tal que* $x_s \notin \pi_{G'}(x_t)$ *hacer*
         $S_{G'}(x_t, x_s) = Minimo\text{-}Corte(x_t, x_s)$
         $f = f + Dep(x_t, x_s|S_d(x_t, x_s)_{G'})$
      iii. *Si* $f < min$ *entonces*
         $min = f$
         $x = x_k$; $y = x_j$
   (b) $G = G \cup \{y \rightarrow x\}$
   (c) $L = L \backslash \{y \rightarrow x\}$

Figura 1.6: El algoritmo BENEDICT.

bien porque se insertan en la estructura o bien porque sus nodos extremos se hallan independientes). También se realiza un proceso final de poda de arcos (similar a los métodos de poda empleados habitualmente para árboles de clasificación): una vez terminado el proceso de inserción de arcos, se procede a una revisión de cada uno de ellos: se prueba a eliminarlos uno a uno, empleando también para ello un test de independencia.

1.4 Redes de Creencia Dinámicas

1.4.1 Introducción

La mayoría de las investigaciones en razonamiento probabilístico se han centrado en la construcción y uso de modelos fundamentalmente estáticos, en los cuales las relaciones temporales entre las variables del modelo son fijas e invariantes en el tiempo. Las predicciones o cálculos de las probabilidades a posteriori dado un conjunto de observaciones no varían con el tiempo. En estos modelos estáticos se tiene solo en cuenta las observaciones actuales para predecir el estado del sistema sin posibilidad de tener en cuenta la historia de la evolución temporal de las observaciones en el sistema.

Aunque algunos problemas dinámicos se pueden resolver con modelos estáticos, parece más razonable considerar en el modelo resultante la evolución temporal del sistema, con los medios necesarios para poder actualizar las relaciones que dependan del tiempo. En comparación con los modelos estáticos, una consideración temporal en el modelo enriquecería el mismo con la información de la tendencia temporal del sistema así como con métodos para poder actualizar el modelo en respuesta a las observaciones de la historia del proceso evolutivo del sistema.

Ejemplo 1.1 *Vamos a suponer que tenemos la red de creencia descrita en la figura 1.7 para monitorizar el proceso de siembra y recolección del trigo.*

En dicha red tenemos la información necesaria para poder obtener mediante algoritmos de inferencia, [100, 84], de manera eficiente $P(CPS|T,ES,SNV)$ *o* $P(F|T,ES,SNV)$*, etc. Hemos de notar, sin embargo, que estas distribuciones de probabilidad a posteriori son válidas solo para un periodo de tiempo determinado, por ejemplo una semana, ya que claramente existen variables en el sistema que dependen del tiempo. Por otra parte, en un proceso dinámico han de tenerse en cuenta no solo las evidencias que tenemos en el actual instante de tiempo, sino también la llegada de nueva evidencia para un proceso de razonamiento.*

En primer lugar, como hemos comentado, parece razonable que un modelo gráfico dinámico pueda tener en cuenta las observaciones históricas, sin embargo existe otra razón por la que es recomendable ampliar el modelo de la red de creencia.

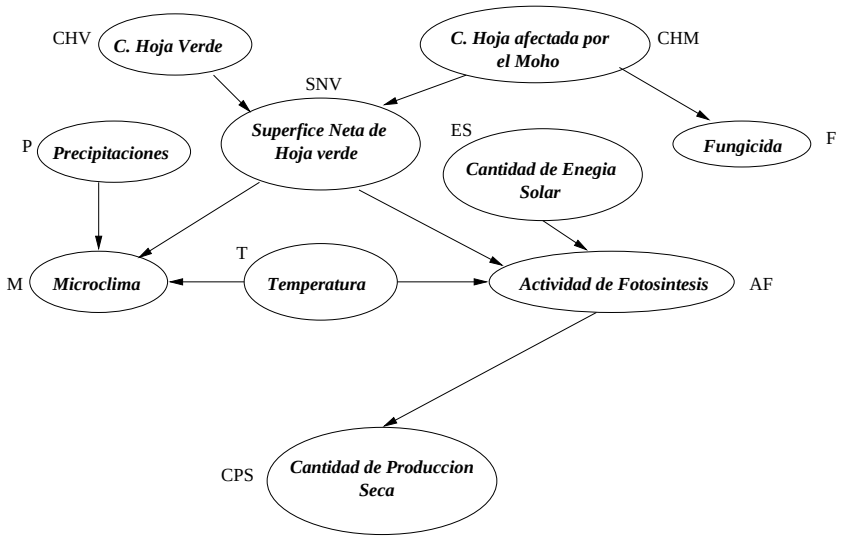


Figura 1.7: Red de creencia para un sistema de producción de trigo.

Si observamos de nuevo la estructura de la red de la figura 1.7, parece lógico establecer la siguiente relación: La cantidad de hoja afectada por el moho es causa directa para la superficie neta de hoja verde y ésta a su vez influye directamente en el microclima, pero de nuevo éste influiría de manera directa en la cantidad de hoja afectada por el moho. Este tipo de relaciones no se pueden representar directamente en una red de creencia ya que se establecería un bucle dirigido no permitido.

Por estas razones ha de plantearse la utilización de un modelo gráfico que permita representar sin ningún problema las situaciones que hemos planteado y realizar procesos de razonamiento válidos.

Se han desarrollado extensiones de las redes de creencia para tratar de forma explícita el tiempo dando lugar a las redes de creencia dinámicas, que caracterizan la evolución temporal del sistema mediante un modelo de evolución que establece las dependencias temporales entre las variables del sistema en cuestión. Para nuestro ejemplo, una modelización temporal del sistema de producción podría parecerse al de la figura 1.8. Con este modelo desde el instante de tiempo $t = 0$ *hasta* $t = n - 1$*, es decir, desde el inicio del proceso dinámico hasta el final del mismo, nos permitiría resolver las cuestiones que planteábamos en un principio, estas son, tener en cuenta la evolución histórica de las observaciones y por otra parte establecer relaciones temporales entre las variables de una manera explícita.*

■

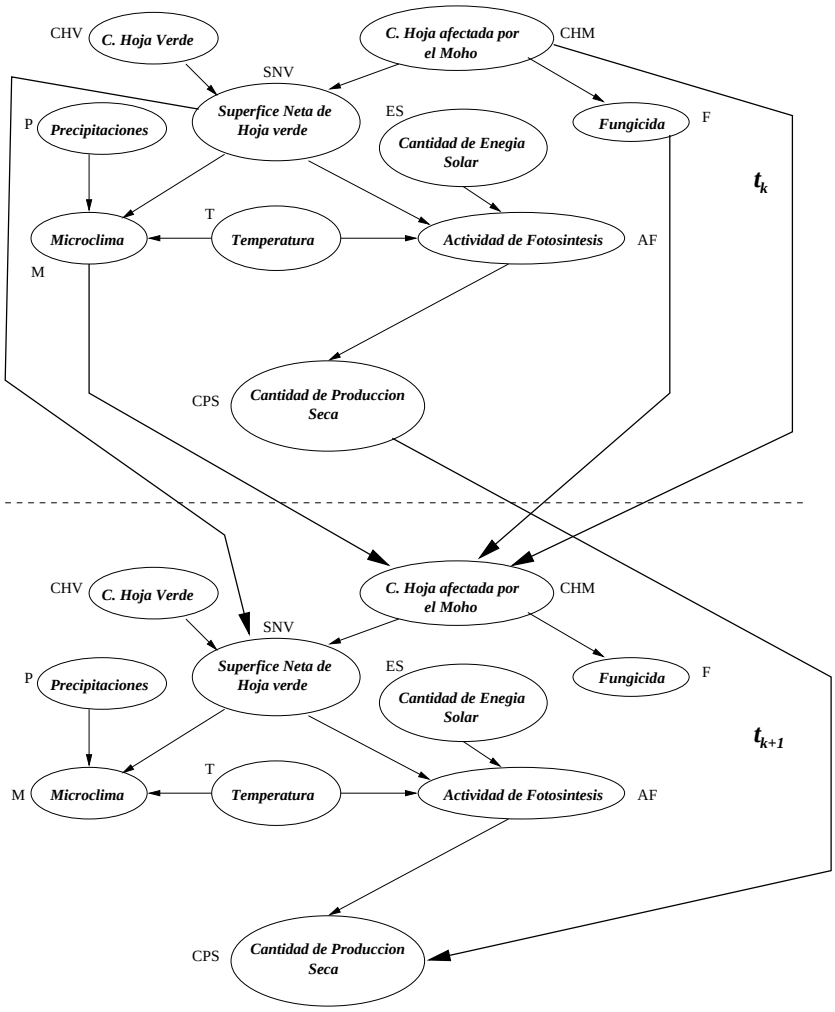


Figura 1.8: Red de creencia dinámica para un sistema de producción de trigo.

Se han desarrollado modelos dinámicos para el razonamiento temporal probabilístico, tales modelos pueden ser aplicados a un gran campo de aplicaciones como la predicción, control, planificación, problemas de simulación, etc.

Los investigadores en el campo de la estadística han desarrollado numerosos métodos para razonar sobre las relaciones temporales entre las variables que describen un modelo. Este campo, generalmente conocido como *análisis de series temporales*, consiste en una colección de muestras de un proceso evolutivo estocástico, es decir, un conjunto de observaciones que se realizan secuencialmente conforme evoluciona el tiempo. Se han obtenido buenos métodos para modelar y resolver este tipo de problemas cuando las relaciones temporales que se describen no son complejas y son lineales. Solo recientemente se han unido de alguna forma este último campo y el del estudio de la representación del conocimiento incierto mediante redes de creencia, dando lugar al modelo de red de creencia dinámica [15, 50, 70, 89, 102, 107, 115, 114, 116, 32, 33].

En general, en un modelo dinámico, consideraremos un conjunto de variables aleatorias $X(k)$, que describen el estado del sistema en el instante de tiempo discreto, $t = k$, como por ejemplo la variable temperatura del ejemplo anterior. En estos modelos nos interesan conocer las creencias relacionadas con un mundo cambiante. Si tenemos la evolución histórica de una determinada observación desde $t = 0, \ldots, t = k$, incluido éste, tendremos una serie de observaciones para un conjunto de variables O en cada periodo de tiempo t, esto es, $O(0), \ldots, O(k)$. El primer problema que se nos puede plantear solucionar será el de calcular la creencia del estado del sistema en el instante de tiempo $t = k$, en base a la evidencia acumulada hasta $t = k$. En términos de probabilidad será calcular la expresión:

$$P(X(k)|O(0), \ldots, O(k))$$

Calcular la expresión anterior de manera directa puede ser bastante complejo, así que podemos simplificar bastante su cálculo si consideramos que el problema es de tipo markoviano, esto es, la distribución del estado actual depende exclusivamente del estado anterior y del conjunto de observaciones actuales. En términos de probabilidad esto quiere decir que:

$$P(X(k)|X(0), \ldots, X(k-1), O(0), \ldots, O(k)) = P(X(k)|X(k-1), O(k))$$

A este tipo de modelos dinámicos markovianos se les denomina en la literatura estadística *Modelos Dinámicos Markovianos Parcialmente Observables*, MDMPO, modelos que se caracterizan por tener un conjunto de observaciones en cada instante de tiempo. En estos modelos el conjunto de observaciones en $t = k$ solo depende del estado actual del sistema, es decir de $X(k)$, en términos de probabilidad:

$$P(O(k)|X(0), \ldots, X(k), O(0), \ldots, O(k-1)) = P(O(k)|X(k))$$

Fijémonos en que parece razonable pensar que el conjunto de observaciones en $t = k$ nos ayude a estimar el estado actual del proceso dinámico junto con el estado previo del sistema. Las expresiones que hemos visto nos llevan de una manera natural a definir un modelo gráfico que establezca las relaciones comentadas.

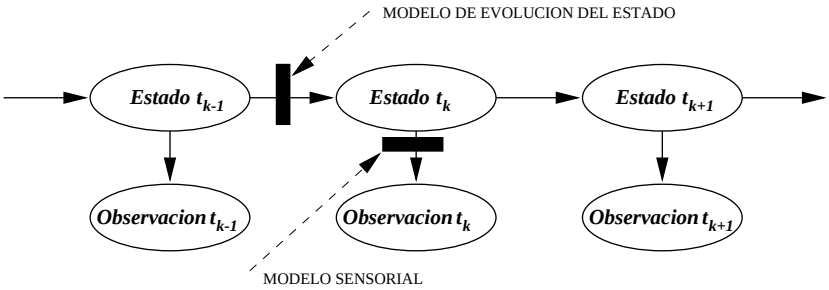


Figura 1.9: Modelo Gráfico para un MDMPO.

En la figura 1.9, el modelo de evolución del estado corresponde al modelo de transición entre estados del sistema, en términos de probabilidad se corresponde con la distribución $P(X(k)|X(k-1))$ y el modelo sensorial se corresponderá a la distribución $P(O(k)|X(k))$.

El conjunto de expresiones que hemos visto hasta ahora nos permite simplificar de manera significativa el cálculo correspondiente a la estimación del estado actual del sistema $P(X(k))$. El cálculo se puede realizar en dos fases: (a) Fase de predicción y (b) Fase de estimación. Estas dos fases son una generalización de las técnicas bien conocidas en el análisis de series temporales con el nombre de *filtros de Kalman* (Kalman filters). Estas técnicas se aplican universalmente en problemas de monitorización y control de todo tipo de sistemas dinámicos, desde plantas químicas hasta proyectiles dirigidos.

El cálculo de $P(X(k))$ se podrá realizar de la siguiente forma:

- Fase de Predicción: Primero, se predice la distribución de probabilidad en aquellos estados que habríamos esperado, con base al conocimiento que disponemos acerca del estado anterior:

$$P(X(k)) = \sum_{X(k-1)} P(X(k)|X(k-1))P(X(k-1))$$

- Fase de Estimación: Tenemos ahora una distribución que se extiende a través de las variables de estado actuales, basada en todo menos en las observaciones recientes. La fase de estimación actualiza lo anterior a través de la observación en el instante $t = k$:

$$P(X(k)|O(k)) = \alpha P(O(k)|X(k))P(X(k))$$

en donde α es una constante de normalización.

1.4.2 Conceptos básicos. Definición formal del modelo

En este punto del trabajo vamos a estudiar y definir de una manera formal cómo se puede representar la evolución temporal del estado de un sistema dinámico mediante redes de creencia. Para ello vamos a seguir la notación y el modelo desarrollado por Kærulff en el trabajo [88].

Formalmente una Red de Creencia Dinámica (RCD) extiende la representación de una red de creencia para modelar procesos temporales. Asumiremos que los cambios de estado ocurren entre periodos de tiempo discretos, los cuales van a ser indexados por enteros no negativos. Así $x_i(k) \in V(k)$ representará el estado de la variable x_i en el periodo de tiempo $t = k$, y $V(k)$ representará el conjunto de variables aleatorias en el periodo de tiempo $t = k$. También asumiremos que hay un número finito de periodos de tiempo n, esto es, el índice k tomará sus valores en un rango finito $[0,(n-1)]$.

De esta forma una RCD puede ser definida como $G = (V,E)$ un grafo dirigido acíclico que describe la estructura del modelo dinámico. Si $t = 0$ es el primer modelo de la red, entonces V lo forman los subconjuntos disjuntos, $V(0),...,V(n-1)$. Es decir,

$$V = V(0,(n-1)) = \bigcup_{t=0}^{n-1} V(t)$$

Al conjunto de arcos dirigidos

$$E^{tmp}(k) = \{x_i(k-h) \rightarrow x_j(k)\} \subset E$$

donde, $0 < k \leq n-1$ y $0 < h \leq k$, se le denominan *arcos temporales* o *relaciones temporales* en el periodo de tiempo $t = k$, los cuales definen el modelo de transición del sistema dinámico.

De esta forma el conjunto de arcos E en G puede ser definido como:

$$E = E(0,(n-1)) = E(0) \cup \bigcup_{t=1}^{n-1} E^*(t),$$

donde $E(t) \subseteq V(t) \times V(t)$ y $E^*(t) = E(t) \cup E^{tmp}(t)$. De esta forma, la estructura gráfica del modelo en el periodo de tiempo t será: $G(t) = (V(t),E(t))$.

Para representar las creencias sobre las posibles trayectorias del proceso dinámico, necesitamos especificar una distribución de probabilidad sobre el conjunto de variables $V(0),V(1),\ldots,V(n-1)$. Esta distribución puede ser extremadamente

compleja, así que es muy usual asumir que el proceso dinámico que estamos representando es *markoviano*. En términos de probabilidad esto último quiere decir que:

$$P(V(k)|V(0),\ldots,V(k-1)) = P(V(k)|V(k-1))$$

es decir, el futuro es condicionalmente independiente del pasado dado el presente. Formalmente, y en términos de relaciones de independencia condicional lo podemos notar como:

$$I(\{V(0),...,V(k-1)\},\{V(k+1),...,V(n-1)\}|V(k))$$

para $k > 0$ y $n > 0$. Esto implicará que el conjunto de relaciones temporales $E^{tmp}(k)$ contendrá solo arcos entre periodos consecutivos de tiempo. También es usual asumir que el proceso dinámico que queremos modelar es *estacionario*, esto quiere decir que $P(V(k)|V(k-1))$ es independiente del índice k. Las consideraciones previas dan lugar a que en cada nodo $x_i(k)$ tengamos almacenadas las distribuciones de probabilidad condicionales $P(x_i(k)|\pi_k(x_i(k)) \cup \pi_{k-1}(x_i(k)))$ donde $\pi_k(x_i) = \{\pi(x_i) \cap V(k)\}$, es decir, el conjunto de padres de x_i restringido al periodo de tiempo k. Estas distribuciones de probabilidad condicional serán las mismas en todos los periodos de tiempo. Esto, además, implicará que las estructuras $G(t)$ sean también las mismas en todos los periodos de tiempo (Figura 1.10).

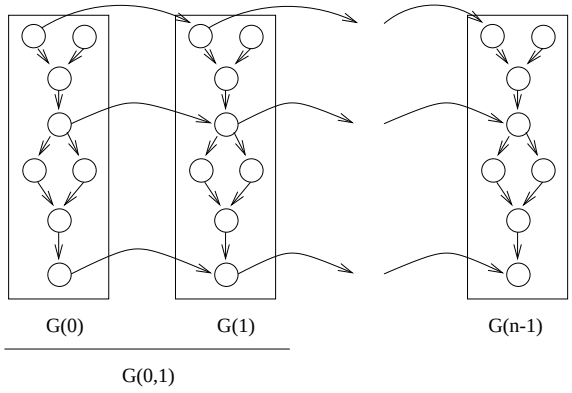


Figura 1.10: Red de Creencia Dinámica. Los arcos temporales aparecen como curvas. $G(0,1)$ representa la unión de los dos primeros periodos de tiempo.

Definiremos ahora la estructura gráfica resultante de la unión de dos periodos de tiempo consecutivos como $G(k,k+1) = (V(k,k+1),E(k,k+1))$, donde $V(k,k+1) = V(k) \cup V(k+1)$ y $E(k,k+1) = E(k) \cup E(k+1) \cup E^{tmp}(k+1)$.

Anteriormente hemos definido el conjunto de padres de un nodo x_i restringido a un periodo de tiempo k. De igual forma se podrá definir el conjunto de hijos de un nodo x_i restringido a un periodo de tiempo k.

En definitiva, una RCD se puede considerar como un grafo dirigido acíclico, en donde un conjunto de variables se repiten en el tiempo y el conjunto de arcos está compuesto por los arcos temporales que interconectan periodos de tiempo consecutivos y arcos no temporales que interconectan las variables en el mismo periodo de tiempo. Esta estructura gráfica representa una distribución de probabilidad conjunta sobre todas las posibles trayectorias del sistema dinámico, el cual consiste básicamente en dos partes:

- Una red inicial $G(0)$ que especifica una distribución sobre el estado inicial del proceso temporal.
- Un modelo de transición definido sobre $G(k,k+1)$, que especifica las probabilidades de transición entre estados del proceso temporal.

1.4.3 Aprendizaje de redes de creencia dinámicas

Existen muy pocos trabajos en la literatura especializada en aprendizaje de redes de creencia que traten el aprendizaje estructural de redes de creencia dinámicas ([47]). Existen bastantes más trabajos ([22, 90]) para el aprendizaje paramétrico en este tipo de estructuras dinámicas.

Friedman y col [61] desarrollan una extensión de una métrica Bayesiana (BDe ecuación 1.11) y una medida de información (BIC ecuación 1.14) para tratar con modelos dinámicos markovianos y estacionarios. Posteriormente tratan el aprendizaje del modelo como un problema de optimización de la misma manera que el aprendizaje de una red de creencia estática (usaremos esta denominación en contraposición a las redes de creencia dinámicas) restringido al grafo compuesto por dos periodos de tiempo consecutivos $G(k,k+1)$. Para este caso sólo es necesario restringir la búsqueda de forma adecuada en el proceso de optimización, de tal forma que los enlaces temporales sólo tengan una posible orientación (orden temporal), y que si encontramos una relación no temporal entre dos variables del mismo periodo de tiempo $x_i(k) \rightarrow x_j(k)$, ésta se replique en el periodo de tiempo siguiente $t = k+1$.

Para estos casos debemos asumir que tenemos definida una base de datos de casos independientes:

$$D = \{\mathbf{v}(0)^1, \ldots, \mathbf{v}(0)^m, \mathbf{v}(1)^1, \ldots, \mathbf{v}(1)^m, \ldots, \mathbf{v}(n-1)^1, \ldots, \mathbf{v}(n-1)^m\}$$

de m casos, donde $\mathbf{v}(k)^i$ representa una instanciación de las variables de $V(k)$ en el caso i-ésimo de la base de datos D. A partir de estos casos D y teniendo definida una métrica $f(G:D)$, donde G representa una red de creencia dinámica, el problema

del aprendizaje se puede formular del mismo modo que el aprendizaje de redes de creencia estáticas. Friedman y col [61] demuestran que si tenemos un sistema dinámico markoviano y estacionario entonces nos podemos centrar en el aprendizaje de dos periodos de tiempo consecutivos $t = k$ y $t = k+1$, esto es, restringirnos a aprender el grafo $G(k,k+1)$. Una vez estimado este modelo, podemos replicar el mismo para el resto de periodos de tiempo.

Teniendo en cuenta el anterior planteamiento, podemos decir que cuanto más eficaz y eficiente sea un método de aprendizaje (basado en métricas) de redes de creencia estáticas, normalmente, más eficaz y eficiente será este método para el aprendizaje de redes de creencia dinámicas. Es por ello que en los dos siguientes capítulos desarrollaremos métodos generales de aprendizaje de redes de creencia estáticas.

Por otra parte, existen una gran cantidad de redes de creencia dinámicas en aplicaciones reales ([57, 61, 138, 87]) en donde hemos podido observar que el número de relaciones no temporales entre variables del mismo periodo de tiempo es muy escaso, incluso es bastante habitual que no exista ninguna relación de este tipo. Es por ello, que estas relaciones puedan ser conocidas y fijadas a priori (bien por un experto, bien por algún algoritmo de aprendizaje restringido a $G(0)$). En este caso, podemos centrarnos en aprender las relaciones temporales a partir de los datos D supuesto que tenemos conocidas las relaciones no temporales. En el capítulo 4 nos centraremos en el estudio de esta última cuestión y desarrollaremos métodos de aprendizaje específicos para el aprendizaje de relaciones temporales.

Capítulo 2

Métodos de Búsqueda Local para el Aprendizaje de Redes de Creencia

2.1 Introducción

Muchos autores, [21, 76, 23, 31, 7],... han utilizado métodos de búsqueda local en el aprendizaje de redes de creencia basado en optimización de alguna métrica. Estos métodos se han mostrado eficientes en este tipo de problemas, obteniéndose buenas soluciones en general. Quizás los métodos locales más representativos de este tipo de búsquedas sean los métodos de ascensión de colinas (hill-climbing) en sus diferentes versiones, deterministas o estocásticas. El método clásico de ascensión de colinas se queda bloqueado en el primer máximo local encontrado, es por esto que se han planteado soluciones para escapar de estos máximos locales; uno de los métodos más empleado quizás para evitar este tipo de situaciones, sea la ascensión de colinas con reinicios aleatorios. En general, este método perturba de forma drástica una solución encontrada al problema para empezar de nuevo una ascensión de colinas con esta nueva solución perturbada, con la esperanza de encontrar una solución que mejore la actual.

La mayoría de las aplicaciones de este tipo de búsquedas al problema de la búsqueda de la red de creencia que mejor explica una base de datos de casos, se basan en el espacio de búsqueda de los grafos dirigidos acíclicos. Los operadores locales clásicos de transformación de un grafo hacia sus vecinos son los de inversión de un arco existente en el grafo, borrado de un arco existente y añadido de un arco nuevo al grafo, todo ello evitando formar ciclos dirigidos. El espacio de búsqueda de los

grafos dirigidos acíclicos es enorme y es por ello que se han planteado esquemas de búsqueda específicos para las redes de creencia, esto es, no tratando el grafo dirigido en sí sino una representación de las clases de equivalencia que estos grafos representan, y a partir de ello realizar transformaciones para movernos en el espacio de las clases de equivalencia que es mucho menor que el anterior [28, 6].

El éxito relativo de los esquemas de búsqueda local con reinicio aleatorio, ha hecho que nos planteemos en este capítulo un algoritmo basado en los principios del anterior esquema, pero realizando un reinicio más fundamentado desde el punto de vista teórico que el anterior caso. Por otra parte, recientemente ha aparecido un nuevo esquema de búsqueda local basado también en el anterior esquema, pero realizando reinicios sistemáticos en vecindades cada vez de orden superior si la búsqueda actual no ha tenido éxito. Esta metaheurística se denomina *Búsqueda en Entorno Variable* y nosotros estudiaremos su adaptación a nuestro problema y su comportamiento en su resolución.

Otros autores [97, 60, 42, 43, 49, 8]... han realizado estudios de esquemas de búsqueda no en el espacio de grafos dirigidos acíclicos sino en el espacio de órdenes compatibles con una red de creencia. En general, estos autores demuestran que el espacio de órdenes es mucho más "suave" que el espacio de búsqueda sobre grafos dirigidos acíclicos, esto quiere decir que es un espacio en donde no se presentan comportamientos muy diferentes para vecinos próximos, cosa que sí ocurre en el espacio de grafos dirigidos acíclicos. Además, el espacio de órdenes es también mucho menor que el espacio de grafos dirigidos acíclicos. Si bien esto último es ventajoso, también hemos de notar que evaluar un orden a través de las métricas habituales de redes necesita el paso intermedio de aprender una red compatible con el orden dado que mejor se adapte a nuestros datos y por consiguiente será mucho más costoso que realizarlo directamente sobre la estructura modificada localmente. Por consiguiente, esta evaluación habrá que realizarla de forma eficiente mediante alguna heurística. En este capítulo también nos dedicaremos a estudiar métodos locales de búsqueda en el espacio de órdenes.

Este capítulo se estructura como sigue: En primer lugar revisaremos la Metaheurística *Búsqueda en Entorno Variable* así como sus posibles extensiones. Seguidamente estudiaremos dos algoritmos de búsqueda local en el espacio de grafos dirigidos acíclicos con múltiples reinicios, el primero se basará en tests de independencia condicional y el segundo será una aplicación de la búsqueda en entorno variable al problema del aprendizaje de redes de creencia. A continuación veremos cómo realizar una búsqueda local en el espacio de órdenes entre las variables, donde estudiaremos cómo evaluar una secuencia de ordenación, un método de ascensión de colinas en el espacio de búsqueda de órdenes, su aplicación en el esquema de búsqueda en entorno variable y finalmente cómo obtener una buena ordenación inicial para comenzar la búsqueda en los anteriores métodos.

2.2 Búsqueda en Entorno Variable

En esta sección del capítulo vamos a revisar las reglas básicas del método de búsqueda en entorno variable (VNS, Variable Neighborhood Search) y alguna de sus extensiones para resolver problemas de optimización [104, 71]. Este método, en general, se puede encuadrar en los métodos de ascensión de colinas con múltiples reinicios. La novedad de esta metaheurística es la forma de realizar estos reinicios de las búsquedas posteriores.

Es conocido que el método de ascensión de colinas clásico queda bloqueado en el primer máximo local que se encuentre. Este no es el caso de otros métodos estudiados en la literatura, que proveen formas, deterministas o estocásticas, para escapar de estos óptimos locales. Uno de estos métodos de reciente aparición es la búsqueda local en entorno variable (VNS). Este método es bastante atractivo desde el punto de vista de la eficiencia además de ser bastante sencillo e intuitivo, y presenta una forma sistemática y razonable de realizar reinicios para posteriormente realizar una nueva ascensión de colinas clásica.

Este método se basa en el principio de búsqueda sistemática en entornos cada vez mayores de vecindad durante el proceso de búsqueda. En este método existe un paso intermedio basado en una ascensión de colinas clásica. Para realizar una búsqueda en entorno variable, en primer lugar se debe definir un conjunto finito y preseleccionado de estructuras de vecindad denotadas como $\mathcal{N}_k(k = 1,\ldots,k_{max})$. $\mathcal{N}_k(x)$ será el conjunto de soluciones en la $k-$ésima vecindad de x (una ascensión de colinas clásica toma como $k_{max} = 1$). Por ejemplo, si una solución x contiene l atributos, un vecino de x en la vecindad k-ésima podría estar formado a partir de k intercambios entre los l atributos. El VNS básico consistirá en los siguientes pasos:

1. Inicialización: Seleccionar el conjunto de estructuras de vecindad $\mathcal{N}_k$, $(k = 1,\ldots,k_{max})$, que utilizaremos durante la búsqueda; Seleccionar algún punto de partida x como inicio de la búsqueda; Seleccionar un criterio de parada.

2. Repetir los siguientes pasos hasta la condición de parada.

   (a) $k = 1$, Hasta que $k = k_{max}$, repetir los siguientes pasos.

      i. Generar un vecino x' aleatorio a partir de $\mathcal{N}_k(x)$ $(x' \in \mathcal{N}_k(x))$.

      ii. Aplicar una búsqueda local con x' de punto de partida hasta alcanzar un máximo local x''.

      iii. Si este máximo local x'' es mejor que el inicial x, entonces movernos a este máximo y fijarlo $x = x''$ y continuar la búsqueda con $k = 1$; en otro caso $k = k+1$.

La condición de parada puede ser seleccionada en base a: (a) Máximo tiempo de ejecución; (b) Máximo número de iteraciones del bucle externo; (c) Máximo número de iteraciones entre dos mejoras de la solución actual; etc.

Hemos de notar que la solución x' de la que partimos se genera de forma aleatoria con el objetivo de evitar ciclos que pudieran ocurrir si se eligiera de forma determinista. Será frecuente necesitar varias soluciones generadas de esta forma en vecindades sucesivas para poder escapar de estos máximos locales obtenidos por el paso de búsqueda local aplicada.

La idea de este esquema de búsqueda es la siguiente: Cuando estamos en un máximo local alcanzado mediante una búsqueda local con mínima vecindad, entonces saltamos a un vecino aleatorio inmediatamente de orden superior y de nuevo realizamos una búsqueda local con mínima vecindad partiendo desde este vecino aleatorio. Si mejoramos el máximo obtenido, entonces de nuevo buscamos con una vecindad de mínimo orden y si no mejoramos nos movemos a una vecindad de orden superior. La filosofía subyacente de este esquema es que si nos encontramos en un máximo (que puede estar cerca del óptimo) no distanciarnos demasiado (solo lo justo y cada vez más) de este máximo para seguir optimizando con la esperanza de que el óptimo quede cerca del anterior, es decir, aprovechar de una manera óptima la información que aporta un buen óptimo local de nuestro problema. En general, esta premisa se cumpliría en aquellos espacios de búsqueda en donde la función objetivo que intentamos maximizar (minimizar) es cuasi convexa, esto es, no existen valles alejados unos de otros en mucha distancia en nuestro espacio de búsqueda.

Una vez que tenemos un método apropiado al problema para realizar un búsqueda local, hemos de notar que es muy fácil implementar el resto de pasos del esquema VNS. Por ejemplo, si $\mathcal{N}_k$ se obtiene mediante k-intercambios de los atributos de una solución x, solo necesitaríamos añadir unas pocas líneas de código a un método de búsqueda local.

Teniendo en cuenta que un óptimo local dentro de una vecindad, en el esquema básico del VNS un óptimo local encontrado en la vecindad $\mathcal{N}_1$, no es necesariamente un óptimo dentro de otra vecindad $\mathcal{N}_k$, se podrían realizar cambios de vecindad dentro de la fase de búsqueda local. Esta búsqueda local se denomina VND (Variable Neighborhood Descent) y sus pasos son los siguientes:

1. Inicialización: Seleccionar el conjunto de estructuras de vecindad $\mathcal{N}_k$, $(k = 1,\ldots,k_{max})$ que utilizamos durante la búsqueda; Seleccionar algún punto de partida x como inicio de la búsqueda; Seleccionar un criterio de parada.

2. Repetir los siguientes pasos hasta que no se produzca ninguna mejora.

   (a) $k = 1$, Hasta que $k = k_{max}$, repetir los siguientes pasos.

      i. Buscar el mejor vecino x' a partir de $\mathcal{N}_k(x)$ $(x' \in \mathcal{N}_k(x))$.

      ii. Si este máximo local x' es mejor que el inicial x, entonces movernos a este máximo y fijarlo $x = x'$; en otro caso $k = k+1$.

El VNS básico es un método siempre ascendente (descendente), en el sentido de que no permitimos nunca un movimiento a una solución peor que el actual óptimo conseguido, salvo obviamente en los nuevos reinicios en la vecindad correspondiente. Sin mucho esfuerzo adicional, se podría transformar el esquema básico del VNS en un método ascendente-descendente si por ejemplo, permitimos aceptar una solución x'' con alguna probabilidad incluso si la solución es peor que el óptimo actual. Por supuesto esta versión está basada en el esquema del enfriamiento simulado (simulated annealing).

Otras versiones del esquema básico podrían ser:

- Buscar la solución x' como la mejor de entre h aleatoriamente generadas a partir de la estructura de vecindad $\mathcal{N}_k$.
- Introducir dos parámetros k_{min} y k_{step} que controlen de alguna forma el movimiento entre vecindades. Esto significa que en lugar de fijar $k = 1$ y $k = k+1$ fijemos $k = k_{min}$ y $k = k + k_{step}$ respectivamente. Estos parámetros guían la intensificación y diversificación en la búsqueda.
- Eliminar directamente la búsqueda local intermedia. Esta variante denominada VNS reducido, podría ser beneficiosa en problemas muy complejos en donde la búsqueda local fuese muy costosa. Este esquema está muy relacionado con los métodos de Monte-Carlo, pero en un modo más sistemático en la generación de soluciones.

Cuando utilizamos más de una estructura de vecindad en nuestra búsqueda, como en el caso del VNS, deberíamos plantearnos las siguientes cuestiones:

- ¿Qué estructuras de vecindad y cuántas de ellas deberían ser utilizadas?
- ¿Cuál debería ser el orden en la búsqueda?
- ¿Qué estrategia debería utilizarse en el cambio entre vecindades?

Además de esto último debemos decidir qué método de búsqueda local debería ser utilizado en el paso apropiado del esquema VNS.

2.2.1 Otras extensiones

Algunos estudios sobre la distribución de los óptimos locales, para una variedad de problemas de optimización combinatoria, muestran que a menudo estos óptimos locales se encuentran concentrados en unos pocos “valles” de gran dimensión y en sus partes más altas (bajas). En otras palabras, hay una o muy pocas regiones en el espacio de búsqueda cuasi-convexas con algún ruido donde se concentran los óptimos locales. El esquema básico VNS, en principio, es más eficiente para buscar

óptimos en una de estas regiones (la primera que se explora), pero perdería eficacia en la búsqueda en otras regiones posiblemente con mejor óptimo local. Por otra parte, la información contenida en el mejor óptimo encontrado se va diluyendo en su importancia conforme nos vamos alejando en las estructuras de vecindad, así el esquema básico del VNS tenderá a degenerar en un esquema con reinicio aleatorio si k_{max} es demasiado grande.

Una extensión del VNS para intentar alcanzar otros "valles" a partir del actual óptimo local es la siguiente: En el esquema del VNS básico una solución $x' \in \mathcal{N}_k(x)$ es utilizada para reiniciar una búsqueda local, obteniéndose de nuevo un óptimo local x''. Si la mejora obtenida por la solución x'' no es lo suficientemente buena con respecto a x, esto es, $f(x'') - (+)f(x)$, dependiendo si estamos maximizando o minimizando, es una cantidad pequeña y además los atributos de las soluciones x'' y x se parecen demasiado, esto es, la distancia entre x'' y x es también pequeña, entonces podemos utilizar una corrección proporcional a la distancia entre las soluciones x y x'' y al valor de la función en x y al valor de la función en x'' en en el último paso del esquema VNS, para relocalizar la búsqueda y de esta forma forzar a buscar en zonas más alejadas del óptimo local (y del "valle" al que pertenece). El esquema modificado es el siguiente:

1. Inicialización: Seleccionar el conjunto de estructuras de vecindad $\mathcal{N}_k$, $(k = 1,\ldots,k_{max})$ que utilizamos durante la búsqueda; Seleccionar algún punto de partida x como inicio de la búsqueda y su valor $f(x)$; Fijar $x_{opt} = x$ y $f_{opt} = f(x)$; Seleccionar un criterio de parada.

2. Repetir los siguientes pasos hasta que la condición de parada se cumpla.

   (a) $k = 1$, Hasta que $k = k_{max}$, repetir los siguientes pasos.

      i. Generar un vecino x' aleatorio a partir de $\mathcal{N}_k(x)$ $(x' \in \mathcal{N}_k(x))$.
      ii. Aplicar una búsqueda local con x' de punto de partida hasta alcanzar un máximo local x''.
      iii. Si $f(x'') < (>) f_{opt}$ entonces fijar $f_{opt} = f(x'')$ y $x_{opt} = x''$.
      iv. Si $f(x'') + (-)\alpha\rho(x,x'') < (>) f(x)$ entonces fijar $x = x''$ y $k = 1$; En otro caso $k = k + 1$.

Para este esquema habrá que definir una constante α de proporcionalidad y una función de distancia entre soluciones ρ, como por ejemplo número de atributos distintos entre dos soluciones x y x'. En esta extensión del VNS básico lo que se pretende fundamentalmente es que si la mejora que obtenemos no es lo suficientemente "buena" con respecto al óptimo actual, entonces seguir explorando en vecindades más alejadas a este óptimo para poder así explorar otras regiones del espacio de búsqueda.

Otra segunda extensión al esquema básico del VNS va orientada a agilizar de manera notable el paso de búsqueda local, pretendiendo preservar la información contenida en el óptimo alcanzado. Esta extensión del VNS realiza una búsqueda local sólo en los atributos de la solución x que han sido cambiados en la etapa correspondiente. Para una solución dada x, generamos una solución x' de forma aleatoria en la vecindad $\mathcal{N}_k$, a partir de esta solución x' en lugar de realizar una búsqueda local en el espacio de búsqueda completo, la realizamos en el espacio de búsqueda de las variables fijadas de forma aleatoria. Los demás pasos del esquema básico del VNS permanecen inalterados. Los pasos de esta otra extensión son los siguientes:

1. Inicialización: Seleccionar el conjunto de estructuras de vecindad $\mathcal{N}_k$, $(k = 1,\ldots,k_{max})$ que utilizamos durante la búsqueda; Seleccionar algún punto de partida x como inicio de la búsqueda; Seleccionar un criterio de parada.

2. Repetir los siguientes pasos hasta que la condición de parada se cumpla.

   (a) $k = 1$, Hasta que $k = k_{max}$, repetir los siguientes pasos.

      i. Generar un vecino x' aleatorio a partir de $\mathcal{N}_k(x)$. $(x' \in \mathcal{N}_k(x))$. Sea $y = (x' \setminus x)$.

      ii. Aplicar una búsqueda local en el espacio de búsqueda definido por y con x' de punto de partida hasta alcanzar un máximo local y'. Sea $x'' = (x' \setminus y) \cup y'$.

      iii. Si este máximo local x'' es mejor que el inicial x, entonces movernos a este máximo y fijarlo $x = x''$ y continuar la búsqueda con $k = 1$; en otro caso $k = k + 1$.

2.3 Métodos de Búsqueda Local en el espacio de Grafos Dirigidos Acíclicos

En esta sección nos vamos a dedicar al estudio de algoritmos de búsqueda local en el espacio de los grafos dirigidos acíclicos, en adelante GDAs. Estos métodos son en general eficientes y en cierta medida eficaces, porque son capaces de encontrar buenas soluciones. Por este motivo nos hemos decidido a estudiar este tipo de algoritmos y proponer métodos basados en este tipo de estrategias. El problema del aprendizaje se podría plantear de la siguiente forma: Tenemos una base de datos de casos $D = \{\mathbf{v}^1,\ldots,\mathbf{v}^m\}$, donde $\mathbf{v}^i$ es un caso de la base de datos. Asumimos que tenemos definida una función de medida para un grafo dirigido acíclico dados los datos de la base de datos de casos $f(G : D)$. Sea $\mathcal{G}_n$ la familia de todos los grafos dirigidos acíclicos G con n nodos. Entonces el problema que queremos resolver será:

$$\text{Encontrar} \quad G^* = \arg\max_{G \in \mathscr{G}_n} f(G:D) \tag{2.1}$$

Una de las razones por las que los métodos de búsqueda local en el espacio de GDAs son eficientes es por la propiedad de descomposición que habitualmente presentan las métricas que se pretenden optimizar, como hemos destacado en el capítulo anterior.

Si en un método de ascensión de colinas con reinicios aleatorios el número de transformaciones aleatorias realizadas es grande, como ocurre habitualmente, la tasa de reutilización de los cálculos previamente realizados será pobre, además de que la ascensión de la colina actual será mucho más costosa, y es también muy probable que la red resultante de esta perturbación aleatoria sea bastante compleja (en el sentido de que el número de padres medio por variable sea elevado, e incluso que algún nodo tenga un número muy elevado de padres, con lo que los estadísticos nuevos a recalcular serán bastante complejos). Por el contrario si el número de transformaciones realizadas es bajo, se corre el riesgo de que no salgamos del máximo actual, es por esto que este número de transformaciones sea un parámetro de difícil ajuste.

Por estas razones, estudiaremos un algoritmo que intenta reiniciar una búsqueda local cuando ésta queda atrapada en un óptimo local, de una manera no totalmente aleatoria, sino intentando buscar un punto nuevo de inicio guiado por los datos que disponemos para el aprendizaje de redes, esto es, aprovechando nuestro conocimiento específico sobre redes de creencia para reiniciar la búsqueda con una buena solución.

En segundo lugar, estudiaremos la adaptación de la metaheurística de búsqueda en entorno variable para el aprendizaje de redes de creencia en el espacio de grafos dirigidos acíclicos.

2.3.1 Un Algoritmo de Búsqueda Local híbrido con Reinicio basado en Tests de Independencia Condicional y Conjuntos Mínimos D-separadores

En esta subsección nos vamos a plantear el realizar una transformación en el reinicio de la ascensión no tan aleatoria y con una fundamentación teórica que nos permita obtener un punto de inicio de la ascensión razonable y probablemente cercano al óptimo (quizás no en tanto a la medida utilizada pero sí en cuanto a la estructura gráfica obtenida). Acid y Campos [4] estudian cómo se puede obtener de una manera eficiente en un grafo dirigido acíclico un subconjunto mínimo que d-separe dos nodos dados de una red de creencia; también hemos visto los conceptos de I-map e I-map minimal. Éstos serán los ingredientes esenciales del algoritmo que estudiaremos a continuación. Se puede demostrar de forma directa, a partir de la definición de I-map, la siguiente proposición.

Proposición 2.1 *Dado un grafo dirigido acíclico G que es un I-map de una distribución de probabilidad conjunta P, entonces para toda pareja de variables x_i y x_j no adyacentes en G, se verifica $I(x_i,x_j|S_d(x_i,x_j)_G)$, siendo $S_d(x_i,x_j)_G$ un conjunto mínimo d-separador de x_i y x_j en G.*

Esta proposición será la base fundamental de la transformación que llevaremos a cabo de un máximo local obtenido por la ascensión de colinas realizada en una etapa de nuestro algoritmo. Nuestra idea es comprobar si el grafo dirigido acíclico obtenido como máximo local es un I-map, para ello realizaremos todos los tests de independencia condicional con el conjunto mínimo d-separador, y si encontramos que alguna pareja de variables no son condicionalmente independientes dado el conjunto mínimo d-separador, entonces el grafo no será un I-map, con lo que tendríamos una razón para seguir buscando otros máximos locales. La pregunta que se plantea ahora es por cuál región de nuestro espacio de búsqueda seguir el proceso. El método que proponemos es el siguiente: Si durante la comprobación de una pareja de variables no adyacentes en el grafo (máximo local) encontramos que no son independientes, esto es, $\neg I(x_i,x_j|S_d(x_i,x_j)_G)$, entonces de alguna forma estamos habilitados (no de forma segura) para poner un arco entre ellos, evitando ciclos dirigidos. Por otra parte, y del mismo modo, si tenemos dos variables adyacentes en el grafo (máximo local), suponemos por un instante que realmente no son adyacentes y realizamos el test $I(x_i,x_j|S_d(x_i,x_j)_{G\setminus\{x_i-x_j\}})$ y éste resulta positivo, entonces de nuevo de alguna forma estamos habilitados (no de forma segura) para borrar este arco.

Siguiendo el esquema planteado previamente de transformación del grafo máximo actual, se puede notar que la introducción de un nuevo arco o el borrado de un arco existente puede influir en los tests posteriores que realicemos, es por ello que el orden en que efectuemos la comprobación de estos tests puede influir en la transformación final del grafo máximo actual. Por otra parte, es necesario realizar todos los posibles tests de independencia condicional ($O(n^2)$), esto es, una vez realizado el test $I(x_i,x_j|S_d(x_i,x_j)_G)$, es necesario realizar el test inverso $I(x_j,x_i|S_d(x_j,x_i)_G)$. Esto es así porque al realizar el segundo test puede que la red haya cambiado con respecto a la situación en el primero de los tests. Por otra parte, es necesario realizar las modificaciones del grafo máximo actual conforme el algoritmo planteado progresa, porque puede suceder que los arcos que se van introduciendo en el grafo influyan en los tests posteriores. Aclaremos estas cuestiones con el siguiente ejemplo.

Ejemplo 2.1 *Suponemos que la red objetivo que queremos aprender es $x_1 \rightarrow x_2 \rightarrow x_3$ y que nuestro grafo máximo local es el vacío; suponemos también que el orden de comprobación es el orden lexicográfico de las variables. Entonces procederíamos como sigue: realizaríamos el test $I(x_1,x_2|\emptyset)$, que resultaría negativo, entonces*

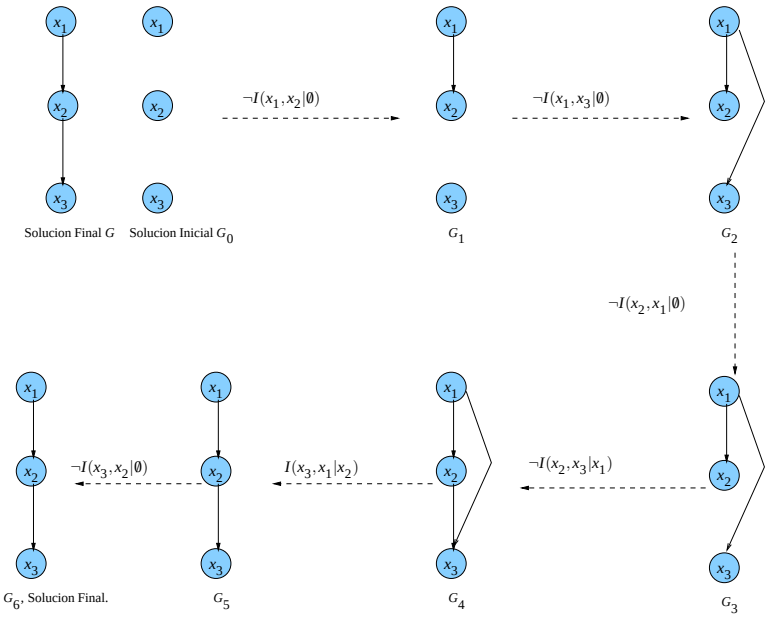


Figura 2.1: Ejemplo de Transformación del Algoritmo IMAPR.

pondríamos un arco entre ambos nodos, por ejemplo $x_1 \rightarrow x_2$*; a continuación realizaríamos el test* $I(x_1, x_3|\emptyset)$*, que también resultaría negativo, con lo que pondríamos un arco entre los nodos comprobados, por ejemplo* $x_1 \rightarrow x_3$*. Posteriormente se realizaría el test* $I(x_2, x_1|\emptyset)$*, que resultaría negativo, con lo que mantendría el arco entre ambos nodos. En la siguiente fase se realizaría el test* $I(x_2, x_3|x_1)$ *que de nuevo resultará negativo, con lo que pondría el arco* $x_2 \rightarrow x_3$*. En la posterior etapa se lleva a efecto el test* $I(x_3, x_1|x_2)$ *que en este caso es positivo, por consiguiente borraríamos el arco* $x_1 \rightarrow x_3$*; y por último realizamos el test* $I(x_3, x_2|\emptyset)$*, que resultaría negativo y por tanto dejaríamos dicho arco, obteniendo en este caso la red que buscamos. Ver figura 2.1 para seguir el ejemplo gráficamente. Si el orden de comprobación fuese* $x_3x_1x_2$*, entonces obtendríamos un grafo completo, que también es un I-map del modelo original, aunque con un número mayor de arcos.*

■

En el anterior ejemplo se puede observar lo que hemos dicho previamente, es muy importante el orden de comprobación de los nodos y es necesario realizar todos los tests posibles. También hemos podido observar mediante el ejemplo que no sólo el orden de comprobación es un parámetro libre en el esquema, sino que también es

libre la elección de la orientación del arco que nuestro esquema decide añadir en el grafo. Por tanto, son dos parámetros que podemos elegir aleatoriamente para que el esquema de transformación no desemboque en un ciclo, situación que se podría dar si lo hiciésemos de forma determinista.

Una vez planteado el esquema de transformación que seguiremos del grafo resultante de una ascensión de colinas clásica en el espacio de GDAs, concretaremos estos conceptos en un algoritmo que denominaremos **IMAPR** (I-map Restart) y que podemos observar en la figura 2.2.

Algoritmo IMAPR

1. *Inicialización: Elegimos una solución inicial G.*
2. *Repetir hasta alcanzar un número de iteraciones.*
   (a) *Realizar una ascensión de colinas clásica en el espacio de estructuras partiendo del grafo G, hasta obtener un máximo local G'. Hacer $G := G'$.*
   (b) *Elegir una secuencia de ordenación entre las variables de forma aleatoria.*
   (c) *Repetir hasta realizar todos los tests de independencia condicional.*
      i. *Seleccionar x_i y x_j siguiendo el orden establecido. Calcular el subconjunto $S_d(x_i,x_j)_G$ borrando el arco previamente caso de que exista un arco entre las variables seleccionadas.*
      ii. *Si $\neg I(x_i,x_j|S_d(x_i,x_j)_G)$ y no existía un arco entre ambas variables, poner un arco entre ambas variables en G con dirección aleatoria, pero evitando formar ciclos dirigidos.*
      iii. *Si $I(x_i,x_j|S_d(x_i,x_j)_G)$ y existía un arco entre ambas variables, borrar este arco de G; en caso contrario, restablecer dicho arco.*

Figura 2.2: Algoritmo IMAPR

Los tests de independencia condicional se realizan, normalmente, utilizando un test de hipótesis basado en la distribución χ^2 [93, 92]. Para ello es necesario fijar un nivel de confianza, en nuestro caso se han fijado dos niveles de confianza distintos, dependiendo de si existe el arco en la estructura comprobada o no: si existe el arco hemos fijado un nivel de confianza elevado, 0.99; de esta forma estamos exigiendo que los datos confirmen de manera sólida la posible dependencia condicional encon-

trada (y facilitando la eliminación del arco en caso contrario). Por otra parte, si no existe el arco, entonces hemos elegido un nivel de confianza más bajo, 0.75; de esta forma somos menos exigentes con el test realizado, posibilitando la incorporación del arco comprobado. En ambos casos la idea es que se favorezca la modificación de la estructura actual, siempre y cuando dicha modificación no se contradiga de forma clara con los datos disponibles.

2.3.2 Búsqueda en Entorno Variable en el espacio de Grafos Dirigidos Acíclicos

En esta subsección nos proponemos aplicar la metaheurística VNS, descrita previamente, al problema del aprendizaje tal como lo hemos planteado al inicio de la sección. Para ello hemos de concretar las cuestiones que se plantean para poder aplicar a un determinado problema la citada metaheurística. Aplicaremos un VNS específicamente diseñado para buscar en el espacio de GDAs. Para realizar esto, necesitamos definir las estructuras de vecindad $\mathcal{N}_k$. En este caso, tenemos las siguientes transformaciones sobre G: *borrado* de un arco, *añadido* de un arco entre dos nodos no adyacentes, e *inversión* de un arco (evitando ciclos dirigidos). Se podría haber elegido algún otro tipo de transformaciones, aunque éstas últimas son las transformaciones clásicas en los métodos de búsqueda local basados en el espacio de GDAs. Una transformación, que pensamos puede dar buenos resultados, pensada específicamente para redes de creencia, puede ser la inversión de un arco en la cual los extremos de este arco tuviesen padres comunes y el borrado posterior de estos arcos entre los padres comunes y los extremos del arco invertido. Pensamos que será útil porque una gran cantidad de métricas habituales, sobre todo las "score-equivalentes", utilizadas en el aprendizaje de redes de creencia tienden a cruzar los padres de dos nodos extremos si hemos equivocado la dirección entre ambos, esto es, tienden de forma natural a realizar la operación de inversión de arcos en una red de creencia. Estas familias así formadas habitualmente tienen una buena medida y suelen caer en un óptimo local difícil de superar, por tanto con este operador de transformación lo que se pretende es intentar salir de estas situaciones, aunque se corre el riesgo de que después de esta transformación las familias de los nodos involucrados no mejoren transitoriamente la medida empleada y por tanto no se escogiese nunca realizar este tipo de transformación. Es por ello que pensamos que este tipo de transformación sería más útil en un esquema ascendente/descendente de búsqueda, esto es, métodos que eventualmente permitieran un empeoramiento del óptimo alcanzado. Lo que sí podremos hacer en el esquema VNS es que en el paso de generación aleatoria en un determinada vecindad elijamos con cierta probabilidad la inversión y borrado de padres comunes como otro operador de transformación del grafo óptimo actual. Veamos lo argumentado anteriormente en el siguiente ejemplo.

Ejemplo 2.2 *Hemos construido la red de creencia de la figura 2.3(a) con cuatro variables y hemos generado mediante muestreo lógico probabilístico una base de datos de 1000 casos que sigue la probabilidad conjunta expresada en esta red. Podemos observar en la figura 2.3(a) la medida K2(log) para esta base de datos generada y la estructura de la red. Si en un proceso de búsqueda llegamos a la estructura de la red de la figura 2.3(b)*[1]*, podemos observar que su medida no dista mucho de la anterior. En la figura 2.3(c) podemos ver que la medida empeora transitoriamente al realizar el esquema de transformación planteado en el párrafo anterior y es peor también que las posibles transformaciones (mejores) clásicas expresadas en las figuras 2.3(d)(e)(f). Sin embargo, si observamos la figura 2.3(h), podemos ver que introduciendo el arco entre las variables* x_1 *y* x_3 *la medida* $K2(log)$ *se acerca mucho a la de la red real 2.3(a).*

Por otra parte podemos observar en las figuras 2.3(d)(e)(f) que a partir de la figura 2.3(b) al aplicar las posibles transformaciones clásicas (mejores) no mejoramos la medida y por consiguiente entraríamos en un máximo local.

■

Para la aplicación del VNS a nuestro problema, las estructuras de vecindad estarán definidas por la cardinalidad de la diferencia simétrica entre los conjuntos de arcos para dos grafos $G = (V,E)$ y $G' = (V,E')$:

$$\rho(G,G') = |(E \setminus E') \cup (E' \setminus E)| \tag{2.2}$$

es decir, $\rho(G,G') = k$, si hay exactamente k arcos que pertenecen a G y no a G' o viceversa. Entonces

$$G' \in \mathcal{N}_k(G) \Longleftrightarrow \rho(G,G') \leq k \tag{2.3}$$

El método de búsqueda elegido para el paso intermedio del VNS es la ascensión de colinas clásica en el espacio de GDAs, es decir, en cada paso de este método, aplicaremos la operación (borrado, añadido o inversión) que maximice la función $f(G:D)$, hasta que se alcance un óptimo local. A este algoritmo lo denominaremos HCST (Hill-Climbing for STructure).

La estrategia de búsqueda entre vecindades que vamos a usar es la empleada en el esquema del VNS básico ($k = 1$ y en cada iteración $k = k+1$) y también la segunda variante descrita anteriormente ($k = k_{min}$ y en cada iteración $k = k + k_{step}$). Como criterio de parada del algoritmo, usaremos el número máximo de iteraciones

[1] situación que se puede dar frecuentemente, porque imaginemos por un instante que la relación más fuerte entre las variables conforme a los datos es la de las variables x_2 y x_3 y partimos de la red vacía de enlaces, entonces una métrica "score-equivalente" no podría distinguir entre ambas orientaciones y podría elegir la orientación incorrecta. Posteriormente, normalmente, el proceso continuaría hasta cruzar los padres de ambas variables.

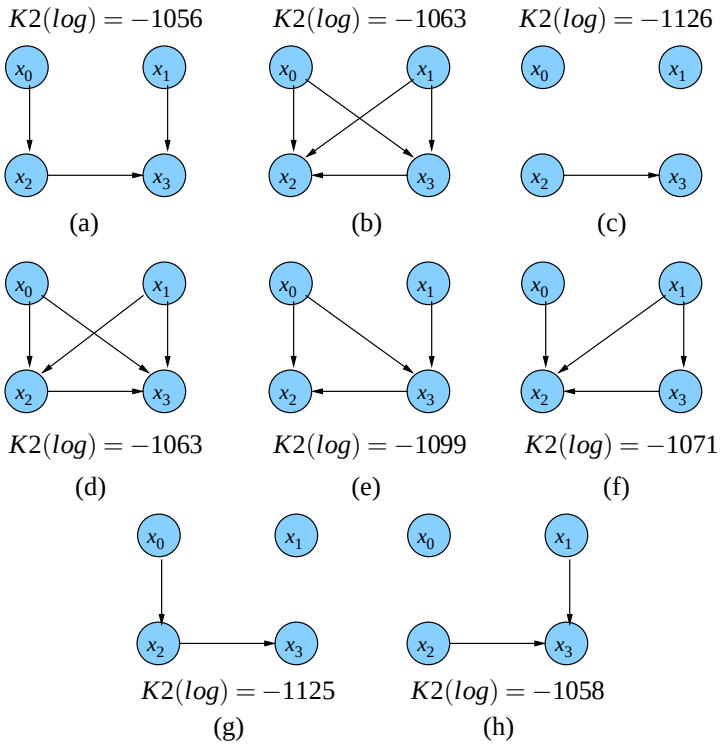


Figura 2.3: Medidas $K2(log)$ para las estructuras del ejemplo anterior.

entre dos mejoras del máximo actual, junto con un número máximo de iteraciones totales. Como k_{max} (el cual es un parámetro libre) en principio se puede utilizar el número de variables, n. El Algoritmo, que denominaremos VNSST, se puede seguir en la figura 2.4.

Algoritmo VNSST

1. *Inicialización: Fijamos la estructura de vecindades definidas por la ecuación 2.3. Fijamos k_{max}, k_{min} y k_{step}. Elegimos una solución inicial dada G. Fijamos número máximo de iteraciones, It.*
2. *Ejecutamos el algoritmo HCST con esta solución. Fijar G_{max}.*
3. $n = 1$.
4. *Repetir hasta alcanzar la condición de parada.*
   ($n > It$ ó dos iteraciones sin mejoras).
   (a) $k = k_{min}$ *y* $n = n + 1$.
   (b) *Repetir hasta que* $k = k_{max}$
      i. *Generar de forma aleatoria una G' de vecindad k ($G' \in \mathcal{N}_k(G_{max})$).*
      ii. *Aplicar el algoritmo HCST con G' como punto de inicio, dando lugar a otro óptimo local G''.*
      iii. *Si $f(G'' : D)$ es mejor que $f(G_{max} : D)$, entonces; $G_{max} = G''$ y $k = k_{min}$; en otro caso $k = k + k_{step}$.*

Figura 2.4: Algoritmo VNSST

2.4 Métodos de Búsqueda Local en el espacio de Órdenes

En esta sección nos dedicaremos a estudiar y proponer métodos de aprendizaje de redes de creencia basados en la búsqueda local de la red máxima a posteriori dados los datos D, pero en lugar de realizarlo en el espacio de grafos dirigidos acíclicos lo realizaremos en el espacio de órdenes entre las variables. Para un grafo G, dado un orden causal σ, esto es, si existe un arco entre dos variables $x_i \rightarrow x_j$, entonces $x_i <_\sigma x_j$, entonces x_i es condicionalmente independiente del resto de variables que la

preceden en el orden dados sus padres $\pi_G(x_i)$, para todo x_i. De esta forma podemos construir una red de creencia de una forma sistemática: Para cada x_i, los padres de x_i en un GDA son el conjunto mínimo de predecesores de x_i (en el orden σ) el cual hace x_i condicionalmente independiente del resto de predecesores.

Sin embargo, diferentes órdenes pueden producir diferentes redes. Entonces se prefieren aquellas redes que son capaces de representar cuantas más relaciones de independencia mejor. Por esta razón tiene sentido buscar un buen orden causal entre las variables.

Nuestro espacio de búsqueda en este caso será el conjunto de $n!$ órdenes, σ, entre las variables del sistema (esto es, el conjunto de todas las permutaciones de n elementos). Este espacio de búsqueda, [60], es mucho menor que el espacio de grafos dirigidos acíclicos, $n!$ frente a [121]:

$$f(n) = \sum_{i=1}^{n} (-1)^{i+1} \left(\begin{array}{c} n \\ i \end{array} \right) 2^{i(n-i)} f(n-i) \text{ donde } f(0) = 1 \text{ y } f(1) = 1 \qquad (2.4)$$

También en diferentes trabajos se demuestra que tiene un comportamiento más "suave" (menos máximos locales y permutaciones vecinas tienen medidas parecidas), frente al espacio de grafos dirigidos acíclicos (redes vecinas tienen medidas muy diferentes), en definitiva tiene un comportamiento en general mucho más homogéneo que el espacio de grafos dirigidos acíclicos [97, 42].

De una manera más formal, el problema de aprendizaje de redes de creencia basado en órdenes se puede formular como: Tenemos una base de datos $D = \{\mathbf{v}^1, \ldots, \mathbf{v}^m\}$, que contiene m instancias de V. Asumiremos que tenemos una métrica descomponible $f(G:D)$ para GDAs. Sea Σ_n el conjunto de todos los órdenes de n elementos y $\mathscr{G}_\sigma$ la familia de todos los GDAs G cuyo conjunto de vértices es V y cuyos arcos son compatibles con el orden σ. Entonces el problema considerado es:

$$\text{Encontrar} \quad \sigma^* = \arg\max_{\sigma \in \Sigma_n} f(\sigma : D) \qquad (2.5)$$

donde

$$f(\sigma : D) = f(G_\sigma : D) = \max_{G \in \mathscr{G}_\sigma} f(G : D) \qquad (2.6)$$

De esta forma, primero buscamos el mejor GDA, G_σ (de acuerdo a la métrica seleccionada f), compatible con un orden σ, y posteriormente seleccionamos el orden σ^* que ha producido el mejor GDA. El GDA G_{σ^*} es la solución deseada para nuestro problema de aprendizaje. Vamos a resolver este problema desde un punto de vista heurístico: los dos procesos de optimización se resuelven utilizando métodos de búsqueda heurísticos.

Para definir un método de búsqueda local, hemos de definir un operador de vecindad entre dos configuraciones σ_k y $\sigma_{k'}$ con mínimo cambio entre ambas a través del cual nos moveremos en el espacio de búsqueda definido. En nuestro caso al

proponer el espacio de búsqueda de órdenes entre las variables, nuestro operador natural será el intercambio entre dos posiciones i y j dentro de la secuencia que define nuestro orden entre las variables, esto es:

$$\sigma_k = (x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n) \rightarrow \sigma_{k'} = (x_1, \ldots, x_j, \ldots, x_i, \ldots, x_n)$$

Más formalmente, si σ es la actual configuración, entonces las $n(n-1)/2$ configuraciones vecinas de σ son aquellas $\sigma_{i,j}$, donde $i < j$, definidas como:

Sean x_u y x_v tal que $\sigma(x_u) = i$ y $\sigma(x_v) = j$. Entonces

$$\sigma_{ij}(x_k) = \begin{cases} \sigma(x_k) & \text{si } x_k \neq x_u \text{ y } x_k \neq x_v \\ j & \text{si } x_k = x_u \\ i & \text{si } x_k = x_v \end{cases} \tag{2.7}$$

Una vez definido el operador local entre secuencias de órdenes vecinas, necesitamos definir cómo evaluar estas secuencias. En nuestro caso será medir la probabilidad a posteriori de una base de datos de casos D dada la secuencia σ. Nuestra propuesta es usar una métrica, f, definida para GDAs y realizar una búsqueda en el espacio de GDAs compatibles con el orden σ. El valor de la función para el grafo así obtenido G_σ, $f(G_\sigma : D)$, será el valor de la medida para la secuencia σ ($f(\sigma : D) = f(G_\sigma : D)$). En otras palabras, tenemos que buscar el mejor conjunto de padres para la variable x_i entre las variables que le preceden en el orden σ. La búsqueda del conjunto de padres para una variable x_i puede realizarse de forma independiente de la búsqueda de los padres del resto de variables.

Por tanto, si la métrica f es una métrica descomponible, deberíamos intentar aprovechar esta propiedad para reducir la complejidad de la evaluación de una secuencia σ_{ij}, usando toda la información previa del cálculo de la configuración anterior σ.

Durante una búsqueda local tal y como la hemos planteado, vamos a realizar una permuta entre dos posiciones i y j de la secuencia mejor que tengamos en una iteración del algoritmo. Entonces hemos de notar que antes del índice i y después del índice j los cálculos serán los mismos que tuviésemos en la etapa anterior y por tanto no hará falta realizarlos de nuevo. Teniendo en cuenta esta situación, la primera forma directa de evaluar de forma heurística una secuencia dada, sería utilizar una heurística local para encontrar los padres de una variable x_h, entre los índices i y j permutados, de entre las variables que le preceden en el orden σ_{ij}. Una forma de realizarlo eficazmente es utilizando la heurística K2, esto es, iniciamos el proceso con el conjunto de padres vacío y vamos incorporando en cada etapa el padre que maximice nuestra medida hasta que no se pueda mejorar más la medida de la familia correspondiente $(x_h \cup \pi_{\sigma_{ij}}(x_h))$. A continuación detallamos los pasos de la función de evaluación:

1. Para aquellos nodos $x_a <_{\sigma_{ij}} x_v$, $f(x_a, \pi_{\sigma_{ij}}(x_a)) = f(x_a, \pi_\sigma(x_a))$.

2. Para aquellos nodos $x_p >_{\sigma_{ij}} x_u$, $f(x_p, \pi_{\sigma_{ij}}(x_p)) = f(x_p, \pi_\sigma(x_p))$.
3. Para aquellos nodos $x_v \leq_{\sigma_{ij}} x_h \leq_{\sigma_{ij}} x_u$, realizar los pasos siguientes:
   (a) Borramos el conjunto de padres calculado en la etapa anterior para una variable x_h, e iniciamos $\pi_{\sigma_{ij}}(x_h) = \emptyset$.
   (b) Elegiremos entre los nodos $x \notin \pi_{\sigma_{ij}}(x_h)$, anteriores en el orden σ_{ij} nuevo aquél que nos maximiza la medida $f(x_h, \pi_{\sigma_{ij}}(x_h) \cup \{x\})$.
   (c) Nos quedamos con este máximo y actualizamos el conjunto de padres de x_h incluyendo el nuevo nodo padre, $\pi_{\sigma_{ij}}(x_h) = \pi_{\sigma_{ij}}(x_h) \cup \{x\}$.
   (d) Repetimos los pasos (b) y (c) hasta que no consigamos mejorar nuestra medida f para x_h.

Sin embargo, hemos de notar que incluso para las variables comprendidas entre los índices permutados, incluidos éstos, los padres de estas variables calculados en la etapa anterior del algoritmo pueden ser compatibles con el nuevo orden e incluso con algo de suerte puede que sean los mismos que en la etapa anterior, ya que el intercambio entre secuencias es mínimo, así que, con la esperanza de realizar los menos cálculos posibles, vamos a proponer una heurística inspirada en la filosofía de búsqueda de padres del proceso anterior entre los índices permutados incluidos éstos, pero intentando reutilizar parte de los cálculos efectuados en la etapa anterior. Los pasos para la evaluación de una nueva secuencia σ_{ij} a partir de la anterior secuencia σ serán los siguientes:

1. Para aquellos nodos $x_a <_{\sigma_{ij}} x_v$, $f(x_a, \pi_{\sigma_{ij}}(x_a)) = f(x_a, \pi_\sigma(x_a))$.
2. Para aquellos nodos $x_p >_{\sigma_{ij}} x_u$, $f(x_p, \pi_{\sigma_{ij}}(x_p)) = f(x_p, \pi_\sigma(x_p))$.
3. Para aquellos nodos $x_v \leq_{\sigma_{ij}} x_h \leq_{\sigma_{ij}} x_u$, realizar los pasos siguientes:
   (a) Borramos del conjunto de padres calculado en la etapa anterior para una variable x_h, aquellos nodos $x \in \pi_\sigma(x_h)$ que no sean compatibles con el nuevo orden σ_{ij}, e iniciamos como máximo $f(x_h, \pi_\sigma(x_h) \setminus C)$, siendo C el conjunto de todos los nodos incompatibles con el orden σ_{ij} y $\pi_{\sigma_{ij}}(x_h) = \pi_\sigma(x_h) \setminus C$.
   (b) Elegiremos entre los nodos $x \in \pi_{\sigma_{ij}}(x_h)$ aquél que eliminándolo del conjunto de padres nos maximiza la medida $f(x_h, \pi_{\sigma_{ij}}(x_h) \setminus \{x\})$.
   (c) Elegiremos entre los nodos $x \notin \pi_{\sigma_{ij}}(x_h)$, anteriores en el orden nuevo σ_{ij}, aquél que nos maximiza la medida $f(x_h, \pi_{\sigma_{ij}}(x_h) \cup \{x\})$.
   (d) Nos quedamos con el máximo de las dos anteriores operaciones y eliminamos o insertamos un padre atendiendo al máximo de las dos, y escogemos como nuevo máximo éste último calculado.

(e) Repetimos los tres pasos anteriores hasta que no consigamos mejorar nuestra medida f para x_h.

Si nos fijamos detalladamente en el método propuesto de evaluación de secuencias, esta heurística es una ascensión de colinas en el espacio de GDAs, pero aprovechando la información proporcionada por la secuencia de ordenación y la propiedad de descomposición habitual en las métricas utilizadas, esto es, no tenemos un operador de inversión de arcos y al seguir una ordenación las variables, se pueden maximizar las familias de cada nodo de forma secuencial e independientemente siguiendo el orden σ_{ij}. Otra variante de esta última forma de evaluar secuencias podría ser la siguiente: en lugar de partir con el conjunto inicial de padres compatibles con el orden, partir de un conjunto vacío de padres. Creemos que su comportamiento sería, en general, similar a la primera forma de evaluar, pero tiene la ventaja de que permitiría borrar un padre de un nodo cuando se estime oportuno. Por otra parte, al iniciar el proceso de búsqueda de padres de cada nodo con el conjunto vacío, se pueden evitar situaciones de tipo siguiente: Suponemos que tenemos maximizada una familia de un nodo x_i en donde este máximo se consigue a costa de introducir más arcos de los necesarios porque esta variable en realidad está mal ordenada, entonces es probable que si mantenemos ciertos padres de esta variable compatibles con el orden no podamos mejorar este máximo encontrado, situación que podríamos intentar salvar calculando todos sus padres de nuevo. Este problema es el mismo que planteábamos en la sección 2.3.2, en la figura 2.3(b), cuando introducíamos una transformación específica para obtener la vecindad en el espacio de GDAs.

Hemos de notar también que, normalmente estas modificaciones locales en la secuencia de ordenación, implicarán una mayor movilidad entre los grafos dirigidos acíclicos compatibles con las secuencias de ordenación, esto quiere decir que un movimiento local entre dos secuencias puede resultar en que los grafos mejores compatibles con estas secuencias no sean vecinos en el espacio de grafos dirigidos acíclicos. También hemos de notar que, al contrario de lo que ocurre en los métodos locales basados en el espacio de grafos dirigidos acíclicos, un movimiento en el espacio de secuencias de ordenación será más costoso desde el punto de vista computacional, aún utilizando las heurísticas de evaluación presentadas previamente.

2.4.1 Método de Ascensión de Colinas en el espacio de Órdenes

Una vez estudiado cómo realizar la evaluación de una secuencia de ordenación σ, pasaremos a describir un algoritmo de búsqueda local para la búsqueda de una buena secuencia, y también de la red de creencia compatible con esta secuencia de ordenación. Este primer algoritmo será una búsqueda en la vecindad marcada por el operador descrito previamente, intercambios de dos variables en una secuencia, de

aquel movimiento que maximiza nuestra medida f, y si ésta mejora el máximo que llevamos hasta ese instante consolidamos el cambio e iteramos de nuevo el proceso hasta que no se pueda mejorar más, esto es, una ascensión de colinas clásica pero esta vez en el espacio de órdenes.

Es bien conocido que en este tipo de algoritmos juega un papel muy importante la inicialización del mismo, en nuestro caso la secuencia de partida y la red compatible con esta secuencia dada, pues de ello dependerá la evolución global del algoritmo, en cuanto a número de iteraciones y camino entre secuencias seguido. En nuestro caso se pueden elegir varias alternativas, como puede ser inicializar con el algoritmo PC [127, 34], extraer de él un orden topológico y posteriormente con este orden ejecutar el algoritmo K2 como red de inicialización; también podemos inicializar con una secuencia aleatoria y la red vacía de enlaces o con una secuencia aleatoria y el resultado de la ejecución del algoritmo K2, etc. Posteriormente en la subsección 2.4.3 estudiaremos cómo construir una buena solución inicial.

Con el objetivo de mejorar la eficiencia de estos métodos, vamos a restringir mediante un parámetro r (radio) los vecinos de una secuencia σ dada, para ello vamos a observar el siguiente hecho: Si la inicialización es buen punto de partida, es lógico pensar que permutas muy alejadas en una secuencia durante el proceso de búsqueda no van a estar muy "relacionadas" y por tanto no se han de esperar mejoras substanciales en nuestra medida. Por otra parte, los métodos propuestos de evaluación de secuencias de ordenación serán más eficientes cuanto menor diferencia exista entre los índices permutados. Por estas razones vamos a proponer introducir en la búsqueda local un parámetro r que define el radio de acción de las permutas entre las variables, de tal forma que las permutas buscadas no superen en ningún caso este radio de acción. Esto implicará no comprobar permutas entre variables que estén a una distancia en una secuencia mayor que este parámetro r. De una manera más formal, los vecinos admisibles para una secuencia σ son aquellas permutas $\sigma_{i,j}$, tal que la "distancia" entre las variables intercambiadas no es superior a r, esto es, $|j-i| \leq r$. Esto nos permitirá agilizar de manera considerable el cálculo de la máxima permuta en cada etapa del algoritmo ya que tendremos un número de vecinos inferior (exactamente $r(n-(r+1)/2)$) y los vecinos descartados son los más complejos de calcular. Un radio $r = n-1$ equivale a no introducir ninguna restricción. En cualquier caso, un intercambio entre dos nodos distantes en una secuencia σ es posible conseguirlo a través de varios intercambios más pequeños y nodos intermedios en la secuencia, por ejemplo, un intercambio de longitud $|j-i|$ puede ser obtenido por medio de tres intercambios de longitud $|j-i|/2$.

El pseudocódigo de este algoritmo, que denominaremos HCSN (Hill-Climbing for Sorting Nodes), se puede ver en la figura 2.5.

Para intuir e ilustrar el comportamiento del parámetro r propuesto en el párrafo anterior, vamos a realizar un pequeño experimento en el siguiente ejemplo.

Ejemplo 2.3 *Posteriormente en la sección 2.4.3 veremos cómo podremos construir una secuencia inicial para el proceso que hemos detallado previamente. El resultado de la aplicación de este algoritmo, que describiremos en la referida sección, en el dominio ALARM [31] para los* 3000 *primeros datos de la base de datos de casos ALARM, es la siguiente secuencia entre las variables:*

$\sigma = \{x_{21}, x_{19}, x_{23}, x_{12}, x_{28}, x_{18}, x_3, x_{22}, x_{24}, x_{31}, x_{30}, x_{17}, x_{16}, x_{37}, x_{36}, x_{35}, \mathbf{x_{32}}, x_{34}, x_{33}, x_{14}, x_{11}, x_{15}, x_{10}, x_{25}, x_2, x_1, x_{26}, \mathbf{x_7}, \mathbf{x_{29}}, x_9, x_6, x_8, x_{20}, \mathbf{x_{27}}, x_{13}, x_5, \mathbf{x_4}\}$

En negrita aparecen las variables que están mal ordenadas según un orden ancestral entre las variables de la estructura real de la red ALARM. Si nos fijamos detalladamente en dicha secuencia podremos observar lo siguiente: la variable x_{32} *está mal colocada porque debería de aparecer después en la secuencia que la variable* x_{34}*, que si nos damos cuenta está justamente a continuación, esto es, con radio 1. La variable con más distancia con las variables que deben precederle en el orden es la variable* x_{29} *a la que debe preceder la variable* x_{27} *y que en la secuencia se encuentra a una distancia de 5.*

En total, como podemos observar en la secuencia, hay solo 5 variables fuera de lugar o mal colocadas según un orden ancestral de la estructura de la red real ALARM.

Como pone de manifiesto este ejemplo, podemos intuir que si la secuencia de partida es buena, el efecto del parámetro radio r no va a influir de manera decisiva en el resultado del algoritmo que hemos planteado.

■

Algoritmo HCSN

1. *Inicialización: Elegimos una solución inicial* σ *y* G_σ.
2. *Repetir hasta que no se pueda mejorar nuestra solución* σ.
   (a) *Escoger aquella secuencia* σ' *que maximiza la medida* f*, mediante intercambios simples de parejas de variables en la secuencia* σ *de radio* r.
   (b) *Si la secuencia* σ' *es mejor que* σ*, entonces seguir iterando el proceso con* $\sigma = \sigma'$.

Figura 2.5: Algoritmo HCSN

Base de Datos ALARM (3000)					
radio	f(inicial)	f(final)	A	B	MO
1	-14,593.25	-14,409.77	4	2	0
5	-14,593.25	-14,409.25	6	2	3
7	-14,593.25	-14,408.67	1	2	0
36	-14,593.25	-14,408.54	2	2	0

Tabla 2.1: Resultados para la Base de Datos ALARM (3000)

Ejemplo 2.4 *Para ilustrar el comportamiento del algoritmo HCSN hemos realizado una ejecución del mismo sobre la red ALARM con los mismos 3000 datos de la base de datos ALARM usados en el anterior ejemplo. La medida que hemos utilizado es la métrica K2(log). Y para ilustrar el comportamiento del parámetro r, lo hemos fijado al máximo,* $r = 36$*, al mínimo* $r = 1$*, al máximo de la distancia entre las variables mal ordenadas* $r = 5$ *y también a* $r = 7$*. La secuencia de partida y la red de partida es la ofrecida por la heurística comentada en el ejemplo anterior, descrita en la sección 2.4.3. La métrica K2(log) para la red ALARM real para estos 3000 primeros casos es de* $-14.412.69$*. Los resultados de estas pruebas los podemos observar en la tabla 2.1. La primera columna es el radio r utilizado en la ejecución, la segunda columna es la métrica K2(log) inicial del proceso, esto es, el resultado de la heurística descrita en la sección 2.4.3, la tercera columna es la métrica K2(log) final que nos ofrece el algoritmo HCSN, la siguiente hace referencia al número de arcos añadidos con respecto a la red real ALARM, la siguiente el número de arcos borrados y la última el número de arcos mal orientados con respecto a la red real ALARM.*

■

2.4.2 Búsqueda en Entorno Variable en el espacio de Órdenes

En nuestro caso, para el problema que queremos resolver, la estructura de vecindad que podemos manejar será la siguiente: La vecindad mínima será la estudiada en la sección anterior:

$$\sigma_k \in \mathcal{N}_1(\sigma) \Longleftrightarrow \sigma_k(x_u) = \sigma(x_v), \sigma_k(x_v) = \sigma(x_u) \text{ y } \sigma_k(x_i) = \sigma(x_i) \forall x_i \neq x_u, x_v$$

$\mathcal{N}_2$ será definido como el intercambio de dos pares de posiciones dentro de una permuta, esto es:

$$\sigma_l \in \mathcal{N}_2(\sigma) \Longleftrightarrow \sigma_l \in \mathcal{N}_1(\sigma_k) \text{ y } \sigma_k \in \mathcal{N}_1(\sigma)$$

De forma similar:

$$\sigma_l \in \mathcal{N}_h(\sigma) \Longleftrightarrow \sigma_l \in \mathcal{N}_1(\sigma_k) \text{ y } \sigma_k \in \mathcal{N}_{h-1}(\sigma)$$

Como condición de parada hemos elegido la misma que el algoritmo VNSST. De esta forma, el algoritmo que denominaremos VNSSN, quedaría como el de la figura 2.6.

Algoritmo VNSSN

1. *Inicialización: Elegimos una solución inicial dada* σ *y* G_σ.
2. *Lanzar el algoritmo HCSN con esta solución inicial para alcanzar una nueva solución y fijar ésta como máximo.*
3. $n = 1$.
4. *Repetir hasta alcanzar la condición de parada.*
   (a) $k = k_{min}$ *y* $n = n + 1$.
   (b) *Repetir hasta que* $k = k_{max}$
      i. *Generar de forma aleatoria una* σ' *de vecindad k.* $(\sigma' \in \mathcal{N}_k(\sigma))$.
      ii. *Aplicar el algoritmo HCSN con* σ' *como punto de inicio dado lugar a otra secuencia localmente máxima* σ''.
      iii. *Si* σ'' *es mejor secuencia que* σ, *entonces movernos a esta secuencia* $\sigma = \sigma''$ *y* $k = k_{min}$; *en otro caso* $k = k + k_{step}$.

Figura 2.6: Algoritmo VNSSN

Otro factor que influirá en la búsqueda para nuestro caso particular será el radio r que impongamos en el algoritmo HCSN. Teniendo en cuenta que es otro parámetro libre en nuestro proceso podemos plantear también un esquema de actualización de este parámetro siguiendo la misma filosofía de la búsqueda en entorno variable VNS. Esto se puede traducir en un esquema de actualización del radio r cada vez mayor conforme nos movamos a vecindades mayores, volviendo a buscar con el radio inicial r cada vez que lo hagamos con vecindad mínima. La elección del incremento del radio r desde luego es de nuevo un parámetro libre y que tendremos que ajustar para cada problema, pero posibles elecciones de este esquema podrían ser: Incrementar en uno el radio r cada vez que incrementamos la vecindad k en k_{step}, volviéndolo a su valor original de partida cuando iniciamos de nuevo la vecindad

$k = k_{min}$. Otra posible opción podría ser, si tenemos un número fijo de iteraciones, incrementar nuestro radio r tal que en la última iteración se alcance el radio máximo, o simplemente un incremento fijo predeterminado. De esta forma lo que intentamos conseguir es la posibilidad de movernos con una mayor amplitud en el espacio de búsqueda cuando hayamos alcanzado un óptimo local y volver a un radio reducido cuando estemos en una "buena" solución.

Generación de una secuencia aleatoria para Redes de Creencia. En el paso apropiado del algoritmo VNSSN donde se genera una solución aleatoria $\sigma' \in \mathcal{N}_k(\sigma)$, se pueden elegir libremente parejas de variables (x_i, x_j) para ser intercambiadas, pero si pensamos más detenidamente en el problema que queremos resolver, nos damos cuenta que el orden topológico de una red de creencia es un orden parcial, esto quiere decir que puede que intercambiemos variables de una secuencia σ y la secuencia resultante sea equivalente, sea compatible también con la red aprendida, con lo que el cambio efectuado no será efectivo. Es por esta razón que nosotros realizaremos estos intercambios en un secuencia σ con un poco más de cuidado, de tal forma que forzaremos de alguna manera a que la secuencia resultante no sea equivalente en el sentido expresado previamente.

El proceso será el siguiente: En primer lugar elegiremos con probabilidad uniforme una variable de la secuencia x_i, posteriormente calcularemos el conjunto de variables descendientes de x_i asociadas a G_σ (el óptimo, GDA, local alcanzado en la etapa actual del algoritmo para la secuencia σ), elegiremos de forma uniforme una variable x_j de este conjunto y realizaremos el intercambio en la secuencia σ de estas dos variables así elegidas. Si x_i es un nodo hoja en la red, entonces realizamos el mismo proceso pero con los nodos ancestros de x_i.

2.4.3 El Problema del Punto de Inicio en la Búsqueda. Heurística K2SN

La heurística de inicialización, a la que hemos hecho referencia en la subsección 2.4.1, es una búsqueda greedy de un buen orden de partida entre las variables y su red asociada, buscada ésta como lo hace el algoritmo K2. Asumiremos de nuevo que tenemos definida una medida f descomponible, tal que $f(G:D) = \sum_{i=1}^{n} f(x_i, \pi_G(x_i))$, entonces lo que hará este algoritmo es lo siguiente: En primer lugar buscaremos aquél nodo (entre todos) que maximiza nuestra medida dado el conjunto vacío de padres, esto es, $f(x_i, \emptyset)$, y escogemos éste nodo como primero en el orden; por supuesto este nodo no tendrá a ningún otro como padre. A continuación escogeremos aquel nodo del resto de nodos que maximiza nuestra medida, pudiendo contener como padres los nodos que le preceden en el orden parcial que vayamos obteniendo. La forma de evaluar este máximo de nuevo será heurística y será como lo hace el algoritmo K2, esto es, se inicializa con el conjunto de padres

vacío e iremos insertando nodos en el conjunto de padres conforme vayamos mejorando su medida y pararemos cuando dejemos de mejorarla [2], y así procederemos con el resto de variables. El pseudocódigo de este algoritmo, que denominaremos K2SN (K2 for Sorting Nodes) se puede observar en la figura 2.7. En esta figura, las llamadas $f = K2(x_i, VISITADOS)$ y $\pi(x_i) = K2(x_i, VISITADOS)$, quieren decir que calculamos el conjunto de padres del nodo x_i de entre los nodos en el conjunto *VISITADOS* como lo hace el algoritmo K2 y f es la medida para esta familia $\{x_i \cup \pi(x_i)\}$. Esta heurística, si nos fijamos bien en ella, no es más que una búsqueda en profundidad acotada a un solo nivel. Esto quiere decir que solo el máximo nodo en cada nivel de nuestro árbol de búsqueda se expande. Una representación gráfica de un ejemplo de cómo el algoritmo K2SN progresa se puede observar en la figura 2.8.

Otra posible forma de buscar los padres en cada etapa del algoritmo podría ser la siguiente: Suponer un número de padres máximo p para cada una de las variables, y entonces quedarnos con la mejor familia para una variable comprobando todas las posibles combinaciones de $q \leq p$ padres de entre los nodos predecesores, al igual que se propone en [96, 60], aunque esta opción es claramente más costosa que la anterior. Otra posible forma de buscar el conjunto de padres sería utilizar una técnica de ramificación y poda al igual que se propone en los trabajos [132, 131]. Esta opción es una clara extensión de este algoritmo en el futuro.

Cómo hemos visto en el anterior ejemplo 3 de este capítulo, esta simple búsqueda acotada en profundidad a un solo nivel nos ofrece una muy buena secuencia de ordenación entre las variables de la red ALARM para los 3000 primeros casos. También hemos observado en el ejemplo de la figura 2.8 que es posible reutilizar bastantes cálculos de los efectuados en los niveles superiores del árbol. Es por ello que se podría realizar una búsqueda acotada en general en l niveles de profundidad para intentar mejorar nuestra solución inicial. Esta opción tiene el claro inconveniente de que a poco que aumentemos la cota de profundidad en el árbol de búsqueda, el proceso de búsqueda se convertirá en impracticable por la cantidad de ramas que hemos de expandir en cada nivel. Sin embargo, este inconveniente se podría aliviar utilizando una técnica de poda en la expansión del árbol de búsqueda.

Hemos estudiado en el capítulo anterior varias métricas o medidas $f(G:D)$, como por ejemplo la BIC, K2, BDe, que cumplen la propiedad de ser descomponibles. Estas tres métricas en particular son bastante utilizadas en la literatura especializada en el tema de aprendizaje automático de redes de creencia. Además estas tres medidas poseen otra propiedad particularmente interesante para el problema que hemos expuesto en el párrafo anterior. Se podría demostrar que para toda familia $\{x_i \cup \pi_G(x_i)\}$, el valor de la medida $f(x_i, \pi_G(x_i))$ es siempre un valor negativo

[2] Es bien conocido que el algoritmo K2 tiene un comportamiento bueno siempre que conozcamos un orden a priori entre las variables.

Algoritmo K2SN

1. $VISITADOS = \emptyset$ *y* $PORVISITAR = VARIABLES$.
2. $max = -\infty$
3. *Para* $i = 1$ *hasta* $|PORVISITAR|$ *hacer*

   Si $f(x_i, \emptyset) > max$ *entonces*
   $max = f(x_i, \emptyset)$ *y* $x = x_i$
4. $VISITADOS = VISITADOS \cup \{x\}$
   $PORVISITAR = PORVISITAR \setminus \{x\}$
   $\pi(x) = \emptyset$
5. *Mientras* $|PORVISITAR| > 0$ *hacer*

   (a) $max = -\infty$

   (b) *Para* $i = 1$ *hasta* $|PORVISITAR|$ *hacer*

   i. *Sea* $x_i \in PORVISITAR$

   ii. $f = K2(x_i, VISITADOS)$

   iii. $\pi(x_i) = K2(x_i, VISITADOS)$

   iv. *Si* $f(x_i, \pi(x_i)) > max$ *entonces*
   $max = f(x_i, \pi(x_i))$ *y* $x = x_i$ *y* $\pi(x) = \pi(x_i)$

   (c) $VISITADOS = VISITADOS \cup \{x\}$
   $PORVISITAR = PORVISITAR \setminus \{x\}$

Figura 2.7: Algoritmo K2SN

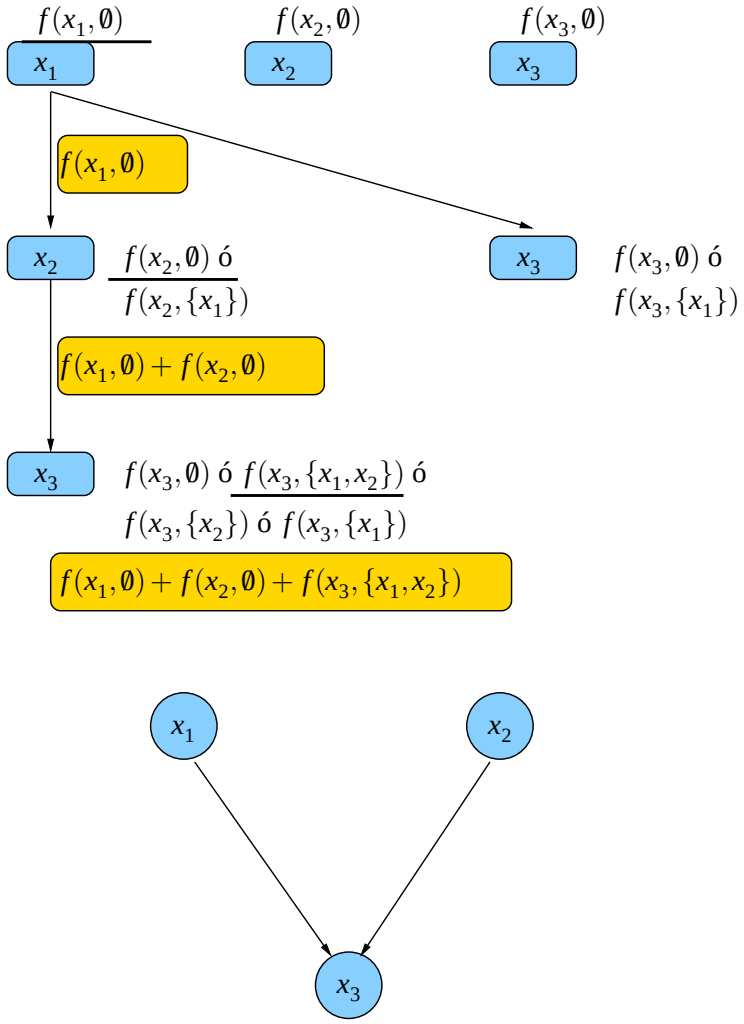


Figura 2.8: Ejemplo del árbol de búsqueda con tres variables para el algoritmo K2SN. El nodo máximo es siempre el que se expande (el más a la izquierda, subrayada la medida máxima en cada etapa). En este ejemplo se han calculado todos los estadísticos necesarios; como podemos observar no siempre se han de calcular todos los estadísticos en cada nivel, es posible reutilizar cálculos previos. En la parte inferior de la figura podemos observar la red resultado y recuadrado la medida parcial en cada etapa del algoritmo; la medida final será la suma de todas las etapas anteriores.

(véanse las ecuaciones 1.11, 1.10, 1.14).

Si tenemos una grafo $G = (V,E)$, los datos D y tenemos una medida $f(G:D)$ (BIC, K2, BDe), entonces se podría demostrar que: $f(G':D) \geq f(G:D)$ para todo G' subgrafo de G, siendo $G' = (V',E')$ y $V' = V \setminus \{x_i\}$ y $E' = E \setminus \{x_j \rightarrow x_i \ \ \forall x_j \in \pi_G(x_i)\}$ para todo $x_i \in V$. Esto quiere decir que al ir incorporando nodo a nodo a la medida f, ésta en cada sumando siempre irá disminuyendo su valor. Por ejemplo en la figura 2.8 tendremos que se cumple:

$$f(x_1,\emptyset) \geq f(x_1,\emptyset) + f(x_2,\emptyset) \geq f(x_1,\emptyset) + f(x_2,\emptyset) + f(x_3,\{x_1,x_2\})$$

Teniendo en cuenta lo anteriormente expuesto, si en un nodo del árbol de expansión vemos que la medida que obtenemos en la secuencia parcial que llevamos es peor (menor) que la que tenemos como máximo, podemos dejar tranquilamente de expandir este nodo porque al expandirlo sólo empeoraremos aún más la solución final encontrada. Esta solución agilizará de forma considerable la búsqueda y la agilizaremos aún más si ordenamos nuestros nodos a expandir en cada nivel del árbol conforme su medida de la función crezca en valor, es decir, los ordenamos de menor a mayor, esto hará que probablemente la primera (o de las primeras) solución que encontremos sea una buena medida de cota superior para podar nuestro árbol de búsqueda.

Aún con esta poda del árbol de búsqueda, el esfuerzo computacional que debemos realizar, incluso con valores pequeños de la cota de profundidad l, puede que sea excesivo. Por consiguiente sería conveniente proponer alguna función aproximada del valor esperado de la solución final al expandir un determinado nodo en un determinado nivel del árbol de búsqueda y utilizar ésta como una función para podar aquellas ramas de árbol que estimemos no van a mejorar la solución parcial que tengamos en ese instante, al estilo de la función de evaluación del algoritmo A* [72, 73, 110, 108]. Si observamos algún proceso de búsqueda, podemos ver que si nuestro máximo actual tiene un determinado valor f_{max}, y vamos a expandir un determinado nodo cuya medida de la solución parcial que ha buscado hasta ese instante es un valor muy cercano a f_{max}, y nos quedan todavía ciertos niveles que expandir para encontrar la solución final, podríamos estimar que no merece la pena seguir expandiendo esta rama del árbol de búsqueda porque en el resto de niveles seguramente va a empeorar la solución que tenemos como máximo en este instante.

La cuestión ahora es qué función utilizamos para estimar a partir de un nodo el valor de la métrica para una solución final a partir de este nodo y en función del valor máximo que tengamos en este instante de la búsqueda. Si en un momento dado tenemos como máximo f_{max} para una solución x y tenemos una cota l en profundidad, la media en cada nivel para llegar a f_{max} (teniendo en cuenta que f_{max} se obtiene como la suma de cada nodo perteneciente a la solución) será f_{max}/l, esto es, suponemos que en cada nivel se ha sumado exactamente la misma cantidad para

llegar a f_{max}. Si vamos a expandir un nodo en el árbol de búsqueda, con la solución parcial x' y con un valor f', si a este valor le sumamos el valor esperado por nivel que tenemos para el máximo actual f_{max}/l por los niveles que aún nos faltan por expandir, tendremos un valor esperado para la solución final que nos ofrecerá la búsqueda al expandir esta rama del árbol, esto es:

$$f' + [(f_{max}/l)(l - na)] \tag{2.8}$$

donde *na* es el nivel actual de la búsqueda antes de expandir un determinado nodo. Entonces si el valor de la función 2.8 es peor que la que llevamos f_{max}, entonces dejaremos de expandir el nodo correspondiente en el árbol. Teniendo en cuenta que esta función no deja de ser una aproximación del valor que se espera de la solución final a partir de un nodo en la búsqueda y estar en función del máximo que hemos obtenido hasta ese instante, el cual puede que esté lejos del óptimo, es arriesgado podar una rama del árbol de esta forma. Es por ello, que vamos a introducir un parámetro $\beta \in [0,1]$ que ajustará de alguna forma (o corregirá) este valor esperado en cada paso de la solución que llevemos en un determinado instante como máximo, esto es: $(f_{max}/l)^{\beta}$, de tal forma, que si $\beta = 0$ estamos en el caso primero, en donde podamos sólo si la solución parcial es peor, y si $\beta = 1$ estamos en el anterior caso, por esta razón este parámetro β puede interpretarse como un compromiso entre eficiencia y eficacia en la búsqueda. La función quedaría como sigue:

$$f' + [(f_{max}/l)^{\beta} - (1 - \beta)](l - na) \tag{2.9}$$

2.5 Experimentación y Conclusiones

En esta sección vamos a presentar los resultados empíricos obtenidos en el aprendizaje de una serie de bases de datos en diferentes dominios mediante los diversos algoritmos presentados en las anteriores secciones del capítulo. Resumidos en formas de tablas, podremos analizarlos y extraer de ellos una serie de conclusiones acerca del comportamiento de los diferentes algoritmos utilizados.

2.5.1 Descripción de los experimentos

Con el objetivo de realizar un estudio comparativo sobre los algoritmos estudiados a lo largo del capítulo se han empleado los siguientes parámetros para los diferentes algoritmos.

En todos los casos hemos utilizado la métrica K2(log) [31] para el proceso de búsqueda de la redes de creencia. Para los experimentos realizados se han utilizado 5 bases de datos diferentes correspondientes a dos dominios distintos, el dominio de ALARM [13](figura 2.10) y el dominio de INSURANCE [17](figura 2.9). Para

el primero de ellos hemos elegido los primeros 3000 casos y los primeros 10000 de la base de datos ALARM; esta base de datos de casos, de 20000 casos, se obtuvo mediante muestreo lógico probabilístico [78] para el estudio realizado en el trabajo [13] y se encuentra disponible en [133]. Esta base de datos y la red ALARM correspondiente, se han convertido en un estándar en la evaluación de algoritmos de aprendizaje; éste es un dominio médico que se compone de 37 variables y 46 arcos. En el segundo caso, también se ha utilizado el muestreo lógico probabilístico para generar 3 bases de datos de casos de 10000 casos cada una con las que realizaremos los experimentos en esta sección. El domino INSURANCE consta de 27 variables y 52 arcos, y trata de los riesgos en los seguros de automóviles.

El propósito de cada uno de los experimentos es recuperar la red a partir de la cual se generaron las bases de datos empleadas en los procesos de aprendizaje. Por otra parte también estamos interesados en mostrar la eficiencia de los diferentes algoritmos empleados. Para poder comparar estos algoritmos se calcularon los siguientes parámetros mostrados en las diferentes tablas.

- Las iteraciones que tuvieron que dar hasta encontrar el óptimo alcanzado por cada uno de los algoritmos. En los tipos de algoritmos estudiados en este capítulo, este parámetro corresponde al número de ascensiones de colinas que tuvieron que realizar durante el proceso de aprendizaje, tanto en el espacio de grafos dirigidos acíclicos como en el espacio de secuencias de ordenación.

- Una vez que alguno de estos algoritmos nos ofrece un resultado, lo hemos comparado de forma estructural con las redes originales utilizadas para generar los datos. Esta comparación nos va a ofrecer tres parámetros. El número de arcos añadidos en la red aprendida que no aparecen en la red original, el número de arcos borrados en la red aprendida que si aparecen en la red original y por último el número de arcos invertidos no siendo isomorfos en ambas redes.

- Por supuesto la medida utilizada en el aprendizaje K2(log), tanto en la media de las ejecuciones realizadas, cuando tenga sentido, como la mejor red encontrada por el algoritmo correspondiente.

- El parámetro denominado KL en nuestras tablas proviene de la entropía cruzada de Kullback, que mide la distancia entre la distribución de probabilidad, P_G, asociada a una red de creencia y la distribución de probabilidad, P, asociada a los casos de la base de datos. Aunque esta medida es en principio difícil de calcular, se puede aprovechar el hecho de que la distribución especificada mediante la red de creencia tiene una forma especial. Si P_G es la distribución conjunta asociada a la red G, que tiene el siguiente conjunto de nodos $X = \{x_1, \ldots, x_n\}$, la entropía cruzada se puede descomponer en los siguientes términos [94], [36]:

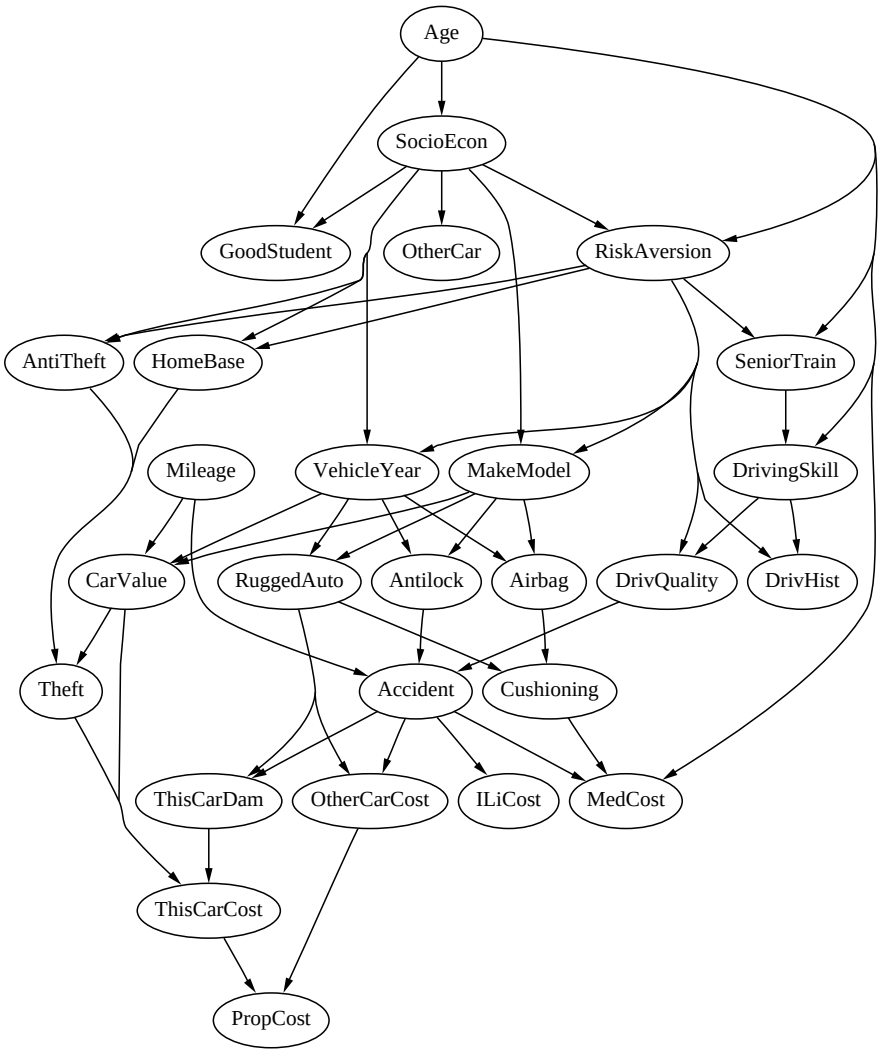


Figura 2.9: Red de creencia INSURANCE.

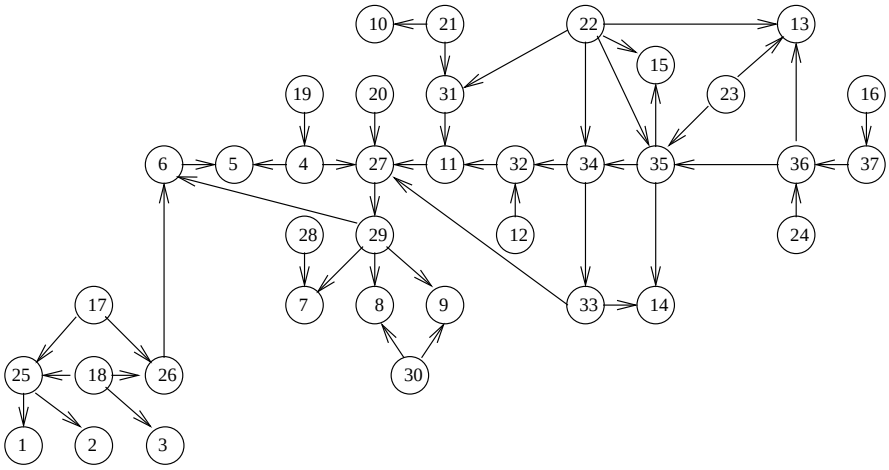


Figura 2.10: Red de creencia ALARM.

$$cross-entropy(P,P_G) = -H_P(X) + \sum_{i=1}^{n} H_P(x_i) - \sum_{i=1,\pi(x_i)\neq\emptyset}^{n} Dep(x_i,\pi(x_i)|\emptyset) \tag{2.10}$$

donde H_P denota la entropía de Shannon sobre la distribución P y Dep es igual a:

$$Dep(X,Y|\emptyset) = \sum_{\mathbf{x}_i,\mathbf{y}_j} P(\mathbf{x}_i,\mathbf{y}_j)\log\frac{P(\mathbf{x}_i,\mathbf{y}_j)}{P(\mathbf{x}_i)P(\mathbf{y}_j)} \tag{2.11}$$

Podemos observar que los dos primeros términos de la expresión 2.10 no dependen del grafo G, son constantes independientes de la topología de la red, así sólo nos interesa el último término, precisamente el parámetro que utilizamos en nuestras tablas como KL. La entropía cruzada será menor cuanto mayor sea el último término. De ahí que la interpretación de este parámetro sea, una red será mejor cuanto mayor sea este valor KL.

- En cuanto a la complejidad del algoritmo hemos obtenido los siguientes parámetros. En primer lugar el número de individuos (Ind) que se han visitado durante el proceso de búsqueda (en cada caso de acuerdo al espacio de búsqueda empleado en el proceso). En segundo lugar, mostramos el número total de estadísticos calculados (TEst) a partir de los datos empleados en el proceso de aprendizaje, esto es, N_{ijk}, la frecuencia para una variable x_i que toma su j-ésimo valor y la k-ésima configuración de su conjunto de padres en la

red correspondiente. Este número de estadísticos no tiene por qué ser el número de estadísticos realmente calculados sobre los datos (nótese que este proceso suele ser el más costoso en el tipo de algoritmo estudiado a lo largo del capítulo) ya que si utilizamos técnicas de hashing este número puede ser sustancialmente reducido; por consiguiente también mostramos en las tablas en número de estadísticos diferentes que aparecieron durante el proceso de búsqueda (EstEv). Por último, la complejidad del calculo de estadísticos puede ser exponencial con respecto al número de variables que intervienen en ellos, es por esto que mostramos el número medio de variables que intervienen en los estadísticos calculados durante el proceso de aprendizaje (Nvars). En términos de comparación, una estimación del tiempo empleado por los diferentes algoritmos puede ser: EstEv $\times 2^{\text{Nvars}}$.

Todos los algoritmos empleados en los experimentos han utilizado tres formas de inicialización de la búsqueda. Inicialización con la red vacía (EMP) de enlaces, y en el espacio de órdenes además la secuencia de ordenación encontrada en la base de datos utilizada; Inicialización con el resultado de la ejecución del algoritmo K2SN descrito en la sección 2.4.3 en su primera y más básica versión (figura 2.7); Inicialización con el resultado de la ejecución del algoritmo PC [127], cuando trabajemos con el espacio de órdenes se extraerá de este resultado un orden topológico como secuencia de inicialización[3].

En los algoritmos deterministas empleados en los experimentos, éstos se han ejecutado una sola vez, sin embargo en los estocásticos se ha ejecutado 10 veces cada uno, para cada inicialización y para cada base de datos de casos, obteniéndose en cada caso la media μ y la desviación estándar σ, así como el mejor resultado de estas ejecuciones.

2.5.2 Análisis de los experimentos

Los resultados de todos los experimentos se pueden observar en las tablas que aparecen a lo largo de esta subsección. Compararemos los resultados de cada algoritmo con el resto de los algoritmos (comparables) y para cada una de las bases de datos utilizadas. Como punto de referencia para todos los casos, se puede utilizar la medida K2 para las bases de datos utilizadas para la estructura verdadera de las redes utilizadas en los experimentos, que son $-14412,69$ para la red ALARM para los primeros 3000 casos, $-47086,57$ para la red ALARM para los primeros 10000 casos y $-58120,95$ para la red INSURANCE (promedio de las tres bases de datos generadas con 10000 casos cada una).

[3]Hemos de notar que a partir de un GDA puede obtenerse más de un orden topológico válido. Nosotros sólo hemos considerado uno de ellos para hacer determinista la inicialización.

Análisis de los algoritmos greedy, ascensión de colinas, HCST y HCSN. En las tablas 2.2 y 2.5 se muestran los resultados para los algoritmos HCST y HCSN (con 2 radios de búsqueda diferentes) para el dominio ALARM para los 3000 primeros casos y en las tablas 2.3 y 2.6 se muestran los resultados, para este mismo dominio, para los 10000 primeros casos de la base de datos utilizada. En ambos casos en el espacio de búsqueda de secuencias de ordenación se ha fijado el radio a dos valores distintos, $r = 7$ y $r = 36$ (aproximadamente $1/5$ del total de variables y el total de variables respectivamente).

En las tablas 2.4 y 2.7 se muestran los resultados obtenidos (en promedio) en el dominio INSURANCE, en este caso sólo se ha ejecutado el algoritmo HCSN con un radio $r = 13$ (aproximadamente la mitad de las variables del dominio).

En cuanto a la base de datos ALARM de 3000 casos, cabe destacar lo siguiente: En primer lugar, si exceptuamos la inicialización EMP en el algoritmo HCSN para un radio 7 (cuyo resultado es especialmente malo debido a que el orden en la base de datos ALARM es también especialmente malo y este radio no es capaz de ofrecer una buena solución) y también exceptuamos la inicialización PC en el algoritmo HCST, en donde su comportamiento es excelente como inicialización para este caso, podemos observar para el resto de casos que el algoritmo HCSN tiene un comportamiento más homogéneo; quizás cabe resaltar el comportamiento de la inicialización K2SN en el algoritmo HCSN ya que tiene un comportamiento excelente como inicialización en este dominio. El comportamiento de la inicialización PC con el algoritmo HCSN es un poco peor, pero hemos de tener en cuenta que a la estructura resultante de la ejecución del algoritmo PC podremos extraerle más de un orden ancestral, nosotros hemos probado sólo con uno de ellos. En cuanto a la eficiencia del algoritmo HCSN destacaremos que si aproximamos el tiempo de ejecución por la expresión $\text{EstEv} \times 2^{\text{NVars}}$, por ejemplo para la inicialización K2SN[4] con radio $r = 7$ el algoritmo es $2,67$ veces más lento que el HCST, y con radio $r = 36$ es $5,16$ veces más lento. Esto quiere decir, como argumentábamos anteriormente, que evaluar un cambio en la estructura será mucho más eficiente que evaluar un intercambio en una secuencia de ordenación y que éstas a su vez dependerán del radio que fijemos en la ejecución del algoritmo.

En cuanto a la base de datos para ALARM de 10000 casos, el comportamiento del algoritmo HCSN es similar al de 3000 casos, cabe destacar que en este caso incluso para la inicialización PC el algoritmo HCSN ofrece un mejor resultado que el algoritmo HCST.

En cuanto al dominio INSURANCE, en primer lugar destacaremos que la inicialización K2SN en este dominio tiene un comportamiento especialmente malo,

[4]Hemos elegido esta inicialización para esta comparación por ser una inicialización informada y no tan costosa como puede ser la inicialización con el algoritmo PC, además en este último caso habría que tener en cuenta los estadísticos calculados para realizar los tests de independencia condicional durante su ejecución (en las tablas aparece como TIC).

HCST	Empty	K2SN	PC
K2	-14425,62	-14520,21	-14403,77
KL	9,220	9,229	9,231
A	6	8	1
B	4	2	1
I	3	2	0
It	1	1	1
Ind ($\times$1000)	76	11	32
EstEv	3375	4970	1880 + TIC
TEst	154637	47041	39704 + TIC
NVars	2,99	3,53	3,29

Tabla 2.2: Resultados para una ascensión de colinas en el espacio de estructuras HCST. ALARM 3000 casos.

hemos de fijarnos en el resultado obtenido por esta inicialización con el algoritmo HCST (tabla 2.4) y la diferencia estructural obtenida. En este dominio el comportamiento del algoritmo HCSN siempre ofrece un mejor resultado que el algoritmo HCST, independientemente de la inicialización.

También hemos de fijarnos que el número de individuos evaluados en el espacio de secuencias de ordenes es significativamente menor que en el espacio de grafos dirigidos acíclicos, esto nos da una idea de que el espacio de secuencias de ordenación es mucho más "suave" u homogéneo que el de las estructuras.

Por último indicar que estos experimentos los hemos realizado con la tercera de las opciones para la evaluación de la secuencias de órdenes[5], se han realizado pruebas con las otras dos opciones y los resultados son similares desde el punto de vista de la calidad y también de la eficiencia, teniendo en cuenta sólo los estadísticos evaluados, ya que en nuestra implementación hemos utilizado técnicas de hashing.

Análisis de los resultados del algoritmo IMAPR. En este caso hemos ejecutado este algoritmo 15 iteraciones en las pruebas realizadas y los resultados se pueden observar en las tablas 2.8, 2.9 y 2.10.

En primer lugar cabe resaltar que todos los algoritmos IMAPR mejoran notablemente la red encontrada por una ascensión de colinas clásica en el espacio de estructuras, en cuanto a la métrica utilizada en la búsqueda, también respecto a la medida KL y número de arcos erróneos. La excepción se da con la inicialización PC en el dominio ALARM, en donde hemos podido comprobar que esta inicializa-

[5]Donde inicializamos el conjunto de padres para cada uno de los nodos al conjunto vacío y los operadores para encontrar el nuevo conjunto de padres son la inserción y el borrado de padres.

HCST	Empty	K2SN	PC
K2	-47267,11	-47081,66	-47133,41
KL	9,266	9,276	9,271
A	11	3	3
B	4	1	1
I	6	0	2
It	1	1	1
Ind ($\times$1000)	72	16	18
EstEv	3280	5384	1804 + TIC
TEst	147008	55614	36539 + TIC
NVars	2,99	3,65	3,21

Tabla 2.3: Resultados para una ascensión de colinas en el espacio de estructuras HCST. ALARM 10000 casos.

HCST	Empty	K2SN	PC
K2	-57998,10	-58895,94	-57869,01
KL	8,423	8,320	8,449
A	10,33	30,67	1,33
B	11,67	14,67	8,67
I	7,67	15,67	1,33
It	1	1	1
Ind	3,74E+04	1,13E+04	1,17E+04
EstEv	2050,33	3801,33	1094,00 + TIC
TEst	7,66E+04	3,55E+04	2,41E+04+ TIC
NVars	3,09	3,97	3,37

Tabla 2.4: Resultados para una ascensión de colinas en el espacio de estructuras HCST. INSURANCE 10000 casos. Promedio 3 bases de datos.

HCSN ($r = 7$)	Empty	K2SN	PC
K2	-14860,49	-14408,67	-14412,28
KL	9,136	9,222	9,220
A	30	1	1
B	6	2	3
I	11	0	1
It	1	1	1
Ind ($\times$1000)	5,2	1,8	2,3
EstEv	14718	10152	17072 + TIC
TEst	1,97E6	5,61E5	7,16E5 + TIC
NVars	4,38	3,92	4,34
HCSN ($r = 36$)	Empty	K2SN	PC
K2	-14409,46	-14408,54	-14418,24
KL	9,223	9,223	9,226
A	2	2	5
B	2	2	2
I	0	0	4
It	1	1	1
Ind ($\times$1000)	12	4	5,3
EstEv	37563	17427	24593 + TIC
TEst	9,42E6	2,84E6	3,96E6 + TIC
NVars	4,54	4,09	4,36

Tabla 2.5: Resultados para una ascensión de colinas en el espacio de órdenes HCSN. ALARM 3000 casos.

HCSN ($r = 7$)	Empty	K2SN	PC
K2	-47513,60	-47079,60	-47117,67
KL	9,269	9,276	9,264
A	23	4	2
B	3	1	3
I	11	0	0
It	1	1	1
Ind ($\times$1000)	7,3	1,3	3,1
EstEv	18333	11384	17266 + TIC
TEst	2,88E6	4,27E5	9,73E5 + TIC
NVars	4,67	4,15	4,36
HCSN ($r = 36$)	Empty	K2SN	PC
K2	-47080,22	-47076,20	-47109,89
KL	9,274	9,274	9,266
A	1	1	1
B	1	1	2
I	0	0	0
It	1	1	1
Ind ($\times$1000)	17	3,3	3,4
EstEv	51609	20160	26263 + TIC
TEst	1,39E7	2,52E6	2,96E6 + TIC
NVars	4,83	4,35	4,39

Tabla 2.6: Resultados para una ascensión de colinas en el espacio de órdenes HCSN. ALARM 10000 casos.

HCSN $(r = 13)$	Empty	K2SN	PC
K2	-57877,99	-58010,39	-57842,38
KL	8,443	8,424	8,442
A	4,00	12,00	4,00
B	9,67	13,00	9,67
I	2,33	10,67	1,33
It	1	1	1
Ind	3,00E+03	4,10E+03	2,55E+03
EstEv	8899,67	20274,67	8851,33
TEst	1,04E+06	1,67E+06	8,96E+05
NVars.	4,02	4,60	4,07

Tabla 2.7: Resultados para una ascensión de colinas en el espacio de órdenes HCSN. INSURANCE 10000 casos. Promedio 3 bases de datos

ción más un HCST es ya un resultado excelente. Hemos de notar que la medida K2 utilizada, es en realidad $\log(\text{K2})$, con lo que las diferencias serán muchos mayores en la medida K2 en escala no logarítmica.

En todos los casos, atendiendo al mejor individuo encontrado en las 10 ejecuciones de nuestro algoritmo[6], la red resultante puede calificarse de muy buena aproximación.

Si bien esto anterior es cierto, no es menos cierto que comparado con el algoritmo VNSST (tablas 2.11, 2.13 y 2.15), el cual utiliza también un esquema de múltiples reinicios y búsquedas en el espacio de estructuras, pero de una forma estocástica, los resultados de IMAPR son más pobres que los de los algoritmos VNSST, además las diferencias de tiempos de ejecución entre ambos algoritmos no son significativas (teniendo en cuenta los tests de independencia condicional que se realizan en cada iteración en el algoritmo IMAPR). Estos resultados, en principio, nos sorprenden un poco ya que en el algoritmo IMAPR se realizan reinicios muchos más guiados por los datos que los reinicios efectuados en los algoritmos VNSST. El problema, pensamos, radica en que si en el esquema de reinicios elegido en el IMAPR elegimos una mala secuencia de ordenación, éste algoritmo tenderá a encontrar una red I-map de los datos y éstos I-map, normalmente con un número mayor de enlaces, serán puntos de difícil salida para la ascensión de colinas correspondiente. Este problema se resuelve satisfactoriamente en el nuevo esquema de vecindad planteado en la sección 2.6.

[6]Se podrían considerar las 10 ejecuciones como una sola ejecución del método reiniciando éste cada 15 iteraciones con el punto inicial elegido.

IMAPR	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-14416,37	7,39	-14405,12	-14405,67	2,98	-14401,29	-14402,06	1,18	-14401,29
KL	9,225	0,003	9,231	9,233	0,003	9,231	9,231	0,000	9,231
A	5,00	2,16	3	5,20	1,75	2	1,90	0,74	2
B	2,70	1,16	1	1,60	0,52	1	1,00	0,00	1
I	2,60	0,52	2	0,70	0,67	0	0,00	0,00	0
It	10,60	4,88		11,60	4,22		5,10	4,41	
Ind	5,24E+05	2,15E+04		4,53E+05	2,02E+04		5,17E+05	2,91E+04	
EstEv	12485,70	452,62		14071,90	393,69		12633,10	940,00	
Test	1,07E+06	4,35E+04		9,47E+05	4,11E+04		1,05E+06	5,93E+04	
Nvars	4,24	0,09		4,48	0,07		4,62	0,06	

Tabla 2.8: Resultados IMAPR ALARM 3000 casos 15 iteraciones.

IMAPR	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-47246,15	48,73	-47111,03	-47080,30	1,75	-47078,27	-47133,41	0,00	-47133,41
KL	9,266	0,002	9,268	9,275	0,001	9,274	9,271	0,000	9,271
A	9,70	2,58	3	1,90	1,45	0	3,30	0,48	4
B	3,60	0,70	2	1,00	0,00	1	1,00	0,00	1
I	4,90	1,52	2	0,00	0,00	0	2,00	0,00	2
It	5,10	5,24		3,60	4,40		3,50	4,40	
Ind	6,07E+05	1,51E+04		5,18E+05	1,16E+04		5,32E+05	2,60E+04	
EstEv	14570,60	391,98		16152,80	477,96		13046,50	563,63	
Test	1,24E+06	3,07E+04		1,08E+06	2,36E+04		1,08E+06	5,30E+04	
Nvars	4,46	0,04		4,54	0,04		4,69	0,09	

Tabla 2.9: Resultados IMAPR ALARM 10000 casos 15 iteraciones.

IMAPR	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-57912,43	40,31	-57842,76	-58123,73	66,71	-57941,45	-57831,29	14,43	-57815,88
KL	8,439	0,006	8,465	8,416	0,014	8,385	8,458	0,002	8,413
A	9,73	0,88	3	16,17	2,63	12	1,60	0,59	1
B	11,20	0,35	8	11,97	0,68	12	7,37	0,49	7
I	8,93	1,79	3	16,50	2,03	10	2,07	0,42	2
It	8,80	4,04		12,13	2,28		3,90	2,08	
Ind	1,93E+05	1,04E+04		1,50E+05	1,10E+04		1,65E+05	9,77E+03	
EstEv	5940,00	291,01		7754,50	316,27		5212,97	288,13	
Test	3,97E+05	2,14E+04		3,23E+05	2,26E+04		3,39E+05	2,01E+04	
Nvars	4,41	0,07		4,64	0,06		4,59	0,06	

Tabla 2.10: Resultados IMAPR INSURANCE 10000 casos 15 iteraciones. Promedio 3 bases de datos

Análisis de los resultados de los algoritmos VNS. Ahora pasaremos a discutir los resultados obtenidos por los algoritmos VNS tanto en el espacio de grafos dirigidos acíclicos como en el espacio de secuencias de ordenación. Estos resultados se pueden observar en las tablas 2.11, 2.12, 2.13, 2.14 para el dominio ALARM y para el algoritmo VNSST con diferentes parámetros, y en la tabla 2.15 para el dominio INSURANCE. En todos estos algoritmos hemos fijado como k_{max} el número de variables en el dominio en cuestión.

Los resultados para los mismos dominios y para el algoritmo VNSSN se pueden observar en las tablas 2.16, 2.17 y 2.18.

Respecto del algoritmo VNSST para ALARM, los 3000 primeros casos y los parámetros $k_{min} = 1$ y $k_{step} = 1$, podemos decir que se obtienen unos magníficos resultados, prácticamente se encuentra siempre la mejor red encontrada por nosotros con una medida K2 de $-14401,29$. Este algoritmo (para la inicialización K2SN) es del orden de $13,2$ veces más lento que el algoritmo HCST, sin embargo hemos de tener en cuenta el número de ascensiones de colinas realizadas por el algoritmo VNSST, que es significativamente superior a este último número. En cuanto a los parámetros $k_{min} = 7$ y $k_{step} = 5$, podemos decir que se obtienen muy buenos resultados, que difieren muy poco del anterior caso, pero con un ahorro de tiempo de cálculo considerable (este algoritmo es del orden de $2,52$ veces más rápido que para los parámetros $k_{min} = 1$ y $k_{step} = 1$ y para la inicialización K2SN), en todos los casos y todas las inicializaciones se alcanza un número considerable de veces la mejor red mencionada anteriormente. Hemos de destacar, que en ambos casos se mejoran los resultados obtenidos sin utilizar la transformación específica diseñada en la sección 2.3.2[7], como puede comprobarse con los resultados obtenidos en el trabajo [48]. Además en este último trabajo se puede comprobar la significativa mayor eficiencia y eficacia del algoritmo VNSST en comparación a una simple ascensión de colinas con múltiples reinicios aleatorios.

En cuanto al algoritmo VNSST con los 10000 primeros casos de la base de datos ALARM, prácticamente se mantiene la misma tónica que en el caso anterior. En este caso se empeora en media un poquito el resultado para la inicialización EMP con los segundos parámetros probados ($k_{min} = 7$ y $k_{step} = 5$), pero esto es porque una sola ejecución de las 10 realizadas dio un resultado muy pobre (-47213).

Para el dominio INSURANCE y el algoritmo VNSST, hemos de decir que en todas las inicializaciones se ha encontrado la mejor red encontrada por nosotros con un medida K2 de $-57773,43$. En este dominio observamos de nuevo que la inicialización con el algoritmo PC ofrece los mejores resultados sobre todo en la diferencia estructural obtenida. Hemos de destacar, como comentábamos más arriba, que la inicialización con el algoritmo K2SN es especialmente mala, como se puede observar en la tabla 2.4, sin embargo el algoritmo VNSST es capaz de superar esta

[7] Donde con cierta probabilidad en la inversión de un arco borrábamos el conjunto de padres comunes.

VNSST	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-14401,30	0,04	-14401,29	-14401,30	0,04	-14401,29	-14401,29	0,00	-14401,29
KL	9,231	0,000	9,231	9,231	0,000	9,231	9,231	0,000	9,231
A	2,10	0,32	2	2,10	0,32	2	2,00	0,00	2
B	1,00	0,00	1	1,00	0,00	1	1,00	0,00	1
I	0,00	0,00	0	0,00	0,00	0	0,00	0,00	0
It	97,90	19,73		71,90	13,40		17,90	11,88	
Ind	4,50E+06	7,48E+05		4,05E+06	1,28E+06		2,90E+06	6,46E+05	
EstEv	37230,80	3561,79		36418,40	3787,11		29803,70	2494,36	
Test	5,32E+06	7,68E+05		4,73E+06	8,67E+05		3,62E+06	4,71E+05	
Nvars	4,41	0,03		4,38	0,04		4,41	0,04	

Tabla 2.11: VNSST ALARM 3000 casos $k_{min} = 1$ y $k_{step} = 1$.

situación y llegar a unos resultados bastante buenos. En este caso este algoritmo, de nuevo para la inicialización K2SN, es del orden de $8,91$ veces más lento que el algoritmo HCST.

Para el dominio ALARM, en las bases de datos probadas, el algoritmo VNSSN llega a unos resultados un poco peores que el correspondiente al VNSST, a pesar de ello vemos que prácticamente llegan a unos resultados buenos y que, como hemos argumentado a lo largo de este capítulo, el espacio de búsqueda en las secuencias de ordenación es más "suave", esto lo demuestra el número de individuos evaluados en ambos casos. Aunque también, como comentábamos anteriormente, evaluar una secuencia vecina es más costoso que evaluar un vecino de un GDA. Por último hemos de destacar el buen comportamiento que tiene en este espacio y para este dominio la inicialización K2SN. En este caso este algoritmo prácticamente tarda el mismo tiempo que el algoritmo VNSST con los parámetros $k_{min} = 1$ y $k_{step} = 1$ y por consiguiente será del orden del doble más lento que el VNSST con los parámetros $k_{min} = 7$ y $k_{step} = 5$.

Sin embargo, para el dominio INSURANCE el algoritmo VNSSN obtiene unos muy buenos resultados, mejorando los resultados ofrecidos por el algoritmo VNSST. Esto nos pone de manifiesto que, por una parte, dependiendo del dominio en cuestión un algoritmo ofrecerá mejores resultados que el otro, y por otra que quizás eligiendo un radio más adecuado en cada dominio, el algoritmo VNSSN ofrecerá mejores resultados. Conjeturamos que quizás con la mitad de las variables del dominio como inicialización del radio, el algoritmo VNSSN ofrecerá muy buenos resultados y más homogéneos que el algoritmo VNSST. En cuanto al tiempo de ejecución hemos de decir que el algoritmo VNSST es del orden de $2,39$ veces más rápido que el VNSSN probado.

VNSST	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-14403,45	4,22	-14401,29	-14402,82	2,39	-14401,29	-14402,53	1,30	-14401,29
KL	9,230	0,002	9,231	9,231	0,002	9,231	9,231	0,000	9,231
A	2,30	1,06	2	2,80	1,40	2	1,50	0,53	2
B	1,30	0,67	1	1,20	0,42	1	1,00	0,00	1
I	0,10	0,32	0	0,30	0,67	0	0,00	0,00	0
It	21,20	5,79		15,20	3,52		3,00	2,31	
Ind	1,67E+06	5,35E+05		1,09E+06	3,49E+05		6,24E+05	2,20E+05	
EstEv	17833,80	2643,55		16919,40	1712,36		10580,00	1764,23	
Test	1,62E+06	3,22E+05		1,19E+06	2,03E+05		7,69E+05	1,71E+05	
Nvars	4,17	0,10		4,15	0,08		4,11	0,08	

Tabla 2.12: VNSST ALARM 3000 casos $k_{min} = 7$ y $k_{step} = 5$.

VNSST	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-47076,20	0,00	-47076,20	-47076,20	0,00	-47076,20	-47076,20	0,00	-47076,20
KL	9,274	0,000	9,274	9,274	0,000	9,274	9,274	0,000	9,274
A	1,90	0,32	2	1,90	0,32	2	2,00	0,00	2
B	1,00	0,00	1	1,00	0,00	1	1,00	0,00	1
I	0,00	0,00	0	0,00	0,00	0	0,00	0,00	0
It	98,10	28,12		58,40	11,93		47,00	17,54	
Ind	4,34E+06	8,91E+05		4,35E+06	1,14E+06		3,77E+06	1,25E+06	
EstEv	35838,90	4098,73		38148,50	4492,77		32901,80	3704,42	
Test	5,17E+06	9,07E+05		4,98E+06	8,44E+05		4,37E+06	8,25E+05	
Nvars	4,42	0,06		4,42	0,05		4,42	0,05	

Tabla 2.13: VNSST ALARM 10000 casos $k_{min} = 1$ y $k_{step} = 1$.

VNSST	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-47091,03	43,06	-47076,20	-47076,88	1,43	-47076,20	-47076,62	0,87	-47076,20
KL	9,274	0,001	9,274	9,274	0,001	9,274	9,274	0,000	9,274
A	2,00	1,76	2	2,10	1,10	2	1,60	0,70	2
B	1,10	0,32	1	1,00	0,00	1	1,00	0,00	1
I	0,70	1,49	0	0,00	0,00	0	0,00	0,00	0
It	23,80	6,49		10,80	3,52		11,90	4,56	
Ind	1,86E+06	4,50E+05		1,13E+06	4,09E+05		1,18E+06	3,20E+05	
EstEv	19060,60	2581,05		17134,20	2573,82		14876,80	2237,52	
Test	1,79E+06	3,10E+05		1,19E+06	2,72E+05		1,24E+06	2,52E+05	
Nvars	4,24	0,08		4,19	0,07		4,23	0,09	

Tabla 2.14: VNSST ALARM 10000 casos $k_{min} = 7$ y $k_{step} = 5$.

VNSST	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-57848,90	3,90	-57773,43	-57843,93	24,45	-57773,43	-57814,30	27,43	-57773,43
KL	8,446	0,041	8,408	8,446	0,041	8,408	8,458	0,041	8,408
A	7,83	3,84	2	6,63	1,24	2	1,97	0,76	2
B	10,57	1,88	8	10,07	0,67	8	7,30	0,20	8
I	8,30	5,26	1	7,17	1,50	1	2,03	0,61	1
It	59,43	13,80		135,70	5,79		30,97	5,84	
Ind	1,35E+06	1,52E+05		2,02E+06	1,87E+05		9,45E+05	5,25E+04	
EstEv	13907,00	776,04		20215,97	880,86		11660,03	401,29	
Test	1,51E+06	1,50E+05		2,10E+06	1,31E+05		1,15E+06	5,89E+04	
Nvars	4,63	0,05		4,71	0,03		4,63	0,04	

Tabla 2.15: VNSST INSURANCE $k_{min} = 1$ y $k_{step} = 1$.

VNSSN	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-14408,83	0,64	-14407,92	-14408,43	0,27	-14407,92	-14408,50	0,23	-14407,92
KL	9,224	0,001	9,225	9,223	0,001	9,225	9,223	0,001	9,225
A	2,30	0,48	3	2,10	0,57	3	2,20	0,42	3
B	2,00	0,00	2	2,00	0,00	2	2,00	0,00	2
I	0,00	0,00	0	0,00	0,00	0	0,00	0,00	0
It	27,80	9,32		8,20	5,83		16,40	3,10	
Ind	1,43E+05	5,43E+04		9,08E+04	5,47E+04		8,33E+04	2,93E+04	
EstEv	46418,30	7044,06		37646,70	8752,57		42358,20	4132,27	
Test	3,24E+07	1,01E+07		2,30E+07	1,25E+07		1,90E+07	6,11E+06	
Nvars	4,60	0,07		4,47	0,09		4,59	0,06	

Tabla 2.16: VNSSN ALARM 3000 casos $r = 7$, $k_{min} = 1$, $k_{step} = 1$ y $k_{max} = 8$.

VNSSN	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-47087.39	12.26	-47077.18	-47076.49	0.90	-47076.20	-47110.08	0.41	-47109.89
KL	9.273	0,004	9.274	9.274	0.000	9.274	9.266	0.000	9.265
A	4.5	3.03	2	1.8	0.42	2	1.9	0.32	2
B	1.6	0.52	1	1.0	0.0	1	2.0	0.0	2
I	0.8	1.03	0	0.0	0.0	0	0.0	0.0	0
It	40.8	15.75		18.1	5.54		12.4	4.35	
Ind	327335	204091		179865	55924		99061	30844	
EstEv	68087	11680		49902	5664		54670	7832	
Test	5.98+E7	2.73+E7		4.16+E7	1.12+E7		2.43+E7	6.53+E6	
Nvars	4,96	0,06		4.79	0,08		4.85	0.1	

Tabla 2.17: VNSSN ALARM 10000 casos $r = 7$, $k_{min} = 1$, $k_{step} = 1$ y $k_{max} = 8$.

VNSSN	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-57826,40	24,26	-57779,29	-57826,01	32,88	-57773,43	-57808,34	30,85	-57774,30
KL	8,446	0,042	8,482	8,447	0,041	8,408	8,448	0,042	8,486
A	3,27	0,31	3	3,33	0,71	2	3,10	0,17	2
B	9,20	0,26	9	8,83	0,06	8	8,93	0,06	8
I	0,97	0,25	1	2,80	0,17	1	0,80	0,35	0
It	11,47	8,38		29,83	1,63		10,63	2,05	
Ind	9,26E+04	3,30E+04		1,64E+05	2,06E+04		9,52E+04	6,69E+03	
EstEv	30911,97	3781,39		49164,03	1091,23		32339,40	629,22	
Test	2,14E+07	5,59E+06		3,17E+07	2,65E+06		2,16E+07	1,84E+06	
Nvars	4,47	0,05		4,69	0,05		4,50	0,01	

Tabla 2.18: VNSSN INSURANCE $r = 13$, $k_{min} = 1$, $k_{step} = 1$ y $k_{max} = 9$.

2.6 Redefiniendo la vecindad para una ascensión de colinas en el espacio de grafos dirigidos acíclicos

Hemos observado, en el ejemplo 2 de este capítulo, que cuando nos equivocamos en la orientación de un determinado enlace, cosa que puede ocurrir a menudo sobre todo si no tenemos más elementos de juicio que solamente el enlace a introducir entre dos variables, entonces la mayoría de los algoritmos tienden a "cruzar" los padres de ambas variables del enlace en cuestión. Hemos observado que esto puede ocurrir también a menudo en el algoritmo IMAPR si se elige una secuencia de ordenación mala, de todas formas es bien conocido que este comportamiento se presenta en una gran cantidad de algoritmos de aprendizaje.

Como hemos indicado anteriormente, esta situación puede ocurrir con bastante asiduidad, imaginemos, centrándonos en el ejemplo 2, que la relación con más "fuerza" es la relación entre las variables x_2 y x_3, con lo cual un algoritmo de búsqueda probablemente introducirá un arco entre ellos en primer lugar. Si utilizamos una medida o métrica "score-equivalente" como puede ser la BIC, entonces el mismo valor tendrá introducir el enlace en una dirección u otra, ya que no tenemos elementos para discernir su orientación correcta. Si el algoritmo decide introducir la orientación contraria a la verdadera, éste tenderá a hacer comunes los padres de ambas variables con una probabilidad muy alta.

Por otra parte, hemos notado que este tipo de estructuras ya tienen un buen valor en la medida o métrica utilizada y que habitualmente los operadores clásicos de borrado, añadido o inversión de un solo arco no son capaces de escapar de estos máximos (óptimos) locales.

Se han desarrollado algoritmos en la literatura que de una forma u otra intentan "no moverse en el espacio de grafos dirigidos acíclicos sino moverse en el espacio de redes de creencia". Por ejemplo en los trabajos [28] y [128], se mueven en el espacio de las clases de equivalencia de redes de creencia, y definen operadores específicos

para moverse por este espacio, pero el problema que se plantea es un poco similar al que ocurre en el espacio de secuencias de ordenación, y es que no se aprovecha la eficiencia que proporciona habitualmente la propiedad de descomposición de las métricas y por consiguiente serán, en general, menos eficientes. También en [2] en su algoritmo BENEDICT-sin orden plantean movimientos en el espacio de clases de equivalencia. Los resultados ofrecidos por estos algoritmos mejoran, normalmente, los resultados ofrecidos por los operadores clásicos de grafos dirigidos acíclicos, pero a costa de perder eficiencia.

Recientemente en [91] se estudia la posibilidad de llegar a una situación de compromiso entre el espacio de clases de equivalencia y el espacio de GDAs. En este trabajo se hace uso intensivo de un operador denominado RCAR (Repeated Covered Arc Reversal), en donde repetidamente se van invirtiendo arcos con padres comunes (un cierto número de ellos prefijado de antemano), para fundamentalmente en una fase posterior relanzar una algoritmo HCST y obtener así un nuevo óptimo local.

Si analizamos bien el problema que planteamos en esta sección y los desarrollos llevados a cabo tanto en [28] y [91], es muy probable que en ambas aproximaciones tampoco se salga a menudo de estos óptimos locales, fundamentalmente porque en ambos casos se mantendrían todos los enlaces de las familias involucradas en estos cruces y posteriormente sólo se intenta mover en una vecindad local, esto es, intentar borrar/añadir un solo arco[8].

Es por ello que nosotros plantearemos un nuevo operador para el espacio de GDAs, en la misma filosofía que en [91], pero pensado específicamente para las redes de creencia. La vecindad clásica para el espacio de GDAs se ha estudiado en las secciones anteriores:

$$\mathcal{N}_A(G) = \{(V,E')/E' = E \cup \{x_j \rightarrow x_i\}, x_j \rightarrow x_i \notin E \text{ y } (V,E') \text{ es un GDA}\}$$

$$\mathcal{N}_B(G) = \{(V,E')/E' = E \setminus \{x_j \rightarrow x_i\}, x_j \rightarrow x_i \in E\}$$

$$\mathcal{N}_I(G) = \{(V,E')/E' = (E \cup \{x_j \rightarrow x_i\}) \setminus \{x_i \rightarrow x_j\}, x_i \rightarrow x_j \in E \text{ y } (V,E') \text{ es un GDA}\}$$

Así la vecindad para un grafo G será la unión de las operaciones anteriores, esto es, $\mathcal{N}(G) = \mathcal{N}_A(G) \cup \mathcal{N}_B(G) \cup \mathcal{N}_I(G)$. Nuestra propuesta es la modificación del operador de inversión, y por consiguiente la redefinición de la vecindad $\mathcal{N}_I(G)$, que ahora quedará definida como:

$$\mathcal{N}_I(G) = \left\{(V,E')/E' = \left(\left(E \cup \{x_j \rightarrow x_i\} \setminus \left(\{x_i \rightarrow x_j\} \cup P_{old}^{x_j,x_i}\right)\right) \cup P_{new}^{x_j,x_i}\right)\right\} \tag{2.12}$$

[8] Hemos visto en el ejemplo 2 que probablemente si nos movemos un sólo arco, entonces transitoriamente empeoremos el valor de la métrica utilizada.

tal que $x_i \rightarrow x_j \in E$ y (V,E') es un GDA y donde:

$$P_{old}^{x_j,x_i} = \begin{cases} \left\{\{x_{p_j} \rightarrow x_j\},\{x_{p_i} \rightarrow x_i\} \in E, x_{p_j} \in \pi_G(x_j), x_{p_i} \in \pi_G(x_i)\right\}, \\ \text{Si } \pi_G(x_i) \cap \pi_G(x_j) \neq \emptyset \\ \emptyset, \\ \text{Si } \pi_G(x_i) \cap \pi_G(x_j) = \emptyset \end{cases} \tag{2.13}$$

y

$$P_{new}^{x_j,x_i} = \begin{cases} \left\{\{x_{pn_j} \rightarrow x_j\},\{x_{pn_i} \rightarrow x_i\}, x_{pn_j} \in A, x_{pn_i} \in B \text{ y } A,B \in \mathscr{P}(\pi_G(x_i) \cup \pi_G(x_j))\right\}, \\ \text{Si } \pi_G(x_i) \cap \pi_G(x_j) \neq \emptyset \\ \emptyset, \\ \text{Si } \pi_G(x_i) \cap \pi_G(x_j) = \emptyset \end{cases} \tag{2.14}$$

donde $\mathscr{P}(\pi_G(x_i) \cup \pi_G(x_j))$, representa el conjunto de las partes de la unión del conjunto de padres de ambos nodos, y π_G representan el conjunto de padres de un nodo en el grafo G.

Esta nueva definición de vecindad quiere decir que cuando nos encontramos que en la inversión de un arco los nodos implicados tienen padres comunes, entonces, además de invertir dicho arco, probaremos como padres de ambos nodos todos los posibles subconjuntos a partir de la unión de los padres que tuviesen ambos nodos. El número de estos subconjuntos sería de $O(2^{2n})$, siendo n el número de variables en la unión de los padres, con lo que, en principio, este operador hace que el número de vecinos de cada grafo crezca de forma exponencial. Sin embargo, hemos de tener en cuenta que habitualmente el número de padres para una cierta variable es un cantidad baja, este hecho se revela en los experimentos que hemos realizado, y por tanto el coste computacional en la práctica no será excesivo. De todas formas, se podría plantear el limitar el número de subconjuntos tal que no superasen una cierta cantidad en su cardinalidad, como por otra parte es habitual en otros trabajos como [60, 96, 31].

2.6.1 Análisis experimental

En este caso vamos a utilizar los mismos dominios y bases de datos que en la sección 2.5, excepto el dominio INSURANCE. Para este caso hemos realizado una implementación de una ascensión de colinas clásica con esta nueva definición de vecindad, además hemos ejecutado el algoritmo IMAPR con esta nueva ascensión de colinas en las mismas condiciones que en la sección 2.5.

Los resultados de la ascensión de colinas clásica se pueden observar en las tablas 2.19 y 2.20. En primer lugar la mejora de los resultados, comparados con los de las

HCST	Empty	K2SN	PC
K2	-14414,55	-14405,48	-14403,77
KL	9,225	9,234	9,231
A	4	3	1
B	2	1	1
I	2	0	0
EstEv	3336	5123	1830 + TIC
TEst	155635	57689	33643 + TIC
NVars	2,93	3,50	3,14

Tabla 2.19: Resultados para una ascensión de colinas en el espacio de estructuras HCST redefiniendo la vecindad. ALARM 3000 casos.

tablas 2.2 y 2.3, en el dominio comprobado es bastante significativa en cuanto a la calidad del resultado. Por otra parte si nos fijamos en los estadísticos evaluados distintos en ambas ejecuciones y lo comparamos con los de la vecindad clásica de GDAs, resulta que el número de ellos es prácticamente similar, al igual que el número de variables medio que intervienen en ellos (esto hace que prácticamente el tiempo de ejecución de ambos sea similar).

En cuanto a los algoritmos IMAPR para la base de datos de ALARM 3000 casos, los resultados obtenidos mejoran de manera notable los obtenidos por la ascensión de colinas con la definición de vecindad clásica, y son prácticamente similares a los obtenidos por los algoritmos VNSST en las tablas 2.11 y 2.12. En cuanto a la eficiencia, centrándonos de nuevo en la inicialización K2SN, decir que el algoritmo IMAPR con la definición de vecindad clásica es del orden de $1,13$ veces más rápido que con la redefinición de vecindad realizada en esta sección.

Para la base de datos ALARM y los primeros 10000 primeros casos, hemos de destacar el comportamiento de IMAPR con esta redefinición de vecindad para la inicialización EMP, para el resto de inicializaciones hemos de notar que una simple ascensión de colinas con esta nueva definición de vecindad obtiene ya un resultado excelente, que el algoritmo IMAPR no es capaz de mejorar, por ser éste ya un óptimo muy difícil de superar. En cuanto a la eficiencia hemos de decir que se mantiene en el mismo orden $(1,3)$ que en el caso anterior.

HCST	Empty	K2SN	PC
K2	-47111,03	-47078,27	-47078,27
KL	9,268	9,274	9,274
A	3	0	0
B	2	1	1
I	2	0	0
EstEv	3327	5147	1973 + TIC
TEst	152742	44500	44558 + TIC
NVars	2,94	3,53	3,21

Tabla 2.20: Resultados para una ascensión de colinas en el espacio de estructuras HCST redefiniendo la vecindad. ALARM 10000 casos.

IMAPR	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-14402,41	1,20	-14401,29	-14402,72	1,27	-14401,29	-14402,54	1,29	-14401,29
KL	9,231	0,001	9,231	9,230	0,001	9,231	9,231	0,000	9,231
A	1,70	0,48	2	1,20	0,63	2	1,60	0,70	2
B	1,00	0,00	1	1,00	0,00	1	1,00	0,00	1
I	0,40	0,84	0	0,00	0,00	0	0,00	0,00	0
It	8,00	3,40		7,10	3,70		5,10	5,43	
EstEv	16489,80	1401,51		16927,10	726,28		14333,30	1179,68	
Test	2,46E+06	8,80E+05		1,98E+06	3,83E+05		2,15E+06	7,15E+05	
Nvars	4,45	0,12		4,39	0,06		4,52	0,10	

Tabla 2.21: Resultados IMAPR ALARM 3000 casos 15 iteraciones. Definición nueva vecindad.

IMAPR	Empty			K2SN			PC		
	μ	σ	Mejor	μ	σ	Mejor	μ	σ	Mejor
K2	-47078,51	0,78	-47078,26	-47078,27	0,00	-47078,26	-47078,06	0,65	-47076,20
KL	9,274	0,000	9,274	9,274	0,000	9,274	9,274	0,000	9,274
A	0,30	0,48	1	0,30	0,48	1	0,30	0,48	2
B	1,10	0,32	1	1,00	0,00	1	1,00	0,00	1
I	0,10	0,32	0	0,00	0,00	0	0,00	0,00	0
It	5,60	4,50		2,60	2,84		3,40	4,67	
EstEv	19175,70	1823,15		20465,70	1803,24		19387,10	3315,64	
Ntest	3,44E+06	9,21E+05		3,26E+06	1,07E+06		5,72E+06	3,89E+06	
Nvars	4,58	0,10		4,58	0,12		4,83	0,25	

Tabla 2.22: Resultados IMAPR ALARM 10000 casos 15 iteraciones. Definición nueva vecindad.

Capítulo 3

Métodos de Búsqueda Distribuidos para el Aprendizaje de Redes de Creencia

3.1 Introducción

Cada vez más tenemos a nuestra disposición computadores a bajo coste con capacidad de multiprocesamiento, por lo que es muy razonable tratar de aprovechar las ventajas que nos ofrecen este tipo de computadores para realizar nuestras tareas de forma más eficiente y más eficaz. Es por ello que en este capítulo nos dedicaremos a estudiar formas de distribuir tareas en los procesos de búsqueda que previamente hemos planteado, así como comprobar en nuestro problema en particular su eficiencia y sobre todo su eficacia. Estos entornos multiprocesador nos ofrecen la posibilidad de realizar de forma simultánea varias tareas aprovechando de manera óptima sus recursos. Fundamentalmente se puede pensar en dos formas de distribuir los problemas en varias tareas atendiendo al objetivo perseguido:

- Subdividir un problema en subproblemas más o menos independientes y resolverlos de forma simultánea en el tiempo, para fundamentalmente aumentar su eficiencia (reducir el tiempo de computación).

- Relacionado con el punto anterior, abordar un problema más o menos complejo realizando más trabajo, y aumentando así nuestras garantías de éxito

al poder realizar más tareas en un mismo periodo de tiempo, para aumentar fundamentalmente su eficacia.

La búsqueda distribuida en el aprendizaje de redes de creencia no ha recibido tanta atención en la literatura como la búsqueda secuencial. Se pueden encontrar algunos trabajos relacionados con el tema: Sangüesa y col. [123] desarrollan un algoritmo paralelo para el aprendizaje de redes posibilísticas. Xiang y col. [137] desarrollan algoritmos paralelos para el aprendizaje estructural de redes de Markov. Lam y col. [95] desarrollan un algoritmo distribuido para el aprendizaje estructural de redes Bayesianas, en donde fundamentalmente se distribuyen los cálculos (búsqueda del conjunto de padres de un determinado nodo) para aumentar la eficiencia en una técnica de ramificación y acotamiento, sin embargo este algoritmo necesita de un orden previo entre las variables. Por último Lozano y col. [103] diseñan algoritmos paralelos para fundamentalmente disminuir el coste de la búsqueda.

Nosotros, debido al problema que queremos resolver, estamos más interesados en la segunda opción, esto es, aumentar de alguna forma las garantías de encontrar una buena solución, explorando más regiones de nuestro espacio de búsqueda en un mismo periodo de tiempo. En primer lugar estudiaremos una versión distribuida del esquema de búsqueda local en entorno variable presentado en el capítulo anterior. A continuación estudiaremos cómo resolver nuestro problema mediante una técnica de reciente aparición, como es el modelo de agentes cooperativos, el modelo de las colonias de hormigas. Finalmente, propondremos un sistema distribuido híbrido entre las búsquedas VNS y las colonias de hormigas.

3.2 Búsqueda en Entorno Variable Distribuida

La principal motivación de realizar una versión distribuida de la búsqueda en entorno variable es aumentar la calidad de la solución encontrada sin incrementar el tiempo de computación. Una primera idea para distribuir la búsqueda en entorno variable directa y simple sería lanzar k búsquedas en paralelo independientes y quedarnos con la mejor de ellas al finalizar estas búsquedas. En la figura 3.1 podemos ver un esquema de funcionamiento de este tipo de algoritmo. Siendo la búsqueda en entorno variable una búsqueda estocástica, podremos esperar que el algoritmo ofreciera diferentes resultados durante distintas ejecuciones, esperando que este simple método mejore la calidad de la solución buscada.

Esta primera aproximación no deja de ser equivalente a realizar k iteraciones del algoritmo VNS simple, esto es, esta versión realizaría la misma búsqueda desde el punto de vista de la calidad de la solución aportada. En lugar de realizar k búsquedas totalmente independientes, podemos de alguna forma colaborar entre las k búsquedas iniciadas de forma paralela, cada cierto tiempo coordinar de alguna forma nuestra búsqueda con el objetivo principal de centrarnos en las zonas más

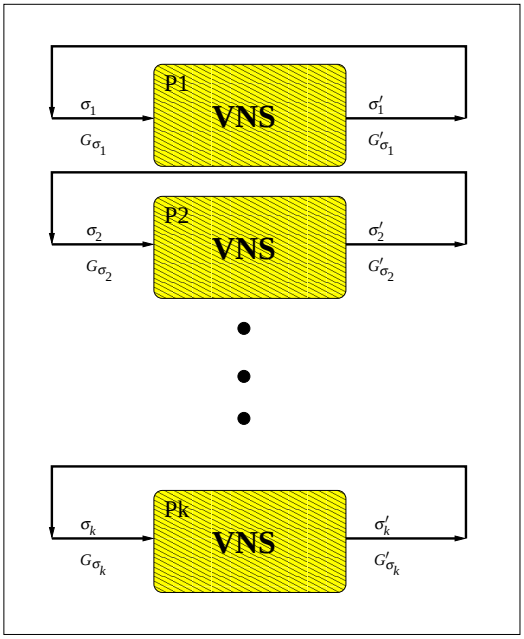


Figura 3.1: Esquema del Algoritmo VNS Distribuido - Búsquedas Independientes

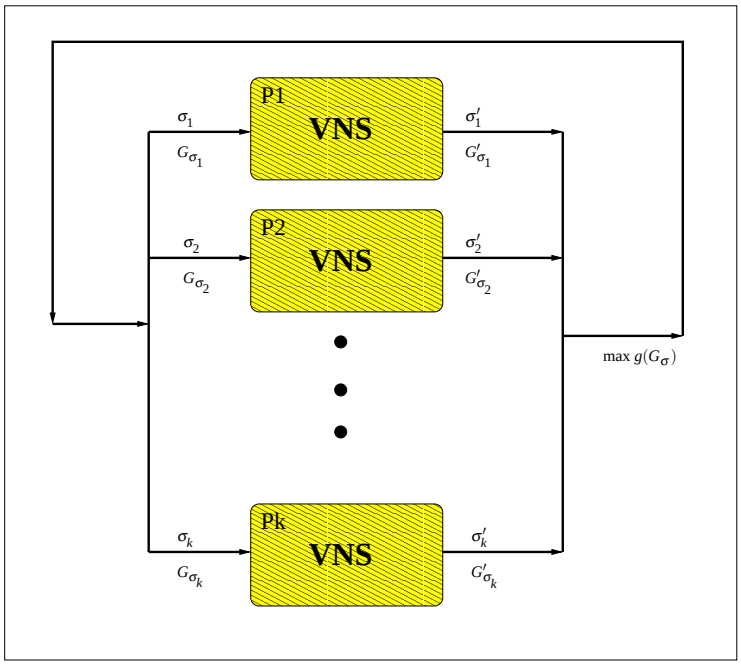


Figura 3.2: Esquema del Algoritmo VNS Distribuido - Búsquedas Coordinadas Globalmente

prometedoras exploradas hasta el momento. Esta coordinación entre búsquedas se puede realizar de distintas maneras y es un problema que de cierta forma queda abierto. Nosotros hemos elegido una forma simple de coordinar nuestra búsqueda y es la siguiente: Una vez que las *k* búsquedas, iniciadas desde algún punto elegido previamente, han terminado el bucle interno del método VNS, esto es, han alcanzado la vecindad máxima de búsqueda, entonces nos quedamos con la mejor solución aportada e iniciamos de nuevo *k* búsquedas en paralelo con esta solución como inicio de la búsqueda. De esta forma en cada iteración del VNS nos centraremos en la región del máximo local alcanzado hasta ese momento, con la esperanza de que el óptimo local no quede lejos del que hemos alcanzado. Esta estrategia de coordinación entre búsquedas no se aleja lo más mínimo de la filosofía inicial de la búsqueda en entorno variable. Esta aproximación puede ser vista como un modelo de coordinación global entre todas las soluciones y hemos de notar que es independiente del problema a tratar. Un esquema de este tipo de algoritmo lo podemos ver en la figura 3.2.

Por último notar que para nuestro problema de aprendizaje de redes de creencia, estas aproximaciones son directamente aplicables, tanto en el espacio de búsqueda de grafos dirigidos acíclicos como en el espacio de órdenes.

3.2.1 Esquema de inicialización de las búsquedas distribuidas

Un forma de inicializar las diferentes búsquedas es por supuesto de forma aleatoria, tanto en el espacio de órdenes como en el espacio de grafos dirigidos acíclicos. Una forma un poco menos aleatoria es elegir una secuencia de ordenación aleatoria entre los nodos y aprender mediante el algoritmo K2 la estructura de la red de creencia inicial para todos o parte de los individuos de la población.

La inicialización aleatoria tiene las ventajas de que al ser al azar: (a) podemos elegir un buen punto de inicio para la búsqueda y (b) explorar más regiones del espacio de búsqueda correspondiente, esto es, diversificaríamos los puntos de inicio en el espacio de búsqueda, pero tiene el claro inconveniente de que con frecuencia perderemos tiempo intentando optimizar malos puntos de inicio.

Otra forma de inicialización puede ser un poco más guiada que las anteriores, de tal forma que nos centremos desde el principio en regiones prometedoras del espacio de búsqueda. Por otra parte, la inicialización de los diferentes puntos de inicio debe ser eficiente ya que no sería deseable invertir demasiado tiempo en buscar estas zonas del espacio de búsqueda prometedoras. Una heurística de búsqueda en el espacio de grafos dirigidos acíclicos bastante eficiente es el Algoritmo B [23]: Este algoritmo realiza una búsqueda local a partir de grafo vacío de enlaces, en donde de forma incremental va introduciendo arcos entre nodos, eligiendo en cada etapa aquél que maximiza una medida, todo ello evitando ciclos dirigidos (ver capítulo 1). El resultado de este algoritmo sería una buena opción como punto de partida para nues-

tra búsqueda, ya que el resultado suele ser una aproximación bastante informada del óptimo que deseamos conseguir, además de ser un algoritmo suficientemente rápido como para plantearlo como inicialización. El problema que se nos plantea ahora es que este algoritmo es determinista y por tanto no sería conveniente inicializar todas las búsquedas con el mismo punto ya que no diversificaríamos suficientemente la búsqueda.

Para solucionar este inconveniente vamos a plantear una versión probabilística del algoritmo B, de tal forma que no se generen (normalmente) los mismos resultados como inicio de las búsquedas posteriores VNS y sigan conservando en cierta medida la propiedad de ser unos puntos de inicio prometedores. Esta versión probabilística, (figura 3.3), consistirá en cambiar la forma de seleccionar el arco a introducir de tal forma que con cierta probabilidad p (parámetro de intensificación o diversificación) se seleccione de forma proporcional a su valor de la métrica el arco que debe ser introducido y con una probabilidad $1-p$ se siga eligiendo el arco que maximiza la métrica utilizada (ecuación 3.1). De esta forma no cambiaríamos de forma radical todos los atributos que nos ofrece la solución determinista del Algoritmo B.

$$i,j = \begin{cases} \arg\max_{i,j} \left\{ [f(x_i, \pi(x_i) \cup \{x_j\}) - f(x_i, \pi(x_i))] \right\} & \text{con probabilidad } 1-p, \\ \text{Selec. con prob. } \frac{[f(x_i,\pi(x_i)\cup\{x_j\}) - f(x_i,\pi(x_i))]}{\sum_{h,u}[f(x_h,\pi(x_h)\cup\{x_u\}) - f(x_h,\pi(x_h))]} & \text{con probabilidad } p. \end{cases} \tag{3.1}$$

En la ecuación 3.1 hay que especificar dos restricciones: (a) Cuando la inclusión de un arco introduce un ciclo dirigido, entonces no se ha de escoger, en el caso de que estemos maximizando simplemente se ignora; y en el caso de que sorteemos simplemente ponemos esta probabilidad a 0. (b) Cuando la inclusión de un arco nos de una diferencia negativa, por un lado en el caso de que estemos maximizando no hay ningún problema, pues el propio algoritmo B se encarga de no escoger este tipo de arcos; y en el caso de que sorteemos de nuevo pondremos esta probabilidad a 0, de esta forma nos aseguramos de que nunca se escogerá la inclusión de este arco [1].

En este algoritmo la matriz $A[i,j]$ representa una matriz de adyacencias en donde guardamos las diferencias entre la medida con los padres del nodo x_i en la actual etapa con la inclusión en el conjunto de padres del nodo x_j y la medida con los actuales padres $\pi(x_i)$ sin la inclusión del nodo x_j. En este caso identificamos la inclusión de ciclos dirigidos en la actual solución con la asignación a esta matriz del valor $-\infty$, esto se hace en cada etapa buscando los ancestros y los descendientes de un determinado nodo x_i.

El mismo esquema se podría utilizar para buscar puntos de inicio prometedores

[1] En el proceso de sorteo generamos un U[0,1], en el caso de que nos genere un 0 la primera vez, repetimos la generación hasta que nos genere un número diferente de 0

Algoritmo B Probabilístico

1. *Etapa de Inicialización:*
   (a) *Para* $i = 1$ *hasta n hacer:* $\pi(x_i) = \emptyset$
   (b) *Para* $i = 1$ *y* $j = 1$ *hasta n hacer:*
   Si $(i \neq j)$ *Entonces* $A[i,j] = f(x_i, x_j) - f(x_i, \emptyset)$
2. *Etapa Iterativa:*
   (a) *Repetir hasta que* $\forall\, i, j\ (A[i,j] \leq 0$ *ó* $A[i,j] = -\infty)$.
      i. *Seleccionar un par de índices i y j siguiendo la ecuación 3.1*
      ii. *Si* $(A[i,j] > 0)$ *Entonces* $\pi(x_i) = \pi(x_i) \cup \{x_j\}$
      iii. $A[i,j] = -\infty$
      iv. *Para* $x_a \in Ancestros(x_i)$ *y* $x_b \in Descendientes(x_i) \cup \{x_i\}$ *hacer:*
      $A[a,b] = -\infty$.
      v. *Para* $k = 1$ *hasta n hacer:*
      Si $(A[i,k] > -\infty)$ *Entonces* $A[i,k] = f(x_i, \pi(x_i) \cup \{x_k\}) - f(x_i, \pi(x_i))$

Figura 3.3: Algoritmo B Probabilístico

en el espacio de órdenes, pero en lugar de utilizar el algoritmo B, podríamos utilizar el algoritmo K2SN. De esta forma podríamos plantear una versión probabilística de este algoritmo en el mismo sentido del caso anterior, realizando una selección proporcional con probabilidad p y una selección por máximo con una probabilidad $1-p$. Esta versión probabilística del algoritmo K2SN la podemos ver en la figura 3.4.

K2SN Probabilístico

1. $VISITADOS = \emptyset$ *y* $PORVISITAR = VARIABLES$.
2. *Mientras* $|PORVISITAR| > 0$ *hacer*
   - (a) *Si* $(q \leq (1-p))$ $(q \in U[0,1])$ *Entonces*
     - i. $max = -\infty$
     - ii. *Para* $i = 1$ *hasta* $|PORVISITAR|$ *hacer*
       - A. *Sea* $x_i \in PORVISITAR$
       - B. $f = K2(x_i, VISITADOS)$
       - C. $\pi(x_i) = K2(x_i, VISITADOS)$
       - D. *Si* $f(x_i, \pi(x_i)) > max$ *entonces* $max = f(x_i, \pi(x_i))$, $x = x_i$
   - (b) *Si* $(q > (1-p))$ *Entonces*
     - i. *Para* $i = 1$ *hasta* $|PORVISITAR|$ *hacer*
       - A. *Sea* $x_i \in PORVISITAR$
       - B. $f = K2(x_i, VISITADOS)$
       - C. $\pi(x_i) = K2(x_i, VISITADOS)$
       - D. $prob[i] = f(x_i, \pi(x_i))$
     - ii. *Normalizar prob y* $x_s = SORTEAR(prob)$, $x = x_s$
   - (c) $VISITADOS = VISITADOS \cup \{x\}$
   - (d) $PORVISITAR = PORVISITAR \setminus \{x\}$

Figura 3.4: Algoritmo K2SN Probabilístico

3.2.2 Un modelo de colaboración inspirado en los Algoritmos Genéticos paralelos. El Modelo de Islas

Una técnica de coordinación más elaborada y menos global que la anterior, e inspirada en una estrategia de colaboración de los algoritmos Genéticos paralelos es la denominada modelo de Islas, proveniente de conceptos de la teoría de la evolución. En este modelo la población está subdividida en subpoblaciones que viven en áreas independientes y directamente no accesibles unas de otras, denominadas islas. La interacción entre estas subpoblaciones se realiza mediante migraciones entre miembros de islas, que deciden viajar entre islas comunicadas bajo una topología de comunicaciones entre ellas normalmente prefijada; esta topología nos dirá qué islas son accesibles desde cierta isla en cada instante de tiempo y los individuos de una determinada isla sólo pueden viajar a una isla vecina determinada por esta topología.

La principal propiedad de este modelo es que su convergencia parece ser más rápida que el equivalente no distribuido. Además es un modelo que se presta a una directa y simple distribución entre los procesadores disponibles. Algunos autores han utilizado este modelo para distribuir procesos de búsqueda, por ejemplo Laursen [101] utiliza este modelo para proponer una método de estrategia de división para una búsqueda “simulated annealing distribuida”.

Las cuestiones principales a resolver en este modelo de islas son el esquema de migración entre islas, esto es, decidir cuándo se realizan las migraciones, cómo y qué individuos migran entre islas y elegir entre los nuevos y viejos individuos en una isla determinada. Estas reglas son dependientes del problema y se deben ajustar para un problema específico.

Nosotros ahora vamos a proponer una adaptación de este modelo de islas para realizar una búsqueda en entorno variable distribuida con una coordinación basada en este modelo. Vamos a suponer que tenemos en cada procesador una población de h individuos, cada uno de ellos puede ser visto como un estado inicial para empezar a realizar una búsqueda en entorno variable de forma paralela, y cada una de ellas alcanzará un estado final cuando se determine que se deben coordinar las búsquedas entre todos los procesadores. Esta fase de coordinación se puede realizar de varias formas, nosotros hemos decidido coordinar las búsquedas de igual forma que en el esquema anterior, esto es, cuando se termine de realizar la búsqueda con el máximo fijado de vecindad, es decir, cuando realizamos una iteración del VNS para cada individuo de nuestra población. Fijémonos que entonces cada procesador puede ser visto como una isla en el modelo descrito previamente.

Una vez alcanzados los estados finales para cada individuo de todos los procesadores disponibles, entonces se iniciará una fase de migración entre procesadores (islas) vecinos. Se deben elegir los individuos que han de migrar y una vez realizada la migración entre procesadores se debe seleccionar entre los individuos (viejos y

nuevos) para de nuevo tener *h* individuos como estado inicial de esta nueva población. Entonces el algoritmo procederá de la siguiente forma:

a) Todos los procesadores realizan una iteración (interna) del VNS, utilizando sus *h* individuos como estados iniciales de la búsqueda.

b) Dependiendo de la topología de comunicación elegida, emparejamos cada procesador con sus procesadores vecinos y hacemos migrar a un número de individuos de un procesador a otro.

c) Seleccionamos en cada procesador *h* individuos, entre los viejos y los nuevos que acaban de migrar a éste.

d) De nuevo realizamos una nueva iteración (interna) del VNS, hasta que alguna condición de parada prefijada se cumpla.

Este algoritmo se puede ver como un híbrido entre Algoritmos Genéticos y Búsqueda en Entorno Variable: entre las fases de coordinación (búsquedas independientes) se realizan búsquedas VNS y en la fase de coordinación se puede ver como un algoritmo genético. La cuestión ahora es cómo realizar la migración entre procesadores y la selección de individuos una vez realizada la migración. Hemos de fijarnos también que este modelo descrito, después lo matizaremos más, contiene como casos particulares los modelos de coordinación que hemos estudiado al principio de la sección.

Como comentamos más arriba, no hay normas generales y universales para elegir el esquema de migración y selección más adecuado para todo tipo de problemas, así como una topología de comunicaciones.

Para elegir una topología de comunicaciones se puede optar por varias opciones: Una topología en anillo, es estrella, (hiper)cubo, etc. Estas topologías pueden ser estáticas (no varían durante la ejecución del algoritmo) o pueden ser dinámicas. Para nuestras pruebas, hemos elegido una topología en anillo, esto es, el procesador 0 realiza una migración de individuos al procesador 1, éste a su vez realizará una migración de individuos al procesador 2, así hasta que el último procesador realizará una migración al procesador primero 0. En general tendremos que los procesadores se comunicarán con el siguiente patrón: $i \rightarrow (i+1) \bmod n$, siendo n el número de procesadores disponibles. Un ejemplo con 4 procesadores se muestra en la figura 3.5.

Ahora debemos decidir tanto el esquema de migración como de selección que vamos a seguir para nuestro algoritmo. Las opciones que se pueden seguir para realizar el esquema de migración, una vez elegido la isla vecina donde vamos a migrar, pueden ser:

- Poner los j mejores individuos en la isla vecina.

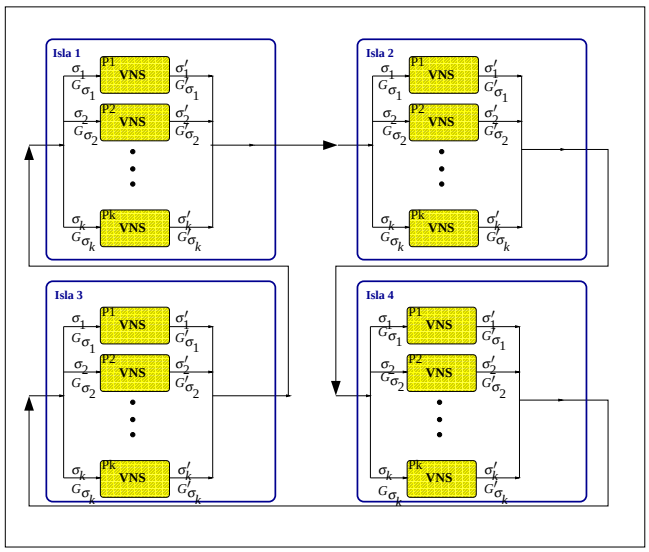


Figura 3.5: Esquema del Algoritmo VNS Distribuido - Búsquedas Coordinadas Modelo de Islas

- Poner solo el mejor individuo.
- Poner todos los individuos.

En nuestro caso vamos a elegir la última opción, es decir, copiamos toda la población h del procesador i al procesador $(i+1) \bmod n$, así temporalmente este último procesador tiene $2h$ individuos en su población. La justificación de esta elección es que nuestro problema es tan complejo que no podemos iterar muchas veces el proceso, con lo cual mediante esta elección aseguramos en poca iteraciones que todos los mejores individuos estén presentes en todos los procesadores. De esta forma dirigimos el proceso de búsqueda hacia las zonas más prometedoras en pocas iteraciones.

Ahora cada procesador debe seleccionar h individuos entre todos los individuos, $(2h)$, para mantener su población. Como podemos imaginar existen múltiples opciones para realizar esta última operación de selección:

- Selección inspirada en el paradigma de los algoritmos Genéticos (selección genética, incluso se podría plantear cruzar y mutar ciertos individuos).
- Selección determinista, se seleccionan directamente los h mejores individuos.

- Selección totalmente aleatoria.

Teniendo en cuenta la filosofía VNS, quizás la estrategia que se deba elegir es la de seleccionar directamente los *h* individuos mejores, esta es la operación de selección que hemos elegido. Por la misma razón planteada en la elección del esquema de migración, en nuestro caso preferimos una rápida convergencia de nuestro algoritmo hacia zonas prometedoras del espacio de búsqueda, ya que como comentamos no podemos realizar demasiadas iteraciones del proceso.

También hemos de notar que aunque hemos elegido un esquema de migración estático entre los *n* procesadores, al realizar como mínimo *n* iteraciones del proceso todos los mejores individuos se mezclarán en todas la islas.

Hemos comentado anteriormente que los esquemas de distribución planteados al principio del capítulo pueden ser vistos como un caso particular del modelo de islas que hemos descrito previamente. Efectivamente si tenemos un solo procesador con *h* individuos, es como si tuviésemos *h* búsquedas en entorno variable de forma independiente, que es el primer esquema planteado. Si tuviésemos *n* procesadores con un solo individuo en su población, después de $i > n$ iteraciones tendríamos el esquema planteado en segundo lugar, pues el mejor individuo estaría replicado en todos los procesadores.

3.3 Aprendizaje Cooperativo y Distribuido de Redes de Creencia mediante Colonias de Hormigas

El Sistema de Colonias de Hormigas (SCH) [53, 54, 52] se ha transformado en más que una metáfora divertida del comportamiento cooperativo y distribuido de ciertas clases de hormigas. Aunque la primera versión, denominada Sistemas de Hormigas, sólo obtuvo pequeñas mejoras en los problemas aplicados, recientes desarrollos y mejoras del anterior sistema han mejorado de largo los primeros resultados obtenidos.

La idea subyacente de este sistema es usar una retroalimentación del sistema, basada en la analogía natural de ciertas especies de hormigas y algunos otros insectos con comportamiento social, para reforzar aquellas soluciones prometedoras que pueden contribuir a mejorar la solución final del sistema. El sistema de refuerzo utilizado es una cantidad de feromona virtual que nos permite memorizar estas soluciones prometedoras, y a partir de éstas intentar mejorar las soluciones en un futuro. Desde luego si sólo permitimos que se refuercen las soluciones mejores, podemos correr el riesgo de que limitemos tanto nuestro espacio de búsqueda que haga que entremos en una solución suboptimal. Para evitar esto se introduce una penalización para aquellas soluciones repetidas en exceso mediante lo que se denomina evaporación de feromona. Esta evaporación se lleva a cabo cada cierto tiempo,

con el compromiso entre el comportamiento cooperativo de las hormigas y la prematura convergencia del sistema, esto es, ni el tiempo puede ser demasiado corto ni demasiado largo, respectivamente.

El comportamiento cooperativo y distribuido en este sistema son conceptos muy importantes, los algoritmos basados en estas ideas hacen uso de la exploración simultánea de diferentes soluciones por una colección de hormigas idénticas. Las hormigas que realizan bien su trabajo influyen de manera decisiva en el comportamiento de futuras hormigas.

3.3.1 El Sistema de Colonias de Hormigas

Los algoritmos de Optimización basados en Colonias de Hormigas [54] son sistemas multi-agente en los que el comportamiento de cada agente (hormiga) está inspirado en el comportamiento de las hormigas reales. Estos algoritmos se inspiran en el proceso seguido por las hormigas para encontrar una ruta mínima entre una fuente de alimento y el nido. Cuando una hormiga pasa por un camino deposita sobre él una sustancia química llamada feromona. Las hormigas usan el rastro de feromona depositado en los caminos para guiarse. Cuando una hormiga se encuentra en un cruce y tiene que decidir el camino a tomar, realiza una decisión de tipo probabilístico en función de la cantidad de feromona depositada sobre cada ramificación. De este modo, inicialmente todos los caminos son igualmente probables, pero con el tiempo los caminos más cortos acaban siendo más frecuentados y, por tanto, reciben un aporte de feromona más alto; mientras que los caminos más largos dejan de ser visitados y la feromona que había en ellos se evapora. Así, podemos ver como la solución final emerge entre la colaboración de los miembros de la colonia.

Inicialmente los algoritmos SCH fueron usados para resolver problemas de búsqueda de caminos en grafos; sin embargo, posteriormente Dorigo y Di Caro [53] introducen una Metaheurística basada en colonias de hormigas, válida para resolver problemas de optimización combinatoria que puedan representarse en forma de grafo. Para ilustrar la resolución de algunos problemas en particular y por la relación estrecha entre estos problemas y el problema que queremos resolver, pasaremos a revisar la aplicación del SCH al problema del viajante de comercio y al problema de la asignación cuadrática.

3.3.1.1 El Sistema de Hormigas para el problema del viajante de comercio

En el problema del viajante de comercio el objetivo es buscar la ruta de mínima longitud que conecta n ciudades. Cada ciudad sólo puede visitarse una y sólo una vez. Sea $d_{i,j}$ la distancia entre dos ciudades i y j. Estas distancias se pueden representar mediante un grafo, en donde los nodos del grafo son las ciudades y los enlaces son las conexiones entre ciudades. La matriz de distancias entre ciudades no tiene

porque ser simétrica, esto es $d_{i,j} \neq d_{j,i}$, este aspecto es irrelevante en el caso de su resolución mediante el sistema de hormigas.

Durante una iteración t del sistema de hormigas cada hormiga k construye una ruta completa, empezando por una ciudad i, elegida de forma aleatoria. Una vez emplazados en una ciudad concreta, la hormiga debe elegir la siguiente ciudad j aún no visitada aplicando una regla de transición probabilística. Para cada hormiga k la transición entre las ciudades i y j en una iteración t del algoritmo depende de los siguientes aspectos:

- Para cada hormiga k se define un conjunto $J_k(i)$ de ciudades que aún no han sido visitadas por esta hormiga durante la construcción de su ruta.
- Se define la visibilidad como la inversa de la distancia entre dos ciudades $\eta_{i,j} = 1/d_{i,j}$. Esta información es estrictamente local a cada pareja de ciudades y puede representar una información de preferencia para elegir la próxima ciudad a visitar estando en una determinada ciudad i. Esta información se utilizará como heurística en el proceso de construcción de una ruta por parte de una hormiga. Hemos de notar que esta información es de carácter estático, es decir, no cambia en ningún momento.
- La cantidad de feromona virtual en una determinada iteración del algoritmo t depositada en una conexión entre las ciudades i y j, que denominaremos $\tau_{i,j}(t)$. Esta feromona se irá actualizando durante cada iteración del proceso y representa la preferencia aprendida por el conjunto de hormigas de transición entre las ciudades i y j.

La regla de transición define la probabilidad para una hormiga k de elegir la ciudad j estando situada en la ciudad i. Se define como:

$$p_k(i,j) = \begin{cases} \frac{[\tau_{i,j}]^\alpha [\eta_{i,j}]^\beta}{\sum_{u \in J_k(i)} [\tau_{i,u}]^\alpha [\eta_{i,u}]^\beta} & \text{si } j \in J_k(i), \\ 0 & \text{en otro caso.} \end{cases} \tag{3.2}$$

donde $\eta_{i,j}$ representa la información heurística acerca del problema y $J_k(i)$ es el conjunto de nodos a los que podemos ir desde el nodo i y que aún no han sido visitados por la hormiga k; finalmente α y β son dos parámetros que representan la importancia relativa de la feromona con respecto a la información heurística.

En una segunda versión los autores cambian esta regla de transición entre los nodos de forma que se pueda decidir de forma paramétrica cuándo se realiza una mayor explotación del conocimiento y cuándo se realiza una mayor exploración en el espacio de búsqueda. Ahora el siguiente nodo j a visitar se elige como:

$$j = \begin{cases} \arg\max_{u \in J_k(i)} \left\{ [\tau_{i,u}][\eta_{i,u}]^\beta \right\} & \text{si } q \leq q_0, \\ J & \text{si } q > q_0. \end{cases} \tag{3.3}$$

donde q es un número aleatorio entre $[0,1]$ y $0 \leq q_0 < 1$ es el parámetro para controlar la explotación o exploración. $J \in J_k(i)$ es un nodo seleccionado aleatoriamente de acuerdo a las probabilidades calculadas usando la ecuación 3.2 tomando $\alpha = 1$.

En cada iteración del algoritmo cada hormiga de la colonia construye de forma progresiva (aplicando la regla anterior 3.3) una solución al problema. En función de las soluciones obtenidas por las hormigas se actualiza la matriz de feromona:

- Actualización *global*: Consiste en *reforzar* la cantidad de feromona en los arcos que forman parte de la mejor solución hasta el momento, S^+. Esto favorece la búsqueda en la vecindad de este camino, es decir, la exploración es más dirigida. La regla de actualización global de feromona es:

$$\tau_{i,j} \leftarrow (1-\rho) \cdot \tau_{i,j} + \rho \cdot \Delta\tau_{i,j} \tag{3.4}$$

  donde (i,j) son las aristas pertenecientes a la solución S^+; ρ es el parámetro que controla la evaporación de la feromona; y $\Delta\tau_{i,j} = \frac{1}{C(S^+)}$, con $C(\cdot)$ la función de coste de una solución.

- Actualización *local*: Durante el proceso de construcción de una solución (secuencia) por parte de una hormiga *k*, cuando se realiza una transición desde el nodo *i* hasta el nodo *j* se actualiza el nivel de feromona en la arista (i,j) aplicando la siguiente expresión:

$$\tau_{i,j} \leftarrow (1-\rho) \cdot \tau_{i,j} + \rho \cdot \tau_0 \tag{3.5}$$

  donde τ_0 es la cantidad de feromona inicial de cada arista. Por tanto cada vez que una arista es visitada su nivel de feromona disminuye, haciéndola cada vez menos atractiva. Esta regla de actualización local favorece la exploración de aristas todavía no visitadas, evitándose así una convergencia prematura hacia un camino común. Sin la regla de actualización local todas las hormigas se limitarían a buscar en la vecindad del mejor camino encontrado hasta el momento, evitándose la posibilidad de que emerjan otras soluciones.

Otras mejoras introducidas por el sistema de colonias de hormigas a los sistemas previos de hormigas son:

- Usar una lista de candidatos. Se trata de tener una lista de candidatos o nodos preferidos desde un determinado nodo *i*, de forma que no sea necesario examinar todos los candidatos en $J_k(i)$. Sólo cuando ya se han visitado todos los nodos candidatos de *i* se pasa a examinar el resto de nodos.

- Usar un optimizador local. Todas o algunas de las soluciones aportadas por las hormigas son optimizadas por medio de algún algoritmo de búsqueda local. Esta técnica se ha mostrado especialmente eficaz en problemas de tamaño grande.

3.3.1.2 El Sistema de Colonias de Hormigas para el problema de la asignación cuadrática

Este problema puede ser definido como sigue: Consideramos un conjunto de actividades *n* que deben ser asignadas a un conjunto de *n* localizaciones o viceversa. Una matriz $D = [d_{i,j}]_{n,n}$, al igual que en el problema del viajante de comercio, representa las distancias entre las diferentes localizaciones, y una matriz $F = [f_{h,k}]_{n,n}$ caracteriza el flujo (transferencia de datos, material, ...) entre la actividad *h* y la actividad *k*. Una asignación es una permutación π de $\{1 \dots n\}$, donde $\pi(i)$ es la actividad que ha sido asignada a la localización *i*. El problema es encontrar la permutación óptima tal que la suma de los productos entre los flujos y actividades sea mínimo. Al igual que el problema del viajante de comercio se quiere establecer un orden entre las variables del problema, la única diferencia es que en este problema sí que es importante tanto el nodo inicial de una permutación como la ordenación entre las variables.

En este caso la matriz de feromona $\tau_{i,j}$ cambia su significado. Si bien en el problema del viajante de comercio, esta matriz representa la feromona virtual depositada sobre la arista (i, j), esto es, la fuerza relativa aprendida de que la ciudad *j* es la siguiente a la ciudad *i*, en el problema que nos ocupa representa que la actividad *j* es asignada en la localización *i*, en definitiva qué lugares ocupan cada una de las actividades en la permutación correspondiente.

El resto de componentes del algoritmo para aplicar el sistema de colonias de hormigas al problema de la asignación cuadrática puede ser el mismo que el caso anterior, excepto qué duda cabe la componente heurística, que en este caso es específica al problema que se quiere resolver.

3.3.1.3 Sistema Híbrido de Hormigas

Gambardella y col. [18, 63] proponen un nuevo sistema de hormigas para optimización combinatoria y lo aplican al problema de la asignación cuadrática, denominado Sistema Híbrido de Hormigas, en el cual cambia de forma considerable el esquema seguido por los algoritmos previos. Las principales novedades son:

- Las hormigas del sistema modifican de forma parcial una solución, más que construir una solución como en los casos anteriores.
- La matriz de feromona se usa como guía para modificar de forma parcial una solución existente, más que como ayuda para la construcción de una solución nueva.

En una fase de inicialización cada hormiga *k* obtiene una permutación π^k de forma aleatoria, esta permutación es mejorada con una búsqueda local parcial que posteriormente definiremos más detalladamente. Los valores iniciales de la matriz de feromona $\tau_{i,j}$ (con el mismo significado que en el problema de la asignación

cuadrática) son asignados al valor $\tau_0 = 1/(Q \cdot C(\pi^+))$, donde π^+ es el coste asociado a la mejor solución aportada por las hormigas en su fase de inicialización. Entonces el sistema consta de tres procedimientos descritos a continuación:

Modificación de la Matriz de Feromona. Cada hormiga k modifica su permutación asociada π^k aplicando un conjunto de R intercambios, donde R es un nuevo parámetro del algoritmo: Un índice i es elegido aleatoriamente entre $\{1,\ldots,n\}$, entonces posteriormente se elige otro índice $j \neq i$ entre $\{1,\ldots,n\}$ y se intercambian en la permutación π^k. La selección del índice j se realiza en uno de los dos siguientes modos: Con probabilidad q, j se elige tal que $\tau_{i,\pi^k(j)} + \tau_{j,\pi^k(i)}$ es máximo, lo que corresponde a la explotación del conocimiento memorizado en la feromona. Con probabilidad $(1-q)$, j es seleccionado con probabilidad:

$$p_{i,j}^k = \frac{\tau_{i,\pi^k(j)} + \tau_{j,\pi^k(i)}}{\sum_{l=1,l\neq i}^{n} (\tau_{i,\pi^k(l)} + \tau_{l,\pi^k(i)})} \tag{3.6}$$

lo que corresponde a la estrategia de exploración. Estos R intercambios generan otra solución $\widetilde{\pi}^k$.

Búsqueda Local. La solución generada en el paso anterior $\widetilde{\pi}^k$ se va a transformar en otra nueva solución $\widehat{\pi}^k$ mediante una búsqueda local. Este procedimiento local va a examinar todos los posibles intercambios de forma aleatoria y se va a quedar con el primer intercambio que mejore la solución inicial $\widetilde{\pi}^k$. Entonces esta solución así generada será $\widehat{\pi}^k$ (realizamos sólo un intercambio y sólo uno tal que mejore la solución inicial). Hemos de notar que $\widehat{\pi}^k$ no tiene por qué ser un óptimo local en una búsqueda a no ser que ningún intercambio mejore la solución inicial.

Actualización de la Matriz de Feromona. Cuando todas las hormigas han realizado los dos anteriores pasos, hemos de actualizar la matriz de feromona de forma similar a cómo lo realizamos en los anteriores casos o utilizando la siguiente regla de actualización:

$$\tau_{i,\pi(i)}(t) \leftarrow (1-\rho) \cdot \tau_{i,\pi(i)}(t) + \rho \cdot \Delta\tau_{i,\pi(i)}(t), \tag{3.7}$$

donde $\Delta\tau_{i,\pi(i)}(t) = 1/C(\pi^+)$, si $(i,\pi(i)) \in \pi^+$, y 0 en otro caso. Como en los anteriores casos $0 \leq \rho < 1$ representa el parámetro de evaporación de la feromona.

Este esquema tiene el problema siguiente: como la actualización de feromona se realiza en función de $C(\pi^+)$ en una iteración dada, la distribución de las nuevas soluciones se determina por soluciones pasadas. Cuando una nueva solución mejora la actual, ésta se convierte en el nuevo máximo lógicamente. En general, deben pasar ciertas iteraciones más hasta que esta nueva solución influya en la construcción de

nuevas soluciones. Es por esta razón que se efectúa un procedimiento, denominado *intensificación*, que se activa cuando se encuentra una mejor solución.

Finalmente, para prevenir la caída en un óptimo local, se realiza un procedimiento denominado *diversificación*, que consiste en volver a actualizar toda la matriz de feromona a su valor de inicialización $\tau_0 = 1/(Q \cdot C(\pi^+))$ si han pasado más de S iteraciones sin que se haya producido una mejora en la solución máxima actual, y se reinicia el proceso.

3.3.1.4 Algoritmo General para Optimización basado en Colonias de Hormigas

Aunque los métodos desarrollados en las secciones anteriores puedan parecer específicos a los problemas que resuelven, se puede establecer una metaheurística para problemas de optimización combinatoria basada en Colonias de Hormigas.

Un sistema basado en colonias de hormigas realiza en general t_{max} iteraciones con la aplicación de dos procedimientos básicos:

- Un procedimiento de construcción/modificación paralela o distribuida de soluciones al problema específico a resolver basada en experiencias previas.
- Una procedimiento de actualización de una matriz de feromona virtual en función de las soluciones aportadas por el proceso anterior.

La construcción/modificación de las soluciones al problema se realiza de forma probabilística en función de un conocimiento heurístico al problema particular que se esta resolviendo y del peso aportado por la feromona virtual (experiencia acumulada en el proceso).

La actualización de la matriz de feromona virtual se realiza en función de la calidad de la solución aportada por una hormiga de la colonia y de un nivel de evaporación de ésta, ρ.

En función de lo comentado en los párrafos anteriores, el sistema de colonias de hormigas se puede aplicar a cualquier problema de optimización combinatoria siempre y cuando se puedan definir las siguientes características en el problema que se pretende resolver:

- Una representación del problema adecuada, tal que nos permita incrementalmente la construcción/modificación de soluciones al problema, usando una regla probabilística de transición que haga uso de la matriz de feromona y de una heurística local apropiada para el problema en cuestión.
- Una heurística η sobre arcos.
- Un método de satisfacción de restricciones el cual fuerce la construcción de un solución posible.

- Una regla de actualización de feromona que especifique cómo se modifica la cantidad de feromona depositada en cada arco del grafo que representa.
- Una regla de transición probabilística en función de la heurística η y de la feromona depositada previamente.

En la figura 3.6 se puede observar un algoritmo general con las consideraciones previas:

Algoritmo SCH

1. *Para cada arco* (i, j) *inicializar la feromona* $\tau_{i,j}(0) = \tau_0$
2. *Para* $t = 1$ *hasta* t_{max} *hacer*
   (a) *Para* $k = 1$ *hasta m hacer*
      i. *Construir una solución al problema* $S^k(t)$ *aplicando la regla de transición probabilística definida para el problema en función de la matriz de feromona* τ *y de la heurística local al problema* η.
   (b) *Para* $k = 1$ *hasta m hacer*
      i. *Calcular el costo asociado para cada solución* $C^k(t)$ *para cada* $S^k(t)$ *construida por cada hormiga k*
   (c) *Si el coste de alguna solución mejora la actual entonces actualizarla*
   (d) *Para cada arco* (i, j) *actualizar la feromona aplicando la regla de actualización de feromona definida para el problema*

Figura 3.6: Algoritmo General de un Sistema de Colonias de Hormigas

3.3.2 Aprendizaje de Redes de Creencia mediante Colonias de Hormigas en el espacio de Órdenes

En esta subsección del capítulo vamos a adaptar el método de colonias de hormigas utilizado en el problema del viajante de comercio para resolver el problema de aprender un red de creencia en el espacio de órdenes. En primer lugar hemos de definir la componente heurística principal de los esquemas anteriormente estudiados para la aportación del conocimiento específico a nuestro problema.

Hemos visto que la heurística K2SN [48] para la obtención de una secuencia de ordenación entre las variables es una heurística eficiente y, a tenor de los resultados

presentados en el capítulo anterior, la red que ofrece como resultado es bastante aceptable, sobre todo teniendo en cuenta la poca información que requiere. Nuestra intención es utilizar esta heurística como la componente heurística para el sistema de colonias de hormigas.

Sin embargo, hemos de tener en cuenta que, mientras la información heurística en general es estática para los problemas presentados previamente, es decir, no varía durante la ejecución, en nuestro caso la heurística que utilizamos es dinámica. Así, por ejemplo en el problema del viajante de comercio, la distancia desde la ciudad i hasta la ciudad j no cambia con el tiempo; en la heurística K2SN el valor del paso del nodo i hasta el nodo j dependerá de los nodos que preceden al nodo j, pues de ellos obtendrá su conjunto de padres y por tanto su valor de la medida descomponible que utilicemos, $f(x_j, \pi(x_j))$, con $\pi(x_j) \in$ Predecesores en el orden. Es por ello que cada hormiga k del sistema realiza una búsqueda K2SN, pero esta vez también teniendo en cuenta la feromona depositada en la arista correspondiente.

A continuación vamos a detallar cada una de las componentes necesarias para el modelado de nuestro sistema de colonias de hormigas.

- En primer lugar hemos de plantear la representación de nuestro problema en forma de grafo. Si bien esto puede parecer fácil dado que nuestra red buscada es un grafo, recordaremos que nuestro espacio de búsqueda es el de las secuencias de ordenación entre las variables. En nuestro problema, el grafo en el que buscaremos las soluciones será un grafo completo definido sobre todas las variables de la red, ya que al ser el espacio de búsqueda el conjunto de permutaciones posibles, el paso de un nodo i a cualquier nodo j debe estar permitido. Hay que resaltar también que nuestro problema es asimétrico, en el sentido de que no es lo mismo, en general, ir desde el nodo i hasta el nodo j que viceversa.

  Para iniciar la matriz de feromona usaremos el mismo criterio que el propuesto para el problema del viajante de comercio por Dorigo y col. [53]; es decir $\tau_0 = \frac{1}{n \cdot C(S_{K2SN})}$, donde n es el número de variables, S_{K2SN} es la solución obtenida por la heurística K2SN y $C(S_{K2SN})$ es el valor de la medida de la solución anterior.

- En el problema del viajante de comercio la componente heurística (llamada visibilidad) se definió como la inversa de la distancia, $\eta_{i,j} = \frac{1}{d_{i,j}}$. En nuestro caso, teniendo en cuenta que es una maximización de la medida f, normalmente vamos a utilizar no la maximización de la verosimilitud $P(D|G)$ si no la maximización de la log-verosimilitud $\log P(D|G)$, con lo que el resultado de f será negativo, es por ello que nosotros utilizaremos como medida $\eta_{i,j} = \frac{1}{|f(x_j, \pi(x_j))|}$, con $\pi(x_j)$ en los predecesores en la secuencia de variables, todo ello calculado por la heurística K2SN.

- Usar una lista de candidatos en el problema del viajante de comercio no es problema porque la información heurística, como hemos comentado previamente, es estática; por tanto, en cada momento se tendrán las *cl* ciudades más cercanas al nodo *i* en esta lista de candidatos y no supondrá realizar ningún calculo adicional. Por otra parte es evidente que para problemas de gran tamaño es bastante interesante mantener esta lista desde el punto de vista computacional. En nuestro problema, mantener esta lista de candidatos sería de un coste computacional elevado, y no merece la pena porque obtener los *cl* nodos más cercanos a un nodo *i* supondría realizar los mismos cálculos que en el caso de no mantener dicha lista, esto es, tendríamos que calcular todos los restantes para quedarnos con los mejores. Sin embargo, sí que podemos hacer estática la información que nos ofrece la heurística K2SN con la que inicializamos el proceso y suponer que los más cercanos a un nodo *i* son los siguientes nodos en esta secuencia de inicialización. Por consiguiente, para nosotros mantener una lista de *cl* candidatos sería quedarse con los *cl* nodos siguientes en la ordenación de partida que nos ofrece la inicialización del proceso. La justificación de esta elección es naturalmente desde el punto de vista computacional y siempre podremos partir de que esta lista es la total. De todas formas hemos de tener en cuenta que la secuencia de partida es lo suficientemente buena como para realizar esta suposición en la elección de la lista de candidatos.

- Otro aspecto importante y diferente al problema del viajante de comercio, es la elección del nodo de partida. En el problema del viajante de comercio el punto de inicio es indiferente para el resultado o previamente es prefijado en el problema. En nuestro caso, la elección de primer nodo es muy importante, ya que deberemos de elegir un nodo como primero de la secuencia que sea un nodo raíz. Una forma de elección podría ser al azar, pero parece que una elección más juiciosa puede ser la elección que realice la heurística K2SN ponderada (utilizando la regla de transición) por la feromona en el punto de partida, esto es, consideraremos el nodo de partida como otro nodo más atendiendo a $\tau_{-1,i}$ y $\eta_{-1,i}$, siendo *i* el nodo inicial. Esto no supone almacenamiento extra en la matriz de feromona, ya que podemos utilizar para almacenar la feromona en el nodo de partida la diagonal de la matriz.

- Las reglas de actualización de feromona global y local son las mismas que en el problema de viajante de comercio (ecuaciones 3.4 y 3.5), donde de nuevo (i, j) son las aristas pertenecientes a la mejor solución encontrada S^+; ρ es el parámetro que controla la evaporación de la feromona; y $\Delta\tau_{i,j} = \frac{1}{|f(S^+)|}$, con $f(\cdot)$ la medida utilizada en nuestro problema.

- Por último, como procedimiento de optimización local, escogeremos de for-

ma natural el algoritmo propuesto en el capítulo anterior HCSN [49].

Una vez descritas las componentes básicas del algoritmo y teniendo en cuenta la metaheurística descrita en la figura 3.6, la componente que hemos de especificar en más detalle es cómo construiría una solución una hormiga en nuestro caso, en la figura 3.7 especificamos con más detalle este punto.

Hibridación con el espacio de grafos dirigidos acíclicos. En el anterior esquema tenemos todos los ingredientes necesarios para poder aplicar la metaheurística del Sistema de Colonias de Hormigas al problema del aprendizaje de redes de creencia en el espacio de órdenes, este espacio de búsqueda es casi el mismo que se aplicaba al problema del viajante de comercio. Sin embargo, hay un punto en nuestra adaptación al problema que puede limitar la eficacia de la búsqueda mediante el Sistema de Colonias de Hormigas. Si observamos de nuevo el algoritmo de la figura 3.7, la forma de buscar los padres de un determinado nodo x_i de entre todos los predecesores (*VISITADOS*) en la secuencia de ordenación es de forma determinista utilizando para ello una heurística totalmente greedy. Así nos enfrentamos de alguna manera a la filosofía introducida en la metaheurística que estamos aplicando, en el sentido de que la evolución de esta búsqueda no se maneje de forma similar a como se hace en el espacio de órdenes.

Nuestra propuesta es mantener una segunda matriz de feromona δ que represente la estructura gráfica de un grafo dirigido acíclico y de esta forma ponderar la aparición de los arcos $x_j \rightarrow x_i$ en las soluciones que van construyendo las hormigas en cada iteración. De esta forma, en cada iteración una hormiga va a buscar los padres de un determinado nodo x_i, en la secuencia que va construyendo, de entre sus predecesores (*VISITADOS*), pero esta vez aplicando una regla de transición probabilística en función de la matriz de feromona δ que posteriormente especificaremos con más detalle.

La regla de actualización de esta nueva matriz de feromona para las estructuras es similar a las presentadas anteriormente salvo un nuevo matiz en la regla de actualización global: La regla de actualización local de la feromona será la siguiente:

$$\delta_{i,j} \leftarrow (1-\rho) \cdot \delta_{i,j} + \rho \cdot \delta_0 \tag{3.8}$$

donde δ_0 será la misma inicialización aplicada para la matriz de feromona de secuencias τ_0.

La regla de actualización global de la matriz de feromona δ será:

$$\delta_{i,j} \leftarrow (1-\rho) \cdot \delta_{i,j} + \rho \cdot \Delta C^* \tag{3.9}$$

donde $\Delta C^* = 1/|f(x_j, \pi(x_j))|$ si y solo si existe el arco $x_i \rightarrow x_j$ en la máxima red obtenida S_G^+, es decir, reforzamos en la matriz de feromona el arco $x_i \rightarrow x_j$ con el

Hormiga K2SN

1. $VISITADOS = \emptyset$ *y* $PORVISITAR = VARIABLES$.
2. $anterior = 0$
3. *Mientras* $|PORVISITAR| > 0$ *hacer*
   (a) $j = anterior$
   (b) *Si* $(q \leq q_0)$ $(q \in U[0,1])$ *entonces*
      i. $max = -\infty$
      ii. *Para* $i = 1$ *hasta* $|PORVISITAR|$ *hacer*
         A. *Sea* $x_i \in PORVISITAR$
         B. $f = K2(x_i, VISITADOS)$
         C. $\pi(x_i) = K2(x_i, VISITADOS)$
         D. *Si* $(\tau_{j,i} * [1/|f(x_i, \pi(x_i))|]^{\beta}) > max$ *entonces* $max = (\tau_{j,i} * [1/|f(x_i, \pi(x_i))|]^{\beta})$, $x = x_i$.
   (c) *Si* $(q > q_0)$ *entonces*
      i. *Para* $i = 1$ *hasta* $|PORVISITAR|$ *hacer*
         A. *Sea* $x_i \in PORVISITAR$
         B. $f = K2(x_i, VISITADOS)$
         C. $\pi(x_i) = K2(x_i, VISITADOS)$
         D. $prob[i] = (\tau_{j,i} * [1/|f(x_i, \pi(x_i))|]^{\beta})$
      ii. *Normalizar prob y* $x_s = SORTEAR(prob)$, $x = x_s$
   (d) *Aplicar la actualización local de feromona con el índice asociado a x y anterior*
   (e) *Hacer anterior igual al índice asociado a x*
   (f) $VISITADOS = VISITADOS \cup \{x\}$
   (g) $PORVISITAR = PORVISITAR \setminus \{x\}$

Figura 3.7: Hormiga K2SN

valor que obtiene su familia ($x_j \cup \pi(x_j)$ con $x_i \in \pi(x_j)$) en el grafo dirigido acíclico máximo (recordemos que hemos asumido que tenemos una métrica descomponible, de todas formas si la métrica f no fuese descomponible, se podría reforzar la feromona del arco correspondiente con el valor total de la solución completa, esto es, el valor de la métrica del grafo dirigido acíclico máximo).

Una vez conocido cómo se actualiza la feromona de forma global y cómo se actualiza la feromona cada vez que se elige un nuevo padre x_i de entre sus predecesores para el nodo x_j, necesitamos conocer cómo realizaremos la selección del conjunto de padres del nodo x_j de entre sus predecesores. Para ello necesitamos establecer dos componentes:

(a) Una regla de transición probabilística que elija el siguiente padre a insertar para el nodo x_j.

(b) Un criterio de parada para dejar de introducir nuevos padres para el correspondiente nodo x_j.

La regla de transición probabilística que vamos a elegir tendrá como componente principal la siguiente expresión:

$$[\delta_{i,j}] \times [f(x_j, \pi(x_j) \cup \{x_i\}) - f(x_j, \pi(x_j))]^\beta$$

donde dependiendo del parámetro q_0 (intensificación o exploración) nos quedaremos con el máximo x_i o realizaremos una selección proporcional en el sentido de las ecuaciones 3.2 y 3.3 (posteriormente las detallaremos más).

El criterio de parada de introducir nuevos padres para un nodo x_j será que ninguno de los padres candidatos a introducir den una diferencia positiva. Esto es, vamos introduciendo padres mientras se vaya mejorando la medida de la familia completa. En la figura 3.8 podemos observar más detalladamente este proceso.

En el algoritmo anterior (figura 3.8), $A[i]$ representa la diferencia entre la medida de introducir x_i como padre de x_j y la medida no introduciéndolo. Teniendo esto último en cuenta, la regla de transición probabilística quedarían de forma más detallada como:

$$i = \begin{cases} \arg\max_{u \ / \ A[u]>0} \left\{ [\delta_{u,j}] \times [A[u]]^\beta \right\} & \text{si } q \leq q_0, \\ J & \text{si } q > q_0. \end{cases} \tag{3.10}$$

donde J es nodo seleccionado aleatoriamente de acuerdo a la siguiente probabilidad:

$$p_k(i,j) = \begin{cases} \frac{[\delta_{i,j}] \times [A[i]]^\beta}{\sum_{u \ / \ A[u]>0} [\delta_{u,j}] \times [A[u]]^\beta} & \text{si } A[i] > 0, \\ 0 & \text{en otro caso.} \end{cases} \tag{3.11}$$

Hemos de notar que en ambas expresiones no se tienen en cuenta aquellas transiciones que nos de un diferencia negativa, en estos casos simplemente se considerará una probabilidad de transición nula.

Hormiga K2

1. *Entradas:* x_j, *nodo al que debemos buscarle los padres.*
   VISITADOS, nodos predecesores en el orden a x_j.
2. $\pi(x_j) = \emptyset$
3. *Para* $i = 1$ *hasta* $|VISITADOS|$ *hacer:*
   $A[i] = f(x_j, \pi(x_j) \cup \{x_i\}) - f(x_j, \pi(x_j))$
4. *Repetir hasta que* $(A[i] \leq 0, \ \forall i)$ *ó* $\pi(x_i) = VISITADOS$.
   (a) *Seleccionar el nodo* x_i *atendiendo a la regla de transición probabilística descrita en 3.10 y 3.11 del conjunto* $VISITADOS \setminus \pi(x_j)$
   (b) *Si* $(A[i] > 0)$ *Entonces* $\pi(x_j) = \pi(x_j) \cup \{x_i\}$
   (c) *Para* $k = 1$ *hasta* $|VISITADOS|$ *hacer:*
   Si $x_k \notin \pi(x_j)$ *Entonces* $A[k] = f(x_j, \pi(x_j) \cup \{x_k\}) - f(x_j, \pi(x_j))$
5. *Salida:* $\pi(x_j)$

Figura 3.8: Hormiga K2

3.3.3 Aprendizaje de Redes de Creencia mediante Colonias de Hormigas en el espacio de Grafos Dirigidos Acíclicos

Parece razonable que si nuestro problema de forma natural es un grafo (dirigido acíclico), utilicemos esta representación para la aplicación de la metaheurística del sistema de colonias de hormigas. Es por ello que en esta subsección nos plantearemos cómo podemos adaptar el sistema de colonias de hormigas, para el aprendizaje de redes de creencia basado en el espacio de grafos dirigidos acíclicos.

Ya hemos comentado en la subsección 3.3.1.4 que el sistema de colonias de hormigas se puede aplicar a cualquier problema de optimización combinatoria si podemos definir:

- Una representación del problema adecuada. La representación que elegimos de forma natural será la de un grafo dirigido acíclico, componente de las redes de creencia que estamos buscando. De esta forma incrementalmente iremos eligiendo arcos entre las variables que pertenecerán a la solución aportada por cada hormiga de la colonia. De esta forma, tal como teníamos en la subsección anterior, tendremos una matriz de feromona δ que pondera la fuerza de cada arco dependiendo de su bondad en la aparición de las distintas soluciones aportadas por las hormigas.

- Una heurística η sobre arcos. En este caso la heurística que se utilizará será la de ir introduciendo aquellos arcos que mejoren nuestra medida. Esto es, se elegirá el arco tal que resulta una mayor diferencia en la expresión:

$$f(x_i, \pi(x_i) \cup \{x_j\}) - f(x_i, \pi(x_i))$$

- Un método de satisfacción de restricciones el cual fuerce la construcción de un solución posible. En este caso será doble: Por un lado tendremos que identificar, una vez introducido un nuevo arco a nuestra solución, aquellos arcos que al introducirlos creen ciclos dirigidos, para en posteriores etapas no tenerlos en cuenta. Y por otro lado tendremos que identificar cuándo la introducción de un arco no mejora nuestra medida, esto es, la diferencia no es positiva en la expresión anterior para ningún arco no introducido previamente.

- Una regla de actualización de feromona. En este punto tenemos que especificar tanto la inicialización de la matriz de feromona δ_0, en donde utilizaremos el mismo criterio que el de anterior caso, $\delta_{i,j} = \delta_0 = 1/n|C(S_{K2SN})|$. Las reglas de actualización global y local de la feromona serán las mismas que la utilizadas en las expresiones 3.9 y 3.8.

- Una regla de transición probabilística. En este caso elegiremos las ecuaciones 3.2 y 3.3.

Hormiga B

1. *Etapa de Inicialización:*
   (a) *Para* $i = 1$ *hasta n hacer:* $\pi(x_i) = \emptyset$
   (b) *Para* $i = 1$ *y* $j = 1$ *hasta n hacer:*
   Si $(i \neq j)$ *Entonces* $A[i, j] = f(x_i, x_j) - f(x_i, \emptyset)$
2. *Etapa Iterativa:*
   (a) *Repetir hasta que* $\forall\, i, j\ (A[i, j] \leq 0$ *ó* $A[i, j] = -\infty)$.
      i. *Seleccionar un par de índices* i *y* j *atendiendo a la regla de transición probabilística descrita en 3.12 y 3.13*
      ii. *Si* $(A[i, j] > 0)$ *Entonces* $\pi(x_i) = \pi(x_i) \cup \{x_j\}$
      iii. $A[i, j] = -\infty$
      iv. *Para* $x_a \in Ancestros(x_i)$ *y* $x_b \in Descendientes(x_i) \cup \{x_i\}$ *hacer:*
      $A[a, b] = -\infty$.
      v. *Para* $k = 1$ *hasta n hacer:*
      Si $(A[i,k] > -\infty)$ *Entonces* $A[i,k] = f(x_i, \pi(x_i) \cup \{x_k\}) - f(x_i, \pi(x_i))$

Figura 3.9: Hormiga B

Teniendo en cuenta estas consideraciones, la hormiga que construye el grafo acíclico solución será la representada en la figura 3.9. Este algoritmo es el algoritmo B [23], en donde la forma de elegir el arco que se introduce en cada etapa se hace con arreglo a una determinada regla de transición probabilística, en lugar de elegir siempre el que mayor mejora produce en la métrica utilizada, además de por supuesto tener en cuenta la ponderación de la experiencia acumulada en las iteraciones previas en la matriz de feromona. En este algoritmo, al igual que el algoritmo de la figura 3.3, la matriz $A[i,j]$, representa una matriz de adyacencias en donde guardamos las diferencias entre la medida con los padres del nodo x_i en la actual etapa con la inclusión en el conjunto de padres del nodo x_j y la medida con los actuales padres $\pi(x_i)$ sin la inclusión del nodo x_j. En este caso identificamos la inclusión de ciclos dirigidos en la actual solución con la asignación a esta matriz del valor $-\infty$, esto se hace en cada etapa buscando los ancestros y los descendientes de un determinado nodo x_i en el actual grafo que tenemos como solución parcial.

Teniendo en cuenta todas estas consideraciones, la ecuación que utilizaremos como regla de transición probabilística quedará de la siguiente forma:

$$i,j = \begin{cases} \arg\max_{h,u \ / \ A[h,u]>0} \left\{ [\delta_{h,u}] \times [A[h,u]]^{\beta} \right\} & \text{si } q \leq q_0, \\ J & \text{si } q > q_0. \end{cases} \tag{3.12}$$

donde J es un nodo seleccionado aleatoriamente de acuerdo a la siguiente probabilidad:

$$p_k(i,j) = \begin{cases} \frac{[\delta_{i,j}] \times [A[i,j]]^{\beta}}{\sum_{u,h \ / \ A[h,u]>0} [\delta_{h,u}] \times [A[h,u]]^{\beta}} & \text{si } A[i,j] > 0, \\ 0 & \text{en otro caso.} \end{cases} \tag{3.13}$$

Hemos de notar que en ambas expresiones no se tienen en cuenta aquellas transiciones que nos de un diferencia negativa, en estos casos simplemente se considerará una probabilidad de transición nula.

3.4 Un Sistema Distribuido Híbrido entre la Búsqueda VNS y el El Sistema de Colonias de Hormigas

A nuestro parecer, lo más importante que aporta el Sistema de Colonias de Hormigas, es la utilización de un conocimiento heurístico específico al problema junto con una búsqueda probabilística que resume las experiencias pasadas durante la visita de diferentes “buenas” soluciones a nuestro problema.

Por otra parte hemos constatado durante las pruebas realizadas, que en el proceso de búsqueda de redes de creencia, las estructuras de redes con un buen valor de la

métrica utilizada, en nuestro caso la métrica K2, normalmente tienen más partes en común que dispares (este tipo de situaciones es muy evidente en dominios como ALARM). Es por ello que parece razonable que estas partes comunes que aparecen repetidas veces en redes de alta calidad tengan una mayor probabilidad de aparecer también en el óptimo global buscado.

También hemos observado que el proceso de búsqueda VNS obtiene durante su ejecución bastantes buenos óptimos locales, esto es, redes de alta calidad sin que sea el óptimo finalmente alcanzado por el proceso, pero como hemos comentado en el párrafo anterior estos óptimos locales normalmente tienen bastante de común en cuanto a la estructura obtenida.

Una forma de acumular estas buenas soluciones tal que influyan en las posteriores búsquedas es ponderar los atributos de éstas, tal como lo hace el Sistema de Colonias de Hormigas mediante el refuerzo de la matriz de feromona, de tal forma que aquellas partes que sean más comunes tengan una mayor probabilidad de aparecer en soluciones posteriores.

Nuestra propuesta sería hibridar una búsqueda local con diferentes inicios, tal como se hace mediante un VNS distribuido, con el sistema de colonias de hormigas. Si observamos el esquema de distribución del VNS planteado en la figura 3.2, añadiéndole el esquema de inicialización planteado en la subsección 3.2.1, podemos pensar en dotar a este esquema de una matriz de feromona $\tau_{i,j}$ en el mismo sentido que tiene en el sistema de colonias de hormigas, de tal forma que las buenas soluciones, óptimos locales, que se obtengan por cada uno de los procesadores involucrados en las búsquedas se refuercen en esta matriz. Una vez reforzadas todas estas soluciones en la matriz de feromona y teniendo en cuenta las hormigas diseñadas en la sección anterior (figuras 3.7, 3.8 y 3.9), podemos utilizar éstas para generar las nuevas soluciones iniciales en la siguiente iteración de las búsquedas distribuidas. El esquema planteado se puede observar en la figura 3.10.

La cuestión principal ahora es cómo manejar la matriz de feromona, esto es, cuándo se realiza el refuerzo/evaporación de la feromona, qué soluciones reforzaran feromona atendiendo a su valor de la métrica utilizada, qué esquema de refuerzo/evaporación de feromona vamos a seguir. A continuación plantearemos la solución seguida por nosotros.

En cuanto a las dos primeras cuestiones planteadas en un principio, podemos pensar actualizar el nivel de feromona cada vez que se alcance un óptimo local en el esquema VNS en función de su valor con respecto a los datos. Sin embargo hemos de notar que, por la forma de proceder del esquema VNS, es muy habitual que alcancemos varias veces el mismo óptimo local, este es el caso cuando mediante un determinado cambio de vecindad no podemos alcanzar otros óptimos locales y seguimos atrapados en el mismo “valle”. Teniendo en cuenta lo que acabamos de comentar, vamos a proponer actualizar el nivel de feromona cuando alcancemos un

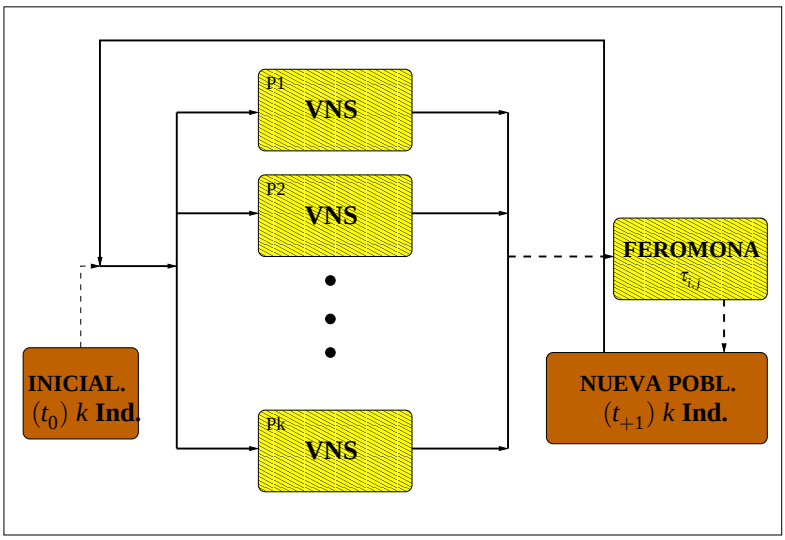


Figura 3.10: Esquema del Algoritmo VNS Distribuido - Búsqueda Híbrida

nuevo óptimo local que mejore el actual, en cada uno de los procesadores, de esta manera recogemos todos los máximos locales nuevos que cada búsqueda distribuida del VNS va recorriendo a lo largo de su ejecución. Así mismo, también, proponemos actualizar el nivel de feromona una vez generada la siguiente población para la iteración posterior con el mejor individuo que tengamos hasta este momento. De esta forma ponderaremos con una mayor fuerza la mejor solución que tengamos para que influya en las generaciones posteriores.

El esquema que proponemos seguir para la actualización de la feromona una vez que hayamos conseguido un nuevo máximo local es el mismo que se propone en el primer sistema de colonias de hormigas propuesto por Dorigo y col. [54]

$$\tau_{i,j}(t) \leftarrow (1-\rho)\cdot\tau_{i,j}(t)+\rho\cdot\Delta\tau_{i,j}(t), \tag{3.14}$$

donde $\Delta\tau_{i,j}(t) = 1/|f(S^+)|$, y $x_i \rightarrow x_j$ es un arco en el grafo que actualiza el nivel de feromona, esto es, el grafo máximo local nuevo, y 0 en otro caso. Como en los anteriores casos $0 \leq \rho < 1$ representa el parámetro de evaporación de la feromona. Hemos de notar que la evaporación de la feromona se efectúa en aquellos arcos que no aparecen en la solución que actualiza la feromona, al poner el valor $\Delta\tau_{i,j}(t)$ igual a 0. Fijémonos que de esta forma aquellos arcos que son comunes a una gran mayoría de soluciones que actualizan feromona tendrán un alto nivel de ésta y por consiguiente tendrán una mayor probabilidad de aparecer en las soluciones iniciales posteriores, como por otra parte buscábamos al principio de la sección. Por último

notar que este esquema es adaptable tanto al problema de aprendizaje de redes de creencia en el espacio de grafos dirigidos acíclicos como en el espacio de secuencias de ordenación entre las variables.

3.5 Experimentación y Conclusiones

En esta sección vamos a presentar la experimentación realizada en el aprendizaje de las mismas bases de datos utilizadas en el capítulo anterior (sección 2.5), excepto la base de datos del dominio ALARM para los 10000 primeros casos. Con respecto a este última base de datos, hemos podido comprobar en los experimentos del capítulo anterior que su comportamiento es muy similar a la base de datos formada por los 3000 primeros casos.

De nuevo resumiremos en forma de tablas los resultados obtenidos, y así podremos analizarlos y extraer de ellos una serie de conclusiones acerca del comportamiento de los diferentes algoritmos utilizados.

3.5.1 Descripción de los experimentos

Para realizar un estudio empírico del comportamiento de los algoritmos propuestos en este capítulo, hemos seguido el mismo procedimiento descrito en la subsección 2.5.1 del capítulo anterior. Esto quiere decir que hemos utilizado los mismos dominios, las mismas bases de datos (excepto ALARM-10000) y hemos extraído los mismos parámetros en las ejecuciones de los algoritmos.

La diferencia estriba en que en este caso la inicialización de los algoritmos se realiza, en el caso de los algoritmos VNS distribuidos, con los procedimientos descritos en la subsección 3.2.1 y en los demás casos el procedimiento se inicializa con la heurística correspondiente.

También, al igual que en el capítulo anterior, se han ejecutado 10 veces los algoritmos para cada una de las bases de datos utilizadas.

3.5.2 Análisis de los resultados

Los resultados de las ejecuciones de los diferentes algoritmos planteados a lo largo de esta capítulo se pueden observar en las diferentes tablas. Al igual que en el capítulo anterior, compararemos los resultados en cada caso, además de comparar estos resultados con los resultados obtenidos en el capítulo anterior, cuando sea posible.

Análisis de los algoritmos VNS distribuidos (DVNS). Modelo de islas. En este caso, hemos realizado experimentos tan sólo con el modelo de islas, por ser éste el

que esperamos nos ofrezca mejores resultados. Para este caso hemos ejecutado el algoritmo distribuido DVNS, tanto en el espacio de grafos dirigidos acíclicos como en el espacio de órdenes, con la topología descrita en las sección 3.2.2. En el espacio de GDAs hemos considerado 4 procesadores y 2 individuos por procesador. En este caso hemos utilizado un esquema VNSST con los siguientes parámetros: $k_{min} = 7$, $k_{step} = 5$ y k_{max} igual al número de variables en el dominio en cada caso. En el espacio de órdenes hemos utilizado 2 procesadores y 2 individuos por procesador. Los parámetros utilizados en el esquema VNSSN son lo mismos que utilizamos en los experimentos del capítulo anterior. En todos los casos hemos considerado un número máximo de iteraciones de 4, utilizando el mismo criterio de parada que para los casos del capítulo anterior.

Análisis para el dominio ALARM - 3000: El algoritmo DVNSST obtiene el mejor individuo encontrado por nosotros $(-14401,29)$ en la gran mayoría de las 10 ejecuciones realizadas en los experimentos. Esto afirmación la demuestra la baja desviación estándar (σ) obtenida en estos experimentos (tabla 3.1). Con respecto al algoritmo VNSST (capítulo 2), hemos de decir que se mejoran los resultados obtenidos en comparación con el algoritmo VNSST con los mismos parámetros empleados en los experimentos del algoritmo distribuido, esto es, $k_{min} = 7$, $k_{step} = 5$ y $k_{max} = 37$ (obsérvese la tabla 2.12). También, prácticamente se igualan los resultados obtenidos por el algoritmo VNSST con los parámetros $k_{min} = 1$, $k_{step} = 1$ y $k_{max} = 37$ en las tres inicializaciones comprobadas (obsérvese la tabla 2.11).

En cuanto a la eficiencia del algoritmo DVNSST empleado en los experimentos, vamos a compararlo con el método VNSST para los parámetros $k_{min} = 1$, $k_{step} = 1$ y $k_{max} = 37$, por ser similares sus resultados. En primer lugar hemos de observar que el algoritmo DVNSST sería del orden de 2 veces más lento [2] que el algoritmo VNSST, si ejecutáramos el primero de forma secuencial. Si suponemos que tenemos un computador MIMD con memoria compartida [82] con 8 procesadores y una arquitectura SMP (symmetric multiprocessors), entonces podríamos tener hasta una aceleración (speedup) teórica de 8, esto es, la velocidad de la ejecución secuencial del algoritmo se vería dividida por un factor de 8, aunque es bien conocido que prácticamente es inalcanzable para la mayoría de tipos de aplicación. De todas formas hemos de notar que para que las dos versiones comparadas aquí se igualaran, prácticamente, en el tiempo de ejecución deberíamos obtener una aceleración del orden de 2. [3] Ciertamente esta aceleración es muy pobre, sobre todo teniendo en cuenta el

[2] Si nos fijamos en la tabla 3.1 y en la tabla 2.11 en el estadístico EstEv y en el estadístico NVars y realizamos la aproximación del tiempo de ejecución por la expresión $\text{EstEv} \times 2^{\text{NVars}}$, entonces tendremos los tiempos aproximados para las versiones secuenciales de ambas aproximaciones. Calculando esta expresión nos resulta que la versión DVNSST, ejecutada de forma secuencial, es del orden de 2 veces más lenta que la versión sin distribuir VNSST.

[3] Ya que, como hemos comentado en la nota anterior, el algoritmo DVNSST es del orden de 2 veces más lento que VNSST si se ejecutara el primero de forma secuencial.

tipo de aplicación que tenemos para estos casos, en donde lo que se hace fundamentalmente es búsqueda e inserción de datos en la memoria de la máquina, lo cual hace que sean fáciles de resolver los problemas de sincronización y coherencia de datos para cada uno de los procesadores. Por todo ello, podemos conjeturar que la versión distribuida empleada en estos experimentos será bastante más rápida que la versión no distribuida empleada en el capítulo anterior. Este análisis de eficiencia también es extensible al espacio de órdenes, aunque en este caso la aceleración teórica es de 4, por consiguiente en este caso no habrá tanta disminución de tiempo como en el anterior caso.

En cuanto al espacio de órdenes (DVNSSN, tabla 3.2) hemos de decir que los resultados son bastantes parecidos al del espacio de estructuras y que mejoran notablemente los resultados obtenidos en la versión no distribuida del capítulo anterior (tabla 2.16) en todas las inicializaciones probadas.

DVNSST	ALARM-3000		
	μ	σ	Mejor
K2	-14401,35	0,18	-14401,29
KL	9,231	0,001	9,231
A	1,90	0,30	2
B	1,00	0,00	1
I	0,00	0,00	0
It	31,60	4,74	
Ind	11,35E06	28,11E05	
EstEv	66908,20	4511,06	
TEst	11,95E06	10,96E05	
NVars	4,59	0,03	

Tabla 3.1: Resultados ALARM 3000 - Algoritmo DVNSST. Modelo de islas 4×2

Análisis para el dominio INSURANCE: Para la búsqueda en el espacio de estructuras (véase la tabla 3.3), el resultado ofrecido por la versión distribuida en el dominio INSURANCE mejora substancialmente el resultado ofrecido por la versión no distribuida (en media para las dos primeras inicializaciones -EMP y K2SN-, obsérvese la tabla 2.15) y en cuanto a la eficiencia podemos destacar que la versión distribuida evalúa una cantidad pequeña más de estadísticos, con lo que siguiendo el análisis efectuado para el dominio ALARM, podemos conjeturar que en este caso la versión distribuida será todavía más eficiente que la no distribuida para este dominio. De todas formas este dato no es sorprendente, ya que en este caso hemos utilizado un esquema VNS con unos parámetros ($k_{min} = 7$, $k_{step} = 5$) que hacen que se realicen un número menor de iteraciones en comparación con el algoritmo VNSST y los

DVNSSN	ALARM-3000		
	μ	σ	Mejor
K2	-14402,29	1,92	-14401,29
KL	9,230	0,002	9,231
A	1,90	0,54	2
B	1,10	0,30	1
I	0,00	0,00	0
It	10,70	7,20	
Ind	40,23E04	15,26E04	
EstEv	81105,20	11644,94	
TEst	10,82E07	34,54E06	
NVars	4,81	0,07	

Tabla 3.2: Resultados ALARM 3000 - Algoritmo DVNSSN. Modelo de islas 2×2

parámetros empleados en el capítulo anterior.

En cuanto al espacio de órdenes (véase la tabla 3.4), hemos de destacar el excelente resultado ofrecido por el algoritmo DVNSSN para este dominio, destacando tanto la media como el mejor individuo encontrado por nosotros. Hemos de recordar que para este dominio la heurística K2SN se comportaba especialmente mal y por consiguiente las inicializaciones de los individuos, para este caso, no son buenos puntos de partidas ya que estas inicializaciones estaban basadas en una versión probabilística de la heurística K2SN (véase la figura 3.4), lo que nos hace pensar que el método propuesto es bastante robusto. Además, por esta misma razón, se ha de realizar un número mayor de iteraciones hasta encontrar el máximo. En cuanto a la eficiencia, este algoritmo sigue la misma tónica que en los casos anteriores, es decir, se evalúan muchos más estadísticos, con lo que será menos eficiente que la búsqueda en el espacio de estructuras, aunque esta versión distribuida será al menos tan rápida como su correspondiente versión no distribuida.

Análisis de los algoritmos DVNS. Hibridación con colonias de hormigas. En este caso nos hemos centrado en el espacio de estructuras para realizar nuestros experimentos. Tanto en la tabla 3.5 como en la tabla 3.6 podemos observar los resultados obtenidos, para los parámetros indicados al pie de cada tabla, tanto en el dominio ALARM como en el dominio INSURANCE. En el dominio ALARM prácticamente se llega al mismo resultado que con el algoritmo DVNSST, se evalúan un número ligeramente menor de estadísticos[4] y se llega en un número menor de

[4] Si bien los parámetros utilizados para las iteraciones del VNS nos lleva a pensar que esto es lógico, pues normalmente se realizarán un número menor de iteraciones, hemos de notar que en el cálculo de

DVNSST	INSURANCE		
	μ	σ	Mejor
K2	-57833,55	22,13	-57773,43
KL	8,450	0,034	8,408
A	6,03	3,83	2
B	9,37	2,11	8
I	5,87	4,98	1
It	20,27	5,92	
Ind	33,57E05	93,79E04	
EstEv	26602,33	2917,77	
TEst	32,62E05	55,77E04	
NVars	4,85	0,06	

Tabla 3.3: Resultados INSURANCE - Algoritmo DVNSST. Modelo de islas 4×2

DVNSSN	INSURANCE		
	μ	σ	Mejor
K2	-57794,14	27,78	-57763,07
KL	8,453	0,033	8,487
A	2,67	1,35	3
B	8,07	0,36	8
I	1,00	1,32	2
It	43,00	22,39	
Ind	10,00E05	69,95E04	
EstEv	85154,90	9201,27	
TEst	17,29E07	82,22E06	
NVars	4,83	0,04	

Tabla 3.4: Resultados INSURANCE - Algoritmo DVNSSN. Modelo de islas 2×2

iteraciones a encontrar el máximo.

En el dominio INSURANCE, se mejoran los resultados ofrecidos por el algoritmo DVNSST. El resto de parámetros significativos prácticamente se mantienen en el mismo orden.

En ambos casos es interesante resaltar que hemos observado que durante el proceso de generación de una nueva población por parte de las hormigas frecuentemente, sobre todo en la primeras iteraciones, se encuentran nuevos óptimos locales que después, normalmente, la iteración correspondiente VNS se encarga de mejorar.

DVNSSTACO	ALARM - 3000		
	μ	σ	Mejor
K2	-14401,35	0,13	-14401,29
KL	9,231	0,000	9,231
A	2,20	0,40	2
B	1,00	0,00	1
I	0,00	0,00	0
It	22,70	5,83	
Ind	30,17E05	24,85E04	
EstEv	41550,20	2211,75	
TEst	63,94E05	50,78E04	
NVars	4,37	0,05	

Tabla 3.5: Resultados ALARM 3000 - Algoritmo DVNSST. Hibridación con colonias de hormigas. 8 procesadores. $(\rho = 0.4, \beta = 2.0, q_0 = 0.8)$. (VNSST $k_{min} = 5, k_{step} = 5, k_{max} = 20$).

Análisis de los Sistemas de Colonias de Hormigas. Estos algoritmos los hemos referenciado como ACO-K2SN para el espacio de órdenes y ACO-B para el espacio de estructuras. Para los experimentos utilizados en este apartado hemos utilizado los siguientes parámetros $\rho = 0.4$, $\beta = 2.0$, $q_0 = 0.8$, 10 hormigas y 100 iteraciones. En el caso del espacio de órdenes no hemos limitado de ningún modo la lista de candidatos, esto es, en cada momento se comprueban todos los nodos restantes no introducidos previamente en la secuencia parcial que tengamos durante su construcción. No hemos realizado unas pruebas sistemáticas para ajustar estos parámetros, sino que hemos elegido éstos por parecer razonables y normalmente del mismo orden que los utilizados para la optimización de otro tipo de problemas [69]. Además

estadísticos están contabilizados los necesitados por las hormigas para la inicialización de la siguiente población.

DVNSSTACO	INSURANCE		
	μ	σ	Mejor
K2	-57824,42	35,42	-57763,07
KL	8,449	0,035	8,487
A	5,70	3,67	3
B	9,48	1,97	8
I	5,67	5,42	2
It	21,67	7,23	
Ind	14,42E05	12,99E04	
EstEv	24471,26	1793,54	
TEst	31,21E05	26,67E04	
NVars	4,73	0,04	

Tabla 3.6: Resultados INSURANCE - Algoritmo DVNSST. Hibridación con colonias de hormigas. 8 procesadores. $(\rho = 0.4, \beta = 2.0, q_0 = 0.8)$. (VNSST $k_{min} = 5, k_{step} = 5, k_{max} = 20$).

hemos utilizado este conjunto de parámetros para los dos dominios, ALARM e INSURANCE.

Debido a la complejidad de la búsqueda, sólo hemos utilizado un procedimiento de búsqueda local, HCST y HCSN respectivamente en el espacio de estructuras y de órdenes, en la última iteración, donde utilizamos las soluciones construidas por todas las hormigas como puntos de inicio de las búsquedas locales. En el caso de la búsqueda local en el espacio de órdenes, no hemos restringido de ningún modo el parámetro *r* (radio).

Además de los parámetros mostrados en las anteriores tablas del resto de experimentos, hemos introducido un nuevo parámetro (K2noHC) en donde podemos comprobar las soluciones aportadas por los sistemas de colonias de hormigas antes de utilizar las búsquedas locales descritas en el párrafo anterior.

Por último hemos de decir que hemos realizado algunas pruebas para el SHC en el espacio de órdenes hibridando éste con el espacio de grafos dirigídos acíclicos, como describíamos al final de la subsección 3.2.2, pero los resultados obtenidos no son nada buenos. Es por ello que no los hemos reflejado en tablas durante esta sección.

Análisis para el dominio ALARM-3000: En ambos casos, tanto en el espacio de estructuras (véase la tabla 3.8), como en el de órdenes (véase la tabla 3.7), se llega a una buena solución, sin embargo son levemente peores que las obtenidas por el resto de los algoritmos de este capítulo.

Centrándonos en el espacio de estructuras, hemos de notar que la aportación de la búsqueda local en la última iteración mejora ligeramente los resultados y no en todos los casos, esto último lo pone de manifiesto el número de iteraciones (parámetro it, en media) hasta alcanzar el óptimo, el cual es de 87,1, menor que el número de iteraciones realizadas, 100. También es de destacar el número de estadísticos evaluados, por ser éstos menores que para el algoritmo DVNSST y prácticamente similares para el caso del algoritmo DVNSSTACO, esto quiere decir que en este caso se reutilizan de una forma más adecuada los cálculos realizados en etapas anteriores del algoritmo, al igual que en el algoritmo DVNSSTACO.

En cuanto al espacio de órdenes, hemos de destacar la gran aportación de la búsqueda local en el proceso de búsqueda, obsérvense los parámetros K2noHC e it en la tabla 3.7, ya que el resultado antes de iniciar el proceso local de búsqueda es bastante pobre y siempre se obtiene el óptimo encontrado en la última iteración. Si bien lo comentado anteriormente es cierto, hemos de notar que la aportación del SCH, en este caso, es la de encontrar buenos puntos de inicio para una búsqueda local posterior. El número de estadísticos evaluados en este caso, contabilizando los efectuados por las búsquedas locales, es bastante parecido al número de éstos en el algoritmo DVNSSN.

SCH - Órdenes	ALARM - 3000		
	μ	σ	Mejor
K2	-14404,07	3,28	-14401,29
KL	9,228	0,003	9,231
A	2,20	0,60	2
B	1,40	0,49	1
I	0,00	0,00	0
It	100,00	0,00	
K2noHC	-14462,46	14,09	
EstEv	89154,80	8099,94	
TEst	58,88E06	51,65E05	
NVars	4,97	0,08	

Tabla 3.7: Resultados ALARM 3000 - Algoritmo ACO-K2SN.

Análisis para el dominio INSURANCE: En el espacio de órdenes (véase la tabla 3.9) hemos de notar que de nuevo la aportación de la búsqueda local es fundamental para mejorar los resultados aportados por las hormigas. Si bien el resultado para las métricas K2 y KL son un poco peores que para el resto de los casos, hemos de notar que en cuanto al número de arcos mal colocados con respecto a la red original tiene un comportamiento bastante aceptable.

SCH - Estructura	ALARM - 3000		
	μ	σ	Mejor
K2	-14406,06	4,54	-14401,29
KL	9,231	0,003	9,231
A	3,30	1,79	2
B	1,10	0,30	1
I	1,70	1,90	0
It	87,10	20,68	
K2noHC	-14407,32	5,01	
EstEv	43349,60	953,87	
TEst	64,76E05	11,67E04	
NVars	4,43	0,02	

Tabla 3.8: Resultados ALARM-3000 - Algoritmo ACO-B.

En cuanto al espacio de estructuras (véase la tabla 3.10) hemos de comentar que, si exceptuamos el comportamiento del algoritmo DVNSSN (espacio de órdenes), este algoritmo nos ofrece los mejores resultados para este dominio. De nuevo se pone de manifiesto la leve aportación de la búsqueda local en este caso. El número de estadísticos evaluados es del orden de los calculados para los casos anteriores para el mismo dominio y el mismo espacio de búsqueda.

SCH - Órdenes	INSURANCE		
	μ	σ	Mejor
K2	-57849,40	36,46	-57772,65
KL	8,444	0,036	8,485
A	4,57	2,64	2
B	9,60	1,56	8
I	2,80	2,81	1
It	100,00	0,00	
K2noHC	-58420,39	138,12	
EstEv	74169,53	3769,06	
TEst	37,27E06	27,68E05	
NVars	4,89	0,03	

Tabla 3.9: Resultados INSURANCE - Algoritmo ACO-K2SN.

SCH - Estructura	INSURANCE		
	μ	σ	Mejor
K2	-57807,81	36,68	-57763,07
KL	8,450	0,036	8,487
A	3,60	1,99	3
B	8,50	1,34	8
I	2,43	2,09	2
It	80,20	21,57	
K2noHC	-57811,04	37,27	
EstEv	28738,77	882,93	
TEst	38,19E05	33,44E03	
NVars	4,35	0,02	

Tabla 3.10: Resultados INSURANCE - Algoritmo ACO-B.

Capítulo 4

Aprendizaje de Relaciones Temporales en Redes de Creencia Dinámicas

4.1 Introducción

Nuestro objetivo en este capítulo es aprender de forma automática a partir de datos el conjunto $E^{tmp}(k)$ de relaciones temporales de una red de creencia dinámica (RCD), supuesto que tenemos conocidas las estructuras para cada periodo de tiempo k, $G(k)$.

En primer lugar hemos de plantear el problema que queremos resolver. Suponemos que tenemos bien definidas las estructuras en los periodos de tiempo $G(k)$, entonces tan solo nos planteamos aprender la estructura correspondiente a $G(k,k+1)$, para ello nos centramos en aprender el conjunto de arcos temporales $E^{tmp}(k+1)$.

Resolver este problema es interesante por varias razones:

- Podemos tener conocidas las estructuras correspondientes a los periodos de tiempo k elicitadas por un experto. Hemos notado que a menudo en aplicaciones reales las estructuras correspondientes a $G(k)$ de una RCD tienen muy pocas relaciones, enlaces no temporales, incluso ninguna relación. Por consiguiente, estas estructuras pueden ser fáciles de obtener a priori.

- Podemos utilizar un algoritmo de aprendizaje automático de redes de creencia estáticas para aprender la estructura que modela cada periodo de tiempo k y a partir de esta salida aprender el modelo evolutivo con los algoritmos que desarrollaremos en este capítulo. Esta forma de proceder la plantearemos en el siguiente capítulo.

En esta sección vamos a estudiar diferentes algoritmos para aprender a partir de bases de datos de casos el modelo de transición de una red de creencia dinámica, para ello dividiremos estos algoritmos en algoritmos basados en métricas más una búsqueda de optimización y algoritmos basados en tests de independencia condicional.

4.2 Definición formal del problema del aprendizaje de relaciones temporales

Definiremos de una manera más formal el problema que pretendemos resolver a lo largo de este capítulo. Suponemos que queremos aprender una red de creencia dinámica markoviana y estacionaria y que para ello tenemos una base de datos de casos:

$$D = \{\mathbf{v}(0)^1, \ldots, \mathbf{v}(0)^m, \mathbf{v}(1)^1, \ldots, \mathbf{v}(1)^m, \ldots, \mathbf{v}(n-1)^1, \ldots, \mathbf{v}(n-1)^m\}$$

de m casos, donde $\mathbf{v}(k)^i$ representa una instanciación de las variables de $V(k)$ en el caso i-ésimo de la base de datos D. También vamos a suponer que conocemos las estructuras $G(k)$ para todos los instantes de tiempo, esto es, desde $k = 0$ hasta $k = n-1$. Entonces el problema que queremos resolver es encontrar de forma automática el conjunto de relaciones temporales $E^{tmp}(k)$. Dadas las premisas de las cuales partimos para resolver nuestro problema, nos podremos centrar en aprender de forma automática el conjunto de relaciones temporales $E^{tmp}(k)$ para cualquiera de las parejas de periodos de tiempo consecutivos que se pueden formar en nuestra base de datos. Esto quiere decir que nosotros elegiremos a priori dos periodos de tiempo consecutivos cualesquiera k y $k+1$ y restringiremos (proyectaremos) nuestras observaciones en D a las instanciaciones concretas de estos dos periodos de tiempo consecutivos, esto es, para $D' = \{\mathbf{v}(k)^1, \ldots, \mathbf{v}(k)^m, \mathbf{v}(k+1)^1, \ldots, \mathbf{v}(k+1)^m\}$. En definitiva, nos centraremos en aprender el modelo de transición de una red de creencia dinámica.

Si el modelo de red de creencia dinámica que queremos aprender es estacionario y markoviano, en el sentido expresado en el capítulo 1, hemos de notar que se puede demostrar la siguiente proposición:

Proposición 4.1 *Sea una RCD markoviana y estacionaria, y sean dos periodos de tiempo consecutivos cualesquiera $G(k,k+1)$, entonces si se cumple para algún subconjunto $S \in V$ que $I(x_j(k), x_i(k+1)|S)$, entonces podremos encontrar un subconjunto S' tal que se siga cumpliendo $I(x_j(k), x_i(k+1)|S')$ y $S' \subseteq V(k,k+1)$.*
Demostración: Consecuencia directa de que la RCD es markoviana, esto es, $I(V(k-1), V(k+1)|V(k))$.

Por otra parte, como hemos supuesto que la RCD es estacionaria y que se repiten las estructuras $G(k,k+1)$ para todo k, podemos estar seguros de que si encontramos una relación de independencia $I(x_j(k),x_i(k+1)|S')$ y $S' \subseteq V(k,k+1)$, entonces ésta se mantendrá en todos los intervalos de tiempo. A partir de estas consideraciones, podemos realizar el aprendizaje del modelo de transición de una RCD tan solo centrándonos en cualesquiera dos periodos de tiempo consecutivos k y $k+1$. Esto también implicará de una forma directa que todos los nodos $x_i(k)$ tendrán el mismo conjunto de padres en todos los periodos de tiempo k.

4.3 Algoritmos locales basados en optimización de métricas

En esta sección estudiaremos algoritmos que se fundamentan principalmente en la búsqueda local de la estructura del modelo de transición de una red de creencia dinámica basados en la optimización de una medida de bondad en el ajuste o métrica. Para ello, hemos de suponer que tenemos definida una métrica o función $f(G(k,k+1) : D')$. Estos algoritmos fundamentalmente consistirán en buscar el mejor conjunto de padres para cada nodo $x_i(k+1)$, conocido el conjunto de padres en su propio intervalo de tiempo $\pi_{k+1}(x_i(k+1))$ y atendiendo a la métrica elegida para el aprendizaje.

En primer lugar plantearemos un algoritmo, basado en el algoritmo K2, en donde la métrica elegida sea una función descomponible. Además estudiaremos otro algoritmo que se fundamenta en otra métrica basada en las medidas de discrepancia entre las relaciones de independencia condicional expresadas en la red candidata y éstas mismas, medidas en nuestra base de datos de casos; para ello se utiliza la medida de entropía de Kullback-Leibler. Este último algoritmo será una adaptación del algoritmo BENEDICT para resolver el problema del aprendizaje de relaciones temporales.

Estos algoritmos utilizarán como método de búsqueda una heurística local greedy (hill-climbing) para plantearse en cada momento el arco que al introducirlo en la actual red mejora la medida que en cada caso se utiliza.

4.3.1 Algoritmo TeReK2

Este algoritmo se basa en la optimización de una medida descomponible como criterio de bondad en el ajuste. La idea principal en que se basa es en la búsqueda de la estructura del modelo evolutivo de la red de creencia dinámica que mejor concuerde con una base de datos de casos, para ello utilizaremos una búsqueda local en el espacio de los posibles arcos temporales del modelo.

Si tenemos una media descomponible $f(G(k,k+1):D')$ y si tenemos especificado un orden causal entre las variables del modelo, al igual que describimos en el capítulo 2, entonces el problema de búsqueda se podría definir como: Encontrar $\pi_k(x_i(k+1))^*$, tal que sea igual a:

$$\arg\max_{\pi_k(x_i(k+1))} f\left(x_i(k+1), \left[\pi_{k+1}(x_i(k+1)) \cup \pi_k(x_i(k+1))\right] : D'\right) \tag{4.1}$$

para todo $x_i(k+1) \in V(k+1)$. Esto quiere decir que el problema consistirá en encontrar el mejor conjunto de padres $\pi_k(x_i(k+1))^*$ en el periodo de tiempo anterior k, para cada nodo $x_i(k+1)$. En primer lugar hemos de notar que el orden causal entre las variables es conocido en nuestro problema, ya que por una parte tenemos especificada la estructura $G(k)$, y que por consiguiente se puede extraer un orden topológico entre las variables de $V(k)$, y por otro lado conocemos que todas las variables $x_j(k) \in V(k)$ son menores que cualquier variable de $V(k+1)$ debido a su orden temporal natural. Por tanto es posible encontrar los posibles padres en k, relaciones temporales, para cada variable $x_i(k+1)$ de forma independiente, esto es, podremos maximizar cada familia por separado, ya que el encontrar un padre en k para una determinada variable $x_i(k+1)$ no introduce ninguna restricción para la búsqueda del resto de conjunto de padres de cada variable en $V(k)$.

Nosotros vamos a abordar esta búsqueda desde el punto de vista heurístico, utilizando para ello una búsqueda local. Para realizar esta búsqueda local es necesario especificar la vecindad utilizada, esto es, definir la vecindad del conjunto de padres de una determinada variable $x_i(k+1)$. Esta definición es bastante sencilla. Será definida por la posible introducción de un nodo de $V(k)$ como padre de $x_i(k+1)$ que previamente no haya sido introducido, esto es:

$$\mathcal{N}(\pi_k(x_i(k+1))) = \left\{ \pi_k(x_i(k+1)) \cup \{x_j(k)\},\ \ \forall x_j(k) \in V(k) \setminus \pi_k(x_i(k+1)) \right\} \tag{4.2}$$

Teniendo en cuenta las definiciones anteriores el algoritmo de búsqueda planteado procederá como sigue: En primer lugar tomará de partida los padres de un nodo $x_i(k+1)$ en su propio intervalo de tiempo, esto es, $\pi_{k+1}(x_i(k+1))$. En cada paso del algoritmo se añade el padre en $V(k)$ cuya inclusión incremente más la medida utilizada de la estructura resultante. Cuando la adición de un padre no pueda incrementar esta medida, se dejan de añadir nodos al conjunto de padres.

Las entradas al algoritmo son: n el cardinal del conjunto de nodos de $V(k+1)$ y una base de datos D' que contiene m casos, proyección de D sobre los periodos de tiempo k y $k+1$. Como salida obtendremos el conjunto de padres de cada nodo $x_i(k+1)$.

En la figura 4.1 podemos observar el pseudocódigo correspondiente al algoritmo que hemos descrito en el párrafo anterior.

Algoritmo TeReK2

1. *Para* $i = 1$ *hasta n hacer*
   (a) $\pi(x_i(k+1)) = \pi_{k+1}(x_i(k+1))$
   (b) $Ok = True$
   (c) $P_{old} = f(x_i(k+1), \pi(x_i(k+1) : D')$
   (d) *Mientras Ok hacer*
      i. $x_j(k) = \arg \max_{x_h(k) \notin \pi(x_i(k+1))} f\left(x_i(k+1), \left[\pi(x_i(k+1)) \cup \{x_h(k)\}\right] : D'\right)$
      ii. $P_{new} = f\left(x_i(k+1), \left[\pi(x_i(k+1)) \cup \{x_j(k)\}\right] : D'\right)$
      iii. *Si* $P_{new} > P_{old}$ *Entonces*
      $P_{old} = P_{new}$
      $\pi(x_i(k+1)) = \pi(x_i(k+1)) \cup \{x_j(k)\}$
      iv. *En caso contrario* $Ok = false$

Figura 4.1: Algoritmo TeReK2

4.3.2 Algoritmo TeReBenedict

Este algoritmo se basa en el algoritmo BENEDICT que describíamos en el capítulo 1. En concreto se basará en la versión denominada BENEDICT-step. Este algoritmo utiliza una heurística de búsqueda similar en muchos aspectos a la heurística utilizada en el algoritmo K2. En ambos se presupone un orden causal entre las variables y también ambos optimizan siguiendo este orden el conjunto de padres para cada variable del problema en cuestión. Las diferencias fundamentales entre ambos son: Por una lado el algoritmo BENEDICT utiliza tests de independencia condicional, sin coste computacional extra, para limitar el espacio de búsqueda en cada paso del algoritmo. Y por otro lado las métricas en que se fundamentan ambas búsquedas son diferentes. En general, la métrica en la que se basa el algoritmo BENEDICT no es una medida descomponible.

BENEDICT se basa en una medida de discrepancia entre las independencias condicionales que expresan la estructura gráfica de una red de creencia y éstas mismas estimadas a través de la base de datos de casos.

El algoritmo busca una red de creencia candidata tal que la medida de discrepancia definida sea mínima, cuando alcancemos este mínimo entonces ofrece como resultado la red de creencia candidata con la que se ha alcanzado este mínimo. Para

medir la discrepancia entre la red candidata y la base de datos de casos se utiliza la medida de entropía de KL $D(X,Y|Z)$.

$$D(X,Y|Z) = \sum_{x,y,z} P(x,y,z) \log \frac{P(x,y|z)}{P(x|z)P(y|z)}$$

La discrepancia total entre el grafo de la red candidata y la base de datos de casos se mide entonces por:

$$f(G:D) = \sum_{x_i <_L x_j \ / \ x_i \notin \pi(x_j)} D(x_i, x_j | S_d(x_i, x_j))$$

donde $<_L$ es un orden entre las variables previamente establecido [1] y $S_d(x_i,x_j)$ es el subconjunto mínimo d-separador para las variables x_i y x_j en el grafo de la red candidata.

Una vez recordado el funcionamiento del algoritmo BENEDICT, podemos ver que la adaptación para el aprendizaje de relaciones temporales en una red de creencia dinámica es directa. Para su adaptación hemos de notar, como en los algoritmos anteriores, que en el caso que nos ocupa las variables sobre las que vamos a trabajar están ordenadas de forma natural debido a su orden temporal, es decir, $x_i(k) \in V(k) < x_j(k+1) \in V(k+1)$, además de poder obtener un orden ancestral entre los propios nodos de $V(k)$. Entonces el algoritmo procedería de la siguiente forma:

Inicialmente partiremos de un grafo $G(k,k+1)$ sin ninguna relación temporal, esto es, $E^{tmp}(k+1) = \emptyset$. Iniciamos el proceso con el nodo $x_1(k+1)$, primero en el orden topológico de $V(k+1)$ y fijamos los posibles enlaces temporales candidatos hacia este nodo $\{x_i(k), x_1(k+1)\} \forall x_i(k) \in V(k)$, posteriormente medimos $f(G(k,k+1):D')$ sin introducir ninguna relación temporal y tomamos esta medida como mínimo actual. Una vez realizada esta inicialización iremos introduciendo arcos del conjunto de arcos temporales candidatos que minimizan la medida f en cada momento. Al igual que describe Acid [2] se pueden ir realizando tests de independencia condicional conforme se vaya calculando la medida con la inclusión de los arcos candidatos, de tal forma que si estimamos independientes dos variables $x_i(k)$ y $x_j(k+1)$ no volveremos a plantearnos la inclusión de este arco temporal en etapas posteriores del algoritmo. El algoritmo queda descrito en la figura 4.2.

Hemos de notar que al seguir el orden ancestral definido por la estructura $G(k+1)$, conocemos por una proposición [2] que los conjuntos mínimos d-separadores se encuentran entre la unión de los conjuntos ancestrales de los nodos comprobados. Por esta razón, si nosotros estamos buscando los padres de un determinado nodo

[1] El algoritmo BENEDICT puede trabajar, con un poco más de complejidad, sin establecer un orden previo entre las variables, sin embargo para nuestra adaptación para el aprendizaje de relaciones temporales no es necesario.

Algoritmo TeReBenedict

Se fija $G = G(k, k+1)$ *inicialmente con* $E^{tmp}(k+1) = \emptyset$.

Se fija un orden topológico para las variables de $V(k+1)$, *empezando por los nodos raíces.*

Para $i = 1$ *hasta n hacer*

1. *Se fija* $L = \{x_j(k) \rightarrow x_i(k+1)\}\ \forall x_j(k) \in V(k)$
2. $f = 0$
3. *Para cada enlace de L hacer*
   (a) $S_d(x_j, x_i) =$ *Conj. Min. d-separador en G*
   (b) $f = f + D(x_j, x_i | S_d(x_j, x_i))$
   (c) *Si* $I(x_j, x_i | S_d(x_j, x_i))$ *Entonces* $L = L \setminus \{x_j \rightarrow x_i\}$
4. $Min = f$
5. *Minimizado = true*
6. *Mientras* $L \neq \emptyset$ *y Minimizado hacer*
   (a) *Minimizado = false*
   (b) *Para cada enlace de L* ($x_j \rightarrow x_i$) *hacer*
      i. $G' = G \cup \{x_j \rightarrow x_i\}$
      ii. *f=0*
      iii. *Para cada nodo* $x_j(k) \in V(k) \setminus \pi_{G'}(x_i)$ *hacer*
         A. $S_d(x_j, x_i) =$ *Conj. Min. d-separador en* G'
         B. $f = f + D(x_j, x_i | S_d(x_j, x_i))$
         C. *Si* $I(x_j, x_i | S_d(x_j, x_i))$ *Entonces* $L = L \setminus \{x_j \rightarrow x_i\}$
      iv. *Si* $f < Min$ *Entonces*
         $Min = f$
         Minimizado = true
         Enlace = $\{x_j \rightarrow x_i\}$
   (c) $G = G \cup Enlace$
   (d) $L = L \setminus Enlace$

Figura 4.2: Algoritmo TeReBenedict

$x_i(k+1)$ y suponiendo que el algoritmo hasta esta etapa ha sido capaz de extraer el conjunto de padres de cada nodo anterior en el orden a $x_i(k+1)$ de forma correcta, entonces podemos estar seguros que los conjuntos d-separadores minimales serán calculados correctamente.

Otra puntualización del anterior algoritmo es la siguiente: En el algoritmo BENEDICT para calcular la medida f de una red candidata, se han de sumar todos y cada uno de los asertos de d-separación encontrados hasta la etapa *i*-ésima que implican al nodo x_i en la medida global de discrepancia utilizada, sin embargo en nuestra adaptación para el aprendizaje de relaciones temporales que hemos propuesto, sólo sumamos los asertos de d-separación entre parejas de nodos $x_j(k)$ y $x_i(k+1)$. Este hecho se justifica porque los asertos de d-separación entre variables del propio periodo de tiempo $\langle x_i(k+1), x_j(k+1)|S\rangle$, dado algún subconjunto S, son conocidos que realmente son condicionalmente independientes y que por tanto siempre sumaríamos una cantidad constante 0 en la sumatoria global de la medida f.

Por último indicar que el paso 3 del algoritmo descrito en la figura 4.2 también es diferente en el algoritmo BENEDICT-step. En este último caso la inicialización de la medida f se efectúa con la sumatoria de todos los asertos de d-separación $\langle x, x_i|\emptyset\rangle$, esto es porque el nuevo nodo x_i a insertar en la estructura en un principio no tiene ningún enlace que lo conecte con los nodos predecesores en el orden. Sin embargo, en nuestra adaptación hemos de notar que esto no es así, pueden existir caminos que unan el nodo correspondiente $x_i(k+1)$ con nodos $x_j(k)$ mediante enlaces temporales previamente introducidos en etapas anteriores, es por ello que en esta adaptación se han de calcular los conjuntos mínimos d-separadores para la inicialización de la medida f.

4.4 Algoritmos basados en relaciones de independencia condicional

En esta sección nos vamos a plantear una serie de algoritmos de aprendizaje que utilizan tests de independencia condicional para la inclusión o borrado de los arcos que conforman el modelo evolutivo de la red de creencia dinámica que queremos aprender. Para cada uno de estos algoritmos demostraremos su correcto funcionamiento y tendrán en común que para ello partiremos de la suposición de que los datos en nuestra base de datos son datos obtenidos a partir de un modelo de independencias isomorfo a un grafo dirigido acíclico que representa a una red de creencia dinámica.

4.4.1 Algoritmo TeRePC

El algoritmo que vamos a presentar a continuación está inspirado en su forma de proceder y en sus fundamentos teóricos en el algoritmo PC. Como vimos en el capítulo 1, el algoritmo PC es una mejora del algoritmo SGS, ambos desarrollados por Spirtes, Glymour y Scheines [127].

El algoritmo PC, en el fondo, se fundamenta en la búsqueda de posibles separadores de una pareja de nodos que los haga independientes para de esta forma poder asegurar que en la salida final estos dos nodos no serán adyacentes. La forma de búsqueda de estos posibles separadores, se hace de tal forma que los tests de independencia condicional se realicen en orden creciente atendiendo a su complejidad. La siguiente proposición nos asegurará el correcto funcionamiento del algoritmo que expondremos posteriormente y que denominaremos **TeRePC**.

Proposición 4.2 *Si consideramos $G(k,k+1)$ como la unión de los dos grafos dirigidos acíclicos en dos periodos de tiempo consecutivos en una RCD, podremos asegurar que $x(k) \in V(k)$ e $y(k+1) \in V(k+1)$, no son adyacentes si y solo si:*

$$\langle x(k), y(k+1) | \pi_k(y(k+1)) \cup \pi_{k+1}(y(k+1)) \rangle \tag{4.3}$$

Demostración: La demostración es clara, teniendo en cuenta que el nodo $y(k+1)$ no puede ser un ancestro de $x(k)$ en $G(k,k+1)$.

Partiendo de esto y siguiendo el proceso del algoritmo PC, podremos dar un algoritmo (fig 4.3) que recupere el conjunto $E^{tmp}(k+1)$ de una red de creencia dinámica. El algoritmo comenzará con un grafo G unión de $G(k)$ y $G(k+1)$ en donde para cada variable o vértice de $G(k+1)$ conectaremos un arco desde todos los nodos de $G(k)$. Procederemos, al igual que PC, de forma que realizamos primero los tests de independencia condicional de un orden menor, empezando con los de orden cero, y así sucesivamente.

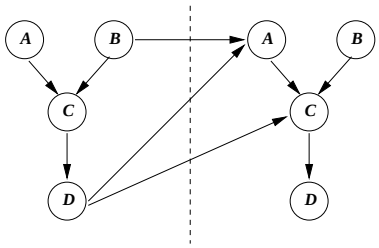
Ejemplo 4.1 *Veremos un pequeño ejemplo del funcionamiento del algoritmo TeRePC. Tenemos el grafo inicial de la figura 4.4 (a).*

- $n = 0$.
  - $I(A(k), B(k+1)|\emptyset) \Rightarrow$ *Eliminamos* $A(k) \rightarrow B(k+1)$.
  - $I(B(k), B(k+1)|\emptyset) \Rightarrow$ *Elim.* $B(k) \rightarrow B(k+1)$.
  - $I(C(k), B(k+1)|\emptyset) \Rightarrow$ *Elim.* $C(k) \rightarrow B(k+1)$.
  - $I(D(k), B(k+1)|\emptyset) \Rightarrow$ *Elim.* $D(k) \rightarrow B(k+1)$.
- $n = 1$.
  - $I(A(k), D(k+1)|C(k+1)) \Rightarrow$ *Elim.* $A(k) \rightarrow D(k+1)$.

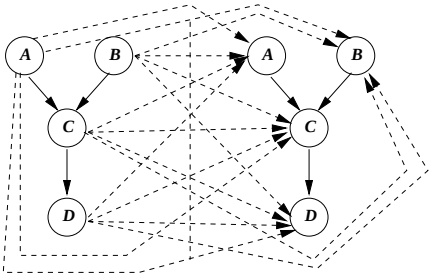
Algoritmo TeRePC

1. *Formar el grafo* $G = G(k) \cup G(k+1) \cup E^{tmp}(k+1)$.
2. $E^{tmp}(k+1) = \{(x(k), y(k+1)) / (x(k), y(k+1)) \in V(k) \times V(k+1)\}$.
3. $n = 0$
4. *Repetir*
   (a) *Repetir*
   Seleccionar un par ordenado de variables $x(k)$ *e* $y(k+1)$ *adyacentes en* G *tal que* $\pi(y(k+1)) \setminus \{x(k)\}$ *tenga un cardinal mayor o igual que n, y seleccionar un subconjunto S de* $\pi(y(k+1)) \setminus \{x(k)\}$ *de cardinalidad n. Si* $I(x(k), y(k+1)|S)$ *eliminar el arco* $x(k) \rightarrow y(k+1)$ *de G.*
   Hasta que todos los pares de variables adyacentes $x(k)$ *e* $y(k+1)$ *tales que* $\pi(y(k+1)) \setminus \{x(k)\}$ *tengan cardinalidad mayor o igual que n y todos los subconjuntos S de* $\pi(y(k+1)) \setminus \{x(k)\}$ *de cardinalidad n hayan sido comprobados para establecer la independencia.*
   (b) $n = n+1$
5. *Hasta que el conjunto* $\pi(y(k+1)) \setminus \{x(k)\}$ *tenga cardinalidad menor que n, para cada par de vértices adyacentes* $(x(k), y(k+1))$.

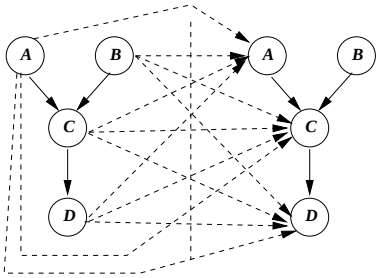
Figura 4.3: Algoritmo TeRePC



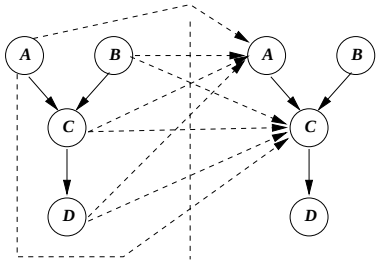
(a) Red de Creencia Dinamica de partida.



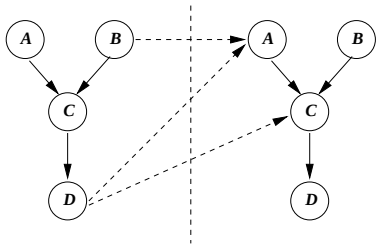
(b) Grafo de partida para el Algoritmo TeRePC.



(c) Grafo resultante para n=0.



(d) Grafo resultante para n=1



(e) Resultado Final.

Figura 4.4: Ejemplo de funcionamiento del Algoritmo TeRePC.

- $I(B(k), D(k+1)|C(k+1)) \Rightarrow$ *Elim.* $B(k) \rightarrow D(k+1)$.
- $I(C(k), D(k+1)|C(k+1)) \Rightarrow$ *Elim.* $C(k) \rightarrow D(k+1)$.
- $I(D(k), D(k+1)|C(k+1)) \Rightarrow$ *Elim.* $D(k) \rightarrow D(k+1)$.

El resultado del algoritmo para $n = 0$ *y para* $n = 1$ *lo podemos observar en la figura 4.4(c) y (d).*

- $n = 2$.
  - $I(A(k), A(k+1)|D(k), B(k)) \Rightarrow$ *Elim.* $A(k) \rightarrow A(k+1)$.
  - $I(A(k), C(k+1)|\{D(k), A(k+1))\}) \Rightarrow$ *Elim.* $A(k) \rightarrow C(k+1)$.
  - $I(B(k), C(k+1)|\{D(k), A(k+1))\}) \Rightarrow$ *Elim.* $B(k) \rightarrow C(k+1)$.
  - $I(C(k), C(k+1)|\{D(k), A(k+1))\}) \Rightarrow$ *Elim.* $C(k) \rightarrow C(k+1)$.
  - $I(C(k), A(k+1)|\{B(k), D(k)\}) \Rightarrow$ *Elim.* $C(k) \rightarrow A(k+1)$.

  El resultado del algoritmo para $n = 2$ *lo podemos observar en la figura 4.4(e). Cuando* $n = 3$ *el algoritmo se detiene.*

■

4.4.2 Algoritmo de aprendizaje de relaciones temporales basado en el subconjunto Manto de Markov parcial

En este punto vamos a estudiar otro algoritmo para aprender de forma automática las relaciones temporales de una red de creencia dinámica, que se basa en la realización de tests de independencia condicional únicos entre variables de periodos de tiempo consecutivos.

La idea básica del algoritmo es realizar test de independencia condicional entre los nodos de $V(k)$ y $V(k+1)$, estos tests se realizan en el orden ancestral de los nodos marcado por sus estructuras $G(k)$ y $G(k+1)$. Los tests que se realizan son únicos para cada pareja de nodos.

Es bien conocido en la literatura sobre redes de creencia que el conjunto Manto de Markov (MM) de un nodo x es el conjunto formado por la unión de sus padres, hijos y padres de estos hijos. Ahora definiremos lo que nosotros entendemos como el conjunto Manto de Markov restringido para un periodo de tiempo k.

Definición 4.1 *Dada una RCD y un nodo* $x_i(k) \in V(k)$*, definiremos el Manto de Markov restringido al periodo de tiempo k, y lo notaremos como* $MM_k(x_i(k))$*, como:*

$$MM_k(x_i(k)) = \{\pi_k(x_i(k)) \cup hij_k(x_i(k)) \cup \pi_k(hij_k(x_i(k)))$$

Este subconjunto del conjunto Manto de Markov es el correspondiente a la unión de los padres de $x_i(k)$ restringidos a su propio intervalo de tiempo, los hijos de esta misma variable también restringidos a su propio intervalo de tiempo y los padres de éstos últimos nodos también restringido a su propio intervalo de tiempo k.

En segundo lugar definiremos lo que entendemos como el conjunto Manto de Markov del nodo i-ésimo $x_i(k) \in V(k)$, según un orden ancestral para $G(k)$, asociado al nodo j-ésimo $x_j(k+1)$, según un orden ancestral para $G(k+1)$, de una red de creencia dinámica. A este conjunto lo vamos a notar como: $MM^{i,j}$.

Definición 4.2 *(Manto de Markov i,j) Definido un orden ancestral para $G(k)$ y también para $G(k+1)$, entonces:*

$$MM^{i,j} = \{MM_k(x_i(k)) \cup H^{i,j} \cup PH^{i,j}\}$$

donde $H^{i,j} = \{x_h(k+1) \in V(k+1) | (x_i(k), x_h(k+1)) \in E^{tmp}(k+1)$ *y* $x_h(k+1) < x_j(k+1)\}$ *y* $PH^{i,j} = \{x_p \in \pi(x_h(k+1)) | x_h(k+1) \in H^{i,j}\}$

En la siguiente figura 4.5, podemos ver un ejemplo de los nodos que pertenecen al subconjunto $MM^{i,j}$ para un nodo.

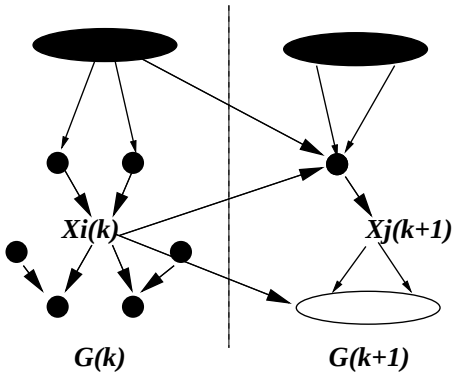


Figura 4.5: Ejemplo de Manto de Markov i,j. Los nodos sombreados pertenecen a este subconjunto

El subconjunto $H^{i,j}$ son los hijos del nodo $x_i(k)$ en el periodo de tiempo siguiente $k+1$ que sean menores en el orden al nodo $x_j(k+1)$ y el subconjunto $PH^{i,j}$ son los padres en ambos periodos de tiempo de los nodos del subconjunto anterior.

Es bien conocido que en cualquier red de creencia, todo nodo x de la red es condicionalmente independiente del resto de nodos de la red dado el subconjunto Manto de Markov. Si nos fijamos de nuevo en la figura 4.5, podemos notar que nosotros no hemos incluido en nuestra definición de Manto de Markov parcial i,j

aquellos nodos que, perteneciendo al Manto de Markov total, conocemos que en los caminos entre $x_i(k)$ y $x_j(k+1)$ que incluyen estos nodos excluidos se forma al menos un nodo cabeza-cabeza; es por ello que para realizar un hipotético tests de independencia condicional no sea necesario su inclusión.

A continuación estableceremos un resultado, que nos da una condición necesaria y suficiente para poder determinar si dos nodos $x(k)$ e $y(k+1)$ son adyacentes o no en una red de creencia dinámica.

Proposición 4.3 *Sea G un grafo dirigido acíclico que define una RCD. Para cualesquiera dos periodos de tiempo k y* $k+1$ *consecutivos en G, tenemos que:*

$$\exists\, x_i(k) \rightarrow x_j(k+1) \Longleftrightarrow \neg\langle x_i(k), x_j(k+1) | MM^{i,j}\rangle$$

donde $x_i(k)$ *es el nodo i-ésimo de* $V(k)$ *según un orden ancestral para* $G(k)$ *y* $x_j(k+1)$ *es el nodo j-ésimo de* $V(k+1)$ *según un orden ancestral para* $G(k+1)$.

Demostración: La condición necesaria es directa teniendo en cuenta que si los nodos son adyacentes en G entonces ningún subconjunto es capaz de d-separarlos, en particular, el conjunto $MM^{i,j}$ tampoco.

Demostraremos, ahora la condición suficiente: Si se da $\neg\langle x_i(k), x_j(k+1) | MM^{i,j}\rangle$, esto quiere decir que existe al menos un camino activo entre ambos nodos dado el subconjunto $MM^{i,j}$. Supongamos que este camino no es un arco directo entre los nodos $x_i(k)$ y $x_j(k+1)$.

Si el camino C activo es de la forma:

$$x_i(k) \leftarrow x_s(k) \ldots x_j(k+1)$$

entonces el camino queda bloqueado al pertenecer el nodo $x_s(k)$ al conjunto $MM^{i,j}$ y no ser un nodo cabeza-cabeza.

Si el camino C activo es de la forma:

$$x_i(k) \rightarrow x_s(k) \rightarrow \ldots x_j(k+1)$$

entonces el camino queda bloqueado al pertenecer el nodo $x_s(k)$ al conjunto $MM^{i,j}$ y no ser un nodo cabeza-cabeza.

Si el camino C activo es de la forma:

$$x_i(k) \rightarrow x_s(k) \leftarrow x_{s'}(k) \ldots x_j(k+1)$$

entonces el camino queda bloqueado al pertenecer tanto el nodo $x_s(k)$ como el nodo $x_{s'}(k)$ al conjunto $MM^{i,j}$. Además hemos de notar que el nodo siguiente en el camino a $x_s(k)$, también pertenece a $V(k)$.

Si el camino C activo es de la forma:

$$x_i(k) \rightarrow x_s(k+1) \rightarrow x_{s'} \ldots x_j(k+1)$$

entonces este camino queda bloqueado si $x_s(k+1) < x_j(k+1)$, porque entonces el nodo $x_s(k+1)$ pertenece al conjunto $MM^{i,j}$. Ahora bien si el nodo $x_s(k+1) > x_j(k+1)$, entonces entre $x_s(k+1)$ y $x_j(k+1)$, se forma un nodo cabeza-cabeza. Suponemos el nodo cabeza-cabeza más cercano al propio $x_s(k+1)$:

$$x_i(k) \rightarrow x_s(k+1) \rightarrow x_{s'} \ldots \rightarrow x_c(k+1) \leftarrow \ldots \leftarrow x_j(k+1)$$

entonces al ser $x_s(k+1) > x_j(k+1)$, también $x_c(k+1) > x_j(k+1)$ y por tanto el camino queda bloqueado porque $x_c(k+1) \notin MM^{i,j}$, a no ser que haya algún descendiente de $x_c(k+1)$ incluido en este subconjunto, pero esto es imposible porque estos descendientes también serían mayores que $x_j(k+1)$ y tampoco pertenecerían al subconjunto $MM^{i,j}$.

Si el camino C activo es de la forma:

$$x_i(k) \rightarrow x_s(k+1) \leftarrow x_{s'} \ldots x_j(k+1)$$

entonces puede ocurrir igual que el caso anterior que:

Si $x_s(k+1) < x_j(k+1)$, entonces el camino está bloqueado, porque tanto $x_s(k+1)$ como $x_{s'}$ pertenecen al subconjunto $MM^{i,j}$.

Si $x_s(k+1) > x_j(k+1)$, entonces el camino está también bloqueado, porque $x_s(k+1) \notin MM^{i,j}$.

Entonces la única forma de activar este camino es siendo éste un arco directo entre ambos nodos $x_i(k)$ y $x_j(k+1)$. ■

Una vez vista la proposición anterior y teniendo en cuenta la definición de $MM^{i,j}$, entonces podemos plantear un algoritmo en donde organizaremos los tests de independencia condicional de tal manera que esté bien definido en todo momento el subconjunto $MM^{i,j}$.

La base del algoritmo que vamos a plantear es la siguiente: Empezamos a realizar tests de independencia condicional dado el subconjunto anteriormente definido como $MM^{i,j}$ por nodos raíces de las estructuras $G(k)$ y $G(k+1)$. Hemos de tener en cuenta entonces que si empezamos por los nodos raíces el subconjunto $MM^{i,j}$ es conocido, y continuaremos realizándolos conforme progresamos en los índices según algún orden ancestral para dichas estructuras; si el resultado del test es negativo entonces colocaremos un arco en el conjunto $E^{tmp}(k+1)$ entre los nodos comprobados y en caso contrario pasamos a la siguiente etapa del algoritmo, esto es, pasar a comprobar los nodos siguientes en el orden ancestral definido. Hemos de notar que si progresamos de esta forma en nuestro algoritmo los arcos que no conocemos en cada una de las etapas serán aquellos que parten desde nodos $x_h(k)$ mayores en el orden al nodo $x_i(k)$ comprobado, hacia cualquier nodo de $k+1$, pero

entonces hemos de notar que los caminos quedan bloqueados por los hijos y padres de $x_i(k)$ (fig. 4.5). Tampoco conocemos los posibles arcos que parten desde el propio $x_i(k)$ hacia nodos mayores en el orden que el nodo $x_j(k+1)$ comprobado, pero como hemos probado, estos caminos incluyen forzosamente al menos un nodo cabeza-cabeza que no pertenece al conjunto $MM^{i,j}$ y por consiguiente al no estar instanciado estos caminos quedan bloqueados.

A continuación presentamos el algoritmo **ART** (*Aprendizaje de Relaciones Temporales*). El pseudocódigo correspondiente a este algoritmo lo podemos observar en la figura 4.6.

Algoritmo ART

Ent: $G(k)$ *y* $G(k+1)$ *Ordenados topológicamente empezando por los nodos raíces.*

Sal: $E^{tmp}(k+1)$

1. $E^{tmp}(k+1) = \emptyset$
2. *Para* $j = 1$ *hasta n hacer*
   (a) *Seleccionar el nodo* $x_j(k+1)$ *de* $G(k+1)$
   (b) *Para* $i = 1$ *hasta n hacer*
      i. *Seleccionar el nodo* $x_i(k)$ *de* $G(k)$
      ii. *Si* $\neg I(x_i(k), x_j(k+1)|MM^{i,j})$ *entonces*
      iii. $E^{tmp}(k+1) = E^{tmp}(k+1) \cup \{(x_i(k), x_j(k+1))\}$

Figura 4.6: Algoritmo ART

Ahora podemos pasar a observar el comportamiento del algoritmo mediante un ejemplo.

Ejemplo 4.2 *Suponemos que tenemos la red de creencia dinámica de la figura 4.7, veamos cómo progresa el algoritmo que anteriormente hemos descrito. En un principio tenemos que el conjunto de relaciones temporales es el vacío y que la ordenación en los nodos de* $V(k)$ *y* $V(k+1)$ *es la que aparece en los subíndices de la figura.* $(x_i), i = 1, ..., 5$.

Como entrada al algoritmo tenemos que la ordenación de los nodos es la si-

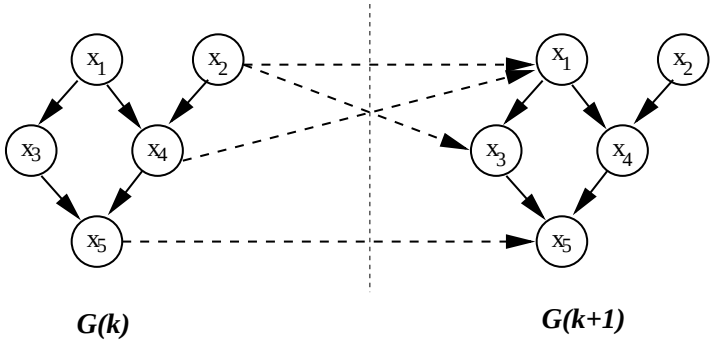


Figura 4.7: Red de creencia dinámica. Ejemplo del Algoritmo ART.

guiente:

$$V(k) = \{x_1(k), x_2(k), x_3(k), x_4(k), x_5(k)\}$$

$$V(k+1) = \{x_1(k+1), x_2(k+1), x_3(k+1), x_4(k+1), x_5(k+1)\}$$

y que: $E^{tmp}(k+1) = \emptyset$

Nos centraremos en unos cuantos casos solamente para describir el funcionamiento del algoritmo, para el resto de casos se proseguiría de igual forma.

- *Para* ($j = 1, i = 1$)
  $MM^{i,j}(x_1(k)) = \{x_3(k), x_4(k), x_2(k)\}$.

  Por tanto comprobamos si $I(x_1(k), x_1(k+1) | \{x_3(k), x_4(k), x_2(k)\})$*, como podemos observar en la figura 4.7, todos los posibles caminos están bloqueados por este subconjunto* $MM^{i,j}(x_1(k))$.

- *Para* ($j = 1, i = 2$)
  $MM^{i,j}(x_2(k)) = \{x_4(k), x_1(k)\}$.

  En este caso tendremos que ver si $I(x_2(k), x_1(k+1) | \{x_4(k), x_1(k)\})$*. Si nos fijamos de nuevo en la estructura de la figura 4.7, podemos observar que si el arco* $x_2(k) \rightarrow x_1(k+1)$ *no existiese todos los caminos entre dichos nodos quedarían bloqueados al igual que en el caso anterior de nuevo por el conjunto* $MM^{i,j}(x_2(k))$*. Todos los caminos que parten desde* $x_2(k)$ *hacia nodos propios de su instante de tiempo k quedarían bloqueados por* $\{x_4(k), x_1(k)\}$*, sin embargo existe un nodo más en el conjunto completo* $MM(x_2(k))$ *que es* $x_3(k+1)$ *que no podremos incluir porque no sabemos de la existencia del arco* $x_2(k) \rightarrow x_3(k+1)$*; afortunadamente conocemos que los caminos que llegan de esta forma hacia* $x_1(k+1)$ *lo hacen a través de sus descendientes y por tanto formaría siempre algún nodo cabeza-cabeza no incluido en el conjunto* $MM^{i,j}$*, como ocurre en el ejemplo.*

De esta forma podremos concluir que existe un arco entre $x_2(k)$ y $x_1(k+1)$.

Siguiendo de igual forma para el resto de casos, al finalizar todas las etapas del algoritmo recuperaremos todos los arcos temporales de la red de creencia dinámica del ejemplo.

■

Ahora pasaremos a demostrar el correcto funcionamiento del Algoritmo ART mediante el siguiente teorema.

Teorema 4.1 *Sea G un grafo dirigido acíclico de una RCD y sea P una distribución sobre V isomorfa a G, entonces partiendo de dos grafos dirigidos acíclicos que representan dos periodos de tiempo consecutivos $G(k)$ y $G(k+1)$ en una RCD el* **algoritmo ART**, *obtiene correctamente el conjunto $E^{tmp}(k+1)$.*
Demostración: La demostración del teorema la haremos por inducción sobre el índice j.
Caso base $j = 1$:

- Caso base $i = 1, j = 1$: En este caso se realiza el test de independencia condicional:

  $$I(x_1(k), x_1(k+1)|MM^{1,1})$$

  $MM^{1,1} = MM_k(x_1(k)) = \pi_k(x_1(k)) \cup Hij_k(x_1(k)) \cup \pi_k(Hij_k(x_1(k)))$, este conjunto es conocido perfectamente en esta situación. Así que aplicando la proposición 4.3, tendremos que existe el arco $x_1(k) \rightarrow x_1(k+1)$, si y solo si $\neg I(x_1(k), x_1(k+1)|MM^{1,1})$.

- Paso de inducción: Suponemos ahora que se cumple para $i = h, j = 1$ y pasamos a demostrar que también se cumple para $i = h+1, j = 1$.

  Se realiza el test de independencia condicional:

  $$I(x_{h+1}(k), x_1(k+1)|MM^{h+1,1})$$

  y $MM^{h+1,1} = MM_k(x_{h+1}(k))$, el cual es conocido en esta etapa de forma correcta, por tanto se sigue de la proposición 4.3 que existe el arco $x_{h+1}(k) \rightarrow x_1(k+1)$, si y solo si $\neg I(x_{h+1}(k), x_1(k+1)|MM^{h+1,1})$.

Paso de inducción: Suponemos que el algoritmo ART recupera de forma correcta los arcos temporales hasta $j = l$.

Tendremos que demostrarlo ahora para $j = l+1$. Para ello de nuevo efectuaremos otra demostración por inducción sobre *i*.

- Caso base $i = 1, j = l+1$: Entonces se realizará el test de independencia condicional:

  $$I(x_1(k), x_{l+1}(k+1) | MM^{1,l+1})$$

  como hasta los nodos menores o iguales que $x_l(k+1)$ se conocen todos los arcos temporales que le llegan desde cualquier nodo $x_i(k)$, entonces se conoce de forma correcta el conjunto $MM^{1,l+1}$ y por tanto se sigue de la proposición 4.3 que existirá el arco $x_1(k) \rightarrow x_{l+1}(k+1)$, si y solo si $\neg I(x_1(k), x_{l+1}(k+1) | MM^{1,l+1})$.

- Paso de inducción: Entonces suponemos que se recuperan de forma correcta los arcos temporales hasta el nodo $x_h(k)$ para $j = l+1$. Tendremos que demostrar que recupera de forma correcta el arco temporal $x_{h+1} \rightarrow x_{l+1}$. En este caso se realiza el test de independencia condicional:

  $$I(x_{h+1}(k), x_{l+1}(k+1) | MM^{h+1,l+1})$$

  según la definición del conjunto $MM^{h+1,l+1}$, en la etapa considerada del algoritmo el conjunto es conocido de forma correcta y por tanto, de nuevo, aplicando la proposición 4.3 existirá el arco $x_{h+1}(k) \rightarrow x_{l+1}(k+1)$, si y solo si $\neg I(x_{h+1}(k), x_{l+1}(k+1) | MM^{h+1,l+1})$.

■

4.4.3 Algoritmo de aprendizaje de relaciones temporales basado en el conjunto mínimo d-separador

Aunque el algoritmo ART, descrito en la sección anterior, tiene un orden $O(n^2)$, siendo n el número de nodos en un instante de tiempo k, los tests de independencia condicional que se realizan pueden ser exponenciales con respecto al número de variables que intervienen en el conjunto sobre el que se condiciona. Sería lógico pensar en mejorar la eficiencia del algoritmo anterior, intentando reducir al máximo el número de variables que intervienen en los tests de independencia condicional, manteniendo el número de tests de independencia condicional que se hacía anteriormente en el algoritmo ART. Lo que pretendemos es eliminar en cada etapa del algoritmo aquellos nodos que sabemos seguro que no intervienen en los caminos entre los nodos que se comprueban desde el punto de vista de la d-separación.

La reducción de los nodos que intervienen en los tests de independencia condicional se basa en el algoritmo para la obtención del conjunto mínimo que d-separa dos variables. Evidentemente la estructura completa no se conoce y por tanto tendremos que incorporar elementos provisionales para situarnos en el caso peor posible

en cada etapa del algoritmo y asegurar que los tests de independencia condicional que se hagan son válidos para determinar la existencia o no de los arcos temporales comprobados en cada etapa del algoritmo. Como hemos estudiado en el punto anterior, de los arcos que no conocemos en un paso del algoritmo ART, sólo influyen un subconjunto de ellos en los tests de independencia. Aprovecharemos esta circunstancia para colocarnos en la situación más desventajosa y suponer que existen todos los arcos que no conocemos y que influyen en los tests de independencia condicional que realicemos.

Recientemente se ha desarrollado un algoritmo para encontrar el conjunto mínimo de nodos que d-separan en un dag dos nodos cualesquiera [4], éste es utilizado, por ejemplo, en el algoritmo de aprendizaje de redes de creencia BENEDICT [5, 7]. Nosotros emplearemos este algoritmo para reducir el tamaño del conjunto MM^{ij}, sin comprometer por ello las garantías que un test basado en dicho subconjunto ofrece para determinar si un arco temporal existe o no.

Conforme el algoritmo ART progresa podemos conocer que algunos nodos del conjunto $MM^{i,j}$ no harán falta considerarlos, ya que no intervienen en ningún camino que conecta los nodos que comprobamos en la iteración i-ésima, y en consecuencia no influyen, desde el punto de vista de la d-separación, en los tests de independencia condicional que realicemos. Esta situación la podemos ver en el siguiente ejemplo.

Ejemplo 4.3 *Consideremos la red de creencia dinámica de la figura 4.8, los arcos que aparecen punteados son los que todavía no conocemos.*

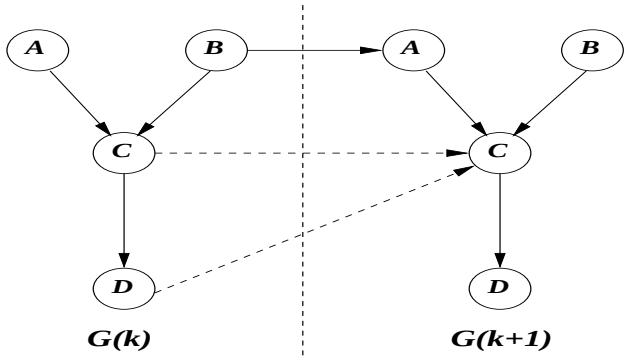


Figura 4.8: Podemos observar que desde el nodo A del periodo de tiempo k no parte ningún arco hacia $k+1$.

Suponemos que estamos en la iteración en la cual se están considerando los nodos $C(k)$ y $C(k+1)$, el algoritmo anterior efectuaría el test de independencia

condicional:

$$I(C(k),C(k+1)|MM^{i,j}) = I(C(k),C(k+1)|\{A(k),B(k),D(k)\})$$

Sin embargo podemos observar en la figura 4.8 que en la iteración i-ésima conocemos que desde el nodo $A(k)$ no parte ninguna flecha hacia el periodo de tiempo $k+1$, y que por lo tanto no influye en el test de independencia condicional que realizamos. Podríamos asegurar la existencia o no del arco tan solo realizando el test $I(C(k),C(k+1)|\{B(k),D(k)\})$.

■

Esta reducción es bastante importante ya que por un lado sabemos que los test de independencia condicional pueden ser exponenciales con respecto al número de nodos que intervienen en el mismo y por otro lado desde el punto de vista práctico es más fiable realizar tests de independencia condicional con un número menor de nodos.

Estas situaciones se podrían detectar en el caso de que encontrásemos la no existencia de un camino entre los nodos que comprobamos, en el caso del ejemplo anterior $C(k)$ y $C(k+1)$, a través de los nodos ancestrales de los mismos. Estos caminos se pueden detectar con el algoritmo que busca el conjunto mínimo d-separador en la etapa correspondiente. Si aplicásemos dicho algoritmo al ejemplo de la figura 4.8(sin tener en cuenta los arcos desconocidos en ese instante), obtendríamos que el conjunto mínimo d-separador es o bien el nodo $B(k)$ o bien el nodo $A(k+1)$. De esta forma podremos asegurar que el nodo $A(k)$ no influye en los tests de independencia condicional que realicemos en esta etapa.

Ahora bien, si por una parte podemos ver que el nodo $A(k)$ no tiene un camino por los nodos ancestros del nodo $C(k+1)$, de éste ultimo no conocemos todos los caminos existentes porque pudiera ser que tuviese más padres desde los descendientes de $C(k)$, como ocurre en el ejemplo anterior. Por tanto si hiciésemos un test de independencia condicional con el conjunto mínimo d-separador en este instante, no podríamos asegurar la existencia de un arco entre dichos nodos, ya que existe un camino a través de los descendientes de $C(k)$. Lo que si sabemos, con certeza en este instante, es que no habrá más caminos ancestrales entre $C(k)$ y $C(k+1)$ que incluyan el nodo $A(k)$, por tanto lo podremos excluir de los test de independencia condicional si ningún problema. Como hemos visto en la sección anterior, los únicos caminos que no conocemos y que pueden influir, son los que parten desde los nodos mayores (en el orden definido por $G(k)$) que el nodo $C(k)$ hacia el nodo $C(k+1)$. Por consiguiente, para asegurarnos que hacemos los tests de independencia condicional correctos, de forma que aseguraremos la existencia de un arco entre $C(k)$ y $C(k+1)$ podríamos poner estos arcos transitoriamente, en nuestro ejemplo

pondríamos el arco $D(k) \rightarrow C(k+1)$. Una vez realizada esta operación bastaría hacer el cálculo del conjunto mínimo d-separador entre los nodos $C(k)$ y $C(k+1)$ y realizar el test de independencia condicional correspondiente con dicho conjunto calculado.

Una vez descrita de manera informal la idea subyacente del próximo algoritmo que vamos a estudiar, pasaremos a formalizar éste. Para ello necesitaremos las siguientes definiciones:

Definición 4.3 *Dado un orden ancestral para $G(k)$ y $G(k+1)$, definimos los conjuntos de relaciones temporales de $G(k,k+1)$:*

$$E^{+}_{i,j}(k+1) = \{(x_s(k), x_j(k+1)) \mid x_s(k) > x_i(k)\}$$

$$E^{-}_{i,j}(k+1) = \{(x_i(k), x_j(k+1))\} \cup \{(x_h(k), x_l(k+1)) \mid x_l(k+1) > x_j(k+1)\}$$

El subconjunto $E^{+}_{i,j}(k+1)$ son todos los posibles arcos temporales que parten desde nodos mayores a $x_i(k)$ en el orden de $G(k)$ hacia el nodo $x_j(k+1)$ de $G(k+1)$. Y el subconjunto $E^{-}_{i,j}(k+1)$ son todos los posibles arcos temporales que parten desde cualquier nodo de $G(k)$ hacia nodos mayores que $x_j(k+1)$ en el orden en $G(k+1)$ más el propio arco temporal entre los nodos $x_i(k)$ y $x_j(k+1)$.

Definición 4.4 *Definimos el subgrafo de $G(k,k+1)$ aumentado, $G_{i,j}$ como:*

$$G_{i,j} = (G(k,k+1) \setminus E^{-}_{i,j}) \cup E^{+}_{i,j}$$

Este grafo $G_{i,j}$ es el original al cual le borramos los arcos del subconjunto $E^{-}_{i,j}$ y le añadidos los arcos del subconjunto $E^{+}_{i,j}$.

A continuación establecemos una nueva caracterización a partir de las definiciones anteriores del conjunto $E^{tmp}(k+1)$.

Proposición 4.4 *Sea G un grafo dirigido acíclico que define una RCD, para cualesquiera dos periodos de tiempo k y $k+1$ consecutivos en G, tenemos que:*

$$\exists\, x_i(k) \rightarrow x_j(k+1) \in E^{tmp}(k+1) \iff \quad (4.4)$$

$$\iff \neg\langle x_i(k), x_j(k+1) | S^{i,j}_d \rangle_{G(k,k+1)} \text{ (d-separación en } G(k,k+1)). \quad (4.5)$$

donde $x_i(k)$ es el nodo i-ésimo de $V(k)$ según un orden ancestral para $G(k)$, $x_j(k+1)$ es el nodo j-ésimo de $V(k+1)$ según un orden ancestral para $G(k+1)$ y $S^{i,j}_d$ es el conjunto mínimo que d-separa $x_i(k)$ de $x_j(k+1)$ en el grafo $G_{i,j}$.

Demostración: La condición necesaria es directa. Demostraremos ahora la condición suficiente. Supondremos que se da $\neg\langle x_i(k), x_j(k+1) | S^{i,j}_d \rangle$ y que no existe el arco $x_i(k) \rightarrow x_j(k+1)$.

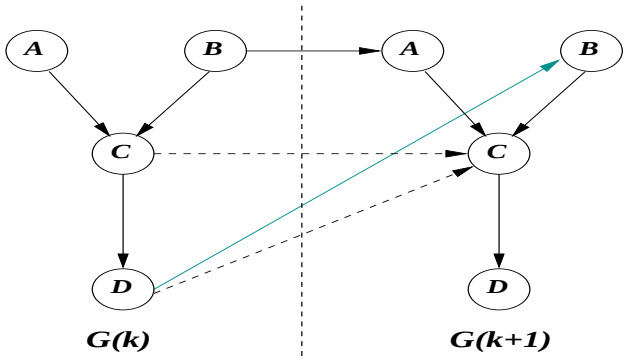


Figura 4.9: Ejemplo de grafo $G_{i,j}$ (para los nodos $C(k)$ y $B(k+1)$) para el grafo de la figura 4.8. Los arcos punteados corresponden al conjunto $E^{-}_{i,j}$ y el arco más claro corresponde al conjunto $E^{+}_{i,j}$.

En primer lugar demostraremos que:
$\forall\, x_h(k) > x_i(k) \mid x_h \in Hij_k(x_i(k))$ ó $x_h(k) \in \pi_k(Hij_k(x_i(k)))$, entonces estos nodos $x_h(k)$ pertenecen siempre al conjunto $S_d^{i,j}$. Bien, en el grafo $G_{i,j}$, existe un arco directo entre los nodos $x_h(k)$ y $x_j(k+1)$, según definición del grafo $G_{i,j}$. Entonces existen caminos entre $x_i(k)$ y $x_j(k+1)$ de la siguiente forma en el grafo aumentado:

$$x_i(k) \rightarrow x_h(k) \rightarrow x_j(k+1)$$

entonces para poder d-separar este camino, forzosamente ha de incluirse en $S_d^{i,j}$ el nodo $x_h(k)$.

Y si el camino es de la forma:

$$x_i(k) \rightarrow x_h(k) \leftarrow x_{ph}(k) \rightarrow x_j(k+1)$$

entonces como hemos incluido en $S_d^{i,j}$ los nodos $x_h(k)$, el camino anterior no estaría bloqueado por el conjunto $S_d^{i,j}$, a no ser que incluyamos también los nodos $x_{ph}(k)$ en dicho conjunto. Por lo tanto, estos nodos también deben incluirse forzosamente en el conjunto mínimo d-separador.

Si $\neg\langle x_i(k), x_j(k+1) | S_d^{i,j}\rangle$, entonces existe un camino C (no arco directo) entre $x_i(k)$ y $x_j(k+1)$ activado por el conjunto $S_d^{i,j}$.

Si el camino C es de la forma:

$$x_i(k) \rightarrow x_h(k) \rightarrow \ldots x_j(k+1)$$

entonces, el camino C está bloqueado porque, como hemos visto en el párrafo anterior, el nodo $x_h(k)$ pertenece siempre al conjunto $S_d^{i,j}$.

Si el camino C es de la forma:

$$x_i(k) \rightarrow x_h(k) \leftarrow x_{ph}(k) \ldots x_j(k+1)$$

entonces, puede ocurrir dos casos:

- Caso en que $x_{ph}(k) > x_i(k)$, entonces el camino C queda bloqueado gracias a que sabemos que el nodo $x_{ph}(k) \in S_d^{i,j}$, según demostramos en el primer párrafo.
- Caso en que $x_{ph}(k) < x_i(k)$, entonces el camino C será de la siguiente forma:

$$x_i(k) \rightarrow x_h(k) \leftarrow x_{ph}(k) \ldots x_s(k) \rightarrow x_u(k+1) \ldots x_j(k+1)$$

  - Caso en que $x_u(k+1) < x_j(k+1)$, entonces el arco $x_s(k) \rightarrow x_u(k+1)$ también existe en el grafo $G_{i,j}$, de lo que se sigue que este camino C es de la misma forma en $G(k,k+1)$, por lo tanto $S_d^{i,j}$ d-separa en camino en ambos grafos.
  - Caso en que $x_u(k+1) > x_j(k+1)$, entonces en $G(k,k+1)$ se forma algún nodo cabeza-cabeza: $x_i(k) \rightarrow x_h(k) \leftarrow x_{ph}(k) \ldots x_s(k) \rightarrow x_u(k+1) \rightarrow \ldots \rightarrow x_c(k+1) \leftarrow \ldots x_j(k+1)$
    Situémonos en $x_c(k+1)$ el nodo cabeza-cabeza más cercano a $x_u(k+1)$. Entonces el arco $x_s(k) \rightarrow x_u(k+1)$ no existe en el grafo $G_{i,j}$, de lo que podemos seguir que $x_c(k+1)$ no podrá pertenecer nunca al conjunto mínimo d-separador $S_d^{i,j}$ y por consiguiente el camino quedaría bloqueado. Tampoco ninguno de sus descendientes puede pertenecer a este conjunto mínimo d-separador ya que si $x_u(k+1) > x_j(k+1)$, entonces cualquier descendiente suyo también lo será. Sin embargo puede ocurrir que $x_c(k+1)$ pertenezca al conjunto $S_d^{i,j}$ porque haya otro camino C' que se bloquee con la inclusión de $x_c(k+1)$, pero de ser así, no queda más remedio que $x_c(k+1)$ sea ancestro de $x_j(k+1)$[2], de donde obtendríamos que $x_c(k+1) < x_j(k+1)$, cosa que contradice nuestra hipótesis ya que si observamos el camino C $x_c(k+1) > x_j(k+1)$. Por la misma razón ninguno de sus hijos puede bloquear ningún otro camino C'.

Si el camino C es de la forma:

$$x_i(k) \leftarrow x_p(k) \ldots x_j(k+1)$$

Entonces al igual que en los casos anteriores tendríamos las siguientes posibilidades:

[2]Hemos de notar que el conjunto mínimo d-separador se encuentra entre los ancestros de $x_j(k+1)$ o los ancestros de $x_i(k)$, entonces si $x_c(k+1)$ bloquea algún camino C', entonces ha de ser ancestro de $x_j(k+1)$ porque de $x_i(k)$ no puede ser nunca ya que no pueden existir arcos del tipo $x_s(k+1) \rightarrow x_u(k)$ en $G(k,k+1)$ en ningún camino.

- Que el camino C fuese de la forma:

$$x_i(k) \leftarrow x_p(k) \ldots x_s(k) \rightarrow x_u(k+1) \ldots x_j(k+1)$$

y que $x_u(k+1) < x_j(k+1)$, siguiendo el mismo razonamiento que el caso anterior se concluye que los caminos son iguales en ambos grafos y que por tanto el conjunto mínimo d-separador bloquearía el camino en ambos grafos.

- El mismo camino C y que $x_u(k+1) > x_j(k+1)$, al igual que el caso anterior se concluye que es una situación en que se forma un nodo cabeza-cabeza que bloquea el camino al no incluirse ni él ni ninguno de sus descendientes en el conjunto mínimo d-separador.

Si el camino C es de la forma:

$$x_i(k) \rightarrow x_u(k+1) \ldots x_j(k+1) \; u \neq j$$

Entonces al igual que en los casos anteriores tendríamos los siguientes casos:

- Si $x_u(k+1) < x_j(k+1)$, entonces el camino es el mismo en $G(k,k+1)$ y en $G_{i,j}$ y por tanto el conjunto $S_d^{i,j}$ bloquea el camino.
- Si $x_u(k+1) > x_j(k+1)$, entonces el arco $x_i(k) \rightarrow x_u(k+1)$ no pertenecerá al grafo $G_{i,j}$. Y por tanto siguiendo el mismo razonamiento de casos anteriores, el camino quedaría bloqueado.

■

El pseudocódigo del algoritmo que denominaremos FART lo podemos observar en la figura 4.10

A continuación pasaremos a ver un ejemplo, en donde compararemos los tests de independencia condicional que realizan los algoritmos que hemos estudiado anteriormente.

Ejemplo 4.4 *Tenemos los instantes de tiempo k y $k+1$ de una red de creencia dinámica que se describe en la siguiente figura.(4.11).*

Veamos para algunas etapas de los algoritmos qué tests han de realizar los algoritmos ART y FART.

- $(A(k), A(k+1))$.

  - ART: $I(A(k), A(k+1) | \{C(k), B(k)\})$

  - FART: Conecto el resto de nodos de k: $C(k)$, $B(k)$, *etc con* $A(k+1)$.
  $S_d^{i,j} = \{C(k), B(k)\} \;\; \Rightarrow \;\; I(A(k), A(k+1) | \{C(k), B(k)\})$
  Desconecto $C(k)$, *etc de* $A(k+1)$.

Algoritmo FART

Ent: $G(k)$ *y* $G(k+1)$ *Ordenados topológicamente a partir de los nodos raíces.*

Sal: $E^{tmp}(k+1)$

1. $E^{tmp}(k+1) = \emptyset$
2. *Para* $j = 1$ *hasta n hacer*
   (a) *Seleccionar el nodo* $x_j(k+1)$ *de* $G(k+1)$
   (b) *Para* $i = 1$ *hasta n hacer*
      i. *Seleccionar el nodo* $x_i(k)$ *de* $G(k)$
      ii. *Calcular* $S = \{x_h(k) \in G(k) / x_h(k) > x_i(k)\}$
      iii. *Conectar con un arco cada* $x_h(k) \in S$ *con* $x_j(k+1)$
      iv. *Calcular el conjunto* $S_d^{i,j}$ *mínimo d-separador entre* $x_i(k)$ *y* $x_j(k+1)$
      v. *Si* $\neg I(x_i(k), x_j(k+1) | S_d^{i,j})$ *Entonces* $E^{tmp}(k+1) = E^{tmp}(k+1) \cup \{(x_i(k), x_j(k+1))\}$
      vi. *Desconectar cada* $x_h(k) \in S$ *de* $x_j(k+1)$

Figura 4.10: Algoritmo FART

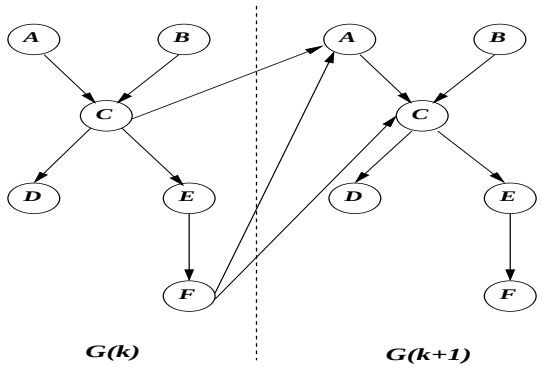


Figura 4.11: Ejemplo comparativo entre ART y FART

- $(C(k), A(k+1))$.

  - *ART:* $\neg I(C(k), A(k+1) | \{A(k), B(k), D(k), E(k)\})$

  - *FART: Conecto* $D(k), E(k), F(k)$ *con* $A(k+1)$.
  $S_d^{i,j} = \{D(k), E(k)\} \Rightarrow \neg I(C(k), A(k+1) | \{D(k), E(k)\})$
  Desconecto los anteriores arcos.
  $E^{tmp}(k+1) = E^{tmp}(k+1) \cup \{(C(k), A(k+1))\}$

- $(C(k), B(k+1))$.

  - *ART:* $I(C(k), B(k+1) | \{A(k), B(k), D(k), E(k), A(k+1), F(k)\})$

  - *FART:* $S_d^{i,j} = \{D(k), E(k)\} \Rightarrow I(C(k), B(k+1) | \{D(k), E(k)\})$

■

Comprobaremos ahora el correcto funcionamiento del algoritmo FART. Para ello demostraremos el siguiente teorema.

Teorema 4.2 *Sea G un grafo dirigido acíclico de una RCD y sea P una distribución sobre V isomorfa a G, entonces partiendo de dos grafos dirigidos acíclicos que representan dos periodos de tiempo consecutivos* $G(k)$ *y* $G(k+1)$ *en una RCD el* **algoritmo FART**, *obtiene correctamente el conjunto* $E^{tmp}(k+1)$.
Demostración: La demostración del teorema la realizaremos por inducción sobre el índice j.
Caso base $j = 1$: Para demostrar el caso base realizaremos inducción sobre el índice i.

- Caso base $i = 1, j = 1$: En este caso se realiza la comprobación:

  $$I(x_1(k), x_1(k+1) | S_d^{1,1})$$

  sobre el grafo $G_{1,1}$, entonces conocemos por la proposición 4.4 que $\exists\, x_1(k) \rightarrow x_1(k+1)$ si y solo si $\neg I(x_1(k), x_1(k+1) | S_d^{1,1})$ sobre $G_{1,1}$, el cual es conocido perfectamente en esta etapa del algoritmo FART.

- Paso de inducción: Supondremos que el algoritmo FART recupera de forma correcta las relaciones temporales hasta $i = h, j = 1$, entonces demostraremos que recupera de forma correcta en la etapa $i = h+1, j = 1$.

  En este etapa se realiza la comprobación:

  $$I(x_{h+1}(k), x_1(k+1) | S_d^{h+1,1})$$

sobre el grafo $G_{h+1,1}$, el cual es conocido perfectamente gracias a que se recuperaron correctamente las relaciones temporales hasta $i = h, j = 1$, entonces conocemos por la proposición 4.4 que $\exists\, x_{h+1}(k) \rightarrow x_1(k+1)$ si y solo si $\neg I(x_{h+1}(k), x_1(k+1)|S_d^{h+1,1})$ sobre $G_{h+1,1}$.

Paso de inducción: Supondremos que el algoritmo FART recupera de forma exacta las relaciones temporales hasta $j = l$. Tendremos que demostrarlo para $j = l+1$. Para ello de nuevo realizaremos inducción sobre el índice i.

- Caso base $i = 1, j = l+1$: Se realiza la comprobación:

  $$I(x_1(k), x_{l+1}(k+1)|S_d^{1,l+1})$$

  sobre el grafo $G_{1,l+1}$, como conocemos perfectamente las relaciones temporales para $j < l+1$, entonces el grafo $G_{1,l+1}$ es conocido de forma correcta y por consiguiente gracias a la proposición 4.4 sabemos que $\exists\, x_1(k) \rightarrow x_{l+1}(k+1)$ si y solo si $\neg I(x_1(k), x_{l+1}(k+1)|S_d^{1,l+1})$ sobre $G_{1,l+1}$.

- Paso de inducción: Suponemos que conocemos de forma exacta las relaciones temporales hasta $i = h, j = l+1$, demostraremos que se recupera de forma correcta $x_{h+1} \rightarrow x_{l+1}(k+1)$. En este caso se realiza la comprobación:

  $$I(x_{h+1}(k), x_{l+1}(k+1)|S_d^{h+1,l+1})$$

  sobre el grafo $G_{h+1,l+1}$, como conocemos las relaciones temporales correctas hasta $i = h, j = l+1$, el grafo $G_{h+1,l+1}$ se define correctamente y sabemos de nuevo por la proposición 4.4 que $\exists\, x_{h+1}(k) \rightarrow x_{l+1}(k+1)$ si y solo si $\neg I(x_{h+1}(k), x_{l+1}(k+1)|S_d^{h+1,l+1})$ sobre $G_{h+1,l+1}$.

■

Desde el punto de vista de la implementación del algoritmo FART, hemos de decir que Acid [2] extiende la búsqueda del conjunto mínimo d-separador para el problema de buscar el conjunto mínimo d-separador entre dos nodos x_i y x_j dado un subconjunto fijo Z, es decir, encontrar S mínimo tal que $I(x_i, x_j|S \cup Z)$. Demuestra que se siguen cumpliendo todas las propiedades que se cumplen para el conjunto mínimo d-separador sin restricciones. A nivel del algoritmo de búsqueda influye de forma beneficiosa ya que básicamente consistirá en eliminar del grafo todos los nodos de Z y a continuación realizar la búsqueda del conjunto S. Esta extensión se puede aplicar en el algoritmo que hemos descrito previamente, pues conocemos de un subconjunto Z que siempre ha de pertenecer al conjunto mínimo d-separador, a saber, los nodos $\forall\, x_h(k) > x_i(k) \mid x_h \in Hij_k(x_i(k))$ ó $x_h(k) \in \pi_k(Hij_k(x_i(k)))$.

4.4.4 Modificaciones de los Algoritmos ART y FART

Siguiendo la filosofía que da lugar al algoritmo que hemos denominado FART, esto es, realizar los tests de independencia condicional con el menor número posible de variables en el conjunto de condicionamiento, podremos modificar los anteriores algoritmos de tal forma que consideremos los conjuntos $MM^{i,j}$ y $S_d^{i,j}$ como una cota máxima del número de variables que deben intervenir en los tests de independencia condicional para asegurar la existencia o no de un arco temporal. Hemos dicho que desde el punto de vista práctico son preferibles los tests de independencia de orden bajo, esta filosofía inspira muchos algoritmos basados en tests de independencia condicional, el más conocido de ellos es el algoritmo PC descrito en el capítulo 1. Lo que pretendemos es seguir esta filosofía y realizar para cada pareja de variables $x_i(k)$ y $x_j(k+1)$ un número de tests de independencia condicional cada vez con un orden mayor empezando por el mínimo y de esta manera evitar realizar en la medida de lo posible los tests de independencia de orden elevado; en cada caso tendremos como cota superior los subconjuntos $MM^{i,j}$ y $S_d^{i,j}$ respectivamente.

A continuación presentaremos la variante del algoritmo ART que efectúa lo anteriormente expuesto. A este algoritmo lo denominaremos ARTSub y podemos observar su pseudocódigo en la figura 4.12.

La idea del algoritmo ARTSub es la siguiente: Sabemos por la proposición 4.3 que para asegurar la existencia de una relación temporal tenemos que realizar un test de independencia condicional con el subconjunto $MM^{i,j}$, pero es claro que en realidad no todos los nodos de este subconjunto serían necesarios para bloquear todos los caminos existentes, rara vez ocurre esto, salvo para aquellos casos en donde efectivamente tengamos un arco temporal. Lo que hacemos es buscar un subconjunto de $MM^{i,j}$ de tamaño mínimo que ya d-separe el par de nodos comprobados en una etapa del algoritmo. De esta manera lo que intentamos es evitar en la medida de lo posible realizar tests de independencia de orden elevado aún a costa de realizar más tests de independencia condicional.

Similar proceder se puede utilizar con el algoritmo FART con el conjunto mínimo d-separador. A continuación mostramos la variante del algoritmo FART con la filosofía descrita (fig. 4.13).

El funcionamiento del algoritmo es el siguiente: Para cada pareja de nodos $x_i(k)$ y $x_j(k+1)$, primero se calcula el conjunto $S_d^{i,j}$ mínimo d-separador en ese paso del algoritmo, notemos que en este caso no hemos introducido ningún arco temporal adicional a la estructura; hemos de tener en cuenta que este subconjunto $S_d^{i,j}$ de variables será el conjunto mínimo que debe de intervenir en la d-separación de estas dos variables, ya que d-separará los caminos que en esta etapa del algoritmo conocemos que serán definitivos en la estructura final. Con este subconjunto calculado realizamos un test de independencia condicional para ambas variables y si resulta positivo entonces ya no deberemos seguir realizando tests de independencia pues ya

Algoritmo ARTSub

Ent: $G(k)$ *y* $G(k+1)$ *Ordenados topológicamente empezando por los nodos raíces.*

Sal: $E^{tmp}(k+1)$

1. $E^{tmp}(k+1) = \emptyset$
2. *Para* $j = 1$ *hasta n hacer*
   (a) *Seleccionar el nodo* $x_j(k+1)$ *de* $G(k+1)$
   (b) *Para* $i = 1$ *hasta n hacer*
      i. *Seleccionar el nodo* $x_i(k)$ *de* $G(k)$
      ii. $Tam = 0$, *Ok = verdadero*
      iii. *Mientras* $Tam \leq ||MM^{i,j}||$ *y Ok hacer*
         A. *Encontrar los subconjuntos S de tamaño Tam de* $MM^{i,j}$
         B. *Si* $I(x_i(k), x_j(k+1)|S)$ *para algún subconjunto S entonces Ok = falso*
         C. $tam = tam + 1$
      iv. *Si Ok = verdadero entonces* $E^{tmp}(k+1) = E^{tmp}(k+1) \cup \{(x_i(k), x_j(k+1))\}$

Figura 4.12: Algoritmo ARTSub

Algoritmo FARTSub

Ent: $G(k)$ *y* $G(k+1)$ *Ordenados topológicamente a partir de los nodos raíces.*

Sal: $E^{tmp}(k+1)$

1. $E^{tmp}(k+1) = \emptyset$
2. *Para* $j = 1$ *hasta n hacer*
   (a) *Seleccionar el nodo* $x_j(k+1)$ *de* $G(k+1)$
   (b) *Para* $i = 1$ *hasta n hacer*
      i. *Seleccionar el nodo* $x_i(k)$ *de* $G(k)$
      ii. *Ok = verdadero*
      iii. *Calcular* $S_d^{i,j}$*, mínimo d-separador entre* $x_i(k)$ *y* $x_j(k+1)$ *(Con el actual grafo aprendido)*
      iv. *Si* $I(x_i(k), x_j(k+1)|S_d^{i,j})$ *entonces*
      Ok = falso
      v. *Calcular* $S = \{x_h(k) \in G(k) | x_h(k) > x_i(k)\}$
      vi. *Conectar con un arco cada* $x_h(k) \in S$ *con* $x_j(k+1)$
      vii. *Calcular el conjunto* $S_d^{i,j'}$ *mínimo d-separador entre* $x_i(k)$ *y* $x_j(k+1)$
      viii. $Tam = ||S_d^{i,j}||$
      ix. *Mientras* $Tam \leq ||S_d^{i,j'}||$ *y Ok hacer*
         A. *Encontrar los subconjuntos S de tamaño Tam de* $S_d^{i,j'} \setminus S_d^{i,j}$
         B. *Si* $I(x_i(k), x_j(k+1)|S \cup S_d^{i,j})$ *para algún subconjunto S entonces*
         Ok =falso
         C. $tam = tam + 1$
      x. *Si Ok = verdadero entonces*
      $E^{tmp}(k+1) = E^{tmp}(k+1) \cup \{(x_i(k), x_j(k+1))\}$
      xi. *Desconectar cada* $x_h(k) \in S$ *de* $x_j(k+1)$

Figura 4.13: Algoritmo FARTSub

sabríamos que no existe un arco temporal entre ambos nodos. Si el test es negativo entonces procederemos igual que en el algoritmo FART, esto es, conectaremos los nodos mayores de $x_i(k)$ en $V(k)$ con el nodo $x_j(k+1)$ y calcularemos el subconjunto de variables $S_d^{i,j'}$. Posteriormente le iremos añadiendo al subconjunto $S_d^{i,j}$ subconjuntos de tamaño cada vez mayor del subconjunto $S_d^{i,j'}$ y realizaremos tests de independencia condicional para cada uno de ellos. Estos tests pueden ser positivos, en cuyo caso se para el proceso o bien pueden ser negativos en cuyo caso seguiría el proceso hasta agotar todos los subconjuntos calculados y concluimos que existe un arco temporal entre los nodos comprobados.

4.5 Refinando los resultados

Bien por la naturaleza greedy de los anteriores algoritmos basados en métricas, o bien por la naturaleza intrínseca de los basados en tests de independencia condicional, los cuales pueden cometer errores en la estimación de independencias a partir de datos, el caso es que todos los algoritmos pueden cometer errores en sus salidas del proceso de aprendizaje, esto es, pueden introducir arcos temporales sin ser éstos verdaderos en el modelo original buscado o a la inversa, borrar arcos temporales que estén presentes en el modelo original. En esta sección lo que pretendemos es ofrecer algoritmos que de alguna forma revisen las salidas de los anteriores algoritmos tal que los modelos resultantes sean más fieles al modelo buscado.

Al igual que venimos haciendo a lo largo de este capítulo, podemos dividir estos algoritmos de revisión de resultados en: a) basados en métricas o b) basados en tests de independencia condicional. Para el primer caso utilizaremos la métrica BIC, por su comportamiento en la preferencia de modelos más simples en las redes candidatas. Para el segundo caso utilizaremos de nuevo tests de independencia condicional buscando en cada momento el conjunto mínimo d-separador en la red candidata.

4.5.1 Refinando mediante la métrica BIC

La idea es la siguiente: Dada una estructura de red de creencia dinámica G como resultado de un proceso de aprendizaje de relaciones temporales, como hemos descrito durante este capítulo, nuestro objetivo es de nuevo maximizar la métrica BIC en cada una de las familias de $x_i(k+1) \in V(k+1)$ utilizando una heurística greedy, la diferencia ahora estriba en que maximizaremos tanto introduciendo nuevos padres de $V(k)$ para cada familia como borrando padres existentes de $V(k)$ en cada una de las familias. Al igual que en los algoritmos anteriores empezaremos a realizar esta búsqueda en los nodos de $G(k+1)$. Una idea parecida utilizan D. Dash y M. Druzdzel, en donde refinan la salida del aprendizaje realizado por el algoritmo PC mediante la métrica BIC [34] y una búsqueda greedy, relanzando el proceso al

llegar a un máximo local. Podemos ver plasmado el pseudocódigo del algoritmo de revisión basado en la métrica BIC en la figura 4.14.

Algoritmo TeReRefinaBIC

1. *Para* $i = 1$ *hasta n hacer*
   (a) $\pi_i(x_i(k+1)) = \pi(x_i(k+1))$
   (b) $OK = True$
   (c) $P_{old} = BIC(x_i(k+1), \pi_i(x_i(k+1))$
   (d) *Mientras OK hacer*
      i. $z(k) = \arg\max_{x(k) \notin \pi_i(x_i(k+1))} BIC(x_i(k+1), \pi_i(x_i(k+1)) \cup \{x(k)\})$
      ó
      $z(k) = \arg\max_{x(k) \in \pi_i(x_i(k+1))} BIC(x_i(k+1), \pi_i(x_i(k+1)) \setminus \{x(k)\})$
      ii. $P_{new} = BIC(x_i(k+1), \pi_i(x_i(k+1)) \cup \{z(k)\})$ *ó*
      $P_{new} = BIC(x_i(k+1), \pi_i(x_i(k+1)) \setminus \{z(k)\})$
      iii. *Si* $P_{new} > P_{old}$ *Entonces*
      $P_{old} = P_{new}$
      $\pi_i(x_i(k+1)) \cup \{z(k)\}$ *ó*
      $\pi_i(x_i(k+1)) \setminus \{z(k)\}$
      iv. *En caso contrario* $OK = false$

Figura 4.14: Algoritmo TeReRefinaBIC

4.5.2 Refinando mediante el conjunto mínimo d-separador

En este caso, el refinamiento del resultado se va a realizar mediante tests de independencia condicional utilizando como conjunto condicionante el mínimo d-separador entre las variables comprobadas. El procedimiento va a consistir en los siguientes pasos: Dada la estructura de red de creencia dinámica G obtenida como resultado de un proceso de aprendizaje de relaciones temporales, comprobaremos pares de variables $x_i(k)$ y $x_j(k+1)$ en el orden ancestral dado por $G(k)$ y $G(k+1)$ intentado introducir arcos temporales que sean dependientes dado el conjunto mínimo d-separador en el instante de la comprobación; seguidamente se inicia otra vez el proceso pero esta vez intentando borrar arcos temporales que sean independientes dado el conjunto mínimo d-separador en el instante de la comprobación. De esta

forma nos aseguramos revisar todos los arcos existentes y no existentes en el resultado inicial posiblemente con otro test de independencia condicional distinto al que se realizó en el proceso de aprendizaje de relaciones temporales. O en el caso de que se haya realizado un aprendizaje mediante optimización de una métrica, pues nos aseguramos de que el grafo resultante es un I-map del modelo que estamos aprendiendo.

Algoritmo TeReRefinaMinDsep

1. *Para* $j = 1$ *hasta n hacer*
   (a) *Seleccionar el nodo* $x_j(k+1)$ *de* $G(k+1)$
   (b) *Para* $i = 1$ *hasta n hacer*
      i. *Seleccionar el nodo* $x_i(k)$ *de* $G(k)$
      ii. *Si* $x_i(k) \rightarrow x_j(k+1) \notin E^{tmp}(k+1)$ *Entonces*
      Calcular el conjunto S_d *mínimo d-separador entre* $x_i(k)$ *y* $x_j(k+1)$
      Si $\neg I(x_i(k), x_j(k+1)|S_d)$ *Entonces*
      $E^{tmp}(k+1) = E^{tmp}(k+1) \cup \{(x_i(k), x_j(k+1))\}$

1. *Para* $j = 1$ *hasta n hacer*
   (a) *Seleccionar el nodo* $x_j(k+1)$ *de* $G(k+1)$
   (b) *Para* $i = 1$ *hasta n hacer*
      i. *Seleccionar el nodo* $x_i(k)$ *de* $G(k)$
      ii. *Si* $x_i(k) \rightarrow x_j(k+1) \in E^{tmp}(k+1)$ *Entonces*
      $E^{tmp}(k+1) = E^{tmp}(k+1) \setminus \{(x_i(k), x_j(k+1))\}$
      Calcular el conjunto S_d *mínimo d-separador entre* $x_i(k)$ *y* $x_j(k+1)$
      Si $\neg I(x_i(k), x_j(k+1)|S_d)$ *Entonces*
      $E^{tmp}(k+1) = E^{tmp}(k+1) \cup \{(x_i(k), x_j(k+1))\}$

Figura 4.15: Algoritmo TeReRefinaMinDsep

La idea del algoritmo, descrito en la figura 4.15, es la siguiente: Primero se recorre cada pareja de variables $(x_i(k), x_j(k+1))$ en donde no exista un arco temporal y se efectúa un test de independencia condicional con el conjunto mínimo d-separador en esa etapa del algoritmo; si el test resulta negativo entonces se introduce un arco temporal entre las variables comprobadas. Posteriormente se realiza el proceso inverso, es decir, para cada pareja de variables $(x_i(k), x_j(k+1))$ en donde

existe un arco temporal en el instante de su comprobación, éste se borra transitoriamente y de nuevo se realiza un test de independencia condicional con el conjunto mínimo d-separador, si es negativo entonces se vuelve a introducir el arco temporal anteriormente borrado.

4.6 Aprendizaje de redes de creencia dinámicas simples

En esta sección vamos a definir un tipo de redes de creencia dinámicas que se presentan frecuentemente en las aplicaciones prácticas y que denominaremos redes de creencia dinámicas simples. La particularidad de este tipo de redes es que en su estructura solo existen relaciones temporales entre nodos de periodos de tiempo consecutivos, esto es, no existe ningún arco que conecte nodos que pertenecen a un mismo periodo de tiempo *k*.

Definición 4.5 *(Redes de creencia simples) Sea una RCD y sea $G = (V,E)$ el grafo dirigido acíclico que define su estructura. Diremos que esta RCD es simple si y solo si:*

$$E = \bigcup_{k=0}^{n-2} E^{tmp}(k+1)$$

siendo n el número de periodos de tiempo definidos en la RCD.

Todos los algoritmos que venimos describiendo a lo largo de este capítulo realizan de una forma completa el aprendizaje de este tipo de redes de creencia dinámicas. El caso es que se puede plantear un algoritmo más eficiente para aprender este tipo de redes. El algoritmo que vamos a plantear se basa en la siguiente proposición.

Proposición 4.5 *Sea G un grafo dirigido acíclico que define una RCD simple. Para cualesquiera dos periodos de tiempo k y $k+1$ consecutivos en G, tenemos que:*

$$\exists\, x_i(k) \rightarrow x_j(k+1) \;\in\; E^{tmp}(k+1) \iff \neg\langle x_i(k), x_j(k+1) | \emptyset \rangle$$

$\forall x_i(k) \in V(k)$ *y* $\forall x_j(k+1) \in V(k+1)$

Demostración: La demostración de esta proposición es trivial, dado que todos los caminos C que pueden unir dos nodos $x_i(k)$ y $x_j(k+1)$ incluyen forzosamente al menos un nodo cabeza-cabeza no instanciado, a no ser que realmente exista una relación temporal entre ambos nodos. ■

Un algoritmo que implemente esta idea es bastante directo, de hecho será igual que los algoritmos ART y FART con el cambio claro de que ahora los tests de independencia condicional serán todos con el conjunto vacío. Hemos de notar que los algoritmos TeRePC, ARTSub, FART y FARTSub, serían equivalentes a este tipo de algoritmo, comentado previamente, ya que TeRePC comienza realizando los tests de independencia condicional precisamente por los conjuntos vacíos, al igual que el algoritmo ARTSub. En cuanto a los algoritmos FART y FARTSub, ambos calcularían como conjunto mínimo d-separador el conjunto vacío.

4.7 Experimentación y Conclusiones

En esta sección realizaremos experimentos para los algoritmos estudiados. Para ello hemos generado redes de creencia dinámicas de forma aleatoria de la siguiente forma:

- Hemos generado redes de creencia dinámicas con 15 variables por cada intervalo de tiempo.
- Para generar la estructura de la red, hemos tomado como parámetros el número medio de padres en el propio intervalo de tiempo k para cada variable, generando éste mediante una distribución de Poisson. Hemos generado redes para valores de 0.5, 1.0 y 1.5 como media de padres en el propio intervalo de tiempo. Las relaciones temporales, al igual que los padres anteriores, se han generado utilizando una distribución de Poisson y tomando valores 1.0, 1.5 y 2.0. Lo cual hace un total de 9 redes de creencia dinámicas generadas.
- Para generar las distribuciones de probabilidad condicional para cada nodo, lo hemos realizado tomando como valor medio del número de estados para cada variable, 2 estados y generando éste también con una distribución de Poisson, con la restricción de un mínimo de 2 estados por variable. Las tablas de probabilidad se generaron utilizando una distribución uniforme elevada al cuadrado para generar probabilidades de mediana extremidad en sus valores.
- A partir de estas redes de creencia dinámicas, generamos una base de datos de casos de 10.000 observaciones, realizando este proceso mediante muestreo lógico probabilístico. Y para cada proceso de aprendizaje, hemos considerado estos casos de 2000 en 2000, empezando por los 2000 primeros casos y terminando por la base de datos completa.
- Se han aprendido las relaciones temporales para estas redes a partir de los datos generados con cada uno de los algoritmos descritos durante el capítulo, obteniéndose para cada uno de ellos la media de los resultados para cada una

de las bases de datos. Los estadísticos de los resultados se muestran en tablas con la siguiente distribución: En primer lugar los casos considerados de la base de datos para el aprendizaje; en segundo lugar el tiempo de ejecución (T) medido en segundos; en tercer lugar la medida KL descrita en el capítulo 2 del resultado ofrecido para cada uno de los algoritmos; en cuarto lugar el número de arcos temporales borrados (B) en comparación con la red original de partida; en quinto lugar el número de arcos temporales añadidos; a continuación la distancia de Hamming (H) (diferencia estructural) como resumen de los dos resultados anteriores; en séptimo lugar se presenta el número de variables en el conjunto de condicionamiento en los tests de independencia condicional o el número medio de variables en los estadísticos evaluados, dependiendo si el algoritmo está basado en tests o en una métrica, respectivamente; por último presentamos el número de tests de independencia condicional o el número de estadísticos evaluados dependiendo del tipo de algoritmo utilizado.

- Para el algoritmo TeReK2 se ha utilizado la métrica K2 en todos los experimentos.

Análisis de los resultados para el Algoritmo ART. Los resultados del algoritmo ART se pueden observar en la tabla 4.1. También podemos observar en las tablas 4.2 y 4.3 los refinamientos del resultado del algoritmo ART con los algoritmos TeReRefinaMinDsep y TeReRefinaBIC respectivamente.

Casos	T	KL	B	A	H	Condic.	Tests
2000	41,62	3,0878	12,22	1,44	13,67	3,22	225
4000	59,70	3,2632	10,11	1,78	11,89	3,50	225
6000	74,73	3,3076	9,44	2,22	11,67	3,65	225
8000	92,67	3,3404	8,67	1,56	10,22	3,60	225
10000	92,43	3,3458	8,33	1,44	9,78	3,55	225

Tabla 4.1: Resultados para el Algoritmo ART con Redes Dinámicas de 15 nodos por intervalo de tiempo

En cuanto al algoritmo ART cabe destacar en primer lugar la tendencia de éste a borrar enlaces, esto es, estima independientes parejas de variables que en realidad no lo son. En general necesita un número elevado de casos para confirmar las relaciones de dependencia entre variables. Esta tendencia se nota en que cuantos más datos contemplamos en el aprendizaje, el número de enlaces borrados decrece. También hemos de notar que este algoritmo, en general, necesita calcular estadísticos de un orden elevado, es decir, los tests de independencia condicional que realiza son de un orden bastante alto, alrededor de 5,5 variables están involucradas en estos tests.

Casos	T	KL	B	A	H
2000	6,95	4,0230	4,22	3,33	7,56
4000	11,77	4,0848	2,22	3,22	5,44
6000	14,80	4,0744	2,11	4,00	6,11
8000	17,52	4,0606	2,00	2,78	4,78
10000	21,62	4,0401	1,78	3,11	4,89

Tabla 4.2: Resultados para el Algoritmo ART + refinamiento con Mínimo d-separador con Redes Dinámicas de 15 nodos por intervalo de tiempo

Casos	T	KL	B	A	H
2000	6,63	3,9564	4,56	0,00	4,56
4000	12,71	4,0077	3,00	0,00	3,00
6000	19,65	4,0758	2,00	0,00	2,00
8000	24,69	4,0780	1,56	0,11	1,67
10000	30,65	4,0824	1,33	0,00	1,33

Tabla 4.3: Resultados para el Algoritmo ART + refinamiento con BIC con Redes Dinámicas de 15 nodos por intervalo de tiempo

El orden elevado de estos tests hace que se necesiten, en general, un número más elevado de casos para que dichos tests sean más fiables.

Analizando la salida del algoritmo ART con un posterior refinamiento con el algoritmo TeReRefinaMDS, lo primero que destaca es el descenso considerable en el número de enlaces borrados y que eleva de una forma muy ligera el número de arcos añadidos. En general, esto puede deberse a la utilización de los conjuntos mínimos d-separadores los cuales harán que el número de variables involucradas en los tests descienda considerablemente. Esto último redundará en que los tests realizados sean más fiables y que en algún sentido éstos se vuelvan más conservadores. Esta última consideración la pone de manifiesto el hecho de que prácticamente se añadan el mismo número de arcos para todas las bases de datos comprobadas.

En cuanto al refinamiento con la métrica BIC, podemos notar que prácticamente a partir de los 4000 primeros datos ofrece un resultado bastante parecido. El resultado de este refinamiento se puede considerar como una buena aproximación a las redes de creencia dinámicas reales y que por tanto el algoritmo ART es un buen punto de inicio para una posterior búsqueda local con la métrica utilizada. En este resultado puede influir de manera decisiva el que los arcos temporales que ofrece el algoritmo ART sean bastantes “sólidos” (dependientes) y que por tanto una búsqueda local posterior se vea favorecida por este hecho.

Análisis de los resultados del Algoritmo ARTSub. Los resultados del algoritmo ARTSub se pueden observar en la tabla 4.4. También podemos observar en las tablas 4.5 y 4.6 los refinamientos del resultado del algoritmo ARTSub con los algoritmos TeReRefinaMinDsep y TeReRefinaBIC respectivamente.

Casos	T	KL	B	A	H	Condic.	Tests
2000	15,07	3,1200	12,56	0,44	13,00	1,41	697,11
4000	42,59	3,2982	9,89	0,44	10,33	1,66	999,11
6000	78,99	3,3405	9,11	0,56	9,67	1,83	1237,33
8000	154,46	3,3884	8,44	0,22	8,67	1,98	1648,22
10000	163,18	3,4008	8,00	0,22	8,22	2,00	1517,33

Tabla 4.4: Resultados para el Algoritmo ARTSub con Redes Dinámicas de 15 nodos por intervalo de tiempo

Casos	T	KL	B	A	H
2000	6,89	4,0230	4,22	3,33	7,56
4000	11,46	4,0782	2,33	3,33	5,67
6000	14,71	4,0940	1,89	3,89	5,78
8000	16,70	4,0716	1,89	2,67	4,56
10000	20,54	4,0399	1,78	2,78	4,56

Tabla 4.5: Resultados para el Algoritmo ARTSub + refinamiento con Mínimo d-separador con Redes Dinámicas de 15 nodos por intervalo de tiempo

Casos	T	KL	B	A	H
2000	6,17	3,9564	4,56	0,00	4,56
4000	11,73	4,0077	3,00	0,00	3,00
6000	17,54	4,0758	2,00	0,00	2,00
8000	22,25	4,0780	1,56	0,11	1,67
10000	26,77	4,0824	1,33	0,00	1,33

Tabla 4.6: Resultados para el Algoritmo ARTSub + refinamiento con BIC con Redes Dinámicas de 15 nodos por intervalo de tiempo

Lo primero que cabe destacar es que el algoritmo ARTSub mantiene la tendencia del algoritmo ART en cuanto al número de arcos borrados, con una ligera mejora. También se mejora ligeramente el número de arcos añadidos y la métrica KL con respecto al algoritmo ART. Este resultado puede explicarse por el hecho de que bajamos el orden de los tests de independencia condicional realizados por el algoritmo

ARTSub y por tanto éstos serán más fiables. Sin embargo, el hecho de bajar el orden de los tests implica un número muy elevado de tests[3] lo cual ha afectado de forma significativa en la eficiencia del algoritmo y también en la eficacia, porque al realizar más tests de independencia condicional tenemos una mayor probabilidad de cometer errores en las estimaciones de independencia o dependencia entre variables.

Por otra parte, cuando se refina la salida del algoritmo ARTSub con los conjuntos mínimos d-separadores, prácticamente se mantienen los resultados con el anterior algoritmo ART más este mismo refinamiento, aunque se mejoran los resultados, estas mejoras son muy leves. También podemos indicar, tanto en este caso como para el caso anterior (algoritmo ART), que el proceso de revisión de los enlaces temporales es bastante eficiente, esto es, es un proceso rápido en comparación con los algoritmos propiamente dichos ART y ARTSub.

Para el refinamiento BIC, hemos de notar que se llegan a los mismos resultados y que prácticamente su tiempo de ejecución es similar comparado con el anterior caso.

Análisis de los resultados del Algoritmo FART. Los resultados del algoritmo FART se pueden observar en la tabla 4.7. También podemos observar en las tablas 4.8 y 4.9 los refinamientos del resultado del algoritmo FART con los algoritmos TeReRefinaMinDsep y TeReRefinaBIC respectivamente.

Casos	T	KL	B	A	H	Condic.	Tests
2000	40,58	3,6165	7,44	3,56	11,00	2,05	225
4000	44,48	3,7208	5,78	3,11	8,89	2,14	225
6000	48,51	3,7340	5,00	3,44	8,44	2,12	225
8000	52,62	3,7683	4,44	2,78	7,22	2,13	225
10000	60,20	3,7766	4,11	3,56	7,67	2,13	225

Tabla 4.7: Resultados para el Algoritmo FART con Redes Dinámicas de 15 nodos por intervalo de tiempo

Los resultados que ofrece el algoritmo FART son considerablemente mejores que los ofrecidos por los algoritmos ART y ARTSub, tanto en distancia de Hamming (H) como en la medida KL. La tendencia de este algoritmo a borrar arcos temporales es sensiblemente menos acusada que en los anteriores algoritmos. Sin embargo, este algoritmo tiende a añadir un número de arcos levemente mayor que en los anteriores casos. Por otra parte, podemos notar que este algoritmo es más eficiente que los anteriores algoritmos, esto es debido a que los estadísticos que ha de calcular son de un orden sensiblemente inferior que en el caso del algoritmo ART y que mantiene el mismo número de tests realizados. Además se observa claramente que la inversión

[3] Hemos de tener en cuenta que, en general, el cardinal del subconjunto $MM^{i,j}$ es elevado.

Casos	T	KL	B	A	H
2000	7,44	4,0354	4,00	3,11	7,11
4000	11,35	4,0564	2,33	3,44	5,78
6000	16,22	4,0470	2,22	4,00	6,22
8000	17,09	4,0727	1,78	2,56	4,33
10000	21,39	3,9932	2,00	3,11	5,11

Tabla 4.8: Resultados para el Algoritmo FART + refinamiento con Mínimo d-separador con Redes Dinámicas de 15 nodos por intervalo de tiempo

Casos	T	KL	B	A	H
2000	6,64	3,9564	4,56	0,00	4,56
4000	12,25	4,0077	3,00	0,00	3,00
6000	18,86	4,0771	1,89	0,00	1,89
8000	23,23	4,0689	1,67	0,11	1,78
10000	29,75	4,0825	1,33	0,00	1,33

Tabla 4.9: Resultados para el Algoritmo FART + refinamiento con BIC con Redes Dinámicas de 15 nodos por intervalo de tiempo

de tiempo en calcular subconjuntos mínimos d-separadores se ve compensanda en la eficiencia de los tests que realiza el algoritmo. Este descenso en el orden de los tests realizados implica una mayor fiabilidad en los mismos, esta puede ser la razón de su mayor eficacia con respecto a los anteriores casos.

Cuando aplicamos el refinamiento con el mínimo conjunto d-separador se reduce considerablemente el número arcos borrados y se mejora levemente el número de arcos añadidos. También se mejora la medida KL con respecto a la salida del algoritmo FART. Sin embargo hemos de notar que estos resultados empeoran de forma leve los ofrecidos por este refinamiento para los algoritmos ART y ARTSub. También hemos de notar que este refinamiento mantiene su eficiencia con respecto al tiempo de cómputo empleado cuando se compara por el tiempo empleado para obtener la primera salida por el algoritmo en cuestión. Por último indicar que prácticamente se mantienen los resultados ofrecidos por los anteriores algoritmos para el refinamiento BIC, observándose sólo muy ligeras diferencias para un par de resultados.

Análisis de resultados para el Algoritmo FARTSub. Estos resultados se pueden observar en las tablas 4.10, 4.11 y 4.12 para el algoritmo FARTSub y los sus refinamientos posteriores con el mínimo d-separador y con la métrica BIC respectivamente.

Casos	T	KL	B	A	H	Condic.	Tests
2000	12,31	3,5784	8,00	2,89	10,89	1,10	378,56
4000	30,92	3,6898	5,89	2,89	8,78	1,30	492,89
6000	46,80	3,7158	5,22	2,78	8,00	1,38	537,44
8000	65,91	3,7502	4,67	2,22	6,89	1,47	583,11
10000	80,87	3,7371	4,44	2,22	6,67	1,53	609,11

Tabla 4.10: Resultados para el Algoritmo FARTSub con Redes Dinámicas de 15 nodos por intervalo de tiempo

Casos	T	KL	B	A	H
2000	7,68	4,0394	3,89	3,00	6,89
4000	10,93	4,0500	2,44	3,22	5,67
6000	14,29	4,0244	2,56	4,22	6,78
8000	16,46	4,0252	2,22	2,78	5,00
10000	22,12	4,0040	1,89	2,67	4,56

Tabla 4.11: Resultados para el Algoritmo FARTSub + refinamiento con Mínimo d-separador con Redes Dinámicas de 15 nodos por intervalo de tiempo

Casos	T	KL	B	A	H
2000	6,56	3,9564	4,56	0,00	4,56
4000	12,22	4,0077	3,00	0,00	3,00
6000	18,23	4,0771	1,89	0,00	1,89
8000	22,73	4,0780	1,56	0,11	1,67
10000	27,83	4,0825	1,33	0,00	1,33

Tabla 4.12: Resultados para el Algoritmo FARTSub + refinamiento con BIC con Redes Dinámicas de 15 nodos por intervalo de tiempo

Este algoritmo prácticamente borra el mismo número de arcos que el algoritmo FART, aunque mejora el número de arcos añadidos. Sin embargo la media KL prácticamente se mantiene en el mismo orden para ambos algoritmos. Como podemos observar en los resultados, el número de variables que intervienen en el conjunto de condicionamiento es menor y por tanto esto ayudará a que los tests realizados sean más fiables. De todas formas hemos de observar también que el número de tests de independencia condicional aumenta considerablemente con respecto al algoritmo FART lo cual redunda en que este último algoritmo sea menos eficiente.

En cuanto al refinamiento con el mínimo conjunto d-separador, podemos observar que los resultados son bastante parecidos al resto de casos comentados previamente. Lo mismo ocurre con el refinamiento con BIC.

Análisis de los resultados para el Algoritmo TeRePC. Los resultados ofrecidos por este algoritmo se pueden observar en la tabla 4.13, al igual que los refinamientos posteriores con el conjunto mínimo d-separador, tabla 4.14, y el refinamiento con la métrica BIC, tabla 4.15.

Casos	T	KL	B	A	H	Condic.	Tests
2000	4,76	3,9087	5,44	0,89	6,33	0,49	370,67
4000	10,68	4,0005	3,33	1,00	4,33	0,60	430,78
6000	18,02	4,0117	2,67	1,00	3,67	0,67	488,78
8000	25,94	3,9996	2,44	0,67	3,11	0,69	529,56
10000	34,06	4,0103	2,11	0,89	3,00	0,74	557,22

Tabla 4.13: Resultados para el Algoritmo TRPC con Redes Dinámicas de 15 nodos por intervalo de tiempo

Casos	T	KL	B	A	H
2000	6,97	4,0254	4,00	3,00	7,00
4000	10,42	4,1019	2,00	3,00	5,00
6000	13,68	4,1033	1,67	3,67	5,33
8000	15,66	4,0966	1,67	2,44	4,11
10000	19,12	4,0926	1,33	2,44	3,78

Tabla 4.14: Resultados para el Algoritmo TRPC + refinamiento con Mínimo d-separador con Redes Dinámicas de 15 nodos por intervalo de tiempo

En primer lugar cabe destacar el buen resultado que ofrece este algoritmo, tanto en la distancia de Hamming (H) como en medida KL. Es sin duda el algoritmo basado en tests de independencia condicional más eficaz de los que hemos estudiado, ofreciendo en general un resultado excelente. Esta eficacia puede ser debida al

Casos	T	KL	B	A	H
2000	5,61	3,9536	4,67	0,11	4,78
4000	10,19	4,0051	3,00	0,00	3,00
6000	14,20	4,0758	2,00	0,00	2,00
8000	18,36	4,0780	1,56	0,11	1,67
10000	23,36	4,0825	1,33	0,00	1,33

Tabla 4.15: Resultados para el Algoritmo TRPC + refinamiento con BIC con Redes Dinámicas de 15 nodos por intervalo de tiempo

hecho del reducido número de variables que intervienen en el conjunto de condicionamiento, lo cual no llevará a realizar unos tests de independencia muy fiables, esto hecho lo pone de manifiesto el que a partir de 4000 datos, el algoritmo empieza a obtener buenos resultados.

También este algoritmo se destaca del resto de los estudiados por su eficiencia, es sin lugar a dudas el más rápido de todos los comprobados hasta el momento. Este hecho cabe especial mención porque si observamos el número de tests de independencia condicional que realiza es del orden de los realizados por el algoritmo FARTSub, y también es mayor que los realizados por los algoritmos ART y FART. Sin embargo, el número bastante reducido de variables que intervienen en los mismos hace que estos sean mucho más rápidos (eficientes) de calcular.

En cuanto a su refinamiento con el conjunto mínimo d-separador, se mantiene la tendencia del resto de casos, esto es, disminuye el número de arcos borrados, pero aumenta el número de arcos añadidos, en este caso de una manera más acusada. En el refinamiento BIC se mantienen los mismos resultados. Esto nos hace concluir que todos los algoritmos vistos hasta ahora basados en tests de independencia condicional, proporcionan un buen punto de inicio para la posterior búsqueda local que se realiza en este refinamiento.

Análisis de los resultados para el Algoritmo TeReBenedict. Este algoritmo aunque realiza tests de independencia condicional, lo hemos encuadrado en los basados en métricas. Por consiguiente, hemos reflejado en número de estadísticos evaluados, en lugar de tests de independencia condicional, y el número de variables completas que intervienen en dichos estadísticos. Hemos de observar que aunque se realizan tests de independencia condicional, éstos no suponen un cálculo extra, sino que se pueden reutilizar los cálculos efectuados en la métrica correspondiente.

Los resultados de este algoritmo junto con sus refinamientos se pueden observar en las tablas 4.16, 4.17 y 4.18.

Los resultados ofrecidos por el algoritmo TeReBenedict son del mismo orden que los ofrecidos por los algoritmos FART y FARTSub, aunque TeReBenedict es

Casos	T	KL	B	A	H	NVars	EstEval
2000	24,27	3,6557	8,00	3,11	11,11	2,46	262,89
4000	41,91	3,7199	6,11	3,00	9,11	2,49	272,22
6000	63,50	3,8236	4,89	2,78	7,67	2,53	280,56
8000	80,93	3,7908	4,78	2,56	7,33	2,57	280,78
10000	102,99	3,8106	4,33	2,44	6,78	2,59	284,11

Tabla 4.16: Resultados para el Algoritmo TRBenedict con Redes Dinámicas de 15 nodos por intervalo de tiempo

Casos	T	KL	B	A	H
2000	6,81	4,0232	4,11	3,22	7,33
4000	10,84	4,0597	2,44	3,11	5,56
6000	14,28	4,0683	1,89	3,44	5,33
8000	15,96	4,0667	1,89	2,67	4,56
10000	20,00	4,0685	1,56	2,67	4,22

Tabla 4.17: Resultados para el Algoritmo TRBenedict + refinamiento con Mínimo d-separador con Redes Dinámicas de 15 nodos por intervalo de tiempo

Casos	T	KL	B	A	H
2000	6,76	3,9552	4,56	0,00	4,56
4000	12,28	4,0051	3,00	0,00	3,00
6000	20,16	4,0769	1,78	0,00	1,78
8000	21,91	4,0780	1,56	0,11	1,67
10000	27,19	4,0825	1,33	0,00	1,33

Tabla 4.18: Resultados para el Algoritmo TRBenedict + refinamiento con BIC con Redes Dinámicas de 15 nodos por intervalo de tiempo

ligeramente menos eficiente. En cuanto al refinamiento con el mínimo conjunto d-separador, hemos de notar que se mejoran de forma muy leve los resultados ofrecidos por estos refinamientos para los algoritmos FART y FARTSub. Para el refinamiento BIC no se puede decir nada nuevo, pues prácticamente se copian los mismos resultados que en los casos anteriores.

Análisis de los resultados para el Algoritmo TeReK2. El resultado de la ejecución de este algoritmo para las bases de datos de prueba se puede observar en la tabla 4.19. Estos resultados son los mejores encontrados por todos los algoritmos que hemos estudiado a lo largo de este capítulo y se puede considerar como unos resultados excelentes. En cuanto a la eficiencia del método cabe notar que es un poco menos eficiente que los algoritmos basados en tests de independencia condicional. Sin embargo, hemos de notar que cualquiera de los algoritmos basados en tests más un refinamiento BIC, que es el mejor resultado encontrado para estos casos, en algunos casos empeora el tiempo de ejecución, sumando ambos tiempos, o en otros es prácticamente el mismo tiempo de cómputo.

En cuanto a los refinamientos, lo que hacen es empeorar los resultados ofrecidos por este algoritmo, es por ello, que no hemos reflejado los resultados en tablas.

Casos	T	KL	B	A	H	NVars	EstEval
2000	21,37	4,2885	0,33	0,78	1,11	3,72	556,44
4000	39,83	4,2232	0,11	0,22	0,33	3,71	552,56
6000	59,02	4,2034	0,00	0,33	0,33	3,72	555,33
8000	78,64	4,1872	0,11	0,67	0,78	3,72	558,44
10000	97,65	4,1785	0,11	0,44	0,56	3,70	555,89

Tabla 4.19: Resultados para el Algoritmo TRK2 con Redes Dinámicas de 15 nodos por intervalo de tiempo

Capítulo 5

Aprendizaje de Redes de Creencia Dinámicas

5.1 Introducción

En este capítulo vamos a estudiar cómo podemos aprender una RCD a partir de datos de un sistema dinámico markoviano y estacionario. Una vez estudiados a lo largo de esta memoria tanto algoritmos para el aprendizaje estructural de redes de creencia estáticas como algoritmos para el aprendizaje de las relaciones temporales de una red de creencia dinámica, conocido su modelo estático, estamos en disposición de abordar el estudio del aprendizaje de redes de creencia dinámicas a partir de los algoritmos anteriores.

Este capítulo se estructura de la siguiente manera: la primera sección se dedica a estudiar el problema del aprendizaje de RCD cuando conocemos el primer instante de tiempo, a partir del cual se empieza a observar el sistema dinámico. En este caso plantearemos un esquema general de aprendizaje de RCD, en donde nos basamos en gran medida en los algoritmos estudiados en los capítulos anteriores, esto es, en los algoritmos de aprendizaje de redes de creencia estáticas y en los algoritmos de aprendizaje de relaciones temporales.

En la siguiente sección nos centraremos en el estudio de la utilización del esquema anterior de aprendizaje de RCD cuando un sistema dinámico se empieza a observar a partir de un periodo de tiempo arbitrario, y éste no tiene por qué coincidir con el instante de tiempo inicial.

Por último estudiaremos cómo adaptar algoritmos de aprendizaje de redes de creencia estáticos al aprendizaje de redes de creencia dinámicas. Fundamentalmente, para realizar esta tarea, consideraremos $G(k,k+1)$ como un modelo estático con unas ciertas restricciones.

5.2 Aprendizaje de RCD cuando se conoce el instante $t = 0$

El problema que pretendemos resolver es el siguiente: Si tenemos que modelar un sistema dinámico en donde existe un instante inicial de tiempo a partir del cual empieza el proceso evolutivo del sistema, entonces existe un periodo que podemos distinguir como periodo de tiempo inicial $k = 0$, cuya característica principal es que las variables en este periodo de tiempo no son dependientes del instante anterior. En términos de independencias condicionales, esto último quiere decir que si $I(x_i(0), x_j(0)|S)$, entonces S es un subconjunto de variables de $V(0)$.

Por otra parte, si queremos aprender una red de creencia dinámica que es markoviana y estacionaria, entonces podremos utilizar un algoritmo de aprendizaje automático para redes de creencia estáticas para inducir la red de creencia para este periodo de tiempo $k = 0$. A partir de esta red que hemos aprendido podremos duplicarla para el resto de periodos de tiempo y entonces podremos utilizar un algoritmo de aprendizaje automático de relaciones temporales para completar de esta manera el aprendizaje de la red de creencia dinámica.

A continuación describiremos el método general de aprendizaje de redes de creencia dinámicas de la manera descrita anteriormente, posteriormente pasaremos a estudiar los problemas que nos plantean las salidas de los algoritmos de aprendizaje automático de redes de creencia estáticos y la aplicación posterior de este método.

5.2.1 Método de aprendizaje general

A partir de las condiciones anteriores se puede estudiar un método general de aprendizaje automático de redes de creencia dinámicas. Este método general es directo, teniendo en cuenta los algoritmos de aprendizaje de relaciones temporales estudiados en el capítulo anterior. El método de aprendizaje general consistirá en los siguientes pasos:

1. Utilizar un algoritmo general de aprendizaje de redes de creencia estáticas restringido al periodo de tiempo inicial del modelo $k = 0$.

2. Duplicar la estructura aprendida en $G(0)$ para los periodos de tiempo posteriores, esto es, $1, \ldots n-1$.

3. Utilizar un algoritmo de aprendizaje de relaciones temporales a partir de dos estructuras correspondientes a dos periodos de tiempo consecutivos k y $k+1$.

Si bien esta metodología descrita parece directa y razonable, hemos de considerar que los algoritmos de aprendizaje automático para redes de creencia estáticas

no son capaces, por lo general, de recuperar de manera perfecta las estructuras subyacentes de los modelos que se quieren aprender. Por otra parte hemos estudiado algoritmos de aprendizaje de relaciones temporales que dependen fuertemente de las estructuras que definen el modelo en los intervalos de tiempo *k*, de hecho siempre hemos partido de que estas estructuras eran perfectamente conocidas. Por tanto, parece lógico considerar los posibles problemas que nos plantean los diferentes algoritmos de aprendizaje automático de redes de creencia estáticas.

Hemos demostrado en el capítulo anterior que si la estructura del grafo dirigido acíclico que representa $G(0)$ es correcta y si ésta es la misma para el resto de instantes de tiempo, si duplicamos $G(0)$ para el resto de instantes de tiempo, entonces los algoritmos basados en tests de independencia condicional, ART, ARTSub, FART, FARTSub y TeRePC recuperan de una manera correcta el conjunto $E^{tmp}(k+1)$ de relaciones temporales de la red de creencia dinámica[1].

Ahora bien, existen algoritmos que no siempre recuperan la estructura correcta, estos algoritmos pueden ofrecernos una buena aproximación de $G(0)$, pero no siempre tiene que ser la correcta. Por tanto, sería lógico estudiar el efecto que sobre los algoritmos ART, ARTSub, FART, FARTSub y TeRePC tiene la situación de que la estructura recuperada para $G(0)$ no fuese la correcta, sino una buena aproximación a la estructura verdadera del grafo.

El comportamiento de los algoritmos basados en tests de independencia condicional estudiados en el capítulo anterior, para aquellos casos en que se parte de buenas aproximaciones, no correctas, del grafo $G(0)$ se describe en la siguiente proposición.

Proposición 5.1 *Sea $\hat{G}(0)$ una aproximación de $G(0)$, estructura para el instante de tiempo inicial $k = 0$ de una red de creencia dinámica. Si duplicamos la estructura $\hat{G}(0)$ para k, $(k > 0)$ y aplicamos los algoritmos de aprendizaje de relaciones temporales basados en tests de independencia condicional para k y $k+1$, entonces el conjunto de relaciones temporales $\hat{E}^{tmp}(k+1)$ recuperado contendrá como mínimo las relaciones temporales $E^{tmp}(k+1)$ de la red de creencia dinámica, es decir, $\hat{E}^{tmp}(k+1) \supseteq E^{tmp}(k+1)$.*

Demostración: Si $\exists$ un arco $w_i(k) \rightarrow w_j(k+1)$ en $E^{tmp}(k+1)$, entonces no habrá ningún subconjunto de nodos que sea capaz de d-separar ambos nodos y por tanto se haga el test de independencia condicional que se haga, los algoritmos basados en dichos tests colocarán un arco entre ambos nodos al no poder d-separarlos con el subconjunto correspondiente.

Esta proposición nos indica que estos algoritmos nos proporcionan siempre como salida un I-map[2] de la red de creencia dinámica correspondiente a dos intervalos

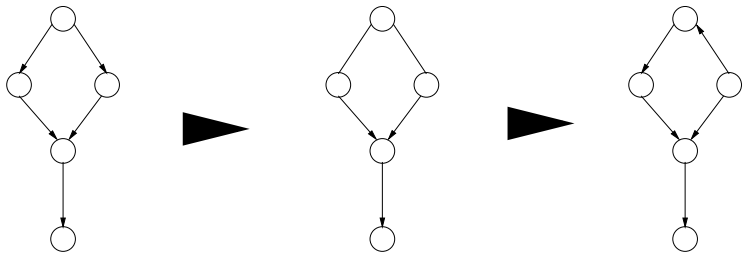
[1] Asumiendo que los tests de independencia no comenten errores.

[2] Siempre y cuando el modelo recuperado $G(0)$ también sea un I-map.

de tiempo consecutivos k y $k+1$, pero muy posiblemente no minimal. Esto lógicamente dependerá de lo buena que sea la aproximación de partida de la estructura correspondiente al intervalo inicial de tiempo $k=0$.

De todas formas, existe un problema adicional en este esquema de aprendizaje de RCD y el aprendizaje del instante inicial de tiempo $k=0$. Si para su aprendizaje utilizamos un algoritmo, como puede ser el algoritmo PC, o en los métodos basados en métricas utilizamos una de ellas con la propiedad de ser "score-equivalente", entonces hemos de notar que, normalmente, la salida será un representante de la clase de equivalencia de todos los modelos isomorfos a esta salida. Esto en principio podría parecer irrelevante para el problema que estamos resolviendo, ya que se representa la misma factorización en todos los modelos isomorfos. Sin embargo, como en el esquema general de aprendizaje, esta misma estructura se utiliza para ser duplicada para el resto de periodos de tiempo, entonces la elección del modelo para representar la estructura "estática" de la RCD de entre todos los modelos isomorfos, podrá tener influencia en las distintas salidas en el conjunto de relaciones temporales al ser aplicados los algoritmos para su aprendizaje estudiados en el capítulo anterior. Veamos esta cuestión con el siguiente ejemplo.

Ejemplo 5.1 *Suponemos que tenemos el grafo dirigido acíclico de la figura 5.1(a) para el instante inicial de tiempo* 0*. El algoritmo PC recuperaría el patrón de la figura 5.1(b)*



(a) Grafo dirigido aciclico G(0) (b) Patron para G(0) (c) Patron completo para G(0)

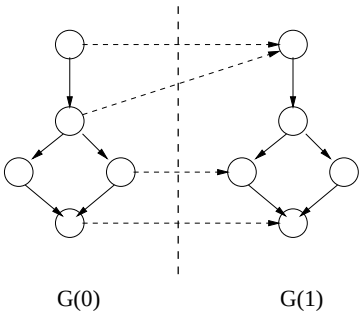
Figura 5.1: Patrón de salida de PC y una vez completadas las orientaciones en el patrón.

Este patrón se podrá completar con cualquier orientación de los arcos sin orientar siempre y cuando no se formen nuevos nodos cabeza-cabeza no acoplados ni tampoco ciclos dirigidos, como se puede observar en la figura 5.1(c).

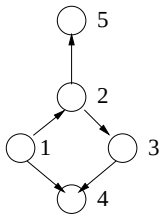
■

Veamos ahora el comportamiento que tendrán los algoritmos ART y FART en este tipo de situaciones. El algoritmo TeRePC y los basados en los anteriores también tendrán un comportamiento bastante similar.

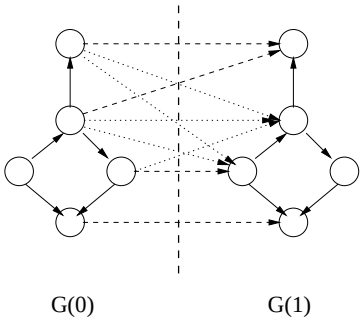
Ejemplo 5.2 *Suponemos que tenemos la red de creencia dinámica de la figura 5.2(a). Ejecutamos el algoritmo PC para recuperar el patrón de $G(0)$, completándolo como aparece el la figura 5.2(b). El orden ancestral fijado para la ejecución posterior del algoritmo ART y FART queda como en la numeración que aparece en la figura 5.2(b). Duplicamos el patrón completo para $G(1)$ y ejecutamos el algoritmo ART. Como resultado queda el que aparece en la figura 5.2(c). El mismo resultado se hubiese producido tras la ejecución del algoritmo FART. Fundamentalmente su comportamiento se puede resumir en que cuando equivocamos una orientación en los arcos, entonces se tienden a "cruzar" los padres en el periodo de tiempo anterior de ambos nodos.*



(a) Red de creencia dinamica de partida



(b) Patron recuperado completo



(c) Red de creencia dinamica resultante

Figura 5.2: Salida del Algoritmo ART y FART para una red de creencia dinámica con el patrón completo de partida mal orientado.

■

Los algoritmos TeRePC, ARTSub y FARTSub, al utilizar subconjuntos de padres, manto de Markov $MM^{i,j}$ y mínimo d-separador respectivamente, su comportamiento también tenderá al observado en el ejemplo anterior. Sin embargo, es posible que incluyan un número menor de arcos temporales superfluos, ya que puede que en alguno de los múltiples tests que realizan se estimen independientes los nodos comprobados. Este comportamiento se debe, fundamentalmente, a que puede que no se instancien nodos cabeza-cabeza en los correspondientes subconjuntos comprobados y que éstos sí que pertenezcan forzosamente a los subconjuntos $MM^{i,j}$ o mínimo d-separador por sus respectivas definiciones y estos nodos cabeza-cabeza hagan que el camino correspondiente quede bloqueado.

Hemos de tener en cuenta que aunque el algoritmo TeReK2 y el TeReBenedict se basan en una búsqueda heurística, también tendrán una tendencia hacia el mismo comportamiento descrito en el ejemplo anterior.

Es claro que este tipo se situaciones se resolverían de una forma correcta si partimos de que el conjunto $V(k)$ de variables posee una ordenación causal previa entre sus elementos. Hemos de notar también que en situaciones reales es muy habitual que el conjunto de relaciones no temporales, entre variables del mismo periodo de tiempo, sea muy escaso, incluso en ocasiones vacío, como describíamos en capítulos anteriores. Por consiguiente, será más probable que este orden entre las variables sea más fácil de obtener.

Otra solución pasaría por comprobar todos los modelos equivalentes para $G(k)$ y quedarnos con aquél que nos ofreciera un número menor de enlaces temporales. Esta solución podría ser muy ineficiente si el número de modelos equivalentes fuese elevado.

Una posible solución eficiente podría ser movernos de forma local en los modelos equivalentes. El método consistiría en detectar un arco inversible[3]; invertirlo; reaprender los arcos temporales solamente para los nodos que conectan el arco invertido; y si el número de relaciones temporales se reduce, entonces quedarnos con esta orientación como correcta; y si el número de relaciones temporales aumenta, entonces quedarnos con la dirección primitiva como correcta. Hemos de tener en cuenta que el fijar un arco con una orientación definitiva, hará que algunos arcos más, que en principio son inversibles, queden determinados a partir de entonces.

Nosotros conjeturamos que para los algoritmos ART y FART esta forma de proceder sería correcta y por tanto ofrecería unos resultados teóricamente correctos.

[3]Un arco inversible es cuando el conjunto de padres de los extremos de los nodos es el mismo, salvo el nodo cola del arco. Esto es, si tenemos $x_i(k) \rightarrow x_j(k) \in G(k)$, entonces es inversible si y solo si $\pi_k(x_j(k)) \setminus \{x_i(k)\} = \pi_k(x_i(k))$ [28].

Para ello deberemos caracterizar cuándo un arco temporal es introducido de forma superflua en el modelo resultante. Esta cuestión no es ni mucho menos trivial, y la dejamos abierta para el futuro. También conjeturamos que para los algoritmos ARTSub, FARTSub y TeRePC, el método propuesto también sería correcto, sin embargo la caracterización de los arcos temporales superfluos debe ser diferente.

Sin embargo, si utilizamos el algoritmo TeReK2 como método de búsqueda de relaciones temporales y utilizamos una función descomponible como función de ajuste, entonces el algoritmo de inversión de arcos previamente descrito obtendría buenos resultados (no podemos asegurar su correcto funcionamiento por el carácter heurístico de la búsqueda). Este hecho se fundamenta en que al ser descomponible la función de ajuste y partiendo de que el modelo para $G(k)$ es un modelo isomorfo al verdadero, entonces tan solo bastaría buscar de nuevo los conjuntos de padres para los nodos cabeza y cola del arco invertido en el esquema anterior. De todas formas es un método que habría que validar experimentalmente en trabajos futuros y así comprobar su eficacia.

5.3 Aprendizaje de RCD cuando no se conoce el instante $t = 0$

Como pone de manifiesto Friedman y col. en su trabajo [61], puede ocurrir que los datos de partida para la realización del aprendizaje de una RCD no sean observaciones que partan desde el inicio del proceso dinámico, sino que dicho proceso se haya empezado a observar desde un instante de tiempo k. En la sección anterior hemos supuesto que este instante de tiempo $k = 0$ era conocido y nos basábamos en gran medida en este conocimiento para el desarrollo del esquema general de aprendizaje de una RCD.

Si empezamos a observar una sistema dinámico a partir de un cierto instante de tiempo k y utilizamos éste como si fuese realmente el instante de tiempo inicial, en el sentido expresado en la sección anterior, entonces muy probablemente se inducirán una serie de arcos, relaciones superfluas, entre las variables del mismo periodo de tiempo. Estas relaciones serán inducidas debido a la existencia de relaciones no directas entre estas variables que involucran a variables del periodo de tiempo anterior, esto es, existirán caminos no bloqueados que conecten este par de variables y que contengan variables del periodo de tiempo anterior no observado. A continuación expondremos un ejemplo de esta situación.

Ejemplo 5.3 *Supongamos que tenemos la RCD descrita en la figura 5.3(a). Si aprendemos mediante el algoritmo PC la red para el instante de tiempo k, entonces su salida será la descrita en la figura 5.3(b).*

■

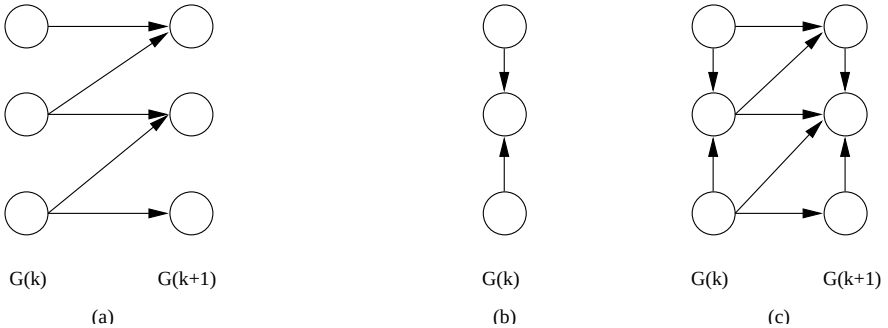


Figura 5.3: Salida del algoritmo PC para el aprendizaje de $G(k)$. Salida del modelo $G(k,k+1)$ utilizando el esquema general de aprendizaje de RCD y el algoritmo ART o FART.

Si utilizamos para el aprendizaje de RCD el esquema descrito en la sección anterior (podemos observar el resultado en la figura 5.3(c)), entonces habrá que tener en cuenta que probablemente estemos considerando un I-map del modelo $G(k)$ no minimal. La solución a este problema es una cuestión abierta que ha de ser objeto de estudio en un futuro. De todas formas se podría aliviar este problema, incorporando una poda al resultado de la ejecución del esquema de aprendizaje anterior de tal forma que tengamos en cuenta estas situaciones. Otra solución es no tener en cuenta este esquema y considerar todas las variables de $V(k,k+1)$ para el aprendizaje. Esta cuestión se aborda en la sección siguiente.

Como comentábamos en el párrafo anterior, se podría plantear un algoritmo de refinamiento del resultado, en la misma filosofía que en el capítulo anterior, una vez que hemos aprendido el modelo $G(k,k+1)$ con el método propuesto anteriormente. Este refinamiento se podría realizar de dos formas: (1) Planteándose de nuevo todos los enlaces $x_i(k+1) \rightarrow x_j(k+1)$ mediante tests de independencia condicional, en este caso de nuevo podríamos recurrir a realizar estos tests con el conjunto mínimo d-separador. (2) O bien, hacer un refinamiento mediante una búsqueda local utilizando para ello una función de ajuste descomponible. Esta opción implica también la elección de los operadores de búsqueda para encontrar un nuevo conjunto de padres para cada variable. En este segundo caso no nos vamos a centrar, porque se podría utilizar la vecindad en grafos dirigidos acíclicos que describiremos en el método de ascensión de colinas de la sección siguiente. En dicho caso, tan sólo hemos de utilizar la salida de este esquema como punto de inicio de la búsqueda planteada en la sección siguiente, es decir, aprenderemos el modelo $G(k)$ mediante un algoritmo de aprendizaje de redes de creencia estáticas; duplicamos este modelo para $G(k+1)$; y aplicamos algún algoritmo de aprendizaje de relaciones tempora-

les. Una vez tengamos el modelo $G(k,k+1)$ utilizamos éste como punto de inicio de la búsqueda.

Por tanto, nos centraremos en cómo realizar el refinamiento de este modelo aproximado $G(k,k+1)$ mediante tests de independencia condicional y conjuntos mínimos d-separadores. Para llevarlo a cabo, en primer lugar nos replantearemos las relaciones no temporales, para posteriormente replantearse de nuevo las relaciones temporales. El algoritmo de refinamiento será bastante simple:

- Una vez tengamos el modelo completo $G(k,k+1)$, ordenamos las variables en orden topológico.

- Para cada enlace de $G(k+1)$, $x_i(k+1) \rightarrow x_j(k+1)$, ordenados por el índice i, según el orden topológico anterior: Borramos transitoriamente este enlace y también su correspondiente en $G(k)$, $x_i(k) \rightarrow x_j(k)$; calculamos el conjunto mínimo d-separador S_d; si los nodos son independientes dado este subconjunto, continuamos con el siguiente enlace; si no lo son, entonces reponemos estos enlaces. Continuaremos hasta que hayamos revisado todos los enlaces.

- Una vez revisados todos los enlaces no temporales en el paso anterior, revisaremos las relaciones temporales $x_i(k) \rightarrow x_j(k+1)$, también según el orden topológico. Para este caso, también utilizamos los tests de independencia condicional con el conjunto mínimo d-separador y con los mismos criterios que en el caso anterior. Continuaremos hasta que hayamos revisado todos los enlaces temporales.

Hemos de notar que en realidad el orden de comprobación es arbitrario. Sin embargo, nosotros hemos elegido un orden ancestral, empezando por los nodo raíces, porque recordemos que el conjunto mínimo d-separador se encuentra entre los ancestros de los nodos a separar. Por consiguiente, empezando por los nodos raíces, los cambios en esta estructura afectarán a los posteriores cálculos de los conjuntos d-separadores para los nodos posteriores, y probablemente con un número menor de arcos y por tanto un número menor de caminos en la comprobación.

Ejemplo 5.4 *Siguiendo con el ejemplo anterior, la salida del refinamiento propuesto será precisamente el modelo buscado, esto es, el modelo mostrado en la figura 5.3(a). Sin embargo, en general, este comportamiento no tendrá por qué darse siempre para todos los modelos aprendidos.*

■

5.4 Aprendizaje de RCD basado en algoritmos para el aprendizaje de RC estáticas

En esta sección nos dedicaremos a describir cómo se realizaría el aprendizaje de RCD, tanto basado en tests de independencia condicional como basado en métodos locales de búsqueda dada una función de ajuste, de forma conjunta, esto es, sin hacer distinción entre las relaciones no temporales y las relaciones temporales, como hemos venido describiendo a lo largo de este capítulo. En la literatura especializada en aprendizaje de redes de creencia estáticas existen una gran variedad de métodos tanto basados en relaciones de independencia como basados en métricas. Nosotros, en este capítulo, realizaremos un estudio general del comportamiento de estos algoritmos cuando han de ser aplicados al aprendizaje de RCD para modelar un sistema dinámico markoviano y estacionario. En primer lugar nos centraremos en los algoritmos basados en tests de independencia condicional y posteriormente nos centraremos en los algoritmos de búsqueda local basados en métricas.

5.4.1 Algoritmos basados en tests de independencia condicional para el aprendizaje de RCD

En este punto describiremos cómo se han de adaptar los métodos basados en tests de independencia condicional para poder ser aplicados al aprendizaje de RCD. Para ello recordemos la primera proposición que comentábamos en el capítulo anterior:

Proposición 5.2 *Sea una RCD markoviana y estacionaria, y sean dos periodos de tiempo consecutivos cualesquiera $G(k,k+1)$, entonces si se cumple para algún subconjunto $S \in V$ que $I(x_j(k), x_i(k+1)|S)$, entonces podremos encontrar un subconjunto S' tal que se siga cumpliendo $I(x_j(k), x_i(k+1)|S')$ y $S' \subseteq V(k,k+1)$.*

Veíamos que esta proposición era consecuencia de la propiedad de Markov que cumplía el sistema dinámico que queremos modelar. Otra consecuencia de esta propiedad es la siguiente, ampliación de la anterior proposición:

Proposición 5.3 *Sea una RCD markoviana y estacionaria, y sean dos periodos de tiempo consecutivos cualesquiera $G(k,k+1)$, entonces si se cumple para algún subconjunto $S \in V$ que $I(x_j(k+1), x_i(k+1)|S)$, entonces podremos encontrar un subconjunto S' tal que se siga cumpliendo $I(x_j(k+1), x_i(k+1)|S')$ y $S' \subseteq V(k,k+1)$.*

Estas dos propiedades nos aseguran que los subconjuntos que hacen independientes dos variables cualesquiera del modelo se han de encontrar en su propio intervalo de tiempo y en el inmediatamente anterior. Esto quiere decir, evidentemente, que cualquier algoritmo basado en tests de independencia condicional se puede centrar tan sólo en dos periodos de tiempo consecutivos para aprender el modelo

correspondiente a un determinado periodo de tiempo. Esto es, si nosotros queremos aprender el modelo para $G(k,k+1)$, tan sólo hará falta considerar las variables correspondientes a estos dos periodos de tiempo.

Además si el modelo es estacionario, como es nuestro caso, entonces estamos seguros de que si aprendemos el modelo en $G(k,k+1)$, éste de replicará para todos y cada uno de los periodos de tiempo.

La consecuencia de todo lo que hemos venido comentando es que cualquier algoritmo de aprendizaje que quiera aprender el modelo en $G(k,k+1)$, podrá aplicar las siguientes restricciones en su método de búsqueda: En primer lugar para encontrar las relaciones no temporales debe tenerse en cuenta tan sólo los posibles tests entre pares de variables de $V(k+1)$, y también estos tests bastaría hacerlos condicionando sobre subconjuntos pertenecientes a $V(k,k+1)$. Por otra parte, si encontramos una relación de independencia/dependencia en $G(k+1)$, ésta debe ser replicada en $G(k)$, para cualquier k. En cuanto a las relaciones temporales hemos de decir lo mismo que en el capítulo anterior, esto es, nos basta realizar tests entre parejas de variables de periodos de tiempo consecutivos y escoger subconjuntos de condicionamiento como en el caso anterior. Otra consideración que debemos realizar es que si encontramos una relación de dependencia entre dos variables $x_i(k)$ y $x_j(k+1)$ su orientación debe ser naturalmente $x_i(k) \rightarrow x_j(k+1)$. A continuación particularizaremos todos estos conceptos adaptando el algoritmo PC para la búsqueda del modelo $G(k,k+1)$ de una RCD markoviana y estacionaria. Esta adaptación se puede observar en la figura 5.4.

Si observamos la adaptación del algoritmo PC para el aprendizaje de RCD descrito en la figura 5.4, podremos notar que la adaptación impone las siguientes restricciones:

- Los pares de variables comprobados son o bien parejas de variables de $V(k+1)$, $(x_i(k+1), x_j(k+1))$, o bien parejas de variables entre los periodos de tiempo consecutivos $(x_i(k), x_j(k+1))$. Fijémonos que si encontramos que una de las parejas del primer tipo están d-separadas por algún subconjunto de $V(k,k+1)$, entonces éstas también estarán d-separadas en cualquier periodo de tiempo. El segundo tipo de parejas de variables, al igual que el algoritmo TeRePC descrito en el capítulo anterior, sirve para buscar las relaciones temporales del modelo.

- Cómo hemos descrito en el punto anterior, si encontramos d-separadas una pareja de variables $(x_i(k+1), x_j(k+1))$, entonces también borramos el enlace correspondiente a las parejas de $V(k)$, $(x_i(k), x_j(k))$.

- En los pasos para la orientación de los enlaces encontrados en los pasos anteriores, después de encontrar los nodos cabeza-cabeza, completamos todas las orientaciones correspondientes a los arcos temporales. Posteriormente se

Algoritmo PCRCD

1. *Formar el grafo $G(k,k+1)$ no dirigido sobre el conjunto de vértices $V(k,k+1)$, con $E = V(k,k+1) \times V(k,k+1)$.*
2. $n = 0$

   Repetir

   - *Repetir*

     - Seleccionar un par ordenado de variables $(x_i(k), x_j(k+1))$ ó $(x_i(k+1), x_j(k+1))$ que sean adyacentes en $G(k,k+1)$, tal que $Adyac(G(k,k+1), x_i) \setminus \{x_j\}$ tiene un cardinal mayor o igual a n, y un subconjunto S de $Adyac(G(k,k+1), x_i) \setminus \{x_j\}$ de cardinal n. Si x_i y x_j están d-separadas dado S, borrar el arco $x_i - x_j$; si $x_i, x_j \in V(k+1)$ entonces también borrar el arco en G(k) de $G(k,k+1)$; y guardar S en $SEPSET(x_i, x_j)$ y $SEPSET(x_j, x_i)$.

     Hasta

     - Todos los pares ordenados de variables adyacentes de x_i y x_j, tal que $Adyac(G(k,k+1), x_i) \setminus \{x_j\}$ sea de cardinalidad n han sido ya comprobados anteriormente.

     $n = n+1$

   Hasta

   - Para cada par ordenado de variables adyacentes x_i y x_j, $Adyac(G(k,k+1), x_i) \setminus \{x_j\}$ tiene una cardinalidad menor que n.
3. *Sea $G'(k,k+1)$ el grafo resultado de la etapa anterior. Para cada triple de vértices (x,y,z) tal que el par (x,y) y el par (y,z) son adyacentes en $G'(k,k+1)$, pero el par (x,z) no es adyacente en $G'(k,k+1)$, orientar $x - y - z$ como $x \rightarrow y \leftarrow z$ si y solo si y no está en $SEPSET(x,z)$.*
4. *Para cada par $(x_i(k), x_j(k+1))$ orientar $x_i(k) \rightarrow x_j(k+1)$.*
5. *Repetir*

   - Si $x_i(k+1) \rightarrow x_j(k+1)$, $x_j(k+1) - x_l(k+1)$ y $x_i(k+1)$ y $x_l(k+1)$ no son adyacentes y no hay cabeza de flecha en $x_j(k+1)$, entonces orientar $x_j(k+1) - x_l(k+1)$ como $x_j(k+1) \rightarrow x_l(k+1)$.

   - Si hay un camino dirigido desde $x_i(k+1)$ hasta $x_j(k+1)$ y un enlace entre $x_i(k+1)$ y $x_j(k+1)$, entonces orientar $x_i(k+1) \rightarrow x_j(k+1)$.

   Hasta que que ningún arco más se pueda orientar.

Figura 5.4: Algoritmo PCRCD

sigue orientando con las reglas marcadas por el algoritmo PC. En este caso, también hemos de tener en cuenta que si orientamos una pareja de variables $x_i(k+1) \rightarrow x_j(k+1)$, entonces también hemos de orientar la pareja correspondiente en $V(k)$, $x_i(k) \rightarrow x_j(k)$. También hemos de notar que las reglas de orientación siempre se han de aplicar exclusivamente a parejas de $V(k+1)$, $(x_i(k+1), x_j(k+1))$, ya que el modelo en $V(k)$ no tiene porqué ser completo, en el sentido de que le faltan las posibles relaciones de sus variables con el periodo anterior $V(k-1)$.

5.4.2 Algoritmos locales de búsqueda para el aprendizaje de RCD

En este punto nos dedicaremos a ver cómo podemos adaptar una búsqueda local, tanto en el espacio de grafos dirigidos acíclicos como en el espacio de secuencias de ordenación, para el aprendizaje de una RCD mediante una función o métrica descomponible.

Nuestro objetivo es realizar un algoritmo de ascensión de colinas, al igual que en el capítulo 2, para aprender un modelo de RCD. Esta búsqueda local se realizará tanto en el espacio de grafos dirigidos acíclicos como en el espacio de secuencias de ordenación. Para poder realizar esta tarea hemos de definir la vecindad que utilizaremos en cada caso.

Tanto en un caso como en el otro, debemos suponer que tenemos definida una función o métrica descomponible $f(G(k,k+1))$, hemos de notar que entonces, al igual que describíamos en el anterior capítulo, al tener un modelo dinámico markoviano y estacionario, podemos maximizar cada familia $(x_i(k+1) \cup \pi(x_i(k+1))$, centrados en $V(k,k+1)$. De esta forma, si encontramos el mejor conjunto de padres para un nodo $x_i(k+1)$, éstos también serán los mejores para este nodo en todos los intervalos de tiempo k.

Ascensión de colinas en el espacio de grafos dirigidos acíclicos. En este caso tendremos que definir la vecindad correspondiente a un determinado grafo $G(k,k+1)$ para recorrer el espacio de grafos dirigidos acíclicos de forma local. Evidentemente esta vecindad será la misma que en el caso de redes de creencia estáticas, pero con las restricciones lógicas que impone el modelo de RCD que queremos aprender. Estas restricciones serán impuestas por la orientación particular que han de tener las relaciones temporales. Así pues, un relación temporal siempre tendrá la misma orientación y sólo podremos introducir nuevas relaciones temporales o borrar alguna de las que existan en el grafo actual, es decir, carece de sentido el operador de inversión de un arco temporal. Con todas estas consideraciones la vecindad $\mathcal{N}(G(k,k+1))$ será la unión de los siguientes subconjuntos:

$$\mathcal{N}_A(G(k,k+1)) = \{(V(k+1), E'(k+1)) | E'(k+1) = E(k+1) \cup \{x_j(k+1) \rightarrow x_i(k+1)\}\}$$

tal que, $x_j(k+1) \rightarrow x_i(k+1) \notin E(k+1)$ y $(V(k+1), E'(k+1))$ es un GDA.

$$\mathcal{N}_B(G(k,k+1)) = \{(V(k+1), E'(k+1)) | E'(k+1) = E(k+1) \setminus \{x_j(k+1) \rightarrow x_i(k+1)\}\}$$

tal que, $x_j(k+1) \rightarrow x_i(k+1) \in E(k+1)$.

$$\mathcal{N}_I(G(k,k+1)) =$$

$$= \{(V(k+1), E'(k+1)) | E'(k+1) = (E(k+1) \cup \{x_j(k+1) \rightarrow x_i(k+1)\}) \setminus \{x_i(k+1) \rightarrow x_j(k+1)\}\}$$

tal que, $x_i(k+1) \rightarrow x_j(k+1) \in E(k+1)$ y $(V(k+1), E'(k+1))$ es un GDA.

Estas tres anteriores son las definiciones para el añadido, borrado e inversión de arcos no temporales en $G(k+1)$, hemos de notar que es el único grafo donde puede darse un ciclo dirigido, no es posible que se den ciclos dirigidos en caminos donde se incluyan arcos temporales. A continuación definiremos las operaciones para los arcos temporales:

$$\mathcal{N}_{AT}(G(k,k+1)) = \{(V(k,k+1), E'(k,k+1)) | E^{tmp'}(k+1) = E^{tmp}(k+1) \cup \{x_j(k) \rightarrow x_i(k+1)\}\}$$

tal que, $x_j(k) \rightarrow x_i(k+1) \notin E^{tmp}(k+1)$.

$$\mathcal{N}_{BT}(G(k,k+1)) = \{(V(k,k+1), E'(k,k+1)) | E^{tmp'}(k+1) = E^{tmp}(k+1) \setminus \{x_j(k) \rightarrow x_i(k+1)\}\}$$

tal que, $x_j(k) \rightarrow x_i(k+1) \in E^{tmp}(k+1)$.

Es decir, tan sólo modificamos el conjunto de relaciones temporales $E^{tmp}(k+1)$ añadiendo una relación no existente o borrando una relación existente en este subconjunto.

Ascensión de colinas en el espacio de órdenes. Siguiendo todo lo estudiado en el capítulo 2 para el aprendizaje de redes de creencia estáticas mediante búsquedas locales en el espacio de órdenes, podemos adaptar fácilmente esta búsqueda al aprendizaje del modelo $G(k,k+1)$ de una RCD. Esta adaptación supondrá una restricción en las secuencias de ordenación visitadas durante la búsqueda. Esta restricción la impone la ordenación temporal de las variables, esto es, las variables de $V(k)$ serán menores, en cualquier secuencia de ordenación, que las variables de $V(k+1)$. Fijémonos pues, que una secuencia $\sigma = \sigma^k \cdot \sigma^{k+1}$ será la composición de una ordenación σ^k sobre las variables de $V(k)$ y una ordenación σ^{k+1} para las variables de $V(k+1)$.

Por otra parte, como hemos comentado más de una vez en esta memoria, al tener una secuencia de ordenación, podemos optimizar de forma independiente cada familia de variables. En nuestro caso, esta consideración nos va a indicar que tan sólo debemos encontrar padres, a partir de las anteriores variables en la secuencia σ, para las variables de $V(k+1)$, pues una vez encontrados los padres óptimos para cada variable, éstos se deben repetir para cada instante de tiempo. Esto tiene

una incidencia directa en la definición de vecindad para una búsqueda local, ya que entonces, tan sólo debemos movernos en las secuencias locales en σ^{k+1} de forma tal como lo hacíamos en el capítulo 2. Esta definición, por consiguiente, quedaría de la siguiente forma:

$$\sigma_l = \sigma^k \cdot (x_1(k+1), \ldots, x_i(k+1), \ldots, x_j(k+1), \ldots, x_n(k+1)) \rightarrow$$

$$\rightarrow \sigma_{l'} = \sigma^k \cdot (x_1(k+1), \ldots, x_j(k+1), \ldots, x_i(k+1), \ldots, x_n(k+1))$$

Evidentemente la forma de evaluar cada secuencia de ordenación será la misma que planteábamos en el capítulo 2.

Con estas definiciones de vecindad se podrán llevar a cabo, sin apenas modificaciones, métodos de búsqueda local tanto para el espacio de grafos dirigidos acíclicos como en el espacio de órdenes. Además de utilizar un método de ascensión de colinas también se podrán utilizar otros métodos locales más sofisticados como puede ser la heurística VNS.

Conclusiones y Trabajos Futuros

Los objetivos principales de esta memoria, descritos en el capítulo de introducción, eran: por un lado estudiar y desarrollar métodos de búsqueda local y distribuidos para el aprendizaje de redes de creencia y por otro estudiar y desarrollar nuevos métodos para el aprendizaje de redes de creencia dinámicas. Estos objetivos han dando lugar a los capítulos posteriores de esta memoria. De estos capítulos podemos extraer una serie de conclusiones que resumiremos en los siguientes puntos:

- En el segundo capítulo se ha realizado un estudio de métodos locales de búsqueda para el problema del aprendizaje de redes de creencia, tanto en el espacio de grafos dirigidos acíclicos como en el espacio de órdenes. De este estudio podemos extraer las siguientes conclusiones:
  - Hemos desarrollado un nuevo método basado en búsqueda local y múltiples reinicios, el algoritmo IMAPR, en donde los reinicios se realizan de una manera informada; de tal forma que estamos libres de ajustar parámetros difíciles tales como el número de perturbaciones o el número de padres máximo para un determinado nodo, que en un esquema de múltiples reinicios aleatorios deberíamos de ajustar en el proceso de búsqueda. Este nuevo método se ha mostrado eficaz para escapar de los óptimos locales alcanzados durante una búsqueda local de ascensión de colinas. De todas formas hemos notado que este esquema de reinicios puede entrar en ciclos de difícil salida, esto es, se llega durante varias iteraciones del proceso al mismo óptimo local. Esta situación puede ser debida a que el proceso de reinicio encuentre I-maps con muy buenas características, y por tanto el proceso posterior de búsqueda local queda atrapado en el "valle" correspondiente.
  - Hemos adaptado la metaheurística VNS al problema del aprendizaje de redes de creencia en el espacio de grafos dirigidos acíclicos, desarrollando el algoritmo VNSST. Hemos diseñado una transformación específica

para este esquema basada en la semántica que tiene una red de creencia. De los resultados obtenidos en la experimentación realizada se puede concluir que este método es muy eficaz y que se mejoran de forma muy significativa los resultados ofrecidos por un esquema de reinicios aleatorios. La transformación específica también mejora los resultados con respecto al esquema VNS sin ella, sobre todo cuando se eligen unos parámetros para el método VNS más eficientes.

  - Hemos diseñado los elementos necesarios para realizar el aprendizaje de redes de creencia en el espacio de órdenes mediante una búsqueda local, definiendo para ello el operador de vecindad necesario y varios métodos de evaluación de las secuencias vecinas. El resultado es el algoritmo denominado HCSN. Una vez diseñados los elementos anteriores, también hemos adaptado el esquema VNS a este nuevo espacio de búsqueda, desarrollando el algoritmo VNSSN. Los resultados de los experimentos ponen de manifiesto la conclusión de que el espacio de órdenes es más homogéneo que el espacio de grafos dirigidos acíclicos, conclusión que apoya a anteriores trabajos de diferentes autores. En este espacio de búsqueda se visitan un número menor de individuos con una mayor calidad en su medida, pero a costa de peder eficiencia en la búsqueda. De todas formas, creemos que esta búsqueda es muy importante en aquellos tipos de aplicaciones donde se busquen diferentes redes de creencia, con significativa diferencia estructural, con alto valor de su medida, como por ejemplo en el trabajo [60].
  - Hemos diseñado un heurística eficiente para la búsqueda en el espacio de órdenes, basada en el algoritmo K2, pero sin la restricción de necesitar a priori un orden entre las variables. Este método, denominado K2SN, se puede utilizar como un punto inicial informado para búsquedas posteriores tanto en el espacio de órdenes como en el espacio de grafos dirigidos acíclicos.
  - Hemos estudiado una nueva definición de vecindad en el espacio de grafos dirigidos acíclicos, muy prometedora a la luz de los resultados previos obtenidos en la experimentación realizada. Esta definición mejora de manera muy significativa la vecindad clásica en grafos dirigidos acíclicos sin repercutir de manera excesiva en el esfuerzo computacional realizado.

- En el capítulo tercero se han desarrollado métodos distribuidos de búsqueda basados en la metaheurística VNS; también se han diseñado todos los elementos necesarios para adaptar la metaheurística del Sistema de Colonias de Hormigas al aprendizaje de redes de creencia estáticas. Del desarrollo de todos estos métodos se pueden extraer las siguientes conclusiones:

  - Hemos desarrollado varios esquemas de distribución para la búsqueda VNS, mostrándose efectivos en la búsqueda tanto en el espacio de grafos dirigidos acíclicos (algoritmo DVNSST) como en el espacio de órdenes (algoritmo DVNSSN). Para estos esquemas distribuidos, hemos desarrollado unas extensiones probabilísticas de algoritmos eficientes de inicialización en la búsqueda, tanto en el espacio de grafos dirigidos acíclicos (algoritmo B probabilístico) como en el espacio de órdenes (algoritmo K2SN probabilístico). Hemos de notar que estas inicializaciones se podrán utilizar también en otros métodos de optimización en el problema del aprendizaje de redes de creencia. También hemos de notar que las versiones distribuidas del esquema VNS desarrolladas son independientes del problema que se resuelve en esta memoria. Así pues, es factible utilizar este esquema para la optimización de otros problemas combinatorios. Por último, hemos de indicar que en un entorno paralelo estos esquemas serían más eficientes que sus correspondientes versiones no distribuidas.
  - Para la adaptación de nuestro problema de aprendizaje para el uso del Sistema de Colonias de Hormigas, hemos desarrollado todos los componentes necesarios tanto para el espacio de grafos dirigidos acíclicos como para el espacio de órdenes. Se ha mostrado especialmente eficaz este tipo de búsqueda en el espacio de grafos dirigidos acíclicos (algoritmo ACO-B) frente al espacio de órdenes (algoritmo ACO-K2SN). Los resultados ofrecidos por el sistema de colonias de hormigas en el espacio de grafos dirigidos acíclicos se pueden calificar de buenos en comparación al resto de métodos. Hemos notado que, si bien en el espacio de grafos dirigidos acíclicos las búsquedas locales posteriores no son especialmente relevantes, sí que lo son en el espacio de órdenes.
  - Para acabar el capítulo hemos desarrollado un novedoso sistema híbrido entre las búsquedas distribuidas VNS y el sistema de colonias de hormigas (algoritmos DVNSST-ACO y DVNSSN-ACO). Este sistema se ha mostrado tan eficaz como los mejores resultados en el espacio de grafos dirigidos acíclicos y más eficiente que el modelo de islas para el mismo espacio de búsqueda. También hemos de notar que este sistema de búsqueda es independiente del problema en cuestión, y por tanto se podría aplicar a otros problemas de optimización, siempre y cuando el sistema de colonias de hormigas fuese aplicable a este otro tipo de problemas.

- En el capítulo cuarto se han desarrollado algoritmos, tanto basados en tests de independencia condicional como basados en métricas más una búsqueda local de optimización, para el aprendizaje automático de las relaciones tem-

porales de una red de creencia dinámica. Las conclusiones en el desarrollo del capítulo pueden resumirse en los siguientes puntos:

  - Hemos desarrollado algoritmos de aprendizaje de relaciones temporales para redes de creencia dinámicas basados en tests de independencia condicional (algoritmos TeRePC, ART, FART, ARTSub, FARTSub), en primer lugar, caracterizando de varias formas cuándo existe una relación temporal; posteriormente hemos mejorado la eficiencia y eficacia de estos algoritmos intentando reducir en el mayor número posible el orden de los tests realizados. Los resultados obtenidos a través de los experimentos nos confirman que cuando menor sea el orden del test realizado más fiable y eficiente es éste. En particular, es de destacar el excelente comportamiento ofrecido por el algoritmo TeRePC.
  - Hemos adaptado algoritmos de búsqueda local para el aprendizaje de relaciones temporales en una red de creencia dinámica. Estos algoritmos (TeReK2 y TeReBenedict) se basan en dos métricas diferentes: una métrica descomponible y otra métrica basada en la discrepancia entre las relaciones de independencia condicional expresadas en la red candidata y éstas mismas medidas en nuestra base de datos [7]. Los resultados obtenidos por estos algoritmos son bastante prometedores, especialmente en el caso de TeReK2.

- En el último capítulo se estudia cómo realizar el aprendizaje automático de redes de creencia dinámicas para modelos dinámicos markovianos y estacionarios. Este capítulo muestra dos esquemas generales de aprendizaje automático de redes de creencia dinámicas:

  - En primer lugar hemos planteado un esquema general de aprendizaje en el supuesto de que el instante de tiempo inicial del proceso dinámico es conocido. Este esquema se basa en los algoritmos estudiados en los anteriores capítulos, utilizando un algoritmo de aprendizaje de redes de creencia estáticas y a continuación otro de aprendizaje de relaciones temporales. Hemos identificado los problemas que puede tener este esquema general y planteado posibles soluciones al mismo.
  - En segundo lugar hemos planteado los problemas que pueden surgir al aplicar el anterior esquema en el supuesto, más general, de que no sea conocido el instante inicial de tiempo. Planteamos una posible solución heurística a los problemas anteriores, basada en un refinamiento del modelo aprendido mediante tests de independencia condicional.
  - Finalmente, estudiamos métodos generales de aprendizaje de redes de creencia dinámicas, tanto basados en tests de independencia condicional

como basados en una función de ajuste descomponible y un método de búsqueda local. Estos esquemas no hacen distinción explícita entre las relaciones temporales y no temporales, como se ha realizado en el esquema anterior.

Trabajos Futuros

Considerando los resultados obtenidos, los trabajos futuros se pueden desglosar en los siguientes puntos, ordenados por capítulos:

- Durante el desarrollo de la investigación llevada a cabo en el segundo capítulo, hemos notado que se puede seguir esta línea de investigación en los siguientes aspectos:
  - En cuanto al algoritmo IMAPR, podríamos diseñar nuevos esquemas de transformación del grafo resultante pensados específicamente para redes de creencia, en el caso de que entráramos de una forma clara en un ciclo. Este esquema de transformación podría ir en la línea del diseñado para el método VNS en el espacio de grafos dirigidos acíclicos, esto es, cuando notásemos que hemos entrado en un ciclo (varias iteraciones llegan al mismo óptimo local), entonces identificar un cierto número de "cliques" y romperlos de alguna forma informada o totalmente de forma aleatoria.
  - Hemos de comprobar el comportamiento que pueden tener otros tipos de transformaciones, pensadas para redes de creencia, en el esquema VNS. También hemos de adaptar otros extensiones del VNS, así como comprobar la eficacia y eficiencia que puede tener la redefinición de vecindad efectuada en la última sección del capítulo como búsqueda local en el esquema VNS.
  - En cuanto a la redefinición de vecindad llevada a cabo en la última sección del capítulo segundo, hemos de realizar un mayor número de experimentos para confirmar las conclusiones obtenidas. Además se deberán estudiar otras variantes de la vecindad diseñada en esta sección, como puede ser en lugar de utilizar la unión de padres, utilizar la intersección de los mismos, o invertir si y sólo si el conjunto de padres de los nodos extremos del enlace a invertir son comunes, esto es, el arco es inversible [28]. Estas variantes, en principio, pensamos que mejorarán la eficiencia del método propuesto, aunque quizás pierdan algo de eficacia.
  - Otra línea de trabajo puede ser el desarrollo de métodos de búsqueda en otros espacios de búsqueda, como el de clases de equivalencia de grafos dirigidos acíclicos, o el de grafos cordales.

- Durante el desarrollo del tercer capítulo hemos notado que se puede seguir esta línea de investigación en los siguientes aspectos:
  - Proponer otros esquemas de migración y selección en el modelo de islas del método VNS distribuido, así como la posibilidad de utilizar otras extensiones del VNS. Implementación y desarrollo de este método en un verdadero entorno paralelo. Este último trabajo es compartible por todos los métodos desarrollados en este capítulo.
  - Comprobar y desarrollar otras alternativas en el Sistema de Colonias de Hormigas, como por ejemplo el Sistema Híbrido de Hormigas, así como la utilización de otros algoritmos como parte heurística del conocimiento específico al problema. En este sentido se prodría plantear el utilizar una ascensión de colinas utilizando la vecindad definida en la última sección del capítulo anterior. También deberemos realizar un estudio profundo del comportamiento de la matriz de feromona en el tipo de problemas que estamos resolviendo para así determinar los aspectos mejorables del sistema.
  - Parece una buena alternativa para nuestro problema el utilizar el sistema de refuerzo de feromona para capturar las partes más “probables” en el espacio de búsqueda. Podría ser prometedor plantear hibridar otros sistemas de optimización, como lo hemos hecho en esta memoria con el VNS.
  - Otra opción interesante podría ser la utilización de diferentes tipos de hormigas dentro del mismo proceso de búsqueda.
- En relación al tema de aprendizaje de redes de creencia dinámicas, creemos que nuestra investigación puede enfocarse hacia las siguientes líneas:
  - Validar tanto teórica como experimentalmente los métodos propuestos para el aprendizaje de redes de creencia dinámicas cuando el instante de tiempo inicial es conocido.
  - Validar de forma teórica y experimental los métodos propuestos en el mismo problema pero cuando el instante de tiempo inicial no es conocido.
  - La metodología de aprendizaje de redes dinámicas basada en aprender por un lado los modelos no temporales y después las relaciones temporales, podría utilizarse también en el aprendizaje de redes estáticas, siempre que se pueda descomponer el conjunto de variables en dos subconjuntos, de tal modo que las variables de un subconjunto sean siempre anteriores a las del otro.

  - Desarrollar los métodos de búsqueda local propuestos en el aprendizaje de redes de creencia dinámicas sin distinguir relaciones temporales y no temporales, así como otros métodos más sofisticados de búsqueda basados en los anteriores, como pueden ser VNS, Tabú, etc.

- Por último, al margen de la estructura seguida en esta memoria, también proponemos las siguientes líneas de investigación con el objetivo de poder construir una herramienta de *Minería de Datos* basada en redes de creencia:

  - Estudiar cómo incorporar el conocimiento explícito de un experto aportado a la luz de las relaciones de dependencia/independencia conseguidas durante un proceso de aprendizaje. De tal forma que el experto nos guíe en posteriores etapas del aprendizaje y así poder colaborar para conseguir un modelo satisfactorio. Para realizar esta tarea es imprescindible poder ofrecerle al experto una salida tal que le ayude a tomar sus decisiones sobre las relaciones de dependencia/independencia ofrecidas por los datos.
  - Estudiar cómo poder tratar con redes híbridas, en donde tengamos tanto variables continuas como varibles discretas. En este sentido, creemos que una línea a seguir sería la de discretizar las variables continuas mediante algún método global incluido en el propio proceso de aprendizaje.
  - Incorporar todo el software desarrollado durante la realización de esta memoria en el entorno de desarrollo de sistemas expertos probabilísticos ELVIRA [117].

Bibliografía

[1] Abramson, B. y A.J.Finizza, "Using belief networks to forecast oil prices", *International Journal of Forecasting*, tomo 7, págs. 299–316, 1991.

[2] Acid, S., *Métodos de Aprendizaje de Redes de Creencia. Aplicación a la clasificación*, Tesis Doctoral, Dpto de Ciencias de la Computación e I.A. Universidad de Granada, 1999.

[3] Acid, S., Campos, L., González, A., Molina, R. y de la Blanca, N. P., "CASTLE: A tool for bayesian learning", en *Proceedings of the ESPRIT 91 Conference, Commission of the European Communities*, págs. 363–377, 1991.

[4] Acid, S. y de Campos, L., "An algorithm for finding minimum d-separating sets in belief networks", en *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, págs. 3–10, Morgan Kaufmann, San Mateo, 1996.

[5] Acid, S. y de Campos, L., "Benedict: an algorithm for learning probabilistic belief networks", en *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems, IPMU'96*, págs. 979–984, 1996.

[6] Acid, S. y de Campos, L., "An algorithm for learning probabilistic belief networks using minimun d-separating sets", Inf. Téc. DECSAI-00001, Dpto de Ciencias de la Computación – Universidad de Granada, 2001, sometido a Journal of Artificial Intelligence Research.

[7] Acid, S. y de Campos, L., "A Hybrid methodology for learning belief nertworks: BENEDICT", *International Journal of Approximate Reasoning*, tomo 27, nº 3, págs. 235–262, 2001.

[8] Acid, S., de Campos, L. y Huete, J., "The search of causal orderings: A short cut for learning belief networks", en *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'2001)*, págs. 216–227, LNAI 2143 – Springer, 2001.

[9] Aliferis, C. y Cooper, G., "An evaluation of an algorithm for inductive learning of Bayesian belief networks using simulated data sets.", en *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, págs. 8–14, 1994.

[10] Andersen, S., Olesen, K., Jensen, F. y Jensen, F., "Hugin: a shell for buiding belief universes for expert systems", en *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, 1989.

[11] Andreassen, S., Wolbye, M., Falck, B. y Andersen., S., "Munim - a causal probabilistic network for the interpretation of electromyographic findings", en *Proceedings IJCAI'87*, págs. 366–372, 1987.

[12] Bacchus, F., "Using First-Order Probability Logic for the Construction of Bayesian Networks.", en *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, págs. 219–226, 1993.

[13] Beinlich, I., Suermondt, H., Chavez, R. y Cooper, G., "The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks", en *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, págs. 247–256, Springer-Verlag, 1989.

[14] Berler, A. y Shimony, S., "Bayes Nets for Sonar Sensor Fusion", en *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, D. Geiger y P. Shenoy, eds., págs. 14–21, Morgan & Kaufmann, 1997.

[15] Berzuini, C., Bellazzi, R. y Quaglini, S., "Temporal reasoning with probabilities", en *Uncertainty in Artificial Intelligence, 6*, págs. 14–21, North-Holland, 1989.

[16] Berzuini, C., Bellazzi, R., Quaglini, S. y Spiegelhalter, D., "Bayesian networks for patient monitoring", *Artificial Intelligence in Medicine*, tomo 4, págs. 243–260, 1992.

[17] Binder, J., Koller, D., Russell, S. y Kanazawa, K., "Adaptive probabilistic networks with hidden variables", *Machine Learning*, tomo 29, 1997.

[18] Bonabeau, E., Dorigo, M. y Theraulaz, G., *Swarm Intelligence: From Natural to Artificial Systems*, Oxford, 1999.

[19] Bouckaert, R., "Belief networks construction using the minimum description length principle", en *Proceedings ECSQARU93*, págs. 41–48, 1993.

[20] Bouckaert, R., "Properties of Bayesian belief networks learning algorithms", en *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, págs. 102–109, 1994.

[21] Bouckaert, R. R., *Bayesian Belief Networks: From Construction to Inference*, Tesis Doctoral, University of Utrecht, 1995.

[22] Boyen, X. y Koller, D., "Tractable inference for complex stochastic processes", en *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, págs. 33–42, Morgan Kaurmann, San Mateo, 1998.

[23] Buntine, W., "Theory refinement on Bayesian networks", en *Proceedings of the 7th Conference on Uncertainty in Artificial Intelligence*, págs. 52–60, Morgan Kaufmann, San Mateo, 1991.

[24] Buntine, W., "Operations for learning with graphical models", *Journal of Artificial Intelligence Research*, tomo 2, págs. 159–225, 1994.

[25] Buntine, W., "A guide to the literature on learning probabilistic networks from data", *IEEE Transactions on Knowledge and Data Engineering*, tomo 8, págs. 195–210, 1996.

[26] Cano, A., Moral, S. y Salmerón, A., "Penniless propagation in join trees", *International Journal of Intelligent Systems*, tomo 15, págs. 1027–1059, 2000.

[27] Castillo, E., Gutiérrez, J. y Hadi, A., *Sistemas Expertos y Modelos de Redes Probabilísticas*, Monografías de la Academia de Ingeniería, 1997.

[28] Chickering, D., "Learning equivalence classes of Bayesian networks structure", en *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, págs. 150–157, Morgan Kaufmann, San Mateo, 1996.

[29] Chickering, D., Geiger, D. y Heckerman, D., "Learning Bayesian networks is NP-hard", Inf. Téc. MSR-TR-94-17, Universidad de California, 1994.

[30] Chow, C. y Liu, C., "Approximating discrete probability distributions with dependence trees", *IEEE transactions on Information Theory*, tomo 14, págs. 462–467, 1968.

[31] Cooper, G. y Herskovits, E., "A Bayesian method for the induction of probabilistic networks from data", *Machine Learning*, tomo 9, nº 4, págs. 309–348, 1992.

[32] Dagum, P. y Galper, A., "Forecasting sleep apnea with dynamic network models", en *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, págs. 64–71, Morgan Kaurmann, San Mateo, 1993.

[33] Dagum, P., Galper, A. y Horvitz, E., "Dynamic network models for forecasting", en *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence*, págs. 41–48, Morgan Kaurmann, San Mateo, 1992.

[34] Dash, D. y Druzdel, M., "A hybrid anytime algorithm for the construction of causal models from sparse data", en *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, págs. 142–149, Morgan Kaufmann, San Mateo, 1999.

[35] Dawid, A., "Conditional independence in statistical theory", *J.R. Statist. Soc. Ser.*, tomo B, nº 41, págs. 1–31, 1979.

[36] de Campos, L., "Independency relationships and learning algorithms for singly connected networks", *Journal of Experimental and Theoretical Artificial Intelligence*, tomo 10, nº 4, págs. 511–549, 1998.

[37] de Campos, L. y Huete, J., "Aproximación de Redes Causales mediante Poliárboles", en *Tercer Congreso en Tecnologías y Lógica Fuzzy. Santiago de Compostela*, págs. 25–33, 1993.

[38] de Campos, L. y Huete, J., "Independence concepts in upper and lower probabilities", en *Uncertainty in Intelligence Systems*, B. Bouchon-Meunier, L.Valverde y R. Yager, eds., págs. 49–59, North-Holland, Amsterdam, 1993.

[39] de Campos, L. y Huete, J., "Learning non probabilistic belief networks", en *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, págs. 57–64, Lecture Notes in Computer Science 747. Eds M. Clarke and R. Kruse and S. Moral, 1993.

[40] de Campos, L. y Huete, J., "Efficient algorithms for learning simple belief networks", en *VI Conferencia de la Asociación Española para la Inteligencia Artificial*, págs. 93–102, 1995.

[41] de Campos, L. y Huete, J., "On the use of independence relationships for learning simplified belief networks", *International Journal of Intelligent Systems*, tomo 12, nº 7, págs. 495–522, 1997.

[42] de Campos, L. y Huete, J., "Aproximación de una ordenación de variables en redes causales mediante algoritmos genéticos", en *Inteligencia Artificial 4*, págs. 30–39, 1998.

[43] de Campos, L. y Huete, J., "Approximating Causal Orderings for Bayesian Networks using Genetic Algorithms and Simulated Annealing", en *8th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'00)*, págs. 333–340, 2000.

[44] de Campos, L. y Huete, J., "A new approach for learning belief networks using independence criteria", *International Journal of Approximate Reasoning*, tomo 24, págs. 11–37, 2000.

[45] de Campos, L., Huete, J. y Moral, S., "Probability Intervals: A tool for uncertain reasoning", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Sstems*, tomo 2, nº 2, págs. 167–196, 1994.

[46] de Campos, L., Lamata, M. y Moral, S., "The concept of conditional fuzzy measure", *International Journal of Intelligent Systems*, tomo 5, págs. 237–246, 1990.

[47] de Campos, L. y Puerta, J., "Learning Dynamic Belief Networks using Conditional Independence Tests", en *8th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU'00)*, págs. 325–332, 2000.

[48] de Campos, L. y Puerta, J., "Stochastic local and distributed search algorithms for learning belief networks", en *III International Symposium on Adaptive Systems (ISAS): Evolutionary Computation and Probabilistic Graphical Model*, págs. 109–115, 2001.

[49] de Campos, L. y Puerta, J., "Stochastic local search algorithms for learning belief networks: Searching in the space of orderings", en *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'2001)*, págs. 228–239, LNAI 2143 – Springer, 2001.

[50] Dean, T. y Kanazawa, K., "A model for reasoning about persistence and causation", *Computational Intelligence*, tomo 5, págs. 142–150, 1989.

[51] Díez, F., "Local conditioning in Bayesian networks", *Artificial Intelligence*, tomo 87, págs. 1–20, 1996.

[52] Dorigo, M. y Di Caro, G., *New Ideas in Optimization*, cap. The Ant Colony Optimization Meta-Heuristic, McGraw-Hill, 1999.

[53] Dorigo, M. y Gambardella, L., "Ant Colony System: a Cooperative Learning Approach to the Traveling Salesman Problem", *IEEE Trans. on Evolutionary Computation*, tomo 1, págs. 53–66, 1997.

[54] Dorigo, M., Maniezzo, V. y Colorni, A., "The Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Trans. on Systems, Man and Cybernetics, Part B*, tomo 26, págs. 29–41, 1996.

[55] Fagin, R., "Multivalued dependencies and a new form for relational databases", *ACM Transactions on Database Systems*, tomo 2, págs. 262–278, 1977.

[56] Fernández, J. M., *Modelos de Recuperación de Información basados en Redes de Creencia*, Tesis Doctoral, Dpto de Ciencias de la Computación e I.A. – Universidad de Granada, 2001.

[57] Forbes, J., Hwang, T., Kanazawa, K. y Russell, S., "The BATmobile: Towards a Bayesian automated taxi", en *IJCAI*, 1995.

[58] Friedman, N., "Bayesian Network Repository", Disponible en la dirección http://www-nt.cs.berkeley.edu/home/nir/public_html/Repository/index.htm, 1997.

[59] Friedman, N. y Goldszmidt, M., "Learning Bayesian network with local structure", en *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, págs. 252–262, Morgan Kaufmann, San Mateo, 1996.

[60] Friedman, N. y Koller, D., "Being Bayesian about network structure", en *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, págs. 201–210, Morgan Kaufmann, San Mateo, 2000.

[61] Friedman, N., Murphy, K. y Russell, S., "Learning the structure of dynamic probabilistic networks", en *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, págs. 139–147, Morgan Kaurmann, San Mateo, 1998.

[62] Friedman, N. y Russell, S., "Nonuniform Dynamic Discretization in Hybrid Networks", en *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, D. Geiger y P. Shenoy, eds., págs. 175–181, Morgan & Kaufmann, 1997.

[63] Gambardella, L. y Dorigo, M., "HAS-SOP: Hybrid Ant System for the sequential ordering problem", Inf. téc., IDSIA 11-97, Lugano - Switzerland, 1997.

[64] Gámez, J., *Inferencia abductiva en redes causales*, Tesis Doctoral, Departamento de Ciencias de la Computación e I.A. Escuela Técnica Superior de Ingeniería Informática. Universidad de Granada, 1998.

[65] Geiger, D., "An entropy-based learning algorithm of Bayesian conditional trees", en *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence*, págs. 92–97, 1992.

[66] Geiger, D. y Heckerman, D., "A characterization of the Dirichlet distribution with application to learning Bayesian networks", en *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, págs. 196–207, Morgan Kaufmann, San Mateo, 1995.

[67] Geiger, D., Paz, A. y Pearl, J., "Learning Causal Trees from Dependence Information", en *Eighth National Conference on Artificial Intelligence (AAAI 90)*, págs. 770–776, 1990.

[68] Geiger, D., Paz, A. y Pearl, J., "Learning simple causal structures", *International Journal of Intelligent Systems*, tomo 8, págs. 231–247, 1993.

[69] Gámez, J. y Puerta, J., "Searching the best elimination sequence in Bayesian networks by using ant-colony optimization", *Pattern Recognition Letters*, 2001, (to appear).

[70] Hanks, S., Madigan, D. y Gavrin, J., "Probabilistic temporal reasoning with endogenous change", en *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, págs. 64–71, Morgan Kaurmann, San Mateo, 1995.

[71] Hansen, P. y Mladenović, N., "Variable Neighborhood Search: Principles and applications", *European Journal Operation Research*, tomo 130, págs. 449–149, 2001.

[72] Hart, P., Nilsson, N. y Raphael, B., "A formal basis for the Heuristic determination fo mininum cost paths", *IEEE Transactions on Systems, Man, and Cybernetics*, tomo 4, nº 2, págs. 100–107, 1968.

[73] Hart, P., Nilsson, N. y Raphael, B., "A correction to: A formal basis for the Heuristic determination fo mininum cost paths", *SIGART Newletter*, tomo 37, págs. 28–29, 1972.

[74] Heckerman, D., A.Mamdani y Wellman, M., "Special issue on Real-World Applications of Bayesian Networks", *Communications of the ACM*, tomo 38, 1995.

[75] Heckerman, D., Geiger, D. y Chickering, D., "Learning Bayesian networks: The combination of knowledge and statistical data", en *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, págs. 293–301, Morgan Kaufmann, San Mateo, 1994.

[76] Heckerman, D., Geiger, D. y Chickering, D., "Learning Bayesian networks: The combination of knowledge and statistical data", *Machine Learning*, tomo 20, págs. 197–243, 1995.

[77] Heckerman, D. y Horvitz, E., "Inferring Informational Goals from Free-Text Queries: A Bayesian Approach", en *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, págs. 230–237, Morgan Kaufmann, San Francisco, CA, 1998.

[78] Henrion, M., "Propagating uncertainty by logic sampling in Bayes networks", en *Uncertainty in Artificial Intelligence, 2*, J. Lemmer y L. Kanal, eds., págs. 149–164, North Holland, Amsterdam, 1988.

[79] Herskovits, E. y Cooper, G., "Kutató: An entropy-driven system for the construction of probabilistic expert systems from Databases", en *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*, M. Kaufmann, ed., págs. 54–62, San Mateo, 1990.

[80] Horvitz, E., Breese, J., Heckerman, D., Hovel, D. y Rommelse, K., "The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users", en *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, págs. 256–265, Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[81] Huete, J., *Aprendizaje de Redes de Creencia mediante la detección de Independencias: Modelos No probabilísticos*, Tesis Doctoral, Departamento de Ciencias de la Computación e I.A. Universidad de Granada, 1995.

[82] Hwang, K. y Xu, Z., *Scalable Parallel Computing: Technology, Architecture, Programming*, Mc Graw-Hill, 1998.

[83] Jensen, A. L. y Jensen, F. V., "MIDAS – An Influence Diagram for Management of Mildew in Winter Wheat", en *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, págs. 349–356, Morgan & Kaufmann, Portland, Oregon, 1996.

[84] Jensen, F., "Implementation Aspects of Various Propagation Algortithms in Hugin", Inf. Téc. R 94-2014, Institute for Electronic Systems. Department of Mathematics and Computer Science, 1994.

[85] Jensen, F., *An introduction to Bayesian Networks*, UCL Press, 1996.

[86] Jensen, F., Andersen, S., Kjærulff, U. y Andreassen, S., "MUNIN. On the Case for Probabilities in Medical Expert Systems. A Practical Exercise", en *Lecture Notes in Medical Informatics, 33*, F. et al., ed., Springer Verlag, 1987.

[87] Jensen, F., Kærulff, U., Olesen, K. y Pedersen, J., "An expert system for control of waste water treatment – a pilot project", Inf. téc., University of Aalborg, 1989.

[88] Kærulff, U., "dHugin: A computational system for dynamic time-sliced bayesian networks", *International Journal of Forecasting*, tomo 11, págs. 89–111, 1995.

[89] Kanazawa, K., *Reasoning about time and probability*, Tesis Doctoral, Departament of Computer Science at University of Brown, 1992.

[90] Kanazawa, K., Koller, D. y Russell, S., "Stochastic simulation algorithms for dynamic probabilistic networks", en *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaurmann, San Mateo, 1995.

[91] Kocka, T. y Castelo, R., "Improved Learning of Bayesian Networks", en *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, San Mateo, 2001.

[92] Kullback, S., *Information theory and statistics*, Dover Publication, 1968.

[93] Kullback, S. y Leibler, R., "On information and sufficiency", *Annals of Mathematical Statistics*, , nº 22, págs. 76–86, 1951.

[94] Lam, W. y Bacchus, F., "Learning Bayesian belief networks, an approach based on the MDL principle", *Computational Intelligence*, tomo 10, nº 4, págs. 269–293, 1994.

[95] Lam, W. y Segre, A., "A parallel learning algorithm for Bayesian inference networks", *IEEE Transactions on Knowledge Discovery and Data Engineering*, 2001, in press.

[96] Larrañaga, P., *Aprendizaje estructural y descomposición de redes Bayesianas via algoritmos genéticos*, Tesis Doctoral, Dpto de Ciencias de la Computación e I.A., Universidad del País Vasco, 1995.

[97] Larrañaga, P., Kuijpers, C., Murga, R. y Yurramendi, Y., "Learning Bayesian network structures by searching for the best ordering with genetic algorithms", *IEEE Transactions on Systems, Man, and Cybernetics*, tomo 26, nº 4, págs. 487–493, 1996.

[98] Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R. y Kuijpers, C., "Structure learning of Bayesian networks by genetic algorithms: a performance analysis of control parameters", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, tomo 18, nº 9, págs. 912–926, 1996.

[99] Lauritzen, S., Dawid, A., Larsen, B. y Leimer, H., "Independence properties of directed Markov fields", *Network*, , nº 20, págs. 491–505, 1990.

[100] Lauritzen, S. y Spiegelhalter, D., "Local computations with probabilities on graphical structures and their applications to expert systems (with discussion)", *The Journal of the Royal Statistical Society (Ser B)*, tomo 50, págs. 157–224, 1988.

[101] Laursen, P., "Problem-independent parallel simulated annealing using selection and migration", en *Parallel Problem Solving from Nature*, tomo 3, págs. 408–417, 1994.

[102] Lekuoma, A., *Modelización gráfica de sistemas dinámicos markovianos parcialmente observados*, Tesis Doctoral, Dpto de Métodos Estadísticos - Universidad de Zaragoza, 1996.

[103] Lozano, J. A., Sagarna, R. y Larrañaga, P., "Parallel Estimation of Bayesian Networks Algorithms", en *III International Symposium on Adaptive Systems (ISAS): Evolutionary Computation and Probabilistic Graphical Model*, págs. 137–144, 2001.

[104] Mladenović, N. y Hansen, P., "Variable Neighborhood Search", *Computer Operation Research*, tomo 24, págs. 1097–1100, 1997.

[105] Myers, J. W., Laskey, B. y Levitt, T., "Learning Bayesian networks from incomplete data with stochastic search algorithms", en *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, págs. 476–485, Morgan Kaufmann, San Mateo, 1999.

[106] Neapolitan, R., *Probabilistic Reasoning in Expert Systems*, John Wiley and Sons, New York, 1990.

[107] Nicholson, A. y Brady, M., "Sensor validation using dynamic belief networks", en *Proceedings of the 8th Conference on Uncertainty in Artificial Intelligence*, págs. 207–214, Morgan Kaurmann, San Mateo, 1992.

[108] Nilsson, N. J., *Inteligencia Artificial: Una nueva síntesis*, Mc Graw-Hill, 2001.

[109] Olesen, K. G., Kjærulff, U., Jensen, F., Jensen, F. V., Falck, B., Andreassen, S. y Andersen, S. K., "A MUNIN Network for the Median Nerve - A Case Study in Loops", *Applied Artificial Intelligence*, tomo 3, págs. 385–404, 1989.

[110] Pearl, J., *Heuristic: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, 1984.

[111] Pearl, J., *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann, San Mateo, 1988.

[112] Pearl, J., Geiger, D. y Verma, T., "Conditional independence and its representation", *Kybernetika*, , nº 25, págs. 33–34, 1989.

[113] Pearl, J. y Paz, A., *Graphoids: A graph-based logic for reasoning about relevancy relations*, Technical Report. CSD-850038. Cognitive Science Laboratory. Computer Science Departament. University of California, Los Angeles, 1985.

[114] Provan, G., "Model selection for diagnosis and treatment using temporal influence diagrams", en *In preliminary papers of the 4th International Workshop on Artificial Intelligence and Statistics*, págs. 469–480, 1993.

[115] Provan, G., "Tradeoffs in constructing and evaluting temporal influence diagrams", en *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, págs. 40–47, Morgan Kaurmann, San Mateo, 1993.

[116] Provan, G. y Clarke, J., "Dynamic networks construction and updating techniques for the diagnosis of acute abdominal pain", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, tomo 15, págs. 299–307, 1993.

[117] Proyecto Elvira, "Entorno para el desarrollo de modelos gráficos probabilísticos", Disponilble en la direccion URL `http://leo.ugr.es/~elvira`, 1997.

[118] Rebane, G. y Pearl, J., "The recovery of causal poly-trees from statistical data.", en *Conference on Uncertainty in Artificial Intelligence*, págs. 222–228, 1987.

[119] R.Fung y Favero, B., "Applying Bayesian Networks to Information Retrieval", *Communications of the ACM*, tomo 38, nº 3, págs. 42–57, 1995.

[120] Rissanen, J., "Modeling by shortest data description", *Automatica,* , nº 14, págs. 465–471, 1978.

[121] Robinson, R., "Counting unlabeled acyclic digraphs", en *Lecture Notes in Statistics 622*, C. Litle, ed., págs. 28–43, Springer Verlag, New York, 1977.

[122] Salmerón, A., *Precomputación en grafos de dependencias mendiante algoritmos aproximados*, Tesis Doctoral, Dpto. de Ciencias de la Computación e Inteligencia Artificial. Universidad de Granada, 1998.

[123] Sangüesa, R., Cortés, U. y Gisolfi, A., "A parallel algorithm for building possibilistic causal networks", *International Journal of Approximate Reasoning*, tomo 18, págs. 251–270, 1998.

[124] Singh, M. y Valtorta, M., "Construction of Bayesian network structures from data: A brief survey and an efficient algorithm", *International Journal of Approximate Reasoning*, tomo 12, págs. 111–131, 1995.

[125] Skaaning, C., Jensen, F., Kjærulff, U. y Madsen, A., "Acquisition and Transformation of Likelihoods to Conditional Probabilities for Bayesian Networks", AAAI Spring Symposium, Stanford, USA.

[126] Spirtes, P., Glymour, C. y Scheines, R., "An algorithm for fast recovery of sparse causal graphs", *Social Science Computer Review*, tomo 9, págs. 62–72, 1991.

[127] Spirtes, P., Glymour, C. y Scheines, R., *Causation, Prediction, and Search*, Lecture Notes in Statistics 81, Springer Verlag., 1993.

[128] Spirtes, P., Richardson, T. y Meek, C., "Learning Bayesian networks with discrete variables from data", en *Proceedings of the First Internationl Conference on Knowledge Discovery and Data Mining*, págs. 294–299, 1995.

[129] Srinivas, S., Russell, S. y Agogino, A., "Automated construction of sparse Bayesian networks from unstructured probabilistic models and domain information", en *Proceedings of the 6th Conference on Uncertainty in Artificial Intelligence*, págs. 295–308, Elsevier Science Publisher B.V. North-Holland, 1990.

[130] Studený, M., "Attemps at axiomatic description of conditional independence", *Kybernetika*, , nº 25, págs. 72–79, 1989.

[131] Suzuki, J., "A construction of Bayesian networks from databases based on the MDL principle", en *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, págs. 266–273, 1993.

[132] Tian, J., "A branch and bound algorithm for MDL learning Bayesian networks", en *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, págs. 580–588, Morgan Kaufmann, San Mateo, 2000.

[133] UCI, "UCI Machine Learning Repository", Disponible en `http://www.ics.edu/~mlearn/MLRepository.html`.

[134] Verma, T. y Pearl, J., "Causal Networks: Semantics and expressiveness", en *Uncertainty in Artificial Intelligence 4*, R. Shachter, T. Lewitt, L. Kanal y J. Lemmer, eds., págs. 69–76, North-Holland, 1990.

[135] Wermuth, N. y Lauritzen, S., "Graphical and recursive models for contingence tables", *Biometrika*, tomo 72, págs. 537–552, 1983.

[136] Wilson, N., "Generating graphoids from generalized conditional probability", en *Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence*, págs. 583–590, 1994.

[137] Xiang, Y. y Chu, T., "Parallel learning of belief networks in large and difficult domains", *Data mining and Knowledge Discovery*, tomo 3, págs. 315–339, 1999.

[138] Zweig, G. y Russell, S., “Speech recognition with dynamic Bayesian networks”, en *AAAI*, 1998.