Universidad de las Ciencias Informáticas
Facultad 3

Título: Resolución de algunos problemas clásicos de optimización mediante las metaheurísticas EDA y ACO

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Dairo Reyes Rodríguez

Tutores: Lic. Ansel Yoan González Rodríguez

Lic. Amilkar Yudier Puris Cáceres

Asesor: Dr. Pedro Yobanis Piñeiro Pérez

Consultante: MSc. Eduardo Rojas Duverger

Junio 2007

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Dairo Reyes Rodríguez

_____

Lic. Ansel Yoan González Rodríguez

_____

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

El estudiante Dairo Reyes durante la confección de esta tesis mostró gran independencia en el desarrollo del trabajo, pues, cabe destacar que sus tutores no se encontraban presencialmente durante este curso en la universidad. El trabajo presenta buena ortografía, redacción y concordancia entre las ideas, además de una buena estructuración de los contenidos y un alto rigor científico. Cabe destacar que la presente tesis es el resultado de cinco semestres de dedicación por parte del estudiante, periodo durante el cual demostró una gran responsabilidad en el cumplimiento de las tareas asignadas por sus tutores, así como en la confección del documento.

Como resultado de esta tesis se pudo concluir que los algoritmos evolutivos basados en Redes Bayesianas simplemente conectadas y los algoritmos de optimización basados en colonia de hormigas son una alternativa viable para abordar el problema de Coloración Robusta y el problema de Clasificación Automática no Supervisada.

Es importante destacar que resultados parciales de este trabajo fueron presentados en diferentes eventos que siguen a continuación:

- Estudio de los algoritmos de estimación de las distribuciones
  - III Jornada Científica Estudiantil a nivel de facultad – Relevante.
- Estudio de los algoritmos de estimación de las distribuciones
  - III Jornada Científica Estudiantil a nivel UCI – Mención.
- Colonia de Hormigas, Estado del Arte
  - IV Jornada Científica Estudiantil a nivel de facultad.
- Colonia de Hormigas en el problema de Clasificación Automática no Supervisada
  - IV Jornada Científica Estudiantil a nivel de facultad.
- Algoritmos Evolutivos Basados en la Estimación de las Distribuciones
  - IV Jornada Científica Estudiantil a nivel de facultad.
- Resolución del Problema de Clasificación Automática no Supervisada utilizando los Algoritmos Evolutivos basados en la Estimación de la Distribución
  - IV Jornada Científica Estudiantil a nivel de facultad.

- Algoritmos de Estimación de Distribuciones en el problema de Coloración Robusta
  - IV Jornada Científica Estudiantil a nivel de facultad – Mención.
- Metaheurísticas para la Resolución de Problemas de Optimización. Aplicaciones Prácticas
  - IV Jornada Científica Estudiantil a nivel UCI – Relevante.
- Metaheurísticas para la Resolución de Problemas de Optimización. Aplicaciones Prácticas
  - IV Jornada Científica Estudiantil a nivel UCI – Ponencia de Mayor Rigor Científico.
- Resolución del Problema de Clasificación Automática no Supervisada utilizando los Algoritmos Evolutivos basados en la Estimación de la Distribución
  - Concurso Mejor Artículo de Promoción Científica – Premio.
- Resolución del Problema de Clasificación Automática no Supervisada utilizando los Algoritmos Evolutivos con Estimación de las Distribuciones basados en redes Bayesianas simplemente conectadas
  - Uciencia 2006 – Publicado en las memorias del evento.
- Metaheurísticas para la Resolución de Problemas de Optimización. Aplicaciones Prácticas
  - XVI Fórum de Ciencia y Técnica de la UCI – Destacado.
- Metaheurísticas para la Resolución de Problemas de Optimización. Aplicaciones Prácticas
  - XVI Concurso Nacional Científico de Computación.

Por todo lo anteriormente expresado considero que el estudiante está apto para graduarse como Ingeniero en Ciencias Informáticas; y propongo que se le otorgue al Trabajo de Diploma la calificación de 5 puntos.

Además considero que el estudiante Dairo Reyes ha demostrado que posee las condiciones necesarias para continuar desarrollando este tema de investigación como estudiante de Maestría.

Lic. Ansel Y. González Rodríguez.

_____ _____

Firma Fecha

DATOS DE CONTACTO

Tutor: Lic. Ansel Yoan González Rodríguez

Licenciado en Ciencia de la Computación, 2004, Universidad de La Habana, actualmente se desempeña como Especialista en informática del departamento de Minería de Datos, del Centro de Aplicaciones de Tecnologías de Avanzada.

Correo electrónico: arodriguez@cenatav.co.cu

Para más información visitar: http://www.cenatav.co.cu/es/Tecnicos/md/Ansel1.html

Tutor: Lic. Amilkar Yudier Puris Cáceres

Licenciado en Ciencia de la Computación, 2004, Universidad Central de Las Villas, actualmente se desempeña como profesor en esa institución.

Correo electrónico: ayudier@cei.uclv.edu.cu

Asesor: Dr. C. Pedro Yobanis Piñero Pérez

Doctor en Ciencias de la Computación, 2005, Universidad Central de Las Villas, actualmente se desempeña como Vicedecano de Producción de la Facultad 3 de la Universidad de las Ciencias Informáticas.

Correo electrónico: ppp@uci.cu

Consultante: MSc Ópticas. Profesor auxiliar Eduardo Rojas Duverger

Especialidad de Física. Instituto Superior Pedagógico: Enrique José Varona. 36 años de experiencia en la docencia, ha impartido asignaturas de Física, Matemáticas, Filosofía, actualmente se desempeña como profesor de la asignatura de Metodología de la Investigación en la Facultad 3 de la Universidad de las Ciencias Informáticas.

Correo electrónico: duverger@uci.cu

"Lo que puedes hacer, o sueñes que puedes hacer, empieza"

Johann W. Goethe

AGRADECIMIENTOS

Quisiera agradecer a todos los que de una forma u otra me han ayudado en la confección de esta tesis de grado. Aunque sé que siempre se quedan nombres por mencionar, trataré de mencionar a todos los que me han aportado en sabiduría y me han apoyado y dado ánimos, para verme graduado.

A mi madre, mis abuelas, mi padre, mis tíos, y en general todos mis familiares, que tanto apoyo me han dado y siempre han confiado en mí.

A mi novia por comprenderme y ayudarme tanto, te amo y lo sabes. Yurita: siempre te lo voy a agradecer.

A todos mis amigos por estar siempre ahí para todo lo que he necesitado, gracias Lázaro (mi primo y amigo), Iradiel, Alién, Dariel, Andrés, Yoandrys, Yuri y todos mis demás amigos y amigas que seguro se me quedan, a todos los quiero igual.

Agradecer de forma especial a Ansel, mi tutor, por enseñarme tanto y aportar tanto a mi vida profesional, a Pedrito y Amilkar, mis otros tutores, por todo lo que han hecho por mí. Creo que a los tres no les pago ni con el sueldo de mis primeros 20 años de trabajo.

Dairo Reyes

RESUMEN

Los Algoritmos de Estimación de las Distribuciones (EDA, según sus siglas en inglés), surgen en la última década del siglo XX como alternativa a los Algoritmos Genéticos, dada la incapacidad de estos últimos de captar las relaciones entre las variables. Los EDA, captan de cada generación la mejor distribución de probabilidades para simular a partir de esta, la nueva generación. Actualmente los EDA son muy utilizados para dar solución a problemas de optimización, reconocimiento de patrones, procesamiento de imágenes, etc. Por otra parte, la metaheurística Optimización basada en Colonia de Hormigas (ACO, según sus siglas en inglés), ha demostrado buenos resultados en la solución de problemas de búsqueda de caminos más cortos y proponen soluciones aceptables en tiempo razonable para una serie de problemas difíciles. En la presente tesis se explora el comportamiento de las metaheurísticas bioinspiradas EDA y ACO, en los problemas de Coloración Robusta y de Clasificación Automática No Supervisada.

PALABRAS CLAVE

Metaheurísticas, Algoritmos bioinspirados, Optimización, Algoritmos de Estimación de las Distribuciones, Optimización basada en Colonia de Hormigas.

MOTIVACIONES

A partir del desarrollo y el auge que ha tomado la ciencia, los científicos, ingenieros e investigadores, se han dedicado al estudio del comportamiento de procesos naturales como la evolución, o de insectos como las hormigas, fascinados por la capacidad de estos para obtener el camino más corto, o la característica del proceso evolutivo para aplicarlo a problemas complejos. A partir de estos estudios han surgido algoritmos bioinspirados, a los que pertenecen EDA y ACO. Las motivaciones en que se ha basado este trabajo para su realización son las siguientes [1] [2]:

- Han demostrado ser un buen método de resolución de problemas difíciles, no lineales y que implican un número elevado de variables [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23].

- Proporcionan una alta probabilidad de encontrar un óptimo global, a diferencia de métodos convencionales que pueden quedar atascados en soluciones locales sub-óptimas [23] [24] [25] [26] [27] [28].

- Poseen un rendimiento aceptable en entornos dinámicos, es decir, cuando el problema o sus objetivos son variables en el tiempo [29] [30].

- Pueden optimizar múltiples criterios simultáneamente (por ejemplo, minimizar costes de transporte y tiempo de entrega) [31].

- Pueden ser utilizados en la resolución de problemas con dominio mixto (diferentes tipos de variables) [32].

- La mayoría de las veces es más fácil entender la fundamentación de estas metaheurísticas que complejos métodos matemáticos utilizados con técnicas exactas.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las metaheurísticas son un conjunto de procesos iterativos que guían y modifican las operaciones de los métodos heurísticos subordinados para producir eficientemente soluciones de alta calidad. Algunos métodos heurísticos son: Búsqueda Local, Recocido Simulado, Reducción, Descomposición, Búsqueda Tabú y Algoritmos Genéticos [33].

Las llamadas tecnologías bioinspiradas nacen de la aplicación de conceptos de inspiración biológica al diseño de sistemas analíticos. El objetivo es comprender e imitar la forma en que los sistemas biológicos aprenden y evolucionan. Para diseñar estos sistemas, además de utilizar la computación tradicional numérico-simbólica, se usan otras metodologías tales como las redes neuronales artificiales, la lógica difusa y la computación evolutiva. Por ello, este intento de emulación del funcionamiento de los seres vivos se debe apoyar en un entorno multidisciplinar que agrupa físicos, informáticos, electrónicos, microelectrónicas y áreas de la ingeniería como la biomédica o la neuromórfica, y aspira a conseguir auténticos sistemas electrónicos dotados de sentidos artificiales que permitan facilitar un sinfín de tareas y resolver problemas hasta ahora no resueltos [41].

Situación problemática

El comportamiento de las metaheurísticas ha sido evaluado por parte de los programadores y analistas de todo el mundo a través de algoritmos o funciones para demostrar su utilidad y su eficacia en la resolución de problemas de optimización. Estas técnicas, que pertenecen al área de Inteligencia Artificial, se aplican a algunos problemas, estos, para lograr una solución optimizada, utilizan métodos heurísticos conocidos.

Dentro del grupo de las metaheurísticas relativamente jóvenes, se encuentran los Algoritmos de Estimación de las Distribuciones y la Optimización basada en Colonia de Hormigas, las cuales, han demostrado buenos resultados tras ser aplicados a una gran variedad de problemas de optimización.

Los problemas de Coloración Robusta y Clasificación Automática No Supervisada son problemas Combinatorios y NP-Duros, por tanto, encontrar las soluciones óptimas puede ser computacionalmente costoso.

Un problema combinatorio, según Hillier y Lieberman: "es un modelo matemático consistente en un sistema de ecuaciones y expresiones matemáticas relacionadas que describen la esencia del problema" [34]. En un problema combinatorio cada modelo presenta variables de decisión, una función objetivo y las restricciones, es imposible enumerar explícitamente todas las soluciones posibles y a veces es difícil encontrar una de ellas. Los problemas NP-Duros en cambio, son problemas que pueden o no estar en la clase NP (NP: No determinísticos en tiempo polinomial), pero que son al menos tan difíciles como los NP-Completos. Las versiones de optimización de un problema NP-Completo es NP-duro [35].

En [36] se puede verificar que el problema de Coloración Robusta ha sido un problema muy tratado mediante diversos algoritmos y métodos heurísticos; entre los que se pueden mencionar: temple simulado, búsqueda tabú, algoritmos genéticos, sistemas de hormigas, híbridos y heurísticas para colorear cierto tipo de grafos. Además de los métodos que propiamente propone: un algoritmo de enumeración parcial y un algoritmo híbrido entre genético y voraz para encontrar soluciones aproximadas. El problema de clasificación automática no supervisada se puede comprobar en [37] y [38] que se han empleado técnicas estocásticas, tales como algoritmos genéticos, recocido simulado y búsqueda tabú también para su resolución.

Estos algoritmos y técnicas mencionadas anteriormente, resuelven en alguna medida los problemas de Coloración Robusta y Clasificación Automática no Supervisada, pero no completamente. Esto quiere decir, que las soluciones dadas no garantizan que resuelva totalmente el problema, sino que sean una aproximación a la solución óptima. Debido a esto, constantemente se realizan esfuerzos por parte de los ingenieros e investigadores por mejorar los resultados que se obtienen con cada algoritmo.

Problema científico

¿El uso de Algoritmos de Estimación de las Distribuciones y Optimización por Colonias de Hormigas es una alternativa viable para resolver eficaz y eficientemente los problemas de Coloración Robusta y Clasificación Automática No Supervisada?

Objeto de la investigación

Implementación y evaluación de Metaheurísticas para la resolución de los problemas de Coloración Robusta y Clasificación Automática No Supervisada.

Objetivos de la investigación

Implementar algoritmos de las metaheurísticas EDA y ACO para los problemas de Coloración Robusta y Clasificación Automática No Supervisada y evaluar los resultados

Objetivos específicos

Desarrollar una aplicación en la plataforma .Net que facilite la experimentación y análisis de los resultados.

Hipótesis

Las metaheurísticas Algoritmos de Estimación de las Distribuciones y Optimización por Colonia de Hormigas han logrado resultados satisfactorios aplicados a otros problemas, si se pueden aplicar estas a los problemas de Coloración Robusta y Clasificación Automática no supervisada, entonces, se podrán encontrar soluciones aceptables.

Campo de acción

Implementación y evaluación de EDA y ACO para la resolución de los problemas de Coloración Robusta y Clasificación Automática No Supervisada

Tareas de la investigación

Estudio del Estado del Arte.

- Búsqueda exhaustiva de la literatura.
    - Selección de los artículos científicos a revisar en la primera etapa.
    - Revisión exhaustiva de la literatura seleccionada.
- Entrevistar a profesionales dedicados al estudio de los temas tratados.

Estrategia de desarrollo.

- Selección y definición de estrategia de trabajo.
- Programación de algunos métodos seleccionados.
- Presentación de artículos científicos a partir del trabajo en publicaciones de revistas, concursos, etc.
- Participación en eventos científicos a nivel de base, provincial, nacional, etc. (al menos en uno como ponente).

Lenguaje y herramientas utilizados

Como lenguaje para programar se ha seleccionado C#, pues es un lenguaje puramente orientado a objetos y de alto nivel. Posee características que lo convierten en un lenguaje potente, como son la seguridad, flexibilidad, modularidad y robustez, que unido a su sencillez y eficiencia, hacen de este lenguaje, uno de los más productivos que existen en la actualidad. Podrían enumerarse decenas de mejoras como la extensibilidad de tipos, componentes, operadores, uso de instrucciones seguras, posibilidad de, explícitamente, hacer uso de código inseguro, etc. Desde un punto de vista práctico, es el lenguaje de .NET con más y mejores ejemplos. Además, la migración a C# es fácil para todos los programadores de Java y C++.

Como herramienta para programar o entorno de desarrollo integrado (IDE, por sus siglas en inglés) se escogió Microsoft Visual Studio .NET 2005. Soporta los lenguajes de la plataforma .NET: C#, VB .NET, J# y Visual C++. Esta herramienta puede utilizarse para construir aplicaciones dirigidas a Windows (utilizando Windows Forms), Web (usando ASP.NET y Servicios Web) y dispositivos portátiles (utilizando .NET Compact Framework). Tiene una interfaz más limpia, mayor cohesión y es personalizable, con ventanas informativas de estado que automáticamente se ocultan cuando no se usan. Incluye un depurador integrado en el entorno de edición.

Organización de la tesis

En la introducción se presentan definiciones y conceptos básicos que son imprescindibles para la comprensión del texto, además de definir el objetivo de la investigación. En el primer capítulo se presentan los algoritmos bioinspirados EDA y ACO, se muestra además un estado del arte de los mismos y sus principales características. En el segundo capítulo se explican los problemas que se tratarán en este trabajo y se propone cómo se le va a dar solución a cada problema y mediante cual metaheurística. En el tercer capítulo se muestran los resultados experimentales de la investigación. Finalmente se presentan las conclusiones y las recomendaciones.

CAPÍTULO 1: ALGORITMOS BIOSINSPIRADOS

Desde el comienzo de la humanidad el hombre ha intentado reproducir los patrones que encuentran a su alrededor. Para ello ha estudiado desde el vuelo de las aves y la capacidad de trabajo en grupo de insectos tales como abejas u hormigas, hasta procesos naturales más complejos como la evolución misma de las especies e inspirado en ello, ha desarrollado disímiles líneas de investigación.

La Computación Bio-Inspirada, que como su propio nombre indica, se inspira en los sistemas naturales. Tiene como característica el ser auto-adaptativa, auto-organizada y ser capaz de auto-aprender. Por su capacidad de resolver problemas complejos que representan algunas dificultades para los métodos de computación tradicionales, la computación inspirada en la naturaleza se utiliza actualmente con éxito en muchos campos, tales como la optimización combinatoria, aprendizaje de máquinas y diseño en ingeniería. Se trata de algoritmos basados en técnicas inspiradas en la biología y la evolución. En ellos una población de individuos, que de ellos, algunos pueden convertirse en la solución, busca en paralelo el óptimo dentro del espacio de soluciones posibles. La población evoluciona utilizando operadores que imitan principios naturales (tales como competición, reproducción, variación, etc.) y según leyes probabilísticas. Entre los algoritmos bio-inspirados podemos citar la computación evolutiva (algoritmos genéticos, estrategias evolutivas, programación evolutiva y programación genética) y los sistemas basados en agentes (optimización por nubes de partículas y las colonias de hormigas) [1].

El presente capitulo está organizado de la siguiente forma. En la sección 1.1 se exponen temas referentes a los Algoritmos basados en Estimación de las Distribuciones, dentro de los que se pueden encontrar una breve explicación sobre los algoritmos genéticos, como su principal antecedente; un estado del arte sobre los EDA: que es un EDA, que estructura general poseen y finalmente los tipos que existen. En la sección 1.2 se presentan temas referentes a la Optimización basada en Colonia de Hormigas; muestra información sobre las hormigas naturales y como surgió ACO a partir de estas, luego se presenta la estructura genérica de un ACO y los modelos de optimización que existen basados en Colonias de Hormigas. Por último, en la sección 1.3, se explican las semejanzas que existen entre los EDA y ACO.

1.1 Algoritmos Basados en Estimación de las Distribuciones

Los Algoritmos Evolutivos, son métodos de búsqueda estocásticos que se basan en las ideas esenciales de la selección natural de las especies y de la genética. Estos modelos computacionales utilizan la evolución artificial para resolver problemas de optimización, búsqueda, aprendizaje y simulación de sistemas dinámicos. La diferencia entre los Algoritmos Evolutivos y otras técnicas de optimización está dada por el hecho de que realizan una búsqueda a partir de una población de soluciones y no a partir de un solo punto.

A partir de la década de los 90 se ha progresado con acierto en el estudio y uso de técnicas heurísticas aplicadas a la optimización. Entre dichas técnicas, la referencia se ha centrado en la computación evolutiva.

Basado en los algoritmos genéticos, ha surgido un nuevo tipo de computación evolutiva conocido como EDA donde se generalizan los algoritmos genéticos sustituyendo los operadores de cruce y mutación por el aprendizaje y muestreo de distribuciones de probabilidad de los mejores individuos de la población en cada iteración del algoritmo.

1.1.1 Antecedentes

Los Algoritmos de Estimación de las Distribuciones surgen a partir de algunas deficiencias encontradas en los algoritmos genéticos. ¿Pero, qué es un algoritmo genético?

Es una técnica de programación que imita a la evolución biológica como estrategia para resolver problemas. Dado un problema específico a resolver, la entrada del algoritmo genético es un conjunto de soluciones potenciales a ese problema que se generan aleatoriamente, codificadas de alguna manera, y una métrica llamada función de aptitud que permite evaluar cuantitativamente a cada posible solución. Estas candidatas pueden ser soluciones que ya se sabe que funcionan. Luego el algoritmo genético evalúa cada candidata de acuerdo con la función de aptitud. En una población de soluciones candidatas generadas aleatoriamente, por supuesto, la mayoría no funcionarán en absoluto, y serán eliminadas. Sin embargo, por casualidad, unas pocas pueden ser prometedoras y pueden mostrar actividad, aunque sólo sea actividad débil e imperfecta, hacia la solución del problema [39].

Estas candidatas prometedoras son escogidas por el algoritmo y se les permite reproducirse. Se realizan múltiples copias de ellas, pero las copias no son perfectas; se introducen cambios durante el proceso de copia mediante los operadores de cruce y mutación. Luego, esta descendencia digital prosigue con la siguiente generación, formando un nuevo conjunto de soluciones candidatas, y son sometidas a una nueva ronda de evaluación de aptitud. Las candidatas que han empeorado o no han mejorado con los cambios en su código son eliminadas nuevamente; pero, otra vez, por puro azar, las variaciones aleatorias introducidas en la población pueden haber mejorado a algunos individuos, convirtiéndolos en mejores soluciones del problema, más completas o más eficientes. De nuevo, se seleccionan y copian estos individuos vencedores hacia la siguiente generación con cambios aleatorios mediante el cruce y la mutación, y el proceso se repite. Las expectativas son que la aptitud media de la población se incrementará en cada ronda y por tanto, repitiendo este proceso cientos o miles de rondas, pueden obtenerse soluciones buenas para el problema tratado [39].

1.1.2 ¿Qué son los EDA?

Los EDA son algoritmos heurísticos de optimización que basan su búsqueda, al igual que los algoritmos genéticos, en el carácter estocástico de la misma. De igual forma que los algoritmos genéticos los EDA están basados en poblaciones que evolucionan. Sin embargo a diferencia de los algoritmos genéticos en los EDA la evolución de las poblaciones no se lleva a cabo por medio de los operadores de cruce y mutación. En lugar de ello la nueva población de individuos se muestrea de una distribución de probabilidad, la cual es estimada de la base de datos conteniendo al conjunto de individuos seleccionados de entre los que constituyen la generación anterior [40].

Mientras que en los algoritmos genéticos las interrelaciones entre las variables representando a los individuos se tienen en cuenta de manera implícita, en los EDA dichas interrelaciones se expresan de manera explícita a través de la distribución de probabilidad asociada con los individuos seleccionados en cada generación [40].

La primera población se genera de forma aleatoria y, a partir de esta, se muestrean los nuevos individuos, comenzando por una distribución de probabilidad estimada de la base de datos que contiene únicamente los individuos seleccionados en la generación anterior. La estimación de dicha distribución de probabilidad conjunta asociada a los individuos seleccionados en cada generación es la principal dificultad de esta

aproximación y requiere la adaptación de métodos para el aprendizaje de modelos a partir de datos, que han sido desarrollados por investigadores en el dominio de los modelos gráficos probabilísticos. El aprendizaje es, generalmente, el paso más costoso en términos de cómputo [40].

1.1.3 Estructura general de un EDA

El algoritmo EDA asume que:

$$p(x,t+1) \approx p^s(x,t)$$

Donde $p(x,t+1)$ representa la distribución de los puntos en la generación $t+1$ y $p^s(x,t)$ es la distribución del conjunto seleccionado S en la generación t, o sea, en cada iteración el algoritmo genera puntos que son similares a los mejores puntos hallados hasta el momento [25].

Algoritmo 1 EDA.

Paso 1 $t \leftarrow 1$. Generar N puntos aleatoriamente.

Paso 2 Evaluar los puntos en $f(x)$. Seleccionar $\tilde{N}$ puntos y crear el conjunto seleccionado S.

Paso 3 Estimar la distribución del conjunto seleccionado $p^s(x,t)$ a partir de S.

Paso 4 Generar N nuevos puntos de acuerdo a $p(x,t+1) \approx p^s(x,t)$.

Paso 5 $t \leftarrow t+1$. Si no se cumple el criterio de parada ir al paso 2.

Como se puede ver, al igual que la mayoría de los algoritmos generacionales, se parte de una población inicial con *N* individuos, generada (en la mayoría de los casos) aleatoriamente. En el segundo paso, un número *Ñ* menor que *N* de individuos se selecciona (normalmente aquellos con los mejores valores en cuanto a la función de evaluación) como base de datos para la estimación del modelo en el conjunto *S*. A continuación se estima la distribución del conjunto seleccionado *S*. En el cuarto paso se generan *N* nuevos puntos o individuos, donde sus probabilidades son semejantes a las de la generación anterior. Este proceso a partir del paso 2 se repite hasta que se satisfaga la condición de parada.

1.1.4 Tipos de EDA

Se han propuesto una gran variedad de algoritmos en la literatura que son parte de los EDA, los cuales pueden clasificarse en tres grandes grupos dependiendo de la complejidad del tipo de dependencias entre variables que tienen en cuenta.

1.1.4.1 Sin interdependencias entre variables

Estos EDA se basan únicamente en distribuciones univariantes $p(x_i)$. Esto significa que la estructura en forma de red Bayesiana (o Gausiana si se trabaja en el dominio continuo) es fija y no contiene arcos. En otras palabras, esto significa que todas las variables del individuo se consideran independientes entre sí [17].

Se pueden citar varios ejemplos de EDA pertenecientes a este grupo. En el dominio discreto se encuentran: PBIL (Population Based Incremental Learning) de 1994, UMDA (Univariate Marginal Distribution - Algorithm) de 1998 y cGA (compact Genetic Algorithm) de 1998. En el dominio continuo están: SHCLVND (Stochastic Hill Climbing with Learning by Vectors of Normal Distributions) de 1996, PBILc (Population Based Incremental Learning. Continuous) de 1998 y UMDAc (Univariate Marginal Distribution Algorithm continuous) de 2000 [17].

A continuación se tomará como ejemplo el algoritmo UMDA, del cual se darán sus principales características y se mostrará su algoritmo.

- **UMDA**

El UMDA es considerado como el EDA más simple pues considera que todas las variables son independientes, por lo que el modelo factoriza como un producto de marginales univariados e independientes. La estructura de la red es el grafo totalmente desconectado, por lo que no se requiere de aprendizaje estructural. La distribución en este caso puede escribirse como un producto de distribuciones univariadas de la forma [17]:

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

Algoritmo 2 UMDA.

Paso 1	$S_0 \leftarrow$ Generar M individuos (la población inicial) al azar.
Paso 2	$S_{t-1} \leftarrow$ Seleccionar $N \leq M$ individuos de S_{t-1} de acorde con un método de selección.
Paso 3	Estimar la distribución de probabilidad conjunta a partir de S.
Paso 4	$S_t \leftarrow$ Muestrear M individuos (la nueva población) de $p_t(x)$.
Paso 5	Repetir desde el Paso 2 hasta que se verifique el criterio de parada.

1.1.4.2 Dependencias bivariadas entre variables

Los EDA pertenecientes a este grupo están basados en distribuciones univariantes $p(x_i)$ y también en condicionales de segundo orden $p(x_i \mid x_j)$.

La diferencia más significativa con respecto al grupo anterior es que la estructura de la red Bayesiana (o Gausiana) puede ser diferente, aunque cada una de las variables puede tener como mucho un padre. Esto requiere un paso previo de selección de la mejor estructura que no existía en los anteriores [16].

En este grupo existen diferentes EDA. En el dominio discreto se encuentran: MIMIC (Mutual Information Maximization for Input Clustering) de 1997, COMIT (Combining Optimizers with Mutual Information Trees) de 1997 y BMDA (Bivariate Marginal Distribution Algorithm) de 1998. En el dominio continuo está: MIMICGc (Mutual Information Maximization for Input Clustering Continuous. Gaussian) de 2000 [17].

- **PADA: EDA basado en redes simplemente conectadas**

A partir de 1996 se reportan los primeros resultados donde se utilizan modelos simplemente conectados en optimización evolutiva: grafos totalmente desconectados, árboles y forestas [60] [61] [66] [26], pero no es hasta 1998 en que se formaliza la idea de construir factorizaciones de las distribuciones de los conjuntos seleccionados [29]. A partir de ese momento, las diferentes escuelas que investigaron (e investigan) este problema se han concentrado casi exclusivamente en el uso de redes multiconectadas, debido a la superior capacidad expresiva de estos modelos con respecto a las redes simplemente

conectadas. También se han reportado trabajos centrados en el uso de redes simplemente conectadas en la optimización de funciones con dominio binario [42].

PADA es un EDA que utiliza como modelo de sus distribuciones los modelos simplemente conectados. Su nombre está dado porque así aparece por primera vez en una publicación en idioma inglés: Polytree Approximation Distribution Algorithm [67]. El esquema general de PADA se muestra a continuación.

Algoritmo 3 PADA.

Paso 1 $t \leftarrow 1$. Generar N puntos aleatoriamente.

Paso 2 Evaluar los puntos en $f(x)$. Seleccionar $\tilde{N}$ puntos y crear el conjunto seleccionado S

Paso 3 Estimar la distribución poliárbol $p^{s}_{poliarbol}(x,t)$ a partir de S.

Paso 3.1 Podar el árbol aprendido (opcional).

Paso 4 Generar N nuevos puntos de acuerdo a $p(x,t+1) \approx p^{s}_{poliarbol}(x,t)$.

Paso 5 Elitismo de tamaño k (opcional).

Paso 6 $t \leftarrow t+1$. Si no se cumple el criterio de parada ir al paso 2.

En el paso 1, como en la mayoría de los algoritmos basados en poblaciones, se genera de manera aleatoria la población inicial con la que comenzará el algoritmo, en el segundo paso del algoritmo se seleccionan los mejores individuos, de acuerdo a cualquier criterio de selección escogido entre los métodos existentes. En el tercer paso (probablemente el de mayor importancia) se estima la distribución poliárbol de los individuos del conjunto seleccionado. PADA utiliza distintos algoritmos de aprendizaje de poliárboles a partir de datos, ejemplo de estos algoritmos son PADAt2, PADAt3, PADAp2 y PADAp3, PADAumda, los dos primeros se utilizan para construir árboles, los dos siguientes para construir poliárboles, y el último, pero no menos importante, utiliza el esquema trivial para grafos totalmente desconectados. Otros modelos como las cadenas y forestas de árboles y poliárboles quedan evidentemente incluidos en los casos mencionados.

El paso 3.1 se refiere a la poda o modificación del poliárbol. La idea es sencilla: una vez que se ha aprendido la red Bayesiana, puede resultar conveniente modificar el grafo y las probabilidades. Esta técnica, aunque puede brindar resultados interesantes, ha quedado fuera del alcance de la investigación actual. El cuarto paso del algoritmo es la generación de los nuevos individuos a partir de la simulación de la distribución estimada en el paso anterior. El quinto paso es el elitismo, técnica que permite que los $k(1 \leq k \leq N)$ mejores individuos de una generación se copien directamente en la siguiente. El objetivo principal del elitismo (no el único) es preservar las mejores soluciones encontradas durante la evolución. Cuando PADA no aplica elitismo se generan en el cuarto paso N individuos, en caso contrario se generan $N-k$ individuos. Generalmente, en PADA se utiliza la estrategia del mejor elitismo, la cual consiste en que el número de individuos elitistas coincide con el número de individuos en el conjunto seleccionado $(k=\tilde{N})$ [63] [64]. El elitismo permite disminuir el número de individuos a evaluar y por tanto el costo del algoritmo.

Los algoritmos de aprendizaje comparten en lo esencial un mismo esquema: parten de un grafo sin aristas que va creciendo con la inserción de las aristas de mayor peso, según dos medidas definidas sobre los arcos $\langle X_i, X_j \rangle$: la dependencia marginal $Dep(X_i, X_j)$ y la dependencia global $Dep_g(X_i, X_j)$. El número máximo de aristas que se pueden insertar es $n-1$, restricción que garantiza que la estructura resultante sea simplemente conectada. No todos los algoritmos emplean las dos medidas. La primera medida tiene en cuenta exclusivamente las dos variables dadas, mientras que la segunda mide en cierto modo, el grado de relación de estas variables cuando se conoce el valor de todas y cada una de las demás. Las dos medidas son valores no negativos. Ambas se basan en los conceptos de información mutua marginal y condicional.

Definición 1.1 *La información mutua marginal* $I(X_i, X_j)$ *de las variables aleatorias* X_i *y* X_j *se define como:*

$$I(X_i, X_j) = \sum_{x_i, x_j} p(x_i, x_j) \log_2 \frac{p(x_i, x_j)}{p(x_i)p(x_j)} \geq 0$$

Definición 1.3 *La información mutua condicional* $I(X_i, X_j \mid X_k)$ *de las variables aleatorias* X_i *y* X_j dado X_k *se define como:*

$$I(X_i, X_j \mid X_k) = \sum_{x_i, x_j, x_k} p(x_i, x_j, x_k) \log_2 \frac{p(x_i, x_j, x_k) p(x_k)}{p(x_i, x_k) p(x_j, x_k)} \geq 0$$

Solamente se insertan aquellas aristas cuyos valores $Dep(X_i, X_j)$ y $Dep_g(X_i, X_j)$ superan determinados pesos umbrales (pequeños valores reales positivos: $\in_0$ y $\in_1$ dados como parámetros del algoritmo), y que además no violen las restricciones correspondientes en el modelo en cuestión. Por ejemplo, ninguna arista que cree un ciclo puede insertarse.

En este punto es importante realizar una reflexión sobre la decisión de usar esquemas que sólo utilizan las dos medidas mencionadas. La motivación fundamental es el costo del algoritmo de aprendizaje. El cálculo de las medidas $Dep(X_i, X_j)$ y $Dep_g(X_i, X_j)$ tiene complejidad $O(n^2)$ y $O(n^3)$ sobre el número n de variables respectivamente. Si se tienen en cuenta relaciones de orden superior los requerimientos de espacio y tiempo aumentan exponencialmente con respecto al orden de la relación. Por esta razón es importante poner una cota superior al orden de las relaciones que se consideran. La restricción a la clase de los poliárboles en buena medida garantiza el cumplimiento de este requisito.

A continuación se mostrará el algoritmo PADAt2 para una mejor comprensión de los EDA basado en redes simplemente conectadas, sin embargo, una completa información de este tema y los demás modelos (PADAt3, PADAp2, PADAp3, PADAumda) puede encontrarse en [42].

PADA árbol-caso cuadrático

En el dominio discreto un ejemplo de EDA es el PADA árbol-caso cuadrático. Una distribución árbol puede escribirse como un producto de distribuciones condicionales bivariadas de la forma:

$$p(x) = \prod_{i=1}^{n} p(x_i \mid x_{j(i)})$$

donde $X_{j(i)}$ es la variable designada como padre de X_i en alguna orientación del árbol [42].

Una representación por árbol puede constar de varias ventajas, entre las cuales se encuentran el almacenamiento económico y la confiabilidad en la estimación de los parámetros $p(x_i \mid x_j)$ a partir de datos, debido a que incluso en muestras pequeñas pueden estar especificadas todas las posibles configuraciones de x_i y x_j [42].

El método para la construcción de árboles a partir de datos, se basa en insertar en el grafo las $n-1$ aristas de mayor información mutua marginal $I(X_i, X_j)$, excepto que cree un ciclo o que $Dep(X_i, X_j) = I(X_i, X_j) \leq \in_0$. Se elige arbitrariamente un nodo como raíz del árbol. El algoritmo Arbol1 sólo requiere calcular marginales de primer y segundo orden [42].

Algoritmo 4 ARBOL1.

Paso 1 Comenzar con un grafo sin aristas G.

Paso 2 Calcular $\frac{n(n-1)}{2}$ valores de información mutua marginal entre todos los pares de variables X_i y X_j. Ordenar los valores calculados en orden decreciente.

Paso 3 Adicionar $n-1$ aristas a G siguiendo el orden de la lista excepto que cree un ciclo o que $Dep(X_i, X_j) = I(X_i, X_j) \leq \in_0$.

Paso 4 Seleccionar arbitrariamente un nodo como raíz.

Se ha demostrado que si los datos provienen de una distribución con estructura de árbol, entonces el algoritmo aprende una distribución que coincide con la verdadera. En caso de que los datos provengan de una distribución que no es un árbol, el algoritmo aprende la distribución con estructura de árbol más cercana a la distribución original con respecto a la distancia Kullback-Leibler [42].

Definición 1.3 *La entropía relativa o distancia Kullback-Leibler entre dos distribuciones* p *y* q *se define como:*

$$D(p \,\|\, q) = \sum_{\{x\}} p(x) \log_2 \frac{p(x)}{q(x)} = \left\langle \log_2 \frac{p(X)}{q(X)} \right\rangle$$

donde $\langle . \rangle$ *es el valor esperado de* $\log_2 \frac{p(X)}{q(X)}$ *según* p *. Es no negativa y se hace cero si y sólo si* $p = q$ *.*

- **MIMIC**

Otro ejemplo de EDA discretos pertenecientes a este grupo es MIMIC, que propone realizar la siguiente factorización de la probabilidad [17]:

$$p^{\pi}(x) = p\left(x_{i_1} \mid x_{i_2}\right) \cdot p\left(x_{i_2} \mid x_{i_3}\right) \cdots p\left(x_{i_{n-1}} \mid x_{i_n}\right) \cdot p\left(x_{i_n}\right)$$

MIMIC se basa en buscar la permutación

$$\pi = (i_1, i_2, ..., i_n)$$

que minimiza la divergencia de Kullback-Leibler entre la estimación $p^{\pi}(x)$ y la distribución real $p(x)$.

Algoritmo 5 MIMIC.

Paso 1	Buscar la variable X_{i_n} con menor entropía.
Paso 2	Seleccionar del conjunto de variables no elegidas hasta el momento la variable cuya entropía condicional media con respecto a la variable seleccionada en el paso anterior es mínima.
Paso 3	Repetir el paso anterior hasta el criterio de parada.

1.1.4.3 Dependencias múltiples entre variables

Los EDA pertenecientes a este grupo consideran tanto distribuciones univariantes como condicionales orden dos o superior, y por lo tanto las estructuras de redes Bayesianas o Gausianas no tiene ninguna restricción en el número de arcos que pueden contener [16].

Esta característica requiere una búsqueda exhaustiva de la mejor estructura gráfica probabilística entre todas las posibles, y por lo tanto estos algoritmos son más costosos en tiempo de ejecución que los de los grupos anteriores, aunque también son capaces de aprender modelos que reflejan más fielmente las interrelaciones entre las diferentes variables que forman parte de los individuos [16].

Dentro de este grupo existen distintos EDA. En el dominio discreto se encuentran: EBNA (Estimation of Bayesian Network Algorithm) de 1999, BOA (Bayesian Optimization Algorithm) de 1999, LFDA (Learning Factorized Distribution Algorithm) de 1999 y EcGA (Extended compact Genetic Algorithm) de 1999 [17].

En el dominio continuo están: EMNA $_{\text{global}}$ (Estimation of Multivariate Normal Algorithm. Global) de 2001, EMNA $_{a}$ (Estimation of Multivariate Normal Algorithm. Adaptive) de 2001, EMNA $_{i}$ (Estimation of Multivariate Normal Algorithm incremental) de 2001, EMNA $_{\text{ee}}$ (Estimation of Multivariate Normal Algorithm, Edge Exclusion) de 2000, EGNA $_{\text{BIC}}$ (Estimation of Gaussian Network Algorithm, BIC) de 2001 y EGNA $_{\text{BGe}}$ (Estimation of Gaussian Network Algorithm, BGe) de 2000 [17].

- **EBNA**

En el dominio discreto, como un ejemplo de EDA pertenecientes a este grupo es el conocido como EBNA. En este se define un score o medida basada en la máxima verosimilitud penalizada conocida como BIC (Bayesian Information Criterion) que mide la idoneidad de una estructura para representar las interdependencias entre los individuos [16].

Utilizando esta medida, se busca la red Bayesiana que lo maximiza. Una vez definida la estructura, la factorización de la probabilidad se realiza de la siguiente forma:

$$p(x) = \prod_{i=1}^{n} p(x_i \mid pa(x_i))$$

donde $pa(x_i)$ es el conjunto de padres de la variable x_i en la Red Bayesiana [16].

Algoritmo 6 EBNA.

Paso 1 $M_0 \leftarrow (S_0, \Theta_0)$.

Paso 2 $S_0 \leftarrow$ Muestreando M_0, obtener N individuos.

Paso 3 $S_{t-1} \leftarrow$ Seleccionar individuos de S_{t-1}.

Paso 4 Buscar la estructura que maximice BIC S.

Paso 5 Calcular $\left\{ \Theta_{ijk}^{l} = \frac{N_{ijk}^{l} + 1}{N_{ij}^{l} + r_i} \right\}$ usando S_{t-1} como conjunto de datos.

Paso 6 $S_l \leftarrow$ Utilizando PLS muestrear M_l para obtener N individuos.

Paso 7 Repetir desde el Paso 2 hasta que se verifique el criterio de parada.

M_0 es un DAG (Directed Acyclic Graph) sin ningún arco $p(X_i = x_i) = \frac{1}{r_i}, i = 1,..., n$.

- **EGNA**

Un ejemplo del dominio continuo es EGNA, que sigue una aproximación similar a EBNA. En este la primera generación se comienza con una estructura sin arcos y el resto de las generaciones comienzan con el modelo obtenido en la generación previa [17].

1.2 Optimización mediante Colonia de Hormigas

Es una metaheurística relativamente joven, que como su nombre lo indica tiene su precedente en el comportamiento de las hormigas naturales, pues toma muchos elementos de estos insectos para implementar estrategias de búsquedas eficientes.

Esta técnica constituye otro enfoque a la resolución de problemas de optimización combinatoria. La idea central en ACO es que un número elevado de agentes artificiales simples son capaces de construir buenas soluciones a problemas de optimización combinatoria difíciles gracias a la utilización de comunicaciones de bajo nivel. Las hormigas reales cooperan en su búsqueda de alimento depositando en el suelo trazas de una sustancia química (feromonas). Una colonia de hormigas artificiales simula este comportamiento. Las hormigas artificiales cooperan mediante la utilización de una memoria común que sería el equivalente a la feromona depositada por las hormigas reales. La feromona artificial se acumula en tiempo de ejecución a través de un mecanismo de aprendizaje. Las hormigas artificiales se implementan como procesadores en paralelo cuya misión es construir soluciones al problema dado utilizando un procedimiento que está movido por una combinación de la feromona artificial, los datos del problema y una función heurística utilizada para evaluar los sucesivos pasos de la construcción. La optimización por colonia de hormigas se ha revelado especialmente útil para la resolución de problemas de enrutamiento [1].

1.2.1 Antecedentes

Las hormigas son insectos sociales que viven en colonias y que por su colaboración mutua pueden mostrar comportamientos muy complejos. Estas con facilidad pueden encontrar alimento y con gran rapidez pueden organizar el camino de ida y retorno desde el hormiguero hasta la fuente, encontrando por lo general, el camino más corto entre ambos lugares, esto es algo significativo considerando que algunos tipos de hormigas carecen del sentido de la visión, en otras palabras, son ciegas, lo cual las hace prescindir de la orientación mediante puntos de referencia. Las hormigas son insectos con instintos básicos lo cual las hace candidatos perfectos para una implementación computacional [15].

Este mecanismo de búsqueda logra ser tan efectivo dado que mientras se mueven en busca de alimento depositan una sustancia química denominada feromona, sustancia que pueden oler y a la cual tienen cierta tendencia a seguir. Experimentos realizados han demostrado que las hormigas prefieren de manera probabilística los caminos marcados con una concentración superior de feromona [43]. Si la hormiga se ve en la necesidad de elegir entre varios caminos, esta escogerá el de mayor concentración de feromona, en caso de no existir, la elección se hace básicamente de manera aleatoria [15] [53].

Cabe señalar que constantemente las hormigas van dejando un rastro de feromona. Los caminos con mayor concentración de esta sustancia resultan más prometedores por que las hormigas que lo transitan tienden a encontrar el alimento más rápido y regresar al hormiguero en menor tiempo, por tanto recibirá cantidades de feromona superiores. Este proceso es complementado por la acción del medio como la evaporación de la feromona, lo cual provocará que los caminos menos usados por las hormigas pierdan interés [15] [56] [57].

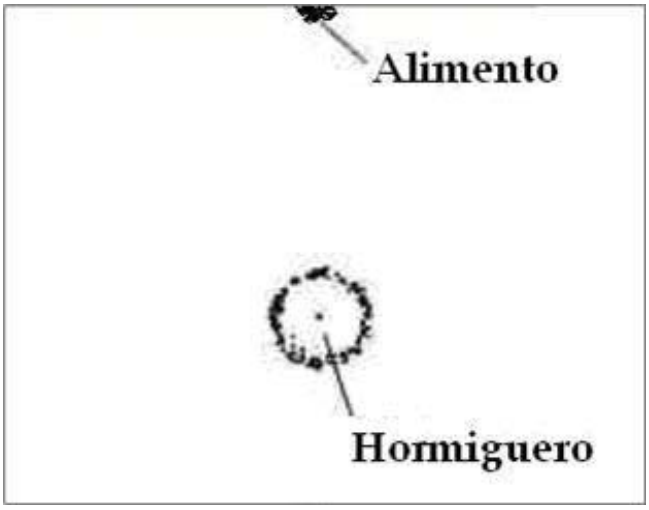

Figura 1.1. Momento en que las hormigas salen del hormiguero en el tiempo t_0 .

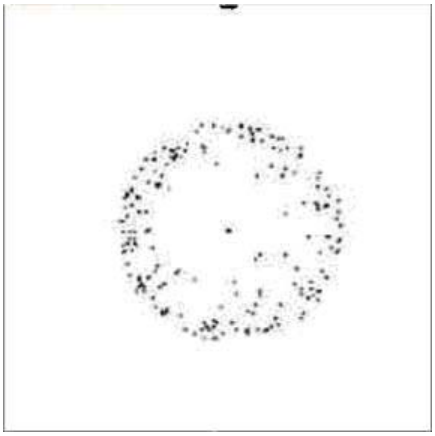

Figura 1.2. Hormigas en el tiempo t_0+1.

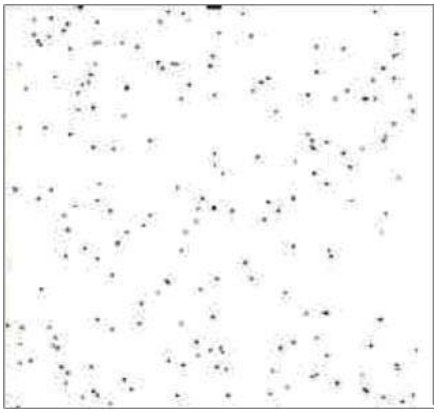

Figura 1.3. Hormigas en el tiempo t_0+2.

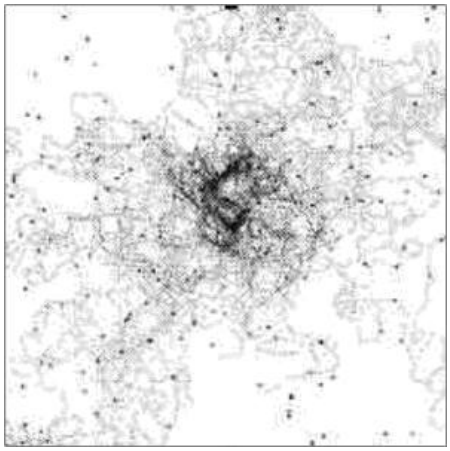

Figura 1.4. Hormigas en el tiempo t_0+3.

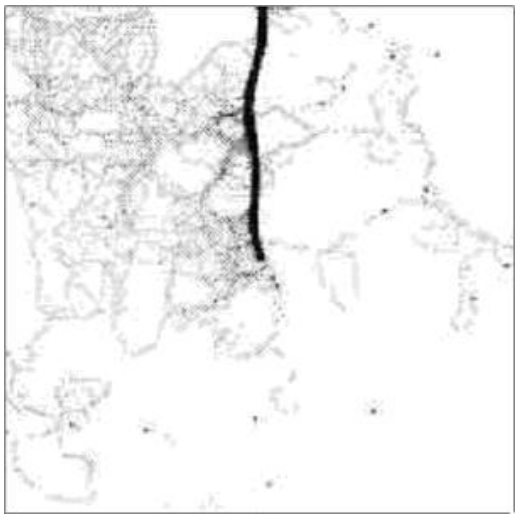

Figura 1.5. Hormigas en el tiempo t_0+4.

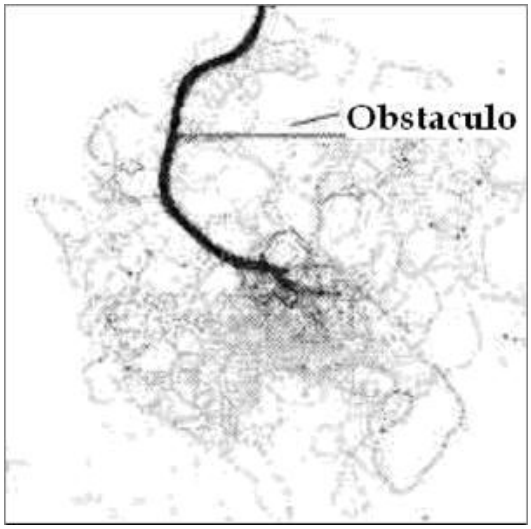

Figura 1.6. Hormigas en el tiempo t_0+5.

En las Figuras 1, 2, 3, 4, 5 y 6 se representan una serie de secuencias del comportamiento del hormiguero en el tiempo, desde que salen en búsqueda del alimento hasta que crean un camino de regreso [54].

1.2.2 ¿Qué son los ACO?

Los algoritmos que pertenecen a los ACO se basan en una colonia de hormigas artificiales, las cuales son agentes computacionales simples que trabajan de manera cooperativa y se comunican mediante rastros de feromona. En estos algoritmos cada hormiga construye una solución al problema, recorriendo un grafo de construcción. Cada arista del grafo, representa los posibles caminos que puede recorrer la hormiga y en cada arista existe información que rige su movimiento [15] [56] [57]:

1. Información heurística: mide la preferencia heurística de moverse desde el nodo *r* hasta el nodo *s*, o sea de recorrer la arista a_{rs} . Se denota por η_{rs} . Las hormigas no modifican esta información durante la ejecución del algoritmo. Esta puede ser calculada de esta manera:

$$\eta_{rs} = \frac{1}{P_{rs}}$$

2. Información de los rastros artificiales de feromona: mide la deseabilidad aprendida del movimiento de r a s . Imita la feromona real. Se denota por τ_{rs} . En las diferentes variantes de algoritmos ACO hacen un uso diferente de esta información pues es en ella que se basa el camino a soluciones satisfactorias.

De la forma que se asuman estos parámetros y cómo sean modificados, es que se basan la diferencias entre los tipos de de algoritmos de ACO.

El modo de operación básico de un algoritmo basado en colonias de hormigas es como sigue: las m hormigas artificiales de la colonia se mueven, concurrentemente y de manera asíncrona, a través de los estados adyacentes del problema, que puede representarse en forma de grafo con pesos.

Este movimiento se realiza siguiendo una regla de transición que está basada en la información local disponible en las componentes (nodos). Esta información local incluye la información heurística y memorística (rastros de feromona) para guiar la búsqueda. Al moverse por el grafo de construcción, las hormigas construyen incrementalmente soluciones [15].

Opcionalmente, las hormigas pueden depositar feromona cada vez que crucen un arco (conexión) mientras que construyen la solución (actualización) en línea paso a paso de los rastros de feromona. Una vez que cada hormiga ha generado una solución, se evalúa esta y puede depositar una cantidad de feromona que es función de la calidad de su solución (actualización en línea a de los rastros de feromona). Esta información guiará la búsqueda de las otras hormigas de la colonia en el futuro.

Además, el modo de operación genérico de estos algoritmos incluye dos procedimientos adicionales, la evaporación de los rastros de feromona y las acciones del demonio. La evaporación de feromona la lleva a cabo el entorno y se usa como un mecanismo que evita el estancamiento en la búsqueda y permite que las hormigas busquen y exploren nuevas regiones del espacio. Las acciones del demonio son opcionales para implementar tareas desde una perspectiva global que no pueden llevar a cabo las hormigas por la perspectiva local que ofrecen.

Ejemplos son: observar la calidad de todas las soluciones generadas y depositar una nueva cantidad de feromona adicional sólo en las transiciones componentes asociadas a algunas soluciones consideradas como buenas, o aplicar un procedimiento de búsqueda local a las soluciones generadas por las hormigas antes de actualizar los rastros de feromona. En ambos casos, el demonio reemplaza la actualización en línea a posteriori de feromona y el proceso pasa a llamarse actualización fuera de línea de rastros de feromona.

1.2.3 Estructura general de un ACO

A continuación se representa la estructura básica de un ACO [15] [44], a groso modo se definirá las diferentes acciones que realizan de modo general:

Algoritmo 7 ACO.

```
Procedimiento Metaheurístico ()
        Inicialización de parámetros
        mientras (un criterio de parada)
                Programación de actividades
                        Generación de Hormigas ()
                        Evaporación de Feromona ()
```

```
                    Acciones del demonio () opcional
                fin Programación de actividades
        fin mientras
Fin de Metaheurístico ()

Procedimiento Generación de Hormigas y actividad ()
        Repetir en paralelo desde k=1 hasta m (numero hormigas)
                Nueva Hormiga (k)
        fin Repetir en paralelo
Fin Generación de Hormigas y actividad ()

Procedimiento Nueva Hormiga (id Hormiga)
        inicializa hormiga(id Hormiga)
        L = actualiza memoria hormiga()
        mientras (estado actual 6= estado objetivo)
                P = calcular probabilidades de transición(A,L,)
                siguiente estado = aplicar política decisión(P,)
                mover al siguiente estado(siguiente estado)
                si (actualizacion feromona en linea paso a paso)
                        Depositar feromona en el arco visitado ()
                fin si
                L = actualizar estado interno()
        fin mientras
        si (actualización feromona en línea a posteriori)
                para cada arco visitado
                        depositar feromona en el arco visitado()
                Fin Para cada arco
        Fin si
liberar recursos hormiga(id Hormiga)
Fin Procedimiento Nueva Hormiga ()
```

El primer paso incluye la inicialización del los valores de los parámetros que se tienen en consideración en el algoritmo. Entre otros, se deben fijar el rastro inicial de feromona asociado a cada transición τ_0, que es un valor positivo pequeño, normalmente el mismo para todas las componentes conexiones, el número de hormigas en la colonia, m, y los pesos que definen la proporción en la que afectarán la información heurística y memorística en la regla de transición probabilística. El procedimiento principal de la metaheurística ACO controla, mediante el constructor Programación_de_Actividades, la planificación de estas tres componentes:

(i). La generación y puesta en funcionamiento de las hormigas artificiales.

(ii). La evaporación de feromona.

(iii). Las acciones del demonio.

La implementación de este constructor determinará la sincronía existente entre cada una de las tres componentes. Mientras que la aplicación a problemas clásicos NP-duros (no distribuidos), normalmente usa una planificación secuencial, en problemas distribuidos como el enrutamiento en redes, el paralelismo puede ser explotado de manera sencilla y eficiente.

Varias componentes son, o bien opcionales, como las acciones del demonio, o bien dependientes estrictamente del algoritmo ACO especifico, por ejemplo cuándo y cómo se deposita la feromona. Generalmente, la actualización en línea paso a paso de los rastros de feromona y la actualización en línea a posteriori de los rastros de feromona son mutuamente excluyentes y no suelen estar presentes a la vez ni faltar ambas al mismo tiempo (si las dos faltan, el demonio suele actualizar los rastros de feromona).

Por otro lado, el procedimiento Actualiza memoria hormiga () se encarga de especificar el estado inicial desde el que la hormiga comienza su camino y, además almacenar la componente correspondiente en la memoria de la hormiga L. La decisión sobre cuál será dicho nodo depende del algoritmo especifico (puede ser una elección aleatoria o una fija para toda la colonia, o una elección aleatoria o fija para cada hormiga, etc.).

Finalmente, los procedimientos Calcular probabilidades de transición y aplicar política decisión tienen en consideración el estado actual de la hormiga, los valores actuales de la feromona visibles en dicho nodo y las restricciones del problema Ω para establecer el proceso de transición probabilística hacia otros estados válidos.

1.2.4 Tipos de ACO

Existen diversos algoritmos que siguen la metaheurística ACO. Entre los disponibles para problemas de optimización combinatoria NP-duros, se encuentran el Sistema de Hormigas (AS, según sus siglas en inglés), el Sistema de Colonia de Hormigas (ASC, según sus siglas en inglés), el Sistema de Hormigas Max-Min (MMAS, según sus siglas en inglés), el AS con ordenación, y el Sistema de la Mejor-Peor Hormiga (BWAS, según sus siglas en inglés).

1.2.4.1 El Sistema de Hormigas

Fue el primer algoritmo de esta metaheurística, desarrollado por Dorigo, Maniezzo y Colorni en 1991. Se presentaron 3 variantes distintas: AS-densidad, AS-cantidad y AS-ciclo, que se diferenciaban en la manera en que se actualizaban los rastros de feromona. En los dos primeros, las hormigas depositaban feromona mientras que construían sus soluciones (esto es, aplicaban una actualización en-línea paso a paso de feromona), con la diferencia de que la cantidad de feromona depositada en el AS-densidad es constante, mientras que la depositada en AS-cantidad dependía directamente de la deseabilidad heurística de la transición η_{rs} . Por último, en AS-ciclo, la deposición de feromona se lleva a cabo una vez que la solución está completa (actualización en línea a posteriori de feromona). Esta última variante era la que obtenía los mejores resultados y es por tanto la que se conoce como AS en la literatura) [15].

El AS se caracteriza por el hecho de que la actualización de feromona se realiza una vez que todas las hormigas han completado sus soluciones, y se lleva a cabo como sigue: primero, todos los rastros de feromona se reducen en un factor constante, implementándose de esta manera la evaporación de feromona [55].

A continuación cada hormiga de la colonia deposita una cantidad de feromona que es función de la calidad de su solución. Inicialmente, el AS no usaba ninguna acción del demonio, pero es relativamente fácil, por ejemplo, añadir un procedimiento de búsqueda local para refinar las soluciones generadas por las hormigas.

Los creadores del AS también propusieron una versión extendida del algoritmo que normalmente mejoraba los resultados obtenidos, llamada AS elitista. En el AS elitista, una vez que las hormigas han depositado feromona en las conexiones asociadas a sus respectivas soluciones, el demonio realiza un

depósito adicional de feromona en las aristas que pertenecen a la mejor solución encontrada hasta el momento en el proceso de búsqueda (a esta se le denominará mejor solución global de aquí en adelante). La cantidad de feromona depositada, que depende de la calidad de la mejor solución global, se incrementa en un factor e, que se corresponde con el número de hormigas elitistas [59].

1.2.4.2 Sistema de Colonia de Hormigas

Es uno de los primeros sucesores del AS que introduce tres modificaciones importantes con respecto a dicho algoritmo:

1. El ASC usa una regla de transición distinta, denominada regla proporcional pseudo-aleatoria. Sea k una hormiga situada en el nodo r, q_0 un parámetro y q un valor aleatorio en [0,1], el siguiente nodo s se elige aleatoriamente mediante la siguiente distribución de probabilidad [8]:

   Si $q \leq q_0$:

$$P_{rs}^{k} = \begin{cases} 1, & \text{si s} = \arg\,\text{máx}\,\left\{\tau_{\text{ru}} \cdot \eta_{ru}^{\beta}\right\} \\ 0, & \text{en otro caso} \end{cases}$$

   Si no $(q > q_0)$:

$$P_{rs}^{k} = \begin{cases} \dfrac{\left[\tau_{rs}\right]^{\alpha} \cdot \left[\eta_{rs}\right]^{\beta}}{\sum_{u \in N^{k}} \left[\tau_{rs}\right]^{\alpha} \cdot \left[\eta_{rs}\right]^{\beta}}, & \text{si s} \in \text{N}_{\text{k}}(\tau) \\ 0, & \text{en otro caso} \end{cases}$$

Como puede observarse, la regla tiene una doble intención: cuando $q \leq q_0$, explota el conocimiento disponible, eligiendo la mejor opción con respecto a la información heurística y los rastros de feromona. Sin embargo, si $q > q_0$ se aplica una exploración controlada, tal como se hacia en el AS. En resumen, la regla establece un compromiso entre la exploración de nuevas conexiones y la explotación de la información disponible en ese momento [58].

2. Sólo el demonio (y no las hormigas individualmente) actualiza la feromona, es decir, se realiza una actualización de feromona fuera de línea de los rastros. Para llevarla a cabo, el ACS sólo considera una hormiga concreta, la que generó la mejor solución global, Smejor-global (aunque en algunos trabajos iniciales se consideraba también una actualización basada en la mejor hormiga de la iteración, en ACO casi siempre se aplica la actualización por medio de la mejor global) [58].

3. La actualización de la feromona se hace, evaporando primero los rastros de feromona en todas las conexiones utilizadas por la mejor hormiga global (es importante recalcar que, en el ACS, la evaporación de feromona sólo se aplica a las conexiones de la solución, que es también la usada para depositar feromona) tal como sigue:

$$\tau_{rs} \leftarrow (1-\rho)\cdot \tau_{rs}, \forall a_{rs} \in S_{mejor-global}$$

A continuación, el demonio deposita feromona usando la regla:

$$\tau_{rs} \leftarrow \tau_{rs} + \rho \cdot f\left(C\left(S_{mejor-global}\right)\right), \forall a_{rs} \in S_{mejor-global}$$

Adicionalmente, el demonio puede aplicar un algoritmo de búsqueda local para mejorar las soluciones de las hormigas antes de actualizar los rastros de feromona [58].

4. Las hormigas aplican una actualización en línea paso a paso de los rastros de feromona que favorece la generación de soluciones distintas a las ya encontradas. Cada vez que una hormiga viaja por una arista a_{rs}, aplica la regla:

$$\tau_{rs} \leftarrow (1-\varphi)\cdot \tau_{rs} + \varphi \cdot \tau_0$$

donde $\varphi \in (0,1]$ es un segundo parámetro de decremento de feromona. Como puede verse, la regla de actualización en línea paso a paso incluye tanto la evaporación de feromona como la deposición de la misma; pues la cantidad de feromona depositada es muy pequeña (τ_0 es el valor del rastro de feromona inicial y se escoge de tal manera que en la práctica, se corresponda con el limite menor de rastro de feromona, esto es, con la elección de las reglas de actualización de feromona del ACS ningún rastro de feromona puede caer por debajo de τ_0), la aplicación de esta regla hace que los rastros de feromona entre las conexiones recorridas por las hormigas disminuyan. Así, esto lleva a una técnica de exploración adicional del ACS ya que las conexiones atravesadas por un gran número de hormigas son cada vez menos atractivas para el resto de hormigas que las recorren en la iteración actual, lo que ayuda claramente a que no todas las hormigas sigan el mismo camino [15] [58].

1.2.4.3 Sistema Mix-Max hormiga

El MMAS, desarrollado por Stutzle y Hoos en 1996, es una de las extensiones del AS que mejor rendimiento muestran. Extiende el AS en los siguientes aspectos [15] [45] [46]:

1. Se aplica una actualización de los rastros de feromona fuera de línea, de manera similar a como se hace en el ACS. Después de que todas las hormigas hayan construido su solución cada rastro de feromona sufre una evaporación:

$$\tau_{rs} \leftarrow (1-\rho)\cdot \tau_{rs}$$

y a continuación la feromona se deposita siguiendo la siguiente fórmula:

$$\tau_{rs} \leftarrow \tau_{rs} + f\left(C\left(S_{mejor}\right)\right), \forall a_{rs} \in S_{mejor}$$

La mejor hormiga a la que se le permite añadir feromona puede ser la que tiene una solución mejor de la iteración o la solución mejor global. Los resultados experimentales demuestran que el mejor rendimiento se obtiene incrementando gradualmente la frecuencia de escoger la mejor global para la actualización de feromona Además, en el MMAS las soluciones que ofrecen las hormigas suelen ser mejoradas usando optimizadores locales antes de la actualización de feromona.

2. Los valores posibles para los rastros de feromona están limitados al rango $[\tau_{\min}, \tau_{\max}]$. Por lo tanto, la probabilidad de un estancamiento del algoritmo disminuye al darle a cada conexión existente una probabilidad, aunque bastante pequeña, de ser escogida. En la práctica, existen heurísticas para fijar los valores de $\tau_{\min}$ y $\tau_{\max}$. Se puede ver que, a causa de la evaporación de la feromona, el nivel máximo de feromona en los rastros está limitado a:

$$\tau^*_{\max} = \frac{1}{(\rho \cdot C(S^*))}$$

donde S^* solución óptima. Basado en este resultado, la mejor solución global puede usarse para estimar $\tau_{\max}$ sustituyendo S^* en la ecuación de $\tau^*_{\max}$ por $S_{mejor-global}$. Para $\tau_{\min}$ normalmente solo es necesario escoger su valor de tal manera que sea un factor constante menor que $\tau_{\max}$. Para poder incrementar la exploración de nuevas soluciones, el MMAS utiliza en ocasiones re-inicializaciones de los rastros de feromona.

3. En vez de inicializar los rastros de feromona a una cantidad pequeña, el MMAS los inicializa a una estimación del máximo permitido para un rastro (la estimación puede obtenerse generando una solución S' con una heurística voraz y reemplazando dicha solución S' en la ecuación de $\tau^*_{\max}$). Esto lleva a una componente adicional de diversificación en el algoritmo, pues al comienzo las diferencias relativas entre los rastros de feromona no serán muy acusadas, lo que no ocurre cuando los rastros de feromona se inicializan a un valor muy pequeño.

1.3 Relaciones entre los EDA y los ACO

Existen similitudes entre el modo de operación de los algoritmos de ACO y los algoritmos evolutivos como el uso de una población de individuos que codifican soluciones al problema y que son generadas de manera estocástica.

Un algoritmo EDA conocido es el Aprendizaje Incremental Basado en Poblaciones (”Population-Based Incremental learning”) o PBIL, que es además el más parecido a los algoritmos de ACO, más concretamente al ACS. PBIL trabaja con un vector de probabilidades $P = (p_1,..., p_n)$ de dimensión n igual al número de variables del problema, el cual codifica una distribución de probabilidad que representa un prototipo de buenas soluciones y que se usa para generar una población de posibles soluciones (vectores binarios) en cada iteración [15].

Este vector de probabilidades sufre una adaptación durante la ejecución del algoritmo. En el modelo básico del PBIL, se actualiza dependiendo de la composición de la mejor solución generada en la iteración actual usando una regla de actualización similar a la regla de actualización de feromona fuera de línea del ACS. Por otro lado, las componentes de P también sufren mutaciones aleatorias para evitar la posibilidad de una convergencia prematura [15].

Por último, se debe recordar que las similitudes entre ACO y la computación evolutiva EDA han motivado la integración de ciertos aspectos de los últimos en los algoritmos de ACO para mejorar su rendimiento. Este es el caso en particular del SMPH (Sistema Mejor-Peor Hormiga); este incorpora varios conceptos de los EDA, principalmente de PBIL, como la mutación de los rastros de feromona y la penalización (evaporación adicional de feromona) por la peor solución encontrada en la iteración actual [15].

CAPÍTULO 2: PROBLEMAS TRATADOS

El presente capítulo muestra los problemas que serán tratados con las metaheurísticas anteriormente expuestas. En la sección 2.1 se dará la definición del problema de clasificación automática no supervisada y en la sección 2.2 la del problema de Coloración Robusta. Por último en la sección 2.3 se abordará cómo serán resueltos estos dos problemas.

2.1 Clasificación Automática no Supervisada

El problema de clasificación de datos consiste en agrupar en clases o grupos un conjunto de elementos descritos por un conjunto de variables, de modo que exista similitud entre los elementos de cada clase y estas a su ves se encuentren bien diferenciadas entre si. La medida de la proximidad se basa en las propiedades que describen los elementos y se define usualmente como la proximidad en un espacio multidimensional [47].

La búsqueda de una solución óptima según algún criterio puede resultar computacionalmente muy costosa en dependencia de las dimensiones del problema, por lo que además de los métodos usuales de particionamiento, se ha introducido en los últimos años métodos de optimización estocástica tales como algoritmos genéticos, recocido simulado, búsqueda tabú, y colonia de hormigas [47].

Los procesos de clasificación generalmente se presentan con dos enfoques diferentes. El primer enfoque trata como a partir de conjuntos de observaciones, establecer la existencia de clases o agrupaciones en estas; mientras que en un segundo enfoque se parte de un número conocido de clases y el propósito es establecer una regla con la cual podamos clasificar una nueva observación en una de las clases existentes. A partir de este análisis, son identificados tres tipos [48]:

- Clasificación con aprendizaje (o supervisada): se conoce que un universo de objetos se agrupan en un número dado de clases de las cuales se tiene, de cada una, una muestra (no todos) de entes que se sabe pertenecen a ella.

- Clasificación con aprendizaje parcial (o semisupervisada): es análogo al supervisado, excepto en que hay al menos una clase de objetos de la que no se tiene una muestra.

- Clasificación sin aprendizaje (o no supervisada): no se conoce cómo se agrupan las entidades, aunque es necesario determinar el número de entidades que se quieren establecer.

2.1.1 Problema de Clasificación de Datos

Sea C un conjunto de elementos denotados por x_i, $i = 1,...n$ sobre los que se han medido las variables cuantitativas $x_i(j), j = 1,...m.$ Se denota por la distancia euclidiana utilizada para medir la proximidad entre dos elementos de R^m. Se quiere obtener una partición $P = (C_1, C_2, ..., C_k)$ de C en k clases, bien separadas entre sí y lo más homogéneas posible.

Este problema puede ser resuelto minimizando la inercia inter-clases:

$$\min W(P) = \frac{1}{n} \sum_{i=1}^{k} \sum_{\substack{j \\ x_i \in C_i}} d(x_j, g_i)$$

donde:

$$g_i = \frac{1}{C_i} \sum_{\substack{j \\ x_i \in C_i}}^{k} x_j$$

siendo g_i el centro de masa de la clase [47].

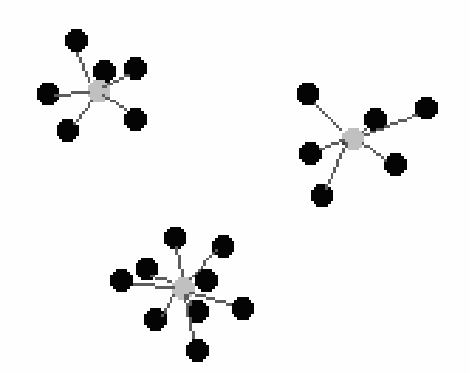

Figura 2.1. Representación del problema particionando 3 clases.

2.2 Coloración Robusta

El problema de colorear los vértices de un grafo consiste en asignar un color a cada vértice de forma que cualquier par de vértices adyacentes no tengan el mismo color. El Problema de Coloración Robusta (PCR) surge a principios del nuevo milenio como una ampliación del Problema de Coloración Mínima, y como respuesta a las inquietudes de muchos especialistas de la rama que buscaban conocer cuan estable podría ser la solución para un problema determinado al que se le aplicara coloración mínima, tomando en cuenta que este se puede aplicar a disímiles situaciones resultando muy efectivo su desenvolvimiento en la solución de dichas situaciones [36].

2.2.1 Definición de Coloración de Grafos

Dado un grafo $G = (V, E)$ con $|V| = n$ n, se define una función de coloración como una aplicación:

$$C : \{1,...n\} \longrightarrow \{1,2,...\}$$

que identifica a C_i como el color del vértice $i \in V$ de forma que dos vértices adyacentes $(i, j) \in E$ no tienen el mismo color, es decir $C_i \neq C_j$.

Dada una función de coloración C de un grafo G, al conjunto de vértices que tienen asignado el mismo color se le llama clase de color:

$$V_c(k) = \{i \in V \mid \mathrm{C_i} = k\} \forall k \in \{1,2,...\}$$

Una k-coloración es una función de coloración que no utiliza más de k colores:

$$C^k : \{1,...n\} \longrightarrow \{1,2,...,k\}$$

Un grafo es k-coloreable si admite una k-coloración. Al mínimo valor k tal que G es k-coloreable se le llama el número cromático del grafo y se denota por $X(G)$ [36].

2.2.2 Coloración Mínima

Dado un grafo G, el Problema de Coloración Mínima (PCM) busca una función de coloración que no utilice más de $X(G)$ colores. En 1972 Richard Karp demostró que el problema de coloración mínima de grafos es NP-Completo, en 1973 Stockmeyer y en 1976 Garey, Johnson y Stockmeyer reforzaron el resultado de Karp demostrando que el problema de colorear un grafo con k colores es NP-Completo para cualquier k mayor o igual a 3 siendo k el número cromático [36].

- **Ejemplo de cómo aplicar la Coloración Mínima: Coloración Mínima de un Mapa**

El ejemplo consiste en determinar el menor número de colores que permite pintar un mapa geográfico de tal manera que dos regiones con una frontera común tengan colores diferentes. Al mapa se le puede asociar un grafo, que será un grafo planar, cuyos vértices son cada una de las regiones del mapa y habrá una arista entre dos vértices si las regiones correspondientes tienen una frontera común. Una coloración de los vértices es equivalente a una coloración de las regiones. El problema se reduce a determinar el número cromático del grafo asociado al mapa. El problema de los cuatro colores está resuelto y es equivalente a afirmar que el número cromático de cualquier grafo planar es siempre menor o igual que cuatro [36].

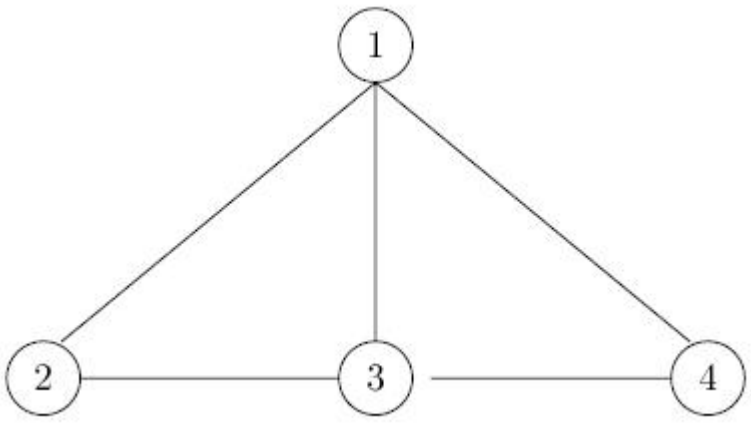

Figura 2.2. Representación del grafo del ejemplo de Coloración Mínima en un mapa.

2.2.3 Problema de Coloración Robusta

Algunos problemas de planificación del tiempo o de recursos se pueden plantear como problemas de coloración de los vértices de un grafo frecuentemente tratando de minimizar el número de colores a utilizar.

Los vértices representan los elementos a programar y cada arista identifica la incompatibilidad entre los elementos correspondientes. Esta incompatibilidad representa el conflicto entre recursos cuando no puedan ser compartidos los elementos a programar.

Una coloración válida identifica una asignación de recursos compatibles. En estos problemas es común que el objetivo sea utilizar el máximo número de colores, de tal manera que a cada par de elementos a programar que no pueden compartir el mismo recurso se les asigne un color diferente. Cada clase de color contendrá los elementos entre los que no hay conflicto y que pueden compartir el recurso en cuestión.

En algunas circunstancias, el recurso a minimizar con el planteamiento de máximo número de colores no es crítico, sino que interesa que una solución al problema sea estable, en el sentido de que al añadir o cambiar aristas al grafo la coloración continúe siendo válida. Estas consideraciones muestran que el problema de coloración mínima es un modelo muy restrictivo para este tipo de problemas.

Se presenta un nuevo problema de coloración de grafos que contempla las extensiones mencionadas. En este problema cada planificación se haría teniendo en cuenta un cierto número de recursos sabiendo que existen programaciones alternativas coloraciones del grafo asociado más flexibles y que puedan ser valoradas con otros criterios [36].

- **Ejemplo de Coloración Robusta**

Se quiere pintar el mapa cuyo grafo asociado se ha representado en la Figura 2.3.

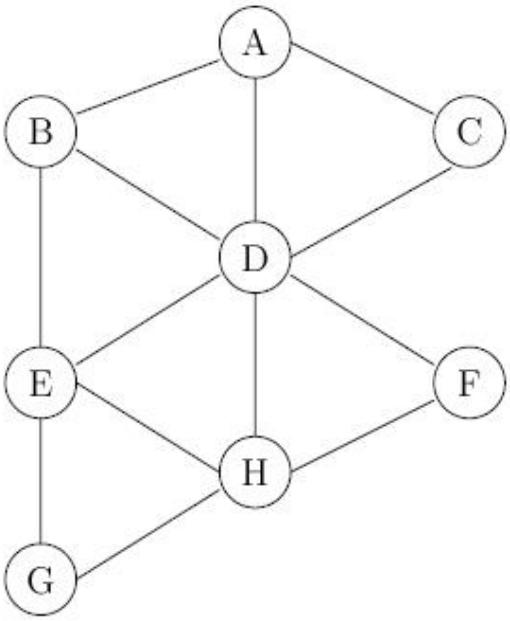

Figura 2.3. Representación del grafo del ejemplo de PCR de un mapa.

Sean las 4-coloraciones representadas en el Cuadro 1 que dan lugar a las clases de color $V_{C_1^4}(1)=\{D\}$, $V_{C_1^4}(2)=\{B,C,F,G\}$, $V_{C_1^4}(3)=\{A,H\}$, $V_{C_1^4}(4)=\{E\}$ y $V_{C_2^4}(1)=\{D,G\}$, $V_{C_2^4}(2)=\{B,F\}$, $V_{C_2^4}(3)=\{A,E\}$, $V_{C_2^4}(4)=\{C,H\}$ respectivamente.

Con la coloración C_2^4 las clases de color son homogéneas en lo que se refiere al número de elementos de las mismas, esto haría preferible a la coloración C_2^4 en lugar de la C_1^4 [42].

Tabla 2.1. Coloraciones del ejemplo de PCR de un mapa

	A	B	C	D	E	F	G	H
c_1^4	3	2	2	1	4	2	2	3
c_2^4	3	2	4	1	3	2	1	4

2.2.3.1 Modelo Matemático

Variables:

1. $G(n,a)$: Grafo a colorear.
2. n: Cantidad de Nodos de G.
3. a: Aristas de G.
4. c: Cantidad de Colores.
5. A: Matriz de Adyacencia (cuadrada de orden n) asociada a G. Contiene las probabilidades de que exista una rista entre dos nodos. Cuando toma valor 0 no existe y cuando toma valor 1 sí.
6. $A_{ij} \in \Re : i, j = \{1..n\}; i, j \in \mathrm{N}$.
7. C: Vector (de orden n) de colores que pueden adquirir los nodos de G.
8. $C_k \in \mathrm{N}; k = \{1..n\}; k \in \mathrm{N}$.
9. $K(i,j) = \{0, siC_i \neq C_j\}$ | 1, en caso contrario.

Función a Minimizar:

$$F(A,c) = \sum_{i=1, j=1}^{n} K(i,j).A_{ij}$$

2.3 Resolución de los Problemas Mediante las Metaheurísticas EDA y ACO

2.3.1 Clasificación Automática no Supervisada mediante ACO

Al problema se le ha dado solución a través del algoritmo ACS. Para aplicar este, se necesitan un grupo de k hormigas para obtener una posible partición del conjunto C en k clases. Se utilizó por tanto en cada ciclo M grupos de k hormigas que generan simultáneamente M posibles particiones de C en k clases.

La forma en que cada grupo de hormigas genera una partición es descrita a continuación:

Inicialmente cada hormiga selecciona aleatoriamente un elemento de C de los que no han sido seleccionados hasta el momento.

Teniendo en cuenta que en la partición óptima no todas las clases deben tener la misma cantidad de elementos, las hormigas van incorporando elementos a su clase en un orden aleatorio, que permite la repetición.

Cuando una hormiga debe decidir cual elemento incorpora a su clase entre los que no pertenecen a la lista tabú se utiliza un criterio probabilístico. Se introduce P_{Kj} como la probabilidad de que una hormiga k (que ya ha incorporado algunos elementos a su clase) decida recoger el elemento xj, no clasificado aún. Para ello, se define:

$$P_{kj} = \frac{\left[\tau_j^k(t)\right]^{\alpha}\left[\eta_j^k\right]^{\beta}}{\sum_{j / x_j \notin tabu(i)}\left[\tau_j^k(t)\right]^{\alpha}\left[\eta_j^k\right]^{\beta}}$$

donde $\tau_j^k(t)\ y\ \eta_j^k$ calculan convenientemente.

Se define:

$$\tau_j^k(t) = \sum_{i/x_j \in C_k} \tau_{ij}(t),$$

donde $\tau_{ij}(t)$ representa la feromona asociada al arco que conecta los elementos x_i y x_j. Es decir, asociamos al arco virtual que conecta una hormiga k con una observación x_j, la suma de los niveles de feromona sobre los arcos que conectan las observaciones ya clasificadas en la clase de dicha hormiga con la observación x_j.

La actualización de los niveles de feromona sobre los arcos virtuales conectando observaciones constituye otro aspecto que debe ser adecuado al problema de clasificación y es realizada de dos formas.

1. Paso a paso. Cuando una hormiga avanzar al próximo nodo o individuo, es actualizada la feromona en ese arco mediante la siguiente expresión:

$$\tau_{ij}(t) = (1-\rho) * \tau_{ij} + \rho * \tau_0$$

2. Actualización global. Se realiza al finalizar cada ciclo a la mejor solución encontrada hasta el momento, de acuerdo a la siguiente expresión:

$$\tau_{ij}(t) = (1-\rho) * \tau_{ij} + \rho * \Delta\tau$$

donde:

$$\Delta\tau = \begin{cases} \frac{1}{L} & \text{si } x_i \text{ y } x_j \text{ pertenecen a una misma clase de la mejor partición generada en el ciclo} \\ 0 & \text{en otro caso} \end{cases}$$

donde *L* es la inercia intraclase asociada a la mejor clasificación obtenida en el ciclo, esto es:

$$L = \sum_{k=1}^{K} \sum_{i / x_i \in C_k} d(x_i, g_k)$$

siendo g_k el centro de gravedad de la clase C_k.

2.3.2 Coloración Robusta y Clasificación Automática no Supervisada mediante EDA

Para optimizar $F(A,c)$ y $W(P)$, en los problemas de Coloración Robusta y de Clasificación Automática no Supervisada respectivamente, se hizo uso de los EDA. En dichos problemas cada punto o solución no es más que un vector *n-dimensional* de variables, donde cada dimensión representa un elemento y su valor el número de la clase a la que ha sido asignado.

A continuación se presentan las variantes de EDA, que además del UMDA se han usado para abordar estos problemas.

2.3.2.1 PADAt2 y PADAt3

Estas variantes asumen un modelo de árbol como distribución el cual puede escribirse como un producto de distribuciones condicionales bivariadas de la forma [42]:

$$p(x) = \prod_{i=1}^{n} p\left(\frac{x_i}{x_{j(i)}} \right)$$

donde $x_{j(i)}$ es la variable designada como padre de x_i en alguna orientación del árbol.

2.3.2.2 PADAp2 y PADAp3

PADAp2 y PADAp3 asumen un modelo de poliárbol como distribución que puede escribirse como un producto de distribuciones condicionales de la forma [42]:

$$p(x) = \prod_{i=1}^{n} p\left(x_i \mid x_{j1(i)}, x_{j2(i)}, ..., x_{jm(i)}\right)$$

donde $x_{j1(i)}, x_{j2(i)}, ..., x_{jm(i)}$ conjunto de padres (puede ser vacío) de la variable x. Los padres de cada variable son mutuamente independientes, tal que:

$$p\left(x_{j1(i)}, x_{j2(i)}, ..., x_{jm(i)}\right) = \prod_{k=1}^{m} p\left(x_{jk(i)}\right)$$

CAPÍTULO 3: RESULTADOS EXPERIMENTALES

El presente capitulo expone los resultados experimentales obtenidos y está organizado de la siguiente forma. Contiene tres secciones que tratan sobre el problema de Clasificaron Automática no Supervisada mediante EDA y ACO y el de Coloración Robusta mediante EDA. De cada uno se muestra el diseño del experimento para el problema al que se le quiere dar solución y finalmente, se muestra el análisis de los resultados obtenidos.

3.1 Clasificación Automática no supervisada mediante EDA

3.1.1 Diseño del experimento

Para mostrar los resultados de abordar el problema en estudio con EDA se ha tomado como conjuntos de datos de prueba los que usualmente se utilizan en la literatura: Objetos Notas [49], Objetos Amiard [50], Objetos Thomas [49]; de los cuales se conoce el óptimo para diferentes números de clases a particionar.

La metodología que se sigue en los experimentos es la siguiente:

1. El algoritmo se corre 100 veces sobre un conjunto de prueba a particionar en *k* clases.
2. Después se calcula el por ciento del número de veces que se alcanza el óptimo en esas corridas (E) y la generación media de convergencia (G). El tamaño de la población se fijó en 50, así como el por ciento de selección en 30.

3.1.1.1 Experimento I: UMDA, PADAt2, PADAt3, PADAp2, PADAp3 en conjunto de datos de prueba Objetos Notas.

En este experimento se corren los algoritmos UMDA, PADAt2, PADAt3, PADAp2, PADAp3 sobre el conjunto de datos de prueba Objetos Notas fijando el número de clases a particionar en 2, 3 y 4.

Tabla 3.1. Resultados del experimento sobre el conjunto de datos de prueba Objetos Notas.

	UMDA		*PADAt2*		*PADAt3*		*PADAp2,*		*PADAp3*	
Clases	*E*	*G*	*E*	*G*	*E*	*G*	*E*	*G*	*E*	*G*
2	100	2	100	2	100	3	100	3	100	3
3	98	26	96	21	98	17	94	19	97	17
4	99	39	100	16	100	22	99	45	99	36

3.1.1.2 Experimento II: UMDA, PADAt2, PADAt3, PADAp2, PADAp3 en conjunto de datos de prueba Objetos Amiard

En este experimento se corren los algoritmos UMDA, PADAt2, PADAt3, PADAp2, PADAp3 sobre el conjunto de datos de prueba Objetos Amiard fijando el número de clases a particionar en 2, 3 y 4.

Tabla 3.2. Resultados del experimento sobre el conjunto de datos de prueba Objetos Amiard.

	UMDA		*PADAt2*		*PADAt3*		*PADAp2,*		*PADAp3*	
Clases	*E*	*G*	*E*	*G*	*E*	*G*	*E*	*G*	*E*	*G*
2	13	92	52	109	30	105	42	72	37	112
3	29	104	96	91	77	107	46	82	35	110
4	5	112	44	125	21	139	12	105	7	102

3.1.1.3 Experimento III: UMDA, PADAt2, PADAt3, PADAp2, PADAp3 en conjunto de datos de prueba Objetos Thomas

En este experimento se corren los algoritmos UMDA, PADAt2, PADAt3, PADAp2, PADAp3 sobre el conjunto de datos de prueba Objetos Thomas fijando el número de clases a particionar en 3, 4 y 5.

Tabla 3.3. Resultados del experimento sobre el conjunto de datos de prueba Objetos Thomas.

	UMDA		*PADAt2*		*PADAt3*		*PADAp2,*		*PADAp3*	
Clases	*E*	*G*	*E*	*G*	*E*	*G*	*E*	*G*	*E*	*G*
3	22	493	99	113	100	175	2	201	3	244
4	54	229	64	337	47	395	0	-	18	398
5	79	326	46	404	43	432	1	406	3	390

3.1.2 Análisis de los resultados obtenidos

3.1.2.1 Conjunto de datos de prueba Objetos Notas

El conjunto de datos de prueba Objetos Notas (9 elementos y 5 variables que los caracterizan) es un problema fácil para los Algoritmos basados en la Estimación de la Distribución, los cuales obtienen un por ciento de éxito superior a 93 (Ver Figura 3.1).

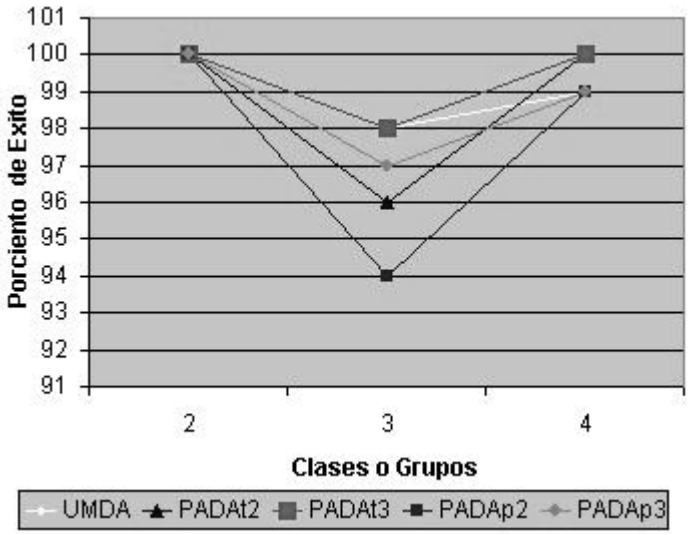

Figura 3.1. Por ciento de Éxito Objetos Notas.

La velocidad de convergencia (inversamente proporcional al número de la iteración en que se encuentra el óptimo) en UMDA, PADAt2, PADAt3, PADAp2 y PADAp3 disminuye con el aumento del número de clases (Ver Figura 3.2). Esto se debe a que el aumento del número de clases produce un aumento de la cardinalidad de cada variable, por lo que se necesita un población mayor para que los algoritmos de aprendizaje de la estructura del modelo probabilístico sean eficaces, lo cual no es posible pues el tamaño de población ha sido fijado.

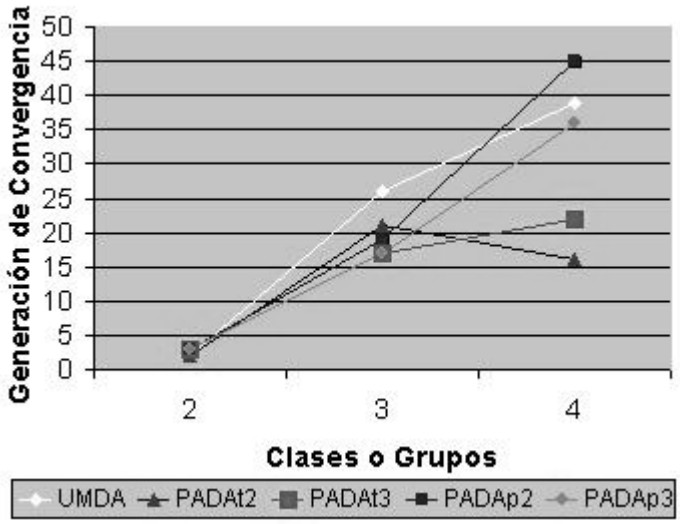

Figura 3.2. Generación de Convergencia Objetos Notas.

3.1.2.2 Conjunto de datos de prueba Objetos Amiard

El conjunto de datos de prueba Objetos Amiard (23 elementos y 16 variables que los caracterizan) no es un problema fácil para los Algoritmos de Estimación de la Distribución debido a que el éxito que logran varia como se muestra en la Figura 3.3. Nótese que UMDA aporta los peores resultados, lo cual se debe a su incapacidad de captar las relaciones de dependencia probabilística; y además como PADAt2 y PADAt3 logran aprender del modelo probabilístico para 3 clases, lo que se traduce en un por ciento de éxito superior.

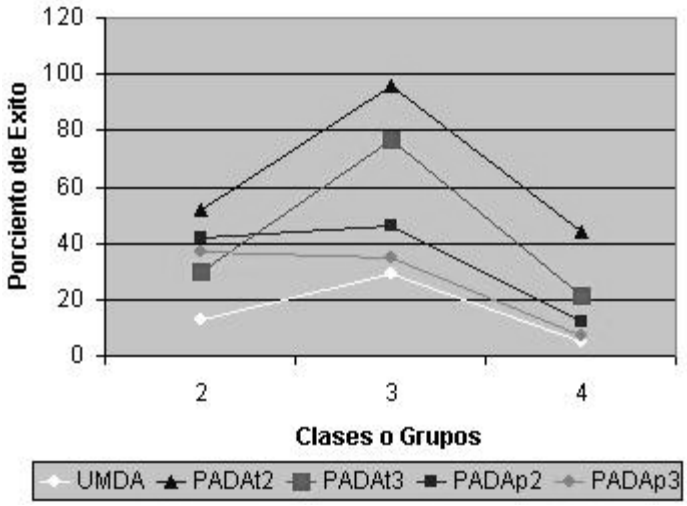

Figura 3.3. Por ciento de Éxito Objetos Amiard.

Una cuestión importante, es por qué los algoritmos en estudio al resolver el conjunto de datos de prueba Objetos Amiard, obtienen mejores resultados al particionar el conjunto en 3 clases, que al particionarlo en 2 clases. Como se podría esperar según lo expuesto en la sección anterior, en la que se muestra que el aumento del número de clases para un tamaño de población fijo, disminuye la velocidad de convergencia, lo cual puede influir negativamente en el por ciento de éxito. La respuesta a esta interrogante se expresa a continuación.

Aunque el conjunto de datos es el mismo, la distribución de los elementos en $R^m m = 16$ en este caso, puede provocar que al particionar en k clases, estas estén bien separadas entre sí y homogéneas, mientras que particionar el conjunto en p clases $(p < k)$ sea un proceso más difícil debido a que las clases no estén bien separadas entre sí o no sean homogéneas. Un ejemplo de ello en R^2 se muestra en la Figura 3.4.

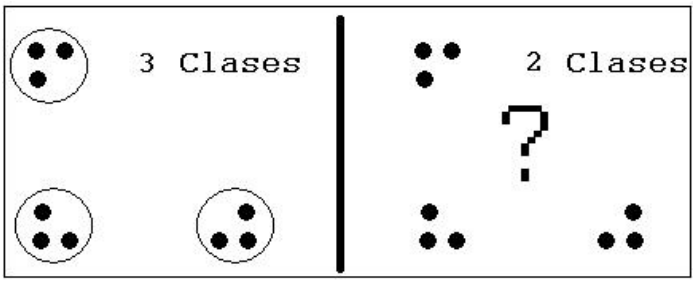

Figura 3.4. Grupos.

La velocidad de convergencia para este conjunto de datos de prueba no disminuye notablemente de 2 clases a 3 clases, lo cual está en correspondencia con la explicación que se acaba de expresar: el problema es más difícil para 2 clases, que para 3, por lo que el efecto negativo provocado por el aumento del número de clases para un tamaño de población fijo, se contrarresta (Ver Figura 3.5).

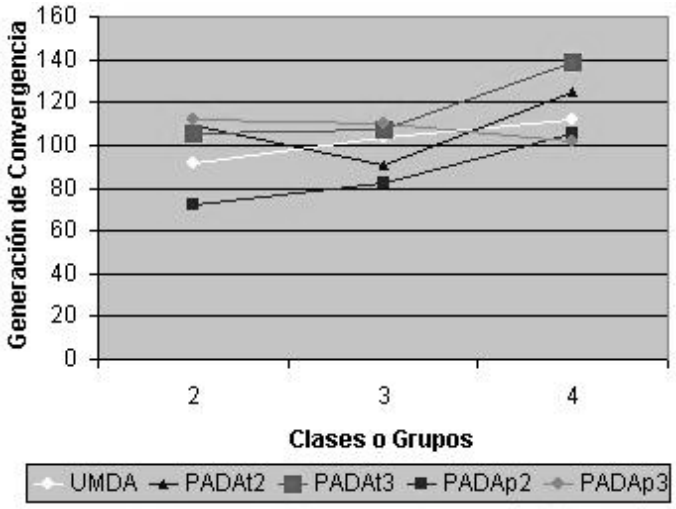

Figura 3.5. Generación de Convergencia Objetos Amiard.

3.1.2.3 Conjunto de datos de prueba Objetos Thomas

Objetos Thomas (24 elementos y 24 variables que los caracterizan) es un conjunto de datos de prueba difícil para los algoritmos en estudio. En este caso se aprecia como PADAt2 y PADAt3 superan a PADAp2 y PADAp3, lo cual se debe a que el modelo probabilístico de los datos que se va aprendiendo durante el proceso evolutivo se aproxima más a árboles que a poliárboles (Ver Figura 3.6).

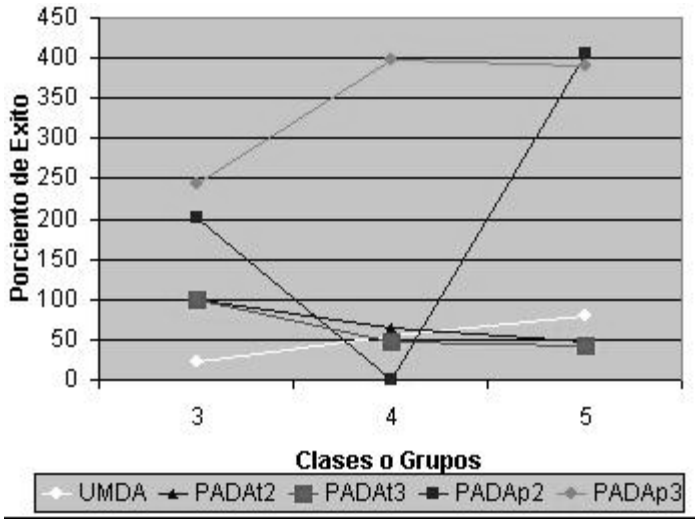

Figura 3.6. Por ciento de Éxito Objetos Thomas.

Nótese como el éxito de PADAt2 y PADAt3 disminuye, mientras en UMDA aumenta, al aumentar de número de clases. En este caso la disminución de éxito de PADA es esperada pues está en correspondencia con todo lo expuesto en secciones anteriores; pero eso no explica el aumento del éxito de UMDA. A continuación se comenta esta particularidad.

En el caso de UMDA, el aumento de la cardinalidad de las variables produce también un aumento en el tamaño de la población necesaria para aprender el modelo probabilístico de los datos. Pero el hecho de que el modelo asumido es de total independencia, provoca que ese aumento sea exponencialmente menor, lo cual se traduce en que el aumento del número de clases no afecte determinantemente el tamaño de la población, en este caso fijo, y que el éxito a su vez no se vea tan influenciado como en PADAt2, PADAt3, PADAp2 y PADAp3.

La velocidad de convergencia como en los conjuntos de datos de prueba anteriores se ve afectada con el aumento del número de clases (Ver Figura 3.7).

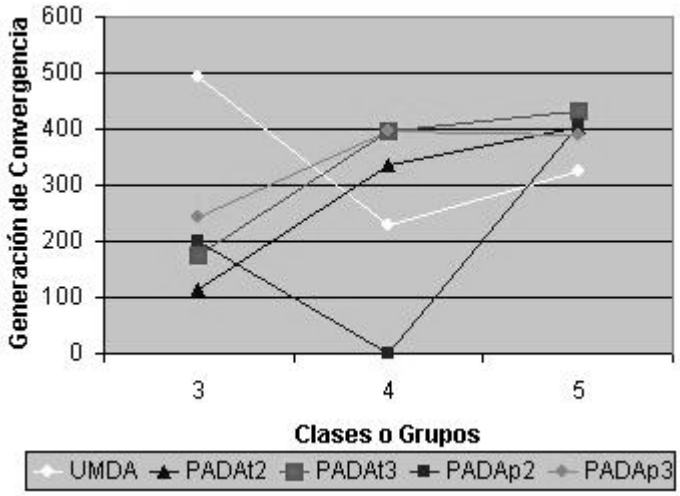

Figura 3.7. Generación de Convergencia Objetos Thomas.

3.2 Clasificación Automática no supervisada mediante ACO

3.2.1 Diseño del experimento

Para el experimento con el algoritmo ACS se utilizaron dos juegos de datos populares en la literatura, Objetos Iris [51] y Objetos Thomas. Se realizaron 35 corridas de cada uno, donde hubo variación entre los distintos parámetros. Para el parámetro que ajusta el número de hormigas, se utilizaron 400. Para la feromona inicial se utilizó el valor constante 0.1. En cambio, en la constante de evaporación, se hizo uso de los valores 0.1, 0.01 y 0.2, en las distintas corridas. Para q_0 se utiliza el valor 0.9 y finalmente para beta se hicieron las pruebas con los valores 3 y 5.

3.2.2 Análisis de los resultados obtenidos

En ambos conjuntos de datos: Objetos Iris y Objetos Thomas, las soluciones generadas mostraron un corportamiento bastante aceptable y que se acerca al óptimo esperado de ejemplos vistos en otras bibliografías [52].

3.2.2.1 Conjunto de datos de prueba Objetos Iris

Tabla 3.4. Mejores soluciones del problema con 2, 3 y 4 clases para el conjunto de datos Objetos Iris.

CANTIDAD DE HORMIGAS	*Fo*	*Qo*	*CLASES*	*CONST. DE EVAPORACIÓN*	*BETA*	*RESULTADO PROMEDIO OBTENIDO*
400	0.1	0.9	2	0.1	5	1.104917013327128
					3	1.1177268564809018
				0.01	5	1.1049170133271278
					3	1.1223220087841235
				0.2	5	1.10578490284254
					3	1.1142012436518978
			3	0.1	5	0.537092
					3	0.5370920000000001
				0.01	5	0.537092
					3	0.5381346666666667
				0.2	5	0.537092
					3	0.5370919999999999
			4	0.1	5	1.0692633001422476
					3	0.9859833570412517
				0.01	5	1.08860564248459
					3	0.9985195827406353
				0.2	5	0.9706363205310574
					3	0.9790350877192983

3.2.2.2 Conjunto de datos de prueba Objetos Thomas

Tabla 3.5. Mejores soluciones del problema con 2, 3 y 4 clases para el conjunto de datos Objetos Thomas.

CANTIDAD DE HORMIGAS	*Fo*	*Qo*	*CLASES*	*CONST. DE EVAPORACIÓN*	*BETA*	*RESULTADO PROMEDIO OBTENIDO*
400	0.1	0.9	2	0.1	5	312.7951388888889
					3	316.32638888888886
				0.01	5	315.03125
					3	315.02430555555554
				0.2	5	312.7951388888889
					3	316.32638888888886
			3	0.1	5	258.8177083333333
					3	264.2604166666667
				0.01	5	265.234375
					3	266.6510416666667
				0.2	5	258.4114583333333
					3	262.1302083333333
			4	0.1	5	235.15972222222226
					3	235.15972222222226
				0.01	5	232.2013888888889
					3	232.20138888888889
				0.2	5	234.89583333333334
					3	235.2013888888889

3.3 Coloración Robusta mediante EDA

3.3.1 Diseño del experimento

Para mostrar los resultados de abordar el problema Coloración Robusta con EDA, se ha tomado como conjuntos de datos de prueba 6 distribuciones de grafos distintas de 5, 10, 15, 14, 20 y 30 nodos de los cuales se conoce el valor óptimo de estabilidad para diferentes coloraciones (dígase cantidad de colores utilizados en la distribución).

Tabla 3.6. Juego de datos con 6 distribuciones de grafos.

TEST	*GRAFO*	*COLORES*	*ÓPTIMO ESPERADO*	*ARCHIVO ADJUNTO*
01	G_1 (5,aG1)	3	0.800000011920929	G(5,aG1)[3].txt
02	G_2 (10,aG2)	3	0.300000011920929	G(10,aG2)[3].txt
03	G_3 (15,aG3)	3	0	G(15,aG3)[3].txt
04	G_4 (14,aG4)	5	0	G(14,aG4)[5].txt
05	G_5 (20,aG5)	4	5.00079965591431	G(20,aG5)[4].txt
06	G_6 (30,aG6)	11	4.76229953765869	G(30,aG6)[11].txt

La metodología que se sigue en los experimentos es la siguiente: cada algoritmo se corre 100 veces sobre cada distribución; después se calcula el por ciento del número de veces que se alcanza el óptimo en esas corridas.

Para esto, se utilizaron los algoritmos: UMDA, PADAt2, PADAt3, PADAp2, PADAp3; el tamaño de la población se fijó en 100, el por ciento de selección en 30 y se utilizó variación del parámetro de mutación implícito de (ap = 0.03); las variables utilizadas variaron su dimensión de acuerdo a la de los nodos de los grafos utilizados.

3.3.2 Análisis de los resultados obtenidos

Con el análisis de los resultados obtenidos, mostrados en la Tabla 3.7 y la Figura 3.8, se puede apreciar que el por ciento de aciertos en las corridas aumenta a la par con la complejidad de las redes conectadas que se utilizan. Por otro lado, ha medida que aumenta la cantidad de variables y consecuentemente la complejidad del problema, se observa una disminución en la eficiencia de aprendizaje de las redes y la caída del por ciento de éxito.

Tabla 3.7. Por Ciento de Eficiencia en Pruebas Realizadas.

	UMDA	*PADAt2*	*PADAt3*	*PADAp2,*	*PADAp3*
G_1 {5} [3]	100	99	99	99	100
G_2 {10} [3]	50	67	64	67	73
G_3 {15} [3]	79	80	83	87	94
G_4 {14} [5]	58	65	70	77	89
G_5 {20} [4]	79	46	43	1	3
G_6 {30} [11]	69	71	76	77	82

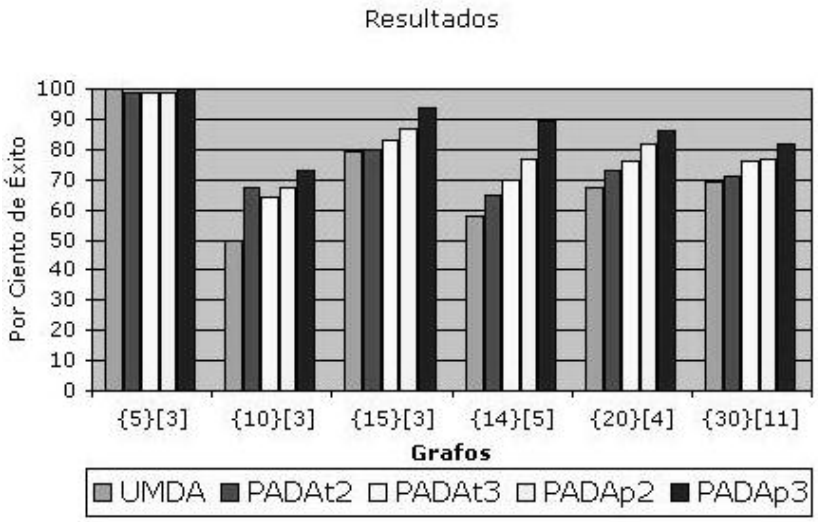

Figura 3.8. Por Ciento de éxito de los algoritmos UMDA, PADAt2, PADAt3, PADAp2, PADAp3.

Este es un problema complejo para UMDA, pues posee un número grande de dependencias entre las variables a medida que aumenta la cantidad de Nodos de los Grafos. En esos casos de complejidad apreciable en las distribuciones, a UMDA le resulta difícil acercarse al modelo de dependencias óptimo; en la figura se puede observar como el éxito de este algoritmo desciende a medida que aumenta el tamaño de los grafos.

Por otra parte PADAt2 y PADAt3 muestran mejores resultados que UMDA, pero sólo hasta el punto en que una red constituida por árboles es suficiente para aprender el modelo de dependencias, aunque no suficientes, pues nunca alcanzan el 100%.

También se aprecia como PADAt2 y PADAt3 superan a PADAp2 y PADAp3, lo cual se debe a que el modelo probabilístico de los datos que se va aprendiendo durante el proceso evolutivo se aproxima más a árboles que a poliárboles.

CONCLUSIONES

Las metaheurísticas EDA y ACO han demostrado ser una estrategia exitosa para resolver problemas, demostrando la utilidad de los principios evolutivos. Mantienen un buen comportamiento en problemas con funciones objetivos difíciles, por lo que deben ser considerados una opción más si se está tratando con problemas de cierto grado de complejidad. Como una nueva herramienta en la computación evolutiva que son, estas metaheurísticas tienen muchas posibilidades de investigar en el área de esta rama tan amplia.

Según los problemas tratados con EDA y ACO, se puede concluir:

- Los Algoritmos basados en la Estimación de la Distribución y específicamente UMDA, PADAt2, PADAt3, PADAp2 y PADAp3 son una alternativa viable para abordar el problema de Coloración Robusta y el problema de Clasificación Automática no Supervisada.

- El tamaño de la población y la cantidad de iteraciones fijadas por estos algoritmos es un parámetro que influye directamente su eficacia.

- PADAt2 y PADAt3 reportan los resultados más alentadores pero sus capacidades tienen un límite si se aplica a una coloración muy compleja; esto indica cuanto queda por hacer para lograr una red de aprendizaje capaz de captar un modelo de dependencias probabilísticas en su totalidad.

- El tamaño de la población fijada por estos algoritmos es un parámetro que influye directamente en su eficacia.

- PADAt2 y PADAt3 reportan los resultados más alentadores.

- La Optimización mediante Colonia de Hormigas (ACO) es una alternativa factible para abordar el problema de Clasificación Automática no Supervisada.

- Se utilizó el Sistema de Colonia de Hormigas (ACS) como algoritmo, mostrando resultados aceptables.

- La variación de los parámetros Cantidad de Clases, Beta y Constante de Evaporación influyen directamente en los resultados obtenidos por el algoritmo mencionado anteriormente.

Como se ha podido apreciar estas metaheurísticas resultan herramientas fáciles de implementar pues por la propia naturaleza de su filosofía están al alcance del entendimiento de cualquier persona interesada en el tema, no solo por lo poco difícil de su implementación si no por los resultados mostrados en la práctica.

Estas deben ser temas de estudio en cualquier campo donde la optimización esté entre los parámetros a considerar. Es importante tener en cuenta lo siguiente, el algoritmo a utilizar depende del problema al cual se le quiere dar solución, es por eso que no esta de más exhortar al estudio de este tipo de algoritmos que constituyen una herramienta más a tener en cuenta.

RECOMENDACIONES

Se recomienda la continuidad de este trabajo, preparando nuevos experimentos que puedan aportar resultados novedosos. Sería interesante tratar el problema de Coloración Robusta con ACO y comparar los resultados con los obtenidos con EDA. Además utilizar estos algoritmos en la solución de problemas prácticos como la generación de horarios docentes y en la gestión de proyectos.

REFERENCIAS BIBLIOGRÁFICAS

[1]. ESPARCIA A., SHARMAN K. Instituto Algoritmos bio-inspirados en logística. Revista del Instituto Tecnológico de Informática, disponible en: http://www.iti.upv.es/actualidadtic/2004/10/2004-10-CAS.pdf

[2]. PÉREZ BRITO D., MORENO PÉREZ J. A., GARCÍA GONZÁLEZ C. G. Capítulo 14 Búsqueda por entornos variables: desarrollo y aplicaciones en localización. Disponible en http://webpages.ull.es/users/gci/papers/Cap_VNS_Loc_2004.pdf

[3]. EVERITT B. S. (1993) Cluster Analysis. 3a edición. Edward Arnold, Londres

[4]. LERMAN, I.C. (1981) Classification et Analyse Ordinale des Données. Dunod, París

[5]. REINELT G. The Traveling Salesman: Computational Solutions for TSP Applications, volume 840 of Lecture Notes in Computer Science. Springer Verlag, Berlin, Germany, 1994.

[6]. VILLAGRA M. (2006). Convexidad Global en el Problema del Cajero Viajante Bi Objetivo. Artificial Intelligence in Theory and Practice. Editorial Springer, ISBN 0-387-34654-6

[7]. BULLNHEIMER B., KOTSIS G., STRAUSS C. “Parallelization Strategies for the Ant System”, Reporte Técnico POM 9/97, Universidad de Viena, Viena – Austria, 1997.

[8]. BARÁN B., ALMIRÓN M. “Colonias distribuidas de hormigas en un entorno paralelo asíncrono”. XXVI Conferencia Latinoamericana de Informática – CLEI’00, México, 2000.

[9]. DORIGO M., DI CARO G. The Ant Colony Optimization meta-heuristic. En D. Corne, M. Dorigo, y F. Glover, editores, New Ideas in Optimization, páginas 11-32. McGraw Hill, London, UK, 1999.

[10]. DORIGO M., STÜTZLE T. The ant colony optimization metaheuristic: Algorithms, applications and advances. En F. Glover and G. Kochenberger, editores, Handbook of Metaheuristics, páginas 251-285. Kluwer Academic Publishers, 2003.

[11]. MENDOZA GARCÍA B. Uso del Sistema de Colonia de Hormigas para Optimizar Circuitos Lógicos Combinatorios. Maestría en Inteligencia Artificial, Universidad Veracruzana, 2001.

[12]. BARÁN B., HERMOSILLA A. Comparación de un Sistema de Colonias de Hormigas y una Estrategia evolutiva para el Problema del Ruteo de Vehículos con Ventanas de Tiempo en un Contexto Multiobjetivo.

[13]. CANÓS M. J, IVORRA C., LIERN V. Aplicación de heurísticas con valoración fuzzy al problema de la p-mediana. 27 Congreso Nacional de Estadística e Investigación Operativa Lleida, 8-11 de abril de 2003.

[14]. ANDUJAR J.M., CÓRDOVA J.M., FERNÁNDEZ DE VIANA I. Una técnica híbrida de modelización neuroborrosa mediante Colonia de Hormigas

[15]. ALONSO S., CORDÓN O., FERNÁNDEZ DE VIANA I., HERRERA F. La Metaheurística de Optimización Basada en Colonias de Hormigas: Modelos y Nuevos Enfoques. Proyecto “Mejora de Metaheurísticas mediante Hibridación y sus Aplicaciones”. Universidad de Granada.

[16]. BENGOETXEA E. Inexact Graph Matching Using Estimation of Distribution Algorithms. 2002

[17]. LARRAÑAGA P. Algoritmos de Estimación de Distribuciones = Computación Evolutiva + Modelos Gráficos Probabilísticos. 2002.

[18]. LARRAÑAGA P. Synergies Between Probabilistic Graphical Models and Evolutionary Computation. Mexican International Conference on Artificial Intelligence 2002_ MICAI2002_ Mérida _ 25th 2002.

[19]. LARRAÑAGA P. Redes Bayesianas. Universidad de La Laguna _ Universidad de verano _ Adeje, 26 julio 2002.

[20]. MENDIBURU A., BENGOETXEA E., MIGUEL J. Paralelización de algoritmos de estimación de distribuciones. XIII JORNADAS DE PARALELISMO—LLEIDA, SEPTIEMBRE 2002.

[21]. VALDEZ PEÑA S. I., BOTELLO RIONDA S., HERNÁNDEZ AGUIRRE A. Optimización Multiobjetivo de Estructuras, Utilizando Algoritmos de Estimación de Distribuciones e Información de Vecindades. Comunicación Técnica No I-05-07/09-06-2005 (CC/CIMAT). 2005.

[22]. LARRAÑAGA P., LOZANO J.A. Estimation of distribution algorithms. A new tool for Evolutionary Computation. Kluwer Academic Publishers. 2001.

[23]. LOZANO J.A. Algoritmos de Estimación de Distribuciones en Problemas Bioinformáticos.

[24]. MÜHLENBEIN H., MAHNIG T. Theoretical Aspects of Evolutionary Computing, chapter Evolutionary Algorithms: From Recombination to Search Distributions, pages 137–176. Springer Verlag, Berlin, 2000.

[25]. MÜHLENBEIN H., PAAB G. From recombination of genes to the Estimation of Distributioni: Binary Parameters. Lecture Notes in Computer Science, (1411):178–187, 1996.

[26]. MÜHLENBEIN H. The equation for the response to selection and its use for prediction. Evolutionary Computation, 5(3):303–346, 1998.

[27]. MÜHLENBEIN H., MAHNIG T. Mathematical analysis of evolutionary algorithms for optimization. In Evolutionary Computation and Probabilistic Graphical Models. Proceedings of the Third Symposium on Adaptive Systems (ISAS-2001), pages 166–185, Habana, Cuba, March 2001.

[28]. MELIÁN B., MORENO PÉREZ J.A., MORENO VEGA J. M. Metaheuristics: A global view. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19 (2003),pp. 7-28 ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista).

[29]. MÜHLENBEIN H., MAHNIG T., OCHOA A.Schemata, distributions and graphical models in evolutionary optimization. Journal of Heuristics, 5(2):213–247, 1999.

[30]. ALFONSO GALIPIENSO M. I. SALIDO GREGORIO M. A. Modelos y Técnicas para Problemas de Satisfacción de Restricciones. Universidad de Alicante. Dpto de Ciencia de la Computación e Inteligencia Artificial, disponible en http://www.dccia.ua.es/dccia/inf/asignaturas/MTPSR/CAPITULO%201.ppt

[31]. LARRAÑAGA P., LOZANO J.A., MÜHLENBEIN H. Estimation of Distribution Algorithms Applied To Combinatorial Optimization Problems. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.19 (2003), pp. 149-168 ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista).

[32]. PANTRIGO FERNÁNDEZ, J.J. Resolución de Problemas de Optimización Dinámica Mediante la Hibridación entre Filtros de Partículas y Metaheurísticas Poblacionales. Tesis doctoral, Universidad Rey Juan Carlos.

[33]. HERRERA F. Introducción a los Algoritmos Metaheurísticos. Disponible en http://sci2s.ugr.es/seminars/5/Int-Metaheuristicas-UIA-Baeza-2006.pdf

[34]. ARRIBAS I. Optimización Combinatoria. Aplicación al problema de la asignación de rutas. Disponible en http://w3.mor.itesm.mx/~logica/logyprob/sesiones/Sat&Complexity.ppt

[35]. FRAUSTO SÓLIS J. Complejidad Computacional. Tecnológico de Monterrey División de Ingenieria y Ciencias. Disponible en http://w3.mor.itesm.mx/~jfrausto/Algoritmos/material/complejidad1.0.ppt

[36]. RAMÍREZ RODRÍGUEZ J. Extensiones del Problema de Coloración de Grafos. Tesis para optar por el grado de doctor. Universidad Complutense de Madrid. ISBN: 84-669-1855-8. 2001.

[37]. PIZA E., TREJOS J., MURILLO A. Clasificación Automática: Particiones utilizando Algoritmos Genéticos y de Sobrecalentamiento Simulado. Informes de Investigación PI-114-94-228. Universidad de Costa Rica, San Pedro. 1994-1995.

[38]. PIZA E., TREJOS J. “Particionamiento usando sobrecalentamiento simulado y algoritmos genéticos”, Memorias IX Simposio Métodos Matemáticos Aplicados a las Ciencias, J. Trejos (ed.), Turrialba 15–17 febrero, Universidad de Costa Rica –Instituto Tecnológico de Costa Rica, pp. 121–132. 1995.

[39]. MARCZYK A. Algoritmos genéticos y computación evolutiva. (Octubre de 2004).

[40]. LARRAÑAGA P., PUERTA J. Algoritmos de Estimación de Distribución. (12 de diciembre de 2003).

[41]. FERNÁNDEZ E., PUERTA J. Sistemas bioinspirados: los sentidos artificiales. Disponible en http://www.cienciadigital.es/hemeroteca/reportaje.php?id=73. (2001).

[42]. SOTO M. A Singled Connected Factorized Distribution Algorithm and its cost of evaluation. PhD thesis, University of Havana, Havana, Cuba, July 2003. (In Spanish, adviser Alberto Ochoa).

[43]. PASTEELS J. M., DENEUBOURG J.L., GOSS S. Self-organization mechanisms in ant societes (I): Trail recruitment to newly discovered food sources. Experientia Supplementum, 54, páginas 155-175. (1987).

[44]. OSMAN I. H., KELLY J. P. Meta- Heuristics:"Theory and Applications". Kluwer Academic Publishers, Boston, MA. (1996).

[45]. STÜTZLE T., H. HOOS H. Improving the Ant system: A detailed report on the MAXMIN Ant System. Technical Report AIDA- 96-12, FG Intellektik, FB Informatik, TU Darmstadt, Alemania. (Agosto 1996).

[46]. STÜTZLE T., H. HOOS H. MAX-MIN Ant System. Future Generation Computer Systems, 16: 8, páginas 889-914. (2000).

[47]. OTERO J., GUERRERO D. Clasificación automática mediante Colonia de Hormigas. (2004).

[48]. ARCO L., PIÑEIRO P. Y., MATILDE GARCÍA M., ESTRADA E. Plataforma para el desarrollo de sistemas de ayuda al diagnóstico y la clasificación. (2003).

[49]. SCHEKTMAN Y. Estadística Descriptiva: Análisis lineal de datos multidimensionales. Lst part. Proc. Symp. Math. Meth. Appl. Sci. J. Badía et al (eds.), Universidad de Costa Rica. (1978).

[50]. CAILLIEZ F., PAGÀS J. P. Introduction à l Ánalyse des Données. (1976)

[51]. EVERITT B. S. Cluster Analysis. 3ra edición. Edward Arnold, Londres. (1993).

[52]. OTERO J. Automatic Classification by Ant Colony Optimization. Summer School. Cairo. (2005).

[53]. GOSS S., ARON S., J. L. DENEUBOURG, PASTEELS J. M. Self-organized shortcuts in the Argentine ant. Naturwissenschaften, 76, páginas 579-581, 1989.

[54]. CORDÓN O. Dpto. Ciencias de la Computación e Inteligencia Artificial. BIOINFORMÁTICA Tema 2: Algoritmos de Optimización basados en Colonias de Hormigas. Universidad de Granada. Disponible en: http://sci2s.ugr.es/docencia/bioinformatica/Bioinformatica-Tema_2.pdf.

[55]. DORIGO M., MANIEZZO V., COLORNI A., The Ant System: Optimization by a Colony of Cooperating Agents, IEEE Transactions on Systems, Man, and Cybernetics 26:1, 1996, pp. 29-41.

[56]. BARCOS L., RODRÍGUEZ V. M., ÁLVAREZ M. J. Algoritmo basado en la optimización mediante colonias de hormigas para la resolución del problema del transporte de carga desde varios orígenes a varios destinos. V Congreso de Ingeniería del Transporte. 2002.

[57]. PAVEZ SALAZAR A., ACEVEDO ALMONACID H. Un Algoritmo ACS con Selección Dinámica de Movimiento y Operador 2-Opt. Dialnet, Ingeniería informática, ISSN 0717-4195, Nº. 8, 2002. Disponible en http://dialnet.unirioja.es/servlet/articulo?codigo=2097265

[58]. RAJPOOT N., HUSSAIN A., ALI U., SALEEM K., QURESHI M. A Novel Image Coding Algorithm using Ant Colony System Vector Quantization. International Workshop on Systems, Signals and Image Processing. Poznań, POLAND. September 13-15, 2004.

[59]. WHITE T., KAEGI S., ODA T. Revisiting Elitism in Ant Colony Optimization. School of Computer Science, Carleton University 1125 Colonel By Drive, Ottawa, Notario, Canada K1S 5B6. GECCO 2003,. LNCS 2723, pp. 122-133. 2003. Disponible en: http://terri.zone12.com/doc/academic/GECCO-2003-ACS-LBT.pdf.

[60]. BALUJA S., DAVIES S. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In Proceedings of the 14th International Conference on Machine Learning, pages 30–38. Morgan Kaufmann, 1997.

[61]. BONET J., CHARLES I., VIOLA P. MIMIC: Finding optima by estimating probability densities. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, Advances in neural information processing systems, volumen 9, página 424. The MIT Press, Cambridge, 1997.

[62]. CAMPOS L. Independency relationship and learning algorithms for singly connected networks. Journal of Experimental and Theoretical Artificial Intelligence, páginas 511–549, 1998.

[63]. OCHOA A., SOTO M. R., SANTANA R., MADERA J. C., JORGE N. The Factorized Distribution Algorithm and the juction tree: A learning perspective. In Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99), páginas 368–377, Habana, Cuba, Marzo 1999.

[64]. OCHOA A. Finding wavelet packets with estimation distribution algorithms. In A. S. Wu, editor, Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, página 802, Orlando, FL, 1999. Morgan Kaufmann Publishers, San Francisco, CA.

[65]. OJEDA L. M. Una exploración sobre los Algoritmos con Estimación de Distribución: el caso entero y su mutación. Habana, Cuba, Junio 2004.

[66]. PELIKAN M., MÜHLENBEIN H. The Bivariate Marginal Distribution Algorithm. Advances in Soft Computing Engineering Design and Manufacturing, páginas 521–535, 1999.

[67]. SOTO M. R., OCHOA A., ACID S., CAMPOS L. M. Bayesian evolutionary algorithms based on simplified models. In Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99), páginas 360–367, Habana, Cuba, Marzo 1999.

[68]. Soto M., Ochoa A. A Factorized Distribution Algorithm based on polytrees. In Proceedings of the 2000 Congress on Evolutionary Computation CEC00, páginas 232–237, La Jolla Marriott Hotel La Jolla, California, USA, 6-9 Julio 2000. IEEE Press.

GLOSARIO

1 ACO: Ant Colony Optimization

2 ACS: Ant Colony System

3 Algoritmos Bio-inspirados: comprenden e imitan la forma en que los sistemas biológicos aprenden y evolucionan, tienen como característica el ser autoadaptativos, autoorganizados y ser capaz de autoaprender.

4 Algoritmos genéticos: es una técnica de programación que imita a la evolución biológica como estrategia para resolver problemas.

5 AS: Ant System

6 BIC: Bayesian Information Criterion.

7 BMDA: Bivariate Marginal Distribution Algorithm.

8 BOA: Bayesian Optimization Algorithm.

9 BWAS: Best- Worst Ant System

10 cGA: Compact Genetic Algorithm.

11 COMIT: Combining Optimizers with Mutual Information Trees.

12 EBNA: Estimation of Bayesian Networks Algorithm.

13 ECGA: Extended Compact Genetic Algorithm.

14 EDA: Estimation of Distribution Algorithm.

15 EGNA: Estimation of Gaussian Networks Algorithm.

16 FDA: Factorized Distribution Algorithm.

17 Feromona: sustancia que las hormigas pueden oler y a la cual tienen cierta tendencia a seguir.

18 GA: Genetic Algorithm.

19 LFDA: Learning Factorized Distribution Algorithm.

20 Metaheurísticas: conjunto de procesos iterativos que guían y modifican las operaciones de los métodos heurísticos subordinados para producir eficientemente soluciones de alta calidad.

21 MIMIC: Mutual-Information-Maximizing Input Clustering.

22 MIMICGc: Mutual Information Maximization for Input Clustering Continuous. Gaussian

23 MMAS: Max-Min Ant System

24 PADA: Polytree Approximation Distribution Algorithm

25 PADAt2: PADA Quadratic Tree

26 PADAt3: PADA Cubic Tree

27 PADAp2: PADA Quadratic Polytree

28 PADAp3: PADA Cubic Polytree

29 PBIL: Population-Based Incremental Learning.

30 PBILc: Population Based Incremental Learning. Continuous

31 SHCLVND: Stochastic Hill Climbing with Learning by Vectors of Normal Distributions.

32 UMDA: Univariate Marginal Distribution Algorithm.

33 UMDAc: Univariate Marginal Distribution Algorithm continuous