Modelación probabilística basada en modelos gráficos no dirigidos en Algoritmos Evolutivos con Estimación de Distribuciones

Tesis de Doctorado

Roberto Santana Hermida
Instituto de Cibernética, Matemática y Física
Tutor: Alberto Ochoa Rodríguez

La Habana, Febrero 2004

Agradecimientos

Agradezco en primer lugar a mis padres, mi hermana y el resto de mi familia más cercana, sin el apoyo de los cuales esta tesis hubiera sido imposible.

A mi tutor Alberto Ochoa, por su decisiva y continuada participación en mi formación científica. A Marta Rosa Soto por su valiosa ayuda en todas las etapas de realización de la tesis. A Juan Carlos Jiménez y Manuel Lazo por la ayuda ofrecida en todas las ocasiones en que fue precisa. A Eunice Ponce de León, por iniciarme con mucha paciencia en los caminos de la investigación científica. A mis amigos Livang Lozada, Zochil González y Joaquín Rivera, con los cuales compartí las preocupaciones y algunos de los problemas científicos tratados en la tesis.

A Heinz Mühlenbein, cuya guía científica durante mi estancia en el GMD y posteriores recomendaciones respecto al trabajo fueron fundamentales en la obtención de algunos resultados de la tesis. A Thilo Mahnig, con cuya experiencia profesional y ayuda se vieron beneficiados el autor de la tesis y la tesis (el algoritmo LFDA y el formato utilizado en la tesis para la presentación de algoritmos fueron concebidos por Thilo). A Robin Höns por sus observaciones, recomendaciones y fructíferas jornadas de intercambio científico.

A Pedro Larrañaga y José Antonio Lozano cuya guía científica durante mi estancia en la Universidad del País Vasco y fraternal apoyo me permitieron continuar mi aprendizaje en EDAs y utilizar los algoritmos EBNA implementados en su grupo.

A Francisco Pereira, Penousal Machado, Ernesto Costa y Amílcar Cardoso por haberme brindado la oportunidad de trabajar junto a ellos durante mi estancia en la Universidad de Coimbra. Esta estancia resultó fundamental para entender las ventajas, (y también los límites) de los EDAs.

A Ricardo Beausoleil y Rolando Biscay que asumieron la tarea de realizar la oponencia de la tesis para la predefensa y contribuyeron con sus observaciones y recomendaciones a mejorar la calidad de la misma.

A Beatriz Moreno y Yaissel Vidal por su valiosa ayuda en la preparación del documento final de la tesis.

A mis padres, por su ejemplo.

Síntesis

Esta tesis trata sobre el proceso de modelación probabilística en una clase de algoritmos de optimización basados en poblaciones llamados Algoritmos Evolutivos con Estimación de Distribuciones (EDAs). El objetivo principal de la tesis es el desarrollo de EDAs con modelos probabilísticos complejos, basados en modelos gráficos no dirigidos, y capaces de optimizar funciones que no pueden ser optimizadas por EDAs de modelos probabilísticos basados en estructuras gráficas simplemente conectadas. El problema es abordado a partir de dos enfoques diferentes.

El primer enfoque considera las mezclas de distribuciones como modelo probabilístico. Se aborda el aprendizaje y muestreo de mezclas con vistas a su inserción en el marco de los EDAs. Se propone un método para el aprendizaje de mezclas y modificaciones a un algoritmo de aprendizaje existente, para el tratamiento del problema de sobreajuste de los datos. Se analizan algunas de las propiedades de las mezclas para la representación de distribuciones. La tesis introduce dos EDAs basados en mezclas.

El segundo enfoque busca extender la clase de factorizaciones basadas en modelos gráficos no dirigidos para su aplicación en EDAs. Se introducen los grafos de cliques y la aproximación Kikuchi, y se investiga su capacidad para la representación de dependencias probabilísticas. A partir de estos modelos se introducen dos nuevos EDAs cuyo comportamiento es evaluado en la optimización de diferentes funciones.

Índice general

Índice de figuras

Índice de cuadros

Lista de Algoritmos

Lista de Abreviaturas

ACO: Ant Colony Optimization
ADF: Additive Decomposed Function
AIC: Aikaike Information Criterion
BDe: Bayesian-Dirichlet Metric
BIC: Bayesian Information Criterion
BEDA: Boltzmann Estimation of Distribution Algorithm
BMDA: Bivariate Marginal Distribution Algorithm
BOA: Bayesian Optimization Algorithm
cGA: compact Genetic Algorithm
CNF: Conjunctive Normal Form
CFDA: Contraint Factorized Distribution Algorithm
COMIT: Combining Optimizers with Mutual Information Trees
CUMDA: Contraint Univariate Marginal Distribution Algorithm
CS: Classifier System
CVM: Cluster Variation Method
DOPs: Dynamical Optimization Problems
EA: Evolutionary Algorithm
EBNA: Evolutionary Bayesian Network Algorithm
ECGA: Extended Compact Genetic Algorithm
EDA: Estimation of Distribution Algorithm
FDA: Factorized Distribution Algorithm
FDA*: Factorized Distribution Algorithm with fixed model
FDA-learning: Factorized Distribution Algorithm with Learning
GA: Genetic Algorithm
GBML: Genetic Based Machine Learning
GM: Graphical Model
GP: Genetic Programming
GS: Gibbs Sampling
HBOA: Hierachical Bayesian Optimization Algorithm
IDEAS: Iterated Density Estimators Evolutionary Algorithms
LFDA: Learning Factorized Distribution Algorithm
MCMC: Markov Chain Monte Carlo
MDL: Minimum Description Length
MIMIC: Mutual Information Maximization for Input Clustering
MK-EDA: Mixture of Kikuchi Approximations Estimation of Distribution Algorithm

MN-EDA: Markov Network Estimation of Distribution Algorithm
MN-FDA: Markov Network Factorized Distribution Algorithm
MT-FDA: Mixture of Trees Factorized Distribution Algorithm
NFL: No Free Lunch theorem
PADA: Polytree Approximation Distribution Algorithm
PBSMS: Population Based Search Methods that use Selection
PMBGA: Probabilistic Model Building Genetic Algorithms
PBIL: Population Based Incremental Learning
PIPE: Probabilistic Incremental Program Evolution
PLS: Probabilistic Logic Sampling
SGA: Simple Genetic Algorithm
TM: Turing Machine
UMDA: Univariate Marginal Distribution Algorithm

Introducción

El diseño, estudio y aplicación de estrategias de búsqueda heurística constituye un tema de investigación importante tanto en Inteligencia Artificial, como en Optimización. Algunas de estas estrategias heurísticas explícitamente capturan y sintetizan información sobre el espacio de búsqueda creando un modelo de las mejores soluciones. El modelo se usa luego para orientar la exploración hacia regiones promisorias del espacio.

Los Algoritmos Evolutivos con Estimación de Distribuciones (Estimation of Distribution Algorithms (EDAs)) [92, 91, 72] pertenecen a esta clase y son una nueva herramienta para la optimización basada en poblaciones. Se caracterizan por el uso de la modelación probabilística de las soluciones. Los EDAs comparten varias características con sus antecesores los Algoritmos Genéticos (Genetic Algorithms (GAs) [59, 51]). Ambos son métodos de optimización basados poblaciones, e inspirados en la teoría de la selección natural. Mientras los GAs utilizan operadores genéticos para intercambiar información entre las soluciones, los EDAs emplean los referidos modelos probabilísticos. La relación entre EDAs y GAs se ha documentado profusamente en varios trabajos, que incluyen un número de tesis recientes [104, 152, 11].

Un elemento esencial en el funcionamiento de los EDAs es en qué medida logran sus modelos probabilísticos representar las interacciones realmente relevantes en un problema de optimización dado. Consideramos como interacciones relevantes aquellas cuyo desconocimiento por el algoritmo evolutivo puede impedir o retardar la optimización de la función. La caracterización del tipo de interacciones relevantes dada una función a optimizar es un problema que no ha sido lo suficientemente estudiado. Como consecuencia, y a menos que se cuente con información previa sobre la estructura del problema, la primera opción para enfrentar un problema de optimización dado utilizando EDAs es emplear un algoritmo con un modelo probabilístico sencillo. La complejidad de un modelo probabilístico se relaciona al orden de las dependencias y al número de éstas que es capaz de representar el modelo. En este sentido, el modelo más sencillo considera independencia total entre las variables.

Los EDAs con modelos sencillos son eficaces en la resolución de un amplio número de problemas y poco costosos computacionalmente [152]. ¿Pero, qué hacer si este tipo de algoritmos no resulta capaz de resolver el problema? Una posibilidad es el uso de EDAs con modelos más complejos.

Una idea subyacente en el desarrollo de los EDAs es el supuesto de que estos algoritmos pueden ser más eficaces en la medida en que son capaces de representar un número mayor de las interacciones relevantes entre las variables. En este caso se asume que una posible causa del fracaso de EDAs con modelos probabilísticos más sencillos, es la incapacidad de los referidos modelos

para capturar interacciones de un orden mayor. Se ha reconocido que la existencia de dependencias no constituye una fuente de dificultad per se, sólo las interacciones malignas, o frustrantes son difíciles de tratar [62]. Sin embargo, para muchos problemas difíciles los resultados obtenidos hasta el momento han demostrado que el reto consiste en encontrar modelos capaces de representar interacciones de mayor orden, sin dejar de ser factibles desde el punto de vista del costo computacional.

Existe un aspecto relativo al uso de la modelación probabilística que trasciende el objetivo de obtener algoritmos de optimización más robustos. Una de las lecciones del uso de los EDAs es que resulta posible establecer una estrecha relación entre el problema de encontrar una estimación precisa de la distribución, y el objetivo más ambicioso de alcanzar una apropiada descomposición del problema. Las diferentes técnicas de factorización de la distribución que utilizan los EDAs sirven como una fuente de conocimiento sobre la estructura oculta del problema de optimización. Esta estructura, que se refleja usualmente en las interacciones entre las variables capturadas por los modelos probabilísticos, puede utilizarse para una descomposición del problema de optimización en diferentes subproblemas. Este hecho, implícitamente aprovechado por los algoritmos de muestreo que usan los EDAs, podría servir igualmente a otros métodos usados para la resolución de problemas.

Hay varias maneras de incrementar la capacidad de representación de un modelo probabilístico. Nos concentramos en las dos siguientes alternativas:

1. La extensión de la clase de factorizaciones utilizadas en la estimación y muestreo de distribuciones.
2. El uso de mezclas de modelos probabilísticos.

Uno de nuestros objetivos en esta tesis es demostrar que estas alternativas son efectivas para el desarrollo de modelos capaces de representar dependencias más complejas, cuya utilización en EDAs tiene un impacto positivo en la optimización. De la investigación en el primer tema surgen como propuestas los grafos de cliques y la aproximación Kikuchi como métodos que extienden el tipo de factorizaciones utilizadas para la aproximación de distribuciones. En ambos casos, junto a la propuesta del modelo se introducen algoritmos para su aprendizaje a partir de los datos, y para el muestreo de los mismos.

La investigación en el segundo tema conduce al estudio de las mezclas de árboles [80], y a la introducción de las mezclas de aproximaciones Kikuchi, en la aproximación de distribuciones. Los modelos probabilísticos mencionados anteriormente se utilizan como componentes de modelación probabilística de cuatro EDAs introducidos en esta tesis. Estos algoritmos se evalúan empíricamente en cuanto a su capacidad para alcanzar el óptimo del problema y el número de evaluaciones necesarias para alcanzar el óptimo.

Motivaciones

Entre las motivaciones de esta investigación se incluye la atención dedicada en el campo de los EDAs a la resolución de problemas con alto número de interacciones. Esta atención se ha

reflejado en el énfasis puesto en la creación de EDAs con modelos probabilísticos capaces de representar un número mayor de dependencias entre las variables. Sin embargo, las propuestas en este sentido se han limitado al uso de factorizaciones basadas en un conjunto reducido de modelos gráficos, los cuales establecen restricciones particulares a las factorizaciones. Incluso en el terreno de este tipo de modelos gráficos, la investigación en EDAs se ha concentrado en el uso de las redes Bayesianas [42, 91, 104, 11]. Sin embargo, desarrollos de EDAs con modelos probabilísticos basados en grafos no dirigidos son escasos [99, 54, 132] y los resultados no son comparables a los obtenidos con redes Bayesianas. Este hecho señala la necesidad de investigar la conveniencia de usar este tipo de modelos, proponer algoritmos para su aprendizaje y procedimientos para el muestreo a partir de ellos. Una vez que recientes trabajos han tratado exhaustivamente el impacto del uso de modelos simples en los EDAs [152], surge como motivación adicional complementar esta investigación con el estudio de modelos más complejos, capaces de representar dependencias probabilísticas de mayor orden.

Objetivos de la investigación

Objetivo general: Desarrollo de Algoritmos Evolutivos con Estimación de Distribuciones con modelos probabilísticos complejos, basados en modelos gráficos no dirigidos, y capaces de optimizar funciones que no pueden ser optimizadas por EDAs de modelos probabilísticos basados en estructuras gráficas simplemente conectadas.

Desde el punto de vista metodológico abordamos el problema concentrando nuestro trabajo en los siguientes objetivos:

Objetivos específicos:

1. Investigar el uso de las factorizaciones basadas en modelos gráficos no dirigidos como método de estimar distribuciones probabilísticas en EDAs. Proponer nuevas clases de factorizaciones, concebir algoritmos para su aprendizaje y muestreo, e introducirlos en el contexto de la optimización con EDAs.

2. Investigar el uso de mezclas de distribuciones como método para estimar distribuciones probabilísticas en EDAs. Proponer algoritmos para su aprendizaje y muestreo, estudiar los aspectos relacionados con su aprendizaje que influyen en el funcionamiento de los EDAs.

Actualidad, novedad y contribuciones

La investigación en EDAs se ubica en el estado del arte de los Algoritmos Evolutivos. Este hecho se refleja en la publicación de números especiales dedicados a los EDAs [70], la inclusión de artículos sobre el tema en recientes compilaciones sobre el estado del arte de los algoritmos evolutivos y de búsqueda heurística [87, 168, 71], la publicación de un libro [72], la organización de talleres dedicados a esta temática, y particularmente en el grado de diseminación que ha alcanzado la investigación en el tema. En nuestro país, en un documento programático sobre el desarrollo de la ciencia en Cuba para los próximos años se incluye a los Algoritmos Genéticos

entre sus objetivos. Todo lo anterior ayuda a ilustrar la actualidad e importancia del tema de investigación.

Entre los aspectos más novedosos de la tesis podemos citar los siguientes:

1. Se introduce una nueva clasificación de los EDAs, basada en el tipo de factorización usada por estos algoritmos.
2. Se introducen los modelos probabilísticos basados en grafos de cliques como una manera de extender la capacidad de representación de los modelos probabilísticos al espacio de las factorizaciones inválidas.
3. Se introducen los modelos probabilísticos basados en aproximaciones Kikuchi como una manera de extender la capacidad de representación de los modelos probabilísticos al espacio de las factorizaciones inválidas no ordenadas.
4. Se introduce la mezcla de aproximaciones Kikuchi como modelo probabilístico.
5. Se introducen algoritmos de aprendizaje de aproximaciones Kikuchi y mezcla de aproximaciones Kikuchi.
6. Se introduce por primera vez en EDAs el algoritmo de muestreo de Gibbs para el muestreo de las distribuciones.
7. Se introduce por primera vez un EDA capaz de cambiar entre clases diferentes de modelos probabilísticos de acuerdo a la complejidad de los datos.

Estructura de la tesis

En el Capítulo 1 se presenta una revisión de la evolución de los EDAs desde su surgimiento hasta la actualidad. La revisión cubre el papel de los EDAs en el contexto de otras heurísticas de optimización, así como otros campos pertinentes que se usarán en la tesis. El capítulo 2 presenta un EDA basado en mezclas de árboles. Se estudian diferentes aspectos relacionados con el aprendizaje de mezclas que influyen en el funcionamiento de los EDAs

Los capítulos 3 y 4 proponen dos extensiones de la clase de factorizaciones utilizadas para la aproximación de distribuciones, explicando su utilización en la estimación y muestreo de distribuciones. En el capítulo 3 se introduce el MN-FDA, un EDA que está basado en grafos de cliques. El MN-EDA, un EDA más general basado en la aproximación Kikuchi, se introduce en el capítulo 4. En este capítulo se introduce también el MK-EDA, un algoritmo basado en mezclas de aproximaciones Kikuchi. Los algoritmos se validan en la optimización de varias funciones teóricas y prácticas.

En el anexo A introducimos las funciones usadas en los experimentos de la tesis, y discutimos la metodología que se ha seguido en el diseño y ejecución de los experimentos.

1 Contexto: Algoritmos Evolutivos con Estimación de Distribuciones

1.1. Introducción

Llamamos Métodos de Búsqueda Basados en Poblaciones y que usan Selección (Population Based Search Methods that use Selection (PBSMS)) a la clase de estrategias de búsqueda no determinista, comúnmente usadas como métodos de optimización, y con las siguientes características:

1. Utilizan un conjunto de puntos (en lugar de un solo punto) para realizar la búsqueda en el espacio de soluciones.
2. En cada iteración, usualmente llamada generación, un subconjunto de puntos es seleccionado.
3. Mediante la aplicación de operadores de variación sobre el subconjunto seleccionado, una nueva población es creada.
4. El algoritmo itera (evoluciona) hasta que se verifica alguna condición de parada.

Los Algoritmos Genéticos [59, 51] pueden ser considerados como una subclase de los PBSMS. La característica distintiva de esta subclase es la aplicación de los operadores de recombinación y mutación al conjunto de los individuos seleccionados. Otra clase de PBSMS comprende aquellos algoritmos que emplean modelación probabilística de la información contenida en el conjunto seleccionado. Estos algoritmos, que no emplean operadores genéticos, son el objeto de estudio central de esta tesis.

1.1.1. Notación

Sea $X = (X_1, \ldots, X_n)$ un vector de variables aleatorias discretas. Usaremos $x = (x_1, \ldots, x_n)$ para denotar una asignación de valores a las variables. S denotará un conjunto de índices en $\{1, \ldots, n\}$, y X_S (respectivamente x_S) un subconjunto de variables de X (respectivamente x) determinadas por los índices en S. En esta tesis, y a menos que se especifique lo contrario, abordaremos la maximización de una función $f(X) \Rightarrow R$. Nuestro análisis estará enfocado al espacio de los vectores binarios (i.e. $x_i \in \{0,1\}$). Una población $P = \{x^1, \ldots x^N\}$ es un conjunto de vectores. x^i es usado para representar un vector arbitrario de la población, y N es el número de vectores en el conjunto P.

1.2. Algoritmos Genéticos

Los GAs se inspiran en la teoría de la Evolución Natural. Su propósito es evolucionar una población de soluciones, de tal manera que sea posible, con el paso de las generaciones, obtener soluciones cada vez mejores. El criterio de evaluación de las soluciones se especifica previamente.

Para resolver un problema dado, las soluciones tienen que ser codificadas de una forma apropiada, y tiene que ser definida una función que mida la calidad de las soluciones. Los GAs fueron creados por Holland [59], y se ha demostrado que son capaces de resolver una amplia clase de problemas [51].

En los GAs el vector x^i que representa una solución es llamado individuo o cromosoma[1], los componentes del vector se denominan loci, y las variables, genes. Los valores asociados a cada variable reciben el nombre de alelos. El valor de la función de evaluación es llamado valor de la función de adaptación, o valor de adaptación[2].

El pseudo-código de un AG general se muestra en el algoritmo 1.1. El GA comienza generando una población de individuos los cuales son evaluados. Se selecciona un conjunto de individuos a partir de su valor de adaptación. El proceso de selección determina cuales individuos serán usados para generar los individuos de la próxima generación. Después de la selección, pares de individuos del conjunto seleccionado recombinan sus genes para formar dos nuevos individuos que son llamados hijos. Este proceso se llama recombinación. El operador de mutación cambia de forma aleatoria, y de acuerdo a un valor probabilidad previamente definido, el alelo de un gen seleccionado.

Algoritmo 1.1: Esquema general de un Algoritmo Genético

```
1  t ⇐ 0. Generar N individuos aleatoriamente.
2  do {
3      Evaluar los puntos en la función objetivo.
4      Seleccionar un conjunto S de k ≤ N individuos de acuerdo a un método
       de selección.
5      Generar N individuos aplicando los operadores de recombinación y mu-
       tación en el conjunto seleccionado.
6      t ⇐ t + 1
7  } until El criterio de parada ha sido satisfecho
```

1.2.1. Selección

Los métodos de selección son divididos en dos clases [144]: (1) Esquemas proporcionales, y (2) Esquemas ordinales. Los esquemas proporcionales seleccionan un individuo basados en su valor de adaptación relativo comparado con los restantes. Los esquemas ordinales seleccionan un

[1] En el caso general un individuo puede tener más de un cromosoma.

[2] También se suele utilizar el término en inglés "fitness" para designar el valor de adaptación.

individuo basados en su posición relativa una (sub)población ordenada. Ejemplos de esquemas proporcionales son la selección proporcional [51] y la selección Boltzmann [85]. Ejemplos de esquemas ordinales son la selección por truncamiento y la selección por torneo.

Expresiones para las probabilidades de selección determinadas por la selección proporcional y la selección Boltzmann son respectivamente dadas por las ecuaciones (1.1) y (1.2).

Definición 1.1 (Selección proporcional). *La probabilidad asignada por la selección proporcional a un vector x se define como:*

$$p^s(x) = \frac{p(x)f(x)}{\sum_{\tilde{x}} p(\tilde{x})f(\tilde{x})} \tag{1.1}$$

Definición 1.2 (Selección Boltzmann). *La probabilidad asignada por la selección de Boltzmann a un vector x se define como:*

$$p^s(x) = \frac{p(x)e^{\frac{f(x)}{T}}}{\sum_{\tilde{x}} p(\tilde{x})e^{\frac{f(\tilde{x})}{T}}}, \tag{1.2}$$

T es un parámetro de la selección llamado temperatura.

1.2.2. Cruzamiento

La recombinación o cruzamiento es la manera en que los GAs garantizan el intercambio de información entre las diferentes soluciones. En el entrecruzamiento, dos o más individuos combinan partes de sus soluciones para crear uno o más descendientes. Este operador se aplica de acuerdo a un valor de probabilidad previamente especificado que es un parámetro del algoritmo. Existen operadores de cruzamientos clásicos y heurísticos. Entre los primeros están el cruzamiento uniforme, y el cruzamiento en uno, dos o múltiples puntos.

1.2.3. Mutación

La mutación consiste en la perturbación aleatoria de los valores de las variables. Las variables son escogidas en el cromosoma de manera aleatoria, y mutadas de acuerdo a un valor de probabilidad previamente especificado. Este valor de probabilidad es un parámetro del algoritmo. Uno de los efectos de este operador es la inyección de diversidad en la población, mejorando la capacidad exploratoria del algoritmo.

1.3. Algoritmos Evolutivos con Estimación de Distribuciones

Algunos PBSMS se caracterizan por la modelación probabilística de la información contenida en el conjunto seleccionado. En cada generación estos métodos construyen un modelo probabilístico de las soluciones seleccionadas. El modelo probabilístico tiene que ser capaz de capturar un número de dependencias estadísticas que pueden representar relaciones relevantes entre las

variables. Las dependencias son entonces usadas para generar nuevas soluciones muestreando el modelo probabilístico. Cabe esperar que las soluciones generadas compartan un grupo de características con las seleccionadas. De esta manera la búsqueda se orienta hacia áreas promisorias del espacio de búsqueda.

La integración en un solo corpus de todos los algoritmos evolutivos conocidos que usan modelación probabilística resulta una tarea difícil. Estos algoritmos han sido progresivamente introducidos por investigadores de diferentes campos de investigación. En algunos casos los algoritmos se diferencian prácticamente en todos los aspectos exceptuando el uso del modelo probabilístico.

Identificar las razones que han motivado la introducción de estos algoritmos es también una cuestión difícil, dadas las numerosas aproximaciones existentes, muchas no relacionadas entre sí. Sin embargo, detrás de estas aproximaciones pueden identificarse dos elementos comunes: El tratamiento del problema de la estructura de enlace [56, 55, 106], y la meta de alcanzar un mejor y más riguroso análisis teórico de los GAs [92, 87].

En esta tesis utilizaremos el término Algoritmos Evolutivos con Estimación de Distribuciones [92] (EDA) para referirnos a cualquier algoritmo que use la estimación de distribuciones probabilísticas en lugar de los operadores genéticos. Aunque no todas las propuestas se adecuan convenientemente al esquema EDA, este marco conceptual ha sido utilizado anteriormente [72] como un modelo para estudiar el tipo de algoritmos que se analizan.

El algoritmo 1.2 muestra los pasos de un EDA. Los EDAs son también conocidos como Algoritmos Evolutivos con Estimadores de Densidad Iterados (Iterated Density Estimators Evolutionary Algorithms (IDEAS) [12]), y Algoritmos Genéticos con Construcción de Modelos (Probabilistic Model Building Genetic Algorithms (PMBGA) [109]).

En términos de la teoría de propagación de bloques constructivos [51], el problema de la estructura de enlace se define por el efecto causado en la evolución por el rompimiento de importantes soluciones parciales conocidas como bloques constructivos (building blocks). El problema de la estructura de enlace está determinado, no solo por las características del problema de optimización, sino también por la representación usada para resolverlo. Se han propuesto diferentes alternativas [33, 162] que manipulan la representación de las soluciones con el objetivo de hacerlas menos vulnerables a los efectos destructivos de los operadores genéticos.

Otra solución ha sido la creación de operadores genéticos capaces de tratar con las interacciones entre las variables. Sin embargo, estos operadores han sido frecuentemente diseñados considerando las características específicas de los problemas, restringiendo su uso a dominios muy particulares. El problema de la estructura de enlace ha recibido particular atención desde el enfoque EDA conocido como PMBGA [108].

El análisis teórico de la dinámica de los GAs, y el uso de herramientas de análisis más poderosas basadas en la Teoría de la Probabilidad, para el estudio del comportamiento y las propiedades de convergencia de los EAs condujo a la creación de los EDAs.

1.3.1. El lugar de los EDAs en la optimización con métodos heurísticos

La modelación probabilística, en la manera en que es usada por los EDAs, puede ser vista como una forma particular de extraer, representar y usar la información obtenida a lo largo de la

Algoritmo 1.2: Algoritmos Evolutivos con Estimación de Distribuciones

```
1  t ⇐ 0. Generar N puntos aleatoriamente.
2  do {
3      Evaluar los puntos en la función objetivo.
4      Seleccionar un conjunto S de k ≤ N puntos de acuerdo a un criterio de
       selección.
5      Calcular un modelo probabilístico de S.
6      Generar N nuevos puntos muestreando la distribución a partir del modelo
       probabilístico.
7      t ⇐ t + 1
8  } until Hasta que algún criterio de terminación haya sido satisfecho.
```

búsqueda. El análisis de los datos seleccionados persigue el aprendizaje de las relaciones que existen entre las variables del problema. Estas relaciones pueden reflejar características de la estructura del problema. La estimación de distribuciones usando modelos gráficos es el método comúnmente utilizado por los EDAs para realizar una búsqueda eficiente en el espacio. En esta sección analizamos brevemente algunas de las similitudes y diferencias entre los EDAs y otros métodos de optimización heurística.

La optimización basada en colonias de hormigas (Ant Colony Optimization (ACO) [38]), es una técnica de búsqueda que simula el comportamiento de las hormigas en la búsqueda de alimentos. Estos animales depositan al moverse una sustancia química llamada ferormona. Esta sustancia tiene una influencia en las trayectorias de las hormigas que las siguen. Una mayor cantidad de ferormona en una trayectoria específica significa una mayor probabilidad de que una hormiga seleccione este camino. En ACO la cantidad de ferormona puede ser vista como un valor heurístico que es asignado a las soluciones parciales, basado en la frecuencia de la presencia de estas subsoluciones en las buenas soluciones. Como la construcción de nuevas soluciones es llevada a cabo usando una variable auxiliar probabilística basada en el valor de la ferormona, existe una tendencia del algoritmo a formar soluciones que contienen bloques constructivos que han mostrado ser buenos en pasos anteriores. El modelo probabilístico usado por el algoritmo no considera interacciones entre las variables como ha sido señalado en [82]. Un análisis detallado de la relación entre los EDAs y ACO aparece reflejado en [161].

La búsqueda Tabú [50] permite a heurísticas de búsqueda local escapar de mínimos locales donde el algoritmo no puede encontrar ninguna solución en la vecindad que mejore el valor actual de la función objetivo. La aproximación general es evitar quedar atrapado en un ciclo, evitando o penalizando movimientos que llevarían al algoritmo, en la próxima iteración, a puntos del previamente visitados. Esto se logra almacenando las últimas soluciones en términos de las acciones usadas para transformar una solución en la próxima. Una vez que una acción ha sido realizada será considerada Tabú para las próximas T iteraciones, donde T es el tiempo de duración del estatus tabú asignado a la acción. Una solución está prohibida si es obtenida aplicando una acción tabú a la solución actual.

Al igual que ocurre en los EDAs la información obtenida de las soluciones anteriores y las acciones realizadas sobre ellas es usada para explorar otras áreas del espacio de búsqueda. Sin

embargo, una diferencia fundamental es que mientras la información almacenada por la técnica Tabú se utiliza para evitar áreas visitadas poco promisorias, en los EDAs la información almacenada es sobre las áreas promisorias.

Los Algoritmos Culturales (Cultural Algorithms [116]) están basados en las teorías de la evolución cultural. La experiencia de los individuos de una población en la resolución de problemas es coleccionada, mezclada, generalizada y especializada dentro de un espacio de creencias. De esta manera el algoritmo intenta acumular conocimiento global sobre el espacio de estados. En estos algoritmos la evolución tiene lugar a nivel de población y a nivel de creencias. La población y la red de creencias interactúan a través de un protocolo de comunicación que determina el conjunto de individuos que son capaces de actualizar el espacio de creencias. De la misma forma el protocolo determina cómo las creencias actualizadas son capaces de impactar la adaptación de cada componente de la población.

Diferentes maneras para la evolución a nivel de población pueden ser utilizadas: GAs, Estrategias Evolutivas (Evolutionary Strategies (ESs)), Programación Evolutiva (Evolutionary Programming (EP)). También diferentes maneras de representar el espacio de creencias pueden ser usadas: redes semánticas, autómatas simbólicos, y redes neurales. Por ejemplo, en [24] el espacio de creencias es representado usando una red de restricciones (Constraint Network (CN) [36]). Aunque la modelación de las soluciones y su uso en la exploración de nuevas áreas del espacio de búsqueda es un elemento común a los EDAs, no existe en nuestro conocimiento ningún reporte del uso de la modelación probabilística para representar el espacio de creencias en los Algoritmos Culturales.

1.4. Implementaciones factibles de los EDAs: modelos factorizados

De forma simplificada, una factorización de una distribución p es el producto de un conjunto de marginales de p. El uso de las factorizaciones cobra una gran importancia en EDAs porque permite obtener una representación condensada de distribuciones de probabilidad que en otro caso serían muy difíciles de almacenar. Este hecho, y la existencia de métodos que permiten muestrear de manera eficiente las distribuciones de probabilidad almacenadas en forma de factorizaciones, ha motivado su uso en los EDAs. Una subclase especial de EDAs agrupa los algoritmos que usan factorizaciones de la distribución probabilística. En esta tesis llamamos a esta subclase Algoritmos Evolutivos con Factorización de distribuciones (Factorized Distribution Algorithms (FDAs)) [3].

Todos los algoritmos EDAs introducidos hasta el momento usan modelos probabilísticos que son factorizaciones. Sin embargo, es importante destacar que en su artículo inicial sobre EDAs [92] Mühlenbein y Paaß no restringían sus propuestas para la estimación de distribuciones en EDAs al uso de factorizaciones. Por otro lado, los autores tampoco proponían un método práctico para la estimación de las distribuciones. En los capítulos 3 y 4, proponemos un nuevo tipo de

[3] En la literatura el término FDA es usado para designar un tipo particular de FDAs, al cual llamamos en esta tesis FDA*.

factorizaciones, así como resultados de su aplicación en EDAs. En las siguientes secciones se repasan varias definiciones y resultados relacionados con la teoría de grafos [6], y la teoría de modelos gráficos [166, 73].

1.4.1. Teoría de grafos

Un grafo no dirigido $G = (V, E)$ se define por un conjunto de vértices V, y un conjunto de aristas E. Una arista entre los nodos i y j se representará por (i, j) o $i \sim j$.

Definición 1.3 (Clique). *Dado un grafo $G = (V, E)$, un clique en G es un subconjunto totalmente conectado de V. Reservamos la letra C para referirnos a un clique. La colección de todos cliques en G se denota como $\mathcal{C}$. C es maximal cuando no está contenido en cualquier otro clique. C es el clique máximo del grafo si es el clique maximal en $\mathcal{C}$ con el mayor número de vértices.*

Definición 1.4 (Grafo cordal). *Un grafo no dirigido es cordal o triangulado, si y sólo si el único ciclo sin cuerdas en el grafo no contiene más de tres vértices.*

Definición 1.5 (Vecindad,N). *La vecindad $N(X_i)$ de un vértice $X_i \in X$ se define como $N(X_i) = \{X_j : X_j \sim X_i \in E\}$. El conjunto de aristas determina únicamente la vecindad G.*

Definición 1.6 (Frontera, bd). *La frontera de un conjunto de vértices, $X_S \subseteq X$ es el conjunto de los vértices $X \setminus X_S$ que son vecinos de por lo menos un vértice en X_S. La frontera de X_S se denota $bd(X_S)$.*

Definición 1.7 (Clausura, cl). *La clausura de un conjunto de vértices, $X_S \subseteq X$ es el conjunto de vértices $cl(X_S) = X_S \cup bd(X_S)$.*

1.4.2. Modelos Gráficos

Los modelos gráficos han devenido herramientas de representación del conocimiento capaces de representar y manejar relaciones de independencia.

Definición 1.8 (Modelos Gráficos). *En el caso más general un modelo gráfico es una estructura integrada por cuatro componentes esenciales [8]: la semántica, la estructura, la implementación y los parámetros. Existe una variedad de semánticas que incluyen modelos dirigidos (redes Bayesianas) y no dirigidos (redes de Markov), grafos de cadenas, etc. La estructura del modelo está relacionada con las relaciones de dependencia e independencia que se muestran en el grafo: la ausencia de algunos vínculos (aristas o arcos) significa la existencia de ciertas relaciones de independencia condicional y/o marginal entre las variables, la presencia de los vínculos puede representar la existencia de relaciones de dependencia directa.*

Fijando la estructura, existe un número de formas "implementar" las dependencias entre las variables aleatorias, tales como tablas de probabilidades condicionales, redes neurales, árboles de decisión, etc. La componente cuantitativa del modelo es una colección de parámetros numéricos, usualmente probabilidades condicionales, la cuales dan una idea de la fortaleza de las dependencias.

En esta tesis tratamos modelos cuya semántica puede ser únicamente de dos tipos: Redes Bayesianas y Redes de Markov. Las implementaciones están basadas en el uso de tablas de probabilidades marginales y condicionales. Adicionalmente las variables aleatorias serán binarias.

Modelos gráficos basados en grafos no dirigidos

Partimos de la notación introducida en la sección 1.1.1. Trabajaremos con distribuciones positivas, denotadas por p. Usamos $p(X_S = x_S)$ y $p(X_i = x_i|X_j = x_j)$ para denotar respectivamente la probabilidad marginal p cuando $X_S = x_S$, y la probabilidad condicional de $X_i = x_i$ dado $X_j = x_j$. Para simplificar la notación, y cuando no se preste a confusión, utilizamos $p(x_S)$ (respectivamente $p(x_i|x_j)$) en lugar de $p(X_S = x_S)$ y $p(X_i = x_i|X_j = x_j)$. Se utiliza la simbología $I(i, j|V \setminus \{i, j\})$ para expresar la independencia de X_i y X_j dado el conjunto de variables restantes, también conocido como el resto [166].

Definición 1.9 (Grafo de independencia condicional). *El grafo de independencia condicional de X es el grafo no dirigido $G = (V, E)$ donde $V = \{1, 2, \cdots, n\}$ y (i, j) no está en el conjunto de aristas E si y sólo si $I(x_i, x_j|x \setminus \{x_i, x_j\})$.*

La información almacenada por el grafo de independencia puede ser utilizada para encontrar factorizaciones de X en distribuciones marginales más simples. En esta tesis, y siempre que no se preste a confusión, nos referiremos a cualquier grafo de independencia de X simplemente como el grafo asociado a X o el grafo de X.

Definición 1.10 (Grafo de cliques). *Un grafo de cliques construido a partir de un conjunto de cliques maximales es un grafo donde cada nodo corresponde a un clique maximal, y existe una arista entre dos nodos si sus correspondientes cliques se solapan.*

Definición 1.11 (Árbol de cliques). *Una propiedad fundamental de los grafos cordales es que su cliques maximales pueden ordenarse de manera tal que formen un árbol, llamado árbol de cliques. Un árbol de cliques es un grafo de cliques acíclico.*

Definición 1.12 (Propiedad de intersección corrida). *En un árbol de cliques cualquier par de cliques que contengan al nodo α son adyacentes en el árbol de cliques, o están conectados por una cadena compuesta enteramente de cliques que contienen a α. A esta propiedad se le llama la propiedad de intersección corrida [73].*

Definición 1.13 (Factorización). *Una factorización de p es una descomposición de p como producto de un subconjunto de sus probabilidades marginales. Relativa a un grafo G una factorización es completa cuando cada declaración de independencia correspondiente a cada par de vértices no adyacentes en el grafo se usa en la factorización de la distribución de probabilidad.*

Definición 1.14 (Factorización válida). *Decimos que una factorización construida a partir de un grafo es válida cuando los cliques maximales del grafo satisfacen la propiedad de intersección corrida. De lo contrario afirmamos que la factorización es inválida.*

Proposición 1.15. *Las siguientes afirmaciones son equivalentes:*

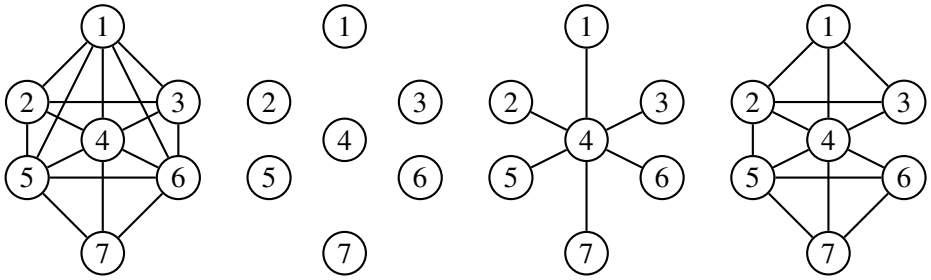

Figura 1.1: a) Grafo de independencia de X; b) Modelo univariado; c) Árbol; d) Subgrafo cordal.

- *G es cordal.*
- *Un árbol de cliques del grafo puede ser construido.*

Prueba: La demostración puede ser encontrada en [166].

En la figura 1.1 a) se muestra un grafo de independencia correspondiente a la probabilidad $p(X_1, \cdots, X_7)$.

Campos Aleatorios de Markov

Definición 1.16 (Campo de Markov Aleatorio). *Una probabilidad p se llama un Campo de Markov Aleatorio (Markov Random Field (MRF)) con respecto a un sistema de vecindad en G si,* $\forall x \in X$, $\forall i \in \{1, \ldots, n\}$, $p(x_i|x \setminus x_i) = p(x_i|bd(x_i))$.

Definición 1.17 (Campo de Gibbs). *Una probabilidad p definida en un grafo G se llama Campo de Gibbs con respecto al sistema de vecindad en G cuando puede ser representada como sigue:*

$$p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} (\Phi_{x_C}), \tag{1.3}$$

donde $\Phi = (\Phi_{C_1}, \ldots, \Phi_{C_r})$ *es un conjunto de funciones no negativas llamadas potenciales de vecindad, uno para cada uno de los r cliques maximales en G. La constante de normalización Z es conocida como la* función de partición *y se define como* $Z = \sum_X \prod_{C \in \mathcal{C}} (\Phi_{X_C})$.

Teorema 1.1. *Teorema Hammersley y Clifford: Sea p un Campo de Gibbs definida en un grafo no dirigido G. Entonces el proceso subyacente X es Markoviano con respecto al grafo. Recíprocamente, la distribución p estrictamente positiva de cualquier MRF definido en G puede ser representada en esta forma factorizada.*

Prueba: En [48].

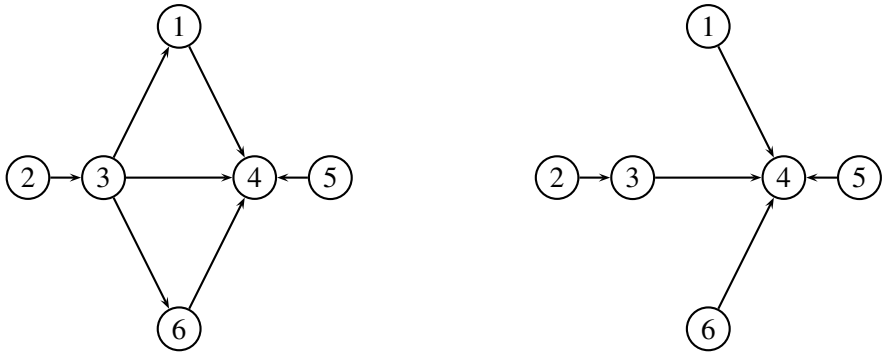

Figura 1.2: Red Bayesiana general y poliárbol.

Redes Bayesianas

Definición 1.18 (Red Bayesiana). *Una red Bayesiana es una representación de la función de probabilidad conjunta $p(X = x)$ (o simplemente $p(x)$). Esta representación consta de dos elementos: la estructura S para X, que es un grafo acíclico dirigido (directed acyclic graph (DAG)) que representa un conjunto de independencias condicionales entre las variables en X, y un conjunto de funciones de probabilidad marginales y/o condicionales.*

La estructura S de X representa el hecho de que X_i y $X_1,\ldots,X_n \backslash Pa_i^S$ son independientes dado Pa_i^S, $i = 2,\ldots,n$. El conjunto de variables Pa_i^S son llamadas los padres de X_i. Por lo tanto la factorización queda expresada como sigue:

$$p(x_1,\ldots,x_n) \quad = \quad \prod_{i=1}^{n} p(x_i|Pa_i^S)$$

El conjunto de funciones de probabilidad marginales y/o condicionales son precisamente aquellas representadas en la ecuación anterior.

Definición 1.19 (Poliárbol). *Un poliárbol es una red Bayesiana que no permite la existencia de ciclos en el grafo no dirigido asociado a su estructura.*

Definición 1.20 (Árbol). *Un árbol es un poliárbol donde cada vértice en el grafo puede tener como máximo un padre en la red. Esto significa que las probabilidades condicionales de una variable dependen del valor de como máximo otra variable.*

Otro caso particular de modelo gráfico es el modelo de independencia total donde ninguno de los nodos tiene padres (todas las variables son independientes). La figura 1.2 muestra las estructuras de modelos gráficos correspondientes a una red Bayesiana general y un poliárbol.

1.4.3. Modelos descomponibles y no descomponibles

Los modelos gráficos pueden ser clasificados en descomponibles y no descomponibles. La clase de modelos descomponibles [102] juega un papel importante en los algoritmos que son tratados en esta tesis. Estos modelos pueden ser representados usando tanto grafos dirigidos como grafos no dirigidos. El criterio topológico que se selecciona para representar las aserciones de independencia depende del tipo de grafo usado: separación para el caso de los grafos no dirigidos y d-separación para los grafos acíclicos dirigidos.

La contraparte gráfica de los modelos descomponibles son los grafos cordales. En el capítulo 3 presentamos una definición formal de éste y otros conceptos relacionados con la teoría de modelos gráficos. Las definiciones que siguen están orientadas a facilitar la comprensión del resto de las secciones de este capítulo.

Un grafo no dirigido es cordal cuando cada ciclo de longitud cuatro o mayor tiene una cuerda. Otra propiedad clave de los grafos cordales es que sus cliques (i.e. los subgrafos más grandes cuyos nodos sean todos adyacentes entre sí.) pueden ser unidos para formar un árbol llamado árbol de cliques (Junction Tree), que cumple que cualquiera dos cliques que contengan un nodo α o son adyacentes en el árbol de cliques, o están conectados por una cadena compuesta completamente de cliques que contienen a α. Esta propiedad es llamada la propiedad de intersección corrida (the running intersection property). El concepto de árbol de cliques, y la propiedad de intersección corrida son tratados con mayor detalle en el capítulo 3.

Los grafos de independencia mostrados en la figura 1.1b-d) son todos cordales, y corresponden a diferentes factorizaciones aproximadas del grafo de independencia mostrado en la figura 1.1a). En la aproximación univariada (figura 1.1b)) se asume que todas las variables son independientes. El valor correspondiente de la distribución de probabilidad conjunta asociada a un vector x es $p(x) = \prod_{i=1}^{7} p(x_i)$. Un modelo que considera un mayor número de dependencias es el árbol (figura 1.1c)), donde cada variable puede depender a lo máximo de otra variable, y el grafo no puede tener ciclos. En este caso, la correspondiente probabilidad conjunta es $p(x) = P(x_4) \cdot \prod_{\substack{i=1 \\ i \neq 4}}^{7} p(x_i|x_4)$. Finalmente, el ejemplo mostrado en la figura 1.1d) presenta el caso de un subgrafo cordal del grafo de independencia. Este subgrafo cordal puede ser representado usando un árbol de cliques. La probabilidad conjunta de esta aproximación es $p(X) = P(x_1,x_2,x_3,x_4) \cdot P(x_5|x_2,x_4) \cdot P(x_6,x_7|x_4,x_5)$.

Decimos que un modelo probabilístico aproximado representado por un grafo no dirigido tiene un mayor poder de expresión cuando puede representar un mayor número de las interacciones del modelo original. En este sentido podemos afirmar que el poder de expresión de los modelos en la figura 1.1b-d) aumenta desde el más simple, mostrado en la figura 1.1b), al más complejo que se muestra en la figura 1.1d).

1.4.4. Medidas estadísticas de dependencia

Definición 1.21 (Información mutua). *Sea $p_{i,j}(x_i,x_j)$ el marginal bivariado correspondiente a las variables binarias x_i y x_j. $I(x_i,x_j)$ denotará la información mutua entre las variables x_i y x_j.*

$$I(x_i,x_j) = \sum_{x_i,x_j} p_{i,j}(x_i,x_j) \cdot \log \frac{p_{i,j}(x_i,x_j)}{p_j(x_j) \cdot p_i(x_i)} \tag{1.4}$$

Definición 1.22 (Divergencia de Kullback-Leibler). *Sean $r(x)$ y $q(x)$ dos distribuciones probabilísticas, la divergencia de Kullback-Leibler $D(r||q)$ se define como:*

$$D(r||q) = \sum_x r(x) \cdot \log \frac{r(x)}{q(x)} \tag{1.5}$$

Usamos la información mutua para descubrir y cuantificar las interacciones entre las variables; y la medida de divergencia de Kullback-Leibler para medir la distancia entre una distribución y sus diferentes aproximaciones. Esta medida de divergencia es siempre no negativa, siendo igual a cero sólo cuando ambas distribuciones son idénticas.

1.4.5. Algoritmos para el aprendizaje de modelos gráficos a partir de los datos

Los modelos gráficos pueden ser aprendidos a partir de los datos. A continuación mencionamos de forma breve un conjunto de alternativas existentes para el aprendizaje de redes Bayesianas.

Existen dos acercamientos fundamentales al aprendizaje de redes Bayesianas a partir de los datos: aprendizaje basado en detección de independencias a partir de pruebas de independencia condicional, y algoritmos basados en la optimización de métricas. El problema de aprender una red Bayesiana general a partir de los datos es NP-completo [21].

Detección de independencias

Estos algoritmos reciben como entradas algunas relaciones de dependencia e independencia entre los subconjuntos de variables del problema, y dan como salida la estructura de la red Bayesiana. Estas relaciones pueden obtenerse por diferentes medios. Relevante para nuestra investigación resulta el caso cuando las relaciones se aprenden de los datos por medio de pruebas de independencia. Este tipo de algoritmos difieren en el costo de las pruebas estadísticas, y en la fiabilidad de los resultados [69]. Dos de los métodos usados para el aprendizaje de la estructura de la red Bayesiana, y que están basados en la detección de independencias condicionales, son el algoritmo PC [155, 156] y los algoritmos para el aprendizaje de poliárboles presentados en [1].

Optimización de métricas

El problema de encontrar una red Bayesiana que aproxime bien los datos puede verse como la optimización de una función sobre el espacio de todas las posibles estructuras gráficas. Esta función puede ser entendida como una medida de la precisión de la estructura para representar las relaciones de independencia existentes en los datos. La función incluye usualmente un término para penalizar la complejidad de la red. Funciones basadas en la verosimilitud son comúnmente

empleadas junto con las funciones de penalización de la complejidad. Algunos ejemplos son el Criterio de Información Bayesiano (Bayesian Information Criterion (BIC) [145]) o el Criterio de Información de Aikaike (Aikaike Information Criterion (AIC) [2]). También funciones Bayesianas y funciones basadas en la teoría de la información puede emplearse [69]. Entre las funciones Bayesianas se encuentran la métrica Bayesiana-Dirichlet (Bayesian-Dirichlet Metric (BDe)). BDe combina el conocimiento anterior sobre el problema y las estadísticas obtenidas a partir de los datos.

El problema de encontrar la red con el valor más alto de función de verosimilitud es NP-completo. Por consiguiente, estrategias heurísticas tienen que ser usadas para obtener soluciones aceptables. Heurísticas glotonas se encuentran entre las mayormente usadas. Una revisión detallada de los métodos utilizados para aprender redes Bayesianas puede encontrarse en [69].

1.4.6. Uso de modelos gráficos

Existen diferentes formas de usar una distribución de probabilidad. Las siguientes son relevantes para nuestro estudio.

Estimadores

Cuando un modelo gráfico es usado como estimador (o evaluador) se asocia un valor de probabilidades a cada punto x. La forma tradicional de usar los modelos gráficos como estimadores es buscando en el modelo las probabilidades marginales y condicionales correspondientes a la solución x. Estos marginales son sustituidos en la factorización representada por el modelo. Éste es el caso de las redes Bayesianas. En el caso de las redes de Markov, una denominada función de partición debe ser conocida para poder usar el modelo como un estimador, esto no es posible en el caso general. Por lo tanto, las redes de Markov no son empleadas frecuentemente como estimadores.

Generadores

Si queremos usar p como un generador esperaremos que el modelo gráfico no de un valor de probabilidad sino uno o más puntos que forman una muestra y donde cada punto x tiene aproximadamente la frecuencia $p(x)$. Un método común para la generación de soluciones es el Muestreo Lógico Probabilístico (Probabilistic Logic Sampling (PLS)). Otros métodos son conocidos como algoritmos de muestreo estocástico [7]. También llamados algoritmos de simulación estocástica o algoritmos de Monte Carlo, comprenden una familia de algoritmos usados para aproximar una distribución p y para estimar esperanzas derivadas de p.

Inferencia

Usado para la inferencia, un modelo gráfico puede proveer información específica sobre la distribución conjunta (ej. probabilidades marginales). La inferencia puede ser hecha en presencia

de información que no fue inicialmente almacenada en el modelo gráfico (evidencia). Cuando el conocimiento es representado en términos de las dependencias probabilísticas contenidas en un modelo gráfico, puede ser usado para inferir la probabilidad de un evento dado que alguna información está disponible.

Modelos gráficos y EDAs

En los EDAs, los modelos gráficos se usan como estimadores para representar el modelo probabilístico de los individuos seleccionados. La estructura del modelo probabilístico se puede fijar de acuerdo a la información existente a priori sobre el problema. La información del problema puede ser incorporada en la construcción del modelo usando priors estructurales o paramétricas. Los modelos probabilísticos se pueden modificar para tratar el caso de las restricciones [132, 138].

Usados como generadores, los modelos gráficos sirven para generar nuevas soluciones. Esto se puede lograr con los algoritmos de muestreo o utilizando el algoritmo de las k configuraciones más probables [152]. El paso de la generación puede ser también modificado para generar solamente soluciones legales [134, 132].

1.5. Física Estadística y EDAs

Un grupo de conceptos y algoritmos concebidos en la física estadística han encontrado aplicación en diferentes dominios de la ciencia de la computación. Entre las aplicaciones posibles de la Física Estadística en modelos gráficos está el uso de los métodos de campo central (mean field methods) para la inferencia aproximada [170, 157, 164], el análisis de las transiciones de fase en problemas de optimización [78] y el estudio de la complejidad algorítmica.

1.6. Algoritmos evolutivos con distribución factorizada

En esta sección analizamos la relación entre GAs y FDAs. Presentamos en detalle aquellos FDAs que serán usados en la tesis.

1.6.1. La transición de GAs a FDAs

Existe consenso en tomar al Cruzamiento Simulado Basado en Bits de (Bit Based Simulated Crossover) [158] como el vínculo entre GAs y FDAs. Este operador de cruzamiento fue uno de los primeros intentos de usar la información probabilística durante la búsqueda. El Cruzamiento Simulado Basado en Bits es idéntico a un método de recombinación donde cada gen de los descendientes es escogido aleatoriamente entre los genes de todos los individuos seleccionados. Esta es una diferencia con los operadores de cruzamiento tradicionales, donde el material

genético de los hijos proviene exclusivamente de los dos padres. Esta forma de efectuar el cruzamiento no respeta posibles interacciones entre las variables porque la selección de los alelos para cualquiera de los genes no está relacionada con la selección de los alelos para ningún otro gen. Mühlenbein y Voigt [93] extendieron este enfoque al introducir la recombinación basada en conjuntos de genes (gene pool recombination).

Un enfoque completamente diferente fue usado en el Aprendizaje Incremental Basado en Poblaciones (Population Based Incremental Learning (PBIL) [3]). PBIL es el primer algoritmo que considera un modelo probabilístico de una población de vectores binarios. Las probabilidades univariadas de las variables son representadas en un vector. Cada componente del vector representa la probabilidad de generar, en la próxima generación, el valor uno para la variable correspondiente. Inicialmente este vector se inicializa con todas las probabilidades iguales a 0,5. Estas probabilidades son adaptadas durante la evolución, moviendo los valores de cada componente hacia los valores de las mejores soluciones encontradas. Desde la creación de PBIL, dos objetivos principales han inspirado la investigación en FDAs: el incremento de la expresividad de los modelos probabilísticos, permitiendo la representación de interacciones de mayor order; y la reducción de la complejidad de los métodos y algoritmos usados para la estimación y muestreo de las probabilidades.

1.6.2. Clasificación de los FDAs

Diferentes clasificaciones de los FDAs pueden ser usadas para describir estos algoritmos. En [69, 107] los FDAs son clasificados de acuerdo a la complejidad de los modelos usados para capturar las dependencias entre las variables. Para nuestro análisis agrupamos los FDAs de acuerdo a la manera en que se realiza el aprendizaje en el modelo gráfico usado. Una clase agrupa a los algoritmos que realizan un aprendizaje paramétrico de las probabilidades, y la otra comprende aquellos algoritmos que hacen un aprendizaje estructural del modelo. El aprendizaje paramétrico y el aprendizaje estructural son también respectivamente conocidos como ajuste de modelo y selección de modelo. A la primera clase pertenecen el PBIL, el Algoritmo Genético Compacto (Compact GA (cGA) [57]), el Algoritmo Evolutivo con Distribución Marginal Univariada (Univariate Marginal Distribution Algorithm (UMDA) [92]) y el Algoritmo Evolutivo con Distribución Factorizada y modelo fijo de las interacciones (Factorized Distribution Algorithm that uses a fixed model of the interactions (FDA*) [91]).

FDAs con aprendizaje paramétrico

De forma similar al PBIL, el UMDA genera nuevas soluciones preservando solamente las proporciones de los valores de todas las variables consideradas independientemente. Este enfoque puede funcionar bien incluso para problemas donde las variables no son completamente independientes. cGA es similar al PBIL, una diferencia importante radica en que existe una correspondencia directa entre la población que es representada y el vector de probabilístico que las representa [109]. Otro algoritmo que realiza un aprendizaje paramétrico es el FDA*, el cual es analizado en la sección 1.7.3.

FDA con aprendizaje estructural

Entre los FDAs que realizan un aprendizaje estructural del modelo se encuentran los siguientes: Algoritmo de Maximización de la Información Mutua para la Clasificación de Entradas (Mutual Information Maximization for Input Clustering (MIMIC) algorithm [29]) que usa una búsqueda glotona para generar el modelo probabilístico en forma de cadena donde cada variable está condicionada a la anterior en la cadena. MIMIC busca en cada generación el modelo en forma de cadena que está más próximo en la distancia de Kullback-Leibler a la distribución de los puntos seleccionados.

En el Algoritmo Evolutivo con Distribución Marginal Bivariada (Bivariate Marginal Distribution Algorithm (BMDA) [111]) las más importantes interacciones bivariadas, de acuerdo al test Chi-cuadrado de Pearson [148] son tomadas en consideración para modelar la distribución probabilística. El modelo probabilístico usado para representar estas dependencias es un bosque. El BMDA es capaz de optimizar una clase más amplia de funciones que el UMDA porque su modelo probabilístico cubre interacciones entre pares de variables. Sin embargo, este algoritmo puede no ser capaz de optimizar funciones con dependencias de mayor orden.

El Algoritmo Genético Compacto Extendido (Extended Compact Genetic Algorithm (ECGA)) [54] usa una factorización de la probabilidad donde las variables están separadas en grupos que no se solapan. Los grupos son encontrados minimizando la medida de Mínima Extensión de la Descripción (Minimum Description Length (MDL) metric) [118] de los datos. En el modelo, se asume que cada grupo es independiente de los restantes. En [4] el modelo probabilístico de MIMIC fue extendido a una clase mayor de grafos de dependencias que pueden ser representados usando redes con forma de árbol. El algoritmo, llamado Combinando Optimizadores Locales con Árboles de Información Mutua (Combining Optimizers with Mutual Information Trees (COMIT)) busca una red Bayesiana donde cada nodo puede tener un solo padre.

1.6.3. Estado del arte en EDAs: FDAs Bayesianos

Diferentes FDAs que utilizan redes Bayesianas como modelos probabilísticos han sido propuestos en la literatura. El Algoritmo Evolutivo con Estimación de Redes Bayesianas (Estimation of Bayesian Network Algorithm (EBNA)) [42] busca la red Bayesiana cuya estructura tiene la verosimilitud a posteriori máxima, y cuyos parámetros puedan ser calculados directamente de los datos. Este algoritmo utiliza la aproximación BIC junto con un algoritmo de búsqueda glotona llamado Algoritmo B. Los resultados aparecidos en [9] muestran que EBNA sobrepasa el desempeño del UMDA en la optimización de funciones aditivas complejas.

El Algoritmo con Distribución Factorizada Basada en Poliárboles (Polytree Approximation of Distribution Algorithm (PADA)) [153] usa el poliárbol como modelo gráfico. La componente de aprendizaje del PADA usa para la construcción del esqueleto del poliárbol tests de detección de independencia. Después se utiliza la información sobre las dependencias marginales y condicionales para darle dirección a algunas aristas, que son transformadas de esta forma en arcos. Al final, el algoritmo puede usar una medida de dependencia para completar el direccionamiento de las aristas restantes. PADA ha sido mejorado [98, 152] para permitir al modelo representar la

clase general de redes Bayesianas simplemente conectadas. También su algoritmo de muestreo ha sido modificado [154] para usar marginales de hasta tercer orden, reduciendo la complejidad del algoritmo de aprendizaje. En [152] PADA es tratado como un esquema que integra cinco tipos de algoritmos de diferente complejidad.

El Algoritmo de Optimización Bayesiana (Bayesian Optimization Algorithm (BOA)) [107] usa la métrica DBe, que permite combinar conocimiento previo del problema con los datos estadísticos. En la propuesta original [107] se usó un simple algoritmo glotón para construir la red. Más tarde [110], grafos de decisión y una medida de complejidad del modelo se incorporaron para mejorar el comportamiento de BOA. El nuevo FDA, llamado BOA jerárquico (Hierarchical BOA (HBOA)) [104, 112] es capaz de resolver problemas jerárquicos difíciles. Otro FDA Bayesiano es el Algoritmo Evolutivo con aprendizaje de distribuciones factorizadas (Learning Factorized Distribution Algorithm (LFDA) [89]).

1.6.4. Otros enfoques para la modelación probabilística en EDAs

Otras aproximaciones a la modelación probabilística en EDAs incluyen el uso de mezclas de distribuciones y la representación de modelos generales factorizados usando grafos no dirigidos. Estos enfoques son tratados en detalle en los capítulos 2 y 3 de esta tesis. Antecedentes en la investigación sobre estos temas se tratan en la sección 1.8.

1.6.5. FDAs para problemas con representación no binaria

Existe un número de casos donde problemas definidos en dominios enteros han sido aproximados usando FDAs [122, 146, 121]. FDAs para dominios contínuos incluyen algoritmos donde se asume que la función de densidad conjunta sigue una distribución n-dimensional normal, factorizada como un producto de densidades univariadas e independientes [123, 150, 147, 13]. Un enfoque diferente en dominios contínuos es realizado utilizando aprendizaje estructural y simulación de redes Bayesianas contínuas [69].

La Evolución de Programas Probabilística e Incremental (Probabilistic Incremental Program Evolution (PIPE)) [124] es un ejemplo de la aplicación de los FDAs en la Programación Genética (Genetic Programming (GP)) [67]. La representación usual en GP son árboles que codifican programas. Cada nodo interno representa una función u operador que es aplicado a los hijos de los nodos. Las hojas representan variables de entrada o constantes. En PIPE la modelación probabilística es realizada usando las probabilidades marginales de las instrucciones y operadores en los nodos de los árboles. PIPE ha sobrepasado a la GP [124] y algunas técnicas de búsqueda basadas en gradiente [125] en la resolución de problemas con baja complejidad algorítmica. Resultados en la aplicación de FDAs en el marco de los Autómatas de Estado Finito han sido presentados en [143, 114, 115].

1.6.6. FDAs para problemas con restricciones

Los problemas con restricciones significan un reto para los EDAs porque resulta difícil la creación de modelos probabilísticos que incorporen las restricciones de los problemas. La primera referencia a la utilización de los FDAs para la resolución de problemas con restricciones apareció en [91], donde se demostró que el FDA* puede manipular restricciones compatibles con la estructura de la función. En [134] se introduce el Algoritmo Evolutivo con Marginales Univariados restringidos (Constraint Univariate Marginal Distribution Algorithm (CUMDA)) y se demuestra que para los problemas tratados el algoritmo es superior a GAs que utilizan funciones de penalidad y a GAs que usan operadores genéticos especiales. En [99] se introduce un FDA que puede resolver problemas con restricciones, donde éstas últimas no tienen que ser compatibles con la estructura del problema, como era el caso en [91].

1.7. FDAs en detalle

En las próximas secciones analizamos en detalle algunos miembros de la familia de los FDAs.

1.7.1. Algoritmo Evolutivo con Distribución Marginal Univariada

El UMDA (algoritmo 1.3) usa marginales univariados para aproximar las distribuciones conjuntas.

Algoritmo 1.3: UMDA

---

1 $t \Leftarrow 0$. Generar N puntos aleatoriamente.
2 **do** {
3 Evaluar los puntos en la función objetivo.
4 Seleccionar un conjunto S de $k \leq N$ puntos de acuerdo a un método de selección.
5 Calcular los marginales univariados $p_i^s(x_i,t)$ a partir de S
6 Generar N nuevos puntos de acuerdo a la distribución $p(x,t+1) = \prod_{i=1}^{n} p_i^s(x_i,t)$.
7 $t \Leftarrow t+1$
8 } **until** Algún criterio de terminación ha sido satisfecho

---

Existen varias razones que hacen del UMDA un algoritmo muy importante. Resultados teóricos derivados para el UMDA exponen la relación entre GAs y FDAs [88]. Otro resultado importante es que el UMDA transforma el paisaje original de la función $f(x)$ en un nuevo paisaje definido para la media de la función calculada a partir de la probabilidad del modelo $W(\tilde{p}) = p(x)f(x)$. Esta transformación suaviza la superficie de la función $f(x)$ de tal forma que si existe una tendencia al óptimo global el UMDA lo encontrará. El UMDA converge a los atractores locales de la media de la función [88].

1.7.2. Algoritmo Evolutivo con Distribución Factorizada basado en Árboles

El Algoritmo Evolutivo con Distribución Factorizada basado en Árboles (Tree-FDA) [132] utiliza como modelo probabilístico al árbol. Aunque el Tree-FDA puede usar otros métodos para construir el árbol [111, 152], el algoritmo propuesto por Chow y Liu [23] es utilizado en su implementación actual. Este algoritmo, al cual llamamos en la tesis algoritmo Chow-Liu, calcula el árbol de cubrimiento de peso máximo (Maximum Weight Spanning Tree (MWST)) a partir de la matriz de información mutua entre las variables.

Algoritmo 1.4: Tree-FDA

1. $t \Leftarrow 0$. Generar N puntos aleatoriamente.
2. **do** {
3. Evaluar los puntos en la función objetivo.
4. Seleccionar un conjunto S de $k \leq N$ puntos de acuerdo a un método de selección.
5. Calcular los marginales univariados y bivariados $p_i^s(x_i,t)$ y $p_{i,j}^s(x_{i,j},t)$ a partir de S
6. Crear un grafo de dependencias G usando algún método previamente especificado y las probabilidades univariadas y bivariadas.
7. Generar N nuevos puntos de acuerdo a la distribución representada por G.
8. $t \Leftarrow t+1$
9. } **until** Algún criterio de terminación ha sido satisfecho

1.7.3. Algoritmo Evolutivo con Distribución Factorizada y modelo fijo de las dependencias (FDA*)

El FDA* construye un árbol de cliques para representar la factorización de la probabilidad conjunta. El árbol de cliques permite un aprendizaje paramétrico eficiente, y muestrear la próxima población. El FDA* no necesita una factorización exacta de la distribución, distribuciones aproximadas pueden ser usadas. El FDA* presentado in [91] emplea factorizaciones basadas en el conocimiento previo sobre la estructura de la función objetivo. Los resultados muestran que el FDA* se desempeña mejor que otros algoritmos evolutivos en la optimización de funciones aditivas. El FDA* ha sido igualmente exitoso en la optimización de problemas decepcionantes con solapamiento entre las variables. La clase de modelos gráficos usados por el FDA* es la clase de los modelos descomponibles. El algoritmo 1.5 muestra los pasos del FDA*. Los conjuntos s_i y b_i son los separadores y residuales tal y como son conocidos en teoría de modelos gráficos.

1.7.4. Una nota sobre la implementación de los EDAs

La disponibilidad de implementaciones de EDAs es todavía escasa. Los algoritmos que han sido introducidos en esta tesis fueron todos implementados en lenguaje C++ siguiendo una metodolo-

Algoritmo 1.5: FDA*

1. $t \Leftarrow 0$. Generar N puntos aleatoriamente.
2. **do** {
3. Evaluar los puntos en la función objetivo.
4. Seleccionar un conjunto S de $k \leq N$ puntos de acuerdo a un método de selección.
5. Calcular las probabilidades condicionales $p^s(\Pi_{b_i}x|\Pi_{c_i}x,t)$.
6. Generar N nuevos puntos de acuerdo a la distribución $p(x,t+1) = \prod_{i=1}^{l} p^s(\Pi_{b_i}x|\Pi_{c_i}x,t)$.
7. $t \Leftarrow t+1$
8. } **until** Algún criterio de terminación ha sido satisfecho

gía de programación orientada a objeto y teniendo en cuenta la necesaria eficiencia en términos de tiempo para experimentos computacionalmente muy costosos.

En esta tesis también hemos incluido resultados obtenidos con otras tres implementaciones EDAs El programa LFDA [75] incluye implementaciones de algoritmos tales como el LFDA, UMDA, y Recocido Simulado (Simulated Annealing [64]). Contiene además una extensa biblioteca de funciones de prueba. La implementación de BOA, disponible en [103], y que permite al usuario definir sus propias funciones. El programa EBNA [42] contiene la implementación de un conjunto de algoritmos Bayesianos. PBIL y MIMIC aparecen también implementados. Todos los programas anteriores han sido programados en lenguaje C++. En la sección A.4.1 analizamos en detalle el marco computacional usado en nuestros experimentos.

1.8. Antecedentes de la investigación y planteamiento del problema

Resulta posible identificar antecedentes en las dos líneas de investigación fundamentales abordadas en la tesis para cumplir nuestro objetivo de *desarrollar Algoritmos Evolutivos con Estimación de Distribuciones con modelos probabilísticos complejos, basados en modelos gráficos no dirigidos, y capaces de optimizar funciones que no pueden ser optimizadas por EDAs de modelos probabilísticos basados en estructuras gráficas simplemente conectadas*.

Estas dos líneas de investigación son:

1. Extensión de la clase de factorizaciones basadas en modelos gráficos no dirigidos.
2. Uso de modelación basada en mezclas de distribuciones.

1.8.1. Extensión de la clase de factorizaciones basadas en modelos gráficos no dirigidos

Ya en [91] los autores planteaban que la propiedad de intersección corrida era una restricción fuerte a ser cumplida por las factorizaciones que usaban los EDAs, en este caso el FDA*. Los autores definían las factorizaciones inválidas como aquellas que no cumplían la referida propiedad. Pero el análisis no presentaba un criterio a seguir para la construcción y uso de factorizaciones de este tipo.

De forma paralela se mostraba en [132] que para problemas de optimización definidos en grafos era posible construir una factorización aproximada a partir de la eliminación de aristas del modelo gráfico determinado por la estructura del problema. El árbol de cliques construido a partir de un subgrafo del modelo gráfico original era usado en todas las generaciones del FDA. En el mismo trabajo fueron presentados experimentos donde el éxito del FDA estaba asociado al número de aristas eliminadas del modelo gráfico. Esta idea fue fundamental para la obtención de los resultados presentados en el capítulo 3 de esta tesis. El estudio de la construcción de modelos probabilísticos definidos para problemas basados en grafos (ej. búsqueda de caminos Hamiltonianos de un grafo), acometido en [141, 138] constituye también un antecedente de nuestra investigación. Este análisis partía de la existencia de una experiencia considerable sobre las limitaciones de los GAs para tratar problemas definidos en grafos [32, 30, 131, 31], y proponía modelos y algoritmos que eran capaces de capturar información relevante del problema, en ocasiones a partir de la propia estructura original.

Otros antecedentes están relacionados con el uso de algoritmos de muestreo diferentes al PLS. La introducción de nuevas clases de factorizaciones precisa la definición de algoritmos de muestreo para éstas. Los antecedentes fundamentales fueron los trabajos relacionados con el muestreo de modelos gráficos en problemas con restricciones [135, 99, 138] y para reducir la complejidad del muestreo en el algoritmo PADA [154].

Existen antecedentes en el estudio de las factorizaciones definidas sobre modelos gráficos dirigidos. Éstas están relacionadas con el desarrollo de los FDAs Bayesianos y están recogidas en la sección 1.6.3.

1.8.2. Uso de modelación basada en mezclas de distribuciones

La investigación en FDAs basados en mezclas de modelos ha recibido más atención que el tema anterior. Sin embargo, con anterioridad a nuestro trabajo con mezclas de árboles la única referencia conocida en este campo eran las aplicaciones de la mezcla de modelos de Priebe en el marco de la optimización evolutiva que utiliza estimadores flexibles de la probabilidad [45, 46]. Este enfoque consideraba modelos para variables contínuas de poca dimensionalidad, y su relación con nuestra propuesta es sólo parcial.

De manera paralela y simultánea a nuestra investigación en mezclas de árboles [139, 140] se dieron a conocer otros trabajos en el uso de mezclas de distribuciones en el campo de los EDAs. En [159] una mezcla de funciones de densidad probabilística Gaussiana es usada para la solución de problemas contínuos multiobjetivos. En [113] los autores emplean mezclas de modelos como

la base para el agrupamiento de datos en la optimización de funciones multimodales discretas y contínuas usando EDAs. En el caso de la optimización discreta, se presenta un marco para el aprendizaje de redes Bayesianas que comparten la misma estructura. Recientemente, mezclas de modelos más complejos se han incorporado como modelos probabilísticos. Un ejemplo en el caso contínuo es el trabajo [22] donde se utilizan mezclas de factores. Igualmente los trabajos en el uso de mezclas Gaussianas han sido extendidos y combinados con enfoques clásicos de los GAs [14, 11].

Tanto en el tipo de mezclas usadas, como los algoritmos empleados para su aprendizaje y muestreo, los algoritmos EDAs introducidos en esta tesis se diferencian de los anteriormente mencionados como se analiza en los capítulos 2 y 4.

1.8.3. Planteamiento del problema

En esta tesis perseguimos el desarrollo de Algoritmos Evolutivos con Estimación de Distribuciones con modelos probabilísticos complejos, basados en modelos gráficos no dirigidos, y capaces de optimizar funciones que no pueden ser optimizadas por EDAs de modelos probabilísticos basados en estructuras gráficas simplemente conectadas.

Desde el punto de vista metodológico abordamos el problema concentrando nuestro trabajo en los siguientes objetivos:

1. Investigar el uso de las factorizaciones basadas en modelos gráficos no dirigidos como método de estimar distribuciones probabilísticas en EDAs. Proponer nuevas clases de factorizaciones, concebir algoritmos para su aprendizaje y muestreo, e introducirlos en el contexto de la optimización con EDAs.

2. Investigar el uso de mezclas de distribuciones como método para estimar distribuciones probabilísticas en EDAs. Proponer algoritmos para su aprendizaje y muestreo, estudiar los aspectos relacionados con su aprendizaje que influyen en el funcionamiento de los EDAs.

2 Aproximaciones basadas en mezclas de árboles

2.1. Introducción

Una de las alternativas para tratar con problemas que exhiben complejas interacciones entre sus variables es la combinación de modelos más simples. Es de esperar que la combinación de estos modelos pueda contribuir a una mejor representación de estas interacciones. Existen varias formas de combinar modelos. En este capítulo investigamos el uso de mezclas de distribuciones en los FDAs. La modelación probabilística usando mezclas de distribuciones finitas [44] consiste en la modelación de una distribución probabilística con una suma pesada de otras distribuciones. Dedicamos especial atención en el capítulo al estudio de los elementos relacionados con el aprendizaje del modelo probabilístico que tienen una incidencia importante en el desempeño del EDA.

2.1.1. Metodología y objetivos

Objetivos

En este capítulo perseguimos alcanzar los siguientes objetivos:

1. Estudiar la conveniencia del uso de mezclas de distribuciones para la modelación probabilística en EDAs.
2. Introducir el Algoritmo Evolutivo con estimación de distribuciones basado en Mezclas de Árboles y evaluar su comportamiento en diferentes problemas de optimización.
3. Investigar el problema del aprendizaje de mezclas de árboles con restricciones en la complejidad del modelo.
4. Evaluar el algoritmo de Esperanza y Maximización (EM) en el aprendizaje de árboles en el contexto de los EDAs.
5. Investigar la influencia de los factores relacionados con el aprendizaje de modelos en la eficiencia de la optimización basada en EDAs.

Metodología

Partimos de una descripción de las mezclas de árboles y un algoritmo para su aprendizaje. El objetivo final es introducir un algoritmo evolutivo con estimación de distribuciones que utiliza mezclas de árboles. Por esa razón, tanto en la presentación de los algoritmos como en la sección experimental dedicamos tiempo a profundizar en la manera en que el aprendizaje incide en una mejor exploración del espacio. Desde el punto de vista metodológico el capítulo sienta también las bases para el análisis del uso de mezclas con componentes más complejas, el cual trata en el capítulo 4.

Estructura del capítulo

En la próxima sección se presentan las mezclas de distribuciones, algunas de sus propiedades y los métodos utilizados para su aprendizaje. En la sección 2.3 se presentan las mezclas de árboles y se demuestran dos teoremas sobre su capacidad de representación. En la sección 2.4 se presenta el algoritmo de aprendizaje de mezclas de árboles utilizado en esta tesis. En la sección 2.5 se explica el algoritmo utilizado para el muestreo de las mezclas de árboles y en la sección 2.6 se presenta el Algoritmo Evolutivo con estimación de distribuciones basado en Mezclas de Árboles y se analiza su complejidad computacional. En la sección 2.7 se discuten algunas de las desventajas del algoritmo de aprendizaje y se introducen un grupo de modificaciones que buscan mejorar su comportamiento. Las sección 2.8 presenta los experimentos utilizados para validar los algoritmos. En las secciones 2.9 y 2.10 se presentan respectivamente el trabajo futuro y las conclusiones del capítulo.

2.2. Mezclas de distribuciones

Definición 2.1. *Una mezcla de distribuciones se define como una distribución de la forma:*

$$Q(x) = \sum_{j=1}^{m} \lambda_j \cdot f_j(x) \tag{2.1}$$

con $\lambda_j \geq 0,\ j = 1,..,m,\ \sum_{j=1}^{m} \lambda_j = 1$.

A las funciones f se les llama densidades o componentes de la mezcla, y a las λ se les denomina proporciones o coeficientes de la mezcla. m es el número de componentes de la mezcla.

Una mezcla de distribuciones puede ser vista como un modelo contentivo de una variable no observada z, que toma valores $j \in \{1...,m\}$ con probabilidad λ_j. En algunos problemas la variable z puede ser conocida u observada.

Ejemplos de mezclas de distribuciones incluyen:

- Mezclas de distribuciones Gaussianas.

- Multiredes Bayesianas.
- Mezclas de árboles.

En las mezclas de distribuciones Gaussianas cada componente es una distribución multivariada Gaussiana. Las mezclas de distribuciones Gaussianas han demostrado ser útiles para representar interacciones no lineales [13]. Una Multired Bayesiana [47] puede ser vista como una red donde las aristas pueden aparecer o desaparecer dependiendo de los valores de ciertos nodos en el modelo gráfico, una noción que se ha llamado aserciones de independencia asimétricas [47]. Las Multiredes Bayesianas constituyen una generalización de las redes Bayesianas. La familia de las distribuciones de probabilidad representables por las redes Bayesianas y por las Multiredes Bayesianas es la misma. Sin embargo, una multired puede dar lugar en la práctica a un ahorro substancial de la complejidad necesaria para aprender y muestrear una red Bayesiana equivalente [8]. La mezcla de árboles es un caso particular de Multired Bayesiana, donde cada componente es un árbol.

2.2.1. Uso de mezclas como herramienta en la clasificación

La mayor parte de las aplicaciones de las mezclas de distribuciones están reportadas en clasificación de datos. Existen dos estrategias diferentes para la clasificación:

- Aprendizaje supervisado: Generación de las descripciones de la clase a partir de ejemplos etiquetados.
- Aprendizaje no supervisado: El problema del descubrimiento automático de clases en los datos.

Como demostraremos, ambas estrategias se pueden utilizar en el marco de los FDAs, pero para cada diverso uso el algoritmo de aprendizaje empleado para aprender la mezcla cambia. Cuando las mezclas se utilizan para el aprendizaje supervisado, se emplea un conjunto de puntos de entrenamiento con su respectiva descripción de la clase. En este caso decimos que cada punto está clasificado. Podemos asociar a cada descripción de la clase un valor de una variable de selección z. En este caso la variable de selección se observa y es posible determinar a qué clase pertenece cada punto. Para cada clase se aprende un modelo probabilístico de los puntos en la clase. Después de terminado este paso, se aprenden los coeficientes de la mezcla.

En el segundo caso, el algoritmo para encontrar la mezcla debe aprender, a partir del conjunto de puntos sin etiqueta de la clase, cuáles son las componentes y los coeficientes de la mezcla. En algunas aplicaciones de las mezclas de modelos, las clases representadas por cada modelo pueden revelar la existencia de subpoblaciones previamente desconocidas o indefinidas en los datos originales [79].

2.2.2. Algoritmos para el aprendizaje de mezclas de modelos

Los enfoques usados para el aprendizaje de mezclas de distribuciones incluyen [79]: métodos graficos, métodos de mínima distancia, métodos de máxima verosimilitud, métodos de los momentos y enfoques Bayesianos.

Cuando la variable de selección es conocida, el aprendizaje de la mezcla se limita al aprendizaje de la estructura y de los parámetros en cada componente, y posteriormente a la determinación de los coeficientes. Cuando la variable de selección no es conocida, una de las alternativas que pueden ser utilizadas es el algoritmo EM, [37] que busca una mezcla que maximice la verosimilitud de los datos. El algoritmo iterativo EM es un método numérico general y confiable para obtener estimadores de parámetros en contextos con datos incompletos. Aunque existen muchas variantes de este algoritmo según el tipo de mezclas, un problema común asociado con todas ellas es que pueden quedar atrapadas en máximos locales de la verosimilitud. Diversos métodos y heurísticas se han propuesto para permitir al EM escapar de los óptimos locales [18], sin embargo estas soluciones no son todavía lo suficientemente robustas y generales.

En [117] los métodos de Monte Carlo basados en cadenas de Markov reversibles (Reversible Markov chain Monte Carlo Methods [52]) se utilizan para encontrar mezclas normales univariadas con un número desconocido y variable de componentes. Estos métodos son capaces de saltar entre los subespacios de los parámetros que corresponden a un número diferente de componentes en la mezcla. Thiesson et al. [160] proponen un método para aprender la mezcla de redes Bayesianas (MBNs) en las cuales cada red Bayesiana está definida por distribuciones Gaussianas condicionales.

2.3. Árboles y modelos basados en mezclas de árboles

Las mezclas de árboles fueron introducidas en [80], donde su utilidad como algoritmo de estimación de densidades y como herramienta de clasificación fue demostrada en un conjunto de problemas. A continuación se introduce formalmente el modelo probabilístico.

Definición 2.2. *Una distribución de probabilidad basada en un árbol se define como:*

$$T(x) = \prod_{i=1}^{n} p(x_i|pa(x_i)) \tag{2.2}$$

donde $pa(x_i)$ es el padre de x_i en el grafo, y $p(x_i|pa(x_i)) = p(x_i)$ cuando $pa(x_i) = \emptyset$, i.e. x_i es la raíz del árbol. La distribución T será también llamada árbol cuando no exista confusión posible.

El grafo (V,E) representa la estructura de la distribución. Para todos los árboles definidos en el dominio V el conjunto de aristas E define únicamente la estructura del árbol. En lo que sigue, y siempre que no se preste a confusión, identificaremos E con la estructura.

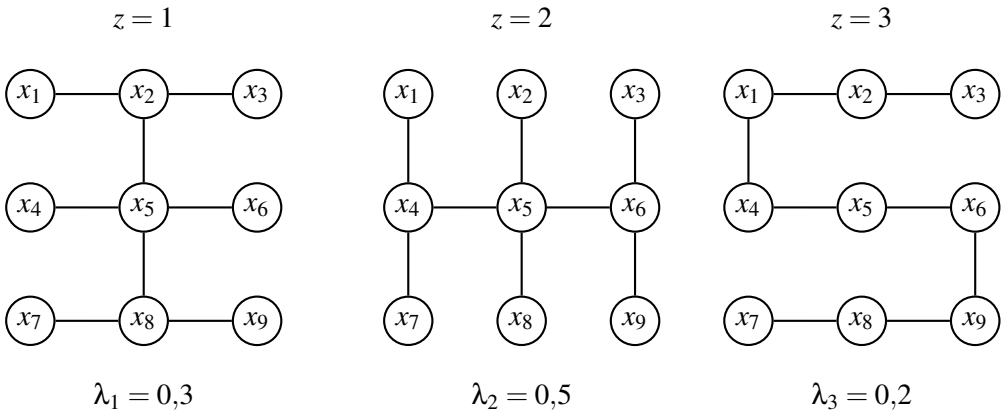

Figura 2.1: Mezcla de árboles con tres componentes.

Definición 2.3. *Una mezcla de árboles se define como una distribución de la forma:*

$$Q(x) = \sum_{j=1}^{m} \lambda_j T^j(x) \tag{2.3}$$

con $\lambda_j \geq 0,\ j = 1,..,m,\ \sum_{j=1}^{m} \lambda_j = 1$.

En este caso las distribuciones basadas en cada árbol son las componentes de la mezcla. Los m árboles pueden tener diversas estructuras y diversos parámetros. Cuando todos los árboles tienen la misma estructura pero los parámetros de los árboles son diferentes, el modelo se llama una mezcla de árboles con estructura compartida [80]. La estructura de una mezcla de árboles con tres componentes se aprecia en la figura 2.1.

2.3.1. Algunos resultados teóricos sobre las mezclas de árboles

Demostramos algunos teoremas que revelan varias propiedades de las mezclas de árboles. Estas propiedades son convenientes para modelos que serán usados en el contexto de los FDAs. Este tema será abordado nuevamente en la sección de los experimentos.

Definición 2.4. *Cuando una probabilidad está factorizada de acuerdo con un modelo univariado, se dice que está en* proporciones de Robbin (Robbin proportions)*. Una población cuyos marginales empíricos factoricen de acuerdo a una distribución en proporciones de Robbin se dice que está en* equilibrio de ligamiento *[84].*

Teorema 2.1. *Dada una población en equilibrio de ligamiento, y un árbol con estructura arbitraria, y marginales bivariados calculados a partir de la población. Entonces la aproximación de la distribución original dada por el árbol es exacta.*

Prueba:

Puesto que las probabilidades condicionales se calculan a partir de una población en equilibrio de ligamiento, $p(x_i|pa(x_i)) = p(x_i), \forall i \in \{1 \dots n\}$. De este hecho, y (2.2) se obtiene,

$$T(x) = \prod_{i=1}^{n} p(x_i) \tag{2.4}$$

□

Teorema 2.2. *Dada una población en equilibrio de ligamiento, y una mezcla de árboles donde cada árbol tiene una estructura arbitraria, y marginales bivariados calculados a partir de la población. Entonces la aproximación de la distribución original dada por la mezcla de árboles es exacta.*

Prueba: A partir del teorema 2.1 se obtiene

$$\begin{aligned} Q(x) &= \sum_{j=1}^{m} \lambda_j \prod_{i=1}^{n} p(x_i|pa(x_i)) \\ &= \prod_{i=1}^{n} p(x_i|pa(x_i))(\sum_{j=1}^{m} \lambda_j) \\ &= \prod_{i=1}^{n} p(x_i|pa(x_i)) \end{aligned} \tag{2.5}$$

□

Los teoremas 2.1 y 2.2 demuestran la capacidad que tiene una mezcla de árboles inicializada con una estructura arbitraria, y probabilidades marginales calculadas a partir de la población, de capturar las relaciones de independencia contenidas en la población.

2.4. Algoritmos para el aprendizaje de mezclas de árboles

Presentamos primeramente un algoritmo para aprender una mezcla de árboles que se ajuste a un conjunto de datos observado. El algoritmo sigue el paradigma de la Máxima Verosimilitud a través del algoritmo EM. En nuestro conocimiento existe solamente otro algoritmo utilizado para aprender mezclas de árboles: en [60] los autores utilizan un algoritmo glotón que a partir de una mezcla inicial de árboles aplica perturbaciones locales hasta que no resulta posible mejorar la verosimilitud del modelo. Este algoritmo simple no es de interés para nuestra investigación pues se puede aplicar solamente a una mezcla de árboles con estructura compartida.

Un algoritmo iterativo para la maximización de la verosimilitud en mezclas de árboles

Llamaremos al algoritmo EM utilizado para aprender mezclas de árboles, e introducido en [80], Algoritmo EM Iterativo (Iterative EM (IEM).) El IEM es el algoritmo de aprendizaje utilizado por el EDA que se introduce en este capítulo, y por lo tanto lo reproducimos en esta tesis.

El problema de aprendizaje es: dado un conjunto de observaciones $D = \{x^1, x^2, ..., x^N\}$, necesitamos encontrar la mezcla de árboles Q que satisfaga

$$Q = \arg\max_{Q} \sum_{i=1}^{N} \log Q(x^i) \tag{2.6}$$

El algoritmo EM introduce una función de verosimilitud llamada verosimilitud logarítmica completa (complete log-likelihood), la cual es la verosimilitud logarítmica tanto de los datos observados como de los no observados dado el modelo estimado en cada paso $M = \{m, T^j, \lambda_j\}, j = 1, \ldots, m$.

$$lc(x^{1...N}, z^{1...N}|M) = \sum_{i=1}^{N} \sum_{j=1}^{m} \delta_{j,z^i} (\log \lambda_j + \log T^j(x^i)) \tag{2.7}$$

donde δ_{j,z^i} es igual a uno si z^i es igual al j-ésimo valor de la variable de selección, y cero en otro caso. La idea en que se basa el algoritmo EM es calcular y optimizar el valor esperado de lc. El paso de expectación (E) consiste en estimar la probabilidad a posteriori de la variable oculta para cada una de las observaciones. En nuestro caso esto significa estimar la probabilidad de que cada árbol genere el punto x^i.

$$\Pr[z^i = j|x^i, M] = \gamma_j(i) = \frac{\lambda_j T^j(x^i)}{\sum_{j'} \lambda_{j'} T^{j'}(x^i)} = E[\delta_j, z^i] \tag{2.8}$$

Estas probabilidades a posteriori son usadas para calcular la esperanza de lc, la cual es una función lineal de los valores $\gamma_j(i)$.

Introducimos las siguientes variables:

$$\Gamma_j = \sum_{i=1}^{N} \gamma_j(x^i), j = 1, ..., m \tag{2.9}$$

$$P^j(x^i) = \frac{\gamma_j(i)}{\Gamma_j} \tag{2.10}$$

Las sumas $\Gamma_j \in [0, N]$ pueden ser interpretadas como el número total de puntos que son generados por la componente T^j. Normalizando las probabilidades a posteriori $\gamma_j(i)$ con Γ_j obtenemos las distribución de probabilidad P^j definida sobre el conjunto de datos.

$$E[lc|x^{1...N}, M] = \sum_{j=1}^{m} \Gamma_j \log \lambda_j + \sum_{j=1}^{m} \Gamma_j \sum_{i=1}^{N} P^j(x^i) \log T^j(x^i) \tag{2.11}$$

La ecuación (2.11) es la expresión de la verosimilitud logarítmica esperada en términos de P^j y Γ. El paso de maximización (M) del algoritmo EM reestima los parámetros del modelo con el

objetivo de maximizar $E[lc|x^{1,\ldots,N}, M]$. Se puede probar [37] que la iteración entre los pasos de Esperanza y Maximización converge a un máximo local de la verosimilitud logarítmica de los datos observados dado el modelo.

Si expresamos la verosimilitud logarítmica completa esperada en términos de P^j y Γ_j, podemos observar que la expresión $E[lc]$ es una suma cuyos términos dependen de los conjuntos disjuntos de los parámetros del modelo. Por lo tanto es posible maximizar separadamente cada término de la suma. De esta forma, se obtienen los nuevos valores para los parámetros λ_j,

$$\lambda_j = \frac{\Gamma_j}{N}, j = 1, \ldots, m \tag{2.12}$$

Para obtener la nueva distribución T^j es necesario maximizar para cada j la información mutua entre T^j y P^j. Este problema es equivalente [23] al problema de ajustar un árbol a una distribución dada, y puede ser solucionado usando el algoritmo Chow-Liu que ha sido mencionado en la sección 1.7.2.

Algoritmo 2.1: Algoritmo EM iterativo (IEM)

```
1  t ← 1; Fijar una mezcla de árboles inicial.
2  Si algún criterio de terminación ha sido satisfecho devolver mezcla inicial y
   salir.
3  do {
4      for j ⇐ 1 to m
5          for i ⇐ 1 to N
6              Calcular γ_j^i, P^j(x^i) usando las ecuaciones (2.8) y (2.10).
7      for j ⇐ 1 to m
8          Calcular los marginales P_v^j, P_uv^j, u, v ∈ V.
9          Calcular la información mutua I_uv^j, u, v ∈ V.
10         Llamar al algoritmo Chow-Liu ({I_uv^j}) para generar E_{T^j}.
11         for (u, v) ∈ E_{T^j}
12             T_uv^j ← P_uv^j
13             T_u^j ← P_u^j
14         λ_j = Γ_j / N
15     t ← t + 1
16 } until Algún criterio de terminación ha sido satisfecho.
```

Teorema 2.3. *Dada una población en equilibrio de ligamiento, una mezcla de árboles inicial donde cada árbol tiene una estructura arbitraria, y sus marginales bivariados se calculan a partir de la población. Si el IEM comienza a partir de este modelo, entonces la distribución aproximada basada en la mezcla de árboles aprendida por el IEM es la distribución univariada exacta.*

Prueba: Del teorema 2.2 tenemos que la aproximación dada por la mezcla de árboles inicial es igual a la distribución exacta. Incluimos entre los criterios de terminación del algoritmo que la verosimilitud sea máxima. Este criterio puede ser utilizado por estar trabajando con distribuciones discretas y tratarse de una población en equilibrio de ligamiento. En el caso de la aproximación basada en la mezcla inicial de árboles, la verosimilitud es máxima. El paso 2 del IEM garantiza que el algoritmo se detendrá en este caso, y dará como salida la aproximación deseada. □

El teorema 2.3 es importante porque garantiza que el IEM puede aprender aproximaciones donde existen independencias entre las variables. Muchos problemas de optimización, incluso algunos con dependencias, pueden resolverse usando los modelos probabilísticos univariados, los cuales asumen independencia total entre las variables.

2.5. Muestreo de la mezcla de árboles

El método clásico para muestrear mezclas de árboles tiene dos pasos. En el primero se decide qué árbol será muestreado. Este paso se realiza escogiendo entre las componentes proporcionalmente a sus coeficientes en la mezcla. Para muestrear el árbol se utiliza el algoritmo PLS. Primeramente la variable en la raíz es instanciada a partir de las probabilidades marginales, después el árbol es recorrido siguiendo una estrategia de primero a lo ancho, instanciando cada variable dado el valor de su padre en el árbol y utilizando las probabilidades condicionales.

2.6. Algoritmo Evolutivo con Factorización de Distribuciones basado en Mezclas de Árboles

Presentamos el algoritmo MT-FDA que emplea mezclas de árboles para modelar la distribución probabilística de la población seleccionada.

El algoritmo 2.2 muestra el pseudo-código del algoritmo MT-FDA. La primera población del algoritmo se genera aleatoriamente. De la población actual, se selecciona un subconjunto de puntos. Una mezcla de árboles Q que aproxima el conjunto seleccionado se encuentra usando un algoritmo de aprendizaje de mezcla de árboles. Los nuevos puntos son generados muestreando a partir de Q.

El aprendizaje del modelo probabilístico es un paso crítico del algoritmo. En próximas secciones analizamos varias modificaciones al algoritmo para aprender mezclas de árboles.

2.6.1. Análisis de la complejidad del MT-FDA

Iniciación

El paso de iniciación consiste en asignar valores a todos los individuos en la población inicial. Tiene complejidad nN.

Algoritmo 2.2: MT-FDA

1 $t \Leftarrow 0$. Generar N puntos aleatoriamente.
2 **do** {
3 Evaluar los puntos en la función objetivo.
4 Seleccionar un conjunto S de $k \leq N$ puntos de acuerdo a un método de selección.
5 Aprender una mezcla de árboles Q a partir del conjunto seleccionado.
6 Generar N nuevos puntos muestreando Q.
7 $t \Leftarrow t+1$
8 } **until** Algún criterio de terminación ha sido satisfecho.

Evaluación

La complejidad computacional de este paso depende del problema. Algunas funciones pueden demandar un tiempo exponencial en el número de variables. Sea $cost_f$ el tiempo asociado a la evaluación de la función f, la complejidad algorítmica de este paso es $O(Ncost_f)$.

Selección

La complejidad del paso de selección depende del método de selección usado. En esta tesis usamos principalmente el método de selección por truncamiento. El algoritmo consiste en seleccionar τN de los mejores individuos de la población. En el caso peor, la complejidad de este paso es $Nlog(N)$.

Algoritmo IEM

La complejidad del IEM para el aprendizaje de mezclas de árboles fue calculada en [80]. La complejidad total es $O(mn^2N + mnr_{MAX}^2)$ donde $r_{MAX} = max_{i\in\{1,\ldots,n\}}|X_i|$. La complejidad de cada paso del algoritmo 2.1 se muestra en el cuadro 2.1.

Pasos	5	7	8	9	10
Complejidad	$O(mnN)$	$O(mn^2N)$	$O(mnr^2{}_{MAX})$	$O(mn^2)$	$O(mnr_{MAX}^2)$

Cuadro 2.1: Complejidad del algoritmo IEM.

Muestreo

La complejidad del muestreo de un árbol tiene orden $O(n)$, la complejidad total del paso de muestreo es $O(Nn)$.

Complejidad total

La complejidad total del MT-FDA es $O(Gm(n^2N + nr_{MAX}^2))$, donde el tamaño de la población N y la cantidad de generaciones G varían según la dificultad del problema. Los resultados presentados en esta tesis sólo son para variables binarias ($r_{MAX}^2 = 2$), por consiguiente la complejidad en nuestro caso es $O(Gmn^2N)$.

2.7. Desventajas y limitaciones del algoritmo de aprendizaje en el contexto EDA

El IEM muestra varias desventajas y limitaciones para su inserción como parte del MT-FDA. La limitación principal es la ausencia de un mecanismo para controlar el problema del sobreajuste.

El problema de sobreajuste de los datos por un modelo probabilístico se refiere a una aproximación demasiado exacta de los datos. Esta aproximación no refleja los rasgos más generales del espacio de la búsqueda. Cuando ocurre en las primeras generaciones de los FDAs, puede llevar a una convergencia prematura de los FDAs. Mientras otros métodos de aprendizaje de modelos probabilísticos, como los métodos de optimización de métricas utilizados para aprender redes Bayesianas (ver sección 1.4.5), asocian una penalidad a la complejidad del modelo, el IEM carece de un mecanismo de este tipo. Adicionalmente, el IEM presenta dos inconvenientes:

1. El número de árboles en la mezcla debe conocerse de antemano y mantenerse fijo durante el aprendizaje.
2. El IEM es muy sensible a la elección de las componentes iniciales de la mezcla. El algoritmo puede converger a óptimos locales de la verosimilitud muy lejanos del óptimo global.

2.7.1. Mejoras al algoritmo de aprendizaje

En esta sección proponemos tres modificaciones al IEM para controlar la complejidad de la mezcla de árboles.

Restricción del número de aristas: mezclas de bosques

En la propuesta original del modelo de mezcla de árboles [80], cada árbol de la mezcla estaba formado por una sola componente conexa. Este hecho puede impedir una correcta representación de aquellas distribuciones que pueden ser representadas de manera más adecuada usando componentes desconectadas (o bosques).

Para resolver esta limitación de las mezclas de árboles hemos considerado mezclas de árboles con componentes desconectadas. Este tipo de modelos pueden ser aprendidos introduciendo un valor de umbral mínimo de dependencia entre dos variables para que la arista correspondiente

sea agregada al árbol. El algoritmo Chow-Liu (sección 1.7.2) se modifica incluyendo esta condición, la cual es evaluada cada vez que una arista va a ser agregada a cualquiera de los árboles.

La magnitud de las dependencias es medida usando la prueba de independencia χ^2, que es una prueba estadística no-paramétrica. Si dos variables X_i y X_j son independientes con un nivel especificado de significación α, entonces la arista no es agregada al árbol. En los EDAs, los valores de umbral han sido incluidos con el mismo objetivo en el aprendizaje de los modelos que utilizan el BMDA [111] y PADA [152].

Como test estadístico usamos la razón de verosimilitud logarítmica indexmedida estadística!razón de verosimilitud logarítmica (log-likehood ratio statistic) [148] (2.13)

$$G^2 = 2 \cdot \sum_{x^i \in D} OBS \cdot log\left(\frac{OBS}{EXP}\right) \tag{2.13}$$

donde el OBS y EXP son respectivamente los valores esperados y observados. En este caso $EXP = N \cdot p(x_i) \cdot p(x_j)$ y $OBS = N.p(x_{\{i,j\}})$. Este estadístico sigue aproximadamente una distribución χ^2 con grados de libertad igual a la dimensión del modelo menos el número de parámetros libres.

En nuestro caso el nivel de significación α es un parámetro del MT-FDA. Hemos usado este test estadístico porque puede ser fácilmente calculado conociendo la información mutua (2.13). $\alpha = 0$ significa que $n-1$ aristas serán incorporadas al árbol independientemente de cual sea el valor de la dependencia. Por otro lado, $\alpha = 1$ implica que las aristas serán incorporadas al árbol solamente si son dependientes con un valor de significación del 100 %. La prueba χ^2 se usó antes en EDAs para aprender para la estructura del BMDA [111] y el FDA-learning [99].

El resultado de introducir un umbral es que cada componente de la mezcla pueda tener a su vez más de una componente desconectada. Por consiguiente, tenemos una mezcla de bosques en lugar de una mezcla de árboles. Sin embargo, abusamos de la notación y llamamos a este modelo una mezcla de árboles, tenga o no componentes desconectadas.

Duración del aprendizaje

A continuación tratamos el problema de la exactitud de la aproximación dada por la mezcla de árboles. Un aspecto asociado a este problema es la determinación del número de iteraciones necesarias para que el IEM pueda obtener una aproximación precisa (pero no super-ajustada) de los datos. En nuestro análisis inicial consideraremos que el número de árboles se mantiene fijo a lo largo de las iteraciones del IEM.

En los experimentos presentados en [139] hemos confirmado que detener el aprendizaje simplemente cuando no se logra ninguna mejora en la verosimilitud puede provocar la convergencia prematura del MT-FDA. La solución propuesta en [139] es fijar el número de pasos de aprendizaje en cada generación. Aunque se lograron mejores resultados usando esta opción, no fue en general una opción satisfactoria.

Durante las primeras generaciones, cuando la diversidad es alta en la población, se requieren más pasos del IEM para lograr una aproximación precisa. A medida que se pierde la diversidad,

menos pasos del IEM son necesarios para obtener un modelo preciso de los datos. El número de pasos también depende del número de árboles.

En esta tesis, la solución que proponemos al problema expuesto anteriormente es detener el IEM cuando el modelo ha logrado un nivel pre-determinado de exactitud en la aproximación. Para explicar la medida usada para evaluar la exactitud del modelo, introducimos alguna nueva notación:

Sea D^c el conjunto formado por exactamente una copia de todos los vectores diferentes en $D = \{x_1, x_2, ..., x_N\}$. Esto significa $x \in D^c \Longrightarrow x \in D$ y $x_i, x_j \in D^c \Longrightarrow x_i \neq x_j$.

Definición 2.5. *Sea $\tilde{P}^j(D^c)$ la suma de las probabilidades asignada por el árbol T^j a todos los puntos en D^c. Llamaremos medida de sobreajuste $\tilde{P}(D^c)$ a la suma de las probabilidades asignada por la mezcla a todos los puntos en D^c.*

$$\tilde{P}(D^c) = \sum_{x_i \in D^c} \sum_{j=1}^{m} \tilde{P}^j(x_i) \tag{2.14}$$

donde

$$\tilde{P}^j(x_i) = \lambda_j T^j(x_i) \tag{2.15}$$

$\tilde{P}$ no es una probabilidad en D^c (ej. $\tilde{P}(D^c) \neq 1$). Sólo cuando todos los puntos del espacio de búsqueda están en D^c, $\tilde{P}(D^c) = 1$. La información necesaria para calcular $\tilde{P}(D^c)$ ya está disponible en el algoritmo, debido a que se necesita para calcular $P^j(x_i)$ (2.8).

La medida de sobreajuste sirve para evaluar la calidad del modelo aprendido en cada paso del IEM. $\tilde{P}$ da una idea de cuál es la probabilidad dada por el modelo a puntos que no están en el conjunto de los datos D, $\tilde{P}(x \in X, x \notin D) = 1 - \tilde{P}(D^c)$. En términos comúnmente usados por los métodos de búsqueda basados en poblaciones, $\tilde{P}$ puede ser entendida como una medida de explotación del espacio de búsqueda.

Definición 2.6. *Llamaremos umbral de sobreajuste al parámetro μ que determina un límite en la medida de sobreajuste que puede alcanzar el IEM.*

De lo anterior se deduce que en principio podemos ejecutar el algoritmo de aprendizaje mientras la medida de sobreajuste no exceda el valor de μ. Cuando el modelo es una mala aproximación de los datos, o los puntos son una muestra muy pequeña del espacio de estados $\tilde{P}(D^c) \approx 0$. Cuando el modelo ajusta completamente los datos $\tilde{P}^c = 1$. Para resumir, presentamos la modificación hecha al algoritmo de aprendizaje: El cálculo de μ es incorporado, y la condición $\tilde{P}(D) \geq \mu$ es agregada como criterio de terminación. El uso de medidas que aportan información sobre el espacio de soluciones para modificar las estrategias de búsqueda fue propuesto inicialmente en [126]. La introducción de la medida de sobreajuste se inspira en esta idea.

Número de árboles

Determinar el número de árboles necesarios para aproximar un conjunto de datos es un problema difícil. En [80], la única estrategia que se propone es realizar varias ejecuciones del IEM, con valores diferentes de *m*, y seleccionar el valor óptimo de *m* comparando el valor promedio de la verosimilitud logarítmica en cada caso. Una estrategia similar podría pensarse para determinar el número de árboles a ser usados por el MT-FDA. En lugar de usar la verosimilitud logarítmica, podría utilizarse el promedio de las evaluaciones de las mejores soluciones para un número de ejecuciones preliminares del algoritmo. No obstante, esta estrategia significaría un costo extra en términos de tiempo y evaluaciones de la función. Más aún, no está claro si un número fijo de árboles podría ser mejor que escoger un valor diferente de *m* en cada generación.

En [140] hemos señalado que el número de árboles en el MT-FDA, y la combinación del número máximo de padres y el parámetro α de la red Bayesiana en el LFDA [89], juegan un papel similar en la optimización de ciertas funciones. En el LFDA α permite controlar la complejidad de la red durante el aprendizaje. Disminuir el valor de α o aumentar el número máximo de padres en el LFDA tiene un efecto similar que aumentar el número de árboles en la mezcla de árboles. Si el óptimo de una función puede encontrarse usando un modelo simple, los resultados de ambos algoritmos de optimización pueden deteriorarse comparados con el caso en el cual se utilice un modelo más simple.

De esta observación sigue, que en comparaciones entre FDA Bayesianos y el MT-FDA resulta una buena opción cambiar *m* de acuerdo al valor de α y al máximo numero de padres usados por la red Bayesiana. De hecho, existen resultados teóricos que permiten estimar a partir de los datos el número máximo de padres necesitado por la red Bayesiana [43]. Desgraciadamente, las cotas obtenidas no son lo suficientemente ajustadas como para ser útiles en una estimación práctica del número máximo de padres necesitados por la red Bayesiana.

Otra posibilidad sería adaptar los métodos de cadenas reversibles de Markov, mencionados en la sección 2.2.2, al caso de la mezcla de árboles. De esta manera el número de árboles también sería una variable desconocida a ser encontrada durante el aprendizaje del modelo. Sin embargo, dos obstáculos mayores se erigen: Primero, calcular las probabilidades de aceptación para las transiciones en el espacio de las mezclas de árboles con distinto valor de *m* es una tarea muy difícil. El segundo y más importante es que, al menos para el caso de las mezclas Gaussianas, esta técnica requiere una gran cantidad de transiciones, y tan alto costo computacional no puede ser asumido por el MT-FDA.

El algoritmo para aprender multiredes Bayesianas [160] es similar al algoritmo IEM, entrelaza la búsqueda de los parámetros y de la estructura del modelo, y trata los datos estimados como datos reales. Una diferencia principal es que el algoritmo para aprender redes Bayesianas penaliza la complejidad del modelo. Sin embargo, no está claro si la manera de penalizar al modelo puede adaptarse a modelos no Gaussianos.

Buena parte del tiempo que hemos dedicado al diseño del MT-FDA ha sido empleado tratando de crear un algoritmo capaz controlar la complejidad del modelo limitando el número de árboles. Los resultados fueron sólo parcialmente exitosos y por razones de espacio no fueron incluidos en esta tesis.

Iniciación de los árboles

La calidad de la aproximación inicial tiene una influencia importante en el resultado final del IEM. El IEM busca un máximo local de la verosimilitud. Si el punto inicial en el espacio de los modelos está cerca de este máximo entonces el algoritmo puede tomar menos tiempo para encontrarlo. Encontrar puntos de inicio buenos es no obstante una cuestión difícil, a menos que exista información disponible sobre la estructura del problema.

Hemos analizado tres maneras diferentes de inicializar las mezclas de árboles para el algoritmo IEM, en todas ellas se generan los coeficientes de la mezcla de manera aleatoria, sujetos a la restricción $\sum \lambda_i = 1$. Sigue una descripción de los tres tipos de inicializaciones:

1. Iniciación aleatoria completa: La estructura y los parámetros de todos los árboles se inicializan aleatoriamente.
2. Iniciación aleatoria de la estructura: La estructura se calcula aleatoriamente y los parámetros son aprendidos de los datos.
3. Iniciación parcialmente aleatoria de la estructura: La estructura y parámetros de los árboles son aprendidos usando el algoritmo Chow-Liu, en un segundo paso estas estructuras se modifican parcialmente de forma aleatoria.

Hemos evaluado estas alternativas en la iniciación del IEM. A partir de los experimentos preliminares realizados, hemos seleccionado la iniciación aleatoria de la estructura como la opción para el IEM.

2.7.2. Uso de probabilidades a priori

Otra alternativa al problema del sobreajuste es el uso de probabilidades a priori. Como parte del trabajo de tesis hemos investigado la conveniencia de usar en el contexto del MT-FDA la solución dada al problema del sobreajuste en [80]. La técnica de suavizamiento con los marginales (smoothing with the marginal) es un caso particular de los métodos para suavizar distribuciones presentados en [95]. Estos métodos está concebidos para permitir la ocurrencia de eventos que de otra forma tendrían probabilidad 0. Las distribuciones de probabilidad obtenidas de los datos son suavizadas a partir de su interpolación con distribuciones más generales.

El suavizamiento con los marginales es equivalente a una selección particular de las distribuciones a priori. De hecho es una distribución Dirichlet derivada de las distribuciones marginales bivariadas para el conjunto de los datos [81]. Los resultados obtenidos con el uso de esta técnica no fueron positivos y por razones de espacio no son incluidos en esta tesis.

En [77] se explica cómo introducir mutación en FDAs, y cómo escoger la proporción de la mutación a partir de un resultado derivado teóricamente. En este artículo se muestra que la mutación aumenta en muchos casos el desempeño del algoritmo, y disminuye la dependencia con respecto a la determinación correcta del tamaño de población. Estos resultados están en

correspondencia con nuestros resultados previos en el uso de mutación en EDAs [119, 120], aunque en estos últimos trabajos un análisis teórico de la mutación no fue presentado.

Mahnig and Mühlenbein afirman que cuando r es la probabilidad a priori para una sola variable binaria, la probabilidad a priori r' para un factor $p(x_1,...,x_k)$ y la probabilidad a priori r^* para un factor $p(x_k|x_1,...,x_{k-1})$ deben ser calculadas como

$$r' = r^* = 2^{-(k-1)}.r \tag{2.16}$$

Usar $r'_i = 2^{-(k_i-1)}r$ con $r = \frac{I_\tau M}{n}$, donde I_τ es un valor que mide la presión de la selección y $M = \tau N$, es una opción razonable de la probabilidad Bayesiana a priori para la selección por truncamiento [77]. En nuestro diseño del MT-FDA hemos incluido el uso de la probabilidad a priori propuesta, adaptándola al caso del MT-FDA, donde los marginales usados pueden tener sólo orden uno o dos. Otra asignación usual es $r = 1$. Esta asignación garantiza que todos los valores posibles de cada variable tendrán asociados probabilidades marginales positivas. Esta probabilidad a priori es equivalente a aprender los marginales usando el estimador de Laplace. $r = 1$ ha sido el valor mayormente utilizado en nuestros experimentos.

Adicionalmente, hemos introducido una probabilidad a priori adaptativa que es diferente para cada árbol. Esta probabilidad a priori aumenta la probabilidad a priori de acuerdo a las características del modelo. El incremento es proporcional a la aproximación de los datos dada por el árbol, y inversamente proporcional al coeficiente del árbol (es decir su peso en la aproximación de la mezcla). En este caso, el valor de la probabilidad a priori r^j para el árbol j se define como sigue:

$$r^j = \frac{\tilde{P}(D^c)M}{\lambda^j n} \tag{2.17}$$

Finalmente, sustituimos r por r^j en (2.16). Esta forma de seleccionar las probabilidades a priori se relaciona con los esquemas de mutación adaptativa usados en los GAs [127].

2.8. Experimentos

Una descripción sobre los criterios utilizados en la evaluación de los EDAs introducidos en la tesis, definición de las funciones y problemas de prueba, y la presentación del marco experimental usado en la evaluación de los EDAs se presenta en el anexo A. Antes de comparar el MT-EDA con otros algoritmos, ejecutamos varios experimentos para investigar la validez de las propuestas presentadas en la sección 2.7.1.

2.8.1. Evaluación de la componente de aprendizaje del MT-FDA

Los experimentos mostrados en esta sección son principalmente de valor teórico. Aunque nuestro interés final es evaluar el desempeño de los algoritmos de aprendizaje en la aproximación

de distribuciones que se originan durante la búsqueda, la evaluación de los mismos en distribuciones aisladas del contexto de la optimización evolutiva contribuye a una mejor validación y comprensión del desempeño de los algoritmos.

Exactitud de la aproximación basada en mezclas de árboles

El objetivo del primer experimento es comparar la exactitud de las aproximaciones de la distribución de Boltzmann basadas en árboles y mezclas de árboles.

La distribución de Boltzmann (1.2) puede calcularse exactamente cuando la cardinalidad del espacio de búsqueda es pequeña. Calculamos la distribución asociada a la función $f_{3deceptive}$ (A.3), $n = 9$ para valores diferentes de la temperatura. Para aprender la aproximación, el IEM usa los marginales calculados a partir de la distribución de probabilidad conjunta.

En la figura 2.2 se muestra el valor exacto de la probabilidad de Boltzmann correspondiente al óptimo de la función. Se muestran también las aproximaciones dadas por el árbol, y dos mezclas, para valores diferentes de la temperatura. Las aproximaciones basadas en mezclas están tan cercanas del valor exacto que sus curvas se solapan en la figura. Puede notarse que las mezclas dan una aproximación más precisa del óptimo cuando la temperatura es baja. Sin embargo, cuando se aumenta la temperatura, y se incrementa la probabilidad del óptimo, la aproximación dada por el árbol mejora, alcanzando la de las mezclas.

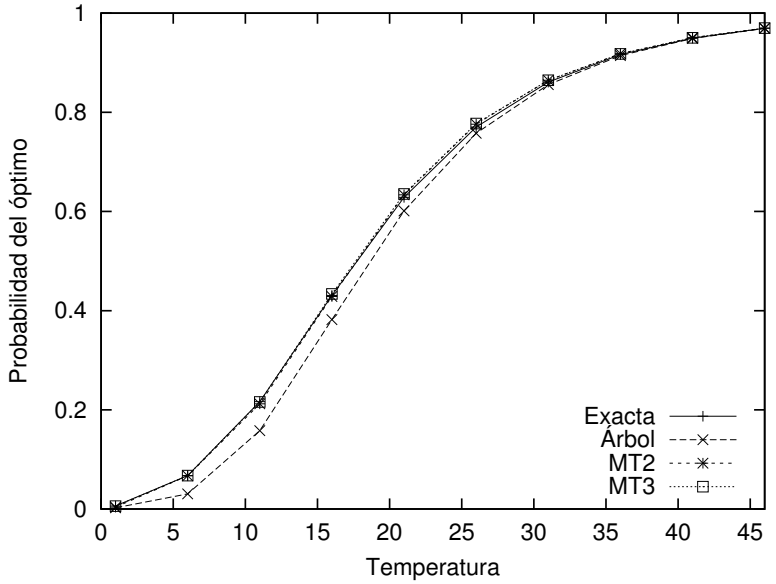

Figura 2.2: Probabilidad de Boltzmann exacta, y aproximaciones dadas por un árbol, y mezclas de árboles con diferente número de componentes.

2.8.2. Investigación de los métodos que modifican el algoritmo de aprendizaje

En los experimentos siguientes evaluamos los métodos que modifican el algoritmo de aprendizaje. Adicionalmente, la conveniencia de usar probabilidades a priori es también evaluada.

Diseño de los experimentos

El desempeño del MT-FDA se evalúa en la solución del problema de la satisfacibilidad. El problema, así como la función de evaluación usada se explican en la sección A.3.1. El conjunto de problemas seleccionados está compuesto por 999 instancias escogidas de manera aleatoria en la región de transición de fase del problema 3-SAT (vea los detalles en la sección A.3.1). Estas instancias están contenidas en el archivo $uf20-91$[1]. Todas las instancias tienen 20 variables y 91 cláusulas, todas son satisfacibles. Con relación a su uso para la validación del MT-FDA, las 999 instancias son equivalentes a la misma cantidad de funciones diferentes.

En un paso previo, las instancias fueron clasificadas en cinco grupos según la dificultad que representan para ser optimizadas usando el UMDA. El criterio usado en la clasificación fue la razón de éxito del UMDA en 100 ejecuciones. Los parámetros usados por el UMDA fueron: $N = 500$, y el número máximo de generaciones igual 25. El cuadro 2.2 muestra la definición de las clases, el número de instancias en cada clase, y la razón de éxito promedio del UMDA en cada clase.

Clase	*I*	*II*	*III*	*IV*	*V*
Definición	$S \leq 20$	$20 < S \geq 50$	$50 < S \geq 75$	$50 < S \geq 90$	$S > 90$
Número inst.	97	114	85	171	532
$\bar{S}$	3,80	24,23	49,09	77,12	98,68

Cuadro 2.2: Clasificación de las instancias del problema SAT de acuerdo a los resultados del UMDA.

El cuadro muestra que la mayoría de las instancias son fáciles de resolver usando el UMDA. Este hecho confirma que el modelo probabilístico simple usado por el UMDA puede realizar de manera satisfactoria la optimización de muchas funciones. Sin embargo, para aproximadamente el diez porciento de los casos (Clase I), los resultados del UMDA son sumamente pobres. Éste es el conjunto de problemas al cual prestamos una mayor atención en nuestro estudio.

El objetivo del experimento siguiente es determinar en cuáles circunstancias las modificaciones propuestas al IEM son efectivas en lograr un mejor desempeño del MT-FDA. Evaluamos el MT-FDA en las cinco clases definidas anteriormente con diferentes conjuntos de parámetros. Los parámetros, y el rango de valores considerado en los experimentos fueron:

- $m = \{2, \cdots, 6\}$
- $\gamma = \{0{,}75; 1{,}0\}$ (ver sección 2.7.1)

[1] El cual está disponible en http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/benchm.html

- $\alpha = \{0; 0{,}75; 1{,}0\}$(ver sección 2.7.1)
- $r = \{0; 1\}$(ver sección 2.7.2)

Para cada posible combinación de los parámetros, 100 ejecuciones del MT-FDA fueron realizadas, y la razón de éxito promedio del algoritmo fue calculada en cada una de las cinco clases. Estos valores se usan como la medida de comparación entre las $(5X2X3X2)$ combinaciones de parámetros en cada clase.

FDA	γ	m	$r = 0$			$r = 1$		
			$\alpha = 0$	$\alpha = 0{,}75$	$\alpha = 1$	$\alpha = 0$	$\alpha = 0{,}75$	$\alpha = 1$
Tree-FDA	–	–	52,23	51,11		52,18	50,64	
MT-FDA	0,75	2	61,68	56,69	56,28	60,36	57,50	56,40
		3	61,96	59,28	58,39	61,51	59,41	57,03
		4	61,24	58,89	56,64	61,10	59,48	57,21
		5	59,56	57,76	56,28	59,16	57,32	55,46
		6	57,87	56,49	55,75	57,27	56,25	53,94
	1	2	61,92	58,49	57,00	62,47	58,87	57,77
		3	63,03	61,35	59,80	63,77	61,37	60,03
		4	63,34	61,94	59,82	63,43	61,75	60,37
		5	60,80	61,67	59,75	61,13	60,17	59,45
		6	59,84	59,44	59,14	59,60	59,49	58,62

Cuadro 2.3: Resultados del Tree-FDA y el MT-FDA para las instancias $uf20-91$ en la clase I.

El cuadro 2.3 es mostrado como un ejemplo de los resultados alcanzados por el MT-FDA ($N = 500$, $\tau = 0{,}15$) en las instancias de la clase I. La razón de éxito correspondiente al Tree-FDA ha sido igualmente incluida. Los mejores resultados se logran con $\{m = 4, \alpha = 0, \gamma = 1, r = 1\}$. Nótese que la ventaja del MT-FDA por sobre el Tree-FDA puede ser de más de un 11 %. Nótese también, que cuando el número de árboles aumenta los resultados del MN-FDA se deterioran, aunque éstos siguen siendo mejores que los obtenidos con el Tree-FDA.

	$N = 100$		$N = 500$	
Parámetro	$r = 0$	$r = 1$	$r = 0$	$r = 1$
r	0	150	78	72
$\alpha = 0$	50	0	44	41
$\alpha = 0{,}75$	0	43	6	9
$\alpha = 1$	0	7	0	0
$\gamma = 0{,}75$	75	5	2	1
$\gamma = 1$	0	70	73	74

Cuadro 2.4: Resultados de los experimentos para la determinación de los parámetros óptimos del MT-FDA.

El cuadro 2.4 presenta una descripción compacta de los resultados de los experimentos. Cada entrada en cuadro muestra el número de veces que la opción del parámetro bajo escrutinio fue la mejor entre todas las posibles asignaciones, considerando que el resto de los parámetros se mantuvo fijo. Los experimentos se realizaron para $N = \{100, 500\}$.

La primera fila en el cuadro 2.4 muestra la influencia del uso de las probabilidades a priori. Para $N = 100$, el uso de probabilidades a priori ($r = 1$) fue la opción mejor en las 150 comparaciones realizadas. En estos casos la diferencia en la razón de éxito lograda por los algoritmos fue sustancial. Cuando el tamaño de la población es aumentado, no se aprecia una diferencia significativa entre usar o no las probabilidades a priori.

Acerca del uso de restricciones en la complejidad del modelo, este efecto demuestra ser beneficioso sólo cuando $N = 100$, y no se introducen probabilidades a priori en la población. Un mayor tamaño de la población, o el uso de probabilidades a priori, hacen innecesario el uso de restricciones en la complejidad del modelo. El uso de probabilidades a priori simultáneamente con un aprendizaje sin restricciones del modelo es la combinación de parámetros con la cual se obtuvieron los mejores resultados del MT-FDA en estos experimentos. Ésta es la opción de los parámetros usada por el MT-FDA en los experimentos que restan por ser mostrados en el capítulo.

Figura 2.3: Razón de éxito del Tree-FDA y el MT-FDA con diferente número de árboles para las instancias SAT $uf20-91$.

Para terminar esta sección dedicada al control de la complejidad del modelo, comparamos los resultados globales del MT-FDA en las cinco clases formadas por las instancias $uf20-91$. Estos resultados se muestran en la figura 2.3.Los resultados para el Tree-FDA han sido igualmente incluidos en la figura. Puede apreciarse la importante diferencia en la razón de éxito alcanzada

por el MT-FDA y el Tree-FDA para la clase más dura de problemas. La mejoría con respecto a los resultados alcanzados por el UMDA también puede ser apreciada si comparamos con los resultados mostrados en el cuadro 2.2. En la clase V, la más fácil para el UMDA, los resultados logrados con el Tree-FDA son ligeramente mejores que los del MT-FDA y el UMDA.

Resultados para las funciones de prueba

El objetivo de los experimentos siguientes es comparar el desempeño de MT-FDA con otros FDAs. Usamos varias funciones normalmente utilizadas para evaluar algoritmos evolutivos y presentadas en el anexo A. En el cuadro 2.5 se muestran los parámetros usados en las comparaciones entre los FDAs.

Función	n	N	τ
OneMax	30	200	0,3
Bigjump(3,1)	30	200	0,3
Symmetric	30	200	0,3
Deceptive4	32	1000	0,15
Deceptive5	30	2000	0,05
Isotorus	36	1000	0,15

Cuadro 2.5: Parámetros de los FDAs utilizados en los experimentos.

El cuadro 2.6 muestra los resultados de la comparación entre el LFDA y el MT-FDA para tres funciones unitarias. Los resultados del LFDA fueron obtenidos a partir de nuestras simulaciones.

	OneMax		Bigjump(3,1)		Symmetric	
	S	$\hat{g}$	S	$\hat{g}$	S	$\hat{g}$
$\text{LFDA}_{0,75}$	99	5,03	100	5,28	100	5,04
$\text{LFDA}_{0,5}$	99	5,68	91	5,74	98	5,30
$\text{LFDA}_{0,25}$	81	6,46	51	6,19	80	5,82
Tree-FDA	100	5,42	100	5,71	100	5,51
MT-FDA_2	100	5,37	97	5,57	100	5,27
MT-FDA_4	100	5,31	88	5,52	93	5,30
MT-FDA_6	88	6,19	69	6,10	76	5,86

Cuadro 2.6: Resultados del MT-FDA y el LFDA para funciones unitarias.

Los resultados muestran la capacidad del MT-FDA para resolver funciones con pocas interacciones como *Onemax*, así como las funciones *Bigjump* y *Symmetric*. En todos los ejemplos, MT-FDA_2 resuelve los problemas con una razón de éxito por encima de 95. Nótese que cuando el número de árboles aumenta, los resultados se deterioran, pero un efecto similar es apreciado para el LFDA cuando el parámetro que controla la complejidad de la red es disminuido.

El cuadro 2.7 muestra los resultados de la comparación entre el MT-FDA y el LFDA para dos funciones decepcionantes y la función *Isotorus*. Estas funciones exhiben fuertes interacciones entre sus variables, ésta es la clase de funciones donde los FDAs Bayesianos superan normalmente a otros métodos de optimización.

	Deceptive4		Deceptive5		Isotorus	
	S	$\hat{g}$	S	$\hat{g}$	S	$\hat{g}$
$LFDA_{0,75}$	62	5,66	61	4,54	74	6,67
$LFDA_{0,5}$	94	4,92	80	4,45	87	5,97
$LFDA_{0,25}$	96	4,84	97	4,00	94	5,36
Tree-FDA	92	5,54	45	6,28	70	6,32
$MT\text{-}FDA_2$	93	5,89	58	6,08	90	5,8
$MT\text{-}FDA_3$	89	6,22	67	5,91	93	5,53
$MT\text{-}FDA_4$	87	6,25	71	5,84	95	5,37
$MT\text{-}FDA_6$	83	6,56	73	6,05	97	5,30
$MT\text{-}FDA_{10}$	59	7,15	63	6,19	96	5,50

Cuadro 2.7: Resultados del MT-FDA y el LFDA para las funciones decepcionantes e Isotorus.

El MT-FDA puede optimizar la función *Deceptive*4 logrando resultados muy cercanos a los del LFDA. No es capaz de alcanzar los mismos resultados que el LFDA para la función *Deceptive*5. En el caso de la función *Isotorus*, los resultados logrados por el MT-FDA son los mejores, estos resultados mejoran a medida que se incrementa el número de árboles. En [139] hemos identificado tempranamente a la función *Isotorus* como un caso donde el MT-FDA es capaz de sobrepasar los resultados alcanzados por los FDAs Bayesianos. Sin embargo las causas que expliquen tal comportamiento del algoritmo para esta función no han sido esclarecidas.

2.8.3. Uso de la variable seleccionada en la búsqueda

Como fue abordado brevemente en la sección 2.2.1, el uso de la variable seleccionada en el contexto del MT-EDA permite maneras diferentes y flexibles de dirigir la búsqueda. Hemos realizado un grupo de experimentos para evaluar la conveniencia de usar una de las variables de la función como la variable seleccionada. Los resultados ayudan a ilustrar cómo los parámetros de la mezcla pueden ser convenientemente utilizados por el MT-FDA. El cuadro 2.8 muestra los resultados para la función Nf^3_{dec} (A.7). Esta función alcanza el óptimo global en varios puntos que están determinados por los valores de la variable x_1. Cuando $x_1 = 1$, hay solamente un óptimo, y el resto de las variables también toman valor uno. Cuando $x_1 = 0$, hay $3^{\frac{n}{3}}$ óptimos (todas las posibles combinaciones de dos unos en cada conjunto de definición). Para cada conjunto de parámetros se ejecutaron 30 experimentos.

En el cuadro se muestra una comparación entre dos MT-FDAs con mezclas de dos árboles. La mezcla del primer MT-EDA usa a x_1 como su variable seleccionada. En el segundo la variable seleccionada es desconocida, como ocurría en los experimentos anteriores. Cuando x_1 se trata como la variable seleccionada, los puntos que cumplen $(x_1 = 0)$ son aproximados por un árbol,

Función	n	N	Var. Seleccionada	τ	S	$\bar{g}$
Nf^3_{dec}	31	2500	desconocida	0,15	24	8,96
	46	3000	desconocida	0,15	21	11,57
	61	3000	desconocida	0,25	12	19,80
	31	1500	x_1	0,15	30	3,10
	46	1500	x_1	0,15	28	6,25
	61	2500	x_1	0,25	23	10,48

Cuadro 2.8: Resultados numéricos para la función Nf^3_{dec}.

y aquellos que satisfacen $(x_1 = 1)$ por el otro. Los coeficientes de la mezcla se calculan como $\lambda_0 = \frac{N_0}{N}, \lambda_1 = \frac{N_1}{N}$, donde N_0 y N_1 son respectivamente el número de individuos que satisfacen $(x_1 = 0)$ y $(x_1 = 1)$. Semejante mezcla permite que el MT-FDA pueda concentrar su búsqueda en el espacio de soluciones definidas por los valores de x_1. En el transcurso de la evolución, uno de los coeficientes debe volverse 0, y un MT-FDA basado en un sólo árbol será ejecutado a partir de ese momento.

En el cuadro 2.8 puede apreciarse la mejoría alcanzada considerando x_1 como la variable seleccionada. Éste es un ejemplo simple de cómo la variable seleccionada también puede incorporarse como una herramienta para influir en la búsqueda.

2.8.4. Análisis de los experimentos

La primera conclusión de nuestros experimentos es haber identificado la complejidad de la relación entre el proceso de aprendizaje de la mezcla de árboles y la manera en que éste se expresa en la búsqueda que realiza el MT-FDA. Hemos validado la conveniencia de emplear las diferentes alternativas propuestas para controlar la complejidad modelo, y definido los marcos apropiados para su aplicación. Hemos establecido que el uso de probabilidades a priori permite prescindir del uso de las restricciones sobre la complejidad del modelo. Sin embargo, como el uso de probabilidades a priori puede implicar un mayor número de evaluaciones para alcanzar la misma razón de éxito, las restricciones en cuanto a la complejidad del modelo quedan como una alternativa para cuando el uso de probabilidades a priori no sea factible.

Por otro lado, hemos mostrado que el MT-FDA mejora los resultados logrados con el Tree-FDA y compite con el LFDA en algunas clases de funciones.

2.9. Trabajo futuro

2.9.1. Mezclas de árboles para el caso entero

Para una amplia clase de problemas la codificación en números enteros es una representación mucho más natural que la exclusivamente binaria. La representación entera impone varias dificultades a los FDAs. Una de las más importantes es la dimensión de las tablas de probabilidad.

En [137, 142] hemos presentado un FDA basado en árboles para tratar problemas definidos en codificación entera. FDAs que usan dependencias bivariadas requieren almacenar, para un problema de n variables, $\frac{n\cdot(n-1)}{2}$ marginales bivariados. El número total de entradas necesarias para almacenar todos los marginales en la tabla es $\frac{r_{MAX}^2\cdot n\cdot(n-1)}{2}$. Una mejora a este problema ha sido presentada en [137, 142]. Esta mejora podría ser extendida al caso de las mezclas de árboles.

2.10. Conclusiones

En este capítulo hemos introducido el MT-FDA como un ejemplo de la combinación de modelos probabilísticos en EDAs. MT-FDA fue uno de los primeros algoritmos [139] que usó semejante aproximación en EDAs. La aplicación de mezclas como una manera de combinar modelos probabilísticos es un resultado novedoso en el campo de los EDAs. El MT-FDA difiere de otros acercamientos, aparecidos casi simultáneamente, como ha sido señalado en la sección 1.8.2. El otro aspecto de novedad está en las modificaciones realizadas al algoritmo de aprendizaje para tratar con el problema del sobreajuste.

Por otro lado, hemos mostrado que algunas características específicas de la mezcla, como la existencia de una variable seleccionada, pueden utilizarse para organizar la búsqueda de una manera más eficiente.

En la parte experimental del capítulo hemos mostrado los resultados de los experimentos realizados en un banco de prueba de 999 instancias del problema 3-SAT, así como en un conjunto de funciones normalmente usadas para evaluar EAs. Hemos confirmado que la combinación de modelos simples en forma de mezclas mejora los resultados alcanzados con un solo modelo. Los FDAs que usan mezclas de árboles pueden también aventajar a FDAs con estructuras más complejas. Sin embargo, en el caso general no debemos esperar que la combinación de modelos simples aventaje a FDAs Bayesianos capaces de considerar mayores interacciones entre las variables. En este sentido, el resultado más importante de nuestra investigación está en haber insertado en el contexto EDA maneras de combinar los modelos, y algoritmos para aprender estos modelos. A continuación se listan los resultados alcanzados en el capítulo.

2.10.1. Resultados del Capítulo

1. Introducción del modelo probabilístico basado en mezclas de árboles para la modelación probabilística en EDAs.

2. Introducción del Algoritmo Evolutivo con estimación de distribuciones basado en Mezclas de Árboles y evaluación de su comportamiento en diferentes problemas de optimización.

3. Investigación del problema del aprendizaje de mezclas de árboles con restricciones en la complejidad del modelo.

4. Evaluación del algoritmo EM en el aprendizaje de árboles en el contexto de los EDAs.

5. Determinación de los factores relacionados con el aprendizaje de mezclas de árboles que influyen en el desempeño del MT-FDA.

6. Diseño de las estructuras de datos, programación en lenguaje C++ y puesta a punto de los programas que implementan el algoritmo MT-FDA.

3 Aproximaciones basadas en grafos de cliques ordenados: MN-FDA

3.1. Introducción

Una vía para la concepción de nuevos métodos para la estimación de distribuciones probabilísticas pasa por la expansión de la clase de factorizaciones que pueden usarse en la aproximación de distribuciones. La clase de factorizaciones que puede obtenerse de los modelos probabilísticos tradicionales es muy restringida. Extender la clase de factorizaciones que podrían emplearse en la aproximación de distribuciones implica el diseño de algoritmos para su identificación o construcción. En el caso de los EDAs, los métodos para inferir las aproximaciones tienen que estar acompañados por algoritmos eficaces de muestreo.

En este capítulo introducimos un EDA cuyo modelo probabilístico es construido a partir de grafos de independencia. El rasgo distintivo del algoritmo es el uso de una factorización de la probabilidad construida a partir de un grafo de cliques ordenado y que no satisface la propiedad de intersección corrida.

3.1.1. Motivaciones

Las motivaciones que nos llevan a investigar el tema tratado en este capítulo son las siguientes:

1. Aunque las factorizaciones basadas en modelos tradicionales han demostrado su utilidad en el marco de los EDAs, en realidad esta clase resulta bastante restrictiva a los efectos de la modelación. Por consiguiente es necesario evaluar la conveniencia de ampliar la clase factorizaciones que es posible utilizar en la aproximación de distribuciones probabilísticas en EDAs.

2. En el caso de problemas con múltiples interacciones, los modelos gráficos no dirigidos que se consideran en esta tesis, han recibido una atención secundaria en relación con las redes Bayesianas. Estos modelos poseen propiedades convenientes para su uso en EDAs y ameritan ser analizados.

3. Los algoritmos de aprendizaje de modelos gráficos basados en pruebas de independencia, tales como los que se analizan en este capítulo, han mostrado su utilidad en el contexto de los EDAs, sin embargo su estudio ha sido relegado a un segundo plano en comparación con la atención recibida por los métodos de aprendizaje basados en métricas.

3.1.2. Objetivos y metodología

A lo largo del capítulo perseguimos alcanzar los objetivos siguientes:

1. Evaluar críticamente las aproximaciones de la probabilidad basadas en factorizaciones válidas usadas por los EDAs.
2. Extender la capacidad de representación de los EDAs basados en grafos no dirigidos.
3. Presentar los grafos de cliques ordenados como una representación que generaliza naturalmente los árboles de cliques.
4. Demostrar que los grafos de cliques ordenados pueden representar aproximaciones consideradas válidas.
5. Presentar un algoritmo para aprender grafos de cliques ordenados a partir de los datos.
6. Presentar un algoritmo para muestrear grafos de cliques ordenados.
7. Introducir un EDA cuyo modelo probabilístico está basado en grafos de cliques ordenados.

Metodología

La metodología utilizada en este capítulo persigue la presentación de las factorizaciones basadas en grafos de cliques ordenados como un paso intermedio entre las factorizaciones válidas y las factorizaciones inválidas desordenadas, que serán introducidas en el próximo capítulo. El capítulo parte de un conjunto de definiciones tradicionalmente usadas en la teoría de modelos gráficos, para luego presentar los grafos de cliques ordenados como modelo con mayor poder de representación que los árboles de cliques. Se presenta un algoritmo para el aprendizaje de los grafos de cliques ordenados a partir de los datos. Este algoritmo es la base del EDA introducido en este capítulo. Algunos de los experimentos que validan el EDA introducido serán presentados en el próximo capítulo. Desde el punto de vista metodológico esta decisión está motivada por dos razones. En primer lugar, la estrecha relación entre las clases de factorizaciones empleadas por los EDAs introducidos en el presente y próximo capítulos. En segundo lugar, porque la presentación de sus respectivos resultados experimentales de manera conjunta permite resaltar las ventajas y desventajas de los respectivos EDAs.

Estructura del capítulo

En la próxima sección se introducen un conjunto de definiciones de la teoría de modelos gráficos que serán empleadas a lo largo del capítulo. Se explica en qué consisten las aproximaciones de distribuciones probabilísticas basadas en modelos gráficos. En la sección 3.3 se introduce un algoritmo para el aprendizaje a partir de los datos de una factorización aproximada basada en grafos de cliques ordenados. La complejidad computacional del algoritmo es analizada. En la sección 3.5 se introduce el Algoritmo Evolutivo con Distribución Factorizada basado en Redes

de Markov y se analiza su complejidad computacional. En la sección 3.6 se presentan experimentos que evalúan la capacidad del MN-FDA para optimizar funciones con pocas interacciones entre las variables. La sección 3.7 analiza un grupo de aproximaciones relacionadas con el trabajo presentado en el capítulo. Las conclusiones del capítulo son presentadas en la sección 3.8.

3.2. Grafos de independencia y factorizaciones

Partimos de la notación introducida en la sección 1.4.

Definición 3.1 (Grafo de cliques ordenado). *Un grafo de cliques ordenado es un grafo de cliques que satisface las condiciones siguientes:*

- *tiene un ordenamiento asociado a los nodos.*
- *tiene un nodo distinguido llamado raíz.*
- *un nodo pertenece al grafo si por lo menos una de las variables en cada nodo no está contenida en los nodos anteriores en el orden.*

3.2.1. Aproximaciones de la distribución basadas en grafos de independencia

El uso de las factorizaciones en la aproximación de distribuciones es importante porque las mismas permiten obtener representaciones condensadas de las distribuciones, lo que facilita su estimación y almacenamiento.

Existen métodos para encontrar factorizaciones válidas a partir de los grafos de independencia. Uno de los más conocidos es el método basado en la triangulización del grafo [68]. Es muy simple construir una factorización inválida, pero no podemos garantizar la precisión de una aproximación que ha sido construida a partir de una factorización inválida arbitraria. Se precisa de un criterio para la búsqueda en el espacio de factorizaciones inválidas, y una clasificación más detallada de este espacio también es deseable. En el próximo capítulo proponemos una clasificación inicial, es esta sección presentamos cómo construir un tipo particular de factorizaciones inválidas.

Si el grafo de independencia G es cordal, la proposición 1.15 garantiza que existe una factorización exacta y completa de la probabilidad, la cual está basada en los cliques del grafo. La factorización puede ser representada usando un árbol de cliques. Si G no es cordal, un super-grafo cordal de G puede ser encontrado agregando aristas a G en un proceso llamado triangulación. El problema estriba en que al adicionar aristas puede aumentar la dimensión del clique máximo del grafo. Si el tamaño de este clique es muy grande no resulta factible el cálculo de sus probabilidades marginales. El problema de encontrar una triangulación con máximo clique de tamaño mínimo es NP-completo.

Nuestra meta es encontrar, a partir del grafo de independencia, una factorización aproximada que contenga tantas dependencias como sea posible, pero sin agregar nuevas aristas al grafo.

Una factorización exacta comprendería todas las dependencias representadas en el grafo de independencia (Ver definición 1.13). *Asumiremos que las factorizaciones aproximadas de la probabilidad son más precisas en la medida en que incluyen un mayor número de las dependencias representadas en el grafo de independencia. Las factorizaciones aproximadas serán representadas usando un grafo de cliques ordenado calculado a partir del grafo de independencia.*

3.3. Aprendizaje de una factorización aproximada basada en un grafo de clique ordenado

El algoritmo para aprender el modelo probabilístico a partir de los datos cuenta con cinco pasos principales.

Algoritmo 3.1: Aprendizaje de un modelo basado en un grafo de cliques ordenado

1. Aprender un grafo de independencia G a partir de los datos (el conjunto de soluciones seleccionadas).
2. Si es necesario, refinar el grafo.
3. Encontrar la lista L de todos los cliques maximales de G.
4. Construir un grafo de cliques ordenado a partir de L.
5. Calcular los marginales para los cliques en el grafo de cliques ordenado.

Analizamos en detalle los diferentes pasos para aprender una factorización aproximada de los datos.

3.3.1. Aprendizaje de un grafo de independencia

La construcción de un grafo de independencia a partir de los datos puede lograrse por medio de pruebas de independencia. Para determinar si una arista pertenece al grafo se puede realizar un test de independencia para cada par de variables dado el resto. No obstante, desde un punto de vista algorítmico es importante reducir el costo de las pruebas de independencia. Por esta razón hemos adoptado la metodología seguida previamente por Spirtes [155]. La idea es empezar con un grafo no dirigido completo, y eliminar aristas realizando tests de independencia condicional entre los nodos unidos en el grafo, pero usando conjuntos condicionantes tan pequeños como sea posible.

Nuestra implementación actual puede realizar tests de independencia de orden cero, uno y dos. Sin embargo, en todos los experimentos presentados en esta tesis usamos tests de independencia de orden uno como máximo. A medida que el orden de los tests de independencia se incrementa, su fiabilidad disminuye y por lo tanto usar tests de independencia de un orden muy grande no es práctico, a menos que se empleen conjuntos de datos (poblaciones) de gran tamaño.

Test de independencia Chi-cuadrado

Para evaluar las relaciones de independencia usamos el test de independencia Chi-cuadrado, que es una prueba no-paramétrica de importancia estadística. Si dos variables X_i y X_j son independientes con un nivel de significación α previamente especificado, la arista que une a los vértices respectivos es eliminada del grafo. En el caso general podemos asumir que cada arista $i \sim j$ en el grafo de independencia inicial es pesada con un valor $w(i,j)$ que mide la magnitud de la interacción entre el par de variables. Esta información podría estar disponible a partir de información previa, o a partir de las pruebas estadísticas realizadas (el valor del test de Chi-cuadrado). Cuando tal información no está disponible, asumimos que todos los valores de las dependencias son iguales a un parámetro w' (es decir $w(i,j) = w', \forall i \sim j \in E$).

Las siguientes pruebas estadísticas fueron evaluadas:

$$X^2 = \sum_{x^i \in D} \frac{(OBS - EXP)^2}{EXP} \tag{3.1}$$

$$G^2 = 2 \cdot \sum_{x^i \in D} OBS \cdot log\left(\frac{OBS}{EXP}\right) \tag{3.2}$$

donde OBS y EXP son respectivamente los valores esperados y observados. En este caso $EXP = N \cdot p(x_i, x_A) \cdot p(x_j, x_A)$ y $OBS = N \cdot p(x_{\{i,j,A\}})$.

La expresión (3.1) es la estadística de chi-cuadrado de Pearson, y la (3.2) es la razón de verosimilitud logarítmica. Ambas tienen aproximadamente distribuciones χ^2 con grados de libertad iguales a la dimensión del modelo menos el número de parámetros libres. En nuestro caso el nivel de significación α es un parámetro del algoritmo.

Después de realizar experimentos preliminares con los tests (3.1) y (3.2), la prueba de Pearson fue escogida para la versión final de nuestro algoritmo. Un elemento importante a tener en cuenta en el uso del test χ^2 es que cuando se realiza un alto número de pruebas, el test revelará algunas relaciones falsamente significantes.

3.3.2. Refinamiento del grafo

Cuando el grafo de independencia es muy denso, podemos esperar que la dimensión de los cliques aumentará más allá de un límite factible. En estos casos es necesario poner un límite r al tamaño del clique máximo. Una alternativa para resolver este problema es, en un paso previo al cálculo de los cliques, hacer el grafo menos denso. Hemos seguido esta estrategia poniendo un límite $r-1$ al número máximo de aristas incidentes a cada vértice. Si el vértice tiene más de $r-1$ aristas incidentes, aquéllas con los pesos más bajos son removidas. De esta manera el clique máximo tendrá siempre un tamaño menor o igual que r.

El algoritmo de refinamiento evita introducir cualquier tendencia en la manera de remover las aristas. Sin embargo, tiene un inconveniente principal: podría darse el caso que existieran más de r variables dependientes de una variable X_i, pero el clique máximo donde X_i está incluida sea

más pequeño que r. En este caso, el procedimiento que elimina las aristas quitaría dependencias del grafo innecesariamente. Otra alternativa para evitar este problema es analizada en la próxima sección.

3.3.3. Cliques maximales del grafo

Para encontrar todos los cliques maximales del grafo se usa el algoritmo de Bron y Kerbosch [16]. Este algoritmo usa una técnica de ramificación y corte para cortar ramas que puedan conducir a cliques no maximales. Una vez que se han encontrado todos los cliques, se guardan en una lista L, y sus pesos se calculan a partir de la información sobre las dependencias. El peso de cualquier subgrafo G' de G se calcula como $W(G') = \sum_{i \sim j \in G'} w(i,j)$. De esta manera se calculan los pesos de los cliques maximales $w(C_i)$.

Una alternativa al algoritmo de refinamiento del grafo es adaptar el algoritmo para la búsqueda de cliques para comprobar cuando los cliques tienen tamaño mayor que r, y remover las aristas con los menores pesos. Este tipo de métodos puede necesitar más de una pasada del algoritmo para encontrar los cliques. Igualmente, debe tenerse cuidado en el crecimiento del número de cliques de pequeño tamaño en este tipo de aproximación al problema. En todos los experimentos presentados en esta tesis hemos utilizado solamente el algoritmo de refinamiento del grafo.

3.3.4. Construcción del grafo de cliques ordenado

El algoritmo 3.2 recibe la lista de cliques L con sus pesos, y devuelve una lista L' de los cliques en el grafo de cliques ordenado. El primer clique en L' es la raíz, y las etiquetas de los cliques en el grafo de cliques ordenado corresponden a su posición en L'. Cada clique en el grafo de cliques ordenado es un subconjunto de un clique en L.

Algoritmo 3.2: Algoritmo para aprender un grafo de cliques ordenado

1. Ordenar los cliques en L en un orden decreciente según sus pesos.
2. Adicionar el elemento $L(1)$ a la lista L'.
3. Eliminar el elemento $L(1)$ de L.
4. **while** L no esté vacía
5. Encontrar el primer elemento C en L tal que $C \cap (L'(1) \cup L'(2) \cdots \cup L'(NCliques)) \neq C$, y el número de variables en $C \cap (L'(1) \cup L'(2) \cdots \cup L'(NCliques))$ sea máximo.
6. **if** $C = \emptyset$ Eliminar todos los elementos en L
7. **else** Insertar C en L'.

Nos concentramos ahora en el paso 5 del algoritmo 3.2. La condición de maximizar el número de variables en $C \cap (L'(1) \cup L'(2) \cdots \cup L'(NCliques))$ implica que el clique C en L que tiene el mayor número de variables solapadas con todas las variables que ya están en L' será agregado a L'. El número de variables solapadas tiene que ser menor que el tamaño del clique, esta restricción significa que por lo menos una de las variables en C no ha aparecido antes. Si existen muchos

cliques con el mismo número máximo de variables solapadas, aquel que aparece primero en L se agrega a L'. Por otro lado, si el número máximo de variables solapadas es cero, entonces existe en el grafo de cliques más de una componente conexa. En este caso tenemos un conjunto de grafos de cliques, sin embargo hemos preferido abusar de la anotación y llamarlo grafo de cliques, ya tenga una o más componentes conexas. Finalmente, la incorporación de los cliques se detiene cuando todas las variables ya están en el grafo de cliques.

Para introducir en el algoritmo 3.2 la posibilidad de calcular un árbol de cliques basta modificar el paso 5. En lugar de considerar el solapamiento con todas las variables que ya están en L', se considera el solapamiento máximo entre todos los posibles solapamientos con cliques en L'. A continuación demostramos un teorema que ilustra la conveniencia de utilizar el algoritmo de aprendizaje en el caso de grafos cordales.

Teorema 3.1. *Sea G un grafo cordal usado como entrada del algoritmo 3.2, entonces el algoritmo dará como salida un árbol de cliques.*

Prueba: Demostramos que el algoritmo 3.2 dará como salida un ordenamiento de los cliques del grafo cordal que respeta las relaciones de precedencia determinadas por las estructura de un árbol de cliques correspondientes al grafo cordal. El elemento $L(1)$ de la lista inicial de cliques será la raíz del árbol. La propiedad de intersección corrida (1.12) garantiza que si dos cliques se solapan, entonces o son adyacentes, o las variables solapadas están contenidas en una cadena de cliques que los unen. El solapamiento máximo corresponde a cliques adyacentes en el árbol de cliques. De lo anterior se deduce que el paso 5 del algoritmo 3.2 garantiza que entre los cliques candidatos a ser adicionados se incorporará aquel que maximice el solapamiento con su padre que ya está en L', por lo que el ordenamiento final de los cliques respeta la estructura del árbol. □

Las probabilidades marginales para cada uno de los cliques son calculadas a partir de las frecuencias asociadas a cada configuración, y normalizando. En nuestra implementación, los parámetros del modelo probabilístico pueden ser modificados agregando una perturbación en forma de distribuciones definidas a priori.

3.3.5. Descripción de un ejemplo del algoritmo de aprendizaje

Introducimos un ejemplo de la aplicación del algoritmo 3.1. La información sobre las dependencias entre las 10 variables de un problema dado es representada por el grafo de independencia mostrado en la figura 3.1 (izquierda). Supongamos que el número máximo de aristas incidentes permitidas es seis. En el paso de refinamiento sólo vértices incidentes al vértice x_5 tienen que ser eliminados. Si existiera información disponible sobre las dependencias de cada arista, se quitarían las dos aristas con las dependencias más débiles. En el ejemplo presente asumimos que todas las dependencias son igualmente fuertes, y dos aristas arbitrarias ($x_1 \sim x_5$ y $x_5 \sim s_{10}$) son removidas. El grafo refinado se muestra en la figura 3.1 (derecha).

En el próximo paso se encuentran todos los cliques máximos del grafo. Hay nueve cliques máximos, todos de orden tres. También en este caso los cliques tienen el mismo peso, por consiguiente arbitrariamente se selecciona el clique (x_1, x_2, x_3) como la raíz.

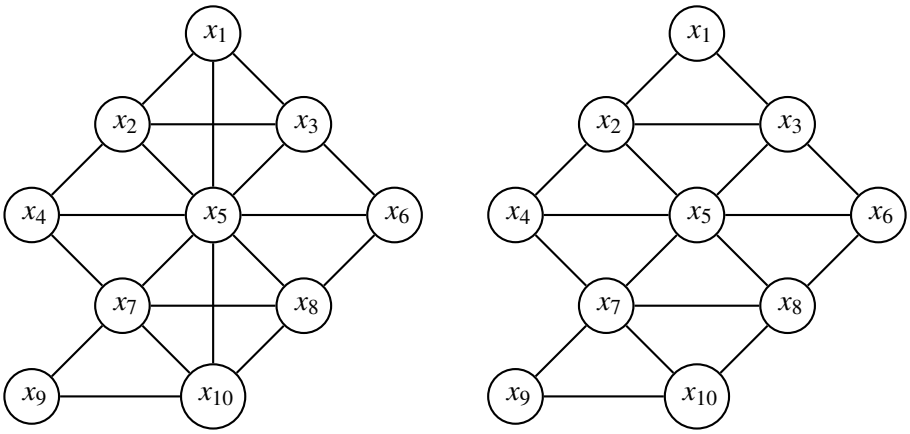

Figura 3.1: Grafos de independencia original y refinado.

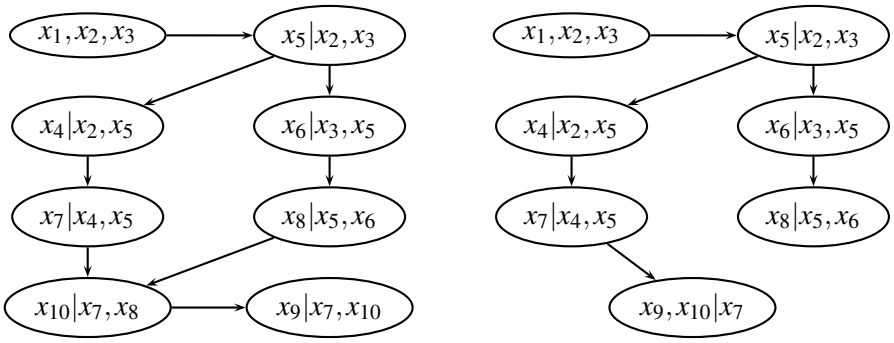

Figura 3.2: Grafo de cliques ordenado y árbol de cliques.

Construcción del grafo de cliques ordenado: En el primer paso el clique con máximo número de solapamientos con todas las variables en el grafo de cliques ordenado es (x_2, x_3, x_5). En el próximo paso, y considerando el número de solapamientos, cualquiera de los cliques (x_2, x_4, x_5) o (x_3, x_5, x_6) puede incorporarse. La figura 3.2 (izquierda) muestra el grafo de cliques final. En los cliques mostrados en la figura, cada nueva variable incorporada al grafo se representa a la izquierda de la barra. Sólo ocho de los cliques son incluidos en el grafo de cliques ordenado, el clique (x_5, x_7, x_8) falta. Su ausencia se explica porque si las variables del clique ya están en alguno de los cliques incorporados en el grafo, ese clique no se adiciona. Por lo tanto, el algoritmo para encontrar el grafo de cliques ordenado no puede garantizar que todas las dependencias serán capturadas en el grafo. Por otro lado, la factorización representada por el grafo de cliques ordenado es inválida porque existe un ciclo que comprende cliques diferentes.

Construcción del árbol de cliques: La figura 3.2 (derecha) muestra el árbol de cliques obtenido usando el algoritmo. Nótese que el árbol de cliques es capaz de representar menos dependencias que el grafo de cliques ordenado. Como el árbol de cliques prohíbe la existencia de ciclos, el clique (x_{10}, x_7, x_8) no puede ser representado totalmente.

3.3.6. Complejidad del algoritmo de aprendizaje

Aprendizaje de la estructura de la Red de Markov

En general la complejidad del algoritmo PC [155] es exponencial. El caso de peor complejidad puede acotarse por $\frac{n^2(n-1)^{k-1}}{(k-1)!}$ [155], donde k es la máxima cardinalidad del vértice. No obstante, la complejidad media puede ser mucho más pequeña que la cota anterior. Por otro lado, nuestra aplicación del PC sólo considera como máximo marginales de tercer orden. El número de operaciones necesarias para calcular los marginales es acotada superiormente por $N\binom{n}{3} + \binom{n}{2}$. La complejidad general de este paso es $O(Nn^3)$.

Refinamiento del grafo

Determinar la cardinalidad de los vértices, y seleccionar aquellos que tienen más de k aristas incidentes tiene un costo computacional de orden $O(n^2)$. Encontrar una permutación arbitraria de los nodos con más de k aristas incidentes tiene un costo de orden $O(n)$. El costo de ordenamiento de los valores de las dependencias para todos los nodos adyacentes a un nodo arbitrario i es a lo sumo de $O(nlog(n))$. Considerando el caso peor para los n nodos (después de las pruebas de independencia, ninguna arista ha sido eliminada y el grafo permanece completo), esta complejidad es de orden $O(n^2log(n))$. La complejidad del caso peor del algoritmo de refinamiento es $O(n^2log(n))$.

Búsqueda de los cliques maximales del grafo

En el trabajo original de Bron and Kerbosch [16] no se calcula la complejidad del algoritmo. Los grafos con el máximo número de cliques maximales tienen $3^{\frac{n}{3}}$ cliques maximales [10]. Por lo tanto el número de cliques maximales puede ser exponencial en n. Sin embargo, reduciendo el número de aristas del grafo reducimos esta cota dramáticamente. De las comparaciones con otros algoritmos para los que se han calculado cotas [10] puede estimarse la peor complejidad del algoritmo como $O(\mu^2)$, donde μ es el número de cliques maximales. Cuando hay a lo sumo k aristas para cada variable y $k << n$, una cota para el número de cliques puede ser dado por kn, y la complejidad del algoritmo de Bron y Kerbosch estimarse groseramente como $O(\mu^2) \approx O(n^2)$.

Construcción del grafo de cliques ordenado y el árbol de cliques

La construcción de grafo de cliques ordenado implica comprobar en cada paso cuál es el clique con mayor solapamiento con todas las variables que ya están en el grafo de cliques ordenado. Comprobar si todas las variables de un clique ya están en el grafo tiene complejidad $O(r)$, donde r es el tamaño del clique máximo. Esta comparación se hará en cada paso para todos los cliques que no se han incorporado. Pero la máxima cantidad de veces que puede realizarse es $n-1$ porque cada vez que se incorpora un clique al menos una variable no estará en el grafo de cliques ordenado. El costo es $r\sum_{i=1}^{n-1}(\mu - i)$, suponiendo que el número de cliques es kn, la complejidad

puede ser acotada por $O(rkn^2)$ En el caso del árbol de cliques la complejidad es mayor puesto que es necesario encontrar por separado el solapamiento con cada uno de los cliques que están en el árbol de cliques. Aquí el costo es $r\sum_{i=1}^{n-1} i(\mu - i)$, y la complejidad puede ser aproximada como $O(rkn^3)$.

Aprendizaje de los parámetros

La complejidad del aprendizaje de los parámetros depende del tamaño de la población N y el número de cliques μ, y su tamaño. El orden de este paso es $O(N\mu) \approx O(Nn^2)$.

Complejidad total del algoritmo de aprendizaje

La complejidad total del algoritmo es $O(Nn^3)$ y está determinada por el aprendizaje de la estructura de la red. El tamaño de la población N depende del problema. En los experimentos realizados en el próximo capítulo investigamos el escalado de este parámetro para diferentes problemas. Aun cuando la peor complejidad de las pruebas de independencia es $O(Nn^3)$, esta cota puede reducirse dramáticamente en el caso promedio porque sólo se hacen pruebas de mayor orden después que las pruebas menos costosas han demostrado que las variables son dependientes.

3.4. Muestreo del grafo de cliques

Siguiendo el orden determinado por las etiquetas del grafo de cliques ordenado se muestrean los puntos. Las variables que corresponden al primer clique en el grafo de cliques son instanciadas muestreando a partir de las probabilidades marginales. Para el resto de los cliques, cada subconjunto de variables que no ha sido instanciado es muestreado condicionado en las variables ya instanciadas que pertenecen al clique. El proceso es muy similar al del algoritmo PLS cuando se usa en árboles de cliques, donde cada clique se muestrea condicionalmente a sólo otro clique (su padre en el árbol de cliques). Existe sin embargo una diferencia importante: la definición de árbol de cliques no admite la existencia de ciclos. Un grafo de cliques ordenado puede contener ciclos, y este hecho permite la representación de más interacciones, pero no provoca que cambie esencialmente el desempeño del algoritmo de muestreo.

Para muestrear un grafo de cliques ordenado el algoritmo 3.3 es empleado. Este algoritmo garantiza que un clique pueda ser muestreado condicionado en las variables solapadas pertenecientes a más de un clique. Tal proceder es posible porque cada clique guarda las probabilidades marginales para todas las variables que contiene. Se muestrean las variables que no han sido instanciadas a partir de las probabilidades condicionales calculadas en el clique, fijando los valores de las variables ya instanciadas. El algoritmo de ordenamiento garantiza que una variable será muestreada condicionada en todas sus predecesoras.

Ejemplo 3.1. *Considere la figura 3.3 que representa un grafo de independencia con ocho cliques maximales de tamaño tres. Este modelo tiene ocho cliques, pero sólo seis variables. El*

Algoritmo 3.3: Algoritmo para muestrear un árbol de cliques

```
1  i = 0
2  d_i = ∅
3  while d_i ≠ X
4      c_i = C_i ∩ d_i
5          if c_i = ∅ muestrear a partir C_i
6          else muestrear a partir de C_i|c_i.
7      d_i = d_i ∪ C_i
```

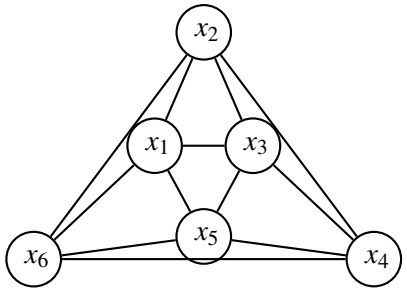

Figura 3.3: Grafo de independencia con ocho cliques maximales de tamaño tres.

muestreo de las variables basado en el algoritmo 3.3 puede hacerse a partir de la siguiente factorización que usa sólo tres cliques.

$$p(x) = p(x_1, x_2, x_3) \cdot p(x_5, x_6|x_1) \cdot (x_4|x_3, x_5)$$

A continuación presentamos un ejemplo de la generación de valores para las variables usando la factorización anterior. Primeramente se muestran las probabilidades marginales correspondientes a los tres cliques involucrados en la generación. Los valores de las variables (x_1, x_2, x_3) *se generan utilizando la probabilidad marginal del clique. El resto de las variables se instancia a partir de las probabilidades condicionales. Nótese que aunque el último clique se solapa con los otros dos, la variable* x_4 *puede ser instanciada a partir de las probabilidades condicionales calculadas usando únicamente los marginales almacenados en el clique que la contiene.*

$$p(x_1,x_2,x_3) = \{0{,}3, 0{,}05, 0{,}05, 0{,}1, 0{,}05, 0{,}05, 0{,}1, 0{,}3\}$$
$$p(x_1,x_5,x_6) = \{0{,}1, 0{,}05, 0{,}05, 0{,}3, 0{,}06, 0{,}04, 0{,}1, 0{,}3\}$$
$$p(x_3,x_4,x_5) = \{0{,}1, 0{,}05, 0{,}05, 0{,}3, 0{,}06, 0{,}04, 0{,}1, 0{,}3\}$$

$$(x_1,x_2,x_3) = (0,0,0)$$
$$p(x_5,x_6|x_1 = 0) = \{0{,}2, 0{,}1, 0{,}1, 0{,}6\}$$
$$(x_5,x_6) = (1,1)$$
$$p(x_4|x_3 = 0, x_5 = 1) = \{0{,}143, 0{,}857\}$$
$$x_4 = (1)$$

3.5. Algoritmo Evolutivo con Distribución Factorizada basado en Redes de Markov

Los algoritmos de aprendizaje y muestreo presentados en las secciones anteriores se han concebido para el diseño de un FDA capaz de aprender y usar factorizaciones que no cumplan la propiedad de intersección corrida. Nuestro algoritmo se llama Algoritmo Evolutivo con Distribución Factorizada basado en Redes de Markov (Markov Network FDA (MN-FDA)) [129], su pseudo-código se presenta en el algoritmo 3.4.

Algoritmo 3.4: MN-FDA

1. $t \Leftarrow 0$. Generar N puntos aleatoriamente.
2. **do** {
3. Evaluar los puntos en la función objetivo.
4. Seleccionar un conjunto S de $k \leq N$ puntos de acuerdo a un método de selección.
5. Aprender un grafo de cliques ordenado de los datos.
6. Calcular las probabilidades marginales para todos los cliques en el grafo de cliques ordenado.
7. Generar una nueva población muestreando a partir del grafo de cliques ordenado.
8. $t \Leftarrow t + 1$
9. } **until** Algún criterio de terminación ha sido satisfecho.

El MN-FDA tiene dos pasos principales correspondientes a los pasos de aprendizaje y muestreo del modelo explicados en las secciones anteriores. La eficacia del algoritmo de aprendizaje también dependerá de cuánta información anterior esté disponible. Cuando ninguna información anterior está disponible un modelo aproximado de las interacciones puede ser completamente

recuperado de los datos. A partir de este modelo se construye el grafo de cliques ordenado. La diferencia principal entre el MN-FDA y anteriores FDAs basados en modelos no dirigidos es que, al poder usar como modelo probabilístico un grafo de cliques ordenado, puede representar factorizaciones que no son válidas.

En nuestra aplicación actual, el nivel de significación α se le asigna el valor 0,75. Esta asignación fue motivada por la necesidad de capturar tantas dependencias como sea posible. Un nivel de significación bajo permite descubrir más dependencias en las últimas generaciones, cuando muchas variables tienden a ser independientes. Aún cuando algunas de las dependencias encontradas puedan ser falsas, esta opción es mejor que perder algunas de las dependencias reales. Si un número muy grande de dependencias son aprendidas, el algoritmo de refinamiento contribuirá a eliminar aquéllas con pesos más débiles, y equilibrará el efecto de un nivel de significación bajo. El número máximo de aristas incidentes para el algoritmo de refinamiento fue nueve.

3.5.1. Análisis de la complejidad del MN-FDA

Los pasos de iniciación, evaluación y selección del MN-FDA son comunes a los del MT-FDA, y la complejidad de estos pasos se corresponde con la calculada en la sección 2.6.1. La complejidad del algoritmo de aprendizaje ha sido calculada en la sección 3.3.6 y es $O(Nn^3)$. La complejidad del algoritmo de muestreo depende de la cantidad de puntos a muestrear, del tamaño del clique máximo, y de la cantidad de cliques que están en el grafo de cliques. El máximo número de tales cliques es n, luego la complejidad del paso de muestreo es: $O(2^r Nn)$. Asumiendo que r es constante la complejidad total del MN-FDA es $O(GNn^3)$, donde el tamaño de la población N y la cantidad de generaciones G varían según la dificultad del problema.

3.6. Experimentos

Los experimentos mostrados en esta sección se limitan a evaluar la capacidad del MN-FDA para optimizar funciones con pocas interacciones entre las variables. Con el fin de ilustrar el comportamiento del MN-FDA para otro tipo de funciones, presentaremos experimentos conclusivos en el capítulo próximo.

En el cuadro 3.1 se comparan los resultados del MN-FDA para diferentes funciones con los resultados publicados en [87] para el UMDA y el LFDA. Para las funciones consideradas en nuestros experimentos, los resultados del LFDA están disponibles en [87] sólo para los valores del parámetro α del LFDA que aparecen en el cuadro[1].

Puede observarse en el cuadro que en estas funciones el MN-FDA logró resultados iguales o mejores que el LFDA. Hemos observado que el algoritmo de aprendizaje usado por el MN-FDA descubre fácilmente variables que son independientes. La red Bayesiana aprendida por los FDAs Bayesianos puede tener problemas para reconocer independencias, particularmente si α es pequeño.

[1] El parámetro α tiene significados diferentes en el LFDA y el MN-FDA. En el LFDA sirve para especificar la densidad de la red Bayesiana.

OneMax				*BigJump*$(30,3,1)$				*Deceptive*4			
n	EDA	N	S	n	EDA	N	S	n	EDA	N	S
30	UMDA	30	75	30	UMDA	200	100	32	UMDA	800	0
30	$\text{LFDA}_{0,25}$	100	2	30	$\text{LFDA}_{0,25}$	200	58	32	FDA*	100	81
30	$\text{LFDA}_{0,5}$	100	38	30	$\text{LFDA}_{0,5}$	200	96	32	$\text{LFDA}_{0,25}$	800	92
30	$\text{LFDA}_{0,75}$	100	80	30	$\text{LFDA}_{0,75}$	200	100	32	$\text{LFDA}_{0,5}$	800	72
30	$\text{LFDA}_{0,25}$	200	71	30	$\text{LFDA}_{0,25}$	400	100	32	$\text{LFDA}_{0,75}$	800	12
30	MN-FDA	30	72	30	MN-FDA	100	92	32	MN-FDA	600	90
30	MN-FDA	100	98	30	MN-FDA	200	100	32	MN-FDA	800	100

Cuadro 3.1: Comparación entre el MN-FDA y otros FDAs para funciones unitarias.

3.7. Trabajo relacionado

Nuestro trabajo está relacionado con el trabajo anterior de Mühlenbein et al. [91], donde las factorizaciones aproximadas son reconocidas como una alternativa para la modelación de distribuciones probabilísticas. Nuestra investigación nos ha llevado a una manera diferente de encontrar estas aproximaciones. Nuestros resultados también están relacionado con el trabajo presentado por Brown et al. [17] en la aplicación de MRFs a GAs. Estos autores utilizan modelos probabilísticos de la función optimizada por el GA para generar nuevas soluciones. Nuestro trabajo muestra un número de diferencias significativas con este enfoque: El uso de pruebas estadísticas para aprender la estructura de interacciones. En [17] la estructura de las interacciones es conocida a priori. La construcción del grafo de cliques ordenado a partir del MN, y el uso del PLS en este grafo de cliques. En [17] el algoritmo de Metrópolis es el empleado para generar nuevas soluciones.

3.8. Conclusiones

En este capítulo hemos presentado un FDA que aproxima la distribución de probabilidad usando un grafo de cliques ordenado. El grafo de cliques ordenado se encuentra calculando los cliques maximales de un grafo de independencia que puede darse como entrada del algoritmo o ser aprendido de los datos. El MN-FDA generaliza otros FDAs al aprender factorizaciones que no tienen que ser válidas. Las conclusiones relativas al desempeño del MN-FDA en la optimización de otras funciones, y a partir de la comparación con otros algoritmos EDAs son presentadas en el capítulo siguiente. A continuación se listan los resultados alcanzados en el capítulo.

1. Introducción de los grafos de cliques ordenados como una representación que generaliza naturalmente los árboles de cliques.

2. Demostración de que los grafos de cliques ordenados pueden representar aproximaciones consideradas válidas.

3. Introducción de un algoritmo para aprender grafos de cliques ordenados a partir de los datos.

4. Introducción de un algoritmo para muestrear grafos de cliques ordenados.

5. Introducción de un EDA cuyo modelo probabilístico está basado en grafos de cliques ordenados.

6. Diseño de las estructuras de datos, programación en lenguaje C++ y puesta a punto de los programas que implementan el algoritmo MN-FDA.

4 Aproximaciones Kikuchi: MN-EDA y MK-EDA

4.1. Introducción

Los grafos de cliques ordenados son capaces de representar una clase más extensa de distribuciones de probabilidad que los árboles de cliques. No obstante, existen otras posibilidades de seguir extendiendo el poder de representación de los modelos usando factorizaciones inválidas. Un camino para la concepción de tales factorizaciones es el estudio de problemas afines abordados en otras disciplinas. Se introduce en este capítulo un método que aproxima distribuciones de probabilidad usando una clase de factorizaciones que hemos denominado "factorizaciones desordenadas". Para el aprendizaje de las factorizaciones, se presenta un algoritmo que es una reformulación de un procedimiento para la aproximación de la energía usado en Física Estadística y llamado aproximación Kikuchi. Con la introducción de la mezcla de aproximaciones Kikuchi, el capítulo cierra la investigación sobre el uso de las mezclas de distribuciones, iniciada en el capítulo 2.

4.1.1. Motivaciones

Las razones que nos llevan a investigar el tema tratado en este capítulo son las siguientes:

1. La necesidad de una taxonomía o manera de clasificar el conjunto de factorizaciones inválidas resulta un imperativo para el uso de tal tipo de factorizaciones, estudios en esta dirección son inexistentes en la literatura anterior a esta tesis.

2. El principal algoritmo de muestreo utilizado por los FDAs ha sido el PLS. En el campo de la Física Estadística existen otros algoritmos que han sido utilizados con buenos resultados en el muestreo. La investigación de las factorizaciones inválidas trae de la mano la necesidad de explorar la utilidad de otros algoritmos de muestreo.

3. Recientes resultados en el desarrollo de algoritmos de inferencia aproximada revelan la relación entre el problema de la minimización de la energía en sistemas físicos y la propagación de creencias en modelos gráficos. Inicialmente concebidos para modelos gráficos acíclicos, estos algoritmos pueden ser utilizados con resultados positivos en grafos con ciclos. La extensión de algunos de estos resultados a los EDAs merece ser considerada, especialmente en el diseño de algoritmos de aprendizaje y muestreo para modelos gráficos con ciclos.

4.1.2. Objetivos y metodología

A lo largo del capítulo perseguimos alcanzar los objetivos siguientes:

1. Estudiar y evaluar críticamente cuáles son los límites propios de los grafos de cliques ordenados en cuanto al poder expresivo del modelo.
2. Introducir la aproximación Kikuchi como modelo probabilístico.
3. Presentar métodos que permitan usar la aproximación Kikuchi como estimador y generador de la probabilidad.
4. Demostrar un conjunto de propiedades Markovianas de la aproximación Kikuchi.
5. Introducir un EDA cuyo modelo probabilístico está basado en aproximaciones Kikuchi.
6. Introducir las mezclas de aproximaciones Kikuchi como modelo probabilístico.
7. Presentar un algoritmo para el aprendizaje de mezclas de aproximaciones Kikuchi.
8. Introducir un EDA cuyo modelo probabilístico está basado en una mezcla de aproximaciones Kikuchi.
9. Presentar los resultados de la comparación de los EDAs introducidos con otros EDAs en la optimización de diferentes funciones.

Metodología

Para cumplimentar los objetivos partimos de una clasificación de las factorizaciones inválidas en dos grupos fundamentales: factorizaciones inválidas ordenadas y desordenadas. Esta clasificación nos permite concentramos en el estudio de las factorizaciones desordenadas. El trabajo se encamina entonces a la obtención de un tipo de factorización desordenada factible de ser utilizada en el contexto EDA. La propuesta obtenida de nuestro estudio la denominamos: aproximación Kikuchi. Como paso previo a la inserción de este tipo de factorización en un contexto EDA, demostramos teóricamente un grupo de propiedades de la misma. Métodos para el aprendizaje y muestreo de las aproximaciones Kikuchi se obtienen a partir de este análisis.

En una segunda etapa de la investigación introducimos las mezclas de aproximaciones Kikuchi. En el análisis de las mezclas, así como en la introducción de un algoritmo para su aprendizaje, se combinan resultados de la investigación expuesta en el capítulo 2, con los alcanzados anteriormente en este capítulo. Las mezclas de aproximaciones Kikuchi son utilizadas en la definición de otro algoritmo EDA. La sección experimental persigue dar una visión general del comportamiento de los EDAs en éste y anteriores capítulos. Incluimos comparaciones con algunos de los EDAs reconocidos en el estado del arte de nuestro campo de investigación.

Estructura del capítulo

En la sección 4.2 se propone una clasificación del conjunto de factorizaciones. Se realiza un breve recorrido por los modelos utilizados por diferentes EDAs, y se analiza la capacidad de representación de los respectivos modelos. En la sección 4.3 se introduce un método para la estimación de distribuciones de probabilidad a partir de la aproximación Kikuchi. En la sección 4.4 se demuestra que la aproximación Kikuchi construida a partir de un grafo cordal es válida. Se demuestra asimismo que la aproximación Kikuchi cumple la propiedad de Markov local. La sección 4.5 presenta un algoritmo para muestrear la aproximación Kikuchi utilizando muestreo de Gibbs. La sección 4.6 introduce un algoritmo para el aprendizaje de la aproximación Kikuchi a partir de los datos. En la sección 4.7 se introduce un EDA basado en aproximaciones Kikuchi y se describe su complejidad algorítmica.

En la sección 4.8 definimos la mezcla de aproximaciones Kikuchi. Presentamos un algoritmo para aprender las mezclas de aproximaciones Kikuchi, y un EDA basado en este modelo. Su complejidad computacional es analizada. En la sección 4.9 se presentan los experimentos que ilustran el comportamiento de los EDAs introducidos en la tesis, y evalúan el desempeño de los algoritmos en la optimización de funciones teóricas y prácticas. La sección 4.11 presenta las conclusiones del capítulo.

4.2. Expandiendo la clase de factorizaciones utilizadas por los EDAs

A partir de las factorizaciones basadas en grafos de cliques ordenados e introducidas en el capítulo anterior, proponemos una clasificación de las posibles factorizaciones.

Definición 4.1 (Factorización ordenada). *Sea una factorización basada en los cliques maximales de un grafo. Decimos que la factorización es ordenada cuando los cliques maximales del grafo pueden ordenarse de forma tal que exista al menos una variable en cada clique que no está contenida en los cliques ya ordenados. Cuando es imposible encontrar este orden, decimos que la factorización es desordenada.*

La propiedad de intersección corrida determina que cada factorización válida sea a su vez una factorización ordenada. La figura 4.1 describe la relación entre las dos diferentes clasificaciones de las factorizaciones.

Ejemplo 4.1. *En este ejemplo p_0 es una factorización ordenada y válida. Los cliques pueden formar un árbol de cliques. p_1 es una factorización ordenada e inválida. En este caso, los cliques no pueden ser ordenados en un árbol de cliques, pero un grafo de cliques ordenado puede ser formado.*

$$p_0 = \frac{p(x_1,x_2,x_3)\cdot p(x_1,x_2,x_5)\cdot p(x_1,x_3,x_6)\cdot p(x_2,x_3,x_4)}{p(x_1,x_2)\cdot p(x_1,x_3)\cdot p(x_2,x_3)}$$

$$p_1 = \frac{p(x_1,x_2,x_3)\cdot p(x_1,x_2,x_5)\cdot p(x_1,x_3,x_6)\cdot p(x_4,x_5,x_6)}{p(x_1,x_2)\cdot p(x_1,x_3)\cdot p(x_5,x_6)}$$

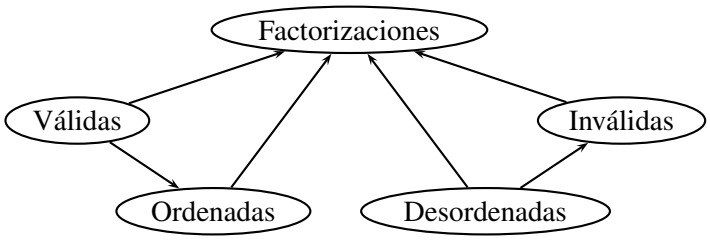

Figura 4.1: Clasificación de los tipos de factorizaciones.

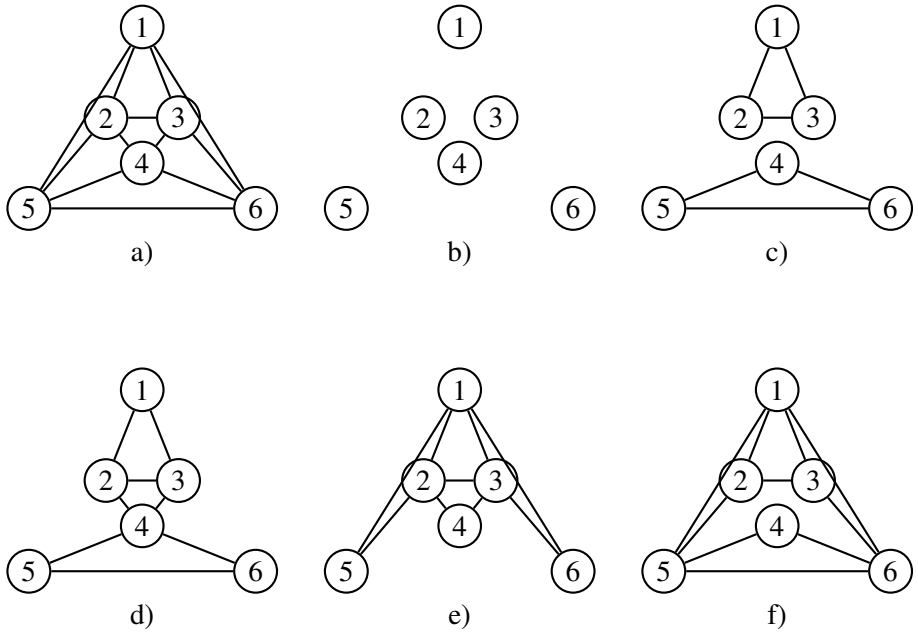

Figura 4.2: Grafos de independencia que ilustran ejemplos de diferentes tipos de factorizaciones usadas por los FDAs: a) grafo de independencia original, b) factorización válida usada por el UMDA, c) factorización válida usada por el ECGA, d),e) factorizaciones válidas usadas por el FDA* y el FDA-learning, f) factorización inválida usada por el MN-FDA.

Considere el grafo de independencia mostrado en la figura 4.2a). El grafo representa las dependencias que existen en una distribución de probabilidad dada. La aproximación más simple de esta distribución basada en una factorización es representada por el grafo de independencia mostrado en la figura 4.2b). No hay ninguna arista entre ningún par de vértices porque el modelo asume que todas las variables son independientes. Éste es el modelo usado por el UMDA.

Factorizaciones más complejas pueden ser obtenidas agrupando subconjuntos de variables con fuertes interacciones (figura 4.2c)). ECGA usa factorizaciones donde las variables están separadas en conjuntos no solapados. El FDA*, y el FDA-learning usan factorizaciones válidas con cliques que se solapan. La figuras 4.2d) y 4.2e) muestran modelos que pueden ser manejados

por estos algoritmos.

El MN-FDA extiende la capacidad de representación de los FDAs anteriores a partir del uso factorizaciones inválidas como la presentada en el ejemplo 4.1. Nótese que de la factorización aproximada mostrada en la figura 4.2b), a la mostrada en la figura 4.2f), hay un incremento en el poder de expresión de los modelos, en el sentido de que pueden representar un número mayor de las interacciones que existen en el modelo original (figura 4.2a)). Nótese también que ninguna de las aproximaciones puede representar todas las interacciones originales. En las próximas secciones introducimos un método para tratar con esta limitación de los modelos.

4.3. Aproximaciones Kikuchi de la probabilidad

El objetivo de esta sección es la introducción de un método para la estimación de distribuciones de probabilidad a partir de factorizaciones desordenadas. La aproximación lograda por este método puede considerarse como una aproximación de un Campo de Gibbs asociado a un grafo de independencias. Por consiguiente, primeramente los conceptos principales relacionados con MRFs son introducidos. Posteriormente, se explica brevemente el camino que va de los MRFs al uso de aproximaciones Kikuchi de la energía. La conexión entre las distribuciones de Gibbs o Boltzmann, y la aproximación Kikuchi de la energía es compleja, y debido a las limitaciones de espacio no podemos describirla aquí. Más detalles sobre esta relación pueden encontrarse en [170, 169].

Dado que el conjunto de potenciales de un MRF 1.16 puede representar todas las interacciones que existen en el MRF, prevemos su uso para representar la distribución de un conjunto del datos. Permítanos suponer que la estructura del MRF está disponible en la forma de un grafo de independencia, y las probabilidades marginales aproximadas para cada clique maximal del grafo también son conocidas. Estudiamos cómo recuperar, a partir de esta información, una distribución de probabilidad aproximada que capture la información contenida en los datos.

Para aprender un MRF, no es suficiente conocer los potenciales de vecindad, también es necesario calcular la función de partición Z. En general, calcular el valor de Z no es factible porque implica la suma en un número exponencial de estados. Por consiguiente, si pensamos aprender un MRF, una solución factible al problema de aproximar Z tiene que ser encontrada. Afortunadamente, un problema similar se ha tratado en el campo de la mecánica estadística.

4.3.1. Aproximación de la energía en sistemas de la Física Estadística

En mecánica estadística se tratan con frecuencia modelos de sistemas de múltiples partículas que interactúan entre sí. Las propiedades macroscópicas del sistema son determinadas por las interacciones entre los elementos microscópicos. Normalmente, las interacciones entre las componentes del sistema también determinan la probabilidad de la configuración global correspondiente. Cada estado x del sistema tiene una energía asociada $E(x)$. Un resultado fundamental de la mecánica estadística es que, en equilibrio termal, la probabilidad de un estado está dada por la Ley de Boltzmann:

$$p(x) = \frac{1}{Z(T)} e^{\frac{-E(x)}{T}}$$

donde T es la temperatura del sistema, y $Z(T)$ la correspondiente función de partición.

Yedidia et al. [170] propusieron usar el postulado de la ley de Boltzmann para definir la energía correspondiente a una probabilidad conjunta de un sistema no-físico. A la temperatura se le asigna un valor arbitrario uno, el cual simplemente determina una escala para las unidades en que se mide la energía. Propuestas similares se han usado con éxito en optimización [64, 91].

Pueden usarse diferentes técnicas para obtener una aproximación de la energía. Kikuchi et. al [63, 83] desarrollaron el Método Variacional basado en Agrupamientos (Cluster Variation Method (CVM)). En la base del CVM está la idea de aproximar la energía como una función de un conjunto de diferentes marginales. El conjunto no forma necesariamente una factorización exacta de la distribución, pero la forma en que se escogen los marginales influye críticamente en la calidad de la aproximación.

4.3.2. Descomposición de un grafo en regiones

Nuestro objetivo es utilizar los métodos empleados para aproximar la energía en la aproximación de la probabilidad. El primer paso es cómo escoger los marginales que participan en la aproximación. Nos concentramos en el problema de encontrar un conjunto apropiado de marginales que permita obtener una aproximación de la distribución.

Definimos una región R del sistema de vecindad $G =< V, E >$ como un conjunto $V' \subset V$. Una descomposición del grafo en regiones (Region based graph decomposition) es un conjunto de regiones $\mathcal{R}$, y un conjunto asociado de contadores (counting numbers) U, integrado por un contador c_R para toda region $R \in \mathcal{R}$. c_R será siempre un entero, pero puede ser cero o negativo para algunos valores de R.

La aproximación Kikuchi de la energía se define a partir de una descomposición del grafo en regiones, calculando las probabilidades marginales para cada región. Los métodos usados para construir la descomposición del grafo en regiones determinan la aproximación Kikuchi final.

Para ser válida[1] una descomposición, debe satisfacer varias restricciones que relacionan las regiones y los contadores. Inspirados en el trabajo de Yedidia et al. [169], llamamos a este subproblema como *problema de encontrar una descomposición válida del grafo en regiones*. En el método CVM, $\mathcal{R}$ se forma a partir de un conjunto inicial de regiones $\mathcal{R}_0$ tal que todos los vértices están en al menos una región de $\mathcal{R}_0$, y cualquier otra región en $\mathcal{R}$ es la intersección de una o más de las regiones en $\mathcal{R}$. El conjunto de regiones $\mathcal{R}$ es cerrado bajo la intersección, y puede ordenarse como un poset.

Decimos que un conjunto de regiones $\mathcal{R}$, y de contadores c_R es una descomposición del grafo válida cuando para cada variable X_i,

[1] La validez de descomposición del grafo en regiones no tiene relación con el concepto de factorizaciones válidas tratado anteriormente

$$\sum_{\substack{R\in\mathcal{R}\\ X_i\subseteq X_R}} c_R = 1 \tag{4.1}$$

Esta condición surge del hecho de que las aproximaciones aceptables de la componente de la entropía de la energía libre son aquéllas en las cuales la suma para cada una de las variables de los contadores asociados a cada región donde aparece la variable es uno [101].

Yedidia et al. [169] introducen el concepto de regiones para definir una aproximación de la energía libre a partir de una descomposición en regiones a partir de grafos de factores. Hemos adoptado su definición para explicar el problema más general de hallar una descomposición de nuestro grafo de independencia en regiones. Un aspecto fundamental en la determinación de una descomposición válida es la selección del conjunto inicial de regiones. El conjunto inicial de regiones usado por los físicos comprende usualmente regiones con el mismo tamaño y topología de conexión.

Introducimos en esta tesis un nuevo método para seleccionar el primer conjunto de regiones. La manera particular en que se seleccionan estas regiones determina varias propiedades convenientes de la aproximación de la probabilidad. Dado un grafo no dirigido arbitrario G, formaremos el conjunto $\mathcal{R}_0$ tomando una región para cada clique máximo en G. Como resultado, todas las regiones $R \in \mathcal{R}$ serán cliques porque son el resultado de la intersección de dos o más cliques. Llamamos a este tipo de descomposición del grafo en regiones como *descomposición del grafo en cliques*.

Definimos la aproximación Kikuchi de la probabilidad, denotada k como:

$$k(x) = \prod_{R\subset\mathcal{R}} p(x_R)^{c_R}, \tag{4.2}$$

donde $\mathcal{R}$ se determina a partir de una descomposición del grafo en regiones.

Ejemplo 4.2. *Descomposición en regiones válida correspondiente al grafo mostrado en la figura 4.2a).*

$$\mathcal{R}_0 = \{\{1,2,5\},\{1,3,6\},\{1,2,3\},\{2,3,4\},\{2,4,5\},\{3,4,6\},\{4,5,6\},\{1,5,6\}\}$$
$$\mathcal{R}_1 = \{\{1,2\},\{2,5\},\{1,3\},\{3,6\},\{2,3\},\{2,4\},\{3,4\},\{4,5\},\{4,6\},\{1,5\},\{1,6\},\{5,6\}\}$$
$$\mathcal{R}_2 = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\}\}$$
$$c_R = 1, \forall R \in \mathcal{R}_0; \quad c_R = -1, \forall R \in \mathcal{R}_1; \quad c_R = 1, \forall R \in \mathcal{R}_2$$

A partir de ahora limitaremos nuestro análisis a las aproximaciones Kikuchi construidas a partir de descomposiciones del grafo en cliques. A menos que se especifique lo contrario, cuando hablamos de una aproximación Kikuchi asumimos que se origina a partir de una descomposición del grafo en cliques.

Ejemplo 4.3. *Aproximación Kikuchi, construida a partir de la descomposición del grafo en cliques presentada en el ejemplo 4.2.*

$$\begin{aligned}
L_0 &= p(x_1,x_2,x_5)\cdot p(x_1,x_3,x_6)\cdot p(x_1,x_2,x_3)\cdot p(x_2,x_3,x_4)\\
&\quad \cdot p(x_2,x_4,x_5)\cdot p(x_3,x_4,x_6)\cdot p(x_4,x_5,x_6)\cdot p(x_1,x_5,x_6)\\
L_1 &= p(x_1,x_2)\cdot p(x_2,x_5)\cdot p(x_1,x_3)\cdot p(x_3,x_6)\cdot p(x_2,x_3)\cdot p(x_2,x_4)\\
&\quad \cdot p(x_3,x_4)\cdot p(x_4,x_5)\cdot p(x_4,x_6)\cdot p(x_1,x_5)\cdot p(x_1,x_6)\cdot p(x_5,x_6)\\
L_2 &= p(x_1)\cdot p(x_2)\cdot p(x_3)\cdot p(x_4)\cdot p(x_5)\cdot p(x_6)\\
k(x) &= L_0\cdot\frac{1}{L_1}\cdot L_2
\end{aligned}$$

En la sección 1.4.6 se explica el uso de los modelos gráficos como estimadores y generadores. Para usar la aproximación Kikuchi como un estimador de un punto dado x, se multiplica (o divide, de acuerdo al valor de c_R) por los marginales definidos en cada región.

El ejemplo 4.3 muestra la aproximación Kikuchi que corresponde a la descomposición del grafo en cliques presentada en el ejemplo 4.2. Nótese que los cliques maximales no pueden organizarse para formar una factorización ordenada. Por consiguiente, la factorización mostrada en este ejemplo es una factorización desordenada. En general, la aproximación Kikuchi será una factorización desordenada y por lo tanto inválida. No obstante, en ciertos casos podemos garantizar que la aproximación será válida. Este tema se analiza en la próxima sección.

Una función de probabilidad $\tilde{p}$ basada en la aproximación Kikuchi puede calcularse normalizando k.

$$\tilde{p}(x) = \frac{k(x)}{\sum_{x'} k(x')}$$

Sin embargo, la determinación de $\tilde{p}$ no es una alternativa realista, porque no resuelve el problema de calcular la función de partición $Z_k = \sum_x k(x)$.

4.3.3. Algoritmo para construir una descomposición en cliques válida

Presentamos una panorámica general del algoritmo que construye descomposiciones en cliques. La descomposición final tiene un solo conjunto de regiones $\mathcal{R}_1$ cuyos contadores $c_R, R \in \mathcal{R}_1$ son enteros diferentes de cero. El algoritmo 4.1 presenta los pasos para encontrar una descomposición en regiones. Sea $\mathcal{A}$ un conjunto auxiliar de regiones, y $\Phi(\mathcal{A})$ una función que encuentra todas las intersecciones entre las regiones en $\mathcal{A}$.

Primeramente, se construye un conjunto de regiones iniciales Q, agregando progresivamente regiones creadas a partir de la intersección entre otras regiones. Cuando todas las posibles intersecciones han sido agregadas, Q es ordenado. El orden es determinado por el criterio de

inclusión. Si $R_i \supseteq R_j$ entonces R_i precede a R_j en el orden. Después, cada región R en $\mathcal{Q}$ es inspeccionada, y el contador c_R se calcula como

$$c_R = 1 - \sum_{\substack{S \in \mathcal{R}_1 \\ S \supset R}} c_S \tag{4.3}$$

donde c_S es el contador de cualquier región S en $\mathcal{R}_1$ tal que S es un superconjunto de R. Si c_R es diferente de cero, la región se agrega a $\mathcal{R}_1$.

Todos los cliques maximales en $\mathcal{R}_0$ estarán en $\mathcal{R}_1$ con $c_R = 1$. El algoritmo encuentra el conjunto parcialmente ordenado de todas las regiones. Cuando semejante poset es cerrado bajo la intersección de regiones, y los contadores se calculan según (4.3), la descomposición en regiones es válida [83].

Algoritmo 4.1: Algoritmo para encontrar una descomposición en regiones válida

```
1   A = R_0
2   Q = A
3   do {
4       A = Φ(A)
5       Q = Q ∪ A
6   } until A = ∅
7   Encontrar un ordenamiento parcial de Q
8   R_1 = ∅
9   U = ∅
10  For R ∈ Q
11      c_R = 1 − Σ_{S∈R_1, S⊃R} c_S
12      if c_R ≠ 0 then
13          R_1 = R_1 ∪ R
14          U = U ∪ c_R
```

El ejemplo 4.4 muestra los diferentes pasos en el cálculo de los contadores que corresponden a la descomposición en cliques mostrada en el ejemplo 4.2.

Ejemplo 4.4.

$$\begin{aligned}
\mathcal{R}_0 &= \{\{1,2,5\},\{1,3,6\},\{1,2,3\},\{2,3,4\},\{2,4,5\},\{3,4,6\},\{4,5,6\},\{1,5,6\}\} \\
\mathcal{Q} &= \{\{1,2,5\},\{1,3,6\},\{1,2,3\},\{2,3,4\},\{2,4,5\},\{3,4,6\},\{4,5,6\},\{1,5,6\}, \\
&\quad \{1,2\},\{2,5\},\{1,3\},\{3,6\},\{2,3\},\{2,4\},\{3,4\},\{4,5\},\{4,6\},\{1,5\},\{1,6\},\{5,6\}, \\
&\quad \{1\},\{2\},\{3\},\{4\},\{5\},\{6\}\} \\
&\quad c_R = 1, \forall R \in \mathcal{R}_0; \\
c_{\{1,2\}} &= 1 - (c_{\{1,2,5\}} + c_{\{1,2,3\}}) = -1
\end{aligned}$$

Permítasenos suponer que todas las regiones de tamaño dos ya se han agregado a $\mathcal{R}_1$, los contadores para estas regiones son todos -1, como se muestra en el ejemplo 4.2. Los mismos se

calcularon de manera semejante al caso de $R = \{1,2\}$ mostrado anteriormente. Las otras seis regiones tienen tamaño uno y se agregarán a $\mathcal{R}_1$. Entre las regiones para las cuales $c_R = 1$ está $R = \{1\}$.

$$c_{\{1\}} = 1 - (c_{\{1,2,5\}} + c_{\{1,2,3\}} + c_{\{1,3,6\}} + c_{\{1,5,6\}} + c_{\{1,2\}} + c_{\{1,3\}} + c_{\{1,5\}} + c_{\{1,6\}}) = 1$$

4.4. Propiedades de la aproximación Kikuchi

La aproximación Kikuchi tiene varias propiedades que pueden usarse en el contexto de los EDAs. Primeramente mostramos que en ciertos casos la aproximación Kikuchi es una factorización válida. Posteriormente demostramos que la aproximación Kikuchi satisface la propiedad de Markov local.

4.4.1. Aproximaciones Kikuchi para grafos cordales

En el estudio de los algoritmos de propagación de creencias la relación entre los árboles de cliques y las aproximaciones Kikuchi de la energía ha recibido una atención particular. Este tema es importante porque ayuda a ilustrar las condiciones bajo las cuales la aproximación Kikuchi de la energía puede ser muy precisa, o incluso exacta. Un análisis detallado de este problema está más allá de los objetivos de esta tesis. Demostraremos que *la aproximación Kikuchi construida a partir de la descomposición en cliques de un grafo de independencia cordal que representa una distribución de probabilidad p, es una factorización válida*.

Dado que el grafo es cordal, es posible formar un árbol de cliques. Llamemos $\{s_1, s_2, \ldots, s_l\}$ a los cliques maximales en el árbol de cliques, and $\{r_1, r_2, \ldots, r_m\}$ a sus solapamientos. p puede ser factorizada como:

$$p(x) = \frac{\prod_{i=1}^{l} p(x_{s_i})}{\prod_{j=1}^{m} p(x_{r_j})^{(t_{r_j}-1)}}, \tag{4.4}$$

donde t_{r_j} es el número de cliques que se solapan en el solapamiento r_j. Un árbol de cliques es un grafo acíclico y conexo, en este caso es evidente (4.5).

$$\sum_{j=1}^{m} (t_{r_j} - 1) = l - 1 \tag{4.5}$$

Usamos (4.5) y la propiedad de intersección corrida 1.12 para demostrar el siguiente teorema.

Teorema 4.1. *Cuando el algoritmo 4.1 es aplicado a un grafo de independencia cordal se obtiene una descomposición equivalente a la factorización exacta determinada por el árbol de cliques correspondiente al grafo.*

Prueba:

Dado que todos los cliques maximales del grafo pertenecen a la descomposición del grafo en cliques, y tienen contadores iguales a uno, todos los s_i están en la factorización final. Mostraremos a continuación que las únicas otras regiones que pertenecen a la descomposición son los solapamientos $\{r_1, r_2, \ldots, r_m\}$ y tienen contadores $\{1-t_{r_1}, 1-t_{r_2}, \ldots, 1-t_{r_m}\}$.

Todos los solapamientos están en el conjunto de regiones $\mathcal{Q}$ porque son producto de la intersección de uno o más cliques maximales. Las intersecciones entre otras regiones también pertenecen a $\mathcal{Q}$. Asumimos que $\mathcal{Q}$ ha sido ya llenada y ordenada. También los cliques maximales que existían en $\mathcal{Q}$ han sido adicionados a $\mathcal{R}_1$. Recuérdese que estos cliques maximales no están incluidos en ningún otro clique y sus contadores son todos iguales a uno. Sean $(R_1, R_2, \ldots R_q)$ las regiones en $\mathcal{Q}$ que suceden a los cliques maximales. q es el número de estas regiones las cuales recordamos están ordenadas siguiendo un criterio de inclusión. Usamos el método de inducción para demostrar que solo las regiones en $\{r_1, r_2, \ldots, r_m\}$ serán adicionadas a $\mathcal{R}_1$.

R_1: R_1 no puede ser una intersección de solapamientos porque cualquiera de estos solapamientos la precedería en $\mathcal{Q}$. R_1 puede ser únicamente la intersección de cliques maximales que no están contenidos en ninguna otra intersección de cliques maximales. Por lo tanto, $R_1 = r_i \in \{r_1, r_2, \ldots, r_m\}$. El contador correspondiente a r_i se halla como:

$$c_{r_i} = 1 - \sum_{\substack{R \in \mathcal{R}_1 \\ R \supset r_i}} c_R = 1 - t_{r_i}$$

Donde en este primer caso todas las regiones R pertenecen a $\{s_1, s_2, \ldots, s_l\}$. Como $1-t_{r_i}$ es diferente de cero, R_1 es añadido a $\mathcal{R}_1$.

R_{k+1}: Supongamos que de las k regiones $\{R_1, R_2 \ldots R_k\}$ de $\mathcal{Q}$ analizadas: 1) Solo aquellas que pertenecen a $\{r_1, r_2, \ldots, r_m\}$ han sido adicionadas a $\mathcal{R}_1$ y, 2) Cada region r_i adicionada a $\mathcal{Q}$ tiene un contador igual a $1-t_{r_i}$. Entonces demostramos que si $R_{k+1} = r_j \in \{r_1, r_2, \ldots, r_m\}$, entonces $c_{R_{k+1}} = 1-t_{r_j}$ y R_{k+1} será añadida a $\mathcal{R}_1$. Si $R_{k+1} \notin \{r_1, r_2, \ldots, r_m\}$ entonces $c_{R_{k+1}} = 0$ y R_{k+1} no será añadida a $\mathcal{R}_1$.

Calculamos el valor del contador de R_{k+1} haciendo una descomposición de $\mathcal{R}_1$ en cliques maximales y solapamientos.

$$\begin{aligned} c_{R_{k+1}} &= 1 - \sum_{\substack{R \in \mathcal{R}_1 \\ R \supset R_{k+1}}} c_R \\ &= 1 - \left(\sum_{\substack{R \in \{s_1, s_2, \ldots, s_l\} \\ R \supset R_{k+1}}} c_R + \sum_{\substack{R \in \{\mathcal{R}_1 \cap \{r_1, r_2, \ldots, r_m\}\} \\ R \supset R_{k+1}}} c_R \right) \end{aligned} \tag{4.6}$$

Si $R_{k+1} = r_j \in \{r_1, r_2, \ldots, r_m\}$, y $r_j \not\subset r_i, r_i \in \mathcal{R}_1$ entonces la segunda parte en (4.6) es cero y

$$c_{r_j} = 1 - \sum_{\substack{R \in \{s_1, s_2, \ldots, s_l\} \\ R \supset r_j}} c_R = 1 - t_{r_j}$$

Este caso corresponde a un solapamiento que no está contenido en ningún otro solapamiento. Como $c_{r_j} \neq 0$, r_j será añadida a $\mathcal{R}_1$.

Si $R_{k+1} = r_j \in \{r_1, r_2, \ldots, r_m\}$, y r_j es el subconjunto de uno o más solapamientos que están ya en $\mathcal{R}_1$, todos los cliques maximales que se solapan en los solapamientos que contienen a r_j forman, junto con los solapamientos, una componente conexa. Este hecho es determinado por la propiedad de intersección corrida y la componente conexa es de hecho un árbol de cliques. Sea b el número total de cliques que contienen solapamientos donde está contenido r_j, entonces por (4.5),

$$\sum_{\substack{r_i \in \mathcal{R}_1 \\ r_i \supset r_j}} 1 - t_{r_i} = 1 - b$$

Exactamente uno de estos b cliques se solapa con los otros $t_{r_j} - 1$ cliques maximales en r_j. De lo contrario el árbol de cliques no sería conexo o acíclico. Rescribimos (4.6) como

$$c_{r_j} = 1 - [(b + t_{r_j} - 1) + (1 - b)] = 1 - t_{r_j}$$

$c_{r_j} \neq 0$ y r_j será añadida a $\mathcal{R}_1$.

El caso restante es cuando $R_{k+1} \notin \{r_1, r_2, \ldots, r_m\}$. En este caso R_{k+1} tiene que estar contenida en al menos uno de los solapamientos que ya están en $\mathcal{R}_1$ por la propiedad de intersección corrida. Todos los cliques maximales que se solapan en solapamientos que contienen a R_{k+1} también forman una componente conexa y acíclica. Pero en este caso no hay cliques maximales que tengan a R_{k+1} como solapamiento, de lo contrario $R_{k+1} \in \{r_1, r_2, \ldots, r_m\}$ lo cual es una contradicción y por lo tanto:

$$c_{R_{k+1}} = 1 - (b + 1 - b) = 0$$

Lo cual significa que la región R_{k+1} no será añadida a $\mathcal{R}_1$. Dado que todas las regiones $r_i \in \{r_1, r_2, \ldots, r_m\}$ estaban en $\mathcal{Q}$, el conjunto final de regiones $\mathcal{R}_1$ comprenderá todos los cliques maximales y en este caso la aproximación Kikuchi es igual a la factorización natural de p mostrada en (4.4). □

4.4.2. Propiedad de Markov local de la aproximación Kikuchi

Cuando el grafo de independencia es cordal, la aproximación Kikuchi se corresponde con una factorización válida de la distribución p subyacente. Si esto ocurre, $k(x) = p(x)$, pero en el caso general k no es una probabilidad. Sin embargo, definimos las funciones marginales y condicionales de la aproximación Kikuchi como:

$$k(x_S) = \sum_{\substack{x' \\ x'_S = x_S}} k(x') \tag{4.7}$$

$$k(x_A | x_B) = \frac{k(x_{\{A \cup B\}})}{k(x_B)} \tag{4.8}$$

Como k no es una probabilidad, $k(x_S)$ y $k(x_A|x_B)$ tampoco lo son. No obstante, ambas pueden usarse respectivamente como aproximaciones de $p(x_S)$ y $p(x_A|x_B)$.

A continuación demostraremos que $k(x)$ satisface la propiedad de Markov local con respecto a la estructura de residuales asociada a la descomposición del grafo en cliques. En las dos demostraciones siguientes usaremos el hecho de que para cualquier variable $X_i, i \in \{1,\ldots,n\}$, $k(x)$ puede ser factorizada en el producto de los marginales de regiones que contienen a X_i, y el producto de marginales de regiones que no contienen a X_i.

Sea $P(x,i,\mathcal{R}) = \prod_{\substack{R\in\mathcal{R}\\ X_R\supseteq X_i}} p(x_R)^{c_R}$, and $\bar{P}(x,i,\mathcal{R}) = \prod_{\substack{R\in\mathcal{R}\\ X_R\not\supseteq X_i}} p(x_R)^{c_R}$, entonces $k(x)$ puede expresarse como en (4.9).

$$\begin{aligned} k(x) &= \prod_{\substack{R\in\mathcal{R}\\ X_R\supseteq X_i}} p(x_R)^{c_R} \prod_{\substack{R\in\mathcal{R}\\ X_R\not\supseteq X_i}} p(x_R)^{c_R} \\ &= P(x,i,\mathcal{R})\cdot\bar{P}(x,i,\mathcal{R}) \end{aligned} \tag{4.9}$$

$P(x,i,\mathcal{R})$ y $\bar{P}(x,i,\mathcal{R})$ se han introducido por razones de conveniencia en la notación, para tener una representación más concisa de $k(x)$. También, la siguiente notación no convencional se usará para representar la sumatoria en todas las variables excepto X_S, cuando $X_S = x_s$.

$$p(x_S) = \sum_{\sim\{x_S\}} p(x') = \sum_{\substack{x'\\ x'_S = x_S}} p(x')$$

Nótese el uso de esta notación en la ecuación (4.10), que junto con las implicaciones (4.11) y (4.12), se usa en nuestras demostraciones. La implicación (4.11) se deriva del hecho de que en la descomposición en cliques cualquier región es un clique. Por consiguiente, cualquier conjunto de variables en la misma región que X_i pertenece a su clausura.

$$\sum_{\sim cl(x_i)} \sum_{\sim\{x\setminus x_i\}} p(x') = \sum_{\sim bd(x_i)} p(x') \tag{4.10}$$

$$X_R \supseteq X_i \Rightarrow X_R \subseteq cl(X_i) \tag{4.11}$$

$$X_R \not\supseteq X_i \Rightarrow X_R \subseteq X \setminus X_i \tag{4.12}$$

Primeramente damos una expresión simplificada para $k(x_i|bd(x_i))$, posteriormente demostramos que $k(x_i|x\setminus x_i) = k(x_i|bd(x_i))$.

Teorema 4.2.

$$k(x_i|bd(x_i)) = \frac{P(x,i,\mathcal{R})}{\sum_{\sim\{x\setminus x_i\}} P(x,i,\mathcal{R})} \tag{4.13}$$

Prueba:

$$
\begin{aligned}
& k(x_i|bd(x_i)) \\
&= \frac{\sum_{\sim cl(x_i)} P(x',i,\mathcal{R})\bar{P}(x',i,\mathcal{R})}{\sum_{\sim bd(x_i)} P(x',i,\mathcal{R})\bar{P}(x',i,\mathcal{R})} \\
&= \frac{P(x,i,\mathcal{R})\sum_{\sim cl(x_i)} \bar{P}(x',i,\mathcal{R})}{\sum_{\sim cl(x_i)}\sum_{\sim\{x\setminus x_i\}} P(x',i,\mathcal{R})\bar{P}(x',i,\mathcal{R})} \\
&= \frac{P(x,i,\mathcal{R})\sum_{\sim cl(x_i)} \bar{P}(x',i,\mathcal{R})}{\sum_{\sim cl(x_i)}\left(\bar{P}(x',i,\mathcal{R})\sum_{\sim\{x\setminus x_i\}} P(x',i,\mathcal{R})\right)} \\
&= \frac{P(x,i,\mathcal{R})}{\sum_{\sim\{x\setminus x_i\}} P(x',i,\mathcal{R})}\frac{\sum_{\sim cl(x_i)} \bar{P}(x',i,\mathcal{R})}{\sum_{\sim cl(x_i)} \bar{P}(x',i,\mathcal{R})} \\
&= \frac{P(x,i,\mathcal{R})}{\sum_{\sim\{x\setminus x_i\}} P(x',i,\mathcal{R})} \\
&\square
\end{aligned}
$$

Teorema 4.3.

$$k(x_i|x\setminus x_i) = k(x_i|bd(x_i)$$

Prueba:

$$
\begin{aligned}
& k(x_i|x\setminus x_i) \\
&= \frac{k(x)}{k(x\setminus x_i)} \\
&= \frac{P(x,i,\mathcal{R})\cdot\bar{P}(x,i,\mathcal{R})}{\sum_{\sim x\setminus x_i} P(x',i,\mathcal{R})\cdot\bar{P}(x',i,\mathcal{R})} \\
&= \frac{P(x,i,\mathcal{R})\cdot\bar{P}(x,i,\mathcal{R})}{\bar{P}(x,i,\mathcal{R})\sum_{\sim x\setminus x_i} P(x',i,\mathcal{R})} \\
&= \frac{P(x,i,\mathcal{R})}{\sum_{\sim x\setminus x_i} P(x',i,\mathcal{R})} \\
&= k(x_i|bd(x_i) \\
&\square
\end{aligned}
$$

También la versión normalizada de la aproximación Kikuchi satisface el teorema 4.3.

$$\begin{aligned}\tilde{p}(x_i|bd(x_i)) &= \frac{\tilde{p}(cl(x_i))}{\tilde{p}(bd(x_i))} \\ &= \frac{\sum_{\sim\{cl(x_i)\}} \tilde{p}(x')}{\sum_{\sim\{bd(x_i)\}} \tilde{p}(x')} \\ &= \frac{\sum_{\sim\{cl(x_i)\}} k(x') \setminus Z_k}{\sum_{\sim\{bd(x_i)\}} k(x') \setminus Z_k} \\ &= k(x_i|bd(x_i))\end{aligned}$$

Los teoremas 4.2 y 4.3 son ambos importantes para nuestro propósito de muestrear la aproximación Kikuchi. El teorema 4.3 demuestra que en la aproximación Kikuchi cada variable es independiente del resto dada su clausura. Por consiguiente, para muestrear solamente es necesario utilizar la probabilidad condicional de cada variable dada su clausura. El teorema 4.2 da una expresión conveniente para el cálculo de la probabilidad condicional de una variable dada su clausura. Esta probabilidad puede ser calculada a partir de las probabilidades marginales definidas en los cliques que involucran variables en la clausura. El teorema 4.2 también muestra una propiedad interesante de la aproximación Kikuchi. Para cada variable X_i, las funciones $k(x_i|bd(x_i))$ son funciones de probabilidad debido a la normalización en el denominador.

4.5. Muestreo de la aproximación Kikuchi usando el muestreo de Gibbs

En la sección 4.3.2 hemos explicado cómo usar la aproximación Kikuchi como un estimador. Para usar esta aproximación como un generador, haremos un uso conveniente de la propiedad de Markov que satisface la aproximación. Como k no es una distribución de probabilidad, asumimos que los puntos serán generados a partir de su expresión normalizada $\tilde{p}(x)$. Para muestrear puntos a partir de $\tilde{p}(x)$, usamos el algoritmo de muestreo de Gibbs (Gibbs Sampling algorithm (GS) [49]). Suponemos que aunque $\tilde{p}(x)$ es demasiado compleja para deducir muestras directamente de ella, sus distribuciones condicionales, $\tilde{p}(x_i|x \setminus x_i) = k(x_i|x \setminus x_i)$ son factibles de calcular.

Para obtener muestras de $\tilde{p}(x)$, comenzar con un estado inicial $x^0 = (x_1^0, \cdots, x_n^0)$. En cada momento t del GS, una posición i a ser actualizada será escogida a partir de x^t, y el nuevo vector x^{t+1} se selecciona usando la regla de transición mostrada en (4.14).

$$x^{t+1} = \begin{cases} x_j^{t+1} = x_j^t & \text{for } j \neq i \\ x_j^{t+1} \approx k(x_i^t|bd(x_i^t)) & \text{for } j = i \end{cases} \tag{4.14}$$

Definimos VS, Cy, e In como los parámetros del GS. VS es el tipo de esquema de visita de las variables, y define la manera en que las variables se seleccionan para ser actualizadas. Esquemas de visitas aleatorios ($VS = 0$), o fijos ($VS = 1$) pueden ser utilizados. Cy es el número de ciclos del GS. Un ciclo comprende la actualización de n variables. In es un parámetro que determina

la manera en que se construye el vector inicial del GS. El vector a partir del cual se aplica el GS puede seleccionarse aleatoriamente ($In = 0$), o muestrearse a partir de una factorización aproximada calculada usando un subgrafo cordal del grafo de independencia ($In = 1$).

El GS garantiza que después de un tiempo suficientemente largo, digamos t pasos, cada estado de la cadena puede tomarse como una muestra de la probabilidad que se pretende aproximar. En nuestro caso las probabilidades marginales se calculan usando el estimador de Laplace, por consiguiente las distribuciones condicionales son todas positivas y el GS determinará una cadena de Markov ergódica, la cual ha de converger a la probabilidad designada $\tilde{p}(x)$.

4.6. Aprendizaje de la aproximación Kikuchi a partir de los datos

En las secciones anteriores hemos asumido que los cliques maximales del grafo de independencia, y sus probabilidades marginales son conocidas. En el contexto de un EDA, se necesita un algoritmo para aprender la estructura y parámetros del modelo. El algoritmo 4.2 es una extensión del algoritmo introducido en el capítulo anterior para aprender factorizaciones ordenadas de los datos. Los pasos del uno al tres son los mismos, en los pasos cuatro y cinco se construye la aproximación Kikuchi.

Algoritmo 4.2: Algoritmo para aprender la aproximación Kikuchi de los datos

1. Aprender un grafo de independencia G de los datos (el conjunto de las soluciones seleccionadas).
2. Si resulta necesario, refinar el grafo.
3. Encontrar el conjunto $\mathcal{C}$ de todos los cliques maximales de G.
4. Construir la descomposición del grafo en cliques.
5. Encontrar las probabilidades marginales para las regiones de la descomposición.

4.7. Un algoritmo EDA basado en aproximaciones Kikuchi

Hemos llamado al algoritmo evolutivo basado en la aproximación Kikuchi como MN-EDA (Markov Network EDA) [128, 130] pues este algoritmo resulta una generalización del MN-FDA introducido en el capítulo anterior. Su pseudo-código se muestra en el algoritmo 4.3. A nuestro juicio existe una diferencia fundamental entre las factorizaciones ordenadas y desordenadas. Por esta causa consideramos a los EDAs basados en factorizaciones desordenadas como EDAs propiamente dichos, y no como FDAs. Esta razón explica que decidamos nombrar al algoritmo introducido en el capítulo anterior MN-FDA, mientras que los que se incluyen en éste son llamados EDAs.

Los parámetros utilizados para las pruebas independencia y el n úmero máximo de cliques son los mismos utilizados para el MN-FDA y presentados en la sección 3.5.

Algoritmo 4.3: MN-EDA

1 $t \Leftarrow 0$. Generar N puntos aleatoriamente.
2 **do** {
3 Evaluar los puntos en la función objetivo
4 Seleccionar un conjunto S de $k \leq N$ puntos de acuerdo a un método de selección.
5 Aprender una aproximación Kikuchi de S usando el algoritmo 4.2.
6 Generar N nuevas soluciones muestreando la aproximación Kikuchi usando (4.14).
7 $t \Leftarrow t+1$
8 } **until** Algún criterio de terminación ha sido satisfecho

Llamaremos MN-EDA^f al MN-EDA que usa un modelo fijo de las interacciones. Este algoritmo puede recibir la descomposición basada en cliques, o calcularlas a partir de un grafo de independencia dado. Una vez que la descomposición en cliques ha sido construida, será la misma en todas las generaciones del algoritmo. En cada generación, el MN-EDA^f sólo realiza un aprendizaje paramétrico de las probabilidades marginales que corresponden a las regiones.

4.7.1. Análisis de la complejidad del MN-EDA

Los pasos de iniciación, evaluación y selección del MN-EDA son comunes a los del MT-FDA, y la complejidad de estos pasos se corresponde con la calculada en la sección 2.6.1. La diferencia principal con el algoritmo MN-FDA está en el cálculo de la descomposición del grafo en cliques, y en la utilización del GS como método de muestreo.

El costo del método para encontrar una descomposición en cliques depende claramente del número de cliques, su tamaño, y lo más importante, del número de solapamientos entre las variables. Este costo es muy difícil de estimar, y la estimación que proponemos está lejos de ser óptima.

El número de comparaciones necesarias para encontrar el primer grupo de solapamientos es $\frac{(\mu)(\mu-1)}{2}$, donde μ es el número de cliques en el grafo. Éste también es un límite para el número de nuevas regiones que pueden encontrarse en un primer conjunto. El proceso se repite tomando este número máximo de regiones como un límite. Un estimador grosero del número total de regiones puede ser $N_R = r \cdot \frac{(\mu)(\mu-1)}{2}$, donde r es el tamaño del clique máximo en el grafo. Tomando $\mu = kn$, $N_R \approx rn^2k^2$.

El costo del algoritmo de muestreo puede estimarse como:

$$N \cdot C_y \cdot \sum_{i=1}^{n} \sum_{R \in \mathcal{R}} (2^r - 1) \approx Nn2^r,$$

donde C_y es el número de ciclos del GS. El costo es exponencial en el tamaño de clique máximo, y lineal en N y n. Considerando C_y y r constantes, la complejidad total del MN-EDA es

$O(GNn^3)$, donde el tamaño de la población N y la cantidad de generaciones G varían según la dificultad del problema. Al igual que en el caso del algoritmo MN-FDA, y como se explica en 3.3.6, la complejidad está dominada por el costo de las pruebas de independencia. Investigamos el problema del escalado del tamaño de la población en la sección de los experimentos.

4.8. Un EDA basado en una mezcla de aproximaciones Kikuchi

Como fue discutido en el capítulo 2, una mezcla de modelos de un determinado tipo puede llegar a representar dependencias de mayor complejidad que un modelo simple del mismo tipo. La posibilidad de usar una aproximación Kikuchi como un estimador señala la conveniencia de explorar su uso como componente de mezclas, y la necesidad de un algoritmo para aprender mezclas de aproximaciones Kikuchi.

En esta sección definimos la mezcla de aproximaciones Kikuchi. Este resultado surge de una combinación de los temas analizados en el capítulo 2, y los presentados anteriormente en este capítulo. Presentamos un algoritmo para aprender las mezclas de aproximaciones Kikuchi, y un EDA basado en este modelo. Los resultados presentados en el capítulo 2 se usan extensivamente en esta sección.

4.8.1. Mezcla de aproximaciones Kikuchi

Definición 4.2 (Mezcla de aproximaciones Kikuchi). *Una mezcla de aproximaciones Kikuchi se define como una distribución de la forma:*

$$Q(x) = \sum_{j=1}^{m} \lambda_j \tilde{p}^j(x) \tag{4.15}$$

con $\lambda_j \geq 0$, $j = 1,\ldots,m$, $\sum_{j=1}^{m} \lambda_j = 1$. $\tilde{p}^j(x)$ *es la aproximación Kikuchi normalizada para la componente* j.

En este caso las descomposiciones en cliques, con sus respectivos marginales asociados constituyen las componentes de la mezcla. Cuando todas las componentes comparten la misma descomposición en cliques, y las distribuciones marginales son diferentes en cada componente, llamamos al modelo una *mezcla de aproximaciones Kikuchi con estructura compartida*. Nótese que cuando todas las descomposiciones en cliques corresponden a grafos cordales, estamos en presencia de una mezcla de árboles del cliques. De hecho, la mezcla de aproximaciones Kikuchi abre la posibilidad de combinar información de factorizaciones válidas e inválidas en un solo modelo.

4.8.2. EM para el aprendizaje de las aproximaciones

Para aprender una mezcla de aproximaciones Kikuchi proponemos usar una versión del algoritmo IEM presentado en la sección 2.4. Para presentar este algoritmo, que hemos llamado

Kikuchi-IEM, no asumiremos ningún supuesto con respecto a la precisión de una mezcla de aproximaciones Kikuchi en el caso general.

Una reformulación del problema de aprendizaje es: Dado un conjunto de observaciones $D = \{x^1, x^2, \ldots, x^N\}$, debemos encontrar la mezcla de aproximaciones Kikuchi Q que satisface:

$$Q = \arg\max_{Q} \sum_{i=1}^{N} \log Q(x^i) \tag{4.16}$$

El algoritmo Kikuchi-IEM utiliza la función de verosimilitud llamada verosimilitud logarítmica completa, la cual es la verosimilitud logarítmica tanto de los datos observados como de los no observados dado el modelo estimado en cada paso $M = \{m, \tilde{p}^j, \lambda_j\}, j = 1, ..., m$.

$$lc(x^{1\ldots N}, z^{1\ldots N}|M) = \sum_{i=1}^{N} \sum_{j=1}^{m} \delta_{j,z^i} (\log \lambda_j + \log \tilde{p}^j(x^i)) \tag{4.17}$$

donde δ_{j,z^i} es igual a uno si z^i es igual al j-ésimo valor de la variable de selección, y cero en otro caso. La idea en que se basa el algoritmo Kikuchi-IEM es la misma que la del IEM, es decir computar y optimizar el valor esperado de lc. Sin embargo, la manera en que los pasos de Estimación (E) y de Maximización (M) se llevan a cabo en ambos algoritmos es esencialmente diferente. En el paso E del Kikuchi-IEM, usamos los valores de las aproximaciones Kikuchi para estimar la probabilidad a posteriori de la variable oculta para cada una de las observaciones. Esto significa estimar la probabilidad de que cada componente de la mezcla genere el punto x^i.

$$\Pr[z^i = j|x^i, M] = \gamma_j(x^i) = \frac{\lambda_j \tilde{p}^j(x^i)}{\sum_{j'} \lambda_{j'} \tilde{p}^{j'}(x^i)} = E[\delta_j, z^i] \tag{4.18}$$

Los valores de Γ_k y $P^k(x^i)$ se calculan como en (2.9) y (2.10). Como en el caso de mezcla de árboles, las sumas $\Gamma_j \in [0, N]$ pueden ser interpretadas como el número total de puntos que son generados por la componente $\tilde{p}^j$. Normalizando las probabilidades a posteriori $\gamma_j(i)$ con Γ_j obtenemos las distribución de probabilidad P^j definida sobre el conjunto de datos.

El paso M del algoritmo Kikuchi-IEM busca estimar los parámetros del modelo de forma tal que se maximice $E[lc|x^{1,\ldots,N}, M]$. Para esto es necesario encontrar el modelo que mejor aproxima los datos en cada componente. En el caso de la mezcla de árboles, este problema puede resolverse usando el algoritmo Chow-Liu.

Mientras es posible calcular el árbol que da la mejor aproximación los datos, un resultado similar parece no ser posible de alcanzar para la aproximación Kikuchi. No hemos propuesto siquiera un procedimiento para la búsqueda en el espacio de las aproximaciones Kikuchi[2]. Por consiguiente, en la versión Kikuchi-IEM del paso M, una aproximación Kikuchi se calcula usando el algoritmo 4.2.

[2]Hipotéticamente tal algoritmo podría ser concebido, utilizando por ejemplo una métrica asociada a cada descomposición en cliques y una penalidad a la complejidad de tal descomposición

Señalamos el siguiente hecho: el éxito del paso M no depende críticamente de la calidad de la aproximación Kikuchi. Si podemos encontrar en cada componente una aproximación Kikuchi (no la mejor aproximación Kikuchi) que sea una aproximación más precisa de los datos que la mejor aproximación que es posible obtener con un árbol, entonces podríamos esperar que la mezcla de estas componentes dará una aproximación mejor que la mezcla conformada por los mejores árboles.

La ventaja de usar aproximaciones Kikuchi como componentes está dada por el hecho de que éstas pueden representar interacciones más complejas. La desventaja es que no garantizaremos que un máximo (incluso uno local) de la verosimilitud logarítmica ha sido alcanzado, como es el caso en el IEM. No obstante, recordamos que si la distribución puede aproximarse con un árbol de cliques con un tamaño máximo de clique manejable, nuestro algoritmo para aprender la aproximación Kikuchi puede aprender una aproximación muy precisa.

El Kikuchi-IEM precisa de un método para construir la mezcla inicial de aproximaciones Kikuchi. En nuestros experimentos hemos calculado la estructura y parámetros de cada aproximación Kikuchi inicial a partir de los datos y para diferentes valores del parámetro α. Para cada valor α se obtienen grafos de independencia y descomposiciones en cliques diferentes. Para la componente i de la mezcla inicial de aproximaciones Kikuchi el valor del α se calcula como $\alpha_i = 0{,}75 - 0{,}05(i-1)$.

Algoritmo 4.4: Kikuchi IEM

1 $t \leftarrow 1$; Fijar mezcla inicial de aproximaciones Kikuchi.
2 Si algún criterio de terminación ha sido satisfecho, devolver mezcla inicial y salir.
3 **do** {
4 **for** $j \Leftarrow 1$ **to** m
5 **for** $i \Leftarrow 1$ **to** N
6 Calcular $\gamma_j^i, P^j(x^i)$ usando ecuaciones (4.18) y (2.10).
7 **for** $j \Leftarrow 1$ **to** m
8 Calcular la aproximación Kikuchi de P^j usando algoritmo 4.2.
9 $\lambda_j = \frac{\Gamma_j}{N}$
10 $t \leftarrow t+1$
11 } **until** Algún criterio de terminación ha sido satisfecho.

4.8.3. EDA basado en una mezcla de aproximaciones Kikuchi

El pseudo-código de un algoritmo evolutivo basado en mezclas de aproximaciones Kikuchi (Mixture of Kikuchi approximations EDA (MK-EDA)) se muestra en el algoritmo 4.5.

Algoritmo 4.5: MK-FDA

1 $t \Leftarrow 0$. Generar N puntos aleatoriamente.
2 **do** {
3 Evaluar los puntos en la función objetivo.
4 Seleccionar un conjunto S de $k \leq N$ puntos de acuerdo a un método de selección.
5 Aprender una mezcla Q de aproximaciones Kikuchi usando un algoritmo de aprendizaje.
6 Generar N nuevos puntos muestreando Q.
7 $t \Leftarrow t+1$
8 } **until** Algún criterio de terminación ha sido satisfecho.

4.8.4. Análisis de la complejidad del MK-EDA

Los pasos de iniciación, evaluación y selección del MK-EDA son comunes a los del MT-FDA y el MN-EDA, y la complejidad de estos pasos se corresponde con la calculada en la sección 2.6.1. La diferencia principal con el algoritmo MN-EDA está en que el MK-EDA aplica el algoritmo Kikuchi-IEM, este hecho implica el cálculo de m aproximaciones Kikuchi en cada paso del aprendizaje. Por otro lado la cantidad de pasos necesarios para llegar a una aproximación conveniente de la verosimilitud depende de los datos.

Siendo $O(Nn^3)$ la complejidad del cálculo de las pruebas de independencias (ver sección 3.5.1) y V el máximo n úmero de pasos del algoritmo Kikuchi-IEM. Si consideramos V y m como constantes, la complejidad en el caso del MK-EDA es $O(Nn^3)$, de forma más precisa $O(VNmn^3)$.

De forma similar, si consideramos que el n úmero máximo de regiones para cada descomposición en cliques es $N_R \approx rn^2k^2$ (ver sección 4.7.1), donde r es el tamaño del clique máximo en los grafos correspondientes a cualquiera de las componentes, el costo del algoritmo de muestreo puede estimarse como:

$$V \cdot N \cdot C_y \cdot \sum_{i=1}^{n} \sum_{R \in \mathcal{R}} (2^r - 1) \approx Nn2^r,$$

donde, al igual que se expone en la sección 4.7.1, el costo es exponencial en el tamaño de clique máximo, y lineal en N y n. Considerando C_y y r constantes, la complejidad total de MK-EDA es $O(mVGNn^3)$. La complejidad computacional del MK-EDA puede ser disminuida usando una única estructura compartida, y variando solamente los parámetros aprendidos.

4.9. Experimentos

Nuestros experimentos persiguen dos objetivos principales. Revelar la manera en que los diferentes componentes del MN-EDA determinan su comportamiento, y evaluar la conducta de los

algoritmos introducidos en esta tesis en la optimización de funciones teóricas y prácticas. Los experimentos se dividen en tres partes. Primero, se ilustra el papel desempeñado por las componentes de aprendizaje y muestreo del MN-EDA. Presentamos varios experimentos entonces donde el MN-EDA se compara con otros EDAs. Finalmente, presentamos resultados sobre la escalabilidad del algoritmo.

4.9.1. Estudio de la aproximación Kikuchi

Comenzamos con un experimento que contribuye a esclarecer las diferencias entre los modelos probabilísticos usados por el MN-FDA y el MN-EDA. El objetivo de nuestro experimento es comparar la calidad de la aproximación de las distribuciones del conjunto seleccionado dadas por una factorización ordenada, y por una aproximación Kikuchi. El problema de optimización es el problema de encontrar el mínimo del modelo de Ising generalizado en una rejilla de dimensiones 4x4, con conexión toroidal.

El problema del modelo Ising generalizado es quizás uno de los ejemplos más claros de la diferencia entre las aproximaciones tradicionales de la probabilidad y la aproximación Kikuchi. En [91] se afirma sin prueba que las factorizaciones exactas de funciones descomponibles aditivamente, definidas en rejillas de dimensión dos, requieren el cómputo de distribuciones marginales condicionales de tamaño $O(\sqrt{n})$, donde n es la cantidad de elementos en la rejilla. Si asumimos que esta proposición es cierta, la complejidad del modelo probabilístico será exponencial en n. En todo caso, el tamaño del clique máximo correspondiente a grafos triangulizados de rejillas es claramente mayor que dos.

El grafo de cliques ordenado y la aproximación Kikuchi usan los cliques del grafo original. En las rejillas las aristas son los cliques maximales, y tienen tamaño dos. Ambas aproximaciones son formas factibles y eficientes de almacenar una factorización de la distribución. Mientras los grafos de cliques ordenados usan sólo algunas aristas del grafo de independencia, la aproximación Kikuchi usa todas las aristas del grafo.

En nuestro experimento, se evalúa la calidad de la aproximación sólo para el primer conjunto seleccionado del EDA. Se genera una población inicial de 1000 individuos, se realiza la selección por truncamiento, y el grafo de independencia es aprendido a partir de los datos. El cuadro 4.1 muestra el conjunto de cliques (aristas) aprendidos del conjunto de soluciones seleccionado. Las 19 aristas aprendidas corresponden a interacciones que existen en la rejilla, la cual define parcialmente la estructura del problema. Por otro lado, 13 de las 32 interacciones representadas en la estructura de la rejilla no se han capturado en el grafo de independencia. Este hecho puede explicarse porque la existencia de una arista en la estructura no es la única condición para la existencia de dependencias, éstas también serán determinadas por la fuerza de las interacciones J_{ij}.

A partir del grafo de independencia no dirigido, los dos posibles modelos probabilísticos son construidos. Aunque el grafo de cliques ordenado puede representar ciclos, cuando el tamaño del clique máximo en el grafo de cliques ordenado es dos, el grafo de cliques no tiene ningún ciclo. Este hecho ocurre porque por lo menos una variable en cada clique de la factorización ordenada tiene que ser nueva en el ordenamiento. Cada clique en el ordenamiento sólo puede conectarse

E	P	W	elp_{JT}	p_{K1}	p_{K100}	E	P	W	elp_{JT}	p_{K1}	p_{K100}
$(6,10)$	1	19,36	0,028	0,039	0,056	$(14,2)$	11	8,55	0,035	0,066	0,059
$(6,5)$	2	11,71	0,019	0,091	0,034	$(14,15)$	12	18,47	0,034	0,076	0,017
$(9,10)$	3	11,22	0,057	0,091	0,056	$(3,2)$	13	8,34	0,055	0,029	0,050
$(8,9)$	4	17,28	0,019	0,059	0,073	$(7,11)$	14	12,77	0,085	0,027	0,049
$(8,12)$	5	15,86	0,041	0,009	0,018	$(1,5)$	–	9,65	0,339	0,053	0,073
$(9,13)$	6	10,12	0,068	0,091	0,033	$(3,0)$	–	7,94	0,282	0,084	0,037
$(1,13)$	7	15,97	0,029	0,030	0,053	$(12,13)$	–	7,32	0,270	0,030	0,051
$(1,2)$	8	13,63	0,035	0,025	0,053	$(6,2)$	–	6,72	0,283	0,078	0,093
$(1,0)$	9	10,55	0,039	0,084	0,053	$(4,8)$	–	6,31	0,315	0,062	0,017
$(4,5)$	10	9,64	0,013	0,054	0,073	$Tot.$			2,05	1,09	0,95

Cuadro 4.1: Errores en la aproximación de los marginales del conjunto seleccionado dados por las aproximaciones válida y Kikuchi.

a exactamente un clique anterior. La factorización ordenada del grafo aprendido está compuesta por los cliques enumerados en las columnas dos y ocho. Estas aristas forman un árbol, más exactamente, un bosque. En este caso, la factorización es válida. Para la aproximación Kikuchi todas las aristas mostradas en las columnas E se usan en el modelo.

Nuevas poblaciones son generadas para cada modelo. En el caso de la aproximación Kikuchi, determinamos que el GS comience a partir de un punto muestreado de la factorización ordenada ($In = 1$). Para medir la calidad de la aproximación, los marginales en las dos nuevas poblaciones son calculados, y se computan los errores en las aproximaciones con respecto a los marginales calculados a partir del conjunto seleccionado. Sean respectivamente P^s_{ij} y P^{new}_{ij} los marginales bivariados de X_i, X_j calculados a partir del conjunto seleccionado, y de la nueva población. El error en la aproximación se calcula como $\sum_{X_i=x_i, X_j=x_j}(P^{new}_{ij} - P^s_{ij})$.

En el cuadro 4.1, p_{JT} representa el error dado por la aproximación válida para los diferentes marginales. Los errores son pequeños para los marginales que corresponden a las aristas numeradas, pero notablemente peores para las aristas restantes. Este hecho ocurre porque el árbol de cliques no puede cubrir estas interacciones y usarlas en el muestreo. Después de sólo un ciclo del GS, la aproximación Kikuchi consigue resultados (P_{K1}) mejores que la aproximación del grafo de cliques ordenado. Para la aproximación Kikuchi no hay diferencias importantes entre los errores de la aproximación para los diferentes cliques. En el cuadro, P_{K100} muestra el error cuando el número de pasos de muestreo es aumentado a 100. Puede observarse que el error global disminuye. No obstante, el error en la aproximación de algunas aristas puede aumentar, probablemente debido a los errores de muestreo del GS.

Resumiendo, los resultados de los experimentos ilustran que cubriendo un mayor número de dependencias que el árbol de cliques, la aproximación Kikuchi puede ser más precisa.

Influencia del GS en los resultados de convergencia

El objetivo del experimento siguiente es determinar si la manera en que el GS es iniciado, así como el número de ciclos que se realizan, tienen un efecto significativo en el desempeño del MN-EDA. El experimento consiste en ejecutar el MN-EDA para las funciones $F_1 = f_{3deceptive}$, $F_2^3 = f_{deceptive3}$, $F_2^4 = f_{deceptive4}$, $F_3 = F_{IsoPeak}$, $F_4 = f_{Isotorus}$ presentadas en la sección A.2.

El objetivo es evaluar el efecto que tiene la variación de los parámetros investigados en la razón de convergencia, y el número de generaciones del MN-EDA. Los resultados de nuestro experimento se muestran en el cuadro 4.2. En todos los experimentos $n = 36$, y para el GS usamos el esquema de visita aleatorio. La parte superior del cuadro muestra los resultados del MN-EDA que usa GS con iniciación aleatoria. La parte inferior muestra el MN-EDA que usa un GS iniciado con un punto muestreado a partir de una factorización ordenada. Esta factorización se ha construido a partir de un del grafo de independencia.

Init	*Ciclos*	F_1		F_2^3		F_2^4		F_3		F_4	
		S	$\bar{g}$	S	$\bar{g}$	S	$\bar{g}$	S	$\bar{g}$	S	$\bar{g}$
Aleatorio	2	1	13,00	4	15,00	4	10,00	51	12,43	97	8,42
	4	56	14,80	67	13,39	61	13,98	100	5,98	98	6,03
	6	80	13,76	87	11,77	98	10,70	99	5,73	96	6,04
	8	96	12,95	95	10,79	100	9,38	98	5,85	90	5,04
	12	98	11,97	100	6,91	99	8,43	94	5,66	88	4,68
	16	95	11,44	99	5,69	97	8,14	87	6,19	95	5,14
	20	91	10,78	100	5,87	95	8,10	77	6,35	90	4,25
Fact.	0	78	7,78	100	4,70	92	5,58	60	5,03	80	5,90
	1	94	8,69	100	4,77	99	5,64	70	4,78	90	4,86
	2	89	8,56	100	4,78	100	5,53	86	5,37	95	4,43
	3	92	9,44	100	4,87	100	5,67	89	6,56	98	4,59
	4	87	8,67	100	4,82	100	5,96	95	7,32	95	4,33
	5	87	9,04	100	4,88	99	5,99	97	6,55	96	4,59

Cuadro 4.2: Razón de éxito y promedio del número de generaciones del MN-EDA para diferentes iniciaciones del GS, y diferente número de ciclos.

Para el MN-EDA que usa GS con iniciación aleatoria la razón de éxito aumenta con el incremento del número de ciclos. Después de dos ciclos del GS, cada variable se ha actualizado como promedio un par de veces. En este caso, si la configuración inicial de los vectores ha sido escogida al azar, la distribución del los puntos muestreados estará lejana de la aproximación Kikuchi deseada. Después de más ciclos del GS la distribución recuperada de los puntos muestreados es una aproximación Kikuchi más exacta. Este hecho contribuye a una búsqueda más eficaz que se expresa en una razón de éxito mayor. Cuando el número de ciclos aumenta, también el promedio de generaciones necesarias para encontrar el óptimo disminuye, es decir el algoritmo se vuelve más eficiente. Sin embargo, debe tenerse cuidado, porque demasiados ciclos de muestreo producen lo que se denomina problema de sobreajuste, ya mencionado en la sección 2.7.

Existen diferencias en la razón de éxito para las funciones. Las funciones F_3 y F_4 necesitan menos ciclos que las otras funciones para lograr una razón de éxito por encima de 90. Para la función F_3, los resultados se deterioran cuando el número de ciclos aumenta. F_3 es una función que puede optimizarse fácilmente con un FDA que use un modelo gráfico simple con forma de cadena. La aproximación Kikuchi tiene una mayor complejidad, si se utiliza con muchos ciclos del GS, se provoca un problema de sobreajuste de los datos.

En principio, usar el MN-EDA con iniciación aleatoria puede no ser una alternativa práctica cuando se necesitan muchos ciclos, debido a que el muestreo es un paso costoso del algoritmo. La alternativa a la iniciación aleatoria es comenzar la simulación del GS a partir de un vector muestreado de una factorización ordenada. Las factorizaciones ordenadas pueden estar basadas en grafos de cliques ordenados. En este caso, si ningún paso del GS se aplica después de iniciar el vector, el MN-EDA se comportará como el MN-FDA. En todos los experimentos siguientes el MN-EDA usa sólo un ciclo del GS. El uso de un sólo ciclo del GS a partir de una solución muestreada de un grafo de cliques ordenado busca evitar el problema de sobreajuste, persiguiendo un equilibrio entre las capacidades de exploración y explotación del modelo. Por otro lado, la iniciación aleatoria puede permitir una mayor diversidad en la población.

Resumiendo: El GS juega un papel importante en el MN-EDA. Este método de muestreo garantiza que los nuevos puntos tendrán una distribución similar a la distribución del conjunto seleccionado. El número de ciclos, y el punto inicial de la simulación son ambos elementos críticos del algoritmo.

4.9.2. Comparaciones con otros EDAs

Comparamos la conducta del MN-EDA con los siguientes FDAs: EBNA$_B$, EBNA$_{local}$, EBNA$_{K2}$, BOA, y el LFDA. Todos son FDAs Bayesianos que utilizan métricas diferentes para construir la estructura de la red Bayesiana. Todos se incluyen dentro de los FDAs más referenciados en la literatura. En los casos de los algoritmos EBNA y BOA, destacamos el hecho de que los métodos de selección y/o las estrategias de remplazamiento normalmente usadas por estos algoritmos no son las que hemos empleado en nuestros experimentos. Hemos adaptado el código de programación de estos algoritmos para trabajar con selección por truncamiento con $T = 0{,}15$. Esta modificación ayudó a concentrarnos en el impacto de las diferencias entre sus correspondientes modelos probabilísticos en el desempeño de los algoritmos y desatender la influencia de otros factores. También incluimos en nuestro diseño de experimentos resultados obtenidos con el UMDA. y el MT-FDA.

Resultados para las funciones decepcionantes

En el cuadro 4.2 observamos que en muchos casos los resultados de MN-EDA mejoran cuando se aumenta el número de ciclos. Ahora analizamos la conducta del MN-EDA con parámetros fijos comparándolo con otros EDAs. En la figura 4.3 mostramos la razón de éxito de MN-EDA, MN-FDA, EBNA$_B$, EBNA$_{local}$, UMDA y LFDA en la optimización de cuatro de las cinco funciones mostradas en el cuadro 4.2. Los resultados para EBNA$_{K2}$ y BOA no han sido incluidos

en la figura pues no resultaron mejores que los obtenidos con otros EDAs. La función F_2^3 no ha sido incluida porque es muy fácil de optimizar por todos los algoritmos. Las razones de éxito correspondientes a MN-EDA y MN-FDA se han tomado del cuadro 4.2.

En la figura puede apreciarse que el UMDA es capaz de optimizar únicamente la función F_4, para el resto de las funciones su razón de éxito es cero. Este hecho nos da una idea de la importancia de capturar las interacciones relevantes para la optimización de estas funciones. El MN-EDA sobrepasa claramente, seguido por MN-FDA, a los FDAs Bayesianos en las funciones F_1 y F_2. Ocupa el tercer puesto para la función F_3, y es segundo para la función F_4. Éste es un comportamiento bueno, particularmente si tenemos en cuenta que los resultados de MN-EDA podrían mejorarse para las funciones F_3 y F_4 aumentando el número de ciclos del GS, como se ha mostrado en el cuadro 4.2. Si excluimos al UMDA, el desempeño del MN-FDA es el peor para las funciones F_3 y F_4.

Figura 4.3: Razones de éxito de diferentes EDAs en la optimización de las funciones decepcionantes.

Experimentos problema SAT, instancias uf20-91, clase I

Utilizamos una vez más las instancias difíciles del problema SAT, en esta ocasión para estudiar el comportamiento del MK-EDA con diferente número de componentes. Deseamos evaluar la influencia de utilizar más de una componente y si el algoritmo de aprendizaje es capaz de encontrar mezclas que aproximen mejor la distribución. Con este objetivo el GS comenzará en un punto generado aleatoriamente. El cuadro 4.3 muestra los resultados del MK-EDA con la cantidad de componentes $j \in \{1,\ldots,5\}$ y el número de pasos del GS entre uno y cuatro. Cuando utiliza una sola componente el MK-EDA es equivalente al MN-EDA. Nótese la mejora en los

resultados cuando se pasa de una componente a dos. La razón de éxito llega a mejorar en más de un 15 porciento. Cuando el número de componentes se incrementa los resultados se deterioran. Pero resulta incluso preferible usar una mezcla de cinco componentes a una mezcla de una sola. Para comparar con los resultados obtenidos por otros EDAs es posible consultar el cuadro 2.2 y la figura 2.3.

Pasos/Num. Comp.	1	2	3	4	5
1	72,78	84,11	78,63	75,84	72,96
2	73,98	89,04	84,90	81,29	78,60
3	72,25	89,28	84,82	81,05	79,10
4	74,11	88,70	84,20	79,92	77,88

Cuadro 4.3: Resultados del MK-EDA en las instancias uf20-91 del problema SAT, clase I.

Experimentos problema SAT, instancias aim

Los experimentos realizados en esta tesis se han concentrado en comparaciones entre EDAs. Este hecho está motivado por el propósito de investigar la influencia de la modelación probabilística en la eficiencia de la búsqueda. Sin embargo, en los siguientes experimentos comparamos los resultados obtenidos con varios EDAs con los resultados publicados para un GA híbrido introducido en [27]. El GA usa un optimizador local muy eficiente basado en una red neuronal de Hopfield. Este algoritmo híbrido llamado GRN se utiliza para la solución del problema SAT.

El experimento persigue los siguientes objetivos:

1. Comparar el desempeño de los EDAs introducidos en la tesis para la solución del problema SAT en un grupo de instancias del problema reconocidas como difíciles.
2. Mostrar una de las formas en que los EDAs pueden ser combinados con optimizadores locales.
3. Comparar el desempeño de los EDAs con el del GA híbrido GRN [27].

Las instancias del problema SAT usadas en este experimento fueron las pertenecientes a los bancos de instancias aim-50-3-4-yes1-j y aim-100-3-4-yes1-j, las cuales son presentadas en la sección A.3.1. Utilizamos también un optimizador local simple (algoritmo 4.6) que es incorporado a los EDAs. Mientras el optimizador basado en redes de Hopfield es un método de optimización avanzado, nuestro algoritmo es menos sofisticado. Sin embargo, este método de optimización local acelera la convergencia del algoritmo. Aplicamos el optimizador local a cada solución durante su evaluación. Existen otras formas de combinar optimizadores locales con EDAs. Nuestra opción es quizás la más simple entre las posibles, pero sirve al propósito de ilustrar de qué forma pueden obtenerse EDAs más eficientes.

Optimizador local para SAT

En nuestras simulaciones utilizamos un optimizador local que acelera la convergencia al óptimo para los EDAs analizados. El seudo-código del optimizador local se muestra en el algoritmo 4.6. El algoritmo trata de realizar un n úmero pequeño de evaluaciones de la función. Para cada cláusula que no es satisfecha, el algoritmo escoge una variable aleatoria, modifica su valor y determina si la nueva solución es mejor o igual que la anterior, en tales casos se acepta el cambio en el valor de la variable.

Algoritmo 4.6: Optimizador local para SAT

1 Almacenar en la lista L los índices correspondientes a las n_{cl}. cláusulas que no son satisfechas por la solución actual x.
2 } **while** L no esté vacía
3 Extraer un miembro i de L escogido aleatoriamente.
4 Seleccionar una variable X_j de la cláusula C_i escogida aleatoriamente.
5 $y = (x_1, \cdots, x_{j-1}, 1 - x_j, x_{j+1}, \cdots, x_n)$
6 si $f(y) \geq f(x)$ entonces $y = x$.

Los experimentos fueron realizados para un tamaño de población $N = 1000$ y 50 generaciones. Se utilizan los parámetros usuales empleados anteriormente para los EDAs. El cuadro 4.4 presenta los resultados para el GA híbrido (GRN), UMDA, Tree-FDA, MT-FDA con dos y cuatro componentes. También incluimos un EDA que no utiliza optimizador local. Éste es el caso del algoritmo mostrado en la última columna del cuadro 4.4. Todas las instancias pueden ser satisfechas, m es el n úmero de cláusulas en cada instancia (óptimo a ser alcanzado). mb es el mejor valor alcanzado por el correspondiente algoritmo, y v es la cantidad de veces que mb fue alcanzado en diez corridas. Los resultados alcanzados por el GRN fueron tomados de [27]. Las primeras filas corresponden a las cuatro instancias de aim-50-3-4-yes1-j, las siguientes cuatro a las instancias de aim-100-3-4-yes1-j (ver sección A.3.1). En la última fila se muestra, como resumen de los resultados, la cantidad de veces que cada algoritmo convergió al óptimo de la función en los 80 experimentos realizados.

Analizando las diferencias entre los EDAs, puede apreciarse en el cuadro 4.4 que los resultados mejoran cuando la complejidad de los modelos se incrementa. La diferencia es clara si comparamos los resultados alcanzados por el UMDA con los obtenidos por el MT-FDA^4 que utiliza el optimizador local. La importancia de introducir el optimizador puede ser constatada observando los pobres valores alcanzados por el EDA que no incorpora el optimizador local. En la comparación entre GRN y MT-FDA^4, el EDA es el ganador para $n = 50$, pero para $n = 100$ ambos algoritmos están parejos, fallan al tratar de encontrar los óptimos globales.

El cuadro 4.5 muestra los resultados obtenidos con los algoritmos FDA, MN-FDA, MN-EDA y MK-EDA. Se emplean los mismos parámetros utilizados en el experimento mostrado en el cuadro 4.4. La única excepción es el EDA mostrado en la última columna. Para éste algoritmo $N = 5000$ y la cantidad de generaciones fue 200. El objetivo de incluir el MN-EDA con éstos parámetros era evaluar la mejoría en los resultados de convergencia al aumentar el tamaño de

		GRN		UMDA		Tree-FDA		MT-FDA^2		MT-FDA^4		MT-FDA^4	
n	m	mb	v	mb	v	mb	v	mb	v	mb	v	mb	v
50	170	170	4	168	10	170	1	170	3	170	5	169	4
50	170	170	4	170	1	170	3	170	4	170	8	170	1
50	170	170	2	170	2	170	6	170	10	170	9	170	2
50	170	170	2	170	8	170	10	170	10	170	10	170	2
100	340	340	1	336	3	337	3	337	1	337	2	336	2
100	340	337	1	337	1	337	2	337	2	337	3	337	1
100	340	339	4	336	6	336	7	336	6	336	10	336	3
100	340	336	4	339	1	340	1	340	2	340	2	336	1
Tot.			13		11		21		29		34		5

Cuadro 4.4: Comparación entre el GRN y diferentes EDAs para las instancias aim del problema SAT.

población. Ésta mejoría es evidente a partir de los resultados mostrados en el cuadro. Otro punto a señalar, relacionado con el tema tratado en el capítulo anterior, es la mejoría de los resultados cuando se utiliza un grafo de cliques ordenado (MN-FDA) en comparación al uso de un árbol de cliques.

		FDA		MN-FDA		MN-EDA		MN-EDA		MK-EDA$_2$		MK-EDA$_4$	
Cy		-		-		1		3		1		1	
n	m	mb	v	mb	v	mb	v	mb	v	mb	v	mb	v
50	170	169	10	170	2	170	1	170	8	170	2	170	10
50	170	170	6	170	7	170	9	170	8	170	9	170	10
50	170	170	7	170	8	170	9	170	10	170	10	170	10
50	170	170	9	170	9	170	10	170	8	170	10	170	10
100	340	337	1	336	6	337	2	336	7	337	1	336	4
100	340	337	1	337	2	337	4	337	1	337	3	340	1
100	340	336	7	336	7	336	9	336	9	336	6	336	10
100	340	340	2	340	3	340	2	339	8	340	2	340	6
Tot.			24		29		31		34		31		47

Cuadro 4.5: Comparación entre MN-FDA, MN-EDA, y el MK-EDA para las instancias aim del problema SAT.

Problema de encontrar la configuración de mínima energía para el modelo de Ising generalizado

En los experimentos siguientes evaluamos el desempeño del MN-EDA para el problema de encontrar un mínimo de la energía del modelo de Ising generalizado. Estudiamos asimismo la

influencia de incorporar al algoritmo que aprende la aproximación Kikuchi información disponible sobre las interacciones entre las variables.

I	MN-EDA		MN-EDA^f		MN-FDA		EBNA$_{\text{K2}}$		MT-FDA	
	S	$\bar{e}$	S	$\bar{e}$	S	$\bar{e}$	S	$\bar{e}$	S	$\bar{e}$
I_1^{16}	100	362,4	100	127,4	100	219,5	96	149,5	100	176,9
I_2^{16}	100	910,0	98	362,7	100	885,0	90	924,6	97	653,2
I_3^{16}	90	587,3	96	206,2	93	774,8	91	525,0	98	380,5
I_4^{16}	100	378,2	100	130,4	100	246,5	93	139,0	100	176,9
I_1^{36}	93	4496,7	97	1970,7	90	4939,4	92	6089,0	91	4173,9
I_2^{36}	90	3967,7	97	1733,4	90	5230,0	94	7142,8	91	3658,5
I_3^{36}	97	2799,5	93	830,3	92	2924,1	91	2881,7	93	2588,1
I_4^{36}	91	3959,6	94	1737,7	90	6074,2	91	7202,6	93	4490,3
I_1^{64}	98	8956,8	100	3838,3	94	11376,8	93	13793,7	91	9908,1
I_2^{64}	91	11251,7	95	11649,5	91	21071,0	93	25702,3	90	19235,8
I_3^{64}	91	10012,8	95	4607,6	91	16188,9	94	18834,6	90	12574,3
I_4^{64}	94	10701,9	90	4347,8	90	15095,1	99	30687,7	94	16213,8

Cuadro 4.6: Razón de éxito y número de evaluaciones promedio de varios EDAs para diferentes instancias del problema Ising generalizado.

Hemos generado cuatro instancias para diferentes valores de $n \in \{16, 36, 64\}$. La generación de las instancias se describe en la sección A.3.2.

Calculamos el tamaño de población necesario para que cinco EDAs diferentes puedan lograr una razón de éxito igual a 90 en 100 experimentos. Comparamos los resultados obtenidos por MN-EDA, MN-FDA, MN-EDA^f, MT-FDA con dos árboles, y EBNA$_{K2}$. No se realizaron experimentos para LFDA y BOA. EBNA$_{K2}$ logró los mejores resultados entre todos los algoritmos EBNA, y por consiguiente los resultados para los otros algoritmos EBNA no fueron incluidos. Para el problema Ising, el MN-EDA^f aprende la descomposición en cliques a partir de la rejilla. La descomposición en cliques incluirá todas las aristas del grafo y sus intersecciones. La aproximación Kikuchi calculada a partir de esta descomposición del grafo es equivalente a la aproximación Bethe usualmente utilizada en Física Estadística [83, 170, 169].

El cuadro 4.6 presenta la razón de éxito de cada algoritmo y el promedio de evaluaciones de la función para cada una de las 12 instancias. Los resultados se resumen en el cuadro 4.7, donde hemos contado el número de veces en que cada algoritmo ha ocupado cada una de las cinco posibles posiciones. El ordenamiento de los algoritmos se realiza a partir del promedio de las evaluaciones necesarias para alcanzar el óptimo. El algoritmo MN-EDA^f es claramente el mejor. Se ubica como el primero en todos los casos exceptuando uno, donde ocupa el segundo lugar. Este hecho ilustra la capacidad de la aproximación Kikuchi para incorporar información sobre la estructura del problema, aun cuando las interacciones entre las variables forman muchos ciclos, como es el caso del modelo Ising. Los algoritmos Bayesianos pueden incorporar información sobre la estructura del problema [104], pero no en forma de ciclos.

El comportamiento del resto de los EDAs se relaciona al tamaño de los problemas. Para $n = 16$, MT-FDA y EBNA$_{K2}$ son los mejores algoritmos. Para $n = 36$, MT-FDA es mejor que el MN-EDA. Para $n = 64$, MN-EDA supera al resto de los algoritmos. Los resultados relativamente buenos logrados por el MT-FDA pueden ser explicados por la presencia de simetría en el problema Ising [94, 105]. Realizando un agrupamiento de las soluciones, y aprendiendo un modelo especializado en cada grupo, una mezcla de árboles puede tratar algunos tipos de problemas simétricos. Pero si se aumenta el número de variables, se necesitarían más componentes en la mezcla para mantener la misma razón de éxito. Para tratar de una manera satisfactoria con los problemas Ising, los EDAs deben mantener un alto grado de diversidad en la población. La competencia del MN-EDA para tratar con estos problemas puede ser mejorada con la incorporación de técnicas usadas con este propósito en los GAs, por ejemplo, los métodos de estratificación (niching methods) [74].

	I^{16}					I^{36}					I^{64}					Tot.				
Alg. \ Pos.	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
MN-EDA^f	4	0	0	0	0	4	0	0	0	0	3	1	0	0	0	11	1	0	0	0
MN-EDA	0	0	0	2	2	0	1	3	0	0	1	3	0	0	0	1	4	3	2	2
MN-FDA	0	0	1	2	1	0	0	0	3	1	0	0	1	3	0	0	0	2	8	2
MT-FDA	0	2	2	0	0	0	3	1	0	0	0	0	3	1	0	0	5	6	1	0
EBNA$_{K2}$	0	2	1	0	1	0	0	0	1	3	0	0	0	0	4	0	2	1	1	8

Cuadro 4.7: Resumen de los resultados de los EDAs en la optimización de varias instancias del problema Ising generalizado.

Funciones con interdependencia entre las subsoluciones y frustración

En el experimento siguiente comparamos el desempeño del MN-EDA y dos FDAs Bayesianos en la optimización de dos funciones difíciles: las funciones HIFF (A.10) y *Cuban*5 (A.9).

Las funciones jerárquicamente descomponibles, como la HIFF, sirven para modelar problemas con interdependencia entre sus subsoluciones [165]. Para optimizar estas funciones, un EDA tiene que poder combinar de manera correcta subsoluciones que corresponden a soluciones óptimas diferentes. La función *Cuban*5 fue introducida y estudiada en [91], donde mostró ser una función aditiva no-separable muy difícil.

Primeramente calculamos un límite superior del tamaño de la población que necesita el MN-EDA para optimizar estas funciones en 30 experimentos consecutivos. En el caso de la función *Cuban*5, $n = 101$, y debido al gran tamaño de la población necesaria para optimizar esta función, sólo diez experimentos exitosos se exigieron para determinar el tamaño de población crítico. Una vez que se han encontrado los tamaños de población, ejecutamos BOA y LFDA con ese tamaño de población, calculando para estos algoritmos su razón de éxito, y el número promedio de evaluaciones. En el cuadro 4.8 se muestran los resultados. En sólo dos de los seis experimentos LFDA y BOA pudieron alcanzar la misma razón de éxito que el MN-EDA, aunque en la mayoría de los casos el número promedio de evaluaciones del MN-EDA es mayor que el correspondiente

al LFDA. Para la función HIFF, $n = 32$, MN-EDA es sobrepasado por ambos EDAs Bayesianos. En el caso de función *Cuban*5, $n = 101$, MN-EDA es el segundo algoritmo detrás del LFDA. Los resultados revelan que también para funciones difíciles, con muchas variables, el MN-EDA es una alternativa a los FDAs Bayesianos.

EDA	*HIFF*					*Cuban*5				
	n	N	S	$\bar{g}$	$\bar{e}$	n	N	S	$\bar{g}$	$\bar{e}$
MN-EDA	32	700	30	4,90	3426,1	37	1850	30	6,40	11834,6
	64	1300	30	9,43	12254,9	69	9600	30	9,23	88631,8
	128	4300	30	16,30	69931,4	101	23600	10	11,90	280829,0
LFDA	32	700	30	4,86	3402,8	37	1850	27	5,85	10830,9
	64	1300	26	8,5	11735,3	69	9600	27	8,70	83547,9
	128	4300	29	12,27	52774,9	101	23600	10	10,90	257230,0
BOA	32	700	30	4,1	2865,9	37	1850	20	6,85	12678,8
	64	1300	26	8,5	11041,9	69	9600	23	9,65	92651,2
	128	4300	20	15,0	64485,0	101	23600	10	12,4	292627,0

Cuadro 4.8: Resultados de los algoritmos MN-EDA, LFDA, y BOA en la optimización de las funciones HIFF y Cuban5.

4.9.3. Experimentos de escalabilidad de los algoritmos

En los experimentos siguientes investigamos la escalabilidad del MN-EDA. Para todas las funciones analizadas, se determina el tamaño de población crítico para el cual el algoritmo converge al óptimo en 90 de 100 ejecuciones. La descripción de este tipo de experimentos se presenta en la sección A.4.2. Los parámetros que describen la manera en que se incrementaron los tamaños de población fueron $Ipsize = 50$, $Apsize = 50$ y $Upsize = 100$. Primero comparamos la escalabilidad del MN-EDA y otros EDAs para la función $f_{3deceptive}$. Usamos los algoritmos siguientes: EBNA$_B$, EBNA$_{local}$, y EBNA$_{K2}$. Adicionalmente investigamos la escalabilidad del MN-FDA. Establecemos un tamaño de población máximo igual a 6000. Si un algoritmo alcanza este tamaño de población sin converger detenemos la simulación. La figuras 4.4 a) y b) muestran respectivamente la escalabilidad del tamaño de la población, y el número promedio de evaluaciones de los algoritmos para la función $f_{3deceptive}$.

Puede observarse que el peor desempeño corresponde al MN-FDA. Este algoritmo no puede optimizar la función $f_{3deceptive}$ para $n = 82$ con menos de 6000 individuos. En el otro extremo está MN-EDA. Para optimizar esta función para $n = 120$, el algoritmo necesita un tamaño de la población $N = 5650$, y un promedio de evaluaciones $\bar{e} = 74151{,}6$. Todos los EDAs Bayesianos tienen una escalabilidad muy similar. Estos algoritmos se comportan mejor que el MN-FDA, y peor que el MN-EDA.

Investigamos ahora el efecto de aumentar el orden de la función aditiva en la escalabilidad del MN-EDA. Usamos la función $f_{deceptivek}$, y los mismos parámetros que en el experimento anterior. La figuras 4.5 a) y b) muestran respectivamente el tamaño de población, y el número

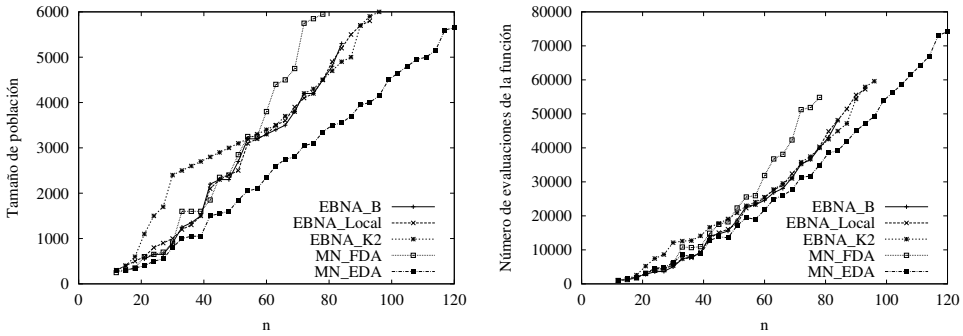

Figura 4.4: Escalabilidad de MN-EDA, MN-FDA y FDAs Bayesianos para la función $f_{3deceptive}$.

de evaluaciones para $k \in \{3,4,5\}$. Para evaluar el incremento en la complejidad del algoritmo, calculamos los coeficientes de los polinomios lineales $\beta log(n) + \gamma$ que mejor aproximan las curvas $(log(n), log(\bar{e}))$. La aproximaciones encontradas fueron las siguientes:

$$log(\bar{e}_{fdeceptive3}) = 1{,}921 log(n) + 0{,}875$$
$$log(\bar{e}_{fdeceptive4}) = 2{,}060 log(n) + 0{,}916$$
$$log(\bar{e}_{fdeceptive5}) = 2{,}440 log(n) + 0{,}433$$

El cambio en los valores de β indica el orden del incremento en el número de evaluaciones de la función cuando el orden de la función aumenta. Como era de esperar, el tamaño de la población y el número de evaluaciones crecen exponencialmente con el orden de la función.

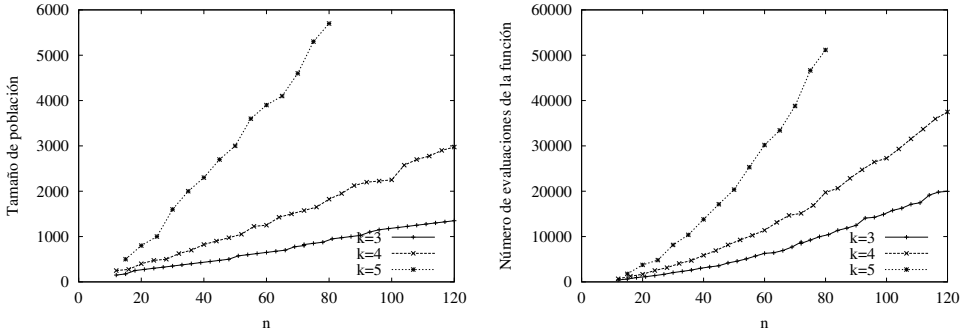

Figura 4.5: Escalabilidad del MN-EDA para la función $f_{deceptivek}$, $k \in \{3,4,5\}$.

4.10. Trabajo relacionado y futuro

Hemos restringido nuestro análisis de las factorizaciones al caso de los modelos no dirigidos. Sin embargo, en el espacio de modelos dirigidos existe una clase de modelos capaces de representar ciclos en las interacciones entre variables. Las Redes de Dependencia Consistente (Consistent Dependency Networks [58]) y los MRFs son representaciones intercambiables. Estas redes dirigidas no han sido investigadas en el contexto de los EDAs. Recientemente, Mühlenbein y Mahnig [90] han señalado la conveniencia de usar otros procedimientos para la estimación de distribuciones de Boltzmann. El Ajuste Proporcional Iterativo (Iterative Proportional Fitting) [65], y los métodos de Campo Central Avanzado (Advanced Mean Fields) [100], se encuentran entre los métodos propuestos para la estimación de distribuciones de probabilidad en EDAs [90].

El uso de métodos de Monte Carlo en EDAs fue inicialmente tratado en [92] donde los autores proponían utilizar el método de Metrópolis en la estimación de distribuciones. En [97] se proponía el uso del GS para el muestreo del FDA*. En ninguno de los dos casos la aplicación de los métodos resultó en algoritmos EDAs factibles.

Nuestro trabajo también se relaciona con recientes investigaciones en algoritmos aproximados para la propagación de creencias [170, 169, 163]. Mientras la mayor parte del trabajo en propagación de creencias se preocupa por el cálculo de las probabilidades marginales en presencia de evidencias, en nuestro caso nos concentramos en el muestreo de distribuciones aproximadas. Los resultados del estudio de los algoritmos de aproximación de creencias pueden ser útiles para entender para qué tipo de distribuciones de probabilidad pueden encontrarse buenas estimaciones basadas en la aproximación Kikuchi. De nuestro estudio se derivan un grupo de problemas teóricos como son:

1. El estudio de otras propiedades Markovianas de la aproximación Kikuchi.
2. Concepción de algoritmos de construcción de aproximaciones Kikuchi basados en métricas.
3. Investigación de otros métodos de descomposición de grafos y sus implicaciones en las propiedades de la aproximación Kikuchi.
4. Relacionado con el estudio de las propiedades Markovianas de la aproximación Kikuchi, investigar el uso del Gibbs Sampling en bloques (Blocked Gibbs Sampling (BGS)) [151, 20, 40] para acelerar la velocidad de mezcla (mixing rate) de la cadena de Markov.

Asimismo identificamos un grupo de mejoras que pueden ser realizadas al MN-EDA y algunas posibles aplicaciones de los temas tratados en el capítulo.

1. Resultados preliminares muestran que el uso de métodos de estratificación, como el remplazamiento restringido por torneo (restricted tournament replacement [104]) , puede acelerar la convergencia del MN-EDA.

2. En muchos casos, durante la evolución de MN-EDA, el grafo de independencia encontrado es cordal. En estos casos la descomposición en cliques corresponderá a un árbol de cliques, y el GS no será necesario, el algoritmo PLS puede utilizarse para el muestreo. Un MN-EDA capaz de cambiar el tipo de método de muestreo usado, de acuerdo a las características del grafo, será ciertamente más eficiente.

3. Otros tipos de métodos de selección deben investigarse. Por ejemplo, la selección Boltzmann con esquema adaptable de la temperatura [76] es un método que combina la conveniencia práctica del truncamiento con la conveniencia de la distribución Boltzmann para el análisis teórico.

4.11. Conclusiones

La contribución principal de este capítulo está en presentar el MN-EDA y el MK-EDA. Estos algoritmos usan la aproximación Kikuchi para estimar la probabilidad, y el método GS para muestrearla. Comparados con EDAs anteriores, éstos son acercamientos completamente diferentes para aproximar y muestrear las distribuciones de probabilidad. Además, el uso del algoritmo EM para aprender la mezcla de aproximaciones Kikuchi es otro resultado novedoso introducido en esta tesis. En nuestro conocimiento, no existen otras aplicaciones anteriores del EM en la estimación de mezcla para modelos donde cada componente no representa una distribución. En la mezcla de aproximaciones Kikuchi las componentes no son necesariamente modelos gráficos que representan distribuciones de probabilidad.

Hemos proporcionado una evaluación empírica sistemática de los algorimos introducidos en la tesis, comparándolos con varios de los EDAs más referenciados, y para diferentes funciones reconocidas como difíciles. El comportamiento desplegado por el MN-EDA muestra que en el contexto de los EDAs, los modelos probabilísticos basados en grafos no dirigidos, y aprendidos realizando pruebas de independencia, pueden ser una alternativa a las redes Bayesianas aprendidas utilizando algoritmos de búsqueda que minimizan una métrica. En el caso del MN-FDA sus resultados no están a la altura de los alcanzados con el MN-EDA. Además de constituir un paso intermedio entre el FDA y el MN-EDA, el MN-FDA queda como un caso particular del MN-EDA, que resulta un algoritmo mucho más general

Adicionalmente, en este capítulo hemos derivado varias propiedades de la aproximación Kikuchi. El alcance de las aplicaciones de estas propiedades está más allá del campo de los EDAs. A continuación se listan los resultados alcanzados en el capítulo.

1. Introducción de una nueva clasificación de los EDAs, basada en el tipo de factorización usada por cada algoritmo.

2. Introducción de la aproximación Kikuchi a partir de una descomposición del grafo en cliques.

3. Demostración de la propiedad Markoviana local de la aproximación Kikuchi.

4. Introducción de la mezcla de aproximaciones Kikuchi.
5. Introducción de algoritmos de aprendizaje de la aproximación Kikuchi y mezcla de aproximaciones Kikuchi.
6. Introducción por primera vez en EDAs del muestreo de Gibbs para el muestreo de las distribuciones.
7. Introducción por primera vez de un EDA capaz de cambiar entre clases diferentes de modelos probabilísticos de acuerdo a la complejidad de los datos.
8. Propuesta del algoritmo MN-EDA, basado en el uso de aproximaciones Kikuchi.
9. Propuesta del algoritmo MK-EDA, basado en el uso de mezcla de aproximaciones Kikuchi.
10. Diseño de las estructuras de datos, programación en lenguaje C++ y puesta a punto de los programas que implementan los algoritmos introducidos en el capítulo.

Conclusiones y Recomendaciones

En este capítulo presentamos las conclusiones más generales de la tesis. Se presenta un resumen de los resultados más importantes alcanzados y se incluye un número de recomendaciones para el trabajo futuro. Conclusiones particulares sobre cada uno de los capítulos aparecen reflejadas en las secciones 2.10, 3.8 y 4.11.

Conclusiones

La investigación desarrollada en esta tesis ha permitido corroborar la importancia que significa considerar, en determinados problemas que no pueden ser resueltos con modelos probabilísticos simples, dependencias probabilísticas de orden superior. Si bien una gran parte de los problemas analizados en la tesis, particularmente los problemas SAT, mostraron ser optimizables con EDAs que emplean estructuras simplemente conectadas, nuestra investigación arrojó que para un conjunto de problemas imposibles de solucionar con los EDAs mencionados anteriormente, EDAs con modelos probabilísticos complejos podían mejorar considerablemente los resultados. Esta conclusión está en concordancia tanto con los resultados que avalan la gran cantidad de problemas donde EDAs con modelos simples tienen aplicación [152], como con aquellos que describen el campo de aplicación y las ventajas de los EDAs Bayesianos, capaces de considerar interacciones complejas [75, 104, 11].

Nuestro trabajo se ha centrado en el estudio de las factorizaciones de la probabilidad como método para la aproximación de distribuciones. Una conclusión importante de nuestra investigación es que tanto el uso de mezclas de modelos como el de las aproximaciones Kikuchi permiten aumentar la cantidad de interacciones representadas por los modelos probabilísticos. Este hecho tiene una influencia positiva en la convergencia de los EDAs basados en estos modelos. Aunque nuestro trabajo constituye un paso de avance en la dirección del estudio de los métodos para la aproximación y muestreo de distribuciones en el marco de los EDAs, el número de interrogantes y temas por discernir es todavía considerable.

A partir de nuestro estudio llegamos a la conclusión de que los modelos probabilísticos basados en grafos no dirigidos, que han recibido escasa atención en el diseño de los EDAs en comparación con las redes Bayesianas, tienen un espacio en la concepción de este tipo de algoritmos. De forma similar, los métodos basados en pruebas de independencia pueden ser utilizados también para aprender estructuras complejas basadas en modelos gráficos no dirigidos. Otra conclusión de nuestra investigación es la importancia del análisis de los resultados obtenidos en campos no directamente relacionados con la Computación Evolutiva, como son la Física Estadística y los métodos de inferencia basados en modelos gráficos, para el desarrollo y la concepción de

algoritmos evolutivos más eficientes. Algunas de las respuestas a muchas de las interrogantes abiertas por la investigación en EDAs pueden encontrarse, o pueden inspirarse, en resultados alcanzados en disciplinas donde la modelación probabilística desempeña un papel importante.

Resultados más importantes

1. Introducción de un EDA basado en mezclas de árboles y con aprendizaje a partir del algoritmo EM (capítulo 2).

2. La tesis reporta el primer análisis detallado sobre el uso de factorizaciones inválidas para la aproximación y muestreo en EDAs. Este análisis incluyó una evaluación crítica de las aproximaciones de la probabilidad basadas en factorizaciones válidas, asi como una nueva clasificación de los EDAs, basada en el tipo de factorización inválida usada por cada algoritmo (capítulos 3 y 4).

3. Introducción de los modelos probabilísticos basados en grafos de cliques como una manera de extender la capacidad de representación de los modelos probabilísticos al espacio de las factorizaciones inválidas ordenadas (capítulo 3).

4. Introducción de los modelos probabilísticos basados en la aproximación Kikuchi como una manera de extender la capacidad de representación de los modelos probabilísticos al espacio de las factorizaciones inválidas no ordenadas (capítulo 4).

5. Introducción de algoritmos de aprendizaje de aproximaciones Kikuchi y mezcla de aproximaciones Kikuchi (capítulo 4).

6. Introducción por primera vez de un EDA que utiliza factorizaciones no válidas de manera sistemática y es potencialmente capaz de cambiar entre clases diferentes de modelos probabilísticos de acuerdo a la complejidad de los datos (capítulo 4).

7. Introducción por primera vez en EDAs del algoritmo de muestreo de Gibbs para el muestreo de las distribuciones (capítulo 4).

8. Introducción de un EDA basado en mezclas de aproximaciones Kikuchi (capítulo 4).

Recomendaciones

Algunos de los tópicos que a nuestro juicio merecerán atención en el futuro, o que constituyen temas de nuestro trabajo actual han sido ya mencionados en las secciones 2.9 y 4.10. En esta sección nos limitamos a exponer un grupo de recomendaciones generales.

Consideramos que el estudio de la clase de factorizaciones inválidas, métodos para su aprendizaje y muestreo, y evaluación en el contexto de los EDAs debe constituir un campo de investigación promisorio para los algoritmos evolutivos que utilizan estimación de distribuciones. Recomendamos la investigación de esta temática, particularmente asociada a la resolución de problemas con restricciones.

Así mismo recomendamos la investigación en métodos de combinación de modelos probabilísticos, que incluyen a las mezclas de distribuciones, pero también a otras posibles alternativas para combinar modelos probabilísticos. En el caso particular de las mezclas de distribuciones recomendamos el diseño de estrategias de aprendizaje factibles de ser aplicadas en el contexto EDA y que permitan adaptar el número de componentes de las mezclas a la complejidad de los datos.

Uno de los desafíos para los algoritmos evolutivos son las funciones y problemas dinámicos [15]. Una aproximación EDA para la solución de estos problemas comprendería el uso de modelos probabilísticos capaces de representar la dinámica de las funciones [136]. Señalamos la optimización de funciones dinámicas como un tema de trabajo futuro. Igualmente se hace necesaria la creación de modelos teóricos que permitan evaluar la calidad de las diferentes aproximaciones probabilísticas utilizadas en el marco de los EDAs.

Desde su creación, los algoritmos evolutivos permitieron una aproximación original a la resolución de problemas de optimización y búsqueda. El surgimiento de los EDAs ha significado la definición de nuevos presupuestos para la concepción de algoritmos evolutivos más eficientes. Uno de estos presupuestos es la importancia de una apropiada estimación de las distribuciones de probabilidad, y una de las alternativas consideradas en la estimación es el uso de factorizaciones. En esta tesis hemos procurado avanzar en el camino de desentrañar algunas de las interrogantes asociadas al uso de las factorizaciones en el contexto de los EDAs.

A Funciones de prueba y marco experimental usado en la evaluación de los EDAs

A.1. Evaluación de la eficacia de los EDAs

La eficacia de un EA es usualmente evaluada estudiando la habilidad del algoritmo para alcanzar la solución óptima, o para alcanzar la mejor de las soluciones conocidas. No obstante, la idea de reducir la evaluación de los GAs al porciento de éxitos alcanzado por el algoritmo ha sido cuestionada. En [51] se declara que "el énfasis usual en la convergencia es una limitación considerable en el pensamiento actual sobre los procedimientos de búsqueda". Como una alternativa, De Jong [61] ha propuesto medidas "online" para evaluar el desempeño continuado de un GA (es decir, su habilidad para conseguir un aceptable desempeño rápidamente). Esta controversia justifica la definición de un marco metodológico para la validación de los algoritmos presentados en esta tesis.

El criterio fundamental usado en esta tesis para evaluar un algoritmo es su habilidad de converger al valor óptimo cuando es conocido, o al mejor resultado conocido hasta el momento.

Para algoritmos con desempeños similares en cuanto a convergencia, es necesario determinar otro criterio que sirva para medir su eficiencia. En muchas aplicaciones, el tiempo que necesita el algoritmo de optimización para alcanzar la solución aceptada es la medida escogida. Sin embargo, en algunos casos el tiempo no resulta el mejor criterio a ser considerado. En [96] se presentan varias situaciones donde considerar el número de evaluaciones de la función llevadas a cabo por el algoritmo resulta una opción mejor. En ocasiones, la evaluación de una solución lleva asociada un costo alto de recursos (ej. cada solución representa la cantidad de reactivos diferentes a ser usados en una reacción química), o un costo asociado a la intervención humana (ej. cada solución es un trabajo artístico de diseño a ser evaluado). En todos estos casos, el tiempo necesitado por el algoritmo es un criterio secundario, siendo el más importante el de disminuir el número de evaluaciones.

En esta tesis consideramos el número de evaluaciones, y no el tiempo, como el criterio principal de eficiencia, para algoritmos que exhiben una razón de éxito similar en la convergencia al óptimo.

A.1.1. Teorema No free lunch

Wolpert y Macready [167] han demostrado que todos los algoritmos de búsqueda tienen el mismo comportamiento promedio si todos los posibles espacios de la búsqueda son considerados.

En este sentido ninguno de ellos es mejor que la búsqueda aleatoria. Otros autores plantean [39] que escenarios de búsqueda más restringidos, donde la complejidad o dificultad se limita en algún sentido, encajan mejor en problemas de optimización de la vida real que el escenario descrito en [167]. Para estos escenarios restringidos es posible demostrar la conveniencia de usar una clase de técnicas de optimización sobre otras.

No resulta una sorpresa reconocer que una meta del estudio sobre GAs haya sido identificar, y hasta donde fuese posible describir, la clase de problemas que un GA puede resolver. Como es de esperar, existen varios dominios de dificultad donde los GAs no tienen éxito. La construcción de funciones donde los GAs encaran dificultades, y la mejora del GA para poder enfrentarlas de manera exitosa, es una de las vías que ha llevado a la creación de los EDAs. Algunas de las funciones diseñadas para evaluar los GAs son útiles también para comparar la conducta de diferentes EDAs en varios dominios de dificultad.

A.1.2. Modalidades de dificultad en EDAs

EDAs pueden utilizarse en la resolución de problemas difíciles donde la identificación y uso de interacciones juega un papel importante para realizar una búsqueda exitosa. Sin embargo, la existencia de interacciones no implica necesariamente que un problema sea difícil. Sólo las interacciones malignas, tal y como han sido definidas en [62], son muy difícil para los GAs tradicionales. Un problema con interacciones malignas es normalmente un problema decepcionante, en el sentido de la decepción propuesto por Goldberg et al. [51, 34].

Por otro lado, las interacciones malignas no son la única fuente de dificultad para la optimización evolutiva. La simetría, que puede ser considerada como una forma particular de multimodalidad, es difícil de tratar con EDAs, así como con otros métodos de optimización. Mientras el uso de modelos probabilísticos más expresivos puede ayudar a representar más interacciones, ésta alternativa no resulta necesariamente útil para resolver otras dimensiones de dificultad, como la simetría. Como parte del trabajo de tesis hemos estudiado el desempeño de los EDAs para problemas simétricos y funciones multimodales, proponiendo funciones de prueba que ayudan a entender la influencia de los diferentes dominios de dificultad en el comportamiento de los EDAs [142, 133].

Los aspectos tratados anteriormente han determinado la selección de las funciones y problemas usados en esta tesis para evaluar los algoritmos. Los problemas se han seleccionado considerando los criterio siguientes:

- Se reconocen como difíciles en la literatura de los EAs.
- Sirven para caracterizar un tipo particular de interacciones entre las variables de los problemas.
- Son problemas considerados como difíciles para otras técnicas de optimización.

Usamos dos clases principales de funciones: las funciones teóricas, y las funciones derivadas de problemas prácticos. En la sección siguiente presentamos las funciones, así como las razones que avalan nuestra selección.

A.2. Funciones teóricas usadas en los experimentos

En las Funciones Aditivas Descomponibles (Additively Decomposed Functions (ADFs)) el valor de la función es la suma de la evaluación de varias subfunciones definidas en subconjuntos de variables de X. En las ADFs, la posible interacción entre las variables está reducida a un subconjunto de ellas. Estas funciones son usadas para simular problemas que pueden descomponerse en subproblemas más pequeños.

Definición A.1. *Una ADF de orden k se define por*

$$f(x) = \sum_{s_i \in S} f_i(\Pi_{s_i} x) \qquad S = \{s_1, \ldots, s_l\} \quad s_i \subseteq X, |s_i| = k \tag{A.1}$$

Definición A.2. *Sea* $u(x) = \sum_{i=1}^{n} x_i$, $f(x)$ *es una función unitaria (unitation function) si* $\forall x, y \in B^n, u(x) = u(y) \Rightarrow f(x) = f(y)$

$u(x)$ es ella misma una función unitaria, normalmente llamada *Onemax* porque alcanza el máximo en $x = (1, \ldots, 1)$. Una función unitaria puede definirse en términos de su cantidad de unos $u(x)$ o de una manera más simple u.

A.2.1. Función BigJump

$$BigJump(u,n,m,k) = \begin{cases} u & para \quad 0 \leq u \leq n-m \\ 0 & para \quad n-m \leq u \leq m \\ k \cdot n & para \quad u = m \end{cases} \tag{A.2}$$

La función *BigJump* se introdujo en [86]. Mientras más grande es el parámetro m para esta función, más ancho es el valle que separa al óptimo global del resto de los óptimos locales. El parámetro k puede ser aumentado para dar un peso mayor al valor del máximo.

A.2.2. Funciones decepcionantes

Las funciones unitarias también son útiles para la definición de una clase de funciones donde la dificultad está dada por las interacciones entre subconjuntos de variables. La siguiente ADFs de orden k se define como una suma de funciones unitarias f_k de k variables.

Función $f_{3deceptive}$:

$$f_{3deceptive}(x) = \sum_{i=1}^{i=\frac{n}{3}} f_{dec}^{3}(x_{3i-2}, x_{3i-1}, x_{3i}) \tag{A.3}$$

donde

Función f_{dec}^{3}:

$$f^3_{dec}(u) = \begin{cases} 0{,}9 & for \quad u = 0 \\ 0{,}8 & for \quad u = 1 \\ 0{,}0 & for \quad u = 2 \\ 1{,}0 & for \quad u = 3 \end{cases}$$

$$f_{deceptivek}(x) = \sum_{i=1}^{i=\frac{n}{k}} f^k_{dec}(x_{k\cdot i-k+1} + x_{k\cdot i-2} + \cdots + x_{k\cdot i}) \tag{A.4}$$

$$f^k_{dec}(u) = \begin{cases} k-1 & para \quad u = 0 \\ k-2 & \quad u = 1 \\ & \cdots \\ k-i-1 & \quad u = i \\ & \cdots \\ k\cdot n & para \quad u = k \end{cases}$$

Las funciones decepcionantes (deceptive) [35, 34] muestran la naturaleza decepcionante de la conducta de los GAs, y para tratar los problemas dados por la convergencia a óptimos locales de la función.

Función *IsoPeak*:

x	00	01	10	11
$IsoC_1$	m	0	0	$m-1$
$IsoC_2$	0	0	0	m

(A.5)

$$F_{IsoPeak}(x) \quad = \quad IsoC_2(x_1, x_2) + \sum_{i=2}^{m} IsoC_1(x_i, x_{i+1})$$

LA función $F_{IsoPeak}$ es también una función decepcionante con conjuntos de definición que se solapan, $m = n-1$.

A.2.3. Funciones multimodales y simétricas

Una función multimodal alcanza el óptimo local o global en más de un punto. En las funciones simétricas hay una correspondencia entre los óptimos locales correspondientes a regiones diferentes. Algunas funciones multimodales pueden definirse en como funciones unitarias.

$$symmetric(u) = \begin{cases} u & si \;\; 2\cdot u < n \\ n-u & \text{en otro caso} \end{cases} \tag{A.6}$$

En [142] hemos introducido la siguiente función multimodal que es utilizada en la tesis. El análisis de esta función se difiere a la sección 2.8.3.

Función Nf^3_{dec}:

$$Nf^3_{dec} = \begin{cases} \sum_{i=1}^{i=\frac{n-1}{3}} f^3_{dec}(X_{3i-1}, X_{3i}, X_{3i}+1) & for \quad x_1 = 1 \\ \sum_{i=1}^{i=\frac{n-1}{3}} 1 - f^3_{dec}(X_{3i-1}, X_{3i}, X_{3i}+1) & para \quad x_1 = 0 \end{cases} \tag{A.7}$$

A.2.4. Funciones con interacciones definidas en una vecindad no lineal

En el análisis de interacciones entre las variables es importante considerar interacciones que no dependen de la codificación lineal de las soluciones. Un ejemplo lo constituyen las funciones definidas en retículos.

u	0	1	2	3	4	5
$IsoT_1$	m	0	0	0	0	$m-1$
$IsoT_2$	0	0	0	0	0	m^2

$$F_{IsoTorus}(x) = IsoT_1(x_{1-m+n}, x_{1-m+n}, x_1, x_2, x_{1+m}) + \sum_{i=2}^{n} IsoT_2(x_{up}, x_{left}, x_i, x_{right}, x_{down}) \tag{A.8}$$

donde x_{up},x_{right},etc., se definen como los vecinos apropiados que rodean a x_i.

A.2.5. Funciones con interdependencia entre los bloques constructivos, y frustración

Las funciones HIFF (A.10) y *Cuban*5 (A.9) son dos funciones reconocidas como difíciles. Las funciones jerárquicamente descomponibles, como la HIFF, sirven para modelar problemas con interdependencia entre los bloques constructivos [165]. La función *Cuban*5 se introdujo y estudió en [91], donde demostró ser una función aditiva no-separable muy difícil. El segundo valor mejor de esta función está muy cerca del óptimo global.

$$Cuban5(x) = F^5_{cuban1}(s_0) + \sum_{j=0}^{m} (F^5_{cuban2}(s_{2j+1}) + F^5_{cuban1}(s_{2j+2})), \tag{A.9}$$

donde $s_i = x_{4i}x_{4i+1}x_{4i+2}x_{4i+3}x_{4i+4}$ and $n = 4(2m+1)+1$

$$F^3_{cuban1}(x) = \begin{cases} 0{,}595 & para \quad x = 000 \\ 0{,}200 & para \quad x = 001 \\ 0{,}595 & para \quad x = 010 \\ 0{,}100 & para \quad x = 011 \\ 1{,}000 & para \quad x = 100 \\ 0{,}050 & para \quad x = 101 \\ 0{,}090 & para \quad x = 110 \\ 0{,}150 & para \quad x = 111 \end{cases}$$

$$F^5_{cuban1}(x) = \begin{cases} 4 * F^3_{cuban1}(x_1, x_2, x_3) & si\ x_2 = x_4\ y\ x_3 = x_5 \\ \\ 0 & \text{en otro caso} \end{cases}$$

$$F^5_{cuban2}(x) = \begin{cases} u(x) & para & x_5 = 0 \\ 0 & para & x_1 = 0, x_5 = 1 \\ u(x) - 2 & para & x_1 = 1, x_5 = 1 \end{cases}$$

La función (HIFF) [165] se define por la función recursiva f, $HIFF(x) = f(x_{\{1,\ldots,n\}})$.

$$f(x_{\{1,\ldots,s\}}) = \begin{cases} 1, & \text{si } (|s| = 1) \\ |s| + f(x_{\{1,\ldots,\frac{s}{2}\}}) + f(x_{\{\frac{s}{2}+1,\ldots,s\}}), & \text{si } (|s| > 1) \text{ y} \\ & (\sum_{i=1}^{|s|} x_i = 0, \text{o} \sum_{i=1}^{|s|} x_i = |s|) \\ f(x_{\{1,\ldots,\frac{|s|}{2}\}}) + f(x_{\{\frac{|s|}{2}+1,\ldots,|s|\}}), & \text{en otro caso,} \end{cases} \tag{A.10}$$

donde s es un conjunto de índices de variables adyacentes en x, $|s|$ es la cardinalidad de s, y $x_{\{1\ldots\frac{s}{2}\}}$ y $x_{\{\frac{s}{2}+1\ldots s\}}$ son respectivamente los conjuntos de variables en las mitades izquierda y derecha de x. En el primer llamado a la función $|s| = n = 2^p$, p es un entero que indica el número de niveles jerárquicos.

A.3. Problemas binarios usados en los experimentos

Un rasgo común a las funciones con interacciones consideradas antes es que las interacciones se concentran en subconjuntos pequeños de variables. Adicionalmente, las interacciones son las mismas en cada uno de subconjuntos de definición de las variables, pues la misma subfunción se evalúa en cada uno de ellos. Esta clase de ADFs es más bien restrictiva a efectos de la comparación de varios EDAs. Una alternativa al uso de un número pequeño de funciones de prueba clásicas es definir una clase aleatoria de funciones [152], permitiendo evaluar los algoritmos es un amplio espectro de funciones. En esta tesis usamos una manera de evaluar los algoritmos que sigue la misma pauta del uso de las funciones aleatorias. Pero en lugar de construir directamente una clase aleatoria de funciones, consideramos problemas prácticos que pueden ser resueltos por la optimización de funciones. Además de estar asociadas con un problema combinatorio específico, las funciones tienen las características siguientes.

1. La complejidad y otras características particulares de estas funciones cambian para las diferentes instancias del problema.

2. En general, los patrones de interacción entre las variables son difíciles de caracterizar, y cambian para subconjuntos diferentes de variables.

3. Instancias de prueba están disponibles, y resultados para otros tipos de algoritmos de optimización también existen.

Sigue una descripción de los problemas y su relevancia para nuestra investigación.

$$\phi = (u_1 \vee \neg u_2 \vee u_3) \wedge (u_5 \vee \neg u_1) \wedge (\neg u_2 \vee u_4)$$

$T^1(u_1) = false$	$T^2(u_1) = true$
$T^1(u_2) = true$	$T^2(u_2) = true$
$T^1(u_3) = false$	$T^2(u_3) = false$
$T^1(u_4) = true$	$T^2(u_4) = true$
$T^1(u_5) = true$	$T^2(u_5) = true$
$x^1 = (0,1,0,1,1)$	$x^2 = (1,1,0,1,1)$

Figura A.1: Dos asignaciones T^1 y T^2, a la fórmula ϕ.

A.3.1. Problema de la Satisfacibilidad

Para explicar el problema, empleamos la notación usada en [26]. Sea $U = \{u_1, u_2, \ldots, u_n\}$ un conjunto de n variables Booleanas. Una asignación (parcial) verdadera para U es una función (parcial) $T : U \rightarrow \{true, false\}$. Correspondiendo a cada u hay dos literales, u y $\neg u$. Un literal u(resp. $\neg u$) es $true$ bajo T si y sólo si $T(u) = true$ (resp. $T(u) = false$). Llamamos una cláusula a un conjunto de literales, y a un conjunto o sucesión (tuplo) de cláusulas, lo denominamos una fórmula. Decimos que u se menciona en una cláusula C si $u \in C$ o $\neg u \in C$. Si ϕ es una fórmula, entonces $vars(\phi)$ es el conjunto de las variables mencionadas en ϕ.

Sea ϕ una fórmula, $U = vars(\phi)$, y C una cláusula en ϕ. Interpretamos ϕ como una fórmula del cálculo proposicional en Forma Normal Conjuntiva (Conjunctive Normal Form (CNF)), de tal forma que una asignación de verdad T para U satisface C si y sólo si por lo menos un literal $u \in C$ es cierto bajo T, y T satisface ϕ si y sólo si satisface cada cláusula en ϕ. Para hacer más breve la notación, llamamos a veces a una asignación que satisface ϕ, una solución.

La figura A.3.1 muestra dos asignaciones T^1 y T^2 a la fórmula ϕ que tiene tres cláusulas. T^2 es una solución, T^1 no lo es porque hace no satisface la primera cláusula.

Definición A.3. *El problema satisfacibilidad (SAT) es el problema de encontrar una solución para una fórmula.*

La restricción de SAT a los casos donde todas las cláusulas tienen longitud k se denota k-SAT. De interés especial son 2-SAT y 3-SAT. El valor más pequeño de k para el cual k-SAT es NP-completo es 3 [25]. Se han propuesto varios algoritmos para el problema SAT. Un estudio con una clasificación detallada es [53]. Los algoritmos más usados para su solución son los de Davis-Putman [28] y GSAT [149], existen variantes de ambos.

Representación y funciones objetivo

La variable X_i se asocia a la variable Booleana u_i, y $(u_i = true) \Leftrightarrow (x_i = 1)$. Como función objetivo se usó la suma de las cláusulas satisfechas por la asignación.

Hemos utilizado dos conjuntos de instancias SAT para validar nuestros algoritmos. El conjunto uniforme aleatorio 3-SAT (Uniform Random-3-SAT) es una familia problemas SAT obtenidos a partir de la generación aleatoria de fórmulas en CNF. El conjunto de prueba $uf20-91$ comprende 1000 instancias muestreadas de la región de transición de fase del conjunto uniforme aleatorio 3-SAT. La transición de fase se evidencia en un rápido cambio en la solubilidad de las instancias el cual puede ser observado cuando se incrementa (o decrementa) el número de cláusulas para un número de variables n fijo [19]. Las instancias, así como una explicación detallada sobre la manera en que fueron generadas, pueden ser encontradas en el banco de instancias SATLIB[1]. Cada instancia en $uf20-91$ tiene 20 variables con 91 claúsulas. Hemos usados 999 de las 1000 instancias.

También utilizamos un subconjunto de las instancias *aim*. Estas instancias tienen solamente una solución (es decir, solamente una asignación de sus valores hacen el predicado verdadero). Las instancias son nombradas aim-xxx-y-y-zzzz-j donde, *xxx* es el número de variables, $y-y$ muestra el razón entre número de cláusulas/número de variables, *zzzz* es "no" o "yes", la forma de denotar si una instancia es o no satisfactoria, *j* es el número de orden de la instancia.

Para cada parámetro, cuatro instancias son incluidas. Aunque realizamos experimentos con todas las instancias, en esta tesis se presentan los resultados sólo para las instancias aim-50-3-4-yes1-j y aim-100-3-4-yes1-j[2].

A.3.2. Modelo de Ising generalizado

Un modelo clásico del fenómeno del magnetismo es el *modelo de Ising generalizado*, descrito por la energía funcional (Hamiltoniano) (A.11) [94]

$$H = -\sum_{i<j\in L} J_{ij}\sigma_i\sigma_j - \sum_{i\in L} h_i\sigma_i \tag{A.11}$$

L es el conjunto de sitios llamado rejilla. Cada variable σ_i correspondiente al sitio $i \in L$ toma el valor 1 o -1, representando el polo sur de un imán pequeño. Una asignación específica de valores para las variables σ_i se llama una configuración. Las constantes J_{ij} son los coeficientes de la interacción, las constantes h_i representan las fuerzas del campo magnético externo.

Definición A.4. *El problema de hallar un estado mínimo del modelo de Ising es el problema de hallar cualquier configuración para la cual la energía es mínima.*

Cuando todos los J_{ij} del modelo Ising son no-negativos, el modelo se llama ferromagnético. Más difíciles de tratar son los modelos donde los valores de h_i y J_{ij} son arbitrarios. En el caso general este problema es NP-completo.

[1] http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/Benchmarks/SAT/RND3SAT/descr.html

[2] Las instancias pueden ser encontradas en http://www.cs.washington.edu/homes/kautz/DIMACS93/benchmarks/cnf/, una explicación detallada de cómo fueron generadas se encuentra en ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/contributed/iwama/README.instance

En nuestros experimentos tomamos $h_i = 0$, $\forall i \in L$. Para generar instancias aleatorias donde $J_{ij} \in \{-1, 1\}$, cada acoplamiento toma valor -1 con probabilidad 0,5, valor $+1$ en otro caso. Siguiendo [104], hemos verificado los resultados usando el servidor *Spin Glass Ground State*, provisto por el grupo del Profesor Michael Juenger[3].

Representación de las soluciones y función objetivo

Cada estado del sistema es representado por un vector binario de tamaño n, donde n es el número total de magnetos. Cada variable en el vector determina el estado del correspondiente magneto: 0 denota el estado -1, 1 denota el estado $+1$. La minimización de la función de energía se transforma en la maximización de $-H$.

A.4. Diseño de los experimentos

Para estudiar los algoritmos introducidos utilizamos tres tipos principales de experimentos:

1. El primer tipo de experimentos sirve para ilustrar el funcionamiento de los EDAs introducidos. Estos experimentos presentan los efectos de cambiar determinados parámetros de los algoritmos. En algunos casos ellos clarifican la manera en la cual instancias diferentes de un mismo problema pueden influir en el rendimiento de los algoritmos.

2. El segundo tipo de experimentos presenta los resultados de la comparación entre EDAs ya existentes con otros EDAs introducidos en la tesis. Como ha sido explicado antes, el criterio principal de comparación es la razón de éxito de los algoritmos y el número de evaluaciones. Nuestro análisis se ha concentrado en los EDAs, evitando la comparación con otros paradigmas de optimización. Las preguntas de hasta qué punto y en cuales circunstancias, son productivas las comparaciones entre paradigmas diferentes de algoritmos han sido motivo de polémica [66]. Además de los argumentos presentados en [66], enfatizamos que nuestra opción es debida a la intención de concentrarnos en el estudio de la influencia de los modelos probabilísticos en la optimización.

3. El tercer tipo de experimentos determina el tamaño de población necesitado por los algoritmos para resolver los problemas cuando el número de variables es incrementado. Este tipo de experimentos es útil para determinar de forma aproximada la complejidad de los algoritmos.

A.4.1. Experimentos que ilustran el comportamiento de los algoritmos

Hay tres categorías principales de factores que afectan el desempeño de los algoritmos en los experimentos computacionales: problema, algoritmo y marco de prueba [5]. En el estudio de los

[3] www.informatik.uni-koeln.de/lsjuenger/projects/sgs.html

algoritmos, y según la meta de la tesis, el diseño de los experimentos ha privilegiado la investigación de los factores de esos algoritmos relacionados con el modelo probabilístico usado por el EDA. Nos concentramos en las estrategias y parámetros usados en el aprendizaje del modelo, o el paso de muestreo del EDA. Hay varias otras componentes importantes de un EDA que han sido fijadas en los experimentos, por ejemplo: el tipo y parámetros del método de selección, y el número de soluciones elitistas. Otros factores entre aquellos que podrían mejorar el desempeño de los algoritmos se han ignorado de forma ex-profesa en los experimentos. Por ejemplo, no hemos investigado el efecto de una iniciación no aleatoria de la primera población, o el uso de estrategias de remplazamiento más sofisticadas.

Marco computacional

Los resultados de la comparación entre EDAs han sido principalmente obtenidos a partir de nuestras propias simulaciones. En el caso de los FDAs Bayesianos, hemos incorporado a nuestro marco de trabajo el código desarrollado para los algoritmos EBNA [41]. A menos que se plantee explícitamente lo contrario, todos los algoritmos usaron los siguientes parámetros.

1. Generación aleatoria de la población inicial.
2. Selección por truncamiento con $T = 0{,}15$.
3. El individuo mejor en cada generación se pasa a la próxima.
4. Criterios de terminación usados fueron:
   - El óptimo fue encontrado.
   - El número máximo de generaciones se ha alcanzado.
   - Todos los individuos en la población son idénticos.

Para BOA y LFDA hemos realizado experimentos utilizando las implementaciones originales. Para estos dos algoritmos, hemos tomado también algunos resultados aparecidos en publicaciones.

Presentación de los resultados

A menos que se afirme lo contrario, en cada comparación entre EDAs, fueron realizadas 100 ejecuciones de cada experimento. A partir de estos experimentos, las estadísticas siguientes fueron calculadas:

- La razón de éxito (S) que corresponde al número de ejecuciones del total donde el óptimo fue encontrado.
- Valor promedio de las mejores soluciones en cada ejecución ($\bar{f}$).

- Valor promedio de las generaciones necesarias para alcanzar el óptimo ($\bar{g}$), calculado a partir de las ejecuciones donde fue encontrado.
- Número promedio de evaluaciones de la función para alcanzar el óptimo ($\bar{e}$), calculado a partir de las ejecuciones donde fue encontrado.

A.4.2. Experimentos para la determinación del tamaño de la población

Aunque los requerimientos relativos al tamaño de población mínimo necesario para la optimización de determinadas funciones pueden ser aproximadamente calculados para algunos FDAs [104], para la mayoría de las funciones es imposible lograrlo. Los experimentos para el cálculo de tamaño de población incluidos en la tesis han sido concebidos para determinar la complejidad de los EDAs que se presentan en la optimización de funciones difíciles.

La estrategia usual es ejecutar varios experimentos incrementando el tamaño de la población hasta que se alcance una razón de éxito pre-determinada. Existen tres aspectos principales que describen el algoritmo para encontrar el tamaño crítico de la población:

1. Número total de ejecuciones.
2. Razón de éxito requerida para aceptar el tamaño de la población.
3. Método utilizado para cambiar los tamaños de población en la búsqueda del tamaño de población crítico.

Debe tenerse cuidado al comparar tamaños de población crítica encontrados usando diferentes métodos.

En cada experimento para encontrar tamaño de población crítica, realizamos siempre 100 ejecuciones, y requerimos que al menos 90 fueran exitosas. El tamaño de la población inicial se fija de acuerdo a un valor $Ipsize$. Si la ejecución no es exitosa, el tamaño de la población se incrementa en un valor $Apsize$. Una vez que el tamaño de la población crítica se ha encontrado para el número actual de variables, el tamaño inicial de la población para el próximo experimento (con mayor número de variables) es calculado agregando $Upsize$ al tamaño de la población crítica actual. Los parámetros ($Ipsize$,$Apsize$,$Upsize$) definen completamente nuestro experimento.

B Publicación de los resultados

Publicaciones:

1. R. Santana, A. Ochoa, y M. R. Soto (1999). Evolutionary Algorithms for Dynamic Optimization Problems: An approach using Evolutionary Theory and the Incident Edge Model. A. S. Wu, editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, págs. 149-152, Orlando, FL, 1999. Morgan Kaufmann Publishers, San Francisco, CA.

2. R. Santana y A. Ochoa (1999). On Estimation Distribution Algorithms. W. A. S., editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, pág. 402, Orlando, FL, 1999. Morgan Kaufmann Publishers, San Francisco, CA.

3. R. Santana (1999). Towards an intelligent genetic search: Defining measures of convergence. *Proceedings of the students sessions, ACAI'99*, págs. 41-42, Chania, Greece.

4. A. Ochoa, M. R. Soto, R. Santana, J. C. Madera, y N. Jorge (1999). The Factorized Distribution Algorithm and the juction tree: A learning perspective. A. Ochoa, M. R. Soto, y R. Santana, editores, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, págs. 368-377, Habana, Cuba.

5. R. Santana, E. P. de León, y A. Ochoa (1999). The Edge Incident Model. *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, págs. 352-359, Habana, Cuba.

6. R. Santana y A. Ochoa (1999). Dealing with constraints with Estimation Distribution Algorithms: The univariate case. *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, págs. 378-384, La Habana, Cuba.

7. E. P. de León y R. Santana (1999). A hybrid genetic algorithm for a Hamiltonian path problem. *Investigación Operacional*, 20(1) 20–29.

8. F. B. Pereira, P. Machado, E. Costa, A. Cardoso, A. Ochoa, R. Santana, y M. R. Soto (2000). Too busy to learn. *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, págs. 720-727, La Jolla Marriott Hotel La Jolla, California, USA. IEEE Press.

9. R. Santana, F. B. Pereira, E. Costa, A. Ochoa, P. Machado, A. Cardoso, y M. R. Soto (2000). Probabilistic evolution and the Busy Beaver problem. D. Whitley, editor,*Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, págs. 261-268, Las Vegas, Nevada, USA.

10. R. Santana, A. Ochoa, y M. R. Soto. Factorized Distribution Algorithms for functions with unitation constraints. *Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001)*, págs. 158-165, Habana, Cuba.

11. R. Santana, A. Ochoa, y M. R. Soto (2001). The Mixture of Trees Factorized Distribution Algorithm. L. Spector, E. Goodman, A. Wu, W. Langdon, H. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, y E. Burke, editores, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001*, págs. 543-550, San Francisco, CA. Morgan Kaufmann Publishers.

12. R. Santana, A. Ochoa, y M. R. Soto (2001). A Factorized Distribution Algorithm for problems with integer representation. L. Spector, E. Goodman, A. Wu, W. Langdon, H. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, y E. Burke, editores, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001*, pág. 780, San Francisco, CA. Morgan Kaufmann Publishers.

13. R. Santana, A. Ochoa, y M. R. Soto(2001). On the use of Factorized Distribution Algorithms for problems defined on graphs. *Electronic Notes in Discrete Mathematics*, volume 8. Hajo Broersma, Ulrich Faigle, Johann Hurink y Stefan Pickl editores. Elsevier publisher.

14. M. R. Soto, A. Ochoa, y R. Santana (2001). On the use of polytrees in evolutionary computation. *Investigación Operacional*, Universidad de la Habana. 22(3) 132-138.

15. R. Santana, A. Ochoa, y M. R. Soto (2002). Solving problems with integer representation using a tree based Factorized Distribution Algorithm. *Electronic Proceedings of the NAISO-2002 conference*., La Habana, Cuba.

16. R. Santana y H. Mühlenbein (2002). Blocked stochastic sampling versus Estimation of Distribution Algorithms. *Proceedings of the 2002 Congress on Evolutionary Computation*, volume 2, págs. 1390-1395. IEEE press.

17. R. Santana (2003). A Markov Network based Factorized Distribution Agorithm for optimization. *Proceedings of the 14th European Conference on Machine Learning (ECML-PKDD 2003), volume 2837 of Lecture Notes in Artificial Intelligence*, págs. 337-348, Dubrovnik, Croatia. Springer-Verlag.

Los resultados de la tesis se han discutido en los seminarios de grupos de investigación en Sistemas Adaptativos e Inteligencia Artificial de las siguientes instituciones.

- Universidad del País Vasco, Espanha.
- Universidad de Coimbra, Portugal.
- Instituto de investigaciones en Inteligencia Artificial, Alemania.
- Universidad de la Habana, Cuba.

El resultado “Algoritmos genéticos de Bajo Costo“ es Premio Nacional de Ciencias 2001. Los resultados de esta tesis forman parte del mismo.

Bibliografía

[1] S. ACID Y L. M. DECAMPOS. Approximations of causal networks by polytrees: An empirical study. En B. BOUCHOR-MEUNIER, R. R. YAGER Y L. A. ZADEH, editores, "Advances in Intelligence Computing", vol. 945 de "Lecture Notes in Computer Science", págs. 149–158. Springer Verlag (1995).

[2] H. AKAIKE. A new look at the statistical identification model. *EEE Transactions on Automatic Control* págs. 716–723 (1974).

[3] S. BALUJA. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA (1994).

[4] S. BALUJA Y S. DAVIES. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. En "Proceedings of the 14th International Conference on Machine Learning", págs. 30–38. Morgan Kaufmann (1997).

[5] R. S. BARR, B. L. GOLDEN, J. P. KELLY, M. G. RESENDE Y W. R. STEWART. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics* **1**, 9–32 (1995).

[6] C. BERGE. "Graphs". North-Holland, Mathematical Library, New York (1985).

[7] J. BESAG. Markov Chain Monte Carlo for statistical inference. Informe técnico Working paper No. 9, Center for Statistical and Social Science, University of Washington (September 2000).

[8] J. BILMES. Dynamic Bayesian multinets. En "Proceedings of the 16th conference on Uncertainty in Artificial Intelligence.", págs. 38–45. Morgan Kaufmann Publishers (2000).

[9] R. BLANCO Y J. A. LOZANO. "Estimation of Distribution Algorithms. A new tool for Evolutionary Optimization", cap. An Empirical Comparison of Discrete Estimation of Distribution Algorithms, págs. 163–176. Kluwer Academic Publishers, Boston/Dordrecht/London (2002).

[10] I. BOMZE, M. BUDINICH, P. PARDALOS Y M. PELILLO. The maximum clique problem. En D.-Z. DU Y P. M. PARDALOS, editores, "Handbook of Combinatorial Optimization (supp. Vol. A)", vol. 4, págs. 1–74. Kluwer Academic Publishers, Boston, MA (1999).

[11] P. A. BOSMAN. "Design and Application of Iterated Density-Estimation Evolutionary Algorithms". Tesis Doctoral, Universiteit Utrecht, Utrecht, The Netherlands (2003).

[12] P. A. BOSMAN y D. THIERENS. Linkage information processing in distribution estimation algorithms. En W. BANZHAF, J. DAIDA, A. E. EIBEN, M. H. GARZON, V. HONAVAR, M. JAKIELA y R. E. SMITH, editores, "Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99", vol. I, págs. 60–67, Orlando, FL (1999). Morgan Kaufmann Publishers, San Francisco, CA.

[13] P. A. BOSMAN y D. THIERENS. Ideas based on the normal kernels probability density function. Informe técnico UU-CS-2000-11, Utrecht University (2000).

[14] P. A. BOSMAN y D. THIERENS. Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. *International Journal of Approximate Reasoning* **31**(3), 259–289 (2002).

[15] J. BRANKE. Evolutionary approaches to Dynamic Optimization Problems -a survey-. En A. S. WU, editor, "Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, Workshop Program", págs. 134–137, Orlando, FL (1999). Morgan Kaufmann Publishers, San Francisco, CA.

[16] C. BRON y J. KERBOSCH. Algorithm 457—finding all cliques of an undirected graph. *Communications of the ACM* **16**(6), 575–577 (1973).

[17] D. F. BROWN, A. GARMENDIA-DOAL y J. A. W. MCCALL. Markov random field modelling of royal road genetic algorithms. En P. COLLET, editor, "Proceedings of EA 2001", vol. 2310 de "Lecture Notes in Computer Science", págs. 65–76. Springer Verlag (2002).

[18] G. CELEUX, D.CHAUVEAU y J. DIEBOLT. On stochastic versions of the EM algorithm. Informe técnico No. 2514. iSSN 0249-6399, INRIA (March 1995).

[19] P. CHEESEMAN, B. KANEFSKY y W. M. TAYLOR. Where the really hard problems are. En "Proceedings IJCAI-91", págs. 331–337 (1991).

[20] S. CHIB y B. CARLIN. On MCMC sampling in hierarchical longitudinal models. *Statistics and Computing* **9**, 17–26 (1999).

[21] D. M. CHICKERING, D. GEIGER y D. HECKERMAN. Learning Bayesian networks is NP-hard. Informe técnico MSR-TR-94-17, Microsoft Research, Redmond, WA (1994).

[22] D. CHO y B. ZHANG. Evolutionary optimization by distribution estimation with mixtures of factor analyzer. En "Proceedings of the 2002 Congress on Evolutionary Computation", vol. 2, págs. 1397–1401. IEEE press (2002).

[23] C. K. CHOW y C.Ñ. LIU. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* **IT14**(3), 462–467 (1968).

[24] C.-J. CHUNG y R. G. REYNOLDS. A testbed for solving optimization problems using cultural algorithms. En "Evolutionary Programming", págs. 225–236 (1996).

[25] S. A. COOK. The complexity of theorem-proving procedures. En "Proceedings of the Third Annual ACM Symposium on Theory of Computing", págs. 151–158, Shaker Heights, Ohio (1971).

[26] S. A. COOK Y D. MITCHELL. Finding hard instances of the satisfiability problem: A survey. En "Proceedings of the DIMACS Workshop on Satisfiability Problems", Washington D.C. (1997).

[27] R. CRUZ. "Solución de problemas de optimización combinatoria mediante Redes Neuronales Artificiales". Tesis Doctoral, Universidad de la Habana (1999).

[28] M. DAVIS Y H. PUTNAM. A computing procedure for quantification theory. *Journal of the ACM* (7), 201–215 (1960).

[29] J. S. DE BONET, C. L. ISBELL Y P. VIOLA. MIMIC: Finding optima by estimating probability densities. En M. C. MOZER, M. I. JORDAN Y T. PETSCHE, editores, "Advances in neural information processing systems", vol. 9, pág. 424. The MIT Press, Cambridge (1997).

[30] E. P. DE LEÓN, A. OCHOA Y R. SANTANA. A genetic algorithm for a Hamiltonian path problem. En "Proceedings of the X International Conference on Industrial and Engineering Applications of AI and Expert Systems", Atlanta. USA (1997).

[31] E. P. DE LEÓN Y R. SANTANA. A hybrid genetic algorithm for a Hamiltonian path problem. *Revista Investigación Operacional* **20**(1), 20–29 (1999).

[32] E. P. DE LEÓN, R. SANTANA Y A. OCHOA. A genetic algorithm for a Hamiltonian path problem: Mutation - crossover interaction. En "Proceedings of the 13th ISPE/IEE International Conference on CAD/CAM Robotics and Factories of the Future 97", págs. 1001–1006, Universidad Tecnológica de Pereira, Colombia (December 1997).

[33] K. DEB. "Binary and floating-point function optimization using messy genetic algorithms". Tesis Doctoral, University of Illinois at Urbana-Champaign, Urbana, Illinois (1991).

[34] K. DEB Y D. E. GOLDBERG. Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence* **10**, 385–408 (1994).

[35] K. DEB, J. HORN Y D. E. GOLDBERG. Multimodal deceptive functions. *Complex Systems* **7**, 131–153 (1993).

[36] R. DECHTER. Constraint networks. En S. C. SHAPIRO, editor, "Encyclopedia of Artificial Intelligence", vol. 1. Addison-Wesley Publishing Company (1992). Second Edition.

[37] A. P. DEMPSTER, N. M. LAIRD Y D. B. RUBIN. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society* **B**(39), 1–38 (1977).

[38] M. DORIGO, V. MANIEZZO Y A. COLORNI. The ant system: An autocatalytic optimizing process. Informe técnico 91-016, Politecnico di Milano, Italy (1991).

[39] S. DROSTE, T. JANSEN y I. WEGENER. Perhaps not a free lunch but at least a free appetizer. En W. BANZHAF, J. DAIDA, A. E. EIBEN, M. H. GARZON, V. HONAVAR, M. JAKIELA y R. E. SMITH, editores, "Proceedings of the Genetic and Evolutionary Computation Conference(GECCO '99)", págs. 833–839, San Francisco, CA (1999). Morgan Kaufmann Publishers, Inc.

[40] O. ELERIAN, S. CHIB y N. SHEPHARD. Likelihood inference for discretely observed non linear diffusions. *Econometrica* (2001).

[41] R. ETXEBERRIA, I. INZA y E. BENGOETXEA. Source code of the EBNA algorithm.

[42] R. ETXEBERRIA y P. LARRAÑAGA. Global optimization using Bayesian networks. En "Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)", págs. 151–173, Habana, Cuba (March 1999).

[43] R. ETXEBERRIA, P. LARRAÑAGA y J. M. PICAZA. Reducing Bayesian networks complexity while learning from data. En "Proceedings of the Causal Models and Statistical Learning Seminar", págs. 151–172, London, UK (March 1997).

[44] B. EVERITT y D. HAND. "Mixture Models: Inference and Applications to Clustering". Chapman and Hall, London (1981).

[45] M. GALLAGHER, M. FREAN y T. DOWNS. Real-valued evolutionary optimization using a flexible probability density estimator. En "Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99", vol. I, págs. 840–846, Orlando, FL (1999). Morgan Kaufmann Publishers, San Francisco, CA.

[46] M. R. GALLAGHER. "Multi-Layer Perceptron Error Surfaces: Visualization, Structure and Modelling Models for Iterative Global Optimization". Tesis Doctoral, University of Queensland, Queensland, Australia (2000).

[47] D. GEIGER y D. HECKERMAN. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence* (82), 45–74 (1996).

[48] D. GEMAN. Random fields and inverse problems in imaging. **1427**, 117–193 (1991).

[49] S. GEMAN y D. GEMAN. Stochastic relaxation, Gibbs distributions, and Bayesian restoration of images. *IEEE transactions on pattern analysis and Machine Intelligence* (6), 721–741 (1984).

[50] F. GLOVER. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research* **5**, 533–549 (1986).

[51] D. E. GOLDBERG. "Genetic algorithms in search, optimization, and machine learning". Addison-Wesley, Reading, MA (1989).

[52] P. GREEN. Reversible jump Markov Chain Monte Carlo computation and Bayesian model determination. *Biometrika* (82), 711–732 (1995).

[53] J. GU, Q.-P. GU Y D. Z. DU. Convergence properties of optimization algorithms for the satisfiability (sat) problem. vol. 45, págs. 209–219 (1996).

[54] G. HARIK. Linkage learning via probabilistic modeling in the ECGA. IlliGAL Report No. 99010, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (1999).

[55] G. R. HARIK. "Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms". Tesis Doctoral, University of Michigan, Ann Arbor (1997). Also IlliGAL Report No. 97005.

[56] G. R. HARIK Y D. E. GOLDBERG. Learning linkage. *Foundations of Genetic Algorithms* **4**, 247–262 (1996).

[57] G. R. HARIK, F. G. LOBO Y D. E. GOLDBERG. The compact genetic algorithm. IlliGAL Report No. 97006, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana (1997).

[58] D. HECKERMAN, D. M. CHICKERING, C. MEEK, R. ROUNTHWAITE Y C. M. KADIE. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research* **1**, 49–75 (2000).

[59] J. H. HOLLAND. "Adaptation in natural and artificial systems". University of Michigan Press, Ann Arbor, MI (1975).

[60] S. IOFFE Y D. A. FORSYTH. Human tracking with mixtures of trees. En "ICCV01", págs. I: 690–695 (2001).

[61] K. A. D. JONG. "An analysis of the behaviour of a class of genetic adaptive systems". Tesis Doctoral, University of Michigan (1975).

[62] L. KALLEL, B.ÑAUDTS Y R. REEVES. Properties of fitness functions and search landscapes. En L. KALLEL, B.ÑAUDTS Y A. ROGERS, editores, "Theoretical Aspects of Evolutionary Computing", págs. 177–208. Springer Verlag (2000).

[63] R. KIKUCHI. A theory of cooperative phenomena. *Physical Reviews* **81**(6), 988–1003 (1951).

[64] S. KIRKPATRICK, C. D. J. GELATT Y M. P. VECCHI. Optimization by simulated annealing. *Science* **220**, 671–680 (May 1983).

[65] U. KJAERULFF. On the effective implementation of the Iterative Proportional Fitting procedure. *Computational Statistics and Data Analysis* (19), 177–189 (1990).

[66] J. KOZA. Comments on cross paradigm comparisons of genetic programming with existing machine learning paradigms. Unpublished manuscript. Distributed to GP Mailing List (1995).

[67] J. R. KOZA. "Genetic programming: On the programming of computers by means of natural selection". The MIT Press, Cambridge, MA (1992).

[68] P. LARRAÑAGA. "Estimation of Distribution Algorithms. A new tool for Evolutionary Optimization", cap. An Introduction to Probabilistic Graphical Models, págs. 25–54. Kluwer Academic Publishers, Boston/Dordrecht/London (2002).

[69] P. LARRAÑAGA, R. ETXEBERRIA, J. A. LOZANO Y J. M. P. NA. Optimization by learning and simulation of Bayesian and gaussian networks. Technical Report EHU-KZAA-IK-4/99, Intelligent Systems Group, University of the Basque Country (December 1999).

[70] P. LARRAÑAGA Y J. LOZANO. Synergies between evolutionary computation and probabilistic graphical models. *International Journal of Approximate Reasoning* **31**(3), 155–1566 (2002).

[71] P. LARRAÑAGA, J. LOZANO Y H. MÜHLENBEIN. Algoritmos de estimación de distribuciones en problemas de optimización combinatoria. *Revista Iberoamericana de Inteligencia Artificial* **19**(2), 149–168 (2003).

[72] P. LARRAÑAGA Y J. A. LOZANO. "Estimation of Distribution Algorithms. A new tool for Evolutionary Optimization". Kluwer Academic Publishers, Boston/Dordrecht/London (2002).

[73] S. L. LAURITZEN. "Graphical Models". Oxford:Clarendon Press (1996).

[74] S. W. MAHFOUD. "Niching methods for genetic algorithms". Tesis Doctoral, University of Illinois at Urbana-Champaign, Urbana, IL (mayo 1995). Also IlliGAL Report No. 95001.

[75] T. MAHNIG. "Populations basierte Optimierung durch das Lernen von Interaktionen mit Bayes'schen Netzen". Tesis Doctoral, University of Bonn, Sankt Augustin, Germany (2001). GMD Research Series No. 3/2001.

[76] T. MAHNIG Y H. MÜHLENBEIN. Comparing the adaptive Boltzmann selection schedule SDS to truncation selection. En "Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001)", págs. 121–128, Habana, Cuba (March 2001).

[77] T. MAHNIG Y H. MÜHLENBEIN. Optimal mutation rate using Bayesian priors for Estimation of Distribution Algorithms. En K. STEINHÖFEL, editor, "Proceedings of the First Symposium on Stochastic Algorithms: Foundations and Applications, SAGA-2001", vol. 2264 de "Lecture Notes in Computer Science", págs. 33–48. Springer (2001).

[78] O. C. MARTIN, R. MONASSON Y R. ZECCHINA. Statistical mechanics methods and phase transitions in optimization problems. *Theoretical Computer Science* **265**, 3–67 (2001).

[79] G. MCLACHLAN Y D. PEEL. "Finite Mixture Models". John Wiley and Sons (2000).

[80] M. MEILA. "Learning Mixtures of Trees". Tesis Doctoral, Massachusetts Institute of Technology (1999).

[81] M. MEILA Y M. JORDAN. Learning with mixtures of trees. *Journal of Machine Learning Research* **1**, 1–48 (2000).

[82] V. V. MIAGKIKH Y W. F. PUNCH. A generalized approach to handling parameter interdependencies in probabilistic modeling and reinforcement learning optimization algorithms. Informe técnico GARAGe99-09-01, Genetic Algorithms Research and Applications Group (GARAGe), Michigan State University (1999).

[83] T. MORITA. Formal structure of the cluster variation method. *Progressive Theoretical Physics Supplements* **115**, 27–39 (1994).

[84] H. MÜHLENBEIN. The equation for response to selection and its use for prediction. *Evolutionary Computation* **5**(3), 303–346 (1997).

[85] H. MÜHLENBEIN Y T. MAHNIG. Convergence theory and applications of the Factorized Distribution Algorithm. *Journal of Computing and Information Technology* **7**(1), 19–32 (1998).

[86] H. MÜHLENBEIN Y T. MAHNIG. Evolutionary optimization using graphical models. *New Generation Computing* **18**(2), 157–166 (2000).

[87] H. MÜHLENBEIN Y T. MAHNIG. "Theoretical Aspects of Evolutionary Computing", cap. Evolutionary Algorithms: From Recombination to Search Distributions, págs. 137–176. Springer Verlag, Berlin (2000).

[88] H. MÜHLENBEIN Y T. MAHNIG. Evolutionary computation and beyond. En Y. UESAKA, P. KANERVA Y H. ASOH, editores, "Foundations of Real-World Intelligence", págs. 123–188. CSLI Publications, Stanford, California (2001).

[89] H. MÜHLENBEIN Y T. MAHNIG. Evolutionary synthesis of Bayesian networks for optimization. *Advances in Evolutionary Synthesis of Neural Systems, MIT Press* págs. 429–455 (2001).

[90] H. MÜHLENBEIN Y T. MAHNIG. Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. *International Journal on Approximate Reasoning* (2002). to appear.

[91] H. MÜHLENBEIN, T. MAHNIG Y A. OCHOA. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics* **5**(2), 213–247 (1999).

[92] H. MÜHLENBEIN Y G. PAASS. From recombination of genes to the estimation of distributions I. Binary parameters. En A. EIBEN, T. BÄCK, M. SCHOENAUER Y H. SCHWEFEL, editores, "Parallel Problem Solving from Nature - PPSN IV", págs. 178–187, Berlin (1996). Springer Verlag.

[93] H. MÜHLENBEIN Y H. M. VOIGT. Gene pool recombination in genetic algorithms. En I. H. OSMAN Y J. P. KELLY, editores, "Proceedings of Metaheuristics International Conference", Norwell (1995). Kluwer Academic Publishers.

[94] B.ÑAUDTS Y J.NAUDTS. Fitness landscapes and problem difficulty: the effect of spin-flip symmetry on the performance of the simple ga. En "Parallel Problem Solving from Nature - PPSN V International Conference", págs. 67–76, Amsterdam, The Netherlands (1998). Springer Verlag. LNCS 1498.

[95] H.ÑEY, U. ESSEN Y R. KNESER. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language* **8**, 1–38 (1994).

[96] A. OCHOA. How to deal with costly fitness functions. En "Proceedings of 13th ISPE/IEE/IFAC International Conference on CAD/CAM/Robotics and Factories of the Future", págs. 788–793, Pereira, Colombia (1997).

[97] A. OCHOA. The Factorized Distribution Algorithm: Initial findings. Personal Communication (1999).

[98] A. OCHOA, H. MÜHLENBEIN Y M. R. SOTO. A Factorized Distribution Algorithm using single connected Bayesian networks. En M. SCHOENAUER, K. DEB, G. RUDOLPH, X. YAO, E. LUTTON, J. J. MERELO Y H.-P. SCHWEFEL, editores, "Parallel Problem Solving from Nature - PPSN VI 6th International Conference", Paris, France (September 16-20 2000). Springer Verlag. LNCS 1917.

[99] A. OCHOA, M. R. SOTO, R. SANTANA, J. C. MADERA Y N. JORGE. The Factorized Distribution Algorithm and the junction tree: A learning perspective. En A. OCHOA, M. R. SOTO Y R. SANTANA, editores, "Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)", págs. 368–377, Habana, Cuba (March 1999).

[100] M. OPPER Y D. SAAD. "Advanced Mean Field Methods : Theory and Practice". MIT press (2001).

[101] P. PAKZAD Y V. ANANTHARAM. Belief propagation and statistical physics. En "Electronic Proceedings of 2002 Conference on Information Sciences and Systems", Princeton University (2002). Paper No.225, CD-ROM, 3 pages.

[102] J. PEARL. "Probabilistic reasoning in intelligent systems: Networks of plausible inference". Morgan Kaufmann, San Mateo, California (1988).

[103] M. PELIKAN. The bayesian optimization algorithm (boa) with decision graphs (May 2000). A C++ Implementation of the BOA with Decision Graphs.

[104] M. PELIKAN. "Bayesian Optimization Algorithm: From single level to hierarchy." Tesis Doctoral, University of Illinois, USA (2002).

[105] M. PELIKAN Y D. E. GOLDBERG. Genetic algorithms, clustering, and the breaking of symmetry. IlliGAL Report No. 2000013, University of Illinois at Urbana-Champaign,Illinois Genetic Algorithms Laboratory, Urbana, IL (2000).

[106] M. PELIKAN, D. E. GOLDBERG Y E. CANTÚ-PAZ. Linkage problem, distribution estimation, and Bayesian networks. IlliGAL Report No. 98013, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (1998).

[107] M. PELIKAN, D. E. GOLDBERG Y E. CANTÚ-PAZ. BOA: The Bayesian Optimization Algorithm. En "Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99", vol. I, págs. 525–532, Orlando, FL (1999). Morgan Kaufmann Publishers, San Francisco, CA.

[108] M. PELIKAN, D. E. GOLDBERG Y F. LOBO. A survey of optimization by building and using probabilistic models. IlliGAL Report No. 99018, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (1999).

[109] M. PELIKAN, D. E. GOLDBERG Y F. LOBO. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications* **21**(1), 5–20 (2002).

[110] M. PELIKAN, D. E. GOLDBERG Y K. SASTRY. Bayesian Optimization Algorithm, decision graphs, and Occam's razor. IlliGAL Report No. 2000020, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (2000).

[111] M. PELIKAN Y H. MÜHLENBEIN. The Bivariate Marginal Distribution Algorithm. En R. ROY, T. FURUHASHI Y P. CHAWDHRY, editores, "Advances in Soft Computing - Engineering Design and Manufacturing", págs. 521–535, London (1999). Springer-Verlag.

[112] M. PELIKAN, K. SASTRY Y D. E. GOLDBERG. Scalability of the bayesian optimization algorithm. *International Journal of Approximate Reasoning* **31**(3), 221–258 (2002).

[113] J. PEÑA, J. A. LOZANO Y P. LARRAÑAGA. "Estimation of Distribution Algorithms. A new tool for Evolutionary Optimization", cap. Benefits of Data Clustering in Multimodal Function Optimization via EDAs, págs. 99–124. Kluwer Academic Publishers, Boston/Dordrecht/London (2002).

[114] F. B. PEREIRA, P. MACHADO, E. COSTA, A. CARDOSO, A. OCHOA-RODRIGUEZ, R. SANTANA Y M. R. SOTO. Too busy to learn. En "Proceedings of the 2000 Congress on Evolutionary Computation CEC00", págs. 720–727, La Jolla Marriott Hotel La Jolla, California, USA (July 2000). IEEE Press.

[115] F. B. PEREIRA, P. MACHADO, E. COSTA, A. CARDOSO, A. OCHOA-RODRÍGUEZ, R. SANTANA Y M. R. SOTO. Too busy to learn. En "Colectânea de Comunicacões", págs. 699–712. Ediliber, Lda., Instituto Politécnico de Coimbra (2000).

[116] R. G. REYNOLDS. An introduction to cultural algorithms. En "Proceedings of the 3rd Annual Conference on Evolutionary Programming", págs. 131–139. World Scientic (1994).

[117] S. RICHARDSON Y P. J. GREEN. On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society* (Series B, 59), 731–792 (1997).

[118] J. J. RISSANEN. Modelling by shortest data description. *Automatica* (14), 465–471 (1978).

[119] J. P. RIVERA Y R. SANTANA. Improving the discovery component of classifier systems by the application of estimation of distribution algorithms. En "Proceedings of the students sessions, ACAI'99", págs. 43–44, Chania, Greece (1999).

[120] J. P. RIVERA Y R. SANTANA. Design of an algorithm based on the estimation of distributions to generate new rules in the xcs classifier system. Informe técnico ICIMAF 2000-100, CEMAFIT 2000-78, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba (June 2000).

[121] A. ROSETE. "Automatic Graph Drawing and Stochastic Hill Climbing". Tesis Doctoral, CEIS, ISPJAE, Cuba (2000). In Spanish.

[122] F. ROTHLAUF, D. E. GOLDBERG Y A. HEINZL. Bad codings and the utility of well-designed genetic algorithms. IlliGAL Report No. 200007, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (2000).

[123] S. RUDLOF Y M. KOPPEN. Stochastic hill climbing by vectors of normal distributions. En "Proceedings of the First online workshop on Soft Computing: (WCS1)", Nagoya, Japan (1996).

[124] R. P. SALUSTOWICZ Y J. SCHMIDHUBER. Probabilistic incremental program evolution: Stochastic search through program space. En M. VAN SOMEREN Y G. WIDMER, editores, "Machine Learning: ECML-97", vol. 1224 de "Lecture Notes in Artificial Intelligence", págs. 213–220. Springer-Verlag (1997).

[125] R. P. SALUSTOWICZ Y J. SCHMIDHUBER. H-pipe: Facilitating hierarchical program evolution through skip nodes. Informe técnico TR-8-98, IDSIA, Lugano, Switzerland (1998).

[126] R. SANTANA. Towards an intelligent genetic search: Defining measures of convergence. En "Proceedings of the students sessions, ACAI'99", págs. 41–42, Chania, Greece (1999).

[127] R. SANTANA. An analysis of the performance of the Mixture of Trees Factorized Distribution Algorithm when priors and adaptive learning are used. Informe técnico ICIMAF 2002-180, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba (March 2002).

[128] R. SANTANA. Estimation of Distribution Algorithms with Kikuchi approximations: Part I. Informe técnico ICIMAF 2003-242, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba (September 2003). Also submitted for publication.

[129] R. SANTANA. A Markov Network based Factorized Distribution Algorithm for optimization. En "Proceedings of the 14th European Conference on Machine Learning (ECML-PKDD 2003)", vol. 2837 de "Lecture Notes in Artificial Intelligence", págs. 337–348, Dubrovnik, Croatia (2003). Springer-Verlag.

[130] R. SANTANA. Estimation of Distribution Algorithms with Kikuchi approximations. (2004). Submitted for publication.

[131] R. SANTANA Y E. P. DE LEÓN. An evolutionary optimization approach for detecting structures on graphs. En DAGLI, AKAY, BUCZAC, ERSOY Y FERNANDEZ, editores, "Smart Engineering System Design: Neural Network, Fuzzy Logic, Rough Sets and Evolutionary Programming", págs. 371–376. ASME press (1998).

[132] R. SANTANA, E. P. DE LEÓN Y A. OCHOA. The Edge Incident Model. En "Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)", págs. 352–359, Habana, Cuba (March 1999).

[133] R. SANTANA Y H. MÜHLENBEIN. Blocked stochastic sampling versus Estimation of Distribution Algorithms. En "Proceedings of the 2002 Congress on Evolutionary Computation", vol. 2, págs. 1390–1395. IEEE press (2002).

[134] R. SANTANA Y A. OCHOA. A Constraint Univariate Marginal Distribution Algorithm. Informe técnico ICIMAF 99-76, CENIA 99-04, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba (1999).

[135] R. SANTANA Y A. OCHOA. Dealing with constraints with Estimation of Distribution Algorithms: The univariate case. En "Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)", págs. 378–384, Habana, Cuba (March 1999).

[136] R. SANTANA, A. OCHOA Y M. R. SOTO. Evolutionary Algorithms for Dynamic Optimization Problems: An approach using Evolutionary Theory and the Incident Edge Model. En A. S. WU, editor, "Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, Workshop Program", págs. 149–152, Orlando, FL (1999). Morgan Kaufmann Publishers, San Francisco, CA.

[137] R. SANTANA, A. OCHOA Y M. R. SOTO. A Factorized Distribution Algorithm for problems with integer representation. En L. SPECTOR, E. GOODMAN, A. WU, W. LANGDON, H. VOIGT, M. GEN, S. SEN, M. DORIGO, S. PEZESHK, M. GARZON Y E. BURKE, editores, "Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001", pág. 780, San Francisco, CA (2001). Morgan Kaufmann Publishers.

[138] R. SANTANA, A. OCHOA Y M. R. SOTO. Factorized Distribution Algorithms for functions with unitation constraints. En "Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001)", págs. 158–165, Habana, Cuba (March 2001).

[139] R. SANTANA, A. OCHOA Y M. R. SOTO. The Mixture of Trees Factorized Distribution Algorithm. Informe técnico ICIMAF 2000-129, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba (January 2001).

[140] R. SANTANA, A. OCHOA Y M. R. SOTO. The Mixture of Trees Factorized Distribution Algorithm. En L. SPECTOR, E. GOODMAN, A. WU, W. LANGDON, H. VOIGT,

M. GEN, S. SEN, M. DORIGO, S. PEZESHK, M. GARZON Y E. BURKE, editores, "Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2001", págs. 543–550, San Francisco, CA (2001). Morgan Kaufmann Publishers.

[141] R. SANTANA, A. OCHOA Y M. R. SOTO. On the use of Factorized Distribution Algorithms for problems defined on graphs. En H. BROERSMA, U. FAIGLE, J. HURINK Y S. PICKL, editores, "Electronic Notes in Discrete Mathematics", vol. 8. Elsevier (2001).

[142] R. SANTANA, A. OCHOA Y M. R. SOTO. Solving problems with integer representation using a tree based Factorized Distribution Algorithm. En "Proceedings of the NAISO-2002 conference." (2002).

[143] R. SANTANA, F. B. PEREIRA, E. COSTA, A. OCHOA-RODRIGUEZ, P. MACHADO, A. CARDOSO Y M. R. SOTO. Probabilistic evolution and the Busy Beaver problem. En D. WHITLEY, editor, "Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference", págs. 261–268, Las Vegas, Nevada, USA (8 July 2000).

[144] K. SASTRY Y D. E. GOLDBERG. Modeling tournament selection with replacement using apparent added noise. En "Intelligent Engineering Systems Through Artificial Neural Networks. Proceedings of the Conference ANNIE 2001", vol. 2, págs. 129–134 (2001).

[145] G. SCHWARZ. Estimating the dimension of a model. *Annals of Statistics* **7**(2), 461–464 (1978).

[146] J. SCHWARZ Y J. OCENASEK. Experimental study: Hypergraph partitioning based on the simple and advanced algorithms BMDA and BOA. En "Proceedings of the Fifth International Conference on Soft Computing", págs. 124–130, Brno, Czech Republic (1999). PC-DIR.

[147] M. SEBAG Y A. DUCOULOMBIER. Extending population-based incremental learning to continuous search spaces. En "Parallel Problem Solving from Nature - PPSN V", págs. 418–427, Berlin Heidelberg (1998). Springer Verlag.

[148] S.E.FIENBERG. "The analysis of cross-classified categorical data". MIT press (1981).

[149] B. SELMAN, H. LEVESQUE Y D. MITCHELL. A new method for solving hard satisfiability problems. En "Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)", págs. 440–446, San Jose, CA, USA (1992).

[150] I. SERVET, L. TRAVE-MASSUYES Y D. STERN. Telephone traffic overloading diagnosis and evolutionary techniques. En "Proceedings of the Third International Conference on Artificial Evolution:(AE'97)", págs. 137–144 (1997).

[151] N. SHEPHARD Y M. PITT. Likelihood analysis of non-Gaussian measurement time series. *Biometrika* **84**(3), 653–667 (1997).

[152] M. R. SOTO. "A Singled Connected Factorized Distribution Algorithm and its cost of evaluation". Tesis Doctoral, University of Havana, Havana, Cuba (July 2003). (In Spanish, adviser A. Ochoa).

[153] M. R. SOTO, A. OCHOA, S. ACID Y L. M. CAMPOS. Bayesian evolutionary algorithms based on simplified models. En "Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)", págs. 360–367, Habana, Cuba (March 1999).

[154] M. R. SOTO, A. OCHOA Y R. SANTANA. On the use of polytrees in evolutionary computation. *Investigación Operacional* **22**(3), 132–138 (2001).

[155] P. SPIRTES, C. GLYMOUR Y R. SCHEINES. "Causation, Prediction and Search", vol. 81 de "Lecture Notes in Statistics". Springer-Verlag, New York (1993).

[156] P. SPIRTES Y C. MEEK. Learning bayesian networks with discrete variables from data. En "Proceedings of the First International Conference on Knowledge Discovery and Data Mining", págs. 294–299, San Francisco (1995). Morgan Kaufmann.

[157] E. B. SUDDERTH, M. J. WAINWRIGHT Y A. S. WILLSKY. Embedded trees: Estimation of Gaussian processes on graphs with cycles. Informe técnico MIT LIDS Technical Report P-2562, Laboratory for Information and Decision Systems, MIT (April 2003).

[158] G. SYSWERDA. Simulated crossover in genetic algorithms. En "Foundations of Genetic Algorithms", págs. 239–295. Morgan Kaufmann (1993).

[159] D. THIERENS Y P. BOSMAN. Multi-objective optimization with iterated density estimation evolutionary algorithms using mixture models. En "Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001), Cuba", págs. 129–136, Habana, Cuba (March 2001).

[160] B. THIESSON, C. MEEK, D. CHICKERING Y D. HECKERMAN. Learning mixtures of Bayesian networks. Informe técnico MSR-TR-97-30, Microsoft Research, Advance Technology Division (February 1997).

[161] S. TSUTSUI. Edge histogram based sampling and its application to aco. En "Proceedings of the Frontiers of Evolutionary Algorithm Conference FEA-2003" (2003).

[162] C. H. M. VAN KEMENADE. Explicit filtering of building blocks for genetic algorithms. En "90", pág. 9. Centrum voor Wiskunde en Informatica (CWI), ISSN 0169-118X (31 1996).

[163] M. J. WAINWRIGHT, T. JAAKKOLA Y A. S. WILLSKY. Tree-based reparameterization framework for analysis of belief propagation and related algorithms. Informe técnico LIDS Technical Report P-2510, Laboratory for Information and Decision Systems, MIT (2001).

[164] M. J. WAINWRIGHT Y M. I. JORDAN. Graphical models, exponential families, and variational inference. Informe técnico Technical Report 649, Department of Statistics, University of California, Berkeley (September 2003).

[165] R. A. WATSON, G. S. HORNBY Y J. B. POLLACK. Modeling building-block interdependency. En "Parallel Problem Solving from Nature - PPSN V International Conference", págs. 97–106, Amsterdam, The Netherlands (1998). Springer Verlag. LNCS 1498.

[166] J. WHITTAKER. "Graphical models in applied multivariate statistics". Wiley Series in Probability and Mathematical Statistics, New York (1991).

[167] D. H. WOLPERT Y W. G. MACREADY. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* **1**(1), 67–82 (April 1997).

[168] M. YAMAMURA, H. KITA Y I. ONO. Introduction to recent progress in genetic algorithms. *Journal of the Japanese Society for Artificial Intelligence* **18**(5), 477–478 (2003).

[169] J. S. YEDIDIA, W. T. FREEMAN Y Y. WEISS. Constructing free energy approximations and generalized belief propagation algorithms. Informe técnico TR-2001-35, Mitsubishi Electric Research Laboratories (August 2002).

[170] J. S. YEDIDIA, W. T. FREEMAN Y Y. WEISS. Understanding belief propagation and its generalizations. Informe técnico TR-2001-22, Mitsubishi Electric Research Laboratories (January 2002).

Índice alfabético