Universidad de Granada

Departamento de Ciencias de la Computación e
Inteligencia Artificial

"Reconocimiento de Perfiles de Regulación Genética Mediante Algoritmos Evolutivos Multiobjetivo"

Tesis Doctoral

Rocío Celeste Romero Zaliz

Granada, Julio 2005

Universidad de Granada

"Reconocimiento de Perfiles de Regulación Genética Mediante Algoritmos Evolutivos Multiobjetivo"

MEMORIA QUE PRESENTA

Rocío Celeste Romero Zaliz

PARA OPTAR POR EL GRADO DE DOCTOR EN INFORMÁTICA

Julio 2005

DIRECTORES

Óscar Cordón García e Igor Zwir

Departamento de Ciencias de la Computación e Inteligencia Artificial

Editor: Editorial de la Universidad de Granada
Autor: Rocio Celeste Romero Zaliz
D.L.: Gr. 1240 - 2005
ISBN: 84-338-3531-9

La memoria titulada *"Reconocimiento de Perfiles de Regulación Genética Mediante Algoritmos Evolutivos Multiobjetivo"*, que presenta D^a. Rocío Celeste Romero Zaliz para optar al grado de doctor, ha sido realizada dentro del programa de doctorado *"Diseño, Análisis y Aplicaciones de Sistemas Inteligentes"* del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección de los doctores D. Óscar Cordón García y D. Igor Zwir.

Granada, Julio 2005

El Doctorando

Fdo: Rocío C. Romero Zaliz

Los Directores

Fdo: Óscar Cordón García e Igor Zwir

Agradecimientos

Aún no puedo llegar a comprender como he podido conseguir terminar este trabajo de tesis, lo cual parece una hazaña sin precedentes. Si hay algo que la facultad me ha enseñado es que todo tiene una justificación científica. Así que intentaré inferir, científicamente hablando, las razones que me han llevado a esta situación para por comprenderla mejor:

Si bien la biología no es una ciencia exacta, la teoría de la evolución ha sido, de alguna manera, validada científicamente. Esto me lleva a pensar que, si he logrado terminar esta tesis, es porque heredé los genes adecuados. ¿Estoy en lo correcto?

Pero dice también la psicología, que la educación y el entorno en donde crece un niño es determinante para su desarrollo, tanto mental como social. ¿Será entonces esa la verdadera razón?

Por otro lado, se dice que tener buenos ejemplos a seguir guían mejor a una persona en su vida. ¿Será esto también cierto?

Pero si seguimos con esta línea de pensamiento podríamos añadir que estar en buena compañía ayuda, ¿no lo creen?

En resumen, es por todo lo anterior que deduzco que todos son razones válidas y de peso, y es por ello que esta memoria está dedicada a la familia, la *biológica* y la *científica*. De la familia biológica no puedo dejar de nombrar a mis padres que me han ayudado a ser quien soy y a conseguir todos mis objetivos. Por supuesto, a mis hermanitas que me han dado su apoyo y "aguantado" mis diversos estados de ánimo.

Para la familia científica deberíamos generar un árbol genealógico. Comenzando por mis "padres científicos", Igor y Óscar, que han puesto mucho de sí para que, tanto mi persona como mi trabajo, saliera adelante con éxito. Siguiendo por un agradecimiento especial a mi "hermana científica pequeña", Cristina. A mi "primos y primas científicos" de España y de Argentina, aquellos en mi misma situación que han estado ahí en las buenas y en las malas, de entre ellos

destaco a mis compañeros de la UBA (Rosana y Daniel) y a mis compañeros del Mecenas (Jesús, Menchu, a los Carlos y al plantel de la 16). A mis "tíos científicos": de Argentina no puedo olvidarme de Anita, Juliana, María Elena, Julio, Marisa, Paula e Isabel; de España, a Raúl, MariCarmen, Miguel y Nacho. Y finalmente, a mis "padrinos científicos", Paco e Irene, uno de cada hemisferio, cuya directivas siempre estuvieron presentes en cada paso de mi trabajo.

Y a todo aquel que por descuido u omisión he olvidado de mencionar que haya aportado su granito de arena en mi vida. A todos, gracias.

Índice general

Índice de figuras

Índice de tablas

Introducción

I. Planteamiento

Durante las últimas décadas, se ha estado acumulando conocimiento, proveniente de diferentes áreas, en repositorios de datos digitales. Al estar almacenada de esta manera, la información puede ser estudiada y compartida más fácilmente por distintos expertos. A pesar de esto, el aumento constante del tamaño de estos repositorios hace casi imposible, para un ser humano, extraer información útil de los mismos. Por esta razón, se han desarrollado diversas técnicas de *minería de datos* para poder revelar información oculta en grandes colecciones de datos [45, 49] y así poder ayudar a los usuarios. Estas técnicas funcionan correctamente con representaciones de datos en forma atributo-valor, es decir, datos no estructurados. Sin embargo, muchos proyectos de adquisición de datos actuales acumulan información estructurada que describe no sólo los objetos de la base de datos, sino también las relaciones que existen entre ellos. Estos conjuntos de datos son estructurados en el sentido de que los objetos que almacenan están descritos por las relaciones entre las características y no solamente por las características en sí mismas. Debido a ello, existe la necesidad de crear técnicas que permitan analizar y descubrir conceptos definidos mediante subestructuras en repositorios de datos estructurados.

En particular, en el área de la *Bioinformática*, se utilizan grandes repositorios de datos que contienen información de secuencias de ADN o proteínas de genomas completos. Por ejemplo, la base de datos GenBank [16] recogía, en 2004, 40.604.319 secuencias biológicas compuestas por 44.575.745.176 pares de bases. Esto es debido, en gran parte, a los avances en la Biología Molecular y al equipamiento disponible para la investigación en este campo, que ha permitido la rápida secuenciación de grandes porciones de genomas de diversas especies. En la actualidad, varios genomas de bacterias y algunos eucariotas simples ya han sido secuenciados por completo. Esta gran cantidad de información necesita

de un alto nivel de organización, indexado y almacenamiento de las secuencias.

Adicionalmente, una gran parte de las bases de datos biológicas almacenan información *estructurada*. Como ejemplo de este tipo de base de datos, se puede mencionar la base RegulonDB [84], que contiene datos sobre la regulación transcripcional, organización de operones y condiciones de expresión para el genoma *Escherichia coli K-12*. Otro ejemplo lo constituye la base de datos *Gene Ontology* [8], que almacena términos asociados a los procesos biológicos, funciones moleculares y componentes celulares de secuencias biológicas de forma jerárquica.

Las técnicas clásicas de aprendizaje no supervisado para la extracción de información en bases de datos incluyen a los algoritmos de clustering (agrupamiento) de datos, de las cuales se han desarrollado varias propuestas. Sin embargo, estas técnicas no pueden aplicarse satisfactoriamente a bases de datos estructuradas.

Para poder analizar repositorios estructurados, existe una alternativa al clustering tradicional, consistente en realizar descubrimiento de subgrupos mediante técnicas de *clustering conceptual*. El clustering conceptual busca no sólo una clasificación de los datos provistos, sino también una descripción simbólica de las clases propuestas (clusters). Consiste en un mecanismo de inducción descriptiva que persigue encontrar patrones interesantes en los datos. La principal ventaja del uso de estas técnicas es la capacidad que presentan no sólo para reconocer grupos en los datos, sino para describir las propiedades que los agrupan.

Sin embargo, tanto las técnicas de clustering tradicional, como los algoritmos de clustering conceptual, presentan algunos problemas:

1. En particular, las técnicas de clustering tradicional subdividen los objetos de la base de datos en grupos completamente disjuntos, lo que no es natural en la mayoría de dominios de datos reales.

2. Consideran a todos los elementos del repositorio igualmente importantes, cuando no suele ser así en la práctica donde no todos tienen el mismo nivel de interés. Algunas instancias pueden contener campos con valores desconocidos o estar expuestos a errores en su definición que pueden representar outliers y, por lo tanto, no ser fácilmente agrupables con el resto de la base de datos.

3. Se centran en la optimización de un único criterio de preferencia. Comúnmente, este criterio suele ser la combinación de dos o más métricas que calculan la bondad de una configuración de clusters. En la mayoría de los casos, estos criterios están contrapuestos. Por ello, la mayoría de las técnicas de clustering actuales se enfocan en encontrar una buena solución de compromiso, es decir, un conjunto de clusters que sean aceptablemente buenos simultáneamente en todos los criterios contemplados mediante la agregación de los mismos en un único criterio de preferencia.

En esta memoria, se propondrá una metodología que realiza descubrimiento de subgrupos mediante clustering conceptual haciendo uso de algoritmos evolutivos multiobjetivo. Esta propuesta pretende evitar los tres problemas anteriores que afectan a las técnicas clásicas de aprendizaje no supervisado. Para ejemplificar la propuesta, se aplicará la metodología a dos problemas de regulación genética, uno en organismos procariotas y otro en eucariotas.

El primer problema es una colaboración con el laboratorio del Dr. Eduardo A. Groisman en la Universidad de Washington con sede en St. Loius, Estados Unidos. Este proyecto está respaldado por una subvención del *Howard Huges Medical Institute* y estudia los mecanismos utilizados por la bacteria *Salmonella enterica*, los cuales le permiten prosperar en diferentes entornos y causar enfermedad. Esta bacteria infecta a millones de personas en el mundo cada año, lo cual resulta en 500.000 fallecidos aproximadamente. La comprensión de estos mecanismos podría derivar en nuevos tratamientos terapéuticos, prevención y estrategias de diagnóstico. Asimismo, la bacteria de *Salmonella* sirve como sistema modelo de otros patógenos intracelulares que no son sencillos de estudiar o carecen de modelos efectivos de infección.

El segundo problema constituye un esfuerzo conjunto dentro del programa de investigación *Inflamación y respuesta del huésped a estímulos externos (Inflammation and the Host Response to Injury)* en el cual participa el Dr. J. Perren Cobb. Este programa está respaldado por el *National Institute of General Medical Sciences (NIGMS)*, una división del *National Institutes of Health* de Estados Unidos y su objetivo es descubrir las razones biológicas por las cuales diferentes pacientes pueden llegar a tener respuestas muy distintas tras sufrir una herida traumática. Este programa interdisciplinario a gran escala constituye el primer intento por resolver problemas que resulten en peligros para la vida, producidos por quemaduras o inflamaciones seguidas a un trauma importante. Este proyecto reune a instituciones médicas e investigadores de las áreas de cirugía, genómica, proteómica, bioestadística, bioinformática, biología computacional y genética, para estudiar la biología molecular de las inflamaciones.

II. Objetivos

Esta memoria tiene como objetivo principal el desarrollo de una metodología que permita descubrir aquellos conceptos ocultos en un conjunto de datos estructurados, brindando al usuario no sólo de un conjunto de clusters que concentran a instancias de la base de datos que estén relacionadas entre sí, sino también de una descripción clara que le permita comprender el por qué, o en base a qué características, esas instancias se encuentran agrupadas. La razón de ser de esta metodología de clustering conceptual basada en algoritmos evolutivos multiobjetivo es la aplicación al descubrimiento de perfiles de regulación genética tanto en organismos procariotas como en eucariotas.

En concreto, el objetivo propuesto se subdivide en los siguientes sub-objetivos:

- Estudiar el problema biológico de regulación genética, tanto en organismos procariotas como en eucariotas. Este estudio debe realizarse desde dos puntos de vista, el *biológico* y el *informático*. Desde el punto de vista biológico, se ha de comprender el funcionamiento de las redes genéticas en ambas clases de organismos. Desde el punto de vista informático, tendremos que investigar los distintos enfoques existentes en la literatura para el reconocimiento de perfiles de regulación genética, ponderando tanto sus aciertos como sus problemas en su aplicación a situaciones reales.

- Proponer una metodología basada en algoritmos evolutivos multiobjetivo para el descubrimiento de subgrupos, mediante clustering conceptual, que permita la extracción de conocimiento oculto en bases de datos estructuradas y que solucione los problemas que afectan a las técnicas clásicas de clustering comentados anteriormente.

- Diseñar modelos de objetos biológicos, utilizando para ello los beneficios del modelado mediante técnicas de lógica difusa, aprendizaje automático y algoritmos evolutivos multiobjetivo. Construir modelos a la vez precisos e interpretables para cada uno de los componentes involucrados en el proceso de regulación genética, esperando descubrir nuevos datos relevantes que difieran de los consensos clásicos existentes en la literatura. Estos modelos son los que se utilizarán luego conjuntamente con la metodología propuesta.

- Aplicar la metodología propuesta a organismos procariotas, utilizando para ello la información existente en la base de datos RegulonDB [84]. Extraer de este repositorio información que relacione diferentes genes y validar este conocimiento con datos obtenidos experimentalmente.

- Aplicar la metodología propuesta a organismos eucariotas, utilizando para ello la información contenida en la base de datos HG-U133A v2.0 de Affymetrix Inc® y la base de datos de *Gene Ontology* [8]. Recuperar, de estos repositorios, información sobre un conjunto de genes evaluados en un estudio de la respuesta inflamatoria de seres humanos al aplicarles una endotoxina en forma intravenosa, en comparación con un grupo de control al cual se le inyecta un placebo [79]. Extraer de los resultados obtenidos, grupos de genes que tengan propiedades similares y validar este conocimiento con los datos experimentales.

III. Resumen

Para desarrollar los objetivos planteados, la memoria está organizada en cinco capítulos, una sección de comentarios finales y un apéndice. La estructura de cada una de estas partes se comentará brevemente a continuación.

En el Capítulo 1, se introduce al lector a los conceptos básicos de biología molecular, los cuales serán necesarios para la adecuada comprensión de los capítulos posteriores. Se comienza con una breve descripción de los componentes principales de los organismos vivientes, siguiendo por los procesos necesarios para el mantenimiento de la vida, y finalizando con una breve reseña acerca de los métodos biológicos para el estudio de secuencias de ADN. Adicionalmente, se introduce la *Bioinformática*, una disciplina moderna y en pleno apogeo, conjuntamente con los problemas clásicos que ésta abarca.

En el Capítulo 2, se presentan las diferentes técnicas y métodos sobre los cuales se basa la metodología principal propuesta en esta memoria. Los temas tratados en este capítulo son: el modelado o la identificación de sistemas, la *lógica difusa*, el *clustering (agrupamiento)* de conjuntos de datos, en particular el *clustering conceptual*, y los *algoritmos evolutivos*, con especial atención a la *programación genética* y a las técnicas de *optimización multiobjetivo*.

En el Capitulo 3, se explica en detalle la metodología propuesta para poder abordar los objetivos comentados anteriormente. Se comienza mostrando el esquema general, para luego desarrollar cada uno de los pasos que la componen. Para una mayor comprensión de la misma, se acompaña cada etapa con ejemplos realizados sobre un dominio sencillo.

En el Capítulo 4, se aplica la metodología propuesta a un problema sobre organismos procariotas, el estudio de la regulación genética en los organismos de *E. coli* y *Salmonella*. En las diferentes secciones que componen este capítulo, se explica como se ha adaptado la metodología general al problema específico, destacando especialmente el proceso de construcción y modelado del conjunto de datos utilizado. Finalmente, se validarán los resultados obtenidos comparándolos con datos experimentales.

En el Capítulo 5, se aplica la metodología propuesta a un problema sobre organismos eucariotas, un estudio de la respuesta genética a procesos inflamatorios en seres humanos [79]. En las diferentes secciones que componen este capítulo, se explica como se ha adaptado la metodología general al problema específico, desde la construcción de la base de datos hasta el proceso final de predicción sobre nuevos genes.

Incluimos luego una sección de "Comentarios Finales", que resume los resultados obtenidos en esta memoria, presentando algunas conclusiones sobre los mismos. Finalmente, se comentarán algunos aspectos sobre trabajos futuros que quedan abiertos en la presente memoria.

Por último, incluiremos también un apéndice donde se presentaran tablas y figuras adicionales asociadas a las dos aplicaciones de la metodología.

Capítulo 1

Fundamentos básicos de biología y bioinformática

En la actualidad podemos decir que todos los seres vivos están formados por células y que estas unidades de materia viva comparten una maquinaria común para sus funciones más básicas. Los seres vivos, aunque infinitamente diversos por fuera, son muy similares por dentro.

En este primer capítulo delinearemos las características universales de todos los seres vivos, inspeccionando brevemente la diversidad celular, y veremos cómo, gracias a un código común en el que están escritas las especificaciones de todos los organismos, es posible leer, medir y desentrañar estas especificaciones para alcanzar un conocimiento coherente de todas las formas de vida, de las más pequeñas a las más grandes.

1.1. Características universales de las células

Las células vivas, como los ordenadores, contienen información, y se estima que llevan evolucionando y diversificándose desde hace más de tres mil millones y medio de años. Todas las células vivas, sin ninguna excepción conocida, guardan su información hereditaria en forma de moléculas de ADN (ácido desoxirribonucleico) de doble cadena –dos largos polímeros paralelos no ramificados formados por cuatro tipos de monómeros[1]–. Estos monómeros están unidos entre sí formando una larga secuencia lineal que codifica la información genética de la célula, de la misma manera que la secuencia de unos y ceros codifica la información en un ordenador [17, 5].

[1] El material esencia o unidad con la cual se construye un polímero.

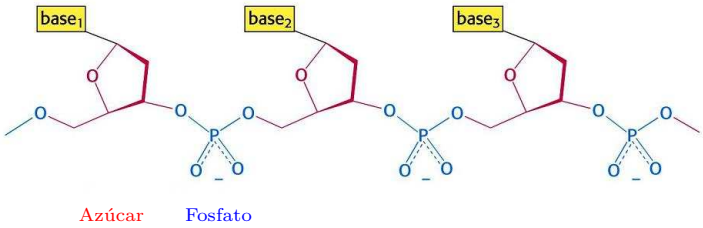


Figura 1.1: Esquema del ADN

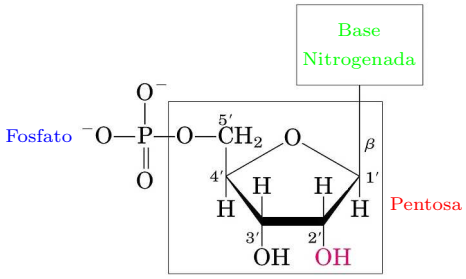
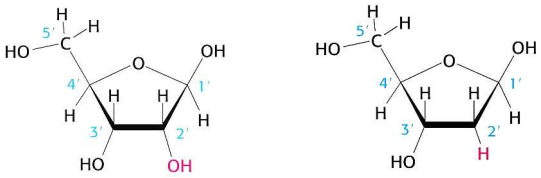


Figura 1.2: Esquema de un nucleótido

Para comprender los mecanismos biológicos, hemos primero de conocer la estructura de la molécula de ADN de doble cadena (ver Figura 1.1). Cada monómero de una de las cadenas sencillas del ADN –denominado **nucleótido** (ver Figura 1.2)– tiene dos partes: un azúcar (la desoxirribosa, ver Figura 1.3) con un grupo fosfato unido y una *base* que puede ser adenina (A), guanina (G), citosina (C) o timina (T) (ver Figura 1.4). Cada azúcar está unido al siguiente azúcar de la cadena por el grupo fosfato mediante un enlace fosfodiéster, formando un polímero cuyo eje central está compuesto por los azúcares fosfato y del cual sobresalen las bases. El polímero de ADN puede crecer por la unión de monómeros a uno de sus extremos. En el caso de una cadena sencilla de ADN, los monómeros pueden incorporarse al polímero de forma aleatoria, sin un orden preestablecido, ya que todos los nucleótidos pueden unirse entre sí del mismo modo en el sentido del crecimiento del polímero de ADN. Por el contrario, en la célula viva existe una limitación, ya que el ADN no se sintetiza como una cadena libre aislada sino sobre un patrón o molde de ADN de otra cadena preexistente. Las bases contenidas en la cadena patrón se unen a las bases de la nueva cadena siguiendo una estricta norma de complementariedad: A se une a T, y C se une a G (ver Figura 1.5). Este emparejamiento une los nuevos monómeros de la cadena y además controla la selección del monómero que se añade a la mis-



(a) Ribosa (b) Desoxirribosa

Figura 1.3: Azúcares

ma. De esta forma, una estructura de doble cadena consiste en dos secuencias complementarias de A, C, G y T. El orden de la secuencia es muy importante, ya que en él reside la información contenida en el ácido nucleico. La orientación viene dada en el sentido 5'-3' o 3'-5', donde el 5' representa el extremo terminal del fosfato y el 3' el extremo final del átomo de carbono de la desoxirribosa. Además, las dos cadenas de nucleótidos se enrollan una sobre la otra generando una doble hélice (ver Figura 1.6).

Los enlaces establecidos entre las bases son débiles si se comparan con las uniones azúcar-fosfato del resto del esqueleto. Esta debilidad permite separar las dos cadenas de ADN sin forzar la rotura de su esqueleto. Cada una de las cadenas puede comportarse como un molde o patrón en la síntesis de una nueva molécula de ADN.

Para llevar a cabo satisfactoriamente la función de almacén de la información, el ADN tiene que hacer algo más que duplicarse antes de cada división celular. Ha de expresar la información que contiene, utilizándola para dirigir la síntesis de otras moléculas de la célula. El mecanismo responsable de este proceso es el mismo en todos los organismos vivos y se inicia con la síntesis secuencial de dos tipos de moléculas: el ácido ribonucleico (ARN) y las proteínas. El proceso comienza con la polimerización sobre un patrón, denominada **transcripción**, proceso en el que diferentes segmentos de la secuencia de ADN se utilizan como molde para la síntesis de moléculas cortas de un polímero muy relacionado con el ADN: el **ácido ribonucleico** o **ARN** (ver Figura 1.7). Después de un proceso complejo denominado **traducción**, muchas de estas moléculas de ARN se utilizan para dirigir la síntesis de polímeros de una clase química radicalmente diferente: las *proteínas*.

En el ARN, el esqueleto del polímero está formado por azúcares ligeramente diferentes a los del ADN –ribosa en lugar de desoxirribosa– y, además, una de las cuatro bases es diferente –uracilo (U) (ver Figura 1.4) en el lugar de la timina (T)–, pero las otras tres bases –A, C, G– son las mismas y se emparejan con su

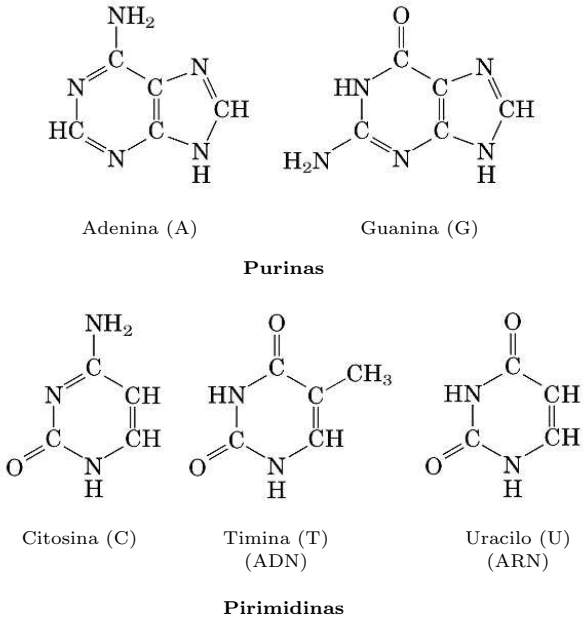


Figura 1.4: Bases nitrogenadas

complementaria, como en el ADN –la A, el U, la C y la G del ARN se unen con la T, la A, la G y la C del ADN, respectivamente. Durante la transcripción, los monómeros de ARN se seleccionan para la polimerización del ARN sobre una cadena molde de ADN, de la misma manera que se seleccionan los monómeros de ADN durante la replicación. El resultado de la transcripción es un polímero de ARN que contiene una parte de la información genética de la célula, aunque escrita en un alfabeto diferente de monómeros del ARN en lugar de monómeros de ADN.

Un mismo fragmento de la secuencia del ADN se puede usar varias veces para guiar la síntesis de muchos transcritos de ARN idénticos. Así, mientras que el archivo de información de la célula es fijo –en el ADN–, los transcritos de ARN se producen en gran número y son desechables. La función de la mayoría de estos transcritos es servir de intermediarios en la transferencia de la información genética, actuando como un **ARN mensajero** (ARNm) que dirige la síntesis de proteínas según las instrucciones almacenadas en el ADN.

Las moléculas de **proteína**, como el ADN o el ARN, son largas cadenas no ramificadas que están formadas por monómeros tomados de un repertorio idéntico en todas las células vivas. Como el ADN y el ARN, las proteínas contienen la información en forma de una secuencia lineal de símbolos. Los monómeros de

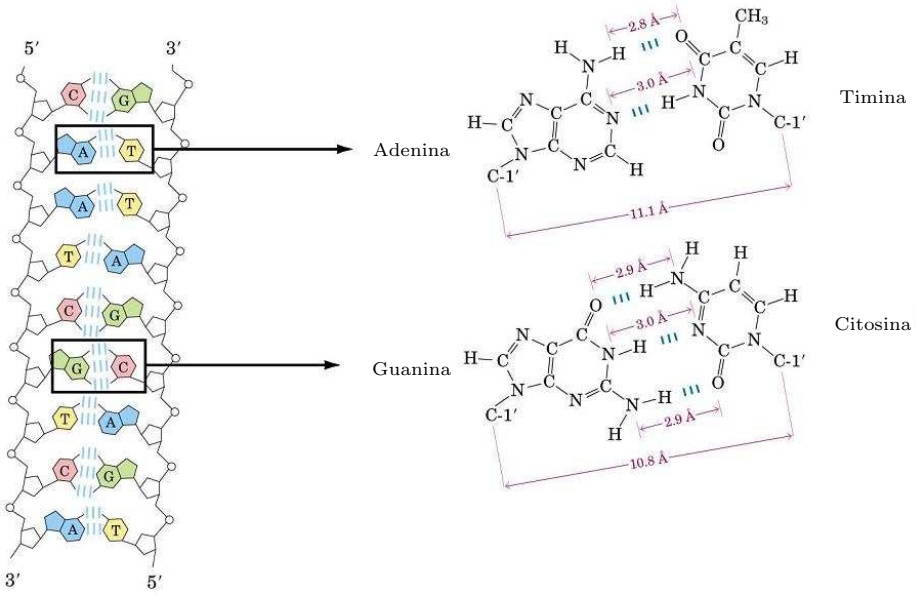


Figura 1.5: Emparejamiento de bases

las proteínas, los **aminoácidos**, son muy diferentes de los del ADN o el ARN y, además, existen veinte tipos diferentes en lugar de tan sólo cuatro. Todos los aminoácidos tienen una estructura central semejante por la que pueden unirse a los demás aminoácidos. Unido a esta estructura central, se encuentra un grupo lateral que confiere a cada aminoácido su carácter químico característico. Cada una de las moléculas proteicas o **polipéptido**, formada por la unión de varios aminoácidos siguiendo una secuencia determinada, se pliega en una estructura tridimensional. Las proteínas tienen muchas funciones –ser catalizadores de reacciones (**enzimas**), mantener estructuras celulares, generar movimientos, traducir señales, etc.– y cada una cumple una función específica según su secuencia de aminoácidos, determinada genéticamente. Las proteínas son las moléculas que ponen en acción la información genética de la célula.

La información contenida en la secuencia de ARNm se lee en grupos de tres nucleótidos; cada triplete de nucleótidos o *codón* especifica (codifica) un aminoácido de una proteína. Debido a que hay 64 posibles codones, pero sólo veinte aminoácidos, necesariamente hay muchos casos en los que varios codones corresponden a un mismo aminoácido. El código se lee por una clase especial de pequeñas moléculas de ARN, el **ARN de transferencia** (ARNt). Cada

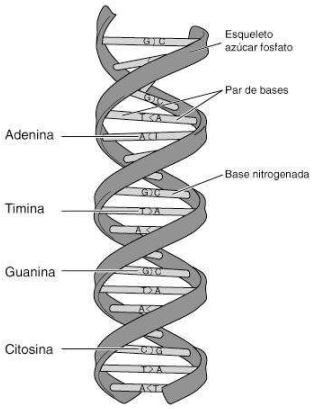


Figura 1.6: Doble hélice

tipo de ARNt une en uno de sus extremos un aminoácido y tiene una secuencia específica de tres nucleótidos en su otro extremo –un *anticodón*– que le permite reconocer un codón o subgrupo de codones del ARNm por emparejamiento de bases.

Para la síntesis de proteínas, un conjunto de moléculas de ARNt cargadas con sus aminoácidos respectivos se une a un ARNm por emparejamiento de sus anticodones con cada uno de los codones sucesivos del ARNm. Después, los aminoácidos se van uniendo de forma que la proteína naciente va creciendo y cada ARNt, relegado de su carga, se libera.

Las moléculas de ADN son muy largas y contienen la especificación de miles de proteínas. Por tanto, fragmentos de esta secuencia completa de ADN se transcriben en diferentes moléculas de ARNm, cada uno de los cuales codifica una proteína diferente. Un **gen** se define como un fragmento de la secuencia de ADN que corresponde a una sola proteína (o a una molécula de ARN catalítica o estructural, para los genes que producen ARN pero no proteína).

En todas las células, la *expresión* de determinados genes está regulada: en lugar de sintetizar el catálogo completo de posibles proteínas en todo momento, la célula ajusta la velocidad de transcripción y de traducción de diferentes genes de forma independiente y de acuerdo con sus necesidades. En el ADN celular existen secuencias de ADN no codificantes –denominadas *ADN regulador*– que están distribuidas entre las regiones codificantes de proteínas, y estas regiones no codificantes se unen a proteínas especiales que controlan la velocidad local de transcripción. Existen también otras regiones no codificantes, algunas de las cuales actúan como elementos de puntuación, indicando el inicio y el final de la información de una proteína. La región del ADN donde se establece cómo y

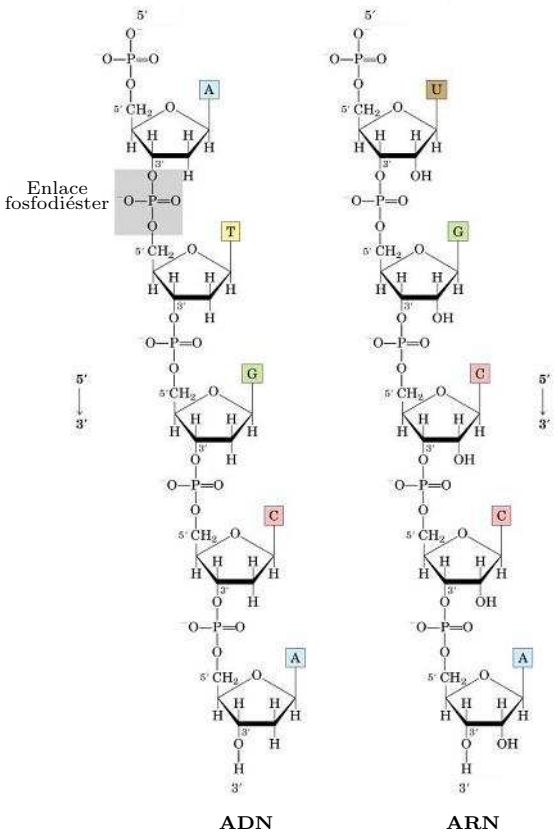


Figura 1.7: Polinucleótidos

cuándo se expresará el gen que se codifica en la región codificante inmediatamente adyacente se conoce como *región promotora*. En este sentido, el **genoma** de una célula –la totalidad de la información genética incluida en su secuencia completa de ADN– dicta no sólo la naturaleza de las proteínas celulares, sino también cuándo y dónde se sintetizarán.

1.2. La evolución de los seres vivos

Los organismos vivos pueden clasificarse en dos grupos atendiendo a su estructura: los organismos **eucariotas** y los **procariotas**. Los eucariotas guardan su ADN en un compartimiento intracelular denominado núcleo. Los procariotas no presentan un comportamiento nuclear diferenciado para almacenar su

ADN. Las plantas, los hongos y los animales son eucariotas; las bacterias son procariotas [5].

Tanto durante el almacenamiento como durante el copiado del material genético se pueden producir accidentes y/o errores aleatorios que pueden alterar la secuencia de nucleótidos –es decir, generar **mutaciones**–. Como consecuencia de ello, cuando una célula se divide, a menudo sus dos células hijas no son idénticas entre sí o a su progenitora. Algunas veces poco frecuentes, el error puede representar un cambio favorable; más probablemente, el error no supondrá diferencias importantes en las capacidades de la célula; y en muchos casos, el error causará daños importantes –por ejemplo, alterando la secuencia de una proteína clave–. Cambios debidos a errores del segundo tipo pueden ser o no perpetuados, dependiendo de si la célula o sus familiares tienen o no éxito en la competencia por los recursos limitados del ambiente donde viven. Los cambios que causan daños importantes no conducen a la célula a ninguna parte, por lo general provocan su muerte, y por tanto, no dejan descendencia. Mediante la repetición de este ciclo de ensayo y error –de *mutación* y *selección natural*– los organismos van evolucionando y sus especificaciones genéticas van cambiando, proporcionándoles nuevas vías de aprovechamiento del entorno más eficaces para poder sobrevivir en competencia con otros organismos, reproduciéndose con más éxito.

Claramente, durante el curso de la evolución, algunas partes del genoma cambian más fácilmente que otras. Un segmento de ADN que no codifica ninguna proteína y que no juega ningún papel regulador importante puede cambiar a una velocidad limitada únicamente por la frecuencia de los errores aleatorios que sufra. Por el contrario, un gen que codifique una proteína esencial altamente optimizada o una molécula de ARN no puede ser alterado tan fácilmente: cuando se produce un error, la célula afectada casi siempre es eliminada.

El material básico de la evolución es la secuencia de ADN que ya existe. No hay ningún mecanismo natural por el que se generen grandes cadenas de ADN de secuencia nueva aleatoria. Así, ningún ADN es completamente nuevo. La innovación puede ocurrir por diversas vías:

- *Mutación intragénica:* un gen ya existente puede ser modificado por mutaciones en su secuencia de ADN.

- *Duplicación génica:* un gen existente puede ser duplicado, generando así un par de genes muy relacionados en una misma célula.

- *Mezcla de fragmentos:* dos o más genes existentes pueden romperse y reagruparse generando un gen híbrido formado por segmentos de ADN que originariamente pertenecían a genes independientes.

- *Transferencia horizontal (intracelular):* un fragmento de ADN puede ser transferido desde el genoma de una célula al de otra célula –incluso de una especie diferente–. Este proceso contrasta con la *transferencia vertical* de información genética, habitual entre los progenitores y la progenie.

Una célula ha de duplicar todo su genoma cada vez que se divide en dos células hijas. Sin embargo, algunos accidentes pueden causar la duplicación de una parte del genoma, manteniendo el genoma original. Cuando un gen se ha duplicado por esta vía, una de las dos copias queda libre para mutar y especializarse en la realización de una función diferente en la misma célula. Repetidos ciclos de este proceso de duplicación y divergencia, durante millones de años, han permitido que algunos genes generen una familia completa de genes en un mismo genoma.

Cuando los genes se duplican y divergen de esta manera, los individuos de una especie resultan dotados de diferentes variantes del gen inicial. Este proceso evolutivo ha de distinguirse de la divergencia genética que ocurre cuando una especie se separa en dos líneas de descendencia diferentes en una bifurcación del árbol de la vida. En este punto, los genes se vuelven diferentes en el curso de la evolución, pero continúan teniendo funciones correspondientes en las dos especies hermanas. A los genes que están relacionados de esta forma –es decir, genes de dos especies separadas que derivan de un mismo gen ancestral presente en el último ancestro común de ambas especies– se los denomina **ortólogos**. A los genes relacionados que derivan de una duplicación en el mismo genoma –y que posiblemente divergirán en sus funciones– se los denomina **parálogos**. A los genes que están relacionados por una descendencia de cualquier tipo se los denomina **homólogos**, un término general que se utiliza para englobar ambos tipos de relación.

Sin embargo, los intercambios horizontales de la información genética juegan un papel muy importante en la evolución bacteriana en el mundo actual. La reproducción sexual genera una transferencia horizontal de información genética a gran escala entre dos linajes celulares inicialmente separados –los de la madre y del padre–. Independientemente de si esto ocurre entre especies o dentro de una misma especie, la transferencia horizontal de genes deja una huella característica: genera individuos que están más relacionados entre sí con un grupo de parientes con respecto a determinados genes y con otros con respecto a otro grupo de genes.

Actualmente se cuenta con mucha información sobre muchos genomas, sus secuencias completas y las funciones de muchos genes conocidos. En muchos casos, ya se habrá determinado experimentalmente la función de uno o más de estos homólogos, y como la secuencia génica determina la función del gen, frecuentemente podremos realizar una buena predicción de la función del nuevo gen: es probable que sea similar a la de sus homólogos.

Los organismos vivos son muy complejos, por lo que cuanto más aprendemos acerca de una especie cualquiera, más atractiva se vuelve como objetivo de estudios posteriores. Cada descubrimiento genera nuevas cuestiones y proporciona nuevas herramientas con las que podemos afrontar las preguntas en el organismo seleccionado. Por esta razón, muchos grupos de investigación han dirigido sus esfuerzos a analizar diferentes aspectos de un mismo **organismo modelo**.

En el enormemente variado mundo de las bacterias, el foco de atención de la biología molecular ha sido la bacteria *Escherichia coli* o *E. coli*, la cual será utilizada en esta tesis.

1.3. Separación, clonación y secuenciación de ADN

Hasta principios de los años setenta, el ADN era la molécula de la célula que planteaba más dificultades para su análisis bioquímico. Actualmente, el ADN a pasado a ser la macro-molécula más fácil de estudiar. Ahora podemos separar una región determinada del ADN, obtener un número de copias casi ilimitado y determinar su secuencia de nucleótidos.

Estos adelantos técnicos en la ingeniería genética han tenido un impacto espectacular en la biología celular, permitiendo el estudio de las células y de sus macro-moléculas mediante sistemas que antes eran inimaginables. La tecnología del ADN recombinante constituye un conjunto variado de técnicas, algunas de las cuales son nuevas y otras han sido adoptadas de otros campos de la ciencia, como la genética microbiana. Las más importantes son:

- La rotura específica del ADN mediante nucleasas de restricción, que facilita enormemente el aislamiento y la manipulación de los genes, individualmente.

- La clonación del ADN, con el uso de vectores de clonación o de la reacción en cadena de la polimerasa, de tal forma que una molécula sencilla de ADN puede ser reproducida generando muchos miles de millones de copias idénticas.

- La hibridación de los ácidos nucleicos, que hace posible localizar secuencias determinadas de ADN o de ARN con una gran exactitud y sensibilidad, utilizando la capacidad que tienen estas moléculas de unirse a secuencia complementaria de otros ácidos nucleicos.

- La secuenciación rápida de todos los nucleótidos de un fragmento purificado de ADN, que hace posible identificar genes y deducir la secuencia de aminoácidos de las proteínas que codifican.

- El seguimiento simultáneo del nivel de expresión de cada uno de los genes de una célula, utilizando microchips de ADN (microarrays) que permiten efectuar simultáneamente decenas de miles de reacciones de hibridación.

A continuación describiremos en más detalle este último ítem, que luego será utilizando en los capítulos posteriores.

Como ya hemos comentado, las técnicas clásicas para el análisis de secuencias permiten examinar la expresión de un solo gen a la vez. Muchos de estos métodos son bastante laboriosos y el análisis es limitado por el número de muestras que pueden cargarse a la vez en un gel de agarosa. Los *microarrays*, desarrollados en los años noventa, han revolucionado la forma en la que actualmente se estudia la expresión génica, al permitir el estudio de los productos de ARN de miles de genes a la vez. Al poder analizar simultáneamente la expresión de tantos genes, hoy en día podemos empezar a identificar y estudiar los patrones de expresión génica que subyacen a la fisiología celular: podemos ver qué genes se encuentran activados (o inhibidos) durante el crecimiento de la célula, o cuáles responden a hormonas o a toxinas.

Los microarrays de ADN no son más que portaobjetos de microscopía recubiertos con un gran número de fragmentos de ADN, cada uno de los cuales contiene una secuencia de nucleótidos que actúa como sonda para un gen concreto. Los microarrays más densos contiene decenas de miles de estos fragmentos en un área menor que el tamaño de un sello de correos y permiten desarrollar en paralelo miles de reacciones de hibridación.

Para utilizar un microarray al estudiar la expresión de un gen, primero se extrae el ARNm de las células que estemos estudiando y se transforma en ADNc. Éste se marca con un producto fluorescente, el microarray se incuba con esta muestra de ADNc marcado y se deja que tenga lugar la reacción de hibridación. El microarray se lava para eliminar el ADNc que no se haya unido y se identifican las posiciones del microarray a las que se han unido los fragmentos de ADN marcado con un microscopio que tiene un escáner láser automatizado. Las posiciones detectadas por el escáner se hacen corresponder con el gen concreto que habíamos dispuesto inicialmente en esa posición.

Habitualmente el ADN fluorescente de las muestras experimentales (marcado, por ejemplo, con un pigmento rojo fluorescente) se mezcla con una muestra de fragmentos de ADNc de referencia marcados con un pigmento fluorescente de otro color (por ejemplo, de color verde) (ver Figura 1.8). Por tanto, si la cantidad de ARN expresada por un gen determinado de las células de interés se encuentra incrementada con respecto a la de la muestra de referencia, la mancha resultante es roja. Por el contrario, si la expresión del gen es menor en relación a la muestra de referencia, la mancha resultante será verde. La utilización de este tipo de referencia interna permite tabular los patrones de expresión de los genes con gran precisión.

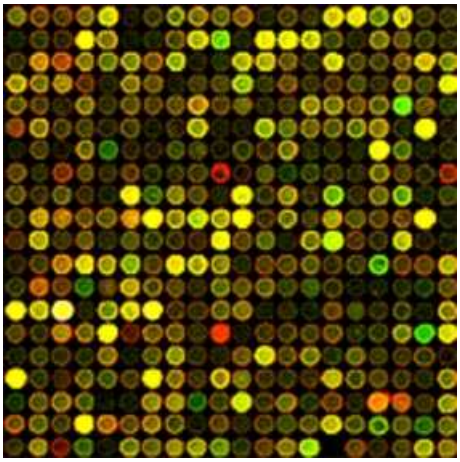


Figura 1.8: Microarray

1.4. Biología computacional y Bioinformática

En las últimas décadas, los avances en la Biología Molecular y el equipamiento disponible para la investigación en este campo han permitido la rápida secuenciación de grandes porciones de genomas de diversas especies. En la actualidad, varios genomas de bacterias, tales como *Saccharomyces cerevisiae*, y algunos eucariotas simples ya han sido secuenciados por completo. El proyecto Genoma Humano [24], diseñado con el fin de secuenciar los 24 cromosomas del ser humano, también está progresando. Las bases de datos de secuencias más populares, como GenBank [16] y EMBL [55], están creciendo de forma exponencial. Esta gran cantidad de información necesita de un alto nivel de organización, indexado y almacenamiento de las secuencias. Es por ello que la Informática ha sido aplicada a la Biología para producir un nuevo campo de investigación llamado *Bioinformática* que permita ayudar a esta organización [9].

1.4.1. Objetivos de la Bioinformática

Las tareas más simples de la Bioinformática conciernen a la creación y mantenimiento de bases de datos de información biológica. Secuencias nucleotídicas (y las secuencias proteicas que derivan de las mismas) componen la mayoría de la información que está almacenada en estos repositorios. Mientras que el almacenamiento y organización de millones de nucleótidos está muy lejos de ser una tarea trivial, el diseño de una base de datos y el desarrollo de una interfaz con la cual los investigadores puedan tanto acceder a la información existente

como agregar nuevas instancias, es simplemente el comienzo.

Tal vez, la tarea más apremiante sea la que involucra el análisis de la información de secuencias. **Biología Computacional** es el nombre dado a este proceso e involucra las siguientes tareas:

- Encontrar genes en secuencias de ADN pertenecientes a varios organismos.
- Desarrollar métodos para la predicción de la estructura y/o la función de nuevas proteínas y secuencias estructurales de ARN.
- Agrupar secuencias de proteínas en familias de secuencias relacionadas y el desarrollo de modelos de proteínas.
- Alinear proteínas similares y generar árboles filogenéticos para examinar las relaciones de la evolución.

El proceso de evolución ha producido secuencias de ADN que codifican proteínas con funciones muy específicas. Es posible predecir la estructura tridimensional de una proteína usando algoritmos derivados de nuestros conocimientos en el campo de la Física, la Química y, en mayor medida, del análisis de otras proteínas con secuencias de aminoácidos similares.

La mayoría de las bases de datos biológicas consisten en largas secuencias nucleotídicas y/o secuencias de aminoácidos. Cada secuencia representa un gen o proteína particular (o una sección de la misma), respectivamente. Mientras que la mayoría de las bases de datos biológicas contienen este tipo de información, también existen otros repositorios que incluyen información taxonómica tales como características estructurales o bioquímicas de los organismos. Sin embargo, el poder y la facilidad de utilizar la información de secuencias hizo que ése sea el método elegido en el análisis moderno.

En las últimas tres décadas, las contribuciones al área de la Biología y de la Química han facilitado el aumento en la velocidad del proceso de secuenciación de genes y proteínas. El advenimiento de la tecnología de clonación ha permitido que secuencias de ADN foráneas sean introducidas en bacterias. De esta manera fue posible la rápida producción de secuencias de ADN particulares, un preludio necesario para la determinación de secuencias. La síntesis de oligonucleótidos dio a los investigadores la habilidad de construir pequeños fragmentos de ADN con secuencias elegidas por ellos mismos. Estos oligonucleótidos son luego utilizados como parte de bibliotecas de ADN y permiten la extracción de genes que contengan esta secuencia. Estos fragmentos de ADN también pueden ser utilizados en reacciones en cadena de polimerización para amplificar secuencias de ADN o modificar estas secuencias. Mediante estas técnicas, el progreso de la investigación biológica ha crecido exponencialmente.

Sin embargo, para que los investigadores puedan beneficiarse de esta información, es necesario cumplir con dos requisitos: (1) tener acceso inmediato al

conjunto de secuencias coleccionadas y (2) tener una forma de extraer de este conjunto solamente aquellas secuencias que interesen al investigador. La simple colección, de forma manual, de toda la información necesaria para un proyecto dado a partir de un artículo de revista publicado puede convertirse rápidamente en una tarea epopéyica. Luego de obtener estos datos, es necesario organizarlos y analizarlos. La búsqueda manual de genes y proteínas relacionadas puede llevarle semanas e incluso meses a un investigador.

La tecnología informática ha proporcionado la solución obvia a este problema. Los ordenadores, no solo pueden acumular y organizar la información de secuencias en bases de datos, sino que también pueden analizar los datos de las secuencias muy rápidamente. La evolución del poder computacional y la capacidad de almacenamiento ha logrado lidiar con la creciente cantidad de información de secuencias que está siendo creada. Los científicos teóricos han desarrollado sofisticados algoritmos que permiten comparar secuencias mediante teoría de probabilidades. Estas comparaciones se han convertido en la base de la determinación de la función de genes, desarrollando relaciones filogenéticas y simulando modelos de proteínas.

La colección, organización e indexado de la información de secuencias en una base de datos es una tarea desafiante por sí misma y ha generado una gran cantidad de información pero de uso limitado. El poder de una base de datos no proviene de la colección de información que tenga, sino de su análisis. Una secuencia de ADN no necesariamente constituye un gen, puede constituir solamente un fragmento de un gen o contener varios genes.

La investigación científica actual, en acuerdo con los principios de la evolución, muestra que todos los genes tienen elementos comunes. Para muchos elementos genéticos es posible construir secuencias consenso, las cuales representan de la mejor manera posible la norma de una clase dada de organismo. Algunos elementos genéticos comunes incluyen promotores, reforzadores, señales de poliadenización y sitios de binding de proteínas. Para estos elementos también se conocen algunas características de sus subelementos.

Los elementos genéticos comunes comparten secuencias similares, siendo éste el hecho que permite la aplicación de algoritmos al análisis de secuencias biológicas.

1.4.2. Problemas clásicos de la bioinformática

A continuación se enunciarán algunos problemas clásicos de la bioinformática [85].

1.4.2.1. Reconocimiento de patrones

> *"Dadas una secuencia biológica y una base de datos de estas mismas secuencias, indicar qué secciones de la secuencia dada tienen alguna*

similitud con aquellas existentes en el repositorio."

Los métodos de reconocimiento de patrones, como sugiere su nombre, se basan en el supuesto de que alguna característica subyacente de una secuencia o estructura biológica puede emplearse para identificar caracteres semejantes en secuencias emparentadas. En otras palabras, si se mantiene o conserva parte de una secuencia o estructura, esta característica puede emplearse para determinar nuevos miembros de la familia. Si se extraen tales caracteres conservados de familias de secuencias conocidas y se archivan en bases de datos, entonces las secuencias recién secuenciadas pueden ser analizadas rápidamente para determinar si contienen esas características familiares previamente reconocidas. En la actualidad, se emplean de forma habitual búsquedas en bases de datos de patrones de secuencias y estructuras para el diagnóstico de relaciones familiares y, en consecuencia, para inferir estructuras y funciones de secuencias recién determinadas.

1.4.2.2. El problema de la búsqueda de genes

"Dada una secuencia de ADN, predecir la ubicación de genes (marcos de lectura abiertos), exones e intrones."

Una solución simple podría ser buscar los codones de parada en regiones a lo largo de la secuencia. Claramente, si varios codones de parada aparecen cerca unos de otros en esta región, entonces es muy probable que la secuencia correspondiente al gen ya hubiera terminado, por ello se puede asumir con seguridad que no es una región codificante. Detectar una secuencia relativamente larga que no contenga codones de parada indica una región codificante. El problema se complica en ADN eucariótico debido a la existencia de exones e intrones. Por otro lado, una complejidad adicional surge del hecho que ciertas secuencias de ADN pueden ser interpretadas de seis maneras diferentes dependiendo del marco de lectura, como ya se ha mencionado. En la mayoría de los casos, en organismos eucariotas, una región de ADN codifica sólo un gen, lo cual puede no ser necesariamente cierto para procariotas.

1.4.2.3. El problema del alineamiento de secuencias

"Dadas dos secuencias de ADN o de proteínas, encontrar el mejor emparejamiento entre ellas."

Para poder realizar un alineamiento, es necesario definir un conjunto de operaciones posibles con sus correspondientes penalidades. Por ejemplo, un fenómeno biológico como la inserción puede ser matemáticamente traducido en una acción de abrir un espacio (gap), lo cual acarrea una penalización. De esta manera, se pueden caracterizar otras propiedades como las deleciones, errores en el alineamiento, desplazamientos de marco, etc., cada una con su penalidad

específica de acuerdo a su frecuencia biológica y su gravedad. Este problema también es extensible a múltiples secuencias.

1.4.2.4. El problema del rearreglo genómico

> *"Dadas dos permutaciones de un conjunto de segmentos genómicos, encontrar el conjunto mínimo de operaciones para transformar una permutación en otra."*

Los eventos de reorganización son raros comparados con mutaciones puntuales. Por ejemplo, una sustitución en un organismo ocurre diez veces en cada generación, mientras que un rearreglo no fatal ocurre una vez cada 5 a 10 millones de años. Ésta baja tasa de rearreglos permite detectar un proceso evolutivo direccional, dado que la posibilidad de reversión es mínima. Por lo tanto, descubrir qué eventos de reorganización han ocurrido, y el orden de estos eventos, puede permitir construir una hipótesis evolutiva.

1.4.2.5. El problema del plegado de proteínas

Debido a que la funcionalidad de una proteína está determinada por su estructura 3D, es muy importante predecir su estructura, con lo que se consigue un mejor entendimiento de su rol en la célula.

> *"Dada una secuencia de aminoácidos, predecir la estructura tridimensional de la proteína."*

Este problema aún no ha sido resulto, es decir, es un problema abierto y en el cual se está trabajando continuamente. Sin embargo, se han desarrollado varios acercamientos para aproximar la estructura de una proteína:

- *Modelado por homología.* Utiliza una base de datos de proteínas para buscar secuencias similares. Si se encuentra una proteína que tiene una identidad de aproximadamente un 30 %, entonces es bastante seguro asumir que las dos proteínas tienen estructuras similares.
- *Threading.* Clasifica las estructuras conocidas en familias con plegados similares. Dada una secuencia de aminoácidos, se puede seleccionar la familia a la cual es más probable que pertenezca la proteína.

1.4.2.6. El problema de la regulación de genes

> *"Dado un conjunto de genes de un genoma, obtener la red de interacciones entre este conjunto, determinando cómo, cuándo y qué genes se activan o reprimen."*

Las redes de regulación de genes son básicamente interruptores de encendido-apagado de una célula que funcionan al nivel de genes. Éstos orquestan dinámicamente el nivel de la expresión para cada gen en el genoma, controlando cuándo y cómo ese gen será transcrito en ARN.

La comprensión de los mecanismos de función celular requiere un estudio exhaustivo del complejo comportamiento de un conjunto de genes interactuando. Actualmente, el modelo formal más usado y aceptado es la red booleana. En este modelo, los estados de un gen se representan por "1" si el gen está activado y "0" si no. La dependencia entre los genes se representa por un enlace funcional entre ellos, y el estado de cada gen es determinado por sus entradas y una función booleana asociada.

A pesar de su aparente simplicidad, las redes de regulación genética son procesos complejos, en los cuales forman parte muchos componentes que afectan directa o indirectamente en la regulación de la expresión.

Capítulo 2

Preliminares

En este capítulo se introducirán los conceptos básicos sobre diferentes métodos y algoritmos computacionales que constituirán la base sobre la cual se apoyará nuestro trabajo. Los temas que se explicarán en las próximas secciones corresponden a los conceptos de *modelado de sistemas*, *lógica difusa*, *algoritmos de clustering (agrupamiento)*, *algoritmos evolutivos*, en particular *algoritmos genéticos* y la *programación genética*, y, finalmente, *técnicas de optimización multiobjetivo*.

2.1. Modelado de sistemas

Cada vez que resolvemos un problema correspondiente al mundo real, debemos comprender que, en realidad, solamente estamos encontrando la solución de un modelo del problema [65]. Todos los modelos son simplificaciones del mundo real, de otro modo serían tan complejos e intrincados como lo es el escenario natural. El problema que existe con los modelos es que cada uno de ellos debe hacer una serie de asunciones respecto del problema real, en otras palabras, dejar algo de éste a un lado. En consecuencia, el error más frecuente que existe cuando tratamos con modelos es olvidar dichas asunciones. En este sentido, un problema puede tener varias soluciones correctas posibles, tan solo dependientes de cómo fue construido el modelo del mismo.

El proceso de resolución de un problema consiste básicamente en dos pasos generales que se dan por separado:

- Creación de un modelo del problema.
- Utilización de ese modelo para la generación de una solución:

$$Problema \Rightarrow Modelo \Rightarrow Solución$$

Tal como hemos dicho, la solución de un problema está dada solamente en términos del modelo y depende de su grado de fidelidad. Esto es, confiamos en que el mejor modelo produzca la mejor solución. Sin embargo, algunas veces los modelos más perfectos resultan inútiles para que un método de solución preciso decida qué hacer, es decir, permita derivar una solución basada en dicho modelo. En esta memoria, trataremos con modelos ($Modelo_\alpha$) y soluciones ($Solución_\alpha$) aproximadas, con el propósito de representar y resolver de la mejor manera posible cierta clase de problemas biológicos para los cuales, debido a su nivel de complejidad e imprecisión, resulta difícil obtener una solución adecuada.

En este sentido, la construcción de modelos basados en la lógica difusa (ver Sección 2.2) ha adquirido un gran interés gracias a que proporciona mayor generalidad, poder expresivo y tolerancia frente a la imprecisión.

El desarrollo de modelos matemáticos de sistemas reales es un tópico central en diferentes disciplinas como la ingeniería y las ciencias. Como hemos mencionado, dichos modelos sirven para resolver problemas reales a través de simulaciones, análisis del comportamiento de un sistema, diseño de nuevos procesos para controlar sistemas, predicciones, etc. El desarrollo inadecuado de dichos modelos puede conducir a la obtención de resultados no exitosos y erróneos de los sistemas en cuestión.

Gran parte de estos sistemas actuales comparten una serie de características que dificultan su modelado con técnicas tradicionales, tales como: necesidad de una importante precisión, tratamiento y respuestas en tiempo real, carencia de conocimiento formal sobre su funcionamiento, posesión de un comportamiento fuertemente no lineal, con alto grado de incertidumbre y características con variaciones en el tiempo. Ejemplos de esta clase de problemas son los procesos complejos de los sistemas de ingeniería aeroespacial o bioquímica, aunque también se encuentran en sistemas ecológicos, sociales o correspondientes al área económico-financiera.

A continuación plantearemos al menos tres paradigmas diferentes en cuanto a la tarea de modelado, es decir, de la comprensión de la naturaleza y el comportamiento de un sistema, y la consecuente tarea de creación de un modelo que pueda ser posteriormente utilizado [10]:

- *Modelado de caja blanca.* Tradicionalmente, el modelado de datos se ha visto como una conjunción de una actividad que permite la comprensión de la naturaleza del sistema y su comportamiento, y de un tratamiento matemático adecuado que conduce hacia la realización de un modelo utilizable. Este enfoque, denominado *modelado de "caja blanca"*, está limitado en la práctica cuando se emplean sistemas complejos y poco comprensibles. Como consecuencia directa de este planteamiento, los sistemas no lineales se simplifican, pasando a ser tratados como sistemas lineales, debido a la imposibilidad de comprender su mecanismo de funcionamiento.

En vista del diagrama previamente empleado, observamos esta situación de modelado como:

$$Problema \Rightarrow Modelo_{\alpha} \Rightarrow Solución_{p}$$

lo que corresponde a solucionar un modelo simplificado (α) de un problema por medio de un método (p) que produce soluciones óptimas y precisas.

- *Modelado de caja negra.* Este es un planteamiento diferente que consiste en estimar los parámetros del modelo en base a los datos de entrada disponibles. A pesar de que estos modelos generalmente pueden ser desarrollados de forma bastante fácil, la estructura identificada no posee un significado físico claro. Estos modelos resultan difícilmente escalables y el comportamiento real del sistema puede ser solamente analizado a través de la simulación numérica. Un inconveniente común de la mayoría de estas técnicas de modelado es que no permiten el uso efectivo de información adicional, tal como conocimiento y experiencia de expertos, y operaciones previas, las cuales son imprecisas y cualitativas por naturaleza. Nuevamente, esto puede ser visto como:

  $$Problema \Rightarrow Modelo_{p} \Rightarrow Solución_{\alpha}$$

  Es decir, un modelo muy preciso solucionado por un método cercano al óptimo, por tanto, una solución aproximada.

- *Modelado de caja gris.* Los hechos descritos anteriormente motivan el desarrollo de técnicas de *modelado híbrido o de "caja gris"*, que combinan las ventajas de los enfoques de tipo caja blanca, utilizando conocimiento físico para modelar las partes conocidas del sistema, y de los de caja negra, para aproximar las partes más inciertas. Para realizar esto, se han introducido algunas metodologías inteligentes, las cuales emplean técnicas motivadas por sistemas biológicos e inteligencia humana (lenguaje natural, reglas, redes semánticas o modelos cualitativos) para desarrollar modelos y controladores para sistemas dinámicos.

  En consecuencia, obtenemos una solución aproximada de un problema en base a un modelo aproximado, con el propósito de ser interpretable y tan preciso como se pueda. Podemos distinguir así varias clases de modelos aproximados, esto es, con distinto grado de precisión (β):

  $$Problema \Rightarrow Modelo_{\beta} \Rightarrow Solución_{\alpha}$$

  Debemos señalar que un modelo aproximado de un problema no necesariamente es una simplificación del mismo, sino que por el contrario puede

ser una descripción aún más precisa que la dada por un modelo de caja blanca. Esto se debe a que estos modelos son capaces de captar la imprecisión e incertidumbre de los problemas del mundo real, que muchas veces son asunciones o excepciones no tenidas en cuenta en el otro caso.

2.2. Lógica difusa

La lógica difusa es una teoría de reciente aparición que se atribuye al investigador Lofti Zadeh. En lo que se considera como el trabajo seminal de la teoría de conjuntos difusos [92] y su posterior aportación sobre el concepto de variable lingüística [93], este investigador proporcionó las bases para el entendimiento y tratamiento de la incertidumbre de forma cualitativa o mediante términos lingüísticos.

Con carácter general, la representación del conocimiento ha sido una de las áreas de mayor interés investigadas en la disciplina de las ciencias de la computación y la inteligencia artificial. Uno de los principales aspectos tratados es el de cómo representar el conocimiento que es lingüísticamente impreciso, para cuya representación se han mostrado ineficaces las técnicas convencionales. Debemos ser conscientes de que los métodos que tradicionalmente se utilizaban para tratar la información hacían uso de datos precisos (cuantitativos), intentando aportar una visión predictiva basada en procesos deterministas. Sin embargo, no resultan útiles cuando se aplican como apoyo de procesos de razonamiento que cuentan con información incierta o definida de manera imprecisa. En este sentido, el desarrollo de la lógica difusa se vio motivado por la necesidad de un marco conceptual que pudiese aplicarse con éxito al tratamiento de la información en entornos de incertidumbre e imprecisión léxica [94].

La lógica difusa pude ser vista como una extensión a la lógica clásica, donde se incorporan nuevos conceptos para trabajar con el problema de representación en un ambiente de incertidumbre e imprecisión. La lógica difusa, como su nombre sugiere, es una forma de lógica cuya forma de razonamiento subyacente es más aproximada que exacta. La diferencia fundamental entre las proposiciones de la lógica clásica y las proposiciones difusas está en el rango de valores de verdad. Mientras que en las proposiciones clásicas sólo existen dos posibles valores de verdad (verdadero o falso), el grado de verdad o falsedad de las proposiciones difusas puede tomar distintos valores numéricos. Asumiendo que la verdad y la falsedad se representan con 1 y 0 respectivamente, el grado de verdad de cada proposición difusa se expresa como un valor en el intervalo [0,1]. La lógica difusa es, en realidad, una forma de lógica multivaluada. Su finalidad última es proveer de una base para el *razonamiento aproximado* con proposiciones imprecisas utilizando la teoría de conjuntos difusos como herramienta principal.

El paso de la lógica clásica a la difusa que acabamos de comentar tiene serias implicaciones, como no puede ser de otra manera, sobre la teoría de conjuntos.

Si un conjunto se utiliza para clasificar los elementos de un universo de estudio, en determinadas situaciones (aquellas no deterministas o definidas de manera vaga) no se deben obtener los mismos resultados si se utilizan los principios de la lógica clásica a si se utilizan los de la lógica difusa.

De forma previa a la introducción de los conjuntos difusos, partamos de lo que se entiende por un conjunto clásico, también denominados como conjuntos *crisp*, mediante la siguiente definición [72]: un conjunto clásico A del universo de discurso o dominio X, puede representarse haciendo uso de la siguiente función característica $\mu_A : X \rightarrow 0, 1$ del conjunto A si y solo si para todo x:

$$\mu_A(x) = \begin{cases} 1 & si\ x \in A \\ 0 & si\ x \notin A \end{cases} \tag{2.1}$$

donde X es el universo de discurso y A un conjunto definido en dicho discurso. Como se puede observar, la función característica que define los conjuntos clásicos es un caso típico de función booleana, de verdadero o falso, expresado numéricamente como 1 ó 0. También se denomina como función discriminante porque discrimina los elementos del universo de discurso entre aquellos que pertenecen al conjunto definido y aquellos que no. Por tanto, se tratan de conjuntos de elementos cuyos límites están fijados de forma determinista.

Por tanto, de manera previa a la resolución de la pregunta que planteamos, definamos lo que se entiende por un conjunto difuso [72]: un conjunto difuso A en el universo de discurso X es un conjunto de pares ordenados de un elemento genérico x y su correspondiente grado de pertenencia $\mu_A(x)$, de manera que $A = \{(x, \mu_A(x)) \ / \ x \in X\}$. Por tanto, como puede apreciarse, la definición de un conjunto difuso es similar a la de un conjunto clásico, con la sustancial salvedad de que cada elemento perteneciente al universo de discurso tiene asociado un *grado de pertenencia* a dicho conjunto difuso. Por tanto, los conjuntos crisp se pueden considerar como un caso específico de conjuntos difusos, en tanto que $0, 1 \in [0, 1]$.

Existen siete tipos de funciones de pertenencia típicas: Triangular, Gamma, S, Gaussiana, Trapezoidal, Pseudo-exponencial y de Trapecio extendido. Las funciones más utilizadas son de la clase triangular, trapezoidal y gaussiana. Las dos primeras se basan en un comportamiento lineal de la función de pertenencia, mientras que la gaussiana se caracteriza por un comportamiento no lineal de dicha función.

Es incorrecto asumir que un conjunto difuso indica, de alguna manera, alguna forma de probabilidad. A pesar del hecho que pueden llegar a tomar valores en un mismo intervalo de definición, es importante comprender que los grados de pertenencia *no* son probabilidades. Una diferencia aparentemente inmediata es que la suma de las probabilidades en un conjunto universal debe ser 1, mientras esto no es un requisito para los conjuntos difusos.

Ya hemos destacado que los conjuntos difusos son extensiones o generalizaciones de los conjuntos clásicos o crisp de la lógica bivaluada. Por tanto, las

mismas operaciones que se determinan para los conjuntos crisp pueden determinarse igualmente para los conjuntos difusos, con el añadido de que existen unos grados de pertenencia asociados. Asimismo, los conjuntos resultantes de las operaciones con conjuntos difusos son también difusos. Las principales operaciones básicas que se van a presentar son: la igualdad, la inclusión, la intersección o conjunción, la unión, y el complemento.

- *Igualdad.* Para que se considere que dos conjuntos difusos son iguales, se deberá satisfacer la condición siguiente:

$$A = B \Leftrightarrow \forall x \in X \ ; \ \mu_A(x) = \mu_B(x) \tag{2.2}$$

- *Inclusión.* Se considera que un conjunto difuso A es un subconjunto de B si:

$$A \subseteq B \Leftrightarrow \forall x \in X \ ; \ \mu_A(x) \leq \mu_B(x) \tag{2.3}$$

- *Intersección.* La intersección entre dos conjuntos difusos se puede obtener mediante el mínimo de ambos:

$$\mu_{A \cap B}(x) = min\{\mu_A(x), \mu_B(x)\} \tag{2.4}$$

- *Unión.* Por el contrario, para la unión de dos conjuntos difusos se considera el máximo en los grados de pertenencia de los elementos del universo de discurso:

$$\mu_{A \cup B}(x) = max\{\mu_A(x), \mu_B(x)\} \tag{2.5}$$

- *Complemento.* El complemento, $\bar{A}$, de un conjunto difuso A con respecto al conjunto universal X se define para todo elemento $x \in X$ como:

$$\bar{A}(x) = 1 - A(x) \tag{2.6}$$

Por otro lado, existen generalizaciones de las operaciones anteriores, puesto que tanto las funciones de pertenencia de los conjuntos difusos como sus operaciones dependen del contexto en el que se apliquen. En este respecto, para poder aplicar la lógica difusa en un sistema informático basado en reglas es preciso que se pueda trabajar con los operadores "Y" y "O", es decir. la intersección y la unión respectivamente. La familia de funciones que se utilizan para tal fin se conocen como *T-normas* y *T-conormas*.

Una *T-norma* generaliza el concepto de intersección de forma que

$$T: \ [0,1] \times [0,1] \rightarrow [0,1] \tag{2.7}$$
$$\mu_{A \cap B} = T[\mu_A(x), \mu_B(x)] \tag{2.8}$$

y además satisface las siguientes propiedades:

- *Conmutativa*: $T(a,b) = T(b,a)$
- *Asociativa*: $T(a, T(b,c)) = T(T(a,b), c)$
- *Monotonía*: $T(a,b) \geq T(c,d), \ si \ a \geq c \ y \ b \geq d$
- *Condiciones frontera*: $T(a,1) = a$

En segundo lugar, la *T-conorma*, también conocida como S-norma, generaliza el concepto de unión de forma que

$$S: \ [0,1] \times [0,1] \rightarrow [0,1] \tag{2.9}$$
$$\mu_{A \cup B} = S[\mu_A(x), \mu_B(x)] \tag{2.10}$$

y además satisface las siguientes propiedades:

- *Conmutativa*: $S(a,b) = S(b,a)$
- *Asociativa*: $S(a, S(b,c)) = S(S(a,b), c)$
- *Monotonía*: $S(a,b) \geq S(c,d), \ si \ a \geq c \ y \ b \geq d$
- *Condiciones frontera*: $S(a,0) = a$

En resumen, la utilización de funciones triangulares para la intersección y/o unión de conjuntos difusos ofrece un abanico bastante extenso para realizar estas operaciones, y, por tanto, para calcular las T-normas y T-conormas en las que se basarán las funciones de pertenencia resultantes. Asimismo, su flexibilidad permite que el investigador proponga sus propias fórmulas con el objeto de adaptarse mejor a las características de los conjuntos difusos con los que trabaje.

2.3. Clustering

El objetivo del agrupamiento de datos (*clustering*) es la clasificación de objetos de acuerdo a similitudes entre ellos, para luego organizar estos datos en grupos. Las técnicas de clustering están incluidas entre los métodos de aprendizaje *no supervisado*, debido a que no utilizan conocimiento de identificadores de clases. La mayoría de los algoritmos de clustering tampoco se basan en asunciones comunes a los métodos estadísticos convencionales, tales como la distribución

estadística subyacente de los datos, y por ello son útiles en situaciones donde existe poco conocimiento sobre los mismos. El potencial de los algoritmos de clustering para revelar las estructuras subyacentes de los datos puede ser explotado no sólo para la clasificación y reconocimiento de patrones sino también para la reducción de la complejidad en modelado y optimización.

Las técnicas de clustering pueden aplicarse a datos que sean cuantitativos (numéricos), cualitativos (categóricos) o una mezcla de ambos. Los datos son típicamente observaciones de algún proceso físico.

Pueden formularse varias definiciones de cluster, dependiendo del número de objetivos del clustering. Generalmente, uno puede aceptar la visión de que un cluster es un grupo de objetos, los cuales son más similares entre sí que los miembros de otros clusters. El término "similaridad" debería ser entendido como la similaridad matemática, medida formalmente. En espacios métricos, la similaridad está definida comúnmente como una *norma* de distancia. La distancia puede medirse entre los vectores de datos en sí, o como la distancia de un vector de datos a algún objeto prototípico del cluster. Los prototipos no son usualmente conocidos de antemano y los algoritmos de clustering los buscan simultáneamente con la partición de los datos. Los prototipos pueden ser vectores de igual dimensión que la de los objetos de datos, pero se pueden también definir como objetos geométricos de "alto nivel", tales como subespacios lineales y no-lineales, o funciones.

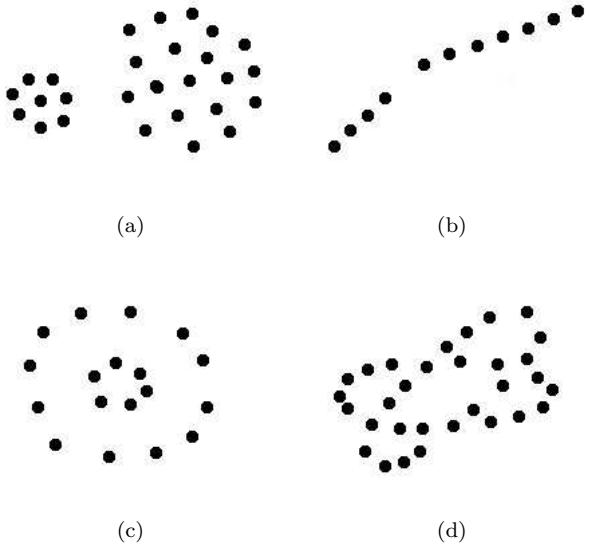


Figura 2.1: Diferentes formas de los clusters

Los datos pueden revelar clusters de diferentes formas geométricas, tamaños y densidades, como se puede ver en la Figura 2.1. Mientras que los clusters (a) son esféricos, los clusters (b), (c) y (d) pueden caracterizarse como subespacios lineales y no-lineales del espacio de datos. El rendimiento de la mayoría de los algoritmos de clustering está influenciado no solo por la forma geométrica y densidad de los clusters individuales, sino también por las relaciones espaciales y distancias entre ellos. Los clusters pueden estar bien separados, conectados en forma continua, o solapados entre ellos. La separación de los clusters está influenciada por el factor de escala y la normalización de los datos.

Se han propuesto muchos algoritmos de clustering en la literatura. Dado que los clusters pueden verse formalmente como subconjuntos del conjunto de datos, una posible clasificación de los métodos de clustering puede ser de acuerdo a si los subconjuntos son difusos o crisp. Los métodos de clustering crisp están basados en la teoría clásica de conjuntos y requieren que un objeto pertenezca o no a un cluster dado. Un clustering crisp significa particionar los datos en un número especificado de subconjuntos mutuamente excluyentes. Los métodos de clustering difuso, sin embargo, permiten a los objetos pertenecer a varios clusters simultáneamente, con distinto grado de pertenencia. En muchas situaciones, los clusters difusos son un concepto más natural que los clusters crisp, dado que los objetos en las fronteras de algunas clases no están forzados a pertenecer completamente a una de ellas. En lugar de ello, se les asignan grados de pertenencia entre 0 y 1 indicando su pertenencia parcial.

A continuación se mostrará como ejemplo el algoritmo de clustering difuso C-medias.

Clustering difuso C-medias. Este método de clustering es una extensión del método de clustering tradicional k-medias, donde los elementos pueden pertenecer a más de un cluster con distinto grado de pertenencia. Por ejemplo, el grado de pertenencia de una instancia k con un valor x_k a un cluster particular V_i se calcula como:

$$u_{i,k} = \left[1 + \left(\frac{\|x_k - V_i\|_A^2}{w_i}\right)^{\frac{1}{m-1}}\right]^{-1} \quad ; \quad \forall i,k;\ 1 < m \leq \infty \tag{2.11}$$

donde las c-particiones de los datos X se suelen almacenar en una matriz dimensión $c \times n$ que contiene el vector donde se representan los grados de similaridad entre las n instancias y las c-particiones de un tipo de característica; $u_{i,k}$ corresponde al grado de pertenencia del valor x_k en la partición difusa i-ésima de X; m representa el grado de imprecisión (*fuzzification degree*); A determina el tipo de norma utilizada, como puede ser la norma euclidiana ($A = 2$); y w_i es un peso para penalización de los términos, el cual se sustituye por 1 en ausencia de información externa.

Si el enfoque es probabilístico, $u_{i,k}$ generalmente corresponde a la probabilidad a posteriori $p(i|x_k)$ de que, dado x_k, provenga de la clase i siguiendo la

regla de Bayes [18, 19, 66]. Si el enfoque es difuso, x_k puede provenir de más de una clase.

El centroide del cluster de la partición V_i se calcula como:

$$V_i = \frac{\sum_{k=1}^{n} {u_{i,k}}^m x_k}{\sum_{k=1}^{n} {u_{i,k}}^m} \quad \forall i \tag{2.12}$$

fórmula basada en el uso de la distancia Euclídea como función de similaridad:

$$\|x - V\|_2 = \sqrt{(x-V)^T(x-V)} \tag{2.13}$$

En resumen, los pasos a seguir se muestran en el Algoritmo 2.1.

Algoritmo 2.1 Clustering difuso c-medias

C-MEDIAS(T número máximo de iteraciones, c número de particiones)

1: Inicializar $V_0 = \{v_1, \ldots, v_c\}$
2: **mientras** $t < T$ y $\|V_t - V_{t-1}\| > \epsilon$ **hacer**
3: Calcular U_t en base a V_{t-1}
4: Actualizar V_t en base a V_{t-1} y U_t
5: **fin mientras**

2.3.1. Clustering conceptual

El agrupamiento conceptual (conceptual clustering) es similar al considerado en el análisis de clusters tradicional, pero está definido de una manera diferente. Dados un conjunto de descripciones en base a atributos de ciertas entidades, un lenguaje de descripción para caracterizar clases de estas entidades y un criterio de calidad de clasificación; el problema consiste en particionar las entidades en clases de tal manera que se maximice el criterio de calidad de clasificación y, simultáneamente, en determinar descripciones generales de estas clases en el lenguaje de descripción dado. Por ello, un método de clustering conceptual busca no sólo una clasificación de las entidades, sino también una descripción simbólica de las clases propuestas (clusters). Un aspecto importante que distingue al clustering conceptual es que, a diferencia del análisis de clusters clásico, las propiedades de las descripciones de clases se toman en consideración en el proceso de determinación de las clases.

Para clarificar la diferencia entre el clustering conceptual y el clustering tradicional, nótese que un método de clustering convencional típicamente determina clusters en base a una medida de similaridad, que es una función definida únicamente utilizando las propiedades (valores de los atributos) de las entidades que están siendo comparadas:

$$Similaridad(A, B) = f(propiedades(A), propiedades(B)) \tag{2.14}$$

donde A y B son entidades a ser comparadas.

En contraste con esta metodología, las técnicas de clustering conceptual agrupan entidades en base a la *cohesión de conceptos*, la cual es una función no sólo de las propiedades (valores de los atributos), sino también de otros dos factores: el *lenguaje de descripción* L, usado por el sistema para describir las clases de entidades, y el *entorno* E, el cual es el conjunto de ejemplos vecinos:

$$CohesiónConceptual(A, B) = f(propiedades(A), propiedades(B), L, E) \tag{2.15}$$

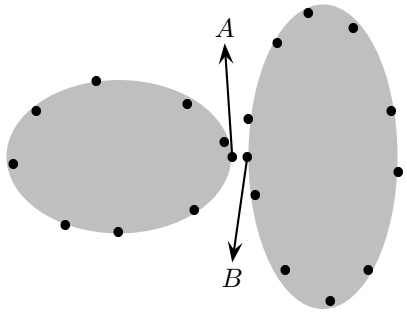


Figura 2.2: Diferencia entre cercanía y cohesión conceptual

Por tanto, dos objetos pueden ser similares, es decir, cercanos de acuerdo a una cierta medida de distancia (o similaridad), pero pueden tener una baja cohesividad conceptual, y viceversa. Un ejemplo de la primera situación se puede ver en la Figura 2.2. Los puntos negros A y B son "cercanos" entre sí y, por ello, serían ubicados en el mismo cluster por cualquier técnica basada únicamente en las distancias entre los puntos. Sin embargo, estos puntos tienen una cohesividad conceptual pequeña debido a que pertenecen a configuraciones que representan diferentes conceptos. Si se dispone de un lenguaje de descripción apropiado, un método de clustering conceptual permitirá agrupar los puntos de la Figura 2.2 en dos "elipses", como lo haría la mayoría de las personas.

Un criterio de calidad de clasificación utilizado en clustering conceptual puede involucrar una variedad de factores, tales como el *ajuste* de la descripción de un cluster a los datos, la *simplicidad* de una descripción u otras propiedades de las entidades o conceptos que los describen.

2.3.2. Descubrimiento de subgrupos

La extracción de modelos para la toma de decisiones a partir de conjuntos de datos en un proceso básico en la minería de datos [66]. Los modelos, dependiendo del dominio al cual se deseen aplicar, podrán ser o bien predictivos o bien descriptivos. En los modelos predictivos, el objetivo principal es la capacidad

de clasificación del modelo así como su interpretabilidad, mientras que en los descriptivos se busca encontrar relaciones o patrones de comportamiento.

Como hemos visto en la Sección 2.1, existen tres tipos de modelos; de caja blanca, de caja negra y de caja gris, en función de su interpretabilidad.

Una clase de modelos descriptivos son los generados para el descubrimiento de subgrupos, donde se pretende obtener modelos descriptivos empleando mecanismos predictivos. Los modelos descriptivos habitualmente están destinados al aprendizaje de reglas de asociación [2]. El aprendizaje de reglas de asociación es un mecanismo de inducción descriptiva que se dedica a descubrir reglas individuales que definen patrones interesantes en los datos.

Por otro lado, se puede definir el descubrimiento de subgrupos como un tipo de modelo descriptivo que se sitúa en la intersección entre la inducción predictiva y descriptiva. Fue formulado inicialmente por Klösgen, con su propuesta de aprendizaje de reglas EXPLORA, y Wrobel con MIDOS [57, 83]. En ellos, el problema del descubrimiento de subgrupos se define de la siguiente forma:

> "Dada una población de individuos y una propiedad de esos individuos en la que estamos interesados, buscar subgrupos en esa población que sean estadísticamente "más interesantes", siendo tan grandes como sea posible y ofreciendo el mayor valor de atipicidad estadística con respecto a la propiedad en la que estamos interesados".

En el descubrimiento de subgrupos, las reglas son de la forma $Cond \rightarrow Clase$, donde la propiedad de interés es el valor de la clase que aparece en el consecuente de la regla. El antecedente de la regla estará compuesto por una conjunción de características (pares atributo-valor) seleccionadas de entre las características que definen las instancias de entrenamiento. Dado que las reglas se han obtenido a partir de prototipos de entrenamiento etiquetados, el proceso de descubrimiento de grupos se centra en encontrar las propiedades de un conjunto determinado de individuos de la población que satisfacen la propiedad de interés dada. El descubrimiento de subgrupos se puede considerar como un mecanismo de inducción descriptiva que persigue encontrar patrones interesantes en los datos. Debido a esta circunstancia, algunas consideraciones estándar llevadas a cabo por los algoritmos de clasificación basados en reglas, tales como el que "las reglas inducidas deben presentar tanta precisión como sea posible" o que "las reglas deben ser tan diferentes como sea posible, para cubrir diferentes porciones de la población", deben de ser relajadas.

En el descubrimiento de subgrupos, el objetivo es encontrar reglas individuales o patrones de interés, los cuales deben ofrecerse en una representación simbólica adecuada de tal forma que puedan ser utilizados con efectividad por potenciales usuarios de esa información. La interpretabilidad de las reglas es por tanto un factor clave en el descubrimiento de subgrupos.

Ésta es la razón por la que a menudo se considera diferente el descubrimiento de subgrupos de las tareas propias de clasificación. El descubrimiento de subgrupos se centra en encontrar subgrupos de población interesantes en vez de maximizar la precisión del conjunto de reglas inducido.

Para evaluar el éxito en descubrimiento de subgrupos, se estudiarán medidas descriptivas sobre el interés de cada regla obtenida. Las medidas de calidad propuestas consistirán en el valor medio del conjunto de reglas obtenido, lo cual nos permite comparar diferentes algoritmos.

2.3.3. Estado del Arte

A continuación, se explicará el funcionamiento de dos métodos para el descubrimiento de subgrupos –APRIORI y SUBDUE–, los cuales serán utilizados en los próximos capítulos.

2.3.3.1. APRIORI

El algoritmo APRIORI [1] consiste en un proceso sencillo de dos etapas: generar y combinar. La primera etapa genera conjuntos de elementos (*itemsets*) más frecuentes de tamaño k y luego, durante la segunda etapa, se combinan para generar *itemsets* de tamaño $k+1$. Solamente luego de explorar todas las posibilidades de asociación conteniendo k elementos, se consideran aquellos conjuntos de $k+1$ elementos. El pseudocódigo del algoritmo APRIORI se muestra en el Algoritmo 2.2.

Algoritmo 2.2 APRIORI

APRIORI (D base de datos)

1: $L_1 \leftarrow \{$1-*itemsets* más frecuentes$\}$
2: $k \leftarrow 2$
3: **repetir**
4: $C_k \leftarrow$ k-*itemsets* generados a partir de L_{k-1}
5: **para todo** $t \in D$ **hacer**
6: Incrementar el contador de todos los candidatos de C_k que estén cubiertos por t
7: **fin para**
8: $L_k \leftarrow$ Todos los candidatos de C_k con soporte mínimo
9: $k \leftarrow k+1$
10: **hasta** $L_{k-1} = \emptyset$

2.3.3.2. SUBDUE

El algoritmo SUBDUE [54] se basa en la extracción de subestructuras en bases de datos estructuradas con atributos con valores discretos que pueden

ser representadas mediante grafos. Esta representación basada en grafos incluye etiquetas de nodos y ejes, tanto dirigidos como no dirigidos, donde los objetos y valores de atributos se hacen corresponder con vértices, y los atributos y las relaciones entre los objetos con ejes.

Algoritmo 2.3 SUBDUE

SUBDUE (grafo G, int $Beam$, int $Limit$)

1: Encolar $Q \leftarrow (v|v$ tiene una etiqueta única en $G)$
2: $bestSub \leftarrow$ primera subestructura en G
3: **repetir**
4: $newQ \leftarrow ()$
5: **para todo** S in Q **hacer**
6: $newSubs \leftarrow S$ extendida por un eje adyacente de G de todas las posibles maneras
7: $newQ \leftarrow newQ \cup newSubs$
8: $Limit \leftarrow Limit - 1$
9: **fin para**
10: Evaluar subestructuras en $newQ$ por compresión de G
11: $Q \leftarrow$ subestructuras en $newQ$ con valores máximos en $Beam$
12: **si** mejor subestructura en Q mejor que $bestSub$ **entonces**
13: $bestSub \leftarrow$ mejor subestructura en Q
14: **fin si**
15: **hasta** Q es vacío ó $Limit = 0$
16: **Devolver** $bestSub$

SUBDUE utiliza una variante del *beam search* [71] para su algoritmo de búsqueda principal. El objetivo de la búsqueda es encontrar la subestructura que mejor comprime el grafo de entrada. Para este fin, se usa el principio de *longitud de descripción mínima (MDL)* [66] para evaluar las subestructuras. Una vez que la búsqueda termina y devuelve la lista de mejores subestructuras, el grafo puede ser comprimido utilizando la mejor de todas. El procedimiento de compresión reemplaza todas las instancias de la subestructura en el grafo de entrada por un único nodo, el cual representa la subestructura. Los ejes entrantes y salientes de la subestructura reemplazada entrarán o saldrán del nuevo nodo que la representa.

De acuerdo a la heurística MDL, la mejor subestructura es la que minimiza la longitud de descripción del grafo una vez comprimido por la subestructura. La compresión se calcula como:

$$compresión = \frac{DL(S) + DL(G|S)}{DL(G)} \tag{2.16}$$

donde $DL(G)$ es el tamaño de la descripción del grafo de entrada, $DL(S)$ es el tamaño de la descripción de la subestructura, y $DL(G|S)$ es la tamaño de

la descripción del grafo de entrada comprimido por la subestructura. SUBDUE implementa el principio MDL en el contexto de la compresión de grafos, es decir, buscando aquella subestructura que minimice ese principio. En este contexto, la mejor subestructura en un grafo es aquella que minimiza $DL(S) + DL(G|S)$ [25]. El tamaño de la descripción de un grafo está basada en la representación de su matriz de adyacencia, contando el número de bits necesarios para almacenar los nodos y sus etiquetas, los ejes y sus etiquetas, y su matriz de adyacencia. La descripción detallada de este cálculo se puede consultar en [54].

El algoritmo de búsqueda intenta maximizar el *valor* de la subestructura, el cual es simplemente la inversa de la *compresión*.

Al comparar dos subestructuras diferentes del mismo repositorio, $DL(G)$ es la misma para ambas. De este modo, la medida resultante es una combinación lineal de $DL(S)$, esto es, el tamaño de la subestructura, y $DL(G|S)$, esto es, el tamaño de la base de datos comprimida. Este último depende del número de apariciones de la subestructura S en la base de datos y del tamaño de S en sí misma.

2.3.4. Clustering generalizado

Las técnicas de clasificación matemática han recibido mucha atención en el pasado como una de las grandes clases de procesos automatizables capaces de producir descripciones de características estructurales de un conjunto de ejemplos desde un conjunto de descripciones conocidas de sus componentes. El descubrimiento de estructuras, las cuales no son sencillas de identificar a simple vista tras un examen cuidadoso de los datos, es de gran importancia para la comprensión de las relaciones que controlan el comportamiento de sistemas, tales como las series económicas-financieras, las moléculas biológicas o las secuencias de ADN. Sin embargo, diferentes factores han limitado la utilidad de estos procedimientos de clasificación.

Por un lado, la aplicación de un concepto de clasificación a un problema específico lleva a obtener diferentes resultados de acuerdo al procedimiento de optimización usado, sin extraer la información subyacente de similaridad que los objetos de este problema poseen. Por otro lado, la mayoría de los procedimientos propuestos intentan clasificar puntos en conjuntos sobre la base de una medida de similaridad entre estos puntos. En la mayoría de las aplicaciones prácticas, no se logra obtener los resultados deseados de esta forma sin el uso de otras herramientas para relajar la medida de semejanza o los requisitos de clasificación [81].

A diferencia de las técnicas de clustering clásicas, el *clustering generalizado* no se basa en la interpretación de los clusters como subconjuntos de un conjunto de datos, es decir, en el hecho que cada par de puntos en el subconjunto esté cercano o sea "similar" en base a nociones predefinidas de distancia o similaridad. Al contrario, busca, en forma general, subconjuntos cohesivos en el sentido

de que los subconjuntos como un todo, exhiban ciertas relaciones y satisfagan ciertas restricciones especificadas en una colección de modelos dados. Para ello, identifica buenos clusters aislados en lugar de un buen clustering global. Enfatizaremos este proceso de clasificación como un proceso de identificación de atributos cualitativos en el sentido de que la similaridad no estará considerada solamente entre puntos, sino también entre estructuras de un conjunto de datos y un modelo parametrizado derivado de una generalización de un concepto prototípico. El proceso de clustering, simplemente considerado como la optimización de un funcional, puede llegar a obtener una colección de clusters precisos con una pobre generalización. Por lo tanto, cualquier metodología exitosa debe también considerar criterios adicionales basados en el tamaño de las subestructuras a ser explicadas.

2.4. Algoritmos evolutivos

La *Computación Evolutiva* (CE) se basa en el empleo de modelos de procesos evolutivos para el diseño e implementación de sistemas de resolución de problemas. Los distintos modelos computacionales que se han propuesto dentro de esta filosofía suelen recibir el nombre genérico de *Algoritmos Evolutivos* (AEs) [11]. Existen cuatro tipos de AEs bien definidos que han servido como base a la mayoría del trabajo desarrollado en el área: los *Algoritmos Genéticos* (AGs), las *Estrategias de Evolución* (EEs), la *Programación Evolutiva* (PE) y la *Programación Genética* (PG).

Un AE se basa en mantener una población de posibles soluciones del problema a resolver, llevar a cabo una serie de alteraciones sobre las mismas y efectuar una selección para determinar cuáles permanecen en generaciones futuras y cuáles son eliminadas. Aunque todos los modelos existentes siguen esta estructura general, existen algunas diferencias en cuanto al modo de ponerla en práctica. Los AGs se basan en operadores que tratan de modelar los operadores genéticos existentes en la naturaleza, como el cruce y la mutación, los cuales son aplicados a los individuos que codifican las posibles soluciones. En cambio, las EEs y la PE aplican transformaciones basadas en mutaciones efectuadas sobre los padres para obtener los hijos, lo que permite mantener una línea general de comportamiento del individuo en su descendencia. Finalmente, la PG codifica las soluciones al problema en forma de programas, habitualmente codificados en una estructura de árbol, y adapta dichas estructuras empleando operadores muy específicos.

Cada individuo de la población recibe un valor de una medida de adaptación que representa su grado de adecuación al entorno. La selección hace uso de estos valores y se centra en los individuos que presentan mayor valor en la media. Los operadores de recombinación y/o mutación alteran la composición de dichos individuos, guiando heurísticamente la búsqueda a través del espacio. Aunque simples desde un punto de vista biológico, este tipo de algoritmos son

suficientemente complejos para proporcionar mecanismos de búsqueda adaptativos muy robustos. Los mismos procedimientos pueden ser aplicados a problemas de distintos tipos, sin necesidad de hacer muchos cambios [42].

En las siguientes secciones, se explicarán con más detalle dos de estos tipos de AEs, los AGs y la PG, que serán los utilizados en este trabajo.

2.4.1. Algoritmos genéticos

Los AGs [42] son una técnica *metaheurística* para la solución de problemas de optimización.

> Una metaheurística es un proceso iterativo que guía a una heurística subordinada combinando inteligentemente diferentes conceptos para explorar y explotar el espacio de búsqueda, usando estrategias de aprendizaje para estructurar la información con el objetivo de encontrar eficientemente soluciones cercanas al óptimo [68].

Los AGs se basan en una analogía de la teoría biológica de la evolución de las especies y toman esta idea para buscar una o más soluciones óptimas entre un conjunto de posibles soluciones.

> La *Teoría de la Evolución* explica el origen y la transformación de los seres vivos como el producto de la acción de dos principios fundamentales: la selección natural y el azar. La selección natural regula la variabilidad de la recombinación y mutación aleatorias de los genes: toda la variedad que observamos en la naturaleza se basa en la capacidad de los seres vivos de producir copias de sí mismos, en que el proceso de reproducción actualiza muchas variantes, y en que, en la interacción con el ambiente, algunas de ellas son seleccionadas para sobrevivir y producir las copias subsiguientes [31].

En lugar de buscar de general a específico o de simple a complejo, los AGs generan descendientes de soluciones realizando repetidas mutaciones y cruces de las mejores soluciones de un conjunto. A cada paso, una colección de soluciones es actualizada reemplazando una fracción de la población por la descendencia de las soluciones más adaptadas al medio. Entonces se puede ver que estas soluciones serán las que tendrán mayor probabilidad de pasar a la próxima generación.

Para poder mostrar el algoritmo propuesto y explicar como funcionan, en general, los AGs es necesario primero comprender la terminología utilizada. En las Figuras 2.3, 2.4 y 2.5 se pueden ver los conceptos en forma gráfica tomando como ejemplo el problema de encontrar el número máximo entre 1 y 15 usando una representación binaria.

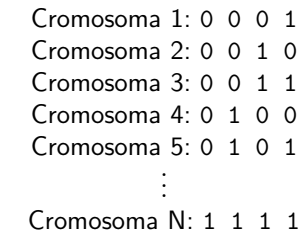


Figura 2.3: Población

Población. Se denomina población al conjunto de individuos que representan las soluciones a optimizar.
Cromosoma - Gen. Se denomina cromosoma a cada individuo de la población. A su vez, se conoce con el nombre de gen a cada parte del cromosoma que tiene significado por sí misma.

Genotipo - Fenotipo. En la naturaleza, un genotipo es la información genética que, al desarrollarse, crea un fenotipo o ser vivo. En el ámbito de los AGs, se denomina genotipo al conjunto de parámetros representado por un cromosoma particular que contiene toda la información necesaria para construir una solución (organismo), a la cual se la denomina fenotipo.

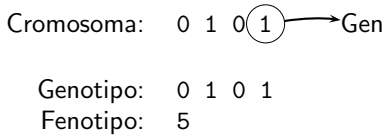


Figura 2.4: Genotipo vs. Fenotipo

Generación. Se denomina generación a cada iteración del algoritmo.

2.4.2. Algoritmo genético básico

Los AGs exploran un espacio de soluciones candidatas en busca de la mejor solución. Cuando decimos la mejor solución, nos referimos a aquella que optimice una cierta función relevante para el problema tratado, a la que se conoce con el nombre de función de *fitness* o función de aptitud. A pesar que existen clases muy diversas de AGs, todas mantienen una estructura en común:

> En cada iteración, todos los miembros de la población se evalúan de acuerdo a la función de fitness. Se genera una nueva población seleccionando de forma probabilística los mejores individuos de la

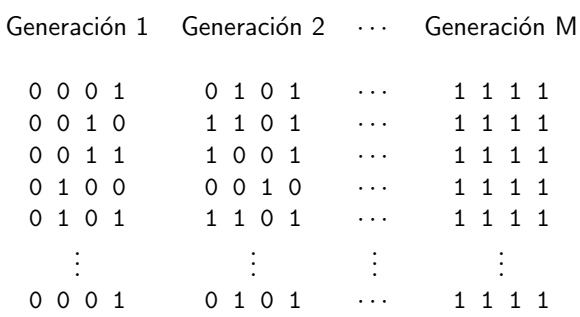


Figura 2.5: Generaciones

población actual. Algunos de estos individuos pasan a la próxima generación automáticamente, mientras que otros se utilizan para procrear nuevas soluciones por medio de cruces de dos individuos, o bien se mutan antes de pasar a la próxima generación.

El algoritmo itera hasta cumplir con un criterio de parada. Éste puede ser la cantidad de generaciones o evaluaciones de la función de fitness realizadas o la obtención de una solución que esté dentro de un cierto umbral de aceptación. El pseudocódigo del algoritmo básico para un algoritmo genético simple se muestra en la Figura 2.4 [42].

2.4.2.1. Representación de los cromosomas

La representación de los cromosomas depende del problema a tratar e influye directamente sobre los resultados del AG. Existen distintos esquemas generales de codificación entre los que destacan los siguientes:

1. La *codificación binaria*: Es la más antigua de todas las existentes. La representación de los cromosomas está definida como cadenas de bits de modo que, dependiendo del problema, cada gen puede estar formado por una subcadena de uno o varios bits.

2. La *codificación real*: La codificación binaria presenta algunos inconvenientes cuando se trabaja con problemas que incluyen variables definidas sobre dominios continuos: excesiva longitud de los cromosomas, falta de precisión, etc. Una posible manera de evitar estos inconvenientes es considerar un esquema de representación real. Aquí, cada variable del problema se asocia a un único gen que toma un valor real dentro del intervalo especificado, por lo que no existen diferencias entre el genotipo y el fenotipo.

Algoritmo 2.4 Pseudocódigo para un AG básico.

entrada: f función de fitness,
f_{umbral} criterio de terminación,
p tamaño de una población,
p_c probabilidad de cruce,
p_m probabilidad de mutación
salida: individuo con mayor f

1: $P \leftarrow$ Generar p individuos al azar
2: **para todo** $h \in P$ **hacer**
3: Calcular $f(h)$
4: **fin para**
5: **mientras** $([\text{máx}_h\, f(h)] < f_{umbral})$ **hacer**
6: $P_S \leftarrow \emptyset$
7: Seleccionar probabilísticamente $(1 - r)p$ miembros de P para agregar a P_S
8: Seleccionar con probabilidad p_c pares de individuos y producir descendencia aplicando el operador de cruce y agregarlos a P_S
9: Seleccionar con probabilidad p_m individuos de P_S. Para cada uno utilizar el operador de mutación y agregar el individuo modificado a P_S
10: $P \leftarrow P_S$
11: **para todo** $h \in P$ **hacer**
12: Calcular $f(h)$
13: **fin para**
14: **fin mientras**
15: **Retornar** individuo con mayor f

3. La *codificación basada en orden*: Este esquema está diseñado específicamente para problemas de optimización combinatoria en los que las soluciones son permutaciones de un conjunto de elementos determinando.

Estos ejemplos de posibles representaciones nos dan una idea genérica del tipo de esquemas que se utilizan más comúnmente. Esto no implica que sean los únicos, o que no se puedan crear esquemas propios, que no tengan relación alguna con los comentados, si es que se adaptan mejor a un problema en particular.

2.4.2.2. Mecanismo de Selección

El mecanismo de selección es el encargado de seleccionar la población intermedia de individuos la cual, una vez aplicados los operadores de cruce y mutación, formará la nueva población del AG en la siguiente generación. De este modo, el mecanismo de selección se encarga de obtener una población intermedia formada por copias de los cromosomas de la población original como

se muestra en la Figura 2.6.

Dos son los métodos de selección más comunes:

- *La Ruleta.* Consiste en crear una ruleta en la que cada cromosoma tiene asignada una fracción proporcional a su aptitud. Esta ruleta se gira varias veces para determinar qué individuos se seleccionarán. Debido a que a los individuos más aptos se les asignó un área mayor de la ruleta, se espera que sean seleccionados más veces que los menos aptos.
- *El torneo.* La selección por torneo se ha popularizado debido a que sólo utiliza información local para elegir a los mejores candidatos, por tanto reduciendo la complejidad de cálculo en poblaciones de gran tamaño. El torneo consiste en seleccionar un conjunto de individuos de la población al azar, dependiendo del tamaño del torneo, para luego comparar entre sí dichos individuos y elegir aquél con mejor valor de aptitud. Este proceso se realiza tantas veces como elementos existan en la población. En el caso de un torneo binario, la competencia se realiza entre dos individuos, seleccionando aquél con mejor función de aptitud.

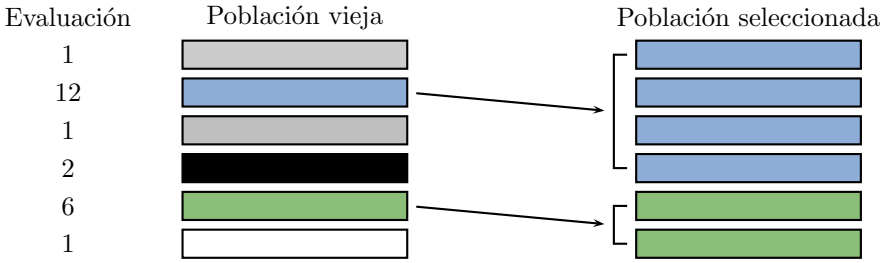


Figura 2.6: Ejemplo de aplicación del mecanismo de selección

2.4.2.3. Operadores genéticos

El operador de cruce constituye un mecanismo para compartir información entre cromosomas. Combina las características de dos cromosomas padre para obtener dos descendientes, con la posibilidad de que los cromosomas hijo, obtenidos mediante la recombinación de sus padres, estén mejor adaptados que éstos. No suele aplicase a todas las parejas de cromosomas de la población intermedia, sino que se lleva a cabo una selección aleatoria en función de una determinada probabilidad de aplicación, la *probabilidad de cruce,* p_c.

El operador de cruce cumple un papel fundamental en el AG. Su tarea consiste en **explotar** el espacio de búsqueda combinando las soluciones obtenidas hasta el momento mediante la recombinación de las buenas características que presenten.

Un ejemplo del cruce más conocido, llamado *cruce simple en un punto*, se muestra de forma gráfica en la Figura 2.7. El cruce simple se basa en seleccionar aleatoriamente un punto de cruce e intercambiar el código genético de los cromosomas padre a partir de dicho punto para formar los dos hijos.

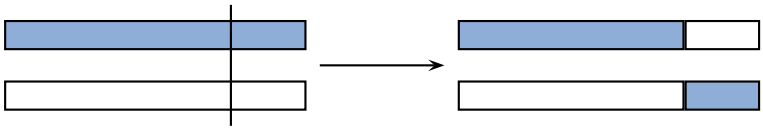


Figura 2.7: Ejemplo de aplicación del operador de cruce simple de un punto

La mutación, en cambio, pretende **explorar** el espacio de búsqueda alterando una de las componentes del código genético de un individuo. La mutación altera localmente el genotipo esperando obtener un individuo mejor.

Debido al efecto de la selección, se sabe que sólo serán elegidas las buenas soluciones para pasar a la próxima generación, mientras que las malas soluciones serán eliminadas.

2.4.3. Algoritmos genéticos para funciones multimodales: *Nichos*

Como el nombre sugiere, las funciones multimodales tienen múltiples soluciones óptimas, de las cuales varias pueden ser óptimos locales. Como ya hemos comentado, los AGs son conocidos por su capacidad para llevar a cabo procesos de búsqueda en espacios complejos. A pesar de ello, un AG clásico puede no trabajar de modo adecuado cuando el espacio de búsqueda es multimodal y presenta varios óptimos locales. En estos casos, los AGs simples se caracterizan por converger hacia la zona del espacio donde se encuentran los mejores óptimos locales, abandonando la búsqueda en las zonas restantes (proceso conocido con el nombre de "deriva genética") [42].

Se han propuesto varios métodos para tratamiento de funciones multimodales en AGs. Veremos a continuación algunos de ellos:

- **Diversidad a través de la mutación**. Encontrar soluciones cercanas a diferentes soluciones óptimas en una población y mantenerlas por varias generaciones son dos temas diferentes. En las etapas iniciales del AG, las soluciones están distribuidas por todo el espacio de búsqueda. Las soluciones cercanas a los múltiples óptimos se enfatizan en las primeras generaciones. Eventualmente, cuando la población contiene buenos clusters de soluciones cercanas a los óptimos, comienza la competencia entre los diferentes óptimos. Este proceso competitivo continúa hasta que la población converge a un solo óptimo. Para lograr mantener las soluciones encontradas en las primeras generaciones, es necesario un operador explícito de

preservación de diversidad. El operador de mutación se utiliza, usualmente, con esa función. Este operador tiene una función tanto constructiva como destructiva. Debido a esto, se suele utilizar una baja probabilidad de mutación. Esto produce que sea insuficiente como único operador de preservación de diversidad.

- **Preselección**. En [20], se utiliza un mecanismo conocido como *preselección*. El concepto principal de este operador es reemplazar con un individuo *parecido*. Cuando se genera un hijo a partir de dos soluciones padre, se compara automáticamente con ambos progenitores. Si el hijo está mejor adaptado que el peor de sus padres, entonces reemplaza al padre. Como un hijo suele ser similar a sus padres, el reemplazo permite que diferentes soluciones co-existan en la población, manteniendo de esta forma la diversidad.

- **Modelo de Crowding**. En [32] se desarrolló este método para introducir diversidad en la población de un AG. En este esquema, se emplea una población solapada donde los individuos reemplazan a aquellos individuos de la población original de acuerdo a su similaridad. Un individuo se compara con una subpoblación de miembros de la población solapada determinada al azar. El individuo con la mayor similaridad es reemplazado por este nuevo individuo.

Sin embargo, la forma más habitual de diseñar AGs multimodales se basa en los conceptos de *nicho* y *especie*. Ambos conceptos fueron introducidos con objeto de mantener múltiples óptimos. Una de las formas más habituales para provocar la formación de especies y la creación de nichos se basa en el esquema de *sharing* o proporción [43]. El proceso de sharing permite mantener en la población de cada generación una cantidad proporcional de individuos en distintas zonas del espacio de búsqueda, manteniendo de esta manera una buena diversidad de soluciones. Cada zona del espacio de búsqueda estará representado en el algoritmo como un *nicho*. En cada una se encontrará un conjunto de soluciones cercanas de acuerdo a una cierta distancia. Estas subpoblaciones crean nichos en dos espacios posibles: el genotípico y el fenotípico. Al primero se lo conoce también con el nombre de *espacio de variables*. Además del espacio de variables, existe también el *espacio de objetivos*. Este espacio está determinado por los objetivos del problema a optimizar. Para medir distancias en un espacio o en el otro, utilizaremos medidas diferentes, las que resulten más adecuadas al espacio y al problema.

Como ocurre en la naturaleza, los individuos de cada nicho comparten la recompensa asociada a dicho nicho entre ellos. Para esta tarea se define una función conocida como *fitness sharing* (función de proporción). Este método permite distribuir la población sobre diferentes picos (máximos o mínimos locales dependiendo del tipo de función de aptitud utilizada) del espacio de búsqueda,

donde la cantidad de individuos que recae en cada pico es proporcional a la calidad del pico si se trata de optimizar una función.

Entonces, la selección de individuos debe permitir que elementos pertenecientes a distintos nichos sean mantenidos en las futuras generaciones. Para poder hacerlo, se genera un torneo. Dos individuos compiten entre sí para determinar cual de los dos pasará a la próxima generación. Para decidir cual de los dos competidores será el ganador, se calcula la cantidad de elementos que existen en los nichos a los cuales pertenecen. La cantidad de elementos en cada nicho se define como:

$$nicho(x_i) = \sum_{x_j \in P} sh(d(x_i, x_j)) \tag{2.17}$$

donde d es la medida de distancia entre dos elementos, P es el conjunto de individuos de la población y sh es la función de sharing. Esta función se define por lo general como:

$$sh(v) = \begin{cases} 1 - v/\sigma_{share} & v \leq \sigma_{share} \\ 0 & v > \sigma_{share} \end{cases} \tag{2.18}$$

En este caso, σ_{share} es el radio del nicho que debe ser especificado por el usuario y determina la mínima separación deseada entre picos. La función de proporción (función de fitness modificada) sobre un individuo x_i se define entonces como $f(x_i)/nicho(x_i)$, donde f es el valor de su función de fitness. Este proceso permite evitar que toda la población converja a una única solución y, en lugar de ello, permite que los individuos de cada nicho converjan independientemente. Es deseable obtener nichos igualmente poblados en las distintas generaciones de forma tal que:

$$\frac{f(x_i)}{nicho(x_i)} = \frac{f(x_j)}{nicho(x_j)} \quad \forall x_i, x_j \quad individuos \tag{2.19}$$

2.4.4. Elitismo

El ciclo de nacimiento y muerte de los individuos está muy relacionado con el manejo de la población. El tiempo de vida de un individuo es típicamente de una generación, aunque en algunos AGs puede ser mayor. La estrategia de elitismo relaciona la vida de los individuos con su aptitud. Estas estrategias son técnicas utilizadas para mantener las buenas soluciones más de una generación. Una estrategia de elitismo habitualmente utilizada en AGs es mantener una copia del mejor individuo encontrado hasta el momento en cada generación. Esto se realiza porque en el cruce los padres suelen ser reemplazados por sus hijos y, por ello, no hay seguridad de que los individuos con mayor aptitud sobrevivan a la próxima generación.

2.4.5. Programación genética

La programación genética (PG) [59] es una técnica que permite a los ordenadores resolver problemas sin ser explícitamente programados. Trabaja utilizando AGs para generar automáticamente programas de ordenador.

Así, en la PG, los individuos de una población son programas de ordenador. Para facilitar el proceso de creación de nuevos programas a partir de dos programas padre, los programas se escriben habitualmente como árboles de expresión. Los nuevos programas se producen mediante la eliminación de ramas de un árbol y su inserción en las de otro. Este proceso simple, conocido como *cruce*, asegura que los nuevos programas sean también árboles y, por lo tanto, sintáctimente válidos (ver Figura 2.8). Por ello, la PG es fundamentalmente diferente de simplemente mezclar líneas de código de máquina.

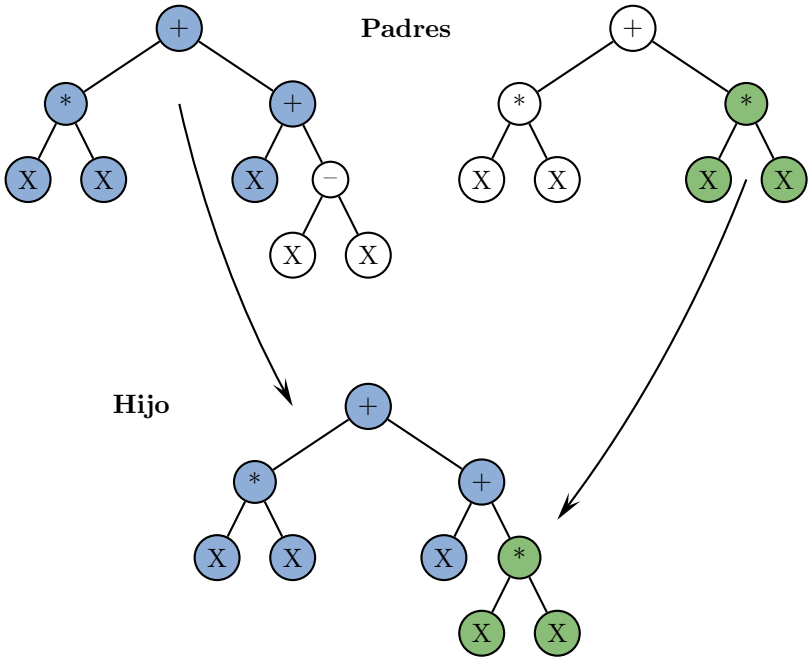


Figura 2.8: $x^2 + (x + (x - x))$ cruzado con $2x^2$ para producir $2x^2 + x$

La secuencia de operaciones en la PG viene dada en la Figura 2.9. Como puede observarse, es básicamente la misma empleada en otros AGs. El operador de mutación no se utiliza generalmente en PG, ya que en la mayoría de los casos no es necesaria [61].

La PG ha demostrado su potencial mediante la evolución de programas en un amplio rango de aplicaciones, incluyendo clasificación o recuperación de texto

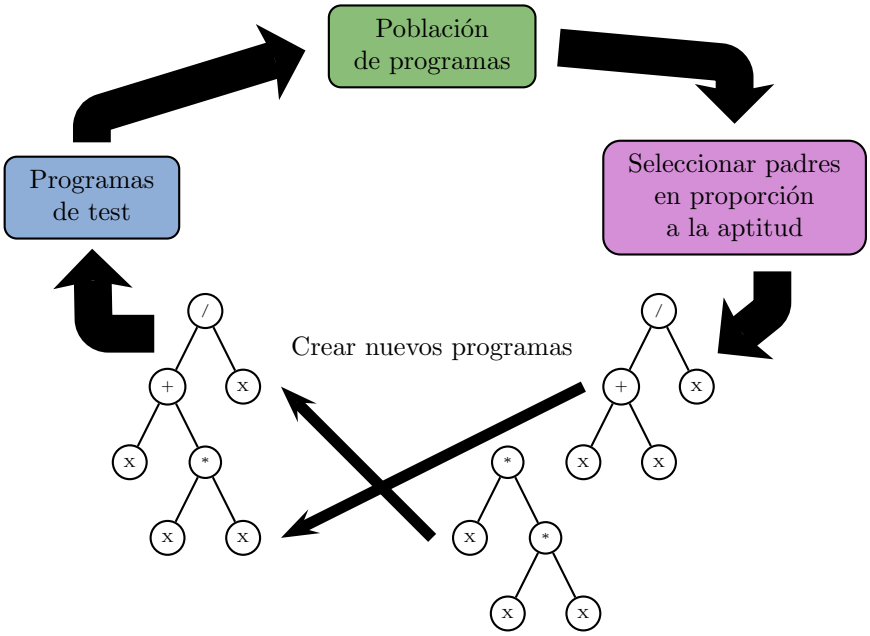


Figura 2.9: Ciclo de PG

[64, 37], reconocimiento de caracteres ópticos [7], clasificación de proteínas [46], procesamiento de imágenes [30], etc.

Las estructuras de datos son conceptos claves en la programación de ordenadores, puesto que proveen de un mecanismo para agrupar datos que lógicamente deberían estar juntos y permiten manipular estos datos como una unidad lógica. El tipado de datos (el cual se considera en muchos lenguajes de programación, incluyendo C++, Ada, Pascal y LISP, entre otros) es una forma de asociar un tipo (o clase) con cada instancia de una estructura de datos, permitiendo así al código manejar cada estructura de datos de manera diferente basándose en su tipo. Incluso estructuras de datos con la misma estructura física subyacente, pueden tener tipos lógicos diferentes. Por ejemplo, una matriz de 2×3, un vector de 6 elementos, y un vector de 3 números complejos, seguramente tendrán la misma representación física pero tendrán diferentes tipos de datos. Diferentes lenguajes de programación utilizan el tipado de datos de forma distinta. Los lenguajes fuertemente tipados, tales como Ada y Pascal, utilizan los tipos de datos en tiempo de generación de programa para asegurarse que las funciones solo recibirán como argumentos los tipos de datos particulares que están esperando. Los lenguajes tipados dinámicamente, tales como LISP, utilizan los tipos de datos en tiempo de ejecución para permitir a los programas manejar los datos

de forma diferente según sus tipos.

La PG clásica no está diseñada para manejar mezclas de tipos de datos. De hecho, una de sus asunciones es la de "clausura", la cual indica que un símbolo no-terminal debería poder manejar cualquier tipo de dato y valor devuelto por un terminal o no-terminal. Mientras que la clausura no prohibe el uso de múltiples tipos de datos, forzar un problema que utiliza múltiples tipos de datos a la restricción de clausura puede afectar negativamente al rendimiento de un algoritmo de PG aplicado al mismo de forma severa e innecesaria. Una manera de solucionar la restricción de clausura es definir los terminales y no-terminales del problema de manera cuidadosa para que no introduzcan tipos de datos múltiples. Sin embargo, esta técnica está limitada en su aplicabilidad. Por ejemplo, los problemas que manipulan matrices o vectores, o aquellos que trabajan con listas, no pueden solucionarse de esta manera.

Una forma de, simultáneamente, forzar clausura y permitir múltiples tipos de datos es a través del tipado dinámico. Con esta idea, cada no-terminal deberá poder manejar como argumentos cualquier tipo de datos que pueda ser devuelto por cualquier terminal y no-terminal. Existen dos maneras (no necesariamente excluyentes) de realizar esto. La primera es tener una función que realice diferentes acciones con diferentes tipos de datos. La segunda es tener funciones que señalen un error cuando los argumentos sean de tipo inconsistente y asignen una aptitud infinitamente mala al árbol. Esto es esencialmente lo que se realiza con "estructuras sintácticas restringidas" [59].

2.5. Problemas multiobjetivo y algoritmos genéticos multiobjetivo

Una diferencia notable entre los métodos de búsqueda y optimización clásicos y los AGs es que, en estos últimos, se procesa una población de soluciones en cada iteración. Esta característica, por sí sola, le da a los AGs una tremenda ventaja para su uso en problemas de optimización con múltiples objetivos.

Existen, al menos, dos dificultades asociadas a los métodos clásicos de resolución de estos problemas:

- Para obtener varias soluciones de la frontera del Pareto con un método clásico de optimización, es necesario ejecutarlo varias veces, comenzando cada vez de una solución inicial diferente.
- Sin embargo, las diferentes ejecuciones de los métodos clásicos sobre distintas soluciones iniciales no garantizan encontrar diferentes soluciones óptimas. Este escenario sólo ocurre si la solución inicial elegida es atraída siempre por el óptimo.

En las siguientes secciones se describirá la optimización multiobjetivo, conjuntamente con los enfoques clásicos para se resolución, se explicará el funciona-

miento de AGMO muy utilizando, el NSGA-II [34], y, finalmente, se introducirán las métricas clásicas para medir la bondad de las soluciones obtenidas mediante AGMOs.

2.5.1. Optimización multiobjetivo

Muchos problemas reales se caracterizan por la existencia de múltiples medidas de actuación, las cuales deberían ser optimizadas, o al menos ser satisfechas, simultáneamente. Como el propio nombre sugiere, el problema de la optimización multiobjetivo consiste en el proceso de optimización simultánea de más de una función objetivo [23].

La falta de metodologías para resolver este tipo de problemas llevó a que, en un principio, se resolvieran como problemas mono-objetivo. Sin embargo, no es correcto tomar esta determinación ya que existen diferencias entre los principios en que se basan los algoritmos que tratan un solo objetivo y los que trabajan con varios. De esta forma, al trabajar con problemas mono-objetivo nos enfrentamos con la búsqueda de una solución que optimice esa única función objetivo, tarea distinta a la que se nos plantea al trabajar con problemas multiobjetivo. En este último caso, no pretendemos encontrar una solución óptima que se corresponda a cada una de las funciones objetivo, sino varias soluciones que satisfagan todos los objetivos a la vez de la mejor manera posible. Como en un problema de optimización con un solo objetivo, también suele existir un número de restricciones que debe satisfacer cualquier solución factible.

La forma general de un problema de optimización multiobjetivo es la siguiente [33]:

Definición 1. *Un problema de optimización multiobjetivo se define como la maximización/minimización de* $\vec{f}$ *sabiendo que:*

$$\begin{array}{ll} f_m(x), & m = 1, 2, \ldots, M; \\ g_j(x) \geq 0, & j = 1, 2, \ldots, J; \\ h_k(x) = 0, & k = 1, 2, \ldots, K; \\ x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, N. \end{array}$$

donde M *corresponde al número de objetivos que tiene el problema,* J *al número de restricciones de desigualdad,* K *al número de restricciones de igualdad, y, finalmente,* N *al número de variables de decisión. Una solución* x *es un vector de* N *variables de decisión:* $x = (x_1, x_2, \ldots, x_n)^T$.
El último conjunto de la Ecuación 2.5.1 restringe cada variable de decisión x_i *a tomar un valor en el intervalo* $[x_i^{(L)}, x_i^{(U)}]$. *Si alguna solución* x *satisface todas las restricciones y los límites de las variables se la conoce como solución factible.*

La mayoría de los algoritmos de optimización multiobjetivo usan el concepto de dominancia es su búsqueda del óptimo. A continuación describimos con detalle este concepto [33]. En los algoritmos de optimización multiobjetivo, la

preferencia entre dos soluciones se especifica en función de que una domine a la otra.

Definición 2. *Se dice que una solución x domina a otra solución y ($x \prec y$)cuando se cumplen las siguientes condiciones:*

- *La solución x no es peor que y en todos los objetivos: $f_i(x) \not\triangleright f_i(y)$ para todo $i = 1, 2, \ldots, M$.*
- *La solución x es estrictamente mejor que y en, al menos, un objetivo: $f_j(x) \triangleleft f_j(y)$ para al menos un $i \in \{1, 2, \ldots, M\}$.*

Si alguna de las condiciones anteriores es violada, la solución x no domina a la solución y. Si x domina a la solución y también es común escribir que x es *no dominada* por y.

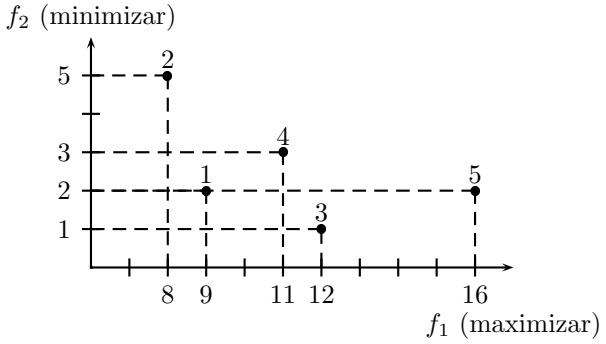


Figura 2.10: Dominancia entre soluciones.

Supongamos el problema con dos funciones objetivo representado en la Figura 2.10: la primera de las funciones f_1 será una función a maximizar, mientras que f_2 será una función a minimizar. Considerando la optimización conjunta de las dos funciones objetivo, es difícil encontrar una solución que sea la mejor respecto a ambas. Utilizando la definición de dominancia podremos decidir qué solución es la mejor de dos soluciones dadas en términos de ambos objetivos. Por ejemplo: si comparamos las soluciones 1 y 2, observamos que la solución 1 es mejor que la 2 en las dos funciones objetivo f_1 y f_2. Esto supone que se cumplen las dos condiciones de dominancia, luego podremos afirmar que la solución 1 domina a la solución 2.

De forma intuitiva podemos decir que si una solución x domina a otra y, entonces x es mejor que y para la optimización multiobjetivo.

El concepto de dominancia proporciona una forma de comparar soluciones con múltiples objetivos. Como ya hemos comentado, la mayoría de los métodos

de optimización multiobjetivo usan este concepto para buscar soluciones no dominadas.

Si, en el ejemplo que mostramos antes, comparamos la solución 3 con la 5, observamos que la solución 5 es mejor que la solución 3 en el objetivo f_1 mientras que es peor en el segundo objetivo, f_2. Vemos que la primera condición no se cumple para ambas soluciones, lo que simplemente nos dice que no podemos concluir que la solución 5 domine a la solución 3 ni tampoco que la 3 domine a la 5. Cuando esto ocurre, es costumbre decir que las soluciones 3 y 5 son no dominadas una respecto de la otra. Teniendo en cuenta ambos objetivos, no podemos decidir cuál de las dos soluciones es la mejor.

Para un conjunto de soluciones dado, podemos realizar todas las posibles comparaciones de pares de soluciones y encontrar cuáles dominan a cuáles y qué soluciones son no dominadas con respecto a las otras. Al final esperamos conseguir un conjunto de soluciones tales que cualesquiera dos no dominen una a la otra, es decir, un conjunto en el que todas las soluciones son no dominadas entre sí. Este conjunto se conoce como conjunto de soluciones no dominadas.

Este conjunto también tiene otra propiedad: dada cualquiera solución que no pertenezca a él, siempre podremos encontrar una solución del conjunto que la domine. Así, este conjunto particular tiene la propiedad de dominar a todas las soluciones que no pertenecen al mismo. En términos simples, esto significa que las soluciones de este conjunto son mejores comparadas con el resto de soluciones.

Una dificultad común en la optimización multiobjetivo es la aparición de un conflicto entre objetivos [47], es decir, el hecho de que ninguna de las soluciones factibles sea óptima simultáneamente para todos los objetivos. En este caso, la solución matemática más adecuada es quedarse con aquellas soluciones que ofrezcan el menor conflicto posible entre objetivos. Estas soluciones pueden verse como puntos en el espacio de búsqueda que están colocados de forma óptima a partir de los óptimos individuales de cada objetivo, aunque puede que dichas soluciones no satisfagan las preferencias del experto que quiera establecer algunas prioridades asociadas a los objetivos.

Para encontrar tales puntos, todas las técnicas clásicas reducen el vector objetivo a un escalar, es decir, a un único objetivo. En estos casos, en realidad, se trabaja con un problema sustituto buscando una solución sujeta a las restricciones especificadas.

A continuación, vamos a repasar tres de las técnicas clásicas más comunes para afrontar problemas con múltiples objetivos. Posteriormente, dedicaremos una sección a analizar los inconvenientes que presentan.

Optimización Mediante Ponderación de los Objetivos. Ésta es probablemente la más simple de todas las técnicas clásicas. En este caso, las funciones objetivo se combinan en una función objetivo global, F, de la siguiente manera [41]:

$$F(x) = \sum_{i=1}^{M} w_i f_i(x) \tag{2.20}$$

donde los pesos w_i se definen como:

$$F(x) = \sum_{i=1}^{M} w_i = 1 \quad ; \quad \forall w_i \in \mathbb{R},\ 0 \leq w_i \leq 1 \tag{2.21}$$

En este método, la solución óptima se controla mediante un vector de pesos w de forma que la preferencia de un objetivo puede cambiarse modificando dichos pesos. En muchos casos, primero se optimiza cada objetivo individualmente y después se calcula el valor de la función objetivo completa para cada uno de ellos. Así podemos conseguir evaluar la importancia que ejerce cada objetivo y encontrar un vector de pesos adecuado. Después, la solución final aceptada se calcula optimizando F según los pesos establecidos.

Las únicas ventajas de usar esta técnica es que se puede potenciar a un objetivo frente a otro y que la solución obtenida es normalmente Pareto-optimal.

Optimización Mediante Funciones de Distancia. Con esta técnica, la reducción a un escalar se lleva a cabo usando un vector de nivel de demanda, $\bar{y}$, que debe especificar el experto. Por esta razón, suele denominarse "programación por metas" (goal programming) [21]. En este caso, la función F se obtiene por medio de la siguiente fórmula:

$$F = \left(\sum_{i=1}^{M} f_i(x) - \bar{y}_i^r \right)^{\frac{1}{r}} \quad ; \quad 1 \leq r \leq \infty \tag{2.22}$$

Normalmente, se elige una métrica Euclídea $r = 2$, considerando $\bar{y}$ como óptimos de los objetivos individuales. Es importante recalcar que la solución obtenida depende enormemente del vector de nivel de demanda establecido, de modo que si éste es malo, no se llegará a una solución Pareto-optimal. Como la solución no está garantizada, el experto debe tener un conocimiento profundo de los óptimos individuales de cada objetivo para establecer adecuadamente $\bar{y}$. De esta forma, el método busca la meta indicada (representada por $\bar{y}$) para cada objetivo.

Esta técnica es similar a la anterior. La única diferencia es que ahora se requiere saber la meta de cada objetivo mientras que en el enfoque de ponderación era necesario conocer su importancia relativa.

Optimización Mediante Formulación Min-Max. Esta técnica intenta minimizar las desviaciones relativas de cada función objetivo a partir de óptimos individuales, esto es, intenta minimizar el conflicto entre objetivos [69]. El problema Min-Max se define como:

Minimizar $f(x) = max[Z_j(x)] \quad ; \quad j = 1, \ldots, M$

donde $Z_j(x)$ se define para el valor óptimo positivo $\bar{f}_j > 0$ como:

$$Z_j(x) = \frac{|f_j - \bar{f}_j|}{\bar{f}_j} \quad ; \quad j = 1, \ldots, M \tag{2.23}$$

Esta técnica puede obtener la mejor solución cuando los objetivos a optimizar tienen igual prioridad. Sin embargo, la prioridad de cada objetivo puede alterarse utilizando pesos en la fórmula. También es posible introducir un vector de nivel de demanda.

Inconvenientes de las Técnicas Clásicas. Todas las técnicas clásicas que se han utilizado para resolver problemas multiobjetivo tienen graves inconvenientes que han dado lugar a que no sean adecuadas en muchas ocasiones. A continuación, mencionamos los más significativos:

- Dado que los distintos objetivos se combinan para formar uno único, sólo podremos obtener una solución Pareto-optimal simultáneamente. En situaciones reales, los expertos necesitan con frecuencia varias alternativas para decidir, pero estas técnicas no pueden ofrecerlas.
- Además, para realizar esta combinación, muchas veces es necesario tener un conocimiento sobre el problema que generalmente es difícil de obtener.
- No funcionan bien cuando los objetivos no son fiables o tienen un espacio de variables discontinuas.
- Si las funciones objetivo no son determinísticas, la elección de un vector de pesos o de niveles de demanda entraña gran dificultad.
- Son muy sensibles y dependientes de los pesos o niveles de demanda usados.

Para solucionar estos problemas, han surgido técnicas avanzadas de optimización multiobjetivo basadas en Enfriamiento Simulado [15, 62], AGs [33, 22], EEs [60], etc. En concreto, los AEs multiobjetivo han demostrado un muy buen comportamiento.

2.5.2. Estado del arte: NSGA-II

El algoritmo NSGA-II [34] es una versión mejorada del *nondominated sorting genetic algorithm (NSGA)*. La idea subyacente al algoritmo NSGA [86] es la utilización de un método de selección basado en ranking para enfatizar las buenas soluciones y un método de nichos para una buena diversidad de soluciones en las sub-poblaciones.

El algoritmo NSGA trabaja como un AG clásico con algunos pasos adicionales para poder obtener un frente de Pareto diverso (ver Figura 2.11). Antes de

Figura 2.11: Diagrama de flujo del algoritmo NSGA

la etapa de selección, se ordena la población en base a la no dominancia de cada individuo. Se asigna entonces un valor de aptitud dummy suficientemente alto a todas las soluciones no dominadas para darle igual potencial reproductivo a todas ellas. Con el fin de mantener la diversidad de la población, el valor de aptitud de estos individuos se calcula proporcionalmente al número de individuos a los que domina. Este procedimiento se conoce como *sharing* (ver Sección 2.4.3). Luego del sharing, se ignora temporalmente a los individuos no dominados para así procesar al resto de la población de la misma manera. Los individuos resultantes conformarán el segundo frente de Pareto. A este nuevo conjunto de soluciones se les asigna un nuevo valor dummy de aptitud, el cual se mantiene estrictamente menor que el mínimo valor de aptitud del frente de Pareto previo. Este proceso continúa con el resto de la población hasta clasificar a todos los individuos en varios frentes de Pareto.

La población es entonces seleccionada de acuerdo a sus valores de fitness *dummy*, para luego aplicarles los operadores genéticos de la misma manera que

en un AG clásico. Dado que los individuos que se encuentran en el primer frente tienen el valor de fitness máximo, serán siempre los que obtengan un mayor número de copias que el resto de la población, resultando en una rápida convergencia de la población hacia las regiones no dominadas mientras que el *sharing* ayuda a distribuirlas uniformemente en esta región.

El algoritmo NSGA tiene algunos problemas que la extensión realizada por el NSGA-II intenta solucionar:

- Alta complejidad computacional en la determinación del orden de las soluciones no dominadas.
- Falta de elitismo.
- Necesidad de especificar los parámetros del sharing.

El algoritmo NSGA-II incluye las modificaciones necesarias para superar estos problemas. Primero, se reduce la complejidad computacional reescribiendo el código original de la ordenación de una manera más eficiente, guardando los datos temporales en cada paso para su posterior reutilización. Segundo, se agrega un procedimiento de elitismo que compara la población actual con la población anterior de las mejores soluciones no dominadas en cada generación del algoritmo. Finalmente, para evitar la necesidad de parámetros en el proceso clásico de sharing, se utiliza un nuevo procedimiento de sharing.

Para conseguir una estimación de la densidad de soluciones que rodean a una solución particular de la población, se calcula la distancia promedio de dos soluciones x_b y x_c a cada lado de esta solución en cada uno de los objetivos. Estos dos puntos se seleccionan de acuerdo al siguiente procedimiento. El cómputo de la distancia de crowding requiere reordenar la población de acuerdo a cada uno de los objetivos en orden descendente. Luego, para cada objetivo, se asigna a aquellas soluciones en los bordes (soluciones con un valor máximo o mínimo) un valor de distancia infinito. Entonces, se asigna a todas las soluciones intermedias un valor de distancia igual a la diferencia absoluta normalizada de los valores en el objetivo en cuestión de las dos soluciones adyacentes (las soluciones x_b y x_c).

El valor de distancia global de crowding se calcula finalmente como la suma de los valores de distancia individual correspondientes a cada objetivo. Cada función objetivo es normalizada antes de calcular el valor de distancia. Una solución con un valor de distancia menor está, en algún sentido, más rodeada de otras soluciones.

Luego de calcular el valor de distancia a cada solución de la población, podemos comparar dos soluciones según su grado de proximidad con otras soluciones. Gracias a ello, se define un operador de comparación de crowding $\prec_c$ para guiar este proceso de selección:

$$x_i \prec_c x_j \quad \text{si} \quad (\mathrm{x}_{\mathrm{i}_{\mathrm{orden}}} < \mathrm{x}_{\mathrm{j}_{\mathrm{orden}}}) \vee ((\mathrm{x}_{\mathrm{i}_{\mathrm{orden}}} = \mathrm{x}_{\mathrm{j}_{\mathrm{orden}}}) \wedge (\mathrm{x}_{\mathrm{i}_{\mathrm{distancia}}} > \mathrm{x}_{\mathrm{j}_{\mathrm{distancia}}})) \tag{2.24}$$

donde $x_{i_{orden}}(x_{j_{orden}})$ es el $orden-$ésimo frente donde la solución $x_i(x_j)$ está ubicado, e $x_{i_{distancia}}(x_{j_{distancia}})$ es la distancia de crowding de $x_i(x_j)$. Esto significa que preferimos, entre dos soluciones con diferentes rangos de no dominancia, aquella con el rango más bajo. Si ambas soluciones pertenecen al mismo frente, entonces preferiremos aquella solución que esté ubicada en la región menos densa. El esquema general de funcionamiento del algoritmo NSGA-II se grafica en la Figura 2.12.

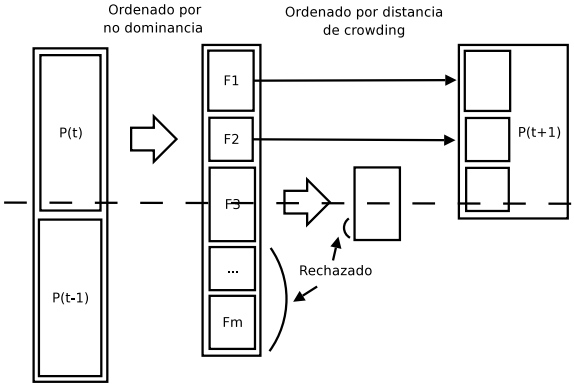


Figura 2.12: Esquema del algoritmo NSGA-II

2.5.3. Evaluación de los resultados de un algoritmo genético multiobjetivo

Como venimos mencionado, los AEs multiobjetivo basados en el Pareto se caracterizan por devolver muchas soluciones distintas a un problema determinado con diferentes objetivos a cumplir, igualmente válidas entre sí. Esto plantea la siguiente pregunta: ¿cómo podemos medir la calidad del algoritmo en lo que respecta a la generación de estas soluciones?

Dedicaremos las dos subsecciones siguientes a describir la forma en que podemos responder esta pregunta.

2.5.3.1. Métricas para la Medición de la Calidad de los Paretos

En el caso de la optimización multiobjetivo, la definición de calidad es sustancialmente más compleja que para la optimización de problemas mono-objetivo, ya que la optimización en sí implica varios objetivos. Un conjunto Pareto de soluciones no dominadas debería cumplir los siguientes requisitos:

1. Estar compuesto por un número alto de soluciones.

2. Estar compuesto por un número alto de soluciones distintas.

3. Minimizar la distancia existente con respecto al frente del Pareto óptimo para el problema en cuestión (es decir, el conjunto real de soluciones de dicho problema, el cual es a veces desconocido).

4. Presentar una buena distribución de las soluciones que lo componen (en el mejor de los casos, una distribución uniforme). La evaluación de este criterio puede basarse en el uso de una métrica.

5. Maximizar la extensión del frente no dominado (Pareto) obtenido, es decir, para cada objetivo, las soluciones no dominadas deben cubrir la mayor amplitud de valores posible.

En la literatura, podemos encontrar varios intentos para formalizar las definiciones anteriores (o partes de ellas) mediante métricas cuantitativas (véase [97]). Por ejemplo, los dos primeros criterios son sencillos de medir, basta con contar tanto el número de soluciones que componen el frente del Pareto derivado, como el número de soluciones distintas existentes entre ellas.

En el contexto de las investigaciones sobre convergencia al frente del Pareto óptimo (criterio 3), varios autores han considerado la distancia del conjunto de soluciones no dominadas generado al Pareto óptimo [38], de la misma manera que la función $\mathcal{M}_1$ que introduciremos posteriormente. La distribución no se tenía en cuenta, porque el interés no estaba en este aspecto. Sin embargo, en estudios comparativos, se ha visto que la distancia por sí sola no es suficiente para la evaluación de la calidad del Pareto y, consecuentemente, del comportamiento del algoritmo que lo derivó, ya que frentes con distribuciones muy distintas pueden tener la misma distancia al frente del Pareto óptimo.

De este modo, queda claro el hecho de que este criterio no es suficiente por sí solo y, además, presenta el problema de que es necesario conocer el frente real del Pareto para poder aplicarlo.

En [98], Zitzler y Thiele presentaron dos métricas complementarias para evaluar la calidad de los conjuntos de soluciones no dominadas generados por AEs multiobjetivo basadas en los criterios 4 y 5 mostrados anteriormente. Por un lado, se tiene en cuenta la distribución de las soluciones no dominadas en el espacio genotípico o en el espacio objetivo (métrica $\mathcal{M}_2$). Por otro, se considera el tamaño del área ocupada por dichas soluciones en el frente del Pareto (métrica $\mathcal{M}_3$).

Pasamos a describir la composición de las tres últimas métricas mencionadas. Sean $\bar{X}$ el frente del Pareto óptimo para el problema a resolver, X' el conjunto de soluciones no dominadas generado por el AE multiobjetivo, a y $\bar{a}$ dos soluciones pertenecientes respectivamente a los dos conjuntos anteriores, σ un parámetro de vecindad y $\|\cdot\|_H$ una medida de distancia:

- La función $\mathcal{M}_1$ proporciona la distancia media al conjunto Pareto óptimo $\bar{X}$. Como puede verse en la figura 2.13, las soluciones pueden estar en el

Pareto óptimo o fuera de él. En el caso de estar dentro del frente óptimo del Pareto, su distancia al mismo será cero. En el caso de no estar dentro del Pareto óptimo, se calcula la distancia a la solución del Pareto más cercana y se toma este valor como indicador de la calidad de esa solución:

$$\mathcal{M}_1(X') = \frac{1}{|X'|} \sum_{a' \in X'} min\{\|a' - \bar{a}\|_H; \bar{a} \in \bar{X}\} \tag{2.25}$$

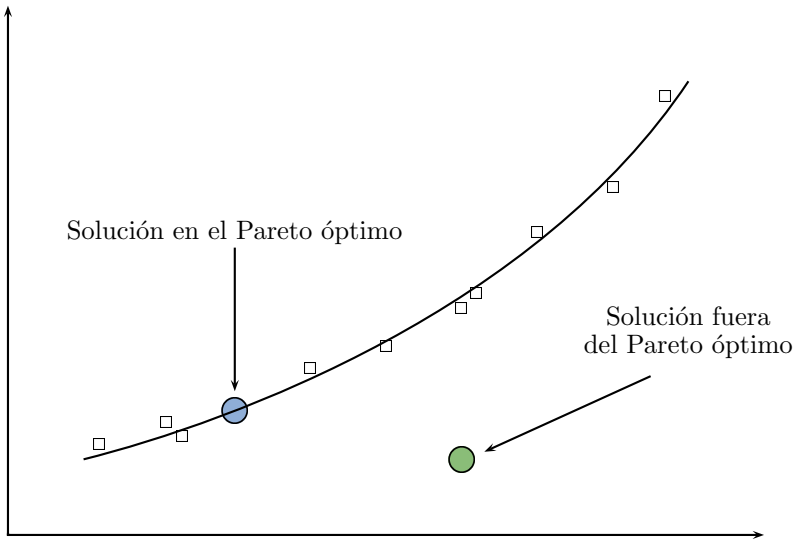


Figura 2.13: Soluciones posibles

Lógicamente, cuanto menor sea el valor de $\mathcal{M}_1$, menor será la distancia existente entre ambos conjuntos de soluciones no dominadas y mejor será la calidad del Pareto derivado.

- La función $\mathcal{M}_2$ tiene en cuenta la distribución de las soluciones del conjunto Pareto derivado X' con respecto al número de soluciones no dominadas que lo componen ($|X'|$):

$$\mathcal{M}_2(X') = \frac{1}{|X' - 1|} \sum_{a' \in X'} |\{a' \in X'; \|a' - b'\| > \sigma\}| \tag{2.26}$$

Nótese como, para cada solución a' del conjunto X', se contabiliza cuántas de las soluciones restantes están a una distancia mayor de σ de ella. Finalmente, se calcula el valor medio de la suma de la cuenta correspondiente a cada solución. De este modo, el valor de $\mathcal{M}_2$ está definido en $[0, |X' - 1|]$

y el Pareto generado será tanto mejor cuanto mayor sea dicho valor para un parámetro de vecindad adecuado. Por ejemplo, el valor $\mathcal{M}_2 = |X' - 1|$ indica que, para cada solución del Pareto, no existe ninguna otra solución a una distancia menor de σ de ella.

- Por último, la distribución de las soluciones a lo largo del Pareto óptimo puede estar aglomerada en una sola zona del espacio de búsqueda como puede verse en la Figura 2.14, lo que no es deseable. Por ello, se calcula la función $\mathcal{M}_3$ que considera la extensión del frente descrito por X':

$$\mathcal{M}_3(X') = \sqrt{\sum_{i=1}^{M} max\{\|a'_i - b'_i\|_H; a', b' \in X'\}} \tag{2.27}$$

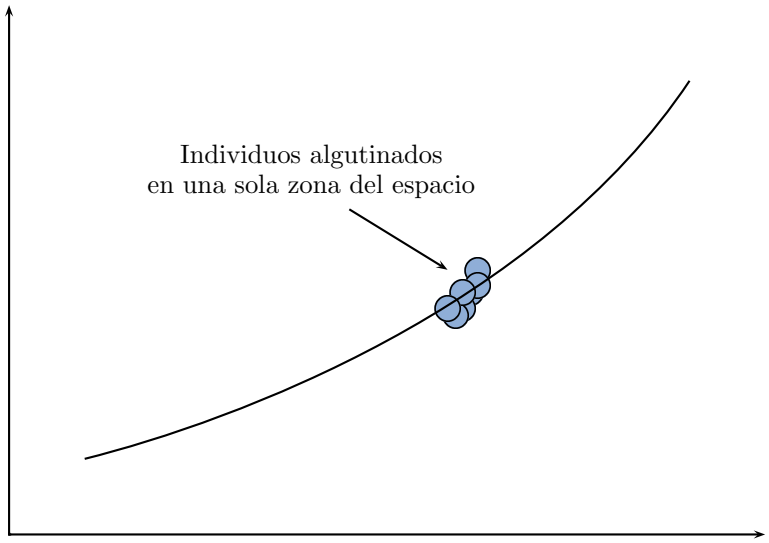


Figura 2.14: Soluciones aglutinadas en una zona del Pareto óptimo

Así, $\mathcal{M}_3$ mide la distancia máxima en cada dimensión para determinar el área que ocupa el Pareto.

Las métricas anteriores están definidas en el espacio genotípico, es decir, en el espacio de las soluciones al problema. Análogamente, existen tres métricas $\mathcal{M}_1^*$, $\mathcal{M}_2^*$ y $\mathcal{M}_3^*$ definidas sobre el espacio objetivo, es decir, sobre el espacio de vectores de valores de los objetivos. Sean Y e Y' los conjuntos de los vectores objetivo que corresponden a X y X', respectivamente; p y p' vectores objetivo de dichos conjuntos, y $\sigma^* > 0$ y $\|\cdot\|_{H^*}$ como antes, tenemos:

$$\mathcal{M}_1^*(Y') = \frac{1}{|Y'|} \sum_{p' \in Y'} min\{\|p' - \bar{p}\|_{H^*}; \bar{p} \in \bar{Y}\} \tag{2.28}$$

$$\mathcal{M}_2^*(Y') = \frac{1}{|Y' - 1|} \sum_{p' \in Y'} |\{p' \in Y'; \|p' - q'\| > \sigma^*\}| \tag{2.29}$$

$$\mathcal{M}_3^*(Y') = \sqrt{\sum_{i=1}^{N} max\{\|p'_i - q'_i\|_{H^*}; p', q' \in Y'\}} \tag{2.30}$$

Nótese que en el caso de problemas con dos objetivos, el valor de $\mathcal{M}_3$ equivale a la distancia existente entre las dos soluciones más extremas.

Adicionalmente, se define la métrica $\mathcal{S}$ [96], que calcula el tamaño del espacio dominado por las soluciones de un conjunto Pareto:

> Sea $A = (x_1, \ldots, x_l) \subseteq X'$ un conjunto de l soluciones. La función $S(A)$ devuelve el volumen cubierto por la unión de los politopos $p_1, \ldots, p_l$, donde cada p_i está formado por la intersección de los siguientes hiperplanos que surgen desde x_i siguiendo los ejes cartesianos: para cada eje en el espacio de objetivos, existe un hiperplano perpendicular al eje y que pasa a través del punto $(f_1(x_i), \ldots, f_m(x_i))$. En el caso de dos dimensiones, cada p_i representa un rectángulo definido por los puntos $(0, 0)$ y $(f_1(x_i), f_2(x_i))$.

2.5.3.2. Calidad de las soluciones obtenidas entre algoritmos

Hasta el momento, las medidas propuestas verifican la corrección de cada algoritmo en diversos atributos con respecto a la solución deseada. Si, en cambio, se desea hacer una comparación directa entre las diferentes versiones del mismo algoritmo o distintos algoritmos multiobjetivo, se debe realizar comparando ambas con el Pareto óptimo de por medio. Para realizar una comparación directa, se proponen dos índices que permiten ver si un algoritmo es mejor que otro sin tener en cuenta su parecido o inclusión en el Pareto óptimo.

Antes de enunciar las medidas propuestas es necesario explicar la notación a utilizar, nótese que los objetivos a optimizar se están maximizando:

Se dice que b domina globalmente a (también escrito como $a \prec b$) sii

$$\begin{array}{ll} \forall i \in \{1, \ldots, n\} : f_i(a) \leq f_i(b) & \wedge \\ \exists j \in \{1, \ldots, n\} : f_j(a) < f_j(b) & \end{array}$$

En el caso de tener solamente dos objetivos f_1 y f_2 la definición dada se reduce a buscar que

$$f_1(a) \leq f_1(b) \wedge f_2(a) \leq f_2(b) \quad \wedge$$
$$f_1(a) < f_1(b) \vee f_2(a) < f_2(b)$$

Adicionalmente, se dice que b cubre a ($a \preceq b$) sii $a \prec b$ o $f(a) = f(b)$. Se dice que b domina localmente a (también escrito como $a \lhd b$) sii

$$\forall i \in \{1, \ldots, n\} : f_i(a) \leq f_i(b) \quad \wedge$$
$$\exists j \in \{1, \ldots, n\} : f_j(a) < f_j(b) \quad \wedge$$
$$\|a - b\| \leq \sigma$$

Si utilizamos dos objetivos, entonces la definición dada se reduce a buscar que

$$f_1(a) \leq f_1(b) \wedge f_2(a) \leq f_2(b) \quad \wedge$$
$$f_1(a) < f_1(b) \vee f_2(a) < f_2(b) \quad \wedge$$
$$\|a - b\| \leq \sigma$$

Adicionalmente, se dice que b cubre localmente a ($a \trianglelefteq b$) sii $a \lhd b$ o $f(a) = f(b)$.

La primera medida propuesta utiliza una función que mapea un par de soluciones (X_1,X_2) a un intervalo [0,1] [97]:

$$\mathcal{C}(X_1, X_2) = \frac{|\{a_2 \in X_2; \exists a_1 \in X_1 : a_1 \trianglelefteq a_2|}{|X_2|} \tag{2.31}$$

El valor extremo $\mathcal{C}(X_1, X_2) = 1$ significa que todas las soluciones en X_2 están dominadas o son iguales que las soluciones de X_1. El valor extremo $\mathcal{C}(X_1, X_2) = 0$, por su parte, representa la situación en la cual ninguna de las soluciones de X_2 está cubierta por el conjunto X_1. Es necesario notar que tanto $\mathcal{C}(X_1, X_2)$ como $\mathcal{C}(X_2, X_1)$ tienen que ser considerados ya que $\mathcal{C}(X_1, X_2) \neq 1 - \mathcal{C}(X_2, X_1)$.

La segunda medida propuesta utiliza una función que mapea un par de soluciones (X_1,X_2) a un intervalo [0,1] [78]:

$$\mathcal{ND}(X', X'') = |\{a' \in X' \wedge a' \notin X'' : (\forall a'' \in X'' : a'' \not\preceq a')\}| \tag{2.32}$$

La medida $\mathcal{ND}(X', X'')$ compara dos conjuntos de soluciones no dominadas y devuelve el número de soluciones de X' que no son iguales y no son dominadas por ningún miembro de X''. Una vez más, ambas medidas, $\mathcal{ND}(X', X'')$ y $\mathcal{ND}(X'', X')$, deben ser tenidas en cuenta. Existe una diferencia clara entre las medidas $\mathcal{ND}$ y $\mathcal{C}$: la última muestra la relación de dominancia entre dos conjuntos de soluciones, mientras que la primera cuenta el número de soluciones novedosas, perteneciente al primer conjunto que no descubre el segundo.

Capítulo 3

Metodología

La creciente disponibilidad de repositorios de datos biológicos, tales como bases de datos que contienen información sobre interacciones proteína-proteína o caminos metabólicos, no ha sido secundada por el desarrollo de herramientas que permitan la extracción de conocimiento de estos repositorios. Esta problemática se ha renovado debido a las nuevas tendencias de acumulación de grandes cantidades de datos en bases de datos estructuradas, las cuales pueden verse como redes de información cuyos nodos son los objetos del repositorio y cuyos ejes corresponden a las relaciones existentes entre estos objetos. Varias técnicas de clustering conceptual se han aplicado para extraer subestructuras relevantes de estos datos estructurales. Sin embargo, los clusters más relevantes generalmente difieren de los patrones más frecuentes y, por ello, sigue siendo un reto.

En este capítulo se propondrá una metodología, llamada *Clustering Conceptual basado en Evolución MultiObjetivo (CC-EMO)*, que identifica conceptos de clusters representados como subestructuras en bases de datos estructuradas mediante la optimización conjunta de diferentes criterios.

3.1. Problemática

Durante las últimas décadas, se ha estado acumulando conocimiento, proveniente de diferentes áreas, en repositorios de datos digitales. Al estar almacenada de esta manera, la información puede ser estudiada y compartida más fácilmente por distintos expertos. A pesar de esto, el aumento constante del tamaño de estos repositorios hace casi imposible, para un ser humano, extraer información útil de los mismos. Por esta razón, se han desarrollado diversas técnicas de *minería de datos* para poder revelar información oculta en grandes colecciones de datos [45, 49] y así poder ayudar a los usuarios. Estas técnicas funcionan

correctamente con representaciones de datos en forma atributo-valor, es decir, datos no estructurados. Sin embargo, muchos proyectos de adquisición de datos actuales acumulan información estructurada que describe no sólo los objetos de la base de datos, sino también las relaciones que existen entre ellos. Estos conjuntos de datos son estructurados en el sentido de que los objetos que almacenan están descritos por las relaciones entre las características y no solamente por las características en sí mismas. Debido a ello, existe la necesidad de crear técnicas que permitan analizar y descubrir conceptos definidos mediante subestructuras en repositorios de datos estructurados.

Por otro lado, los algoritmos de clustering clásicos se centran en la optimización de un único criterio de preferencia. Comúnmente, este criterio suele ser la combinación de dos o más métricas que calculan la bondad de una configuración de clusters, como pueden ser: el número de clusters, la cantidad de elementos que cubre cada cluster, la cohesión de los miembros de cada cluster, cuán disjuntos son los distintos clusters, etc. Sin embargo, se utilizan mayoritariamente dos criterios: la *especificidad* y la *sensibilidad*. La especificidad se calcula generalmente como el tamaño de la subestructura que define el concepto que reune a las instancias agrupadas en un cluster, mientras que la sensibilidad corresponde habitualmente al número de instancias que pertenece a un cluster. En la mayoría de los casos, la especificidad y la sensibilidad de un cluster son objetivos que están contrapuestos. Por ello, la mayoría de las técnicas de clustering actuales se enfocan en encontrar una buena solución de compromiso, es decir, un conjunto de clusters que sean aceptablemente buenos en ambos objetivos simultáneamente mediante la agregación de los mismos en un único criterio de preferencia.

El enfoque clásico de clustering presenta varios problemas importantes. Primero, no es natural encontrar en dominios de datos reales un conjunto de conceptos que definan diferentes clusters que subdividan los objetos de la base de datos en grupos completamente disjuntos. De hecho, es considerablemente más probable tener objetos que pertenezcan a más de un cluster a la vez. Segundo, no todos los elementos de un repositorio de datos tienen el mismo nivel de interés, es decir, algunas instancias pueden contener campos con valores desconocidos o estar expuestos a errores en su definición que pueden representar outliers y, por lo tanto, no ser fácilmente agrupables con el resto de la base de datos. Al usuario final del proceso de clustering le serán útiles solamente aquellos conceptos que sean relevantes. Tercero, como hemos mencionado en la Sección 2.5.1, la combinación de diferentes funciones de preferencia en un único criterio generalmente lleva a obtener resultados subóptimos.

3.2. Propuesta

Para poder solucionar los problemas mencionados en la sección anterior, nos basaremos en el hecho de que las medidas para extraer subestructuras interesan-

tes de una base de datos estructurada necesitan la consideración de diferentes aspectos y, por ende, trabajan en el campo de la *optimización multiobjetivo*. Es por ello que proponemos una metodología a la cual llamaremos *Clustering Conceptual basado en Evolución MultiObjetivo (CC-EMO)* que realiza clustering conceptual de las instancias de un repositorio de datos estructurado de un dominio dado, identificando patrones comunes representados por subestructuras, mediante el uso de técnicas de optimización multiobjetivo.

Mediante el uso de un enfoque multiobjetivo, obtendremos el mejor conjunto de subestructuras que sean conjuntamente óptimas en especificidad y sensibilidad. Por lo tanto, dada una base de datos estructurada, el algoritmo CC-EMO propuesto en este trabajo generará, en una sola ejecución, un conjunto Pareto de subestructuras de un repositorio dado. Este conjunto Pareto estará compuesto de varias subestructuras que representarán diferentes conceptos, cada uno de ellos cubriendo un subconjunto de elementos de la base de datos. Para ello, cada cluster tiene un concepto inherente que agrupa un conjunto de instancias, las cuales están determinadas por su semántica. Las soluciones de este proceso de optimización serán: (1) aquellas subestructuras con un gran número de instancias cubiertas pero que pueden contener elementos que no sean altamente cohesivos, (2) aquellas subestructuras con una homogeneidad muy alta pero que representan una menor cantidad de instancias, y (3) todas las posibles subestructuras no dominadas con valores intermedios de ambos objetivos entre (1) y (2).

Con este propósito, haremos uso de los *Algoritmos Evolutivos (AEs)* [11], que han demostrado obtener buenos resultados en este campo [33, 22]. Algunos ejemplos de aplicaciones al aprendizaje automático y a la minería de datos utilizando AEs se pueden encontrar en [44, 39, 26].

3.2.1. Metodología general

La técnica multiobjetivo descrita en los párrafos anteriores constituye el corazón de la metodología que proponemos en este trabajo. Pero, para poder obtener buenos resultados, es necesario hacer uso de técnicas de pre y post procesamiento que faciliten la búsqueda de las mejores soluciones.

El flujo de información en el que se basa nuestra metodología se muestra en la Figura 3.1. La metodología consta de cuatro etapas: *Construcción de la base de datos estructurada*, *Clustering conceptual multiobjetivo*, *Compactación la base de datos* y *Predicción*. Durante la primera fase se realiza el pre-procesamiento de la base de datos donde está contenida la información a ser estudiada. Mediante una transformación de estos datos en bruto en objetos estructurados, se genera un nuevo repositorio de datos. La transformación de cada elemento de la base de datos incluye no sólo los atributos de cada instancia, sino también las relaciones que éstos tienen entre sí. Esta nueva base de objetos estructurados servirá de entrada al proceso central de la metodología, el *clustering conceptual*

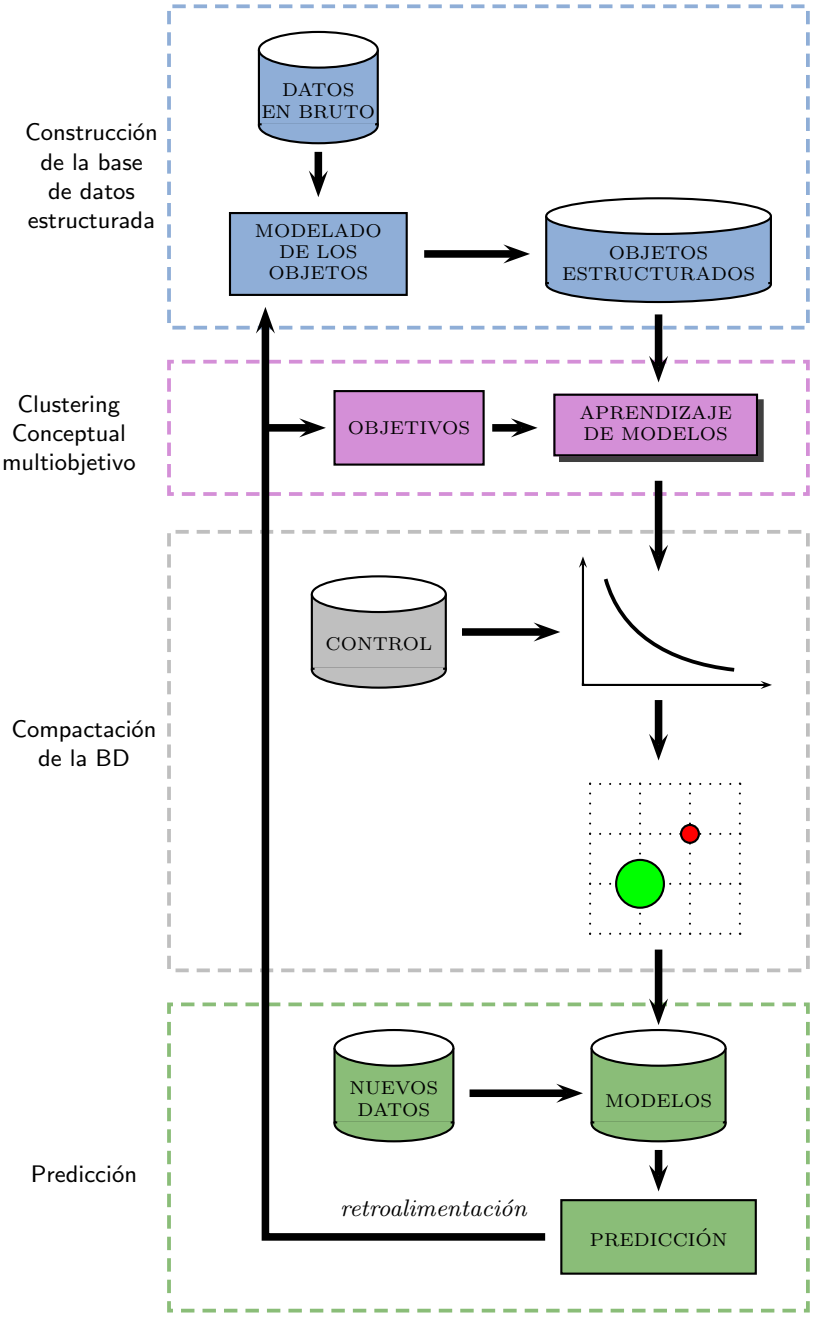


Figura 3.1: Esquema general de la metodología CC-EMO

multiobjetivo. El agrupamiento de las instancias de la base de datos se realiza mediante la optimización de diversos objetivos, los cuales son definiciones específicas de los dos criterios genéricos de especificidad y sensibilidad para cada problema en particular. Los conceptos aprendidos son aquellas soluciones que no están dominadas entre sí. Esto quiere decir que los conceptos descubiertos no son peores que ningún otro del conjunto de soluciones en todos los objetivos. En otras palabras, las soluciones encontradas no son comparables entre sí, ya que si un concepto es mejor que otro en algún objetivo, será, a su vez, peor en otro. Se genera entonces un conjunto Pareto de soluciones no dominadas, el cuál se depura luego en la etapa de compactación utilizando para ello una nueva base de datos de control. Esta nueva base de datos contiene información externa e independiente de los mismos datos en bruto que han sido utilizados al comenzar el proceso. A la luz de estos datos, se seleccionan aquellos conceptos más relevantes y se genera una nueva base de datos con estos conceptos para poder utilizarlos en la etapa de predicción. En esta última fase se realiza una clasificación de nuevos datos utilizando los conceptos generados por la metodología. Al estudiar la clasificación realizada en el proceso de inferencia, es natural encontrar pequeños fallos que pueden ser modificados en un proceso de retroalimentación y corregir no sólo el sistema de modelado de los datos, sino también los objetivos del algoritmo de clustering, para así poder, en una nueva iteración, producir mejores predicciones.

En las secciones siguientes se explicará con más detalle cada uno de estos pasos utilizando un dominio sencillo para ilustrarlos adecuadamente.

3.3. Construcción de la base de datos estructurada

Para poder comprender mejor el funcionamiento de la metodología propuesta, utilizaremos un dominio de datos muy sencillo que nos permitirá ejemplificar cada paso de la misma. Hemos denominado *geométrico* al dominio empleado y es una simplificación del considerado en [54]. En este dominio se representan figuras geométricas y sus posiciones relativas en el espacio. En nuestra simplificación trabajaremos solamente con pilas de figuras geométricas.

Nuestra base de datos de objetos en bruto consiste en instancias donde se representan pilas de figuras geométricas en forma lineal. Por ejemplo, tendremos una instancia *"cuadrado - rectángulo"* que representa una pila que contiene a un cuadrado sobre un rectángulo. Para poder trabajar de forma eficiente con los datos de este dominio, es necesario poder modelarlos de manera que se facilite su procesamiento. Cada instancia de la base de datos original se representa como un árbol en la base de objetos estructurados. El árbol está compuesto por dos clases de nodos y dos clases de ejes. Los nodos pueden ser de dos tipos diferentes: nodos *objeto* o nodos *figura*. Los nodos figura pueden ser instanciados

en: “triángulo”, “cuadrado”, “elipse”, “rectángulo” o “círculo”, mientras que los nodos objeto sólo pueden tener el valor “objeto”. Existen dos posibles relaciones en este dominio, representadas como ejes en los árboles: la relación “sobre” y la relación “forma”. La relación *forma* solamente puede ocurrir entre un nodo de tipo objeto y un nodo de tipo figura, mientras que la relación *sobre* sólo puede unir dos nodos de tipo objeto. Desde el punto de vista semántico, la relación *forma* determina la propiedad visual del nodo objeto que está conectado al nodo figura, mientras que la relación *sobre* determina las posiciones relativas de los objetos. Debido al hecho que sólo existen dos posibles relaciones, las instancias y las subestructuras se representan como árboles binarios.

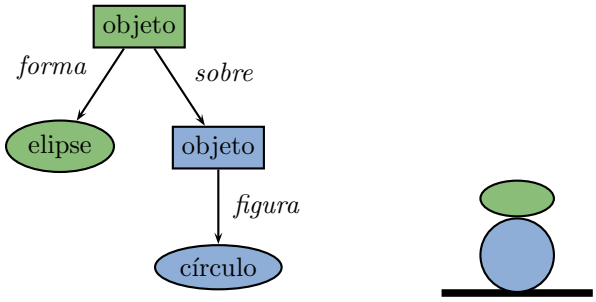


Figura 3.2: Ejemplo de instancia de la base de datos de objetos estructurados del dominio *geométrico*

En la Figura 3.2 se puede ver un ejemplo de una instancia de la base de datos donde las diferentes formas de los nodos representan diferentes tipos de nodos y las diferentes etiquetas de los ejes muestran las diferentes relaciones. En esta figura se puede ver, a la izquierda, la subestructura que representa el concepto del cluster (genotipo) y, a la derecha, su semántica (fenotipo). En este caso hay dos objetos, uno con forma circular y otro oval. La relación *sobre* los conecta determinando la posición relativa en la pila de objetos. En este caso, el objeto con forma oval está sobre el objeto de forma circular.

Mediante esta representación se pueden llegar a generar posibles conceptos de clusters. En la Figura 3.3 se muestra una subestructura posible para el domino geométrico. Las cajas con signos de interrogación corresponden a objetos cuya relación forma no está descrita y sólo las relaciones de posición están determinadas mediante ejes de tipo *sobre*. Esta subestructura representa el concepto que cubre a todas aquellas instancias con al menos tres figuras geométricas en la pila en las que haya un objeto de forma triangular con otro objeto sobre él y uno por debajo de él, cualquiera sea su forma. Este ejemplo simple muestra el poder de expresión de las subestructuras que pueden ser descubiertas por nuestra metodología utilizando una representación de datos adecuada y a la vez sencilla de interpretar por cualquier usuario. Un modelado incorrecto o demasiado específico puede llegar a sesgar el espacio de búsqueda y, por lo tanto, proveer

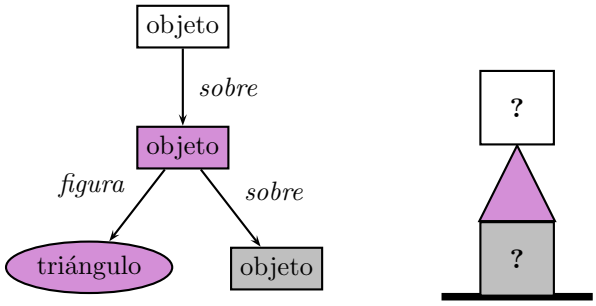


Figura 3.3: Ejemplo de subestructuras del dominio *geométrico*

de información de menor utilidad para el usuario final. Mediante este modelado de objetos, se construye una nueva base de datos constituida por estos objetos estructurados (en este caso, árboles) que permitirá al algoritmo principal poder inferir mejores conceptos de clusters.

3.4. Clustering Conceptual multiobjetivo

El proceso de clustering por el cual se aprenden los conceptos de cada cluster constituye el corazón de la metodología. El clustering se realiza mediante una técnica evolutiva de optimización multiobjetivo que optimiza conjuntamente los objetivos específicos del problema.

En el caso del dominio geométrico, se utilizarán dos objetivos clásicos de clustering: *especificidad* y *sensibilidad*. La sensibilidad de un cluster se calcula, simplemente, como la cantidad de instancias que se encuentran agrupadas en el cluster dado. La especificidad hace referencia al tamaño de la subestructura que define el concepto del cluster. Como en el dominio geométrico la representación de los conceptos es en forma de árbol, el tamaño de la subestructura se calcula como el número de nodos sumado al número de ejes que posea. Por ejemplo, la subestructura de la Figura 3.3 tiene tamaño 7, ya que su genotipo consta de 4 nodos y 3 ejes. Las funciones objetivo, resumidas en las ecuaciones 3.1 y 3.2 y utilizadas en este ejemplo, son estándar y pueden considerarse en esta metodología para cualquier dominio en el cual no se tenga mucho conocimiento. A partir de su aplicación a un conjunto de datos y a través de la retroalimentación que éste provea, se podrán ir depurando hasta conseguir un conjunto de objetivos específicos para el problema. A su vez, se puede ver que estos dos objetivos son claramente contrapuestos ya que, generalmente, a medida que el tamaño de la subestructura, aumenta la descripción del cluster se hace más específica y éste cubre una menor cantidad de elementos. Por lo tanto, es necesario obtener todas aquellas posibles soluciones de compromiso entre los dos objetivos utilizando el concepto de dominancia de Pareto introducido en la Sección 2.5.1:

$$Soporte(x_{subestructura}, BD) = \sum_{i \in BD} \begin{cases} 1 & Cubre(x_{subestructura}, i) \\ 0 & sino \end{cases} \quad (3.1)$$

$$Tamaño(x_{subestructura}, BD) = Tamaño(x_{subestructura}) \quad (3.2)$$

Debido a que, en la mayor parte de los dominios reales, el espacio de búsqueda es sumamente grande, es necesario hacer uso de alguna heurística para acelerar la búsqueda. A continuación, se describirá el método llamado *Clustering Conceptual basado en Evolución MultiObjetivo (CC-EMO)* que realizará la búsqueda en un tiempo razonable obteniendo los resultados deseados. Si bien los conceptos que se extraerán pueden no ser óptimos, el algoritmo heurístico propuesto ha demostrado obtener buenas soluciones en los dominios experimentados.

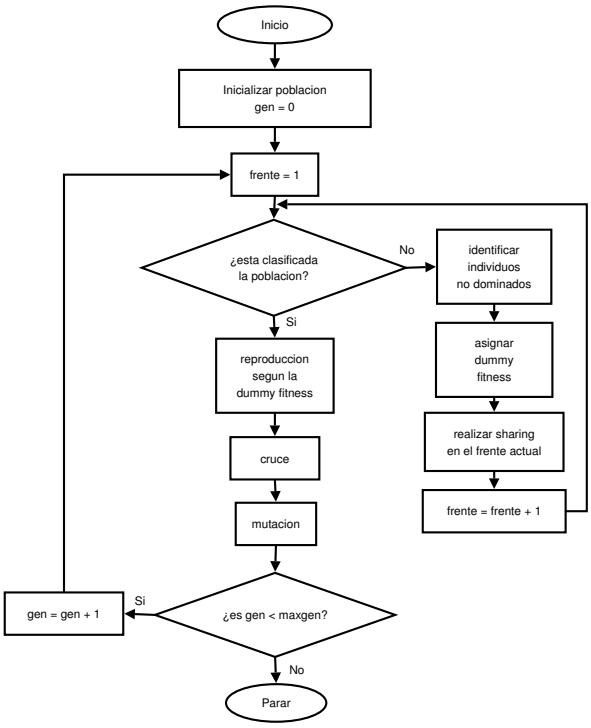
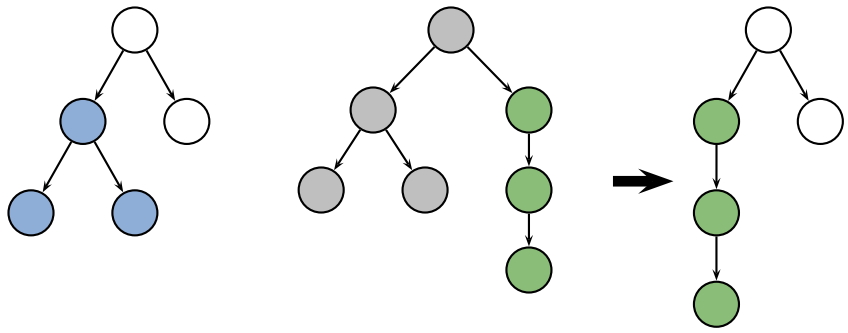


Figura 3.4: Algoritmo CC-EMO

La implementación actual del algoritmo CC-EMO está basada en el algoritmo NSGA-II [34] descrito en la Sección 2.5.2. Los pasos básicos de CC-EMO se

muestran en el diagrama de flujo de la Figura 3.4 y sus componentes principales se describen a continuación:

Representación de los cromosomas. El algoritmo CC-EMO utiliza una representación interna de cromosomas en forma de árbol, lo cual lo convierte en un algoritmo de Programación Genética (PG) [59, 12] (ver Sección 2.5.1). Los problemas a los cuales se les puede aplicar este algoritmo son aquellos que puedan ser representados en forma de grafos que no contengan ciclos. Como vimos en la Sección 2.5.1, la PG es un AE basado en evolucionar programas codificados como estructuras tales como árboles de expresiones. La PG ha sido utilizada ampliamente para resolver varios problemas reales en diferentes dominios, tales como: modelado de sistemas no lineales [75], recuperación de información [27, 28] y minería de datos [90], obteniendo buenos resultados. La codificación de un cromosoma está compuesta de nodos, que representan las características, y ejes que los conectan, los cuales representan relaciones entre las características de una instancia de la base de datos. Cada nodo y eje que conforman un cromosoma tiene un nombre, el cual describe al nodo (su semántica), y una etiqueta asociada que indica el tipo de nodo. Esta etiqueta será utilizada por los operadores genéticos, como veremos más adelante.



(a) Padre 1 (b) Padre 2 (c) Hijo

Figura 3.5: Operación de cruce de un algoritmo de PG

Operadores genéticos. Sobre los cromosomas que componen la población de un PG, se aplican los operadores de *cruce* y *mutación*. El operador de cruce utilizado es el estándar para PG, que consiste en el intercambio de dos subárboles elegidos al azar de cada uno de los árboles padres, como se muestra en la Figura 3.5. Con el fin de producir un descendiente válido, se impone una restricción al operador de cruce: la raíz de ambos subárboles debe tener la misma etiqueta. Reemplazando subárboles con la misma etiqueta en su nodo raíz, se mantendrán las relaciones válidas en el cromosoma descendiente final y, por lo tanto,

se obtendrá un descendiente válido. En el área de PG, el uso de este tipo de cromosoma con restricciones se conoce como PG *basada en tipos* [59].

Los operadores de mutación utilizados son también clásicos en PG:

- *Borrado de una hoja,* seleccionada aleatoriamente y eliminada conjuntamente con el eje que la conecta al árbol (ver Figura 3.6).
- *Modificación de un nodo,* seleccionado aleatoriamente y reemplazado por otro perteneciente al conjunto posible de nodos con la restricción de tener la misma etiqueta (ver Figura 3.7).
- *Agregación de una hoja,* creada aleatoriamente y conectada al árbol por un nuevo eje (ver Figura 3.8).

Optimización multiobjetivo. Los objetivos que procura maximizar conjuntamente CC-EMO son, como ya hemos mencionado, *especificidad* y *sensibilidad*. Para el caso específico del dominio geométrico, ambos objetivos se calculan como se muestra en las ecuaciones 3.1 y 3.2 utilizando las funciones *Cubre* y *Tamaño* definidas en las ecuaciones 3.3 y 3.4.

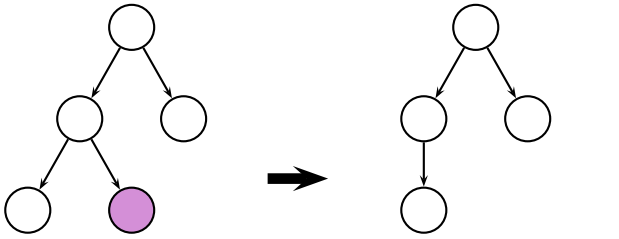
$$Cubre(arbol_A, arbol_B) = arbol_A \subseteq arbol_B \tag{3.3}$$

$$Tamaño(arbol_X) = \#nodos(arbol_X) + \#ejes(arbol_X) \tag{3.4}$$

La función *Cubre* nos dice si un árbol dado (la subestructura que define un cluster) cubre a otro árbol (la representación de una instancia de la base de datos), comprobando si el primer árbol es un subárbol del segundo. En la Figura 3.9 se muestra una subestructura y dos instancias cubiertas por la misma. En color celeste se ilustra en el genotipo la relación de inclusión ($\subseteq$) entre la subestructura y la instancia.

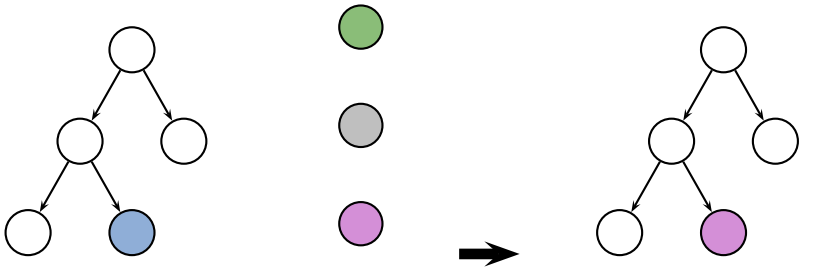
Relación de no dominancia. Es necesario hacer notar que dos conceptos con valores muy similares en sus objetivos, por ejemplo, el mismo tamaño de la subestructura y un soporte muy similar, pueden no describir el mismo grupo de instancias de la base de datos, como puede verse en la Figura 3.10. La dominancia clásica excluiría a uno de ellos, aquél con el menor número de instancias cubiertas, lo cual nos haría perder clusters potencialmente interesantes. Debido a que estos conceptos no describen las mismas instancias, debemos mantener ambas soluciones y, por tanto, serán necesarias ciertas restricciones para poder producir un conjunto Pareto sin perder subestructuras interesantes.

Ésta es una problemática clásica de los problemas de optimización multimodales, como ya se ha descrito en la Sección 2.4.3. Para solucionarla, existen diversas técnicas que evitan que, dadas dos soluciones óptimas locales pertenecientes a distintas zonas del espacio, una de ellas sea eliminada. En nuestra metodología, utilizaremos una técnica de *nichos* calculada en el espacio de instancias como se explicará a continuación.



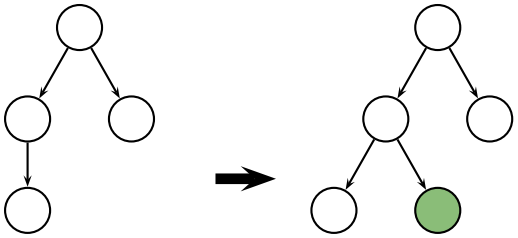
(a) Árbol original (b) Árbol mutado

Figura 3.6: Mutación: borrado de una hoja



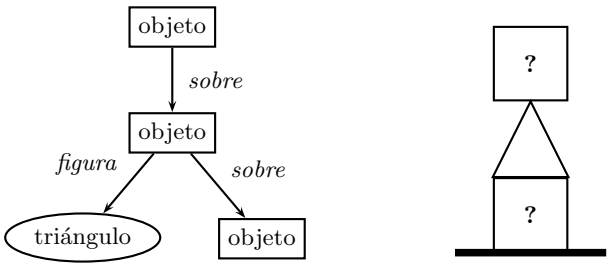
(a) Árbol original (b) Nodos posibles (c) Árbol mutado

Figura 3.7: Mutación: modificación de un nodo

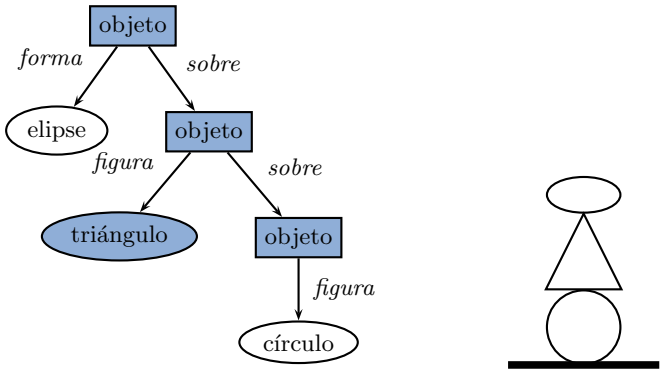


(a) Árbol original (b) Árbol mutado

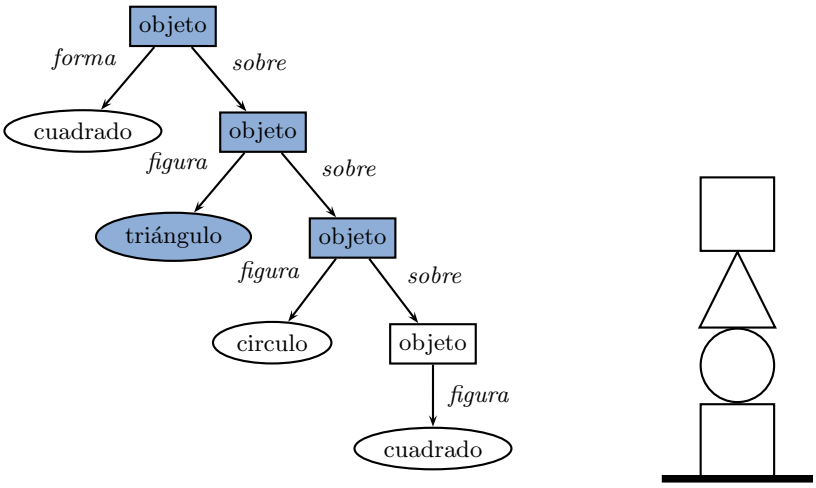
Figura 3.8: Mutación: agregación una hoja



(a) Modelo



(b) Instancia 1



(c) Instancia 2

Figura 3.9: Ejemplos de cobertura de una subestructura (a) a dos instancias (b,c)

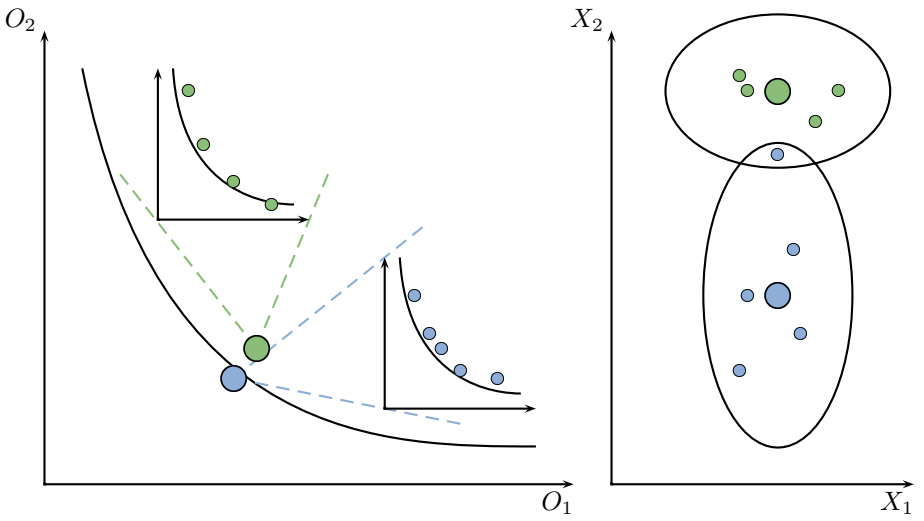


Figura 3.10: Dos soluciones en el conjunto Pareto. A pesar de que a simple vista parece que una domina a otra, las dos soluciones pertenecen a dos espacios diferentes teniendo en cuenta las variables y no los objetivos.

La no dominancia entre soluciones se modificará entonces para hacerla tener en cuenta las instancias cubiertas por cada concepto. Para ello, dos soluciones son comparadas por dominancia si y solo si tienen al menos un 50 % de instancias en común, esto es, si el valor del *Coeficiente de Jaccard* [53] (ver ecuación 3.5, donde X e Y son los conjuntos de instancias cubiertas por cada solución) entre ellos es mayor que 0,5, si no, se dice que las soluciones son no dominadas.

$$jaccard(X, Y) = \frac{X \cap Y}{X \cup Y} \tag{3.5}$$

Con esta restricción, dos soluciones serán comparadas únicamente si la intersección de las instancias que cubren es mayor al 50 % de su soporte. Entonces, la dominancia de Pareto se reformula como:

$$\begin{aligned} i \preceq j \quad \text{sii} \quad & (jaccard(f_2(i), f_2(j)) > 0{,}5) \wedge \\ & f_1(x) \geq f_1(y) \wedge f_2(x) \geq f_2(y) \wedge (f_1(x) > f_1(y) \vee f_2(x) > f_2(y)) \end{aligned} \tag{3.6}$$

donde f_1 es la función objetivo que mide la cohesión de un cluster y f_2 es la función objetivo que mide la cobertura del cluster.

3.5. Evaluación de los clusters generados

Con la finalidad de poder medir la bondad de los resultados obtenidos por nuestra metodología, realizaremos una comparación con los resultados obtenidos utilizando otras técnicas de clustering. La implementación ha sido codificada en el lenguaje `Eiffel` (`ISE Eiffel v4.5`) y ejecutada en un Intel® Pentium® 4 CPU 3,20GHz. En primer lugar, compararemos con los resultados obtenidos por el algoritmo APRIORI introducido en la Sección 2.3.3.1, para luego compararlos con otra metodología de descubrimiento de subgrupos en bases de datos estructuradas conocida como SUBDUE y explicada en la Sección 2.3.3.2.

En el caso del dominio geométrico, utilizaremos una base de datos de 100 instancias de pilas de figuras geométricas elegidas de forma aleatoria. La población inicial del algoritmo estará compuesta por subárboles seleccionados también de forma aleatoria de la base de datos. Los parámetros utilizados por los algoritmos para este dominio se muestran en la Tabla 3.1. El algoritmo CC-EMO se ha ejecutado diez veces con diferentes semillas y el promedio de estas ejecuciones es el que se muestra en adelante.

Parámetro	Valor
Tamaño de la población	100
Número de evaluaciones	10000
Probabilidad de cruce	0,6
Probabilidad de mutación	0,2

Tabla 3.1: Parámetros para dominio geométrico

El primer paso en la comparación consiste en ver de forma gráfica los resultados obtenidos. Para ello, graficamos el conjunto Pareto para cada uno de los enfoques utilizados (APRIORI, SUBDUE y CC-EMO) en el espacio de objetivos, como puede verse en la Figura 3.11. Para generar el conjunto Pareto correspondiente al APRIORI, se ha ejecutado este algoritmo y se han extraído las mejores primeras 100 soluciones (el tamaño de la población es igual para todos los enfoques y corresponde a lo indicado en la Tabla 3.1). A este conjunto de soluciones se les ha aplicado el criterio de no dominancia y extraído finalmente el conjunto Pareto. De forma similar se ha obtenido el conjunto Pareto para SUBDUE, pero el conjunto de 100 soluciones corresponde a la unión de tres conjuntos de 33 elementos, cada uno asociado a las soluciones obtenidas optimizando el objetivo de tamaño de la subestructura, el número de instancias y el MDL, respectivamente.

El conjunto Pareto para el CC-EMO está compuesto de la unión de las soluciones obtenidas por las diez ejecuciones del algoritmo. Es necesario recordar que, a pesar de que se han dibujado soluciones que a primera vista parecerían estar dominadas por otras, el coeficiente de Jaccard entre las instancias cubiertas

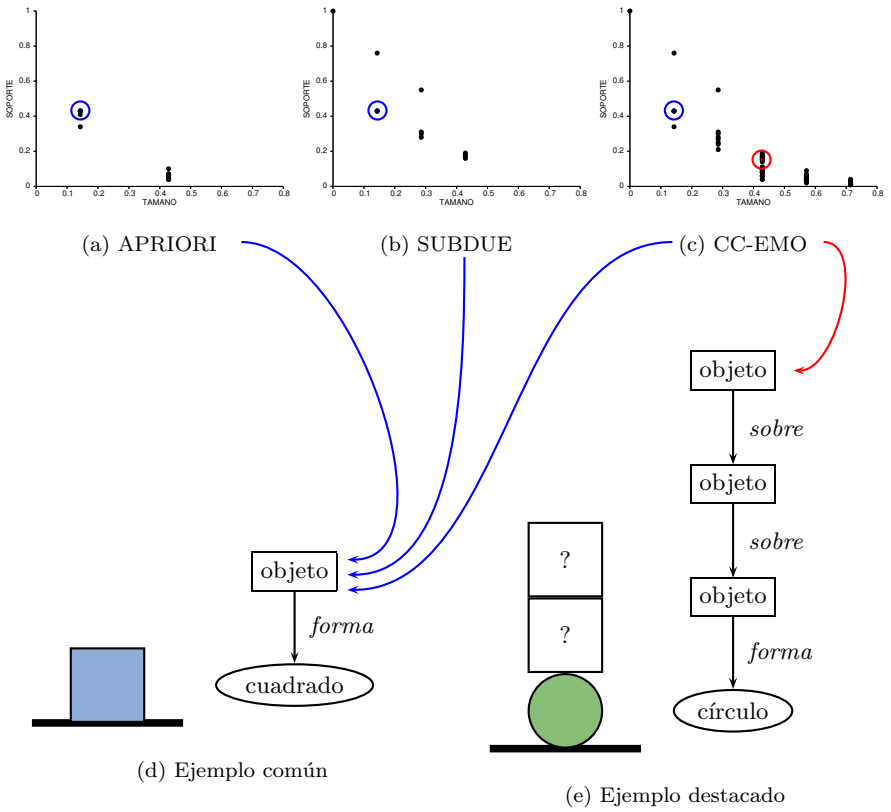


Figura 3.11: Frentes de Pareto obtenidos para el dominio geométrico. A pesar de que hay soluciones dibujadas que parecerían ser dominadas por otra solución, esto no es así ya que no pertenecen al mismo vecindario en el espacio de variables.

por cada solución es menor a 0,5 y, por lo tanto, no pueden ser comparadas entre sí (recordar la discusión de la Sección 3.4).

Como es fácil de observar, APRIORI y SUBDUE obtienen un número limitado de soluciones. En el caso de APRIORI, éste realiza la búsqueda de soluciones optimizando un único objetivo, el número de instancias cubiertas, no siendo capaz de encontrar las soluciones óptimas. Esto es debido a que el sistema APRIORI no ha sido diseñado para trabajar con bases de datos estructuradas.

En el caso de SUBDUE, éste identifica correctamente un conjunto de soluciones interesantes, pero pierde varios clusters valiosos. No puede obtener un conjunto adecuado de soluciones de compromiso debido a que el principio de

	$\mathcal{S}(X)$	$\mathcal{M}_3^*$
APRIORI	1,007766	0,82202
SUBDUE	2,72272	1,12631
CC-EMO promedio (*desvest*)	3,18960 (*0, 08025*)	1,30384 (*0, 03789*)

(a) Métricas $\mathcal{S}$ y $\mathcal{M}_3^*$

	$\mathcal{M}_2^*$	$\|X\|$
APRIORI	14	14
SUBDUE	12	13
CC-EMO promedio (*desvest*)	25,92379 (*2, 60579*)	26,70000 (*2, 62679*)

(b) Métrica $\mathcal{M}_2^*$

$\mathcal{C}(X', X'')$	APRIORI	SUBDUE	CC-EMO promedio (*desvest*)
APRIORI	-	0,00000	0,00000 (*0, 00000*)
SUBDUE	0,07143	-	0,00000 (*0, 00000*)
CC-EMO promedio (*desvest*)	0,07857 (*0, 02259*)	0,00000 (*0, 00000*)	-

(c) Métrica $\mathcal{C}$

$\mathcal{ND}(X', X'')$	APRIORI	SUBDUE	CC-EMO promedio (*desvest*)
APRIORI	-	10,00	10,80 (*0, 42164*)
SUBDUE	10,00	-	3,30 (*1, 25167*)
CC-EMO promedio (*desvest*)	24,60 (*2, 45855*)	16,90 (*1, 66333*)	-

(d) Métrica $\mathcal{ND}$

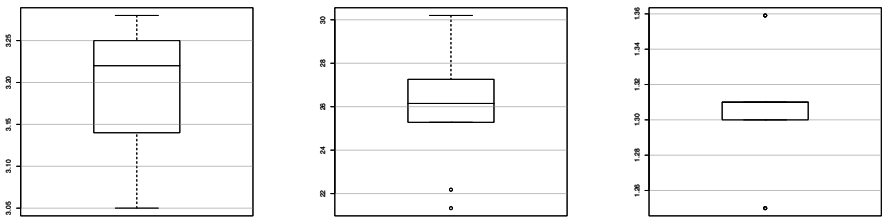
Tabla 3.2: Resultados de las métricas para el dominio geométrico.

MDL es en realidad, una suma ponderada de objetivos –tamaño y soporte–, y se sabe que este tipo de enfoque produce soluciones subóptimas, como hemos comentado en la Sección 2.5.1 [33].

Finalmente, el conjunto Pareto de CC-EMO contiene la mayor parte de las soluciones encontradas por el sistema APRIORI y el SUBDUE, y además obtiene una buena diversidad de soluciones de compromiso. También es interesante notar que el algoritmo CC-EMO cubre una porción del frente de Pareto correspondiente a aquellas soluciones con valores bajos de soporte pero con valores altos en tamaño, que los otros algoritmos no son capaces de descubrir.

Para verificar numéricamente estas afirmaciones, se calcularán diversas métricas sobre los resultados de los diferentes algoritmos, seleccionadas de entre los introducidos en la Sección 2.5.3.

Comenzaremos estudiando las métricas para conjuntos individuales. En la Tabla 3.2, se muestran los resultados de las métricas $\mathcal{M}_2^*$, $\mathcal{M}_3^*$ y $\mathcal{S}$. En la Figura 3.12 se muestran los boxplots asociados a las diez ejecuciones de CC-EMO para



(a) $\mathcal{S}$ (b) $\mathcal{M}_2^*$ (c) $\mathcal{M}_3^*$

Figura 3.12: Boxplots de las métricas $\mathcal{S}$, $\mathcal{M}_2^*$ y $\mathcal{M}_3^*$ para el CC-EMO en el dominio geométrico

estas métricas. La métrica $\mathcal{M}_2^*$ muestra la diversidad de soluciones encontradas en el conjunto Pareto y, como puede verse en la Tabla 3.2(b), CC-EMO obtiene el conjunto Pareto más diverso, lo cual también puede intuirse a la vista del frente del Pareto dibujado en la Figura 3.11. Por otro lado, la métrica $\mathcal{M}_3^*$ para un problema de optimización de dos objetivos se calcula simplemente como la diferencia de las soluciones más distantes del frente de Pareto. De los resultados de esta métrica, mostrados en la Tabla 3.2(a), y de la Figura 3.11, podemos inferir que CC-EMO cubre correctamente los extremos del frente de Pareto seguido por SUBDUE y APRIORI. Finalmente, la métrica $\mathcal{S}$ devuelve un valor numérico que representa la cobertura del área del Pareto. Como puede verse en la Tabla 3.2(a) y en la Figura 3.11, CC-EMO obtiene el mayor número y, por lo tanto, cubre mejor el espacio del frente del Pareto.

Ahora estudiaremos las métricas de comparación entre distintos conjuntos Pareto. La métrica $\mathcal{C}$ es aquella que muestra la dominancia entre las soluciones de los diferentes enfoques. Como se puede inferir de los conjuntos Pareto obtenidos, ninguna solución de CC-EMO es dominada por ninguna otra solución de los otros dos algoritmos como puede verse en la tercera columna de la Tabla 3.2(c) y en la Figura 3.13(a). El sistema APRIORI no domina a ninguno de los otros dos enfoques, mientras que SUBDUE supera a APRIORI pero no a CC-EMO. Ninguna solución del SUBDUE domina a las de CC-EMO y viceversa.

Finalmente, recordemos de la Sección 2.5.3 que la métrica $\mathcal{ND}$ nos dice el número de soluciones de un algoritmo que no dominan y no son iguales a las soluciones de la otra población. En otras palabras, cuenta el número de individuos de un conjunto Pareto que no hayan sido encontrados y no estén dominados por la otra población. De la Tabla 3.2(d) y la Figura 3.13(b), es fácil ver que CC-EMO descubre un número mayor de soluciones del Pareto que los

otros dos sistemas, a pesar que es cierto que tanto APRIORI como SUBDUE encuentran soluciones novedosas que CC-EMO no logra obtener. Sin embargo, la cantidad de estas soluciones (10,8 y 3,3 de la Tabla 3.2(d)) es mucho menor en comparación con la métrica $\mathcal{ND}$ para CC-EMO (24,6 y 16,9 de la Tabla 3.2(d)).

3.6. Compactación de la base de datos

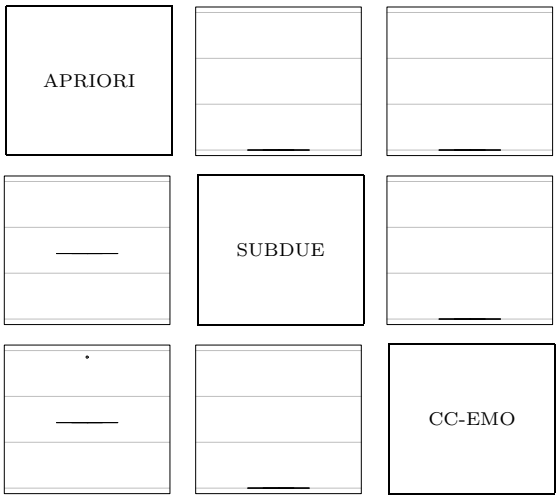
Debido a que no todos los clusters obtenidos en la etapa de clustering conceptual pueden ser igualmente relevantes para todos los usuarios de la metodología, es necesario depurar este conjunto. Es posible reducir la cantidad de soluciones y, por lo tanto, comprimir la base de datos de manera inteligente. Utilizando información externa e independiente de los datos, correspondiente a un agrupamiento de esta nueva información, llamado *clusters de control*, es posible seleccionar subconjuntos de clusters del conjunto total obtenido que sean más relevantes con respecto a estos clusters de control. Es decir, seleccionar para cada cluster de control, un grupo de clusters que expliquen a las mismas instancias de la base de datos de una manera diferente utilizando información independiente.

En algunos dominios, es posible tener información externa e independiente para realizar la compactación, pero en el caso del dominio geométrico no contamos con dicha información y por ello generamos estos datos al azar (clustering de control) con la única finalidad de mostrar el proceso a realizar.

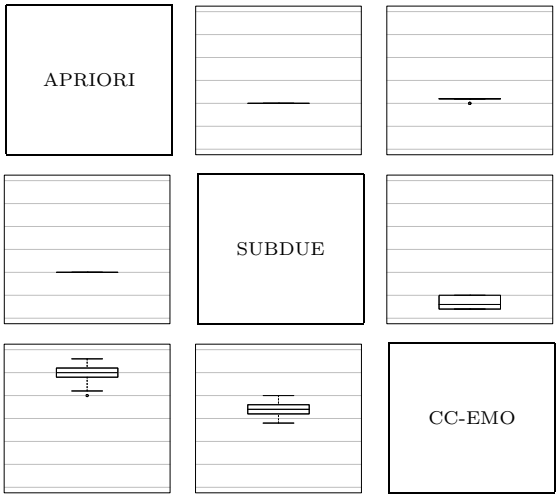
A partir de estos datos externos, podemos realizar una comparación entre los clusters obtenidos mediante CC-EMO y la clasificación externa. Esta comparación se realizará calculando la intersección de todos los clusters obtenidos por CC-EMO, a los cuales llamaremos *ClustersB*, contra aquellos del conjunto de control, llamados *ClustersA*. Para decidir si la intersección entre un ClusterA y un ClusterB es significativa, calcularemos su p-value utilizando la ecuación 3.7 [87]:

$$P(ClusterA, ClusterB) = 1 - \sum_{i=0}^{k-1} \frac{\binom{f}{i}\binom{g-f}{n-i}}{\binom{g}{n}} \tag{3.7}$$

donde f es el número total de instancias que pertenecen al ClusterA, n es el tamaño del ClusterB, k es el número de instancias pertenecientes a la intersección de ambos clusters (el de ClusterA y el de ClusterB), y g es el número total de instancias en la base de datos. Esta fórmula nos da la probabilidad de observar al menos k elementos del ClusterA en el ClusterB. Cuando más cercano a 1 sea el valor del p-value, mayor probabilidad hay de que la intersección sea simplemente por azar y no sea relevante. En contraposición, cuanto menor sea este valor, más relevante será la intersección. Para la comparación se utilizará un valor de umbral pequeño δ que permitirá decidir a partir de qué valor de p-value se considerará que las intersecciones obtenidas no son aleatorias. Este valor de umbral es específico para cada problema.

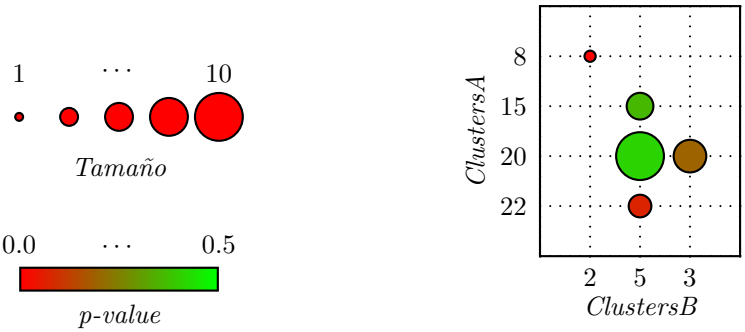


(a) Boxplots de la métrica $\mathcal{C}$



(b) Boxplots de la métrica $\mathcal{ND}$

Figura 3.13: Boxplots de las métricas $\mathcal{C}$ y $\mathcal{ND}$ para CC-EMO en el dominio geométrico



(a) Leyenda (b) Comparación

Figura 3.14: Ejemplo de intersección entre el clustering obtenido por la metodología CC-EMO y el clustering de control

En la Figura 3.14 se muestra un gráfico donde se compara el clustering obtenido por la metodología y el clustering de control. Se puede ver que, en la Figura 3.14(b), cada intersección se representa con un circulo. Sólo se grafican las intersecciones detectadas con un p-value por debajo del umbral seleccionado ($\delta = 0,5$). El eje horizontal representa los clusters de ClusterB mientras que el eje vertical muestra los clusters de ClusterA. Los círculos aumentan su tamaño a medida que la cantidad de elementos de la intersección crece. El círculo de menor tamaño corresponde a una intersección donde hay un solo elemento, mientras que el mayor círculo contiene 10 elementos. El color de los círculos también varía dependiendo del p-value correspondiente a la intersección. Cuanto menor sea el p-value, más rojo será el círculo y, por lo tanto, más relevante será la intersección; y cuanto más verde sea, menos relevante será. Sobre el rectángulo de muestra con la graduación de colores, se indica el rango de los p-value. En el ejemplo se ha seleccionado un δ de 0,5 y, por ello, el rango se define en el intervalo [0;0,5]. Se han presentado cinco intersecciones en el ejemplo: {(ClusterB #2, ClusterA #8), (ClusterB #5, ClusterA #15), (ClusterB #5, ClusterA #20), (ClusterB #5, ClusterA #22), (ClusterB #3, ClusterA #20)}. La intersección (ClusterB #5,ClusterA #20) es la que contiene la mayor cantidad de elementos, pero también es una de las menos relevantes debido a que el color del circulo es claramente verde. Por otro lado, la intersección (ClusterB #2, ClusterA #8) tiene pocos elementos pero, a su vez, es la más relevante ya que es la más rojiza. Las demás intersecciones ejemplifican diferentes situaciones donde el tamaño y el p-value varían.

Mediante esta comparación con el conjunto de control, es posible saber si los clusters obtenidos por nuestra metodología son similares a aquellos obtenidos en el clustering de la base de datos de control. Esta comparación se realizará

no sólo con los resultados de CC-EMO, sino también con los de los otros dos algoritmos con los cuales comparamos: APRIORI y SUBDUE.

A la luz de esta comparación, es posible realizar una depuración de los resultados obtenidos, ya que el conjunto Pareto generado puede llegar a tener varias soluciones muy similares con objetivos contrapuestos que pueden ser subsumidos por una única solución. Por lo tanto, nuestra metodología identifica aquellas subestructuras correspondientes a ClusterB que resultan *indistinguibles* desde el punto de vista del ClusterA, y las comprime en una única subestructura. Para ello, se realizan los siguientes pasos para cada cluster de control (ClusterA) (ver Algoritmo 3.1): (1) se seleccionan aquellos clusters de ClusterB con un p-value menor al umbral dado, que intersecta al cluster de ClusterA, y se tiene en consideración solamente aquellas instancias incluidas en el ClusterA; (2) se excluyen aquellos clusters de ClusterB que sean dominados luego de recalcular la no dominancia en base al nuevo conjunto reducido de instancias, en lugar de la totalidad de la base de datos (ver las componentes rosas de la Figura 3.15); y (3) se organizan los clusters de ClusterB en clases de equivalencia utilizando la relación de inclusión dada por la función *Cubre* (ver ecuación 3.3) (ver los componentes negros de la Figura 3.15). Luego de la reorganización de los clusters en varias ramas, se selecciona un único cluster, el más general, de cada una de ellas (ver los componentes verdes de la Figura 3.15). Esto es debido a que, a priori, sería ideal tener la unión de todos los elementos de la rama, pero dado que los elementos de una misma rama se relacionan por una función de inclusión, terminaremos eligiendo el concepto más general.

Algoritmo 3.1 Proceso de compactación

COMPACTACIÓN (C ClustersB, D ClustersA, δ umbral)

1: **para todo** $d \in D$ **hacer**
2: $Q \leftarrow \{c \in C \mid P(c,d) < \delta\}$
3: $Q' \leftarrow \{c \in Q \mid (\neg\exists x \in Q \mid (x \prec c))\}$
4: Organizar en clases de equivalencia $Y = \{Y_1, \ldots, Y_r\}$ a los $c \in Q'$ en base a la relación *cubre* (ver ecuación 3.3)
5: $Q'' \leftarrow \emptyset$
6: **para todo** $Y_i \in Y$ **hacer**
7: $Q'' \leftarrow Q'' \cup \{c \in Y_i \mid (\forall x \in Y_i \mid c \text{ es más general que } x)\}$
8: **fin para**
9: Devolver Q''
10: **fin para**

El enfoque anterior distingue nuestra metodología de las técnicas clásicas de aprendizaje supervisado, en las que las estructuras no se construyen en base a su habilidad para distinguir ejemplos etiquetados o clases, sino que pueden ser comprimidas en base a información independiente disponible. Es más, nuestro enfoque utiliza una compresión flexible, donde las estructuras pueden ser

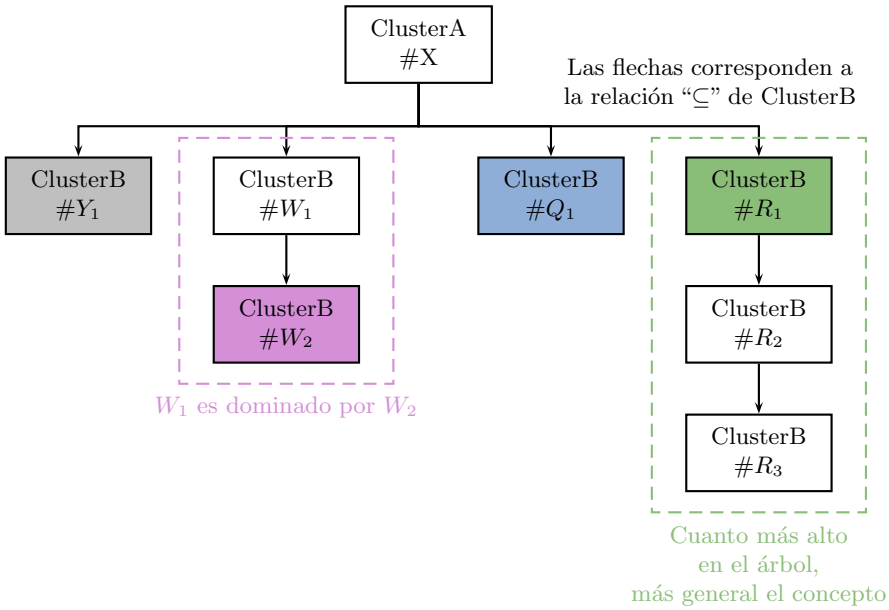


Figura 3.15: Ejemplo de compactación. La jerarquía viene dada por la relación entre los ClusterA y ClusterB. Los clusters coloreados corresponden a aquellos que se mantienen finalmente.

comprimidas y descomprimidas de acuerdo a los datos experimentales de comparación.

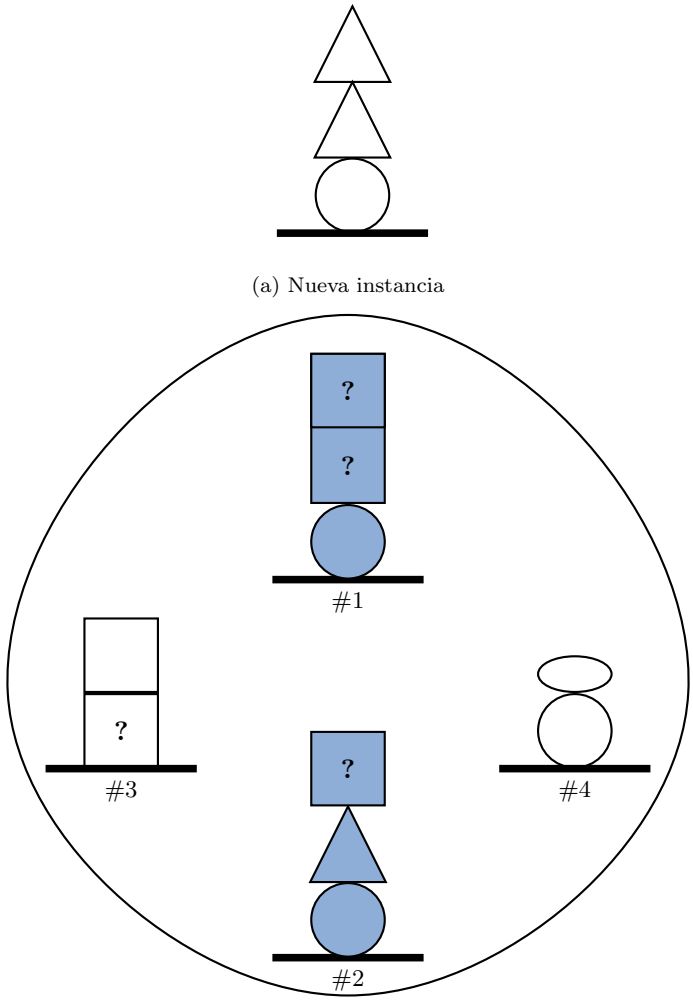
3.7. Predicción

CC-EMO utiliza un método de clasificación no-supervisado difuso basado en prototipos de k-vecinos (knn) [18] para predecir la clasificación de nuevas instancias, utilizando los conceptos extraídos en la fase de clustering y depurados durante la etapa de compactación. Para calcular la pertenencia de una instancia x_q a un conjunto de subestructuras I previamente identificadas, se utilizará la ecuación 3.8.

$$knn(x_q, V_1, \ldots, V_I) = V_i \in \{1, \ldots, I\} \ / \ \mu_{i,q} = max\{\mu_{q,1}, \ldots, \mu_{I,q}\}, \ \forall i \quad (3.8)$$

donde $\mu_{i,q}$ representa el grado de pertenencia de la instancia x_q a la subestructura V_i. Las instancias pueden contribuir a más de una subestructura con un grado de pertenencia μ diferente.

El grado de pertenencia $\mu_{i,q}$ entre una instancia x_q y una subestructura V_i es cero si la subestructura no cubre a la instancia. En el caso de que la



(a) Nueva instancia

(b) Subestructuras

Figura 3.16: Ejemplo de predicción para nuevas instancias. La instancia dada en (a) está cubierta por las subestructuras pintadas en celeste

subestructura si la cubra, entonces el grado de pertenencia será equivalente al objetivo de especificidad utilizado durante el proceso de búsqueda.

Por ejemplo, en la Figura 3.16(b) tenemos un conjunto de 4 subestructuras

($I = 4$) y en la Figura 3.16(a) tenemos una nueva instancia la cual queremos clasificar (x_q). Reemplazando en la fórmula tendremos:

$$knn(x_q, V_1, \ldots, V_4) = V_2 \ / \\ \mu_{i,q} = max\{\mu_{1,q}, \mu_{2,q}, \mu_{3,q}, \mu_{4,q}\} = max\{0{,}43; 0,57; 0; 0\} = 0,57$$

con lo cual clasificaremos a la instancia x_q con la subestructura V_2 ya que la instancia tiene un mayor grado de pertenencia utilizando esa subestructura.

Es interesante observar que las subestructuras descubiertas por la metodología no permiten obtener conceptos de cluster como ser *"dos objetos iguales sobre otro diferente"*. Si al usuario le interesara descubrir esa clase de conceptos, necesitaría modificar el sistema de modelado de los objetos. Si, por otra parte, al usuario le interesara extraer los conceptos de cluster que abarquen las descripciones con mayor cantidad de elementos, debería modificar los objetivos a optimizar, por ejemplo, contabilizar el tamaño de la subestructura solamente como la cantidad de nodos de tipo objeto. De esta manera, se realiza la retroalimentación y se vuelve a utilizar la metodología, y así sucesivamente hasta lograr obtener los resultados deseados.

3.8. Comentarios finales

En el área de la biología, se utilizan grandes repositorios de datos que contienen información de secuencias de ADN o proteínas de genomas completos. Esto es debido, en gran parte, a los avances en la Biología Molecular y en el equipamiento disponible para la investigación en este campo, que ha permitido la rápida secuenciación de grandes porciones de genomas de diversas especies. Estas bases de datos contienen información estructurada, ya que describen no sólo los objetos de las mismas, sino también las relaciones que existen entre ellos. A pesar del aumento constante de información en estos repositorios, se hace casi imposible, para un ser humano, extraer información útil de ellos. Se han desarrollado diversas técnicas de minería de datos para poder revelar información relevante que se encuentra oculta. Sin embargo, estas técnicas no funcionan correctamente con representaciones de datos estructurados. Debido a ello, existe la necesidad de crear técnicas que permitan analizar y descubrir nuevos conceptos, que difieran de los valores consenso, definidos mediante subestructuras en repositorios de datos estructurados.

Para ello, hemos propuesto una metodología llamada *Clustering Conceptual basado en Evolución MultiObjetivo (CC-EMO)* que extrae subestructuras interesantes de una base de datos estructurada considerando diferentes aspectos de las mismas. Dado un dominio, CC-EMO identifica patrones comunes representados por subestructuras mediante el uso de técnicas de optimización multiobjetivo.

Debido a esto, obtenemos el mejor conjunto de subestructuras que sean conjuntamente óptimas en especificidad y sensibilidad en una misma ejecución, a diferencia de otros métodos donde cada criterio de evaluación debe aplicarse individualmente.

Nuestra propuesta se diferencia de las técnicas clásicas de clustering en varios puntos: (1) CC-EMO encuentra las soluciones óptimas en múltiples criterios, lo cual evita los sesgos que pueden resultar de utilizar un esquema específico de peso sobre alguno de ellos; (2) CC-EMO permite la pertenencia de una instancia a más de una subestructura, a diferencia de otros enfoques de clustering que prematuramente fuerzan a las instancias a pertenecer a clusters disjuntos; (3) CC-EMO realiza una selección de características local para cada subestructura, dado que no todas las características son relevantes para todos los grupos, en contraposición a los enfoques que filtran o reducen características para todos los posibles clusters; (4) CC-EMO tiene una naturaleza multimodal que permite la descripción alternativa de un sistema, brindando de varias soluciones adecuadas, recuperando entonces soluciones localmente óptimas, a diferencia de otros métodos que están focalizados en un único óptimo; y (5) CC-EMO difiere de los métodos de aprendizaje supervisado ya que no realiza un agrupamiento basado en una sola clase dada por un experto. Más aún, CC-EMO minimiza el número de subestructuras utilizando una estrategia de compactación flexible que agrupa aquellas que resulten similares, agrupando dinámicamente en base a un criterio dado. Esto lo realiza basándose en la capacidad de describir objetos utilizando características independientes a las aplicadas en el proceso de aprendizaje, en lugar de enfoques que utilizan una compactación irreversible de la base de datos.

Hemos aplicado la metodología a un dominio sencillo basado en [54] y comparado con dos técnicas de descubrimiento de subgrupos actuales –APRIORI y SUBDUE–. Los resultados obtenidos por CC-EMO superan, cuantitativamente, a los otros dos enfoques en todas las métricas utilizadas en la comparativa, mientras que, cualitativamente, se puede observar que en varios casos los resultados obtenidos han sido validado experimentalmente y otros sujetos a próxima experimentación.

Capítulo 4

Aplicación a organismos procariotas

En este capítulo, se aplicará la metodología propuesta al problema de regulación genética en organismos procariotas. Para ello, se utilizará una base de datos que contiene información sobre diversos genes que están regulados por el sistema regulatorio de dos componentes PhoP/PhoQ, tanto en el genoma de *Escherichia coli (E. coli)* como en el de *Salmonella enterica (Salmonella)*. El sistema PhoP/PhoQ es un buen ejemplo de regulación procariota ya que controla la expresión de un gran número de genes en varias familias de *Enterobacterias*. Debido a que PhoP y PhoQ pertenecen al mismo operón, se utilizarán los términos "inducido por el sistema regulatorio de dos componentes PhoP/PhoQ" e "inducido por *phoP*" indistintamente.

El capítulo actual se dividirá en varias secciones siguiendo el esquema utilizado en el anterior. Se comenzará con una introducción al problema desde el punto de vista biológico, para luego mostrar en detalle los pasos de la metodología en su aplicación al problema tratado. Finalmente, se realizará una evaluación de los resultados obtenidos, tanto desde el punto de vista computacional como desde el experimental.

4.1. Problema biológico: *regulación genética*

Uno de los desafíos de la era post-genómica consiste en entender cuándo y cómo se expresan los genes de un genoma. El concepto básico subyacente en el control de la transcripción en bacterias se basa en dos tipos de secuencias de ADN: secuencias que codifican productos que actúan en *trans* y secuencias que codifican productos que actúan en *cis*. Cuando los elementos regulatorios son de

naturaleza y origen diferente a la secuencia genética a controlar, la regulación es de tipo *trans*, mientras que cuando el elemento regulador transcripcional es parte de la cadena nucleotídica donde se localiza el gen a regular, se denomina regulador *cis* [63].

La descripción de los elementos que participan en el proceso de regulación genética resulta esencial para poder descubrir las diferencias entre regiones reguladoras que se encuentran co-reguladas y, por lo tanto, ayudan a comprender los mecanismos subyacentes por los cuales los sistemas regulatorios controlan la expresión de los genes. En otras palabras, las descripciones de las regiones reguladoras relacionadas con los procesos de *binding* y transcripción pueden ayudar a una mejor comprensión de la dinámica de las redes transcripcionales y, a futuro, permitirían realizar predicciones del comportamiento de otros genes. Además, estas descripciones serán esenciales para revelar las diferencias en el control entre genes de distintas especies que no han sido detectados por medio de una comparación global de las bases de datos de los diferentes genomas.

El problema consiste entonces en saber distinguir los elementos que componen una región regulatoria de un gen de la mejor manera posible, como pueden ser los sitios de *binding* o los promotores. En general, los métodos computacionales se han concentrado en identificar propiedades independientes basadas en secuencias consenso que caracterizan diferentes aspectos en los procesos de regulación, tratando de homogeneizar las características de las regiones reguladoras, aún incluso entre especies [88]. Sin embargo, es difícil descubrir las posibles diferencias fenotípicas entre las regiones reguladoras que se encuentren co-reguladas dentro de una red, al igual que entre especies cercanas resultado de la regulación diferencial de genes homólogos, debido al exceso de confianza en la alta conservación de las características regulatorias entre las regiones reguladoras y las especies. Además, las propiedades regulatorias no pueden tratarse de forma independiente ya que sus relaciones están intrínsecamente codificadas en el genoma.

En este capítulo, nos centraremos en el problema de regulación en los organismos procariotas, en particular *E. coli* y *Salmonella*, los cuales presentan una diferencia significativa en la organización de los genes con respecto a los organismos eucariotas. En las bacterias, los genes estructurales están organizados en grupos, mientras que en los eucariotas aparecen en forma individual. Este agrupamiento, llamado *operón*, permite que sean controlados de forma coordinada por medio de interacciones en un promotor único y, como resultado de las mismas, el grupo completo de genes es transcrito o no [5].

4.1.1. La fase de transcripción en organismos procariotas

Como hemos visto en la Sección 4.1, la transcripción implica la síntesis de una cadena de ARN que representa a una de las cadenas de la doble hélice del ADN. Con "representar" queremos decir que el ARN tiene una secuencia

idéntica a una de las cadenas del ADN, que se denomina *cadena codificante*. Esta cadena es complementaria a la otra, la cual proporciona un molde para su síntesis. En la Figura 4.1 se muestra, de forma esquemática, el proceso de transcripción. El inicio de este proceso es un paso muy importante ya que es el punto de regulación principal que utiliza la célula para seleccionar qué proteínas se expresan y a qué velocidad.

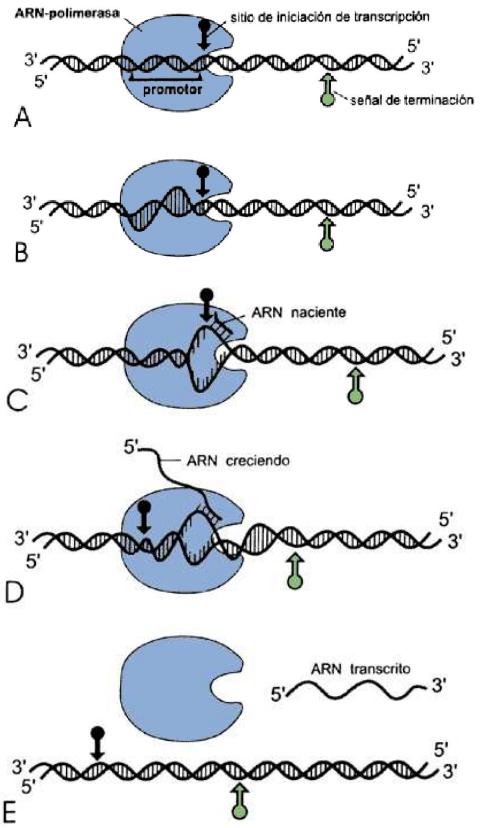


Figura 4.1: Proceso de transcripción del ADN a ARN

Cuando la ARN polimerasa se ha unido fuertemente al promotor del ADN, ésta separa la doble hélice exponiendo un pequeño tramo de nucleótidos de cada hebra. La ARN polimerasa y el ADN sufren una serie de cambios estructurales reversibles que los sitúan en un estado energéticamente más favorable. Con el ADN desenrollado, una de las dos hebras expuestas actúa como molde para el emparejamiento complementario de bases con los ribonucleótidos que se van incorporando, dos de los cuales han sido unidos por la ARN polimerasa

para iniciar la cadena de ARN. Una vez que se han incorporado al ARN los diez primeros nucleótidos aproximadamente, el factor σ relaja su fuerte unión a la ARN polimerasa y se separa de ella. En este proceso, la ARN polimerasa vuelve a sufrir cambios estructurales que le permiten desplazarse rápidamente, transcribiendo sin el factor σ. La elongación de la cadena continúa hasta que las enzima encuentra una segunda señal en el ADN, el *terminador*, donde la ARN polimerasa se detiene y libera el molde de ADN y la cadena de ARN recién sintetizada. Una vez que la ARN polimerasa se ha liberado en el terminador, vuelve a asociarse con una molécula libre de factor σ y puede buscar un nuevo promotor, en el cual volverá a iniciar la transcripción.

4.1.2. Diferentes clases de activadores

La mayoría de los activadores transcripcionales funcionan enlazándose corriente arriba[1] del sitio de inicio de la transcripción en la localización de los promotores. Por lo tanto, los activadores que reclutan a la ARN polimerasa a través de una interacción directa proteína-proteína se dividen en dos clases: *clase I* y *clase II*. Los activadores *clase I* se enlazan en ubicaciones corriente arriba y funcionan realizando una interacción directa con una de las subunidades (αCTD) de la ARN polimerasa. Esta interacción es suficientemente flexible para permitirle el enlace de esta subunidad en diferentes posiciones. Por el contrario, los activadores *clase II* se enlazan a sitios de binding que se superponen con el promotor en la región -35 del mismo. El lugar de posicionamiento de los activadores clase II no puede variar ya que están restringidos a la localización de uno de sus dominios σ [5].

4.1.3. Promotores de ADN en organismos procariotas

Para transcribir correctamente un gen, la ARN polimerasa debe poder reconocer en qué punto del genoma comienza el gen y dónde acaba. Los procesos de inicio y terminación de la transcripción implican una serie de complicadas transiciones estructurales en las moléculas de proteína, de ADN y de ARN que intervienen. De hecho, una comparación de muchos promotores bacterianos diferentes revela que tienen secuencias de nucleótidos heterogéneas. Sin embargo, todas ellas contienen subsecuencias relacionadas, que reflejan parcialmente los aspectos del ADN que son reconocidos directamente por el factor σ. En general, estas características comunes se denominan *secuencias consenso*. Una secuencia de nucleótidos consenso se deduce de la comparación de muchas secuencias que tienen la misma función básica y se escoge el nucleótido que más se repite en cada posición. Por lo tanto, son como un resumen o "promedio" de un gran número de secuencias individuales de nucleótidos.

[1] Las secuencias que se encuentran precediendo al sitio de iniciación se encuentran "corriente arriba" de éste. Aquellas secuencias que se encuentran después del sitio de iniciación (en la secuencia que se transcribe) están "corriente abajo" del mismo.

Una razón por la que los promotores bacterianos difieren en su secuencia de nucleótidos es debida a que esta secuencia es la que determina la fuerza del promotor (es decir, el número de veces que se inicia la transcripción por unidad de tiempo). Los procesos evolutivos han ido modificando cada promotor de forma que la transcripción se inicie tan a menudo como sea necesario, generando un amplio espectro de promotores. Los promotores de los genes que codifican las proteínas más abundantes son mucho más fuertes que los que están asociados a genes que codifican proteínas menos frecuentes. Sus secuencias de nucleótidos son las responsables de estas diferencias.

Es posible generar secuencias consenso que resuman sus características más relevantes. La variación que presentan en sus secuencias de nucleótidos dificulta a los investigadores la labor de localizarlos mediante una sencilla inspección de la secuencia de nucleótidos del genoma. A menudo, hace falta tener información adicional, en parte procedente de la experimentación directa, para localizar con precisión estas señales de ADN de pequeña longitud contenidas en el genoma.

Las secuencias de los promotores son asimétricas y esta característica tiene consecuencias importantes en su localización en el genoma. Dado que el ADN es de doble hebra, en principio, a partir de cualquier gen se podrían transcribir dos moléculas de ARN diferentes, utilizando cualquiera de las dos hebras de ADN como molde. Sin embargo, un gen típico tiene un solo promotor y, por lo tanto, la ARN polimerasa sólo se puede unir en una única dirección. Por tanto, la ARN polimerasa no tiene más opción que transcribir una de las hebras de ADN ya que sólo puede sintetizar el ARN en dirección 5'-3'. Así pues, la elección en cada gen de la hebra molde está determinada no sólo por la localización, sino también por la orientación de su promotor. Las secuencias de los genes revelan que la hebra de ADN utilizada como molde para la síntesis de ARN varía de un caso a otro.

La división de labores entre el núcleo de la enzima que se encarga de la elongación de la cadena y el factor sigma que está implicado en la selección del promotor, plantea la pregunta de si habrá más de un tipo de sigma, cada uno especifico para una clase diferente de promotor. El factor más común, responsable de la transcripción de la mayoría de los genes en condiciones normales, es el σ^{70}. Los factores sigma alternativos σ^{32}, σ^{E} y σ^{54}, se activan en respuesta a cambios ambientales. Además, σ^{28} se usa para la expresión de genes flagelares durante el crecimiento normal, pero sus niveles de expresión responden también a cambios ambientales. Todos los factores sigma excepto σ^{54} pertenecen a la misma familia de proteínas y funcionan de la misma manera general descrita anteriormente.

En particular, el promotor σ^{70} –el cual se encuentra en la mayoría de los genes del organismo *E. coli*– contiene dos secuencias, situadas en el extremo 5' corriente arriba del primer nucleótido que se va a transcribir, que funcionan como centros promotores. Una de ellas, llamada *secuencia Pribnow* o *región -10*, contiene la secuencia consenso `TATAAT` y está centrada a -10 pares de bases

(pb)[2]. La segunda, llamada *región -35*, tiene la secuencia consenso `TTGACA`.

La característica más notable en las secuencias de promotores en *E. coli* es la ausencia de conservación en la secuencia de 60 pb que se asocia con la ARN polimerasa. En otras palabras, la secuencia de nucleótidos de la mayor parte del sitio de unión es irrelevante. No obstante, existen algunos fragmentos que resultan cruciales para su funcionamiento. La conservación únicamente de secuencias consenso de corta longitud es una característica típica de los sitios reguladores (como los promotores), tanto en los genomas procariotas como en los eucariotas.

En los promotores bacterianos hay cuatro fragmentos que se conservan: el sitio de iniciación y la secuencia en la región -10; la secuencia en la región -35; y la separación entre las regiones -10 y -35.

- El sitio de iniciación es, en más del 90 % de los casos, una purina. Es común que el sitio de iniciación sea la base central en la secuencia `CAT`, pero la conservación de este triplete no es lo suficientemente alta como para que se considere como una señal obligatoria.

- Justo corriente arriba del sitio de iniciación, hay una región de 6 pb que es reconocible en casi todos los promotores. El centro del este hexámero se encuentra aproximadamente a 10 pb corriente arriba del sitio de iniciación, aunque la distancia varía en los promotores conocidos desde 9 a 18 pb. El hexámero se denomina con frecuencia región -10, de acuerdo con su localización. El consenso de esta secuencia es `TATAAT` y se puede representar de la forma:

  $$T_{80}A_{95}T_{45}A_{60}A_{50}T_{96}$$

  donde el subíndice indica el porcentaje de aparición de la base encontrada con más frecuencia, variando entre el 45 % y el 96 %. Una posición en la que no hay una clara preferencia por ninguna base se indicaría con una `N`. Si la frecuencia de aparición indica su posible importancia para la unión de la ARN polimerasa, deberíamos esperar que las bases más importantes en la región -10 sean las `TA` iniciales altamente conservadas, y la `T` final casi completamente conservada.

- Otro hexámero altamente conservado se encuentra a 35 pb corriente arriba del sitio de iniciación. Éste se denomina región -35. El consenso es `TTGACA` y, de una forma más detallada, responde a la expresión:

  $$T_{82}T_{84}G_{78}A_{65}C_{54}A_{45}$$

[2]Las posiciones de las bases se numeran en ambas direcciones en referencia al sitio de iniciación al que se le asigna la posición +1.
Los números aumentan corriente abajo. La base situada justo antes del sitio de iniciación corresponde a la posición -1 y los números negativos se van incrementando corriente arriba.

- Las regiones -35 y -10 están separadas entre ellas por una distancia que oscila entre 16 y 18 pb en el 90 % de los promotores. Como excepciones, puede ser algo más corta y tener 15 pb de longitud o ser más larga y tener 20 pb. Aunque la secuencia en el trecho intermedio no es importante, la distancia es crítica para mantener los dos sitios a la separación adecuada para la geometría de la ARN polimerasa.

Un promotor "típico" depende de sus regiones -35 y -10 para ser reconocido por la ARN polimerasa, pero alguna de estas secuencias puede estar ausente (en casos excepcionales) en algunos promotores. En al menos algunos de estos casos, el promotor no puede ser reconocido por la ARN polimerasa en solitario, y la reacción requiere de la intervención de factores auxiliares que suplen la deficiencia en la relación intrínseca entre la ARN polimerasa y el promotor.

Una característica significativa de los promotores para cada enzima es que tienen el mismo tamaño y localización en relación con el sitio de iniciación, y que muestran secuencias conservadas sólo alrededor de los centros de las regiones -35 y -10. Las secuencias consenso para cada serie de promotores difieren unas de otras en una o ambas regiones -35 y -10. Esto significa que una enzima que contenga un determinado factor σ solamente puede reconocer su propia serie de promotores, por lo que la transcripción de los diferentes grupos es mutuamente excluyente. Por tanto, la sustitución de un factor σ por otro apaga la transcripción de la serie de genes antigua y enciende la transcripción de una nueva serie de genes.

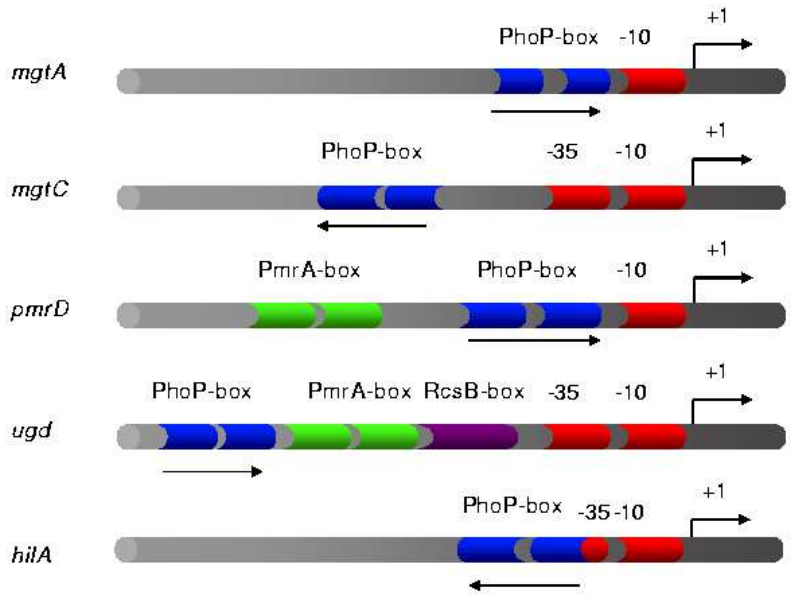
4.2. Construcción de la base de datos estructurada

Para la construcción de la base de datos, recuperamos las regiones intergénicas de los operones (la región anterior, corriente arriba del codón de inicio de la transcripción, y hasta 50 pb corriente abajo del mismo), los promotores y la información de los factores de transcripción de los genomas de *E. coli* y *Salmonella*, existentes en la base de datos RegulonDB [84], provista por H. Salgado. Además, contamos con información de la expresión diferenciada entre la cepa salvaje y la cepa *phoP* de *E. coli*, la cual experimenta las condiciones inducidas por el sistema regulatorio de dos componentes PhoP/PhoQ. Cada instancia de la base de datos corresponde a cada región regulatoria inducida por *phoP* de un determinado gen, como puede verse en la Figura 4.2.

Para el análisis de la red regulada por *phoP*, nos centraremos en las cinco características siguientes:

1. Los submotivos del PhoP-box[3] (*"Motivo"*).

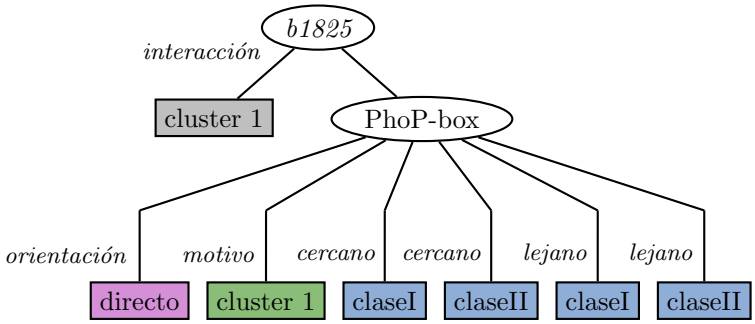
[3]Término asociado a la secuencia de ADN que es característica de las regiones donde hace



(a) Instancias de la base de datos antes al modelado

Nombre	Genoma	Motivo	Orientación	Interacción	Cercano		Medio		Lejano	
					claseI	claseII	claseI	claseII	claseI	claseII
b1825	*E. coli*	1	directo	1	0,42	0,37	0	0	0,56	0,55

(b) Instancia de la base de datos tras el modelado



(c) Instancia de la base de datos estructurada

Figura 4.2: Representación de la base de datos

2. La orientación del PhoP-box (*"Orientación"*).

3. La presencia de sitios de *binding* potenciales para 23 factores de transcripción seleccionados de los promotores regulados por *phoP* ("Interacción").

4. Los niveles inducidos por *phoP* en la expresión genética ("Expresión").

5. La distancia relativa entre el PhoP-box y el sitio promotor y su clase (*"Promotor"*).

A continuación se detallarán los pasos realizados para obtener cada una de las cinco características que conforman la base de datos. La mayor parte de la codificación ha sido realizada usando el lenguaje `perl` y utilizando los módulos para bioinformática de `bioperl v1.5`.

4.2.1. Submotivos del PhoP-box ("Motivo")

Para identificar diferentes modelos de los motivos del sitio de *binding* de PhoP, realizamos una partición del conjunto de datos, compuesto por genes que se sabe que están regulados por *phoP*, en 70 % para entrenamiento y el restante 30 % para prueba.

Debido a que los sitios de *binding* de phoP que tienen una semejanza débil al motivo consenso son difíciles de detectar, utilizaremos un proceso de varias etapas para incrementar la sensibilidad de la búsqueda sin disminuir su especificidad. Primero, codificaremos el motivo del sitio de *binding* de PhoP utilizando como modelo una matriz de pesos para cada posición de la secuencia nucleotídica (matriz de pesos). Se agrupan secuencias que son similares a PhoP en base a esta matriz de pesos, usando el método de clustering difuso C-means [18] (ver Sección 2.3), aplicándole una transformación a los nucleótidos del patrón correspondiente a la matriz de pesos, convirtiéndolos en variables *dummy* [35], y considerando una combinación de índices para estimar el número de clusters [18, 19].

Este modelo consenso inicial se ha utilizado para buscar en las secuencias intergénicas de las regiones promotoras de *E. coli* y *Salmonella* (hasta 800 pb corriente arriba y desde 50 pb corriente abajo del codón de inicio predicho por la información extraída de la base de datos) en un esfuerzo por capturar miembros potenciales del regulador de *phoP* que detecte motivos débiles. Agrupando las secuencias que se asemejan con el modelo inicial dado por el grado de similaridad del modelo a la partición difusa, se detectan cuatro subconjuntos ensamblados en una nueva familia de modelos, **M1** a **M4** (ver Tabla 4.1). Se define además el cluster **M0** que incluye a aquellos genes donde no se encuentra ningún submotivo de PhoP.

binding una proteína reguladora, en este caso PhoP.

4.2.2. Orientación y modelado del PhoP-box ("Orientación")

Se ha detectado en [100] que, a diferencia de los promotores de *phoP* y *mgtA*, varios promotores regulados por *phoP*, como *ompT* y *yhiW* de *E. coli* y *mig-14*, *pipD* y *pagC* de *Salmonella*, contienen posibles sitios de *binding* de PhoP en ambas orientaciones. Algunos de éstos ya se han propuesto para ser regulados indirectamente por *phoP*, debido a la falta de la orientación directa del sitio de binding.

Para identificar correctamente el sitio de *binding* de PhoP y su orientación, agrupamos en forma iterativa las secuencias ubicadas corriente arriba del comienzo de los genes que tienen una *coherencia de expresión* similar [73].

La coherencia de expresión es una métrica utilizada para calcular cuán compacto es un conjunto de expresión de genes. Dado un conjunto de genes K, que comparten un motivo particular y un mismo conjunto de datos de expresión, se calculan las distancias euclídeas (ver ecuación 2.13) entre la media y los perfiles de expresión para cada $P = 0{,}5 \times K \times (K-1)$ par de genes. La coherencia de expresión de un motivo se define como p/P, donde p es el número de pares de genes cuya distancia euclídea supera un pequeño umbral.

Para ello, construimos un modelo de matrices de pesos utilizando la métrica de coherencia de expresión como función de optimización, como se explica a continuación: (1) Seleccionamos como conjunto de datos a todas las secuencias ubicadas corriente arriba de los genes. (2) Aplicamos el programa Consensus [88] con longitudes de patrón entre 14 y 30 pb. (3) Agrupamos los patrones consenso correspondientes a las distintas matrices obtenidas utilizando el algoritmo de clustering difuso C-medias, aplicándoles una transformación a los nucleótidos de estos patrones para convertirlos en variables *dummy* [35]. (4) Evaluamos la sinergia[4] entre los clusters y la expresión de sus miembros mediante la coherencia de expresión. (5) Seleccionamos el grupo con una mayor sinergia. (6) Mientras la distancia intra-clustering del grupo seleccionado sea mayor a un umbral dado, construimos patrones de consenso para el grupo en base a las secuencias que pertenecen al cluster y volvemos a (2); en otro caso, devolveremos la mejor matriz final basada en su e-value [50].

Adicionalmente, se incorporan restricciones para mejorar el poder de discriminación de la función de optimización que guía la heurística. Cada cluster identificado en (2) ha sido utilizado para buscar patrones similares en una colección de sitios de binding de factores de transcripción diferentes a PhoP, los cuales fueron obtenidos de la base de datos RegulonDB y de publicaciones científicas. Cualquier similaridad con estos motivos se ha considerado como un falso positivo (FP) y utilizado como una estrategia de depuración.

[4]La sinergia es la integración de elementos que da como resultado algo más grande que la simple suma de éstos, es decir, cuando dos o más elementos se unen sinérgicamente crean un resultado que aprovecha y maximiza las cualidades de cada uno de los elementos individuales.

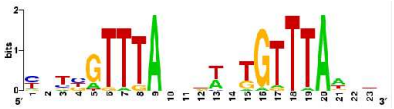
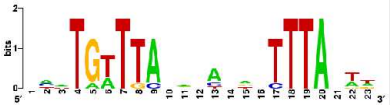
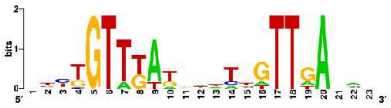
Consenso global	T	TGTTTA	TGTG[TG]	TGTTTA	[AT]	
M1	T	TATT[GT]A	CGTTC	TGTTTA	T	
M2	A	[GTC]TTTA	TG[AT]TT	TGTTTA	A	
M3	A	TGTTTA	[AG]A[AT]A[CT]	[ATG]GTTTA	[AT]	
M4	T	TGTTTA	TAATT	TGTTGA	T	

Tabla 4.1: Motivos del sitio de binding de PhoP.

Una vez determinados los modelos, realizamos una búsqueda utilizando el programa Patser [88] en las regiones intergénicas de los genomas completos de *E. coli* y *Salmonella* en ambas orientaciones, considerando utilizando un umbral reducido correspondiente a dos desviaciones estándar por debajo de la media obtenida en el conjunto de entrenamiento [74]. Se admite la presencia de más de un sitio candidato de binding de PhoP en una misma región promotora.

4.2.3. Sitios de *binding* de los factores de transcripción ("Interacción")

Para el desarrollo de los modelos correspondientes a los sitios de binding para los diferentes factores de transcripción existentes en la base de datos RegulonDB, se siguió la siguiente secuencia de pasos: (1) Generamos matrices de pesos para cada factor de transcripción utilizando el programa Consensus y seleccionamos la mejor matriz final para motivos de longitud entre 14 y 30 pb (utilizando las opciones de simetría de matrices [84]), en el caso que no se haya especificado una longitud específica. (2) Buscamos estos modelos con el programa Patser en las regiones intergénicas de los genomas de *E. coli* y *Salmonella*, utilizando la medida de rendimiento global [14] (ver Sección 4.2.5.2) como umbral. Luego, caracterizamos las distancias entre los distintos sitios de *binding* de los factores de transcripción en la misma región promotora para cada una de las regiones promotoras presentes en la base de datos RegulonDB. (3) Para cada región promotora, construimos un histograma de distancias entre los distintos sitios de *binding* de los factores de transcripción. (4) Ajustamos el histograma utilizando una función de pertenencia difusa. (5) Caracterizamos apropiadamente las distancias entre el posible PhoP-box y otros posibles sitios de *binding* de sitios de transcripción detectados en la misma región, utilizando los modelos aprendidos a partir de los histogramas. Finalmente, agregamos (2) y (5) utilizando operadores de lógica difusa.

4.2.4. Expresión de genes ("Expresión")

El análisis de microarray intenta identificar genes que se expresan diferencialmente bajo condiciones diferentes. El desafío de analizar estos datos es decidir qué constituye un cambio considerable. El enfoque usual consiste en analizar la distribución de los niveles de expresión y utilizar la estadística para ayudar a definir un umbral. Esto funciona razonablemente bien con grandes conjuntos de datos y para genes que muestran un gran cambio en sus niveles de expresión. Sin embargo, la distribución total de los niveles de expresión puede ser en realidad una mezcla de dos o más distribuciones independientes que incluyen genes que exhiben grandes cambios en la expresión, como también aquellos con menores cambios en sus niveles. Por lo tanto, la aplicación de técnicas estadísticas al conjunto total de los datos puede resultar inapropiada para describir el conjunto de

genes.

Para compensar este sesgo, utilizamos un proceso en dos etapas para el análisis de genes regulados por *phoP*. Primero, identificamos grupos de genes cuya expresión muestra diferencias estadísticamente significativas entre la cepa salvaje y *phoP* de *E. coli* en cuatro experimentos GeneChip [100]. Este agrupamiento se realiza utilizando una variación del método de clustering difuso C-medias [19, 40] (ver más adelante) y una combinación de índices de validación para estimar el número de clusters [18, 19]. Segundo, construimos submodelos para tales grupos de expresión calculando sus centroides con la ecuación 2.12 y buscando en el genoma de *E. coli* para caracterizar su similaridad con cada modelo, utilizando la ecuación 2.13. La expresión de cada gen se ha representado como un vector de valores, definidos en el intervalo $[0, 1]$, que indican la similaridad entre la expresión del gen y cada submodelo.

Agrupando los niveles de expresión del conjunto semilla mediante técnicas de clustering difuso, encontramos tres patrones diferentes, dos representando patrones regulados positivamente y uno que define patrones regulados negativamente. Se han modelado los tres patrones en base a los centroides de cada cluster. Los tres modelos que describe la característica "expresión" son: **E1**, correspondiente a los genes regulados positivamente incluyendo *phoP* y *phoQ*, y otros genes regulados por *phoP*, en forma canónica; **E2**, consistente principalmente en genes regulados positivamente, con niveles de expresión más bajos que los de E1; y **E3** que incluye los genes regulados negativamente. Luego, se han utilizado estos modelos para re-examinar los datos de microarray, utilizando como medida la similaridad de modelos. Esto ha permitido recuperar genes adicionales cuyos niveles de expresión son muy débiles y no han podido ser detectados utilizando los umbrales basados estrictamente en filtros estadísticos.

4.2.5. Sitio de *binding* de la ARN polimerasa ("Promotor")

Dado que los promotores σ^{70} del organismo *E. coli* presentan secuencias suficientemente conservadas, se puede llegar a definir un modelo de esta clase de promotor. Debido a que la información biológica no es muy precisa y, en algunos casos, es incompleta, haremos uso de una herramienta que nos permita modelar esta clase de situaciones. Por lo tanto, aprovecharemos la habilidad de la lógica difusa [56] para definir motivos incompletos e imprecisos. Otra ventaja de utilizar esta clase de técnicas es la flexibilidad e interpretabilidad de sus representaciones.

De la descripción biológica de un promotor σ^{70}, podemos extraer tres patrones conservados y dos distancias entre ellos que pueden utilizarse como modelos para la detección de promotores. Los patrones son: la región -10 (`TATAAT`), la región -35 (`TTGACA`) y el sitio de inicio de la transcripción (TSS). Las distancias son las correspondientes a la separación entre las regiones -35 y -10, y entre la región 10 y el TSS. Sin embargo, el tercer y último patrón, el TSS, es muy

sencillo de encontrar por azar en una secuencia de ADN. Esto resulta en una gran desventaja para la detección de este modelo, ya que la gran mayoría de las instancias detectadas de este patrón son falsos positivos debido, en gran medida, a la corta longitud que presenta. Ésta es la razón por la cual algunos algoritmos deciden no utilizar este motivo como parte de la búsqueda, y por eso nosotros tampoco lo utilizaremos.

Si realizamos una búsqueda en forma manual de los patrones descritos en una secuencia de tamaño reducido, encontraremos que existen varias soluciones posibles. Cada una de ellas es mejor que las otras en uno o más patrones y peor en otros. Por lo tanto, no es posible determinar cuál de estas soluciones es la mejor sin tener que realizar el experimento biológico para comprobarlo. Este hecho convierte el problema de reconocimiento de patrones de promotores de genes en un problema *multiobjetivo*. Además, cabe destacar que también es un problema *multimodal* ya que existe más de una solución posible al problema presentado para cada patrón individualmente.

Nuestra propuesta de modelado representa cada promotor, compuesto por las regiones -10 y -35 y la distancia que las separa, como tres modelos difusos parametrizados M^1_α, M^2_α, y M^3_α, donde α representa un modelo aproximado cuya función de pertenencia se aprende a partir de distribuciones de datos [56, 72].

Por lo tanto, para poder resolver el problema de predicción de promotores, se han desarrollado tres modelos difusos diferentes. Los dos primeros, correspondientes a las regiones -10 y -35, se han obtenido utilizando la frecuencia consenso nucleotídica como conjuntos difusos discretos [56]. De esta manera, el modelo difuso asociado al patrón `TTGACA`, M^1_α, se formula como:

$$M^1_\alpha(x) = \mu_{\texttt{TTGACA}}(x) = G(\mu^1_1(x_1), \ldots, \mu^1_6(x_6)) \tag{4.1}$$

donde el conjunto difuso discreto correspondiente a cada nucleótido que compone el motivo se obtiene de la distribución de probabilidades asociada, es decir, de la probabilidad de aparición de cada una de las cuatro letras del alfabeto en la posición que ocupa; y G el operador de agregación difuso [72, 56]. En esta memoria, hemos utilizado la media aritmética como operador de agregación.

	T	T	G	A	C	A
A	3	10	3	**58**	32	**54**
C	9	3	14	13	**52**	5
G	10	5	**68**	10	7	17
T	**78**	**82**	15	20	10	24

(a) Motivo `TTGACA`

	T	A	T	A	A	T
A	3	**89**	26	**59**	**49**	3
C	8	3	10	12	21	5
G	7	1	12	15	11	2
T	**82**	7	**52**	14	19	**89**

(b) Motivo `TATAAT`

Tabla 4.2: Distribución nucleotídica de los motivos obtenida de [48].

Por ejemplo, tomando en cuenta la distribución nucleotídica mostrada en la Tabla 4.2, extraída de [48], el conjunto difuso del primer nucleótido del motivo se define como $\mu_1^1(x_1) = \mathtt{A}/0,03 + \mathtt{T}/0,78 + \mathtt{G}/0,10 + \mathtt{C}/0,09$, y los otros conjuntos difusos correspondientes a las posiciones 2-6 se calculan de una manera análoga.

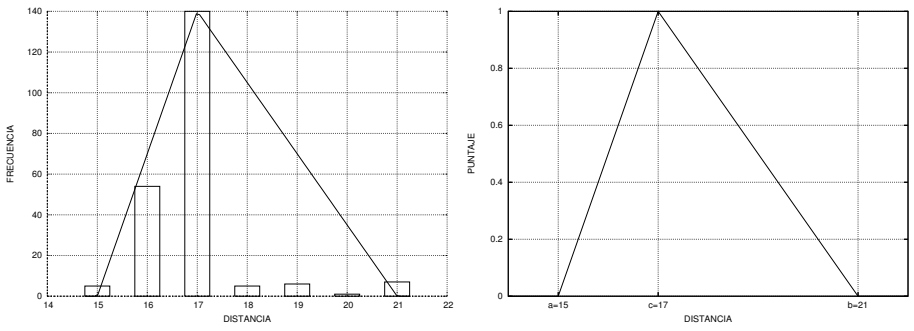
El segundo modelo difuso correspondiente al patrón `TATAAT`, M_α^2, se genera de la forma:

$$M_\alpha^2(x) = \mu_{\mathtt{TATAAT}}(x) = G(\mu_1^2(x_1), \ldots, \mu_6^2(x_6)) \tag{4.2}$$

donde el conjunto difuso discreto asociado al primer nucleótido del motivo se define como $\mu_1^2(x_1) = \mathtt{A}/0,03 + \mathtt{T}/0,82 + \mathtt{G}/0,07 + \mathtt{C}/0,08$, como se muestra en la Tabla 4.2, y los conjuntos difusos correspondientes a las posiciones restantes se calculan de manera análoga.

Se decidió utilizar este método para la construcción de los conjuntos difusos a partir de las distribuciones de probabilidad dado que es una técnica muy simple y económica. Sin embargo, existen otras técnicas para la construcción de conjuntos difusos a partir de este tipo de distribuciones estadísticas [56, 72], algunas de ellas más complejas pero que pueden llegar a obtener mejores resultados. Consideraremos su estudio en trabajos futuros.

El tercer modelo difuso, es decir, la distancia entre los dos patrones previos, M_α^3, se construye como un conjunto difuso, cuya función de pertenencia triangular también se ha aprendido a partir de las distribuciones de datos observadas en los promotores conocidos [48] (ver Figura 4.3(a)).



(a) (b)

Figura 4.3: Representación gráfica de M_α^3

La distancia entre `TTGACA`-box y `TATAAT`-box se mide como una función triangular centrada en 17 donde toma el valor máximo (uno), y a la derecha e izquierda de este punto el valor de la función decrece como puede verse en la ecuación

4.3 y la Figura 4.3.

$$\mu(d) = \begin{cases} -0{,}25 * d + 5{,}25, & si\ 17 \leq d < 21 \\ 0{,}5 * d - 7{,}5, & si\ 15 < d < 17 \\ 0, & en\ otro\ caso \end{cases} \tag{4.3}$$

Con el fin de codificar el algoritmo de reconocimiento de promotores, se utilizaron tres funciones de aptitud, una para cada uno de los objetivos consistentes en la maximización del emparejamiento de la subsecuencia con los tres modelos difusos: `TTGACA`-box (maximizar $f_1(x) = M_\alpha^1(x)$), `TATAAT`-box (maximizar $f_2(x) = M_\alpha^2(y)$) y la distancia entre estos dos patrones (maximizar $f_3(x, y) = M_\alpha^3(dist(x, y))$):

La búsqueda de las soluciones que pertenecen al conjunto Pareto óptimo se realiza utilizando un algoritmo exhaustivo que optimiza simultáneamente los tres objetivos. En trabajos anteriores [29, 78, 77], se han utilizado AGMOs pero, debido a que el espacio de búsqueda del problema es reducido, se pudieron obtener los resultados óptimos con un algoritmo exhaustivo en un tiempo razonable.

4.2.5.1. Ajuste de los modelos

Como se ha visto en la sección anterior, los tres modelos difusos construidos para describir los promotores que queremos identificar en secuencias de ADN se derivan de datos promedio de promotores previamente identificados en diferentes secuencias. Sin embargo, esto no nos asegura el máximo rendimiento en el proceso de predicción, ya que se obtienen una gran cantidad de resultados falsos positivos (FPs). Una posible solución a este problema es el ajuste de las funciones de pertenencia que componen los tres modelos difusos y, en particular, de los umbrales de pertenencia (una subsecuencia será considerada como perteneciente al modelo difuso M_α^i cuando su valor de pertenencia al conjunto difuso correspondiente es mayor o igual que este umbral) de los mismos para incrementar el proceso de predicción, minimizando el número de FPs sin reducir el número de resultados verdaderos positivos (VPs).

De esta manera, se ajustarán las tres funciones de pertenencia de los tres modelos difusos (el motivo `TTGACA`, el motivo `TATAAT` y la distancia entre ellos) y se determinará un umbral de pertenencia para cada uno de ellos para poder diferenciar entre los resultados VPs y FPs.

Para este fin, se ha desarrollado un proceso de ajuste genético [26] para optimizar los parámetros de las funciones de pertenencia y los umbrales de los modelos difusos. Se han tenido en cuenta dos posibles formas de realizarlo: (1) optimizar los modelos y los umbrales por separado; y (2) optimizarlos simultáneamente. La segunda opción ha producido soluciones considerablemente mejores que la primera por lo que ha sido seleccionada como algoritmo de ajuste.

El procedimiento de ajuste, como ya se ha mencionado, intenta disminuir la cantidad de resultados FPs y, al mismo tiempo, no reducir la cantidad de VPs. Estos dos objetivos son claramente contrapuestos y es necesario conseguir una buena solución de compromiso. Para ello, en lugar de optimizar ambos objetivos de forma simultánea, realizaremos un ajuste multiobjetivo para luego seleccionar aquella solución del conjunto Pareto que mejor corresponda con nuestras necesidades.

El algoritmo de ajuste se basa en un algoritmo memético [67] que evoluciona los individuos de la población utilizando un esquema evolutivo generacional clásico basado en un mecanismo de selección por torneo binario [42] (ver Sección 2.4.2.2). La población del algoritmo se compone de un conjunto S de secuencias de ADN de entrada, donde existen varios promotores verdaderos y falsos, previamente identificados, extraídos de [48].

A continuación describimos las distintas componentes del algoritmo evolutivo de ajuste:

Representación de los cromosomas. Los cromosomas están compuestos por tres partes diferentes:

- La primera sección incluye tres números reales, los cuales representan los tres umbrales de pertenencia para cada modelo, definidos como la tupla ($umbral_{\texttt{TTGACA}}$, $umbral_{\texttt{TATAAT}}$, y $umbral_{distancia(\texttt{TTGACA},\texttt{TATAAT})}$).
- La segunda sección del cromosoma codifica los parámetros de las funciones de pertenencia discretas asociadas a los dos motivos, correspondientes a la matriz de probabilidad para cada letra del alfabeto en cada posición de los mismos, como se muestra en la Tabla 4.2.
- Finalmente, la tercera sección corresponde a la función de pertenencia difusa triangular del modelo que especifica la distancia entre los motivos, codificada como una tupla de tres números enteros (a, b y c), como se muestra en la Figura 4.3. Toda distancia menor a a o mayor a b tiene grado de pertenencia 0, mientras que la pertenencia máxima (1) se obtiene en c.

Inicialización. La población inicial consiste en un conjunto de soluciones aleatorias que mantienen las características originales de los modelos de [48] mostrados en la Tabla 4.2. Esto significa que, por ejemplo, en el patrón `TTGACA`, el nucleótido `T` es más frecuente que el nucleótido `C` en la posición 1, por lo que los individuos de la población inicial también conservan esta misma relación. Esto sucede para los dos primeros modelos, mientras que la tercera parte del cromosoma se inicializa siempre con la distribución original de [48], la representada en la Figura 4.3(a).

Relación de no dominancia. Las funciones de aptitud consideradas en el proceso evolutivo de ajuste se muestran en las ecuaciones 4.4 y 4.5, donde $N(x)$ es el

conjunto de individuos en el entorno de x que no son dominados por x. En este contexto, una solución x domina otra solución y como se explica en la Sección 2.5.1. El entorno de un individuo x está compuesto por todas las soluciones situadas a una distancia cercana de x. Esta distancia se calcula utilizando la ecuación 4.8. En esta implementación, todos los cromosomas con distancia ≤ 2 se consideran pertenecientes al mismo entorno.

$$Objetivo_1 = \sum_{i \in S} noesdominado(i, N(i)) \tag{4.4}$$

$$Objetivo_2 = \sum_{i \in S} falsospositivos(i, (N(i)) \tag{4.5}$$

$$noesdominado(x, P) = \begin{cases} 0 & \exists k \in P; k \preceq x \\ 1 & sino \end{cases} \tag{4.6}$$

$$falsospositivos(x, P) = \begin{cases} \frac{|j \in N(i); j \not\preceq i|}{|N(i)|} & noesdominado(x, P) \\ 0 & en\ otro\ caso \end{cases} \tag{4.7}$$

$$dist(x, y) = |x_1 - y_1| + |x_2 - y_2| \tag{4.8}$$

donde x_1/y_1 es la primera posición del primer motivo (`TTGACA`) en el cromosoma x/y y x_2/y_2 es la primera posición del segundo motivo (`TATAAT`) en el cromosoma x/y.

Operadores genéticos. Con respecto a los operadores genéticos, el cruce se aplica a los tres motivos con igual probabilidad. Para el primer y el segundo motivo, se utiliza un cruce en un punto, donde el punto seleccionado se refiere a la posición en el patrón. En este caso particular, las posiciones 2, 3, 4 y 5 de cada motivo pueden ser elegidas e intercambiadas. Por otro lado, para el motivo de la distancia, se selecciona al azar el genotipo de alguno de sus padres para componer el hijo en lugar de utilizar un cruce en un punto, dado que el operador de mutación ya hace esta función, como se explicará más adelante.

Como en el caso del cruce, la mutación se aplica a cada uno de los tres motivos con igual probabilidad. La mutación de los dos primeros motivos se realiza en una posición de 1 a 6 del patrón, sumándole un ruido (valor ϵ pequeño) generado aleatoriamente en el intervalo $\{-10, 10\}$ al valor de probabilidad seleccionado y ajustando las probabilidades restantes de la misma posición para que mantengan una distribución válida. Para el motivo de la distancia, se selecciona uno de los tres números enteros de la tupla con igual probabilidad y se le suma un pequeño ruido aleatorio en el intervalo $\{-3, 3\}$, corroborando que la función de pertenencia triangular siga siendo válida. Es importante notar que este proceso de mutación para el motivo de la distancia produce resultados muy similares a los de un cruce en un punto, por lo cual no se realiza el cruce para este motivo, logrando así reducir los tiempos de ejecución sin la degradación de la calidad de las soluciones finales.

Finalmente, se utiliza un método de búsqueda local solamente en la primera parte del cromosoma, el cual codifica los umbrales para cada modelo difuso. Cualquiera de estos tres valores de umbral puede ser seleccionado con igual probabilidad, luego del cruce y la mutación. El operador de vecino involucra sumarle un ruido aleatorio al número entero seleccionado y el proceso se repite varias veces hasta que la solución resultante reduzca su valor de aptitud. La mejor solución obtenida en este proceso es la que se devuelve.

4.2.5.2. Experimentos

Para la experimentación correspondiente al ajuste de los parámetros de los modelos y el aprendizaje de los umbrales, se han utilizado los datos extraídos de la compilación realizada por Harley & Reynolds [48]. El trabajo realizado por estos autores recoge una serie de secuencias promotoras de ADN correspondientes al organismo *E. coli*, donde se muestran no sólo los nucleótidos que las componen, sino también las ubicaciones exactas de las regiones -10 y -35 dentro de cada una de las secuencias. Como se ha mencionado en la Sección 4.1, puede suceder que un mismo gen contenga más de un promotor asociado, donde cada uno se active en diferentes situaciones. En el artículo de Harley & Reynolds, no sólo se muestra la ubicación del promotor principal, sino que también se indica la presencia de promotores alternativos para algunas de las secuencias. En total, se obtienen 272 secuencias nucleotídicas de distintos tamaños, entre 45 y 64 pb. Cada secuencia tiene asociado al menos un promotor, por lo cual obtendremos una base de datos de 272 secuencias promotoras con sus respectivas posiciones de los modelos descritas. Además, para algunas secuencias se describen uno o más promotores adicionales, sumando un total de 80 promotores alternativos. En conclusión, contamos con una base de datos de 352 secuencias promotoras con su respectiva descripción de los modelos -10 y -35, información que se muestra en la Tabla A.2 del Apéndice de esta memoria.

Se ha ejecutado el algoritmo de ajuste sobre un 80 % de la base seleccionada al azar como conjunto de entrenamiento y el 20 % restante de la base se ha empleado como prueba. Luego de la ejecución del algoritmo de ajuste, se han obtenido varias soluciones posibles, que esencialmente pueden subdividirse en dos grandes grupos:

- Soluciones locales: Involucran a todas aquellas soluciones que tienen un valor de aptitud máximo (de 1) en el $Objetivo_2$. Esto quiere decir que no detectan FPs, pero al mismo tiempo su valor en el $Objetivo_1$ es bajo. Solamente integran este conjunto aquellas soluciones con valores objetivo $(Objetivo_1; Objetivo_2) = (0,487544; 1)$, ya que 0,487544 es el máximo valor en soporte que consigue obtener el algoritmo de ajuste manteniendo el objetivo de FPs en su valor máximo.
- Soluciones globales: Involucran a todas aquellas soluciones que tienen valores altos en ambos objetivos simultáneamente. Esto quiere decir que no

consiguen llevar a cero la cantidad de FPs pero consiguen un valor suficientemente alto, a la vez que abarcan a una gran cantidad de promotores de la base de datos. Solamente integran este conjunto aquellas soluciones con valores objetivo $(Objetivo_1; Objetivo_2) = (0,88968; 0,847753)$, que son los valores máximos obtenidos.

Decimos que son varias las soluciones en cada uno de estos conjuntos ya que, por lo general, pequeñas modificaciones en los valores de umbral y en los modelos no suelen modificar los valores en los objetivos y, por lo tanto, el algoritmo mantiene este conjunto de soluciones.

Para los fines de la detección de promotores, preferimos utilizar soluciones globales que abarquen la mayor cantidad de instancias posibles de la base de datos de entrada y, por lo tanto, hemos escogido una solución global.

A continuación mostraremos cuáles son los resultados obtenidos por el algoritmo de predicción de promotores con y sin el ajuste de los modelos difusos.

La predicción de promotores sin la utilización del ajuste detecta 311 de los 352 promotores de la base de datos, lo cual constituye un 88,35 % del total. Sin embargo, si se hace un estudio utilizando algunas de las métricas estadísticas clásicas, se puede observar que los resultados obtenidos sin el ajuste de los parámetros no son útiles para el usuario final, ya que la cantidad de soluciones obtenidas es muy grande en comparación con los resultados reales. Para ello, calcularemos el número de *verdaderos positivos (VP)*, *falsos positivos (FP)*, *verdaderos negativos (VN)*, *falsos negativos (FN)*, y los ínices de *sensibilidad*, *especificidad* y *precisión* (véase las ecuaciones 4.9, 4.10 y 4.11). Adicionalmente, calcularemos dos métricas estadísticas llamadas *valor de predicción positiva (VPP)* y *rendimiento global (RG)* [14], mostradas en las ecuaciones 4.12 y 4.13. Se intentan maximizar todas estas medidas, con la excepción de los FPs y FNs. En la Tabla 4.3 se muestran los resultados para estas métricas utilizando la predicción de promotores con los motivos difusos originales.

Medida	Valor
VP	0.86
FP	1.00
VN	0.00
FN	0.14

Medida	Valor
Sensibilidad	0.88
Especificidad	0.00
Precisión	0.44
Valor de Predictividad Positiva (VPP)	0.47
Rendimiento Global (RG)	0.45

Tabla 4.3: Resultados para la detección de promotores utilizando los modelos difusos originales

$$sensibilidad = \frac{VP}{VP + FN} \tag{4.9}$$

$$especificidad = \frac{VN}{VN + FP} \tag{4.10}$$

$$precisión = \frac{VP + VN}{VP + VN + FP + FN} \tag{4.11}$$

$$VPP = \frac{VP}{VP + FP} \tag{4.12}$$

$$RG = \frac{precisión + VPP}{2} \tag{4.13}$$

Medida	Valor
VP	0.86
FP	0.27
VN	0.72
FN	0.14

Medida	Valor
Sensibilidad	0.86
Especificidad	0.73
Precisión	0.79
Valor de Predictividad Positiva (VPP)	0.76
Rendimiento Global (RG)	0.78

Tabla 4.4: Resultados utilizando los modelos y umbrales ajustados

Los resultados obtenidos por el algoritmo utilizando los modelos ajustados se muestran en la Tabla 4.4, mientras que aquellos correspondiente a los modelos originales se recogen en la Tabla 4.3. Comparando ambas tablas, se puede observar que, sin utilizar ningún umbral, el número de FPs es muy alto. A pesar que el número de VPs obtenido al ajustar los modelos y los umbrales es menor que sin realizar este procedimiento, la precisión resultante es mucho mejor, como puede observarse de los valores de VPP y RG, que pasan de 0,47 y 0,45 a 0,76 y 0,8, respectivamente.

Las distribuciones de nucleótidos para los motivos `TTGACA` y `TATAAT` ajustados con el AGMO se presentan en la Tabla 4.5. Se puede observar que, con respecto a las distribuciones de nucleótidos originales, existen varios cambios puntuales. Por ejemplo, para el motivo `TTGACA`, la probabilidad de la `G` de la tercera posición aumentó de 68/100 a 88/100. Para el motivo `TATAAT`, por el contrario, la probabilidad de la `A` en la segunda posición disminuyó de 89/100 a 66/100.

Además del reconocimiento de promotores, se necesita poder diferenciar entre promotores de clase I y promotores de clase II y modelar las distancias entre estos promotores y el PhoP-box. A continuación se explica como se realizaron estos dos procesos.

Promotores clase I y clase II. Para diferenciar entre promotores clase I y clase II, utilizamos un analizador sintáctico inteligente que evalúa la calidad de la

	T	T	G	A	C	A
A	0	10	0	**72**	13	**72**
C	12	1	10	15	**79**	17
G	0	6	**88**	5	8	2
T	**88**	**83**	2	9	0	5

(a) Motivo `TTGACA`

	T	A	T	A	A	T
A	2	**66**	25	**46**	**53**	7
C	28	6	22	6	9	10
G	24	5	1	37	26	13
T	**46**	23	**52**	11	12	**70**

(b) Motivo `TATAAT`

Tabla 4.5: Distribución de nucleótidos ajustados para los motivos

región -35 [13, 52], basado en una matriz de pesos.

Distancias entre PhoP-box y promotor. Para construir los modelos de las distancias entre el sitio de binding de PhoP y un promotor, se calculan las distribuciones de distancias para cada tipo de promotor –clase I y clase II–. Posteriormente, se ajustan sus distribuciones en modelos basados en funciones de pertenencia difusa [56] (ver Figura 4.4), generando tres etiquetas difusas –cercano, medio y lejano–.

4.3. Adaptación de CC-EMO para su aplicación al dominio de regulación genética en procariotas

Para poder aplicar el algoritmo CC-EMO a este nuevo dominio, es necesario primero realizar algunas modificaciones sencillas. Uno de los cambios necesarios es la codificación de los cromosomas del AG, para la que se mantendrá la estructura en forma de árbol de la base de datos estructurada, tal como se hace en el dominio geométrico del Capítulo 3. Pero, a diferencia de los árboles del dominio geométrico, ahora contamos con árboles más complejos que no necesariamente son árboles binarios. Así, definimos seis clases diferentes de nodos, uno para cada atributo extraído de la base de datos [100]:

- Tipo 1 (Gen): Esta clase de nodo sólo tiene una posible etiqueta, {*"gen"*}. Este es el nodo raíz de todo cromosoma. A diferencia de las instancias, donde se tenía un nodo raíz que contenía el nombre del gen que representaban (ver Figura 4.2), ahora simplemente se tiene un nodo *dummy* que representa genéricamente a cualquier gen.
- Tipo 2 (PhoP-box): Esta clase de nodo sólo tiene una posible etiqueta: {*"PhoP-box"*}. En un árbol, pueden existir varios nodos de este tipo asociados a un nodo tipo 1 mediante un eje con etiqueta *"tiene"*. Cada nodo

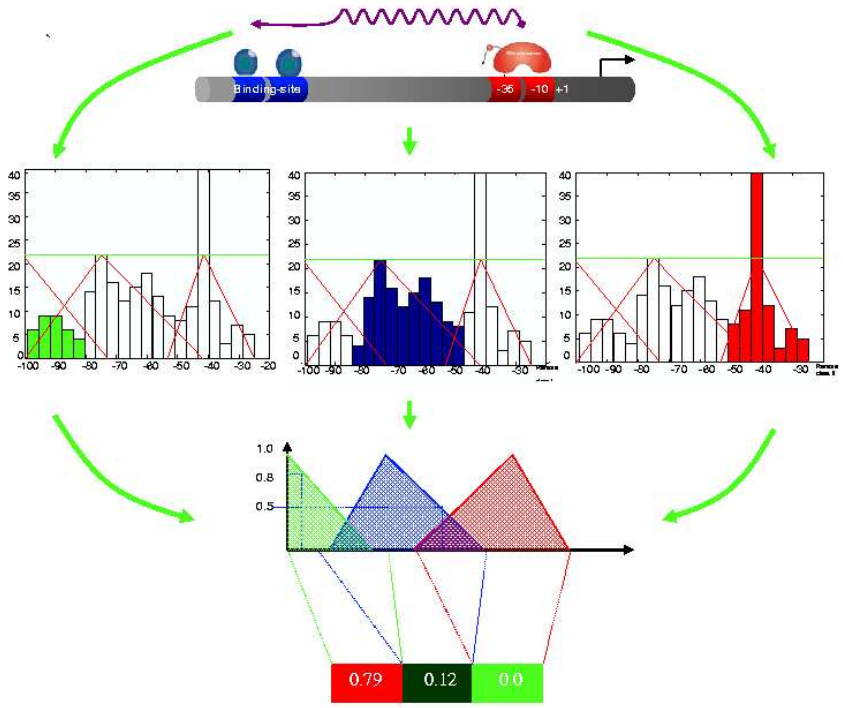


Figura 4.4: Modelado de las distancias entre PhoP-box y promotor.

tipo 2 representa un sitio de *binding* de PhoP cuyas propiedades particulares estarán dadas por los nodos que estén asociados a él.

- Tipo 3 (Motivo): Este tipo de nodo puede tener las etiquetas {0, 1, 2, 3, 4}. Cada número corresponde a un cluster particular del tipo motivo (M0, M1, M2, M3, M4) (ver Sección 4.2.1). El cluster M0 indica que no existe ningún motivo conocido y, por lo tanto, no estará asociado a un nodo de tipo 2, sino al nodo raíz (tipo 1). Los demás clusters estarán asociados a algún nodo tipo 2 por medio de un eje con etiqueta *"motivo"*.
- Tipo 4 (Orientación): Este tipo de nodo puede tener dos etiquetas posibles, {*"directo"*, *"reverso"*}, y estará asociado a un nodo tipo 2 mediante un eje con valor *dirección*. Este nodo representa la orientación del sitio de *binding* de PhoP, el cual puede estar situado en una de las dos hebra del ADN (directo 5' a 3') o en su complementaria (reverso 3' a 5').
- Tipo 5 (Interacción): Esta clase de nodo puede tener las etiquetas {0, 1, 2, 3, 4}. Cada número corresponde a un cluster particular (I0, I1, I2, I3,

I4) (ver Sección 4.2.3) y sólo puede estar asociado a un nodo de tipo 1 por medio de un eje con etiqueta *"interacción"*.

- Tipo 6 (Promotor): Este tipo de nodo es el más complejo, representa la existencia de un promotor. Pueden existir varios promotores por instancia y, por lo tanto, varios nodos de este tipo en el árbol. Existen dos posibles etiquetas, {*"claseI"*,*"claseII"*}, que representan un promotor clase I y un promotor clase II, respectivamente. Los nodos de tipo 6 estarán asociados siempre a un nodo tipo 2 indicando los promotores que trabajan con un sitio de *binding* de PhoP particular. Los ejes posibles son: {*"cercano", "medio", "lejano"*}, indicando a qué distancia se encuentra el promotor en cuestión del sitio de *binding* de PhoP (ver Sección 4.2.5).

Las restricciones de los operadores genéticos vendrán dadas por las relaciones comentadas en la Sección 3.4.

Los objetivos que procura maximizar CC-EMO en forma conjunta son *especificidad* y *sensibilidad*. Para el caso específico del dominio de regulación genética en procariotas, se calculan como se muestra en las ecuaciones 3.1 y 3.2, utilizando las funciones *Cubre* y *Tamaño* definidas en las ecuaciones 3.3 y 3.4, al igual que en el dominio geométrico.

4.4. Evaluación de los clusters

La base de datos utilizada para este dominio está compuesta por 51 genes regulados por *phoP*. Sin embargo, como algunos de ellos presentan diferentes motivos de esta proteína que también se tendrán en cuenta, tendremos un total de 74 instancias en la base de datos, una por cada motivo de PhoP en cada gen regulado por este.

La población inicial de CC-EMO está compuesta por subárboles elegidos aleatoriamente de la base de datos. Consideramos los mismos algoritmos de comparación que en el Capítulo 3, ARPIORI y SUBDUE. Los parámetros de los algoritmos utilizados para este dominio se muestran en la Tabla 4.6. El algoritmo CC-EMO se ha ejecutado 10 veces utilizando distintas semillas y el promedio de estas ejecuciones es el que se considerará de ahora en adelante.

Parámetro	Valor
Tamaño de la población	100
Número de evaluaciones	50000
Probabilidad de cruce	0,6
Probabilidad de mutación	0,2

Tabla 4.6: Parámetros para el dominio de regulación genética en procariotas

En la Figura 4.5 se muestran las soluciones Pareto obtenidas por cada uno de los algoritmos utilizados (APRIORI, SUBDUE y CC-EMO). Es necesario recordar que el conjunto Pareto para CC-EMO está compuesto por la unión de las soluciones para las 10 ejecuciones del algoritmo. Nuevamente, es necesario hacer notar que los frentes de Pareto requieren la misma explicación dada en la Sección 3.5.

Se puede ver en estos gráficos que APRIORI y SUBDUE obtienen un número limitado de soluciones. En el caso particular de APRIORI, éste encuentra alguna de las soluciones óptimas considerando solamente la cantidad de instancias que cubre cada cluster como objetivo a optimizar. Esto es debido a que, a pesar que el algoritmo APRIORI no ha sido diseñado para trabajar con bases de datos estructuradas, la transformación de la base de datos a este sistema ha sido apropiada.

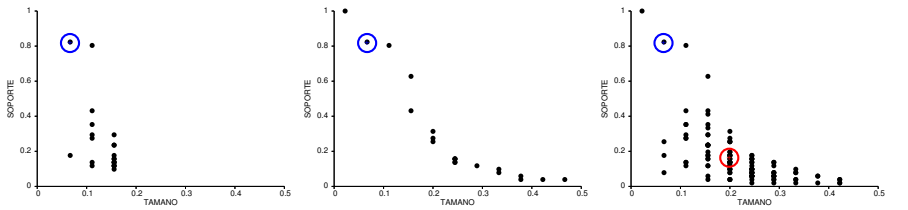
En el caso de SUBDUE, este algoritmo encuentra correctamente las mejores soluciones para los objetivos de sensibilidad y especificidad por separado, lo que quiere decir que descubre los extremos del frente de Pareto, pero no puede llegar a encontrar buenas soluciones de compromiso por las mismas razones comentadas en la Sección 3.5.

Finalmente, el conjunto Pareto para CC-EMO contiene la mayoría de las soluciones encontradas por los sistemas APRIORI y SUBDUE, y además una gran diversidad de soluciones de compromiso.

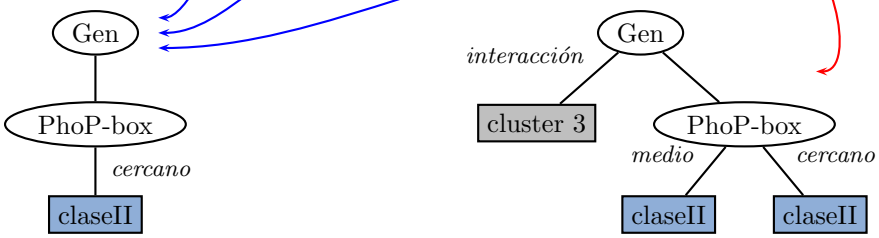
Para verificar estas últimas afirmaciones, consideraremos las mismas métricas empleadas en la Sección 2.5.3.

Comenzaremos estudiando las métricas para conjuntos Pareto individuales. En la Tabla 4.7 se muestran todos los resultados para las métricas $\mathcal{M}_2^*$, $\mathcal{M}_3^*$ y $\mathcal{S}$. Los boxplots asociados a cada métrica para CC-EMO se ilustran en la Figura 4.6. La métrica $\mathcal{M}_2^*$ muestra la diversidad de soluciones encontradas en un conjunto Pareto y, como se observa la Tabla 4.7(b), CC-EMO es claramente el algoritmo con mejor diversidad de soluciones. Este hecho también puede verse en los frentes de Pareto de la Figura 4.5. La métrica $\mathcal{M}_3^*$ muestra que CC-EMO cubre mejor los extremos del frente de Pareto que APRIORI, pero SUBDUE logra detectar una solución aún más en el extremo que CC-EMO no logra encontrar debido a la limitación de tamaño de la población, como puede verse en la Tabla 4.7(a) y en la Figura 4.5. Finalmente, para la métrica $\mathcal{S}$, recogida en la Tabla 4.7(a) y en la Figura 4.6, CC-EMO obtiene el valor más alto y, por lo tanto, cubre mejor el espacio del frente del Pareto.

Ahora estudiaremos las métricas de comparación de conjuntos Pareto. Con respecto a la métrica $\mathcal{C}$, como puede verse en la Tabla 4.7(c) y en la Figura 4.7(a), tanto CC-EMO como APRIORI tienen soluciones que se dominan las unas a las otras, pero CC-EMO obtiene mejores resultados globales. Por otro lado, CC-EMO domina a SUBDUE con un índice mayor que SUBDUE a CC-EMO. El algoritmo SUBDUE y el APRIORI no se dominan entre sí, ya que el valor de la métrica en la comparativa es cero.

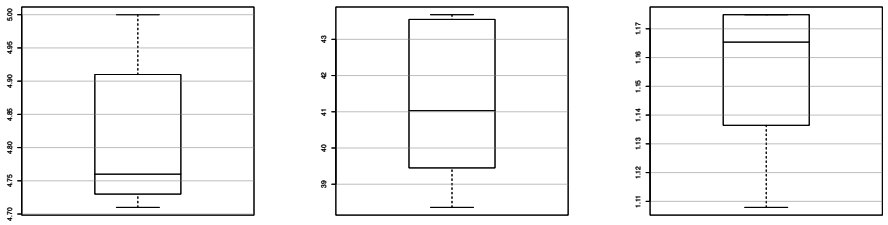


(a) APRIORI (b) SUBDUE (c) CC-EMO



(d) Modelo en común (e) Ejemplo destacado

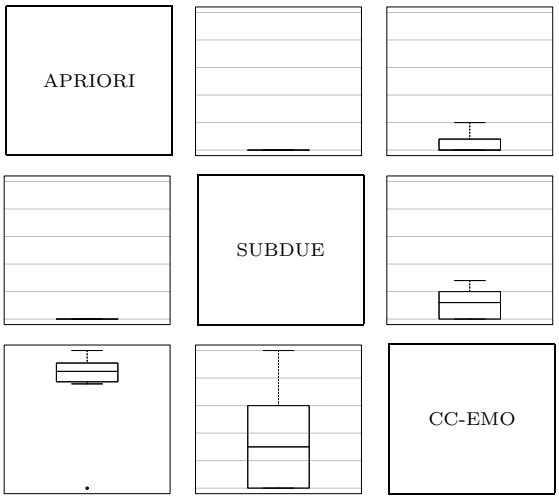
Figura 4.5: Frentes de Pareto para el dominio de regulación genética en procariotas.



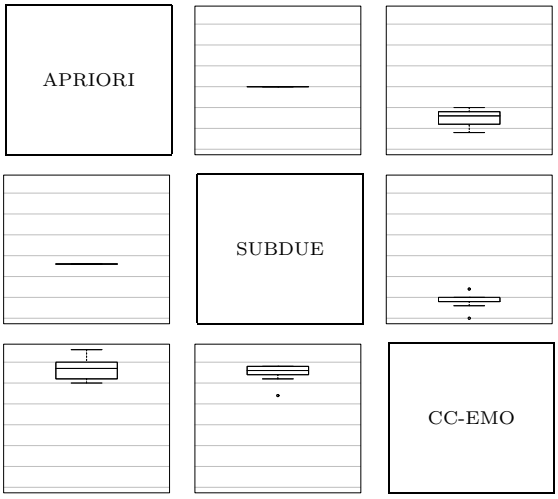
(a) $\mathcal{S}$ (b) $\mathcal{M}_2^*$ (c) $\mathcal{M}_3^*$

Figura 4.6: Boxplots de las métricas $\mathcal{S}$, $\mathcal{M}_2^*$ y $\mathcal{M}_3^*$ para CC-EMO en el dominio de regulación genética en procariotas.

Por último, la métrica $\mathcal{ND}$ muestra en la Tabla 4.7(d) y en la Figura 4.7(b) que CC-EMO descubre más soluciones no dominadas que los otros dos algo-



(a) Boxplots de la métrica $\mathcal{C}$



(b) Boxplots de la métrica $\mathcal{ND}$

Figura 4.7: Boxplots de las métricas $\mathcal{C}$ y $\mathcal{ND}$ para CC-EMO en el dominio de regulación genética en procariotas.

	$\mathcal{S}(X)$	$\mathcal{M}_3^*$
APRIORI	2,5336	0,9024
SUBDUE	3,9095	1,1854
CC-EMO prom. (*desv_est*)	4,81534 (*0, 11009*)	1,15476 (*0, 02411*)

(a) Métricas $\mathcal{S}$ y $\mathcal{M}_3^*$

	$\mathcal{M}_2^*$	$\|X\|$
APRIORI	21,3333	22
SUBDUE	17,3684	20
CC-EMO prom. (*desv_est*)	41,28952 (*1, 95847*)	42,8 (*1, 8738*)

(b) Métrica $\mathcal{M}_2^*$

$\mathcal{C}(X', X'')$	APRIORI	SUBDUE	CC-EMO prom. (*desv_est*)
APRIORI	-	0,00000	0,01203 (*0, 02063*)
SUBDUE	0,00000	-	0,02981 (*0, 024*)
CC-EMO prom. (*desv_est*)	0,02727 (*0, 03178*)	0,09 (*0, 08433*)	-

(c) Métrica $\mathcal{C}$

$\mathcal{ND}(X', X'')$	APRIORI	SUBDUE	CC-EMO prom. (*desv_est*)
APRIORI	-	20	12,4 (*1, 89737*)
SUBDUE	18	-	9 (*1, 76383*)
CC-EMO prom. (*desv_est*)	33,3 (*2, 75076*)	32,3 (*2, 11082*)	-

(d) Métrica $\mathcal{ND}$

Tabla 4.7: Resultado de las métricas para CC-EMO en el dominio de regulación genética en procariotas.

ritmos. APRIORI y SUBDUE encuentran muy pocas soluciones que CC-EMO no llega a obtener (12,4 y 9 en promedio de la Tabla 4.7(d)). La diferencia de valores mostrada por la métrica $\mathcal{ND}$ para CC-EMO en contraste con APRIORI y SUBDUE (33,3 y 32,3 contra solamente 12,4 y 9 de la Tabla 4.7(d)) muestra cuán bien se comporta nuestra metodología en este dominio.

En cuanto a los tiempos de ejecución, para CC-EMO son aproximadamente de unos once minutos, mientras que para APRIORI y SUBDUE son sólo de un par de minutos.

4.5. Compactación de la base de datos

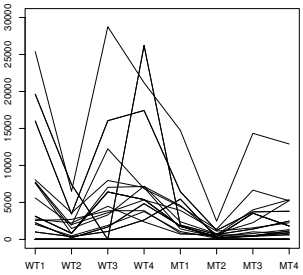
Aparte de la información sobre las propiedades de orientación, motivo, promotor e interacción, contamos con información sobre la expresión de cada uno de los genes regulados por *phoP* de la base de datos (ver Sección 4.2.4). Como ya se ha mencionado dicha sección, se ha agrupado esta información en 3 clusters: **E1**, **E2** y **E3** (ver Figura 4.8(a),(b),(c) respectivamente). Por lo tanto, tenemos los mismos genes agrupados de dos maneras diferentes, utilizando información independiente una de la otra: por un lado, en función de la expresión (ClustersA), y por otro, considerando las cuatro propiedades restantes, haciendo uso de nuestra metodología CC-EMO (ClustersB). Teniendo esta información de control, podremos depurar los clusters encontrados tanto por CC-EMO como por los dos algoritmos de comparación, quedándonos solamente con aquellos que resulten más relevantes de acuerdo a los datos externos.

En las Figuras 4.9(a) y 4.9(b) se ilustran las intersecciones de APRIORI y SUBDUE con respecto a los perfiles de expresión E1, E2 y E3. Además, la Figura 4.9(c) muestra las intersecciones con respecto a CC-EMO. Los gráficos representan cada intersección con un círculo, cuyo tamaño crece a medida que aumenta el número de instancias en la intersección entre los clusters, mientras que el color muestra el p-value para la intersección (ver la Sección 3.6). No se representan las asociaciones con valores de p-value mayores a un umbral de $0,1$, ya que no serían suficientemente significativas.

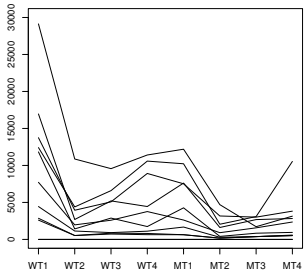
Como puede observarse, APRIORI y SUBDUE encuentran un menor número de clusters relevantes en comparación con CC-EMO. Estudiaremos ahora algunos ejemplos de estas intersecciones encontradas por CC-EMO para comprender mejor los resultados obtenidos.

La compactación de los clusters se realiza como se describió en la Sección 3.6. En la Figura 4.10, se ejemplifica para este dominio el proceso de compactación para el cluster 2 de ClustersA. En este ejemplo, se elige un cluster de cada rama generada en base a las descripciones de los Clusters, remarcándose en color el cluster elegido para cada rama. En el caso de la rama de color verde, tanto el cluster 39 como el 94 de ClustersB involucran a los mismos genes y, por lo tanto, se prefiere el primer cluster al segundo. Pero, dado que el cluster 105 cubre a más genes y, además, es más general que el cluster 39 de ClustersB, se selecciona finalmente el cluster 105 de ClustersB como representante de la rama.

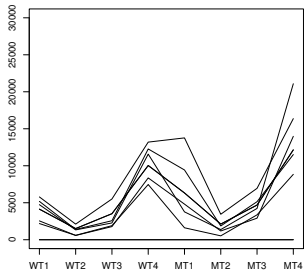
Como primer ejemplo, podemos analizar la descripción del cluster 78 de ClustersB encontrado por CC-EMO que intersecta con el cluster 1 de ClustersA en la Figura 4.11. Esta descripción incluye, entre otros, a los genes {*phoP*, *mgtA*, *ybcU*, *yhiW*} de *E. coli* y a *slyB* de *Salmonella*, los cuales presentan el perfil canónico de los genes regulados por *phoP*, es decir, tienen los mismos patrones de expresión, submotivo de PhoP-box, y comparten los mismos sitios de binding de la ARN polimerasa y de otros factores de transcripción. Este perfil incluye no sólo a los promotores prototípicos *phoP* y *mgtA*, sino también a *yhiW* del



(a) E1



(b) E2

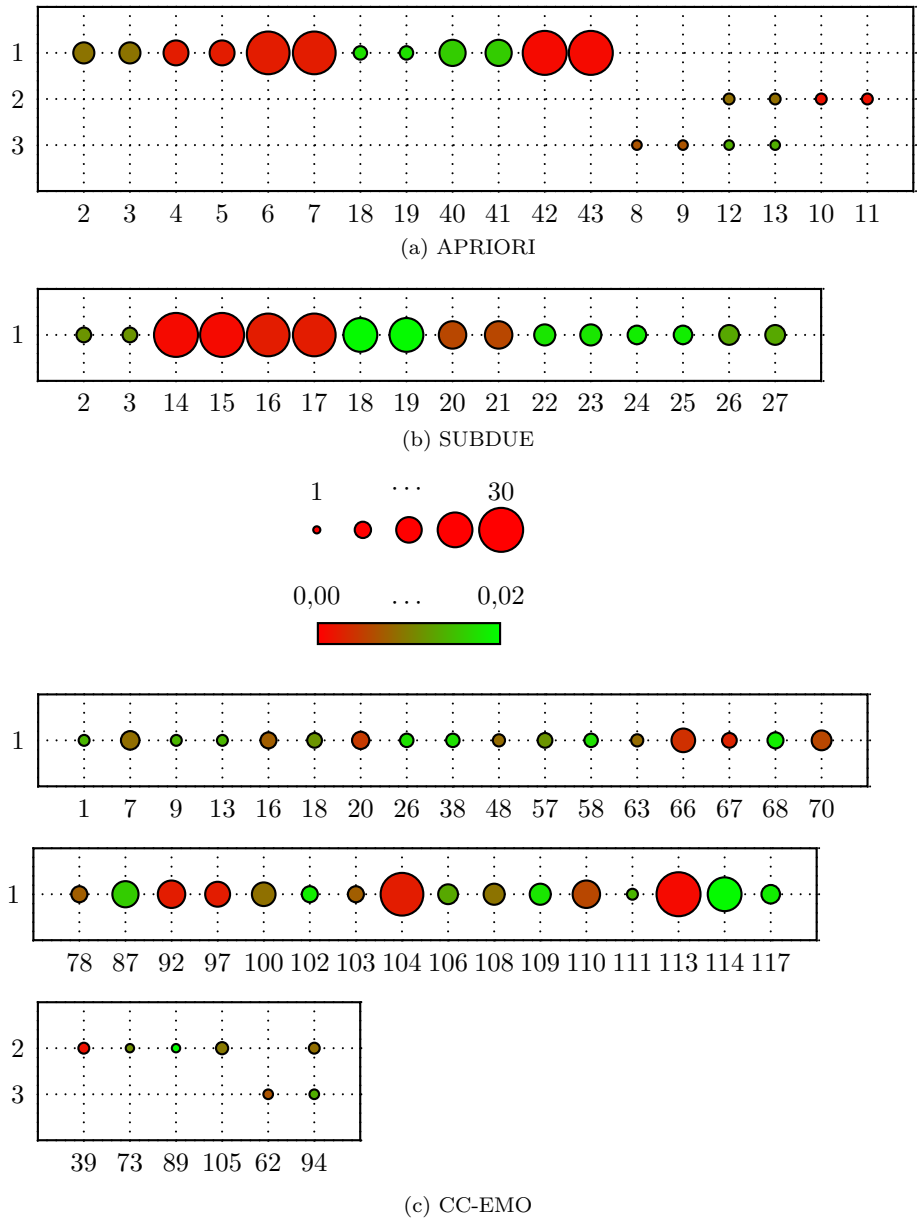


(c) E3

Figura 4.8: Clusters de expresión para el dominio de regulación genética en procariotas.

cual no se tenía información de que se encontrara bajo el control de *phoP*. Este cluster no es detectado por APRIORI ni por SUBDUE.

Otro ejemplo es el cluster 26 de ClustersB encontrado por CC-EMO que intersecta también con el cluster 1 de ClustersA. La descripción de este cluster se muestra en la Figura 4.12 e incluye, entre otros, a los genes *ompT* de *E. coli* y a {*pipD*, *ugtL*, *ybjX*} de *Salmonella*, los cuales comparten sitios de la ARN polimerasa, patrón de expresión y otros factores de transcripción y orientaciones similares, claramente opuestos a las regiones promotoras prototípicas de *phoP* y *mgtA*. Del grupo de genes descrito por el concepto mostrado en la Figura 4.12,



(a) APRIORI

(b) SUBDUE

(c) CC-EMO

Figura 4.9: Intersección de los ClustersA y ClustersB para APRIORI, SUBDUE y CC-EMO.

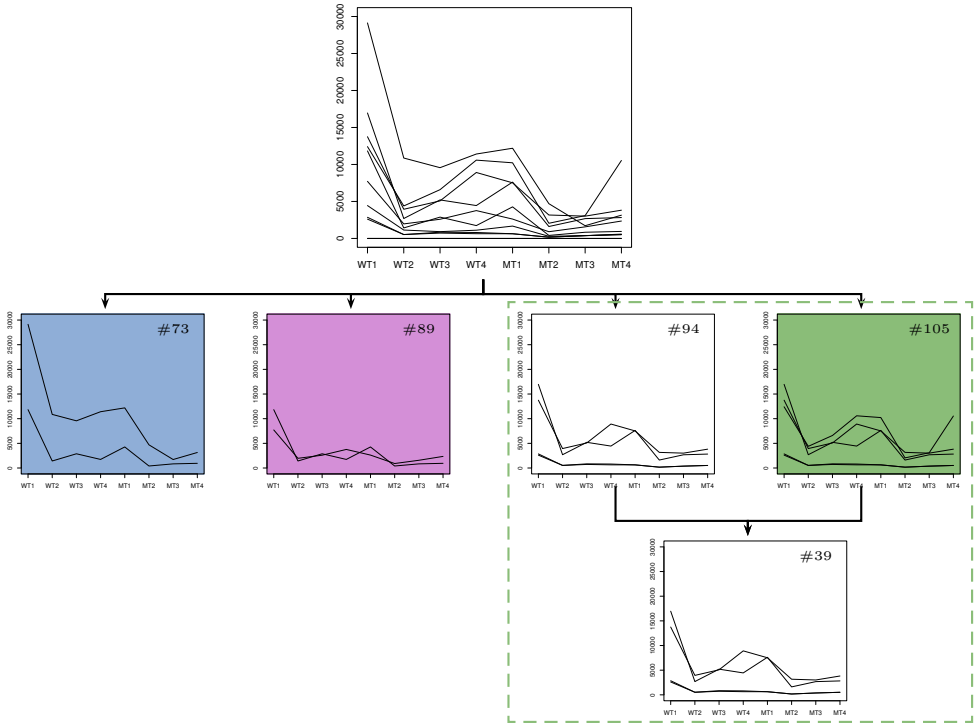


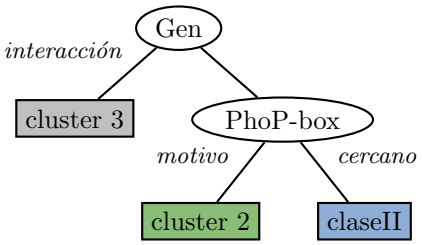
Figura 4.10: Expresión del cluster 2 de ClustersA y su relación con los ClustersB con los cuales interseca. Los ClustersB representados corresponden a la intersección entre éstos y el cluster 2 de ClustersA. La jerarquía viene dada por la relación entre los términos de GO de cada cluster conceptual. Los gráficos remarcados corresponden a aquellos clusters que contienen los mismos genes.

sólo uno no presenta los mismos patrones de expresión que el resto del conjunto. Este cluster no es detectado por APRIORI pero sí por SUBDUE.

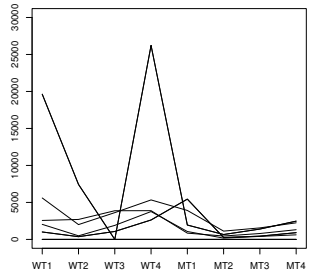
Por último, el cluster 73 de ClustersB encontrado por CC-EMO, que también intersecta con el cluster 2 de ClustersA, es otro ejemplo destacable. La descripción de este cluster se muestra en la Figura 4.13 e incluye, entre otros, a los genes $\{hdeA, hdeD\}$ de *E. coli*, los cuales componen un conjunto de genes estructurales bien diferenciados. Este cluster no es detectado ni por APRIORI ni por SUBDUE.

4.6. Predicción

Una vez concluido el proceso de clustering y la compactación de los clusters generados, se pueden clasificar nuevas instancias en los diferentes conjuntos ob-

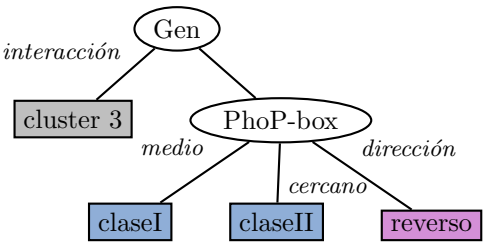


(a) Descripción del cluster

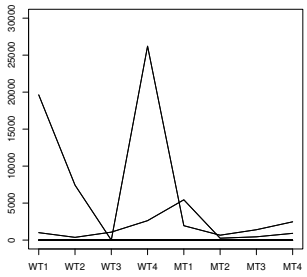


(b) Expresión del cluster intersecando el ClusterA #1

Figura 4.11: Descripción del cluster 78 de ClustersB



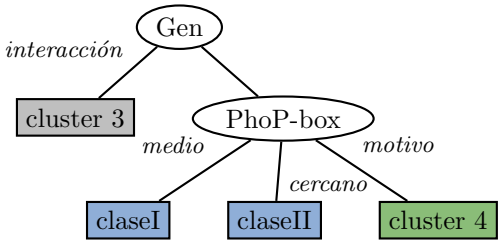
(a) Descripción del cluster



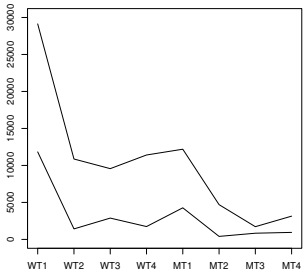
(b) Expresión del cluster intersecando el ClusterA #1

Figura 4.12: Descripción del cluster 26 de ClustersB

tenidos. Dado que los clusters no son disjuntos, puede que una instancia dada pertenezca a más de un conjunto. Tal como se explica en la Sección 3.7, nuestra metodología utiliza un clasificador difuso basado en el k-prototipo más cercano. El cálculo del grado de pertenencia de la observación x_q en el conjunto I de subestructuras previamente identificadas se realiza mediante la ecuación 3.7.



(a) Descripción del cluster



(b) Expresión del cluster intersecando el ClusterA #2

Figura 4.13: Descripción del cluster 73 de ClustersB

Nombre	Genoma	Motivo	Dirección	Interacción	Cercano		Medio		Lejano	
					claseI	claseII	claseI	claseII	claseI	claseII
pagK	*Salmonella*	4	reverso	3	0	0	0,5	0,50505	0	0
pagK	*Salmonella*	4	reverso	3	0	0,55	0	0	0	0

Tabla 4.8: Descripción de la región regulatoria de *pagK*

Por ejemplo, podríamos querer clasificar la región regulatoria de *pagK* de *Salmonella* descrita en la Tabla 4.8. De todos los clusters generados por CC-EMO y depurados en el proceso de validación, *pagK* tiene un grado de pertenencia mayor a cero con los clusters 7, 87, 103, 104, 110 y 113 de ClustersB. Estos valores son 0,15556; 0,11111; 0,2; 0,11111; 0,18888 y 0,06667, respectivamente. Por lo tanto clasificaremos *pagK* como perteneciente al ClusterB 103:

$$\begin{aligned} knn(x_q = pagK, V_7, V_{87}, V_{103}, V_{104}, V_{110}, V_{113}) = V_{103} \ / \\ \mu_{i,q} = max\{\mu_{7,q}, \mu_{87,q}, \mu_{103,q}, \mu_{104,q}, \mu_{110,q}, \mu_{113,q}\} = \\ = max\{0,15556; 0,11111; 0,2; 0,11111; 0,18888; 0,06667\} = 0{,}2 \end{aligned}$$

4.7. Comentarios finales

En este capítulo se ha aplicado la metodología propuesta a un problema de regulación genética en dos organismos procariotas, *E. coli* y *Salmonella*. Para ello, se ha recogido información sobre diversos genes que están regulados directa

o indirectamente por el gen *phoP* en ambos genomas.

Durante la etapa de construcción de la base de datos, se generaron diversos modelos basados en lógica difusa, aprendizaje automático y algoritmos evolutivos multiobjetivo. Gracias a estos modelos, se ha podido detectar nuevas instancias de estos modelos dentro de los genomas utilizados, que no habían sido obtenidas utilizando métodos clásicos. En el caso específico del reconocimiento de promotores, el cual constituye un problema de optimización multimodal y multiobjetivo, se ha conseguido una mayor precisión e interpretabilidad al utilizar sistemas genéticos difusos. Asimismo, el método de reconocimiento propuesto se ha validado mediante la predicción de promotores del organismo *E. coli*, mediante un algoritmo que combina las ventajas de la representación de características basadas en conjuntos difusos y las capacidades de búsqueda de los algoritmos evolutivos multiobjetivo. Se ha conseguido una mejora en la precisión de los modelos gracias al ajuste de los mismos mediante un algoritmo de ajuste multiobjetivo basado en algoritmos meméticos. Esta mejora ha reducido, en gran medida, la cantidad de resultados falsos positivos obtenidos anteriormente.

El proceso de compactación basado en información externa, en este caso, datos sobre los niveles de expresión de cada gen, provee al usuario de una explicación de cada conjunto de genes que se expresan en forma similar. Esta explicación se realiza en base a las características de la región reguladora de estos genes. Diferentes conjuntos de control podrán entonces ser descritos en base a estas características sin necesidad de repetir el proceso de generación de subestructuras.

Por último, es importante destacar la capacidad de la metodología de predecir el comportamiento de nuevos genes en base a los conceptos aprendidos y depurados, convirtiéndola en una herramienta de caja blanca precisa y, a la vez, interpretable.

La aplicación de la metodología propuesta al problema tratado en este capítulo, generó buenos resultados, los cuales han sido validados experimentalmente [100]. Asimismo, los resultados obtenidos han logrado superar a los conseguidos por los otros dos enfoques –APRIORI y SUBDUE– utilizados en la comparativa.

Capítulo 5

Aplicación a organismos eucariotas

La metodología propuesta en el Capítulo 3 se aplicará en este capítulo al problema de regulación genética en organismos eucariotas. Para ello, se utilizará una base de datos que contiene información sobre diversos genes implicados en un estudio de la respuesta inflamatoria de seres humanos al aplicarles una endotoxina en forma intravenosa, en comparación con un grupo de control al cual se le inyecta un placebo [79]. Para modelar esta base de datos, se utilizará un repositorio estructurado llamado *Gene Ontology* [8] que almacena una ontología de genes basada en sus procesos biológicos, funciones moleculares y componentes celulares.

Este capítulo se dividirá en varias secciones siguiendo el esquema utilizado en el Capítulo 3. Se comenzará con una introducción al problema desde el punto de vista biológico, para luego mostrar en detalle los pasos realizados para aplicar la metodología al problema tratado. Finalmente, se realizará una evaluación de los resultados obtenidos, tanto desde el punto de vista computacional como desde el experimental.

5.1. Introducción al problema biológico

La metodología introducida en el Capítulo 3 se aplicará en este capítulo a un problema sobre organismos eucariotas. Este problema consiste en el estudio de la respuesta inflamatoria de seres humanos al aplicarles una endotoxina en forma intravenosa, en comparación con un grupo de control al cual se le inyecta un placebo [79]. En este estudio, se han extraído un conjunto de genes de la sangre de los ocho pacientes tratados, cuatro con la endotoxina (pacientes 1-4)

y cuatro con el placebo (pacientes 5-8). Los datos se han extraído en diferentes instantes de tiempo, a 0, 2, 4, 6, 9 y 24 horas, y se han procesado utilizando GeneChips® y HG-U133A v2.0 de Affymetrix Inc®. Del estudio de los perfiles de expresión de la sangre de los pacientes, podemos extraer y representar gráficamente la información sobre la expresión en el tiempo de cada uno de los genes. En la Figura 5.1 se muestra un ejemplo de esta clase de gráficos de expresión en el tiempo. Es necesario hacer notar que sólo los cuatro pacientes tratados con la endotoxina intravenosa están representados en la figura, y sus series temporales han sido organizadas consecutivamente resultando en gráficos con una distribución regular. Estas series temporales se han agrupado en 24 clusters utilizando el algoritmo de clustering clásico *k-medias* [36], resultando en los 24 gráficos de la Figura 5.2.

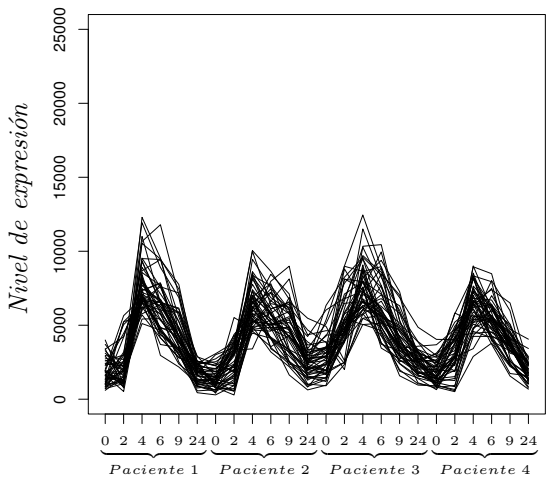


Figura 5.1: Gráfica de la serie temporal correspondiente a la expresión de un conjunto de genes en el tiempo. El eje X corresponde a la hora en que se toma la medición, mientras que el eje Y corresponde al nivel de expresión detectado. Notar que solamente se representa la información de los cuatro pacientes a los cuales se les ha inyectado la endotoxina en forma sucesiva, obteniendo una gráfica con una distribución regular.

Además de la información sobre la expresión en el tiempo de los distintos genes utilizados en este estudio, se cuenta con información sobre varias características que presentan estos mismos genes. Estas características pueden clasificarse en tres grandes grupos: función molecular, participación en algún proceso

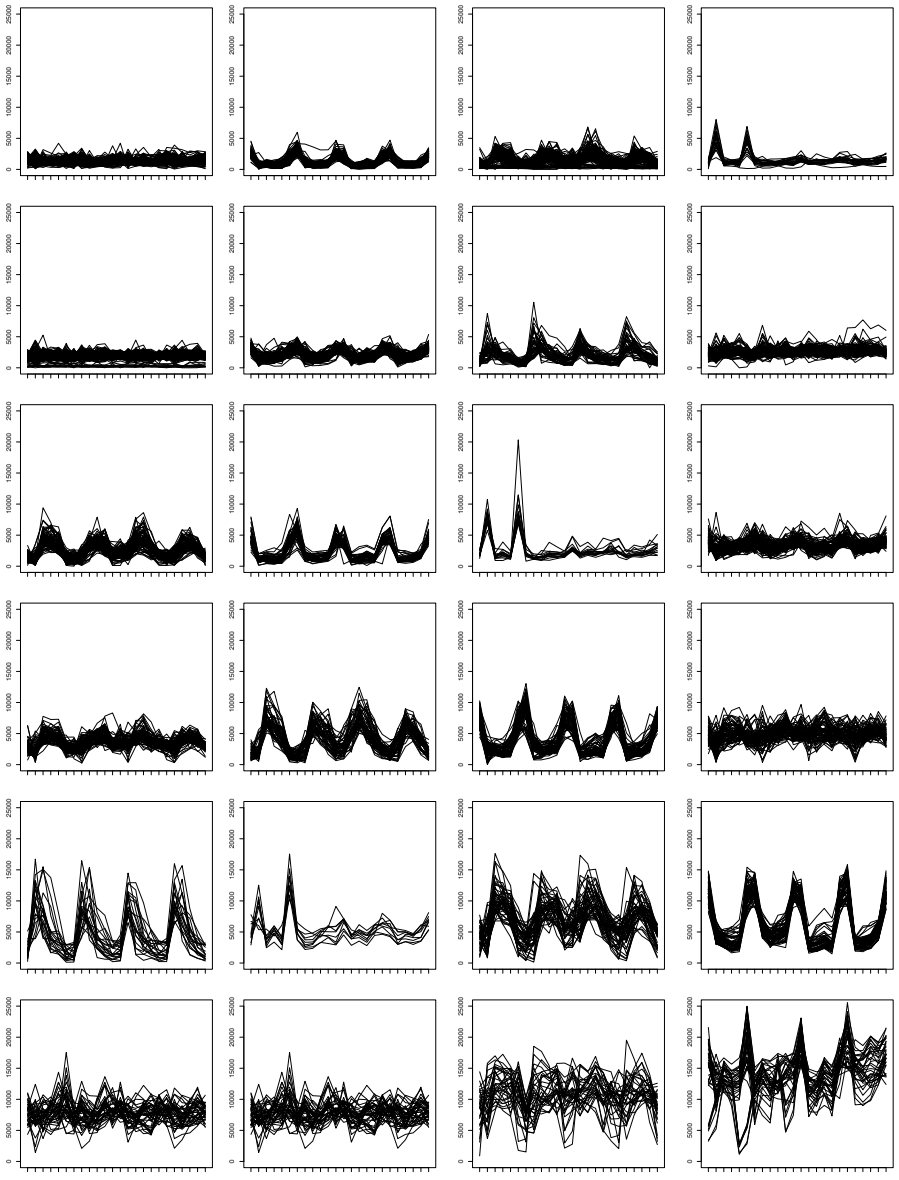


Figura 5.2: Clusters de los datos de expresión. Este agrupamiento ha sido realizado utilizando el algoritmo k-means con $k = 24$.

biológico y lugar en el organismo donde se encuentra presente el gen. Esta información se extrae de la base de datos HG-U133A v2.0 de Affymetrix Inc® y las anotaciones de cada gen corresponden a la ontología de genes definida como parte del proyecto "Gene Ontology" [8]. El objetivo de este capítulo es mostrar la aplicación de la metodología propuesta al estudio biológico de respuesta inflamatoria en seres humanos con el fin de descubrir conceptos de clusters basados únicamente en los datos de las propiedades de los genes, sin hacer uso de la información de expresión. De esta manera, nos permitirá comprender mejor el por qué cierto conjunto de genes se comportan de forma similar, es decir, se expresan manteniendo el mismo esquema. A este problema o dominio lo llamaremos *ontología de genes* ó simplemente *GO*.

Como ya se ha mencionado, una de las bases de datos con la cual se trabajará en esta aplicación es parte del proyecto *"Gene Ontology"*, GO. El objetivo de este proyecto es producir un vocabulario controlado que pueda aplicarse a todos los organismos, aún cuando el conocimiento sobre los roles de los genes y las proteínas en las células siga acumulándose y cambiando continuamente.

En la actualidad, los biólogos gastan mucho tiempo y esfuerzo en buscar toda la información disponible acerca de una pequeña área de investigación sobre la cual estén trabajando. Esta búsqueda es obstaculizada por la gran variabilidad en la terminología que se esté utilizando en un momento de tiempo determinado, es decir, dependiendo de ésta, los resultados de la búsqueda serán diferentes. Esto prohibe una búsqueda efectiva, tanto por parte de los seres humanos, como por parte de los ordenadores. Por ejemplo, si se están buscando nuevos objetivos para antibióticos, podríamos necesitar todos los genes que estén involucrados en la síntesis de proteínas de bacterias y que tengan además secuencias o estructuras significativamente diferentes de aquellas en humanos. Pero, si una base de datos describe estas moléculas como involucradas en la "traducción", mientras que otra utiliza la frase "síntesis de proteínas", será muy dificultoso para el usuario –y aún mas difícil para un ordenador– encontrar términos con una funcionalidad equivalente.

El uso de los términos de GO por varias bases de datos facilita la uniformidad de las consultas. Los vocabularios están estructurados de tal manera que se puede consultar a diferentes niveles. Por ejemplo, se puede usar GO para buscar todos los genes en el genoma del ratón que estén involucrados en la transducción de señales[1], o se puede buscar, en forma más específica, todos los receptores tirosina-kinasa[2]. Esta estructura también permite realizar anotaciones sobre propiedades de genes en diferentes niveles, dependiendo del conocimiento que se

[1]Conjunto de procesos o etapas que ocurren de forman concatenada por el que una célula convierte una determinada señal o estímulo exterior, en otra señal o respuesta específica.

[2]Una tirosina-kinasa es una enzima que puede transferir un grupo fosfato a una tirosina en una proteína. Estas enzimas son una subclase de una clase de proteínas kinasa más amplia. La fosforilación es una función importante en la transducción de señales para regular una actividad enzimática.

tenga de los mismos.

Es importante notar que GO no es una base de datos de secuencias de genes, como puede ser GenBank [16] o EMBL [55], y tampoco un catálogo de genes. Por el contrario, GO describe cómo se comportan los genes en un contexto celular. Por otro lado, GO no intenta describir cada aspecto de la biología. Por ejemplo, la estructura de dominio, la estructura 3D, la evolución y la expresión genética de una proteína, no están contenidas en la base de datos de GO.

Se están desarrollando tres vocabularios (ontologías) estructurados y controlados para describir los genes en términos de sus procesos biológicos asociados, componentes celulares y funciones moleculares, independientemente de las especies biológicas. Un gen puede tener una o más funciones moleculares, ser utilizado en uno o más procesos biológicos y estar asociado a uno o más componentes celulares. Por ejemplo, el gen *citocromo c* puede describirse por los términos "matriz mitocondrial" y "membrana mitocondrial interna".

5.1.1. Las diferentes ontologías

Como ya se ha mencionado, las tres organizaciones principales de GO son función molecular, proceso biológico y componente celular. A continuación analizaremos en detalle cada una de ellas:

- *Función Molecular:* esta ontología abarca aquellas tareas desarrolladas por genes individuales.

  La función molecular describe actividades, tales como actividades catalíticas o de *binding*, a nivel molecular. Los términos de funciones moleculares de GO representan actividades y no entidades, como moléculas o complejos, que realizan acciones y no especifican dónde, ni cuándo, ni en qué contexto se lleva a cabo la acción. Las funciones moleculares generalmente corresponden a actividades que pueden ser realizadas por genes individuales, pero algunas actividades son realizadas por complejos ensamblados de genes. Ejemplos de términos de funcionalidad general son las actividades "catalíticas", actividades de "transporte", o "*binding*". Ejemplos de términos funcionales más específicos son la actividad de adenilato ciclasa o el *binding* del receptor *toll*.

  Es fácil confundir un producto de un gen con su función molecular debido a que, en muchas ocasiones, se describen con exactamente las mismas palabras. Por ejemplo, "alcohol dehidrogenasa" puede describir lo que se puede colocar en un tubo *Eppendorf* (el producto del gen) o la función de éste. Sin embargo, existe una diferencia formal: un producto de un gen simple puede tener varias funciones moleculares, y muchos genes pueden compartir la misma función molecular. Por ejemplo, existen muchos genes con la función "alcohol dehidrogenasa", y sólo algunos de éstos, pero no todos, pueden ser codificados por genes con el nombre *alcohol dehidrogenasa*.

Un producto de un gen particular puede tener tanto la función "alcohol dehidrogenasa" como "acetaldehido dismutasa", y tal vez también otras funciones. Es importante comprender que, cuando se usen términos como "actividad de alcohol dehidrogenasa" en GO, esto corresponde a la función y no a la entidad. Es por esta razón que muchas funciones moleculares de GO tienen agregadas la palabra "actividad".

- *Proceso biológico:* esta ontología comprende objetivos biológicos, tales como la mitosis o el metabolismo de purinas, que son realizados por funciones moleculares.

  Un proceso biológico es llevado a cabo por uno o más ensamblajes ordenados de funciones moleculares. Ejemplos de términos de procesos biológicos de amplio espectro son "crecimiento y mantenimiento" o "transducción de señales". Ejemplos de términos más específicos son "metabolismo de pirimidinas" o "transporte de alfa-glucosidasa". Puede ser difícil distinguir entre proceso biológico y función molecular, pero la regla general es que los procesos deben tener más de un paso distintivo. Esto no debe confundirse con un camino o *pathway*.

- *Componente celular:* esta ontología cubre estructuras subcelulares, localizaciones y complejos macromoleculares.

  Un componente celular es simplemente lo que su nombre indica, un componente de la célula, a condición de que sea parte de algún objeto mayor, el cual puede ser una estructura anatómica (retículo endoplasmático rugoso o núcleo) o un grupo de genes (ribosoma, proteasoma o dímero proteico). Ejemplos de términos de esta ontología son "núcleo" y "telómero".

5.1.2. Estructura de las ontologías

Los términos GO están organizados en estructuras llamadas grafos dirigidos acíclicos (GDAs) [3], que son diferentes a una jerarquía en la que un "hijo", es decir, un término más especializado, puede tener varios "padres", es decir, términos menos especializados. Por ejemplo, el proceso biológico "biosíntesis de hexosa" tiene dos padres, "metabolismo de hexosa" y "biosíntesis de monosacáridos". Esto es debido a que es un subtipo de metabolismo y a que una hexosa es un tipo de monosacárido. Cualquier gen involucrado en la biosíntesis de hexosa anotado con este término, también es automáticamente anotado tanto a "metabolismo de hexosa" como con "biosíntesis de monosacáridos". Esto es debido a que cada término de GO obedece la *regla del camino verdadero*[3]: si el término hijo describe un producto de un gen, también todos sus términos padre deben aplicar a ese producto. Se dice entonces que un nodo de GO, es decir un nodo del GDA, se refiere al término indicado en particular y a todos sus padres.

[3]La regla del camino verdadero define que "el camino desde un término hijo hasta su/s padre/s en el nivel más alto debe ser verdadero".

Las tres ontologías están unidas entre sí por el término de GO 0003673, llamado "Gene_Ontology". Todo término de GO tiene como ancestro a éste término, sea cual sea su tipo (componente celular, función molecular o proceso biológico). Este término no representa ningún concepto biológico real y ha sido declarado como obsoleto y reemplazado por un nodo artificial, que no pertenece a ninguna de las ontologías, llamado "all" (todo), el cual constituye el término más general posible.

Un término hijo puede tener una de dos posibles relaciones con su/s padre/s: "es_un" o "es_parte_de" (ver Figura 5.3). Un mismo término puede tener diferentes relaciones con diferentes padres.

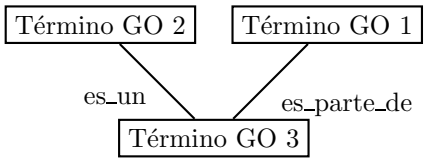
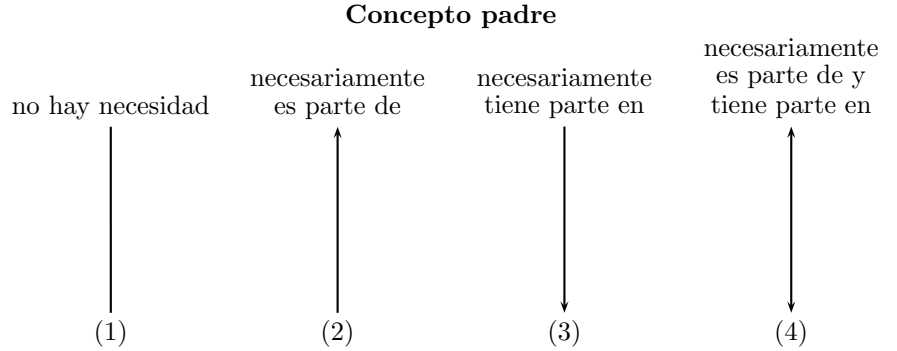


Figura 5.3: Relaciones entre términos de GO

La relación "es_un" significa que un término es una subclase de sus padres. Por ejemplo, "ciclo de una célula mitótica" es_un "ciclo de una célula". No debe confundirse con una *instancia*, la cual es un ejemplo específico. La relación "es_un" es transitiva, lo cual significa que si un término GO A es una subclase del término GO B, y el término GO B es una subclase del término GO C, entonces el término GO A es también una subclase del término GO C.

La relación "es_parte_de" es más compleja. Existen cuatro niveles básicos de restricciones para una relación "es_parte_de":



- El primer tipo (1) no tiene restricciones. Esto es, no se puede realizar otra inferencia de la relación entre padre e hijo aparte de que el padre pueda

o no tener al hijo como parte de él, y que el hijo pueda o no ser parte del padre.

- El segundo tipo (2) significa que, cuando el hijo existe, obligatoriamente es parte del padre. Por ejemplo, "bifurcación de una réplica" es parte de "cromosoma". Por lo tanto, cuando una "bifurcación de una réplica" ocurre, entonces es parte de "cromosoma"; pero "cromosoma" no necesariamente tiene una "bifurcación de una réplica".

- El tercer tipo (3) es exactamente la inversa del tipo (2): cuando un padre existe, tiene al hijo como parte, pero no necesariamente el hijo es parte del padre. Por ejemplo, "núcleo" siempre es parte de "cromosoma", pero "cromosoma" no es necesariamente parte de "núcleo".

- El cuarto tipo (4) es la combinación de los casos (2) y (3). Un ejemplo de éste caso es "membrana nuclear" que es parte de "núcleo". Por ello, "núcleo" siempre tiene parte en "membrana nuclear" y "membrana nuclear" siempre es parte de "núcleo".

La relación "es_parte_de" utilizada en GO es generalmente del segundo tipo (2), "necesariamente_es_parte_de". Notar que los tipos (1) y (3) no se utilizan en GO, ya que violarían la regla de camino verdadero. Al igual que "es_un", "es_parte_de" es transitiva, de tal manera que si un término GO A es parte del término GO B, y el término GO B es parte del término GO C, entonces el término GO A es también es parte del término GO C.

5.1.3. Formato de las anotaciones utilizando GO

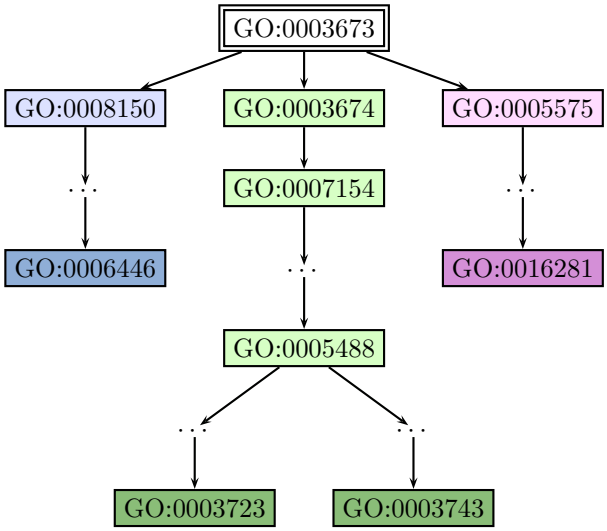
Cada gen se anota con uno o varios códigos de GO de una o varias de sus ontologías. Adicionalmente, cada uno de estos términos tiene asociado un código de evidencia, que determina de qué manera se ha obtenido la relación entre el término y el gen. Ejemplos de evidencia pueden ser IC (inferido por un curador) y TAS (declaración detectable de un autor, lo cual puede ser a través de un experimento publicado en un artículo de revista o en un libro de texto o diccionario). En la Figura 5.4 se puede ver una porción de una entrada de la base de datos de GenBank de un producto de un gen donde se pueden apreciar las anotaciones de GO asociadas. Por ejemplo, esta proteína tiene la función molecular "actividad oxidoreductasa", que corresponde al código de GO 0016491, y la evidencia IEA (inferido de una anotación electrónica, lo que quiere decir que no ha sido verificada por un curador).

```
LOCUS       NP_001082               751 aa            linear   PRI 26-OCT-2004
DEFINITION  amiloride binding protein 1 precursor [Homo sapiens].
ACCESSION   NP_001082
VERSION     NP_001082.1  GI:4501851
DBSOURCE    REFSEQ: accession NM_001091.1
KEYWORDS    .
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Primates; Catarrhini;
            Hominidae; Homo.
REFERENCE   1  (residues 1 to 751)
  AUTHORS   Olive,M., Unzeta,M., Moreno,D. and Ferrer,I.
  TITLE     Overexpression of semicarbazide-sensitive amine oxidase in human
            myopathies
  JOURNAL   Muscle Nerve 29 (2), 261-266 (2004)
   PUBMED   14755492
  REMARK    GeneRIF: Semicarbazide-sensitive amine oxidase is a source of
            oxidative stress in diseased human skeletal muscle; it contributes
            to oxidative stress-induced damage in various inflammatory and
            other myopathies.
...
FEATURES             Location/Qualifiers
     source          1..751
                     /organism="Homo sapiens"
                     /db_xref="taxon:9606"
                     /chromosome="7"
                     /map="7q34-q36"
     Protein         1..751
                     /product="amiloride binding protein 1 precursor"
                     /EC_number="1.4.3.6"
                     /note="diamine oxidase; Amiloride-binding protein-1"
     sig_peptide     1..19
     mat_peptide     20..751
                     /product="amiloride-binding protein 1"
     CDS             1..751
                     /gene="APB1"
                     /coded_by="NM_001091.1:72..2327"
                     /note="go_component: peroxisome [goid 0005777] [evidence
                     NAS] [pmid 1356107];
                     go_function: drug binding [goid 0008144] [evidence NR];
                     go_function: heparin binding [goid 0008201] [evidence
                     IEA];
                     go_function: copper ion binding [goid 0005507] [evidence
                     IEA];
                     go_function: oxidoreductase activity [goid 0016491]
                     [evidence IEA];
                     go_function: amine oxidase activity [goid 0008131]
                     [evidence TAS] [pmid 8144586];
                     go_process: metabolism [goid 0008152] [evidence NR]"
                     /db_xref="GeneID:26"
                     /db_xref="MIM:104610"
ORIGIN
        1 mpalgwavaa ilmlqtamae pspgtlprka gvfsdlsnqe lkavhsflws kkelrlqpss
       61 tttmaknvf liemllpkky hvlrfldkge rhpvrearav iffgdqehpn vtefavgplp
      ...
      721 ngpnyvqrwi pedrdcsmpp pfsyngtyrp v
//
```

Figura 5.4: Ejemplo de anotación de GO

Nombre	Proceso biológico	Función molecular	Componente celular
206621_s_at	GO:0006446	GO:0003723 GO:0003743	GO:0016281

(a) Instancia



(b) Representación estructurada

Figura 5.5: Ejemplo de la representación de una instancia. Los nodos de color celeste corresponden a la ontología de proceso biológico, los verdes a la ontología de función molecular y los rosas a la ontología de componente celular. Los puntos suspensivos reemplazan a uno o varios niveles. Por ejemplo, el nodo GO:0003723 está en realidad mucho más cercano a la raíz del árbol que el nodo GO:0003743.

5.2. Construcción de la base de datos estructurada

La base de datos en bruto con que se trabajará en esta aplicación de la metodología propuesta consiste en el grupo de genes extraídos de la sangre de los pacientes del estudio, indicados por sus códigos en la base de datos de Affymetrix, como pueden ser: {`200008_s_at`, `201011_at`, `201352_at`, etc.}. La construcción de la nueva base de datos estructurada requiere buscar las anotaciones de GO correspondientes a cada una de las instancias de la base de datos en bruto. El modelado de los objetos es directo, ya que contamos con las ontologías de GO,

las cuales son bases de datos estructuradas. Cada instancia de la base de datos tiene uno o varios códigos de GO correspondientes a una o varias de las tres ontologías posibles. En la Figura 5.5(a) se muestra un ejemplo de una instancia de la base de datos. El modelado propuesto para este dominio consiste en incorporar el conocimiento de las jerarquías de GO en cada instancia de tal manera que el algoritmo de clustering conceptual pueda hacer uso de esta información. Para ello, a cada código de GO al cual pertenezca una instancia se le incorporará la información de sus ancestros desde su/s padre/s hasta la raíz de GO. En la Figura 5.5(b) se muestra la representación estructurada de la instancia de la Figura 5.5(a) utilizando las jerarquías de GO. Esta representación estructurada de cada instancia conformará la nueva base de datos de objetos estructurados que servirá de entrada al algoritmo de clustering conceptual. Como se puede ver en la figura, nuevamente contamos con una representación en forma de árbol.

5.3. Adaptación de CC-EMO para su aplicación al dominio de ontología de genes

Para poder aplicar el algoritmo CC-EMO a este nuevo dominio, es necesario primero realizar algunas modificaciones sencillas. Uno de los cambios necesarios es la codificación de los cromosomas del algoritmo evolutivo. Una opción posible consiste en mantener la misma estructura, en forma de árbol, de la base de datos, tal como se hacía en el dominio geométrico del Capítulo 3. Si bien esto es viable y sencillo de realizar, es mucho más eficiente computacionalmente codificar los cromosomas como un vector de códigos de GO y mantener las jerarquías almacenadas de forma externa. De esta manera se ahorra mucho espacio en la representación, ya que no es necesario repetir parte de la jerarquía de GO en cada cromosoma. Cuando sea necesario utilizar esta información, se accederá a ella como se explicará más adelante.

Entonces, para la representación de un cromosoma se utilizará un vector de nodos, que podrán ser de tres tipos: nodos tipo 1 (correspondientes a los posibles nodos de la jerarquía de GO de procesos biológicos), nodos tipo 2 (correspondientes a función molecular) y nodos tipo 3 (correspondientes a componente celular). Cada uno de estos nodos tiene asociada una etiqueta, la cual restringirá la aplicación de los operadores genéticos, tal como se ha explicado en la Sección 3.4. Al utilizar una representación lineal, en contraposición con la representación jerárquica del capítulo anterior, trabajaremos con un AG en lugar de un algoritmo de PG.

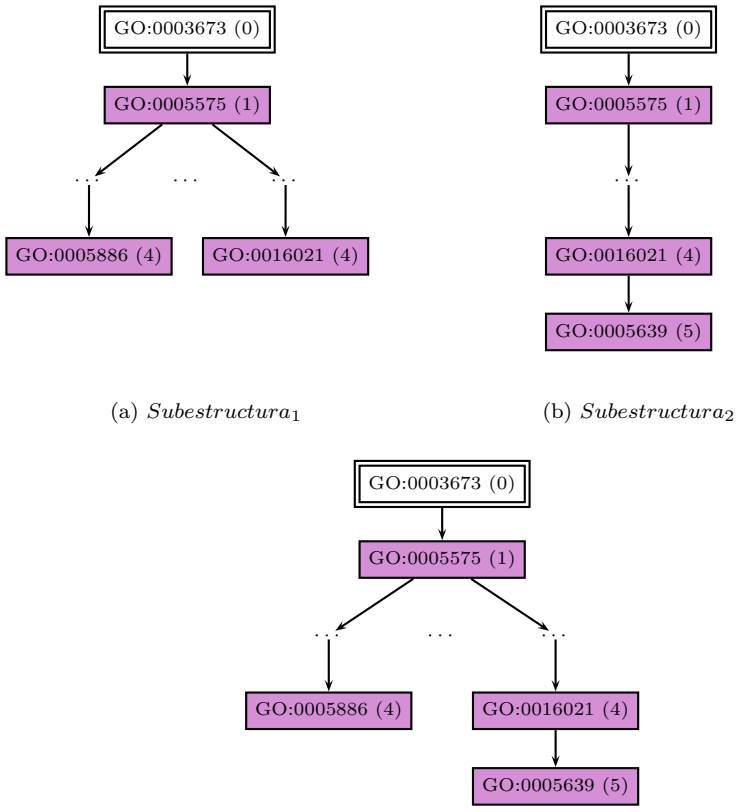
En las primeras pruebas de CC-EMO utilizando este nuevo dominio, las implementaciones clásicas de los objetivos de especificidad y sensibilidad como tamaño y soporte, respectivamente, devolvieron resultados subóptimos. Las subestructuras obtenidas contenían descripciones de GO que se encontraban muy altas en la jerarquía y, por lo tanto, eran muy generales y poco útiles para resol-

ver el problema. Debido a esto, y luego de un estudio detallado de ambos objetivos, se llegó a la conclusión de que el objetivo de especificidad no funcionaba correctamente al no tener en cuenta la información de los niveles de la jerarquía de GO. La razón por la cual el tamaño de la subestructura no redundaba en buenos resultados era que, en este dominio, la especificidad de la subestructura no es linealmente dependiente a su tamaño. Como ejemplo de esta situación ilustramos dos subestructuras en la Figura 5.6: la $Subestructura_1$ de la Figura 5.6(a) tiene tamaño 2 (debido a que se tienen dos nodos en la representación del cromosoma correspondientes a las hojas del árbol), y la $Subestructura_2$ de la Figura 5.6(b) con tamaño 1. Ambas subestructuras cubren la instancia de la Figura 5.6(c), pero la $Subestructura_1$ es más general que la $Subestructura_2$, aún cuando la $Subestructura_1$ es más grande que la $Subestructura_2$ en tamaño. Se dice que una subestructura *cubre* una instancia si la subestructura es un árbol prefijo de la instancia:

> Un árbol T' es un *árbol prefijo* de T, si T puede obtenerse incorporando cero o más árboles a algunos nodos de T'. Notar que cualquier árbol T es prefijo de sí mismo.

En la Figura 5.6, se puede ver que la subestructura $Subestructura_1$ es un árbol prefijo de la instancia, y que la $Subestructura_2$ es igual a la instancia. Por lo tanto, también es un árbol prefijo. Esta definición es muy similar a la mencionada en la Sección 3.4, pero exige que la relación de inclusión del árbol correspondiente a la subestructura en el árbol instancia comience desde la raíz.

Por lo tanto, el objetivo de especificidad debe reformularse basándose no sólo en el tamaño de la subestructura, sino también en los niveles de los términos de GO dentro de las tres ontologías. Para ello, comparamos los niveles cada nodo de la subestructura en la jerarquía de GO, contra su nodo correspondiente en cada instancia de la base de datos cubierta por la subestructura. Veamos un ejemplo: supongamos que contamos con la subestructura de la Figura 5.6(a) y una de las instancias cubiertas por esta subestructura es la de la Figura 5.6(c). Para calcular el nivel de especificidad de la subestructura, debemos comparar cada nodo que la compone con respecto a su correspondiente nodo asociado en la instancia. La subestructura tiene dos nodos hoja {GO:0005886,GO:0016021} al igual que la instancia {GO:0005886,GO:0005639}. El nodo GO:0005886 cubre al nodo GO:0005886 de la instancia, por lo que estos nodos están asociados en el cálculo de especificidad. Lo mismo sucede con el nodo GO:0016021 de la subestructura y el nodo GO:0005639 de la instancia. Recordar que la información de nivel de cada nodo y su termino GO asociado reflejan la especificidad de cada término. Cuanto mayor sea el valor del nivel, más específica es la información del nodo. En el ejemplo, los nodos GO:0005886 y GO:0016021 son de nivel 4, mientras que el nodo GO:0005639 es de nivel 5, según la jerarquía de GO. Para poder reflejar esta información en el nuevo objetivo de *especificidad*, realizaremos



(a) $Subestructura_1$ (b) $Subestructura_2$

(c) Instancia

Figura 5.6: Ejemplo de subestructuras que cubren una misma instancia. Los números entre paréntesis corresponden al nivel del nodo en la jerarquía de GO.

una suma ponderada calculada por la ecuación 5.1:

$$Especificidad(x) = \frac{\sum_{i=1}^{\#instancias\ cubiertas} \left(1 - \frac{\sum_{j=1}^{\#nodos_{instancia_i}} \frac{nivel\ relativo(x)_{j_i}}{nivel\ absoluto_{j_i}}}{\#nodos_{instancia_i}}\right)}{\#instancias\ cubiertas} \tag{5.1}$$

donde $nivel\ relativo(x)_{j_i}$ es la diferencia de niveles entre el nodo j, de la instancia i, y su nodo asociado en la subestructura (x); y $nivel\ absoluto_{j_i}$ es el nivel absoluto del nodo j de la instancia i en la jerarquía de GO. Podemos inferir de la ecuación 5.1 que, cuanto más grande sea el nivel de separación de los nodos

de la subestructura a los nodos de la instancia, mayor será el valor de la fórmula $\frac{nivel\ relativo(x)_{j_i}}{nivel\ absoluto_{j_i}}$. Por lo tanto, la penalización por una baja especificidad será mayor, produciendo un valor menor al restárselo a uno. Recordar que los objetivos del algoritmo CC-EMO se maximizan. La fórmula de especificidad se normaliza para producir un valor de objetivo entre cero y uno.

Aplicando la ecuación 5.2 a nuestro ejemplo, tenemos el siguiente cálculo de la especificidad:

$$\left(1 - \frac{\sum_{j=1}^{\#nodos_{instancia_i}} \frac{nivel\ relativo(x)_{j_i}}{nivel\ absoluto_{j_i}}}{\#nodos_{instancia_i}}\right) = \left(1 - \frac{\sum_{j=1}^{1} \frac{1}{5}}{1}\right) = 0{,}8 \qquad (5.2)$$

Notar que, dado que hay dos posibles nodos en la subestructura que cubren al mismo nodo en la instancia, se realiza el cálculo para ambas alternativas y luego se promedia. El cálculo de especificidad se realiza para todas las instancias cubiertas por la subestructura. Por ejemplo, el árbol de la Figura 5.6(b) tiene un valor de especificidad de 0,0870057 calculado sobre 154 instancias de la base cubiertas por la subestructura que representa.

5.4. Evaluación de los clusters generados

La base de datos utilizada para este dominio, extraída del experimento introducido en la Sección 5.1, está compuesta por 1770 instancias de genes y sus términos GO asociados.

La población inicial de CC-EMO está formada por un 50 % de subárboles elegidos aleatoriamente de la base de datos y un 50 % de instancias completamente aleatorias generadas a partir de la jerarquía de GO. Este procedimiento particular fue necesario debido a que no todos los términos de GO aparecían en la base de datos de entrada, haciendo más dificil que el algoritmo encontrara buenas soluciones contando solamente con los términos de la base de datos original.

Los parámetros de los algoritmos utilizados para este dominio se muestran en la Tabla 5.1. El algoritmo CC-EMO se ha ejecutado 10 veces utilizando distintas semillas y el promedio de estas ejecuciones es el que se considerará a partir de ahora.

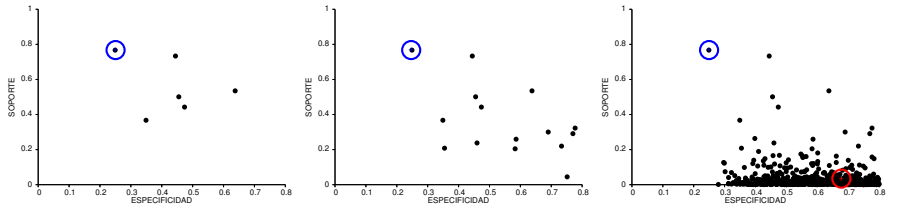
En la Figura 5.7 se muestran las soluciones Pareto obtenidas por cada uno de los algoritmos utilizados (APRIORI, SUBDUE y CC-EMO). Es necesario recordar que el conjunto Pareto para CC-EMO está compuesto de la unión de las soluciones para las 10 ejecuciones del algoritmo y que los frentes de Pareto requieren la misma explicación dada en la Sección 3.5.

Se puede ver en estos gráficos que APRIORI y SUBDUE obtienen un número limitado de soluciones. En el caso particular de APRIORI, dicho algoritmo encuentra alguna de las soluciones óptimas considerando solamente el objetivo de

Parámetro	Valor
Tamaño de la población	200
Número de evaluaciones	20000
Probabilidad de cruce	0,6
Probabilidad de mutación	0,2

Tabla 5.1: Parámetros para el dominio de *ontología de genes*

sensibilidad. Esto es debido a que, a pesar que el algoritmo APRIORI no ha sido diseñando para trabajar con bases de datos estructuradas, la transformación de la base de datos a este sistema ha sido la apropiada. En el caso de SUBDUE, éste encuentra correctamente las mejores soluciones para los objetivos de especificidad y sensibilidad por separado, es decir, descubre los extremos del frente de Pareto, pero no puede llegar a encontrar buenas soluciones de compromiso por las mismas razones comentadas en la Sección 3.5. Finalmente, el conjunto Pareto para CC-EMO contiene la mayoría de las soluciones encontradas por los sistemas APRIORI y SUBDUE, y además una gran diversidad de soluciones de compromiso.



(a) APRIORI (b) SUBDUE (c) CC-EMO

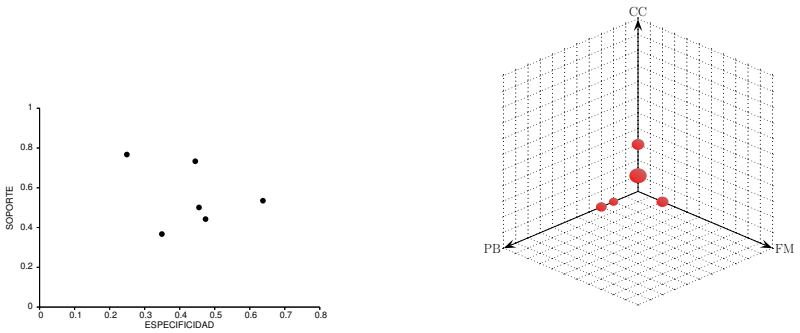
Proceso biológico	*Función molecular*	*Componente celular*
		GO:0005575 cellular_component (level: 1)

(d) Subestructura en común

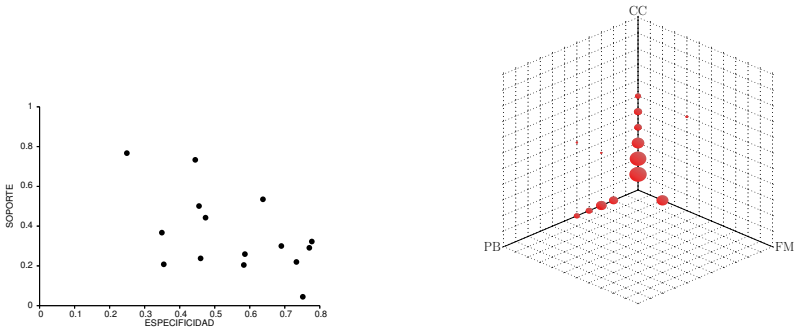
Proceso biológico	*Función molecular*	*Componente celular*
GO:0008151 cell growth and/or maintenance (level: 4)	GO:0005488 binding (level: 2)	GO:0016020 membrane (level: 3)
		GO:0000267 cell fraction (level: 3)

(e) Ejemplo destacado

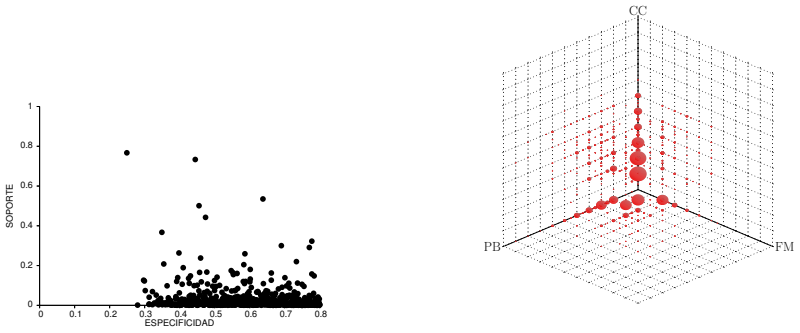
Figura 5.7: Frentes de Pareto para el dominio de *ontología de genes*.



(a) APRIORI



(b) SUBDUE



(c) CC-EMO

Figura 5.8: Espacio de objetivos vs. espacio de variables de cada conjunto Pareto.

	$\mathcal{S}(X)$	$\mathcal{M}_3^*$
APRIORI	1,8517	1,1853
SUBDUE	2,4952	1,1978
CC-EMO prom. (*desv_est*)	4,2983 (*0, 7194*)	1,2334 (*0, 0047*)

(a) Métricas $\mathcal{S}$ y $\mathcal{M}_3^*$

	$\mathcal{M}_2^*$	$\|X\|$
APRIORI	6	7
SUBDUE	17,56	19
CC-EMO prom. (*desv_est*)	186,8680 (*12, 2065*)	188 (*12, 19*)

(b) Métrica $\mathcal{M}_2^*$

$\mathcal{C}(X', X'')$	APRIORI	SUBDUE	CC-EMO prom. (*desv_est*)
APRIORI	-	0,00	0,00000 (*0, 0000*)
SUBDUE	0,00	-	0,00050 (*0, 0016*)
CC-EMO prom. (*desv_est*)	0,00 (*0, 00*)	0,08421 (*0, 04438*)	-

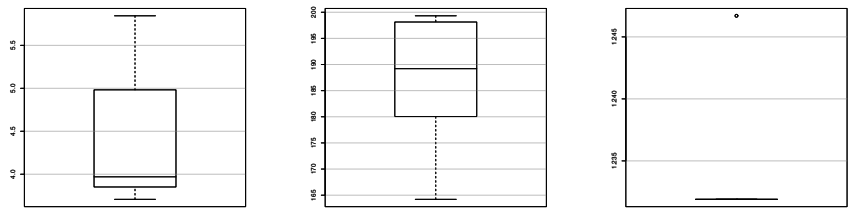
(c) Métrica $\mathcal{C}$

$\mathcal{ND}(X', X'')$	APRIORI	SUBDUE	CC-EMO prom. (*desv_est*)
APRIORI	-	1	1,20 (*0, 42*)
SUBDUE	13	-	1,60 (*1, 17*)
CC-EMO prom. (*desv_est*)	181,80 (*11, 99*)	171,80 (*11, 62*)	-

(d) Métrica $\mathcal{ND}$

Tabla 5.2: Resultado de las métricas para los distintos algoritmos en el dominio de ontología de genes.

Adicionalmente, en la Figura 5.8 se muestran el espacio de variables y el de los objetivos de los conjuntos Pareto de cada algoritmo. El eje PB corresponde a la ontología de proceso biológico y crece a medida que aumenta el nivel de los códigos de GO de la descripción asociada en su respectiva jerarquía. Análogamente, FM corresponde a función molecular y CC a componente celular. Si, por ejemplo, una descripción tiene asociada dos códigos de GO, uno de nivel 6 de PB y uno de nivel 2 de CC, entonces se graficará una esfera en la posición (PB=6,FM=0,CC=2) con un radio del tamaño correspondiente a la cantidad de instancias que cubre de la base de datos. Es interesante ver que APRIORI sólo descubre aquellas soluciones que se encuentran sobre los ejes del gráfico, mientras que el SUBDUE logra extraer algunas que tengan más de un código



(a) $\mathcal{S}$ (b) $\mathcal{M}_2^*$ (c) $\mathcal{M}_3^*$

Figura 5.9: Boxplots de las métricas $\mathcal{S}$, $\mathcal{M}_2^*$ y $\mathcal{M}_3^*$ para CC-EMO en el dominio de ontología de genes.

en distintas jerarquías de GO. Por último, CC-EMO es el enfoque que logra obtener no sólo las soluciones sobre los ejes, sino también una amplia variedad de soluciones con diferentes niveles y que involucran a más de una de las jerarquías de GO.

Para verificar estos últimas afirmaciones, emplearemos un conjunto de las métricas presentadas en la Sección 2.5.3. Comenzaremos estudiando las métricas para conjuntos de Pareto individuales. En la Tabla 5.2 se muestran todos los resultados para $\mathcal{M}_2^*$, $\mathcal{M}_3^*$ y $\mathcal{S}$. Por otro lado, los boxplots asociados a cada una de ellas para CC-EMO ilustran en la Figura 5.9. La métrica $\mathcal{M}_2^*$ muestra la diversidad de soluciones encontradas en un conjunto Pareto, y como se observa en la Tabla 5.2(b), CC-EMO es claramente el algoritmo con mejor diversidad de soluciones. Este hecho también puede verse en los frentes de Pareto de la Figura 5.7. La métrica $\mathcal{M}_3^*$ muestra como CC-EMO está cubriendo correctamente los extremos del frente de Pareto, seguido por SUBDUE y finalmente APRIORI, como puede verse en la Tabla 5.2(a) y en la Figura 5.7. Finalmente, para la métrica $\mathcal{S}$, recogida en la Tabla 5.2(a) y en la Figura 5.9, CC-EMO obtiene el valor más alto, cubriendo mejor el espacio del frente del Pareto.

Ahora estudiaremos las métricas de comparación. Con respecto a la métrica $\mathcal{C}$, como puede verse en la Tabla 5.2(c) y la Figura 5.10(a), no hay ninguna solución de CC-EMO que sea dominada por APRIORI y, a su vez, APRIORI no domina a los otros dos algoritmos, mientras que SUBDUE sólo descubre una solución que domina al conjunto Pareto de CC-EMO. La métrica $\mathcal{ND}$ muestra en la Tabla 5.2(d) y la Figura 5.10(b), que CC-EMO descubre más soluciones no dominadas que los otros dos algoritmos. APRIORI y SUBDUE encuentran muy pocas soluciones que CC-EMO no llega a obtener (1,20 y 1,60 en promedio de la Tabla 5.2(d)). La diferencia de valores en la métrica $\mathcal{ND}$ para CC-EMO

en contraste con APRIORI y SUBDUE (181,89 y 171,80 vs. solamente 1,20 y 1,60 de la Tabla 5.2(d)) muestra cuán bien nuestra metodología se comporta en este dominio.

En cuanto a los tiempos de ejecución, para CC-EMO son aproximadamente de una hora y media, mientras que para APRIORI y SUBDUE son sólo de unos pocos minutos.

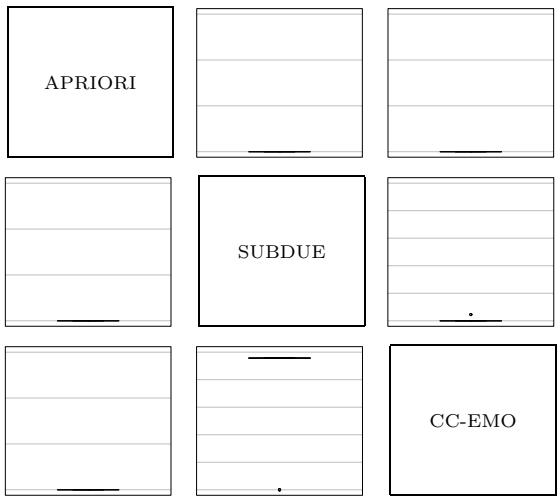
5.5. Compactación de la base de datos

Aparte de la información sobre los términos de GO para cada instancia de la base de datos, contamos con información sobre la expresión de cada uno de estos genes. Como ya hemos mencionado en la Sección 5.1, hemos agrupado esta información en 24 clusters mediante el algoritmo k-medias. Por lo tanto, tenemos los mismos genes agrupados de dos maneras diferentes utilizando información independiente una de la otra. Teniendo esta información de control, podremos realizar una depuración de los clusters obtenidos por CC-EMO conservando solamente aquellos que sean relevantes desde el punto de vista de los perfiles de expresión genética provistos por la información de control. Este proceso se realizará calculando la intersección de todos los clusters obtenidos por CC-EMO, a los cuales llamaremos *ClustersB*, contra aquellos del conjunto de control, llamados *ClustersA*.

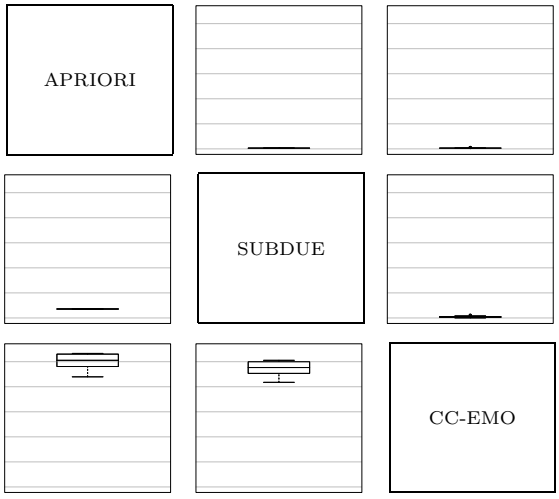
En las Figuras 5.11(a) y 5.11(b) se ilustran las intersecciones de APRIORI y SUBDUE con respecto a los perfiles de expresión. Además, la Figura 5.11(c) muestra las intersecciones con respecto a CC-EMO. Los gráficos representan cada intersección con un círculo, cuyo tamaño crece a medida que aumenta el número de elementos en la intersección entre los clusters, mientras que el color muestra el p-value para la intersección (ver Sección 3.6). Los valores de p-value mayores de $3{,}10^{-4}$ no se representan, ya que no serían suficientemente significativos.

APRIORI y SUBDUE encuentran un menor número de clusters relevantes comparados con CC-EMO. Es más, todos ellos están incluidos en el conjunto de clusters obtenidos por nuestra metodología. Notar que casi todas las intersecciones involucran a los clusters 15, 20 y 21 de ClustersA. Este conjunto de clusters contiene muchos conjuntos en ClustersB que los intersecan. Sin embargo, las descripciones asociadas a ellos no son muy específicas, con solamente uno o dos términos de GO en el segundo o tercer nivel de la jerarquía. A pesar de ello, son descripciones válidas que cubren un amplio conjunto de genes diferentes.

Estudiaremos ahora un par de ejemplos de estas intersecciones encontradas por CC-EMO para comprender mejor los resultados obtenidos. Comenzaremos estudiando el cluster 13 de ClustersA. En la Tabla 5.3(a) se recogen todos los clusters de ClustersB obtenidos por CC-EMO que intersecan al cluster 13 de ClusterA con un p-value menor que $3{,}10^{-4}$. En la Tabla 5.4 se muestran las

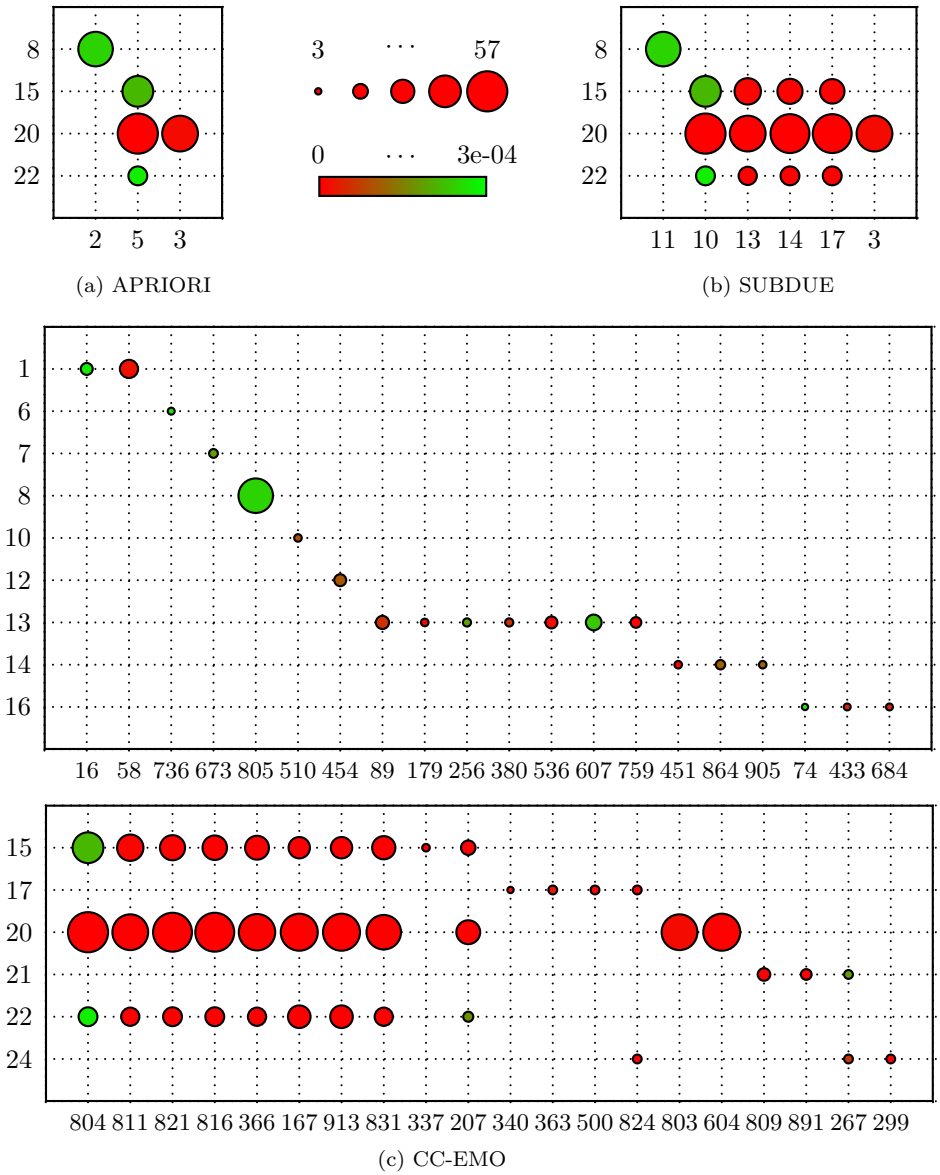


(a) Boxplots de la métrica $\mathcal{C}$



(b) Boxplots de la métrica $\mathcal{ND}$

Figura 5.10: Boxplots de las métricas $\mathcal{C}$ y $\mathcal{ND}$ para CC-EMO en el dominio de ontología de genes.



(a) APRIORI (b) SUBDUE

(c) CC-EMO

Figura 5.11: Intersección de los ClustersA y ClustersB para APRIORI, SUBDUE y CC-EMO. No se muestran todas las intersecciones de CC-EMO en esta gráfica, el gráfico completo se encuentra en el Apéndice A.1.

descripciones de GO para cada una de estas intersecciones. De las descripciones de los clusters de ClustersB que intersecan con el cluster 13 de ClustersA, obtenidas de la Tabla 5.4 y la Figura 5.12, se puede inferir que los genes cubiertos tienen una o varias de las siguientes propiedades:

ClusterB	Tamaño	Intersección	p-value
179	7	5	$2{,}2010^{-6}$
536	69	12	$1{,}5210^{-5}$
759	42	10	$1{,}4310^{-6}$
256	21	6	$1{,}9110^{-4}$
89	104	14	$5{,}7910^{-5}$
380	18	6	$5{,}4310^{-5}$
607	179	18	$2{,}3710^{-4}$

ClusterA #13 - Tamaño: 74

(a) Cluster 13 de ClusterA

ClusterB	Tamaño	Intersección	p-value
363	88	7	$2{,}7110^{-5}$
500	71	7	$6{,}3510^{-6}$
824	36	7	$5{,}0210^{-8}$
340	5	3	$1{,}2110^{-5}$

ClusterA #17 - Tamaño: 20

(b) Cluster 17 de ClusterA

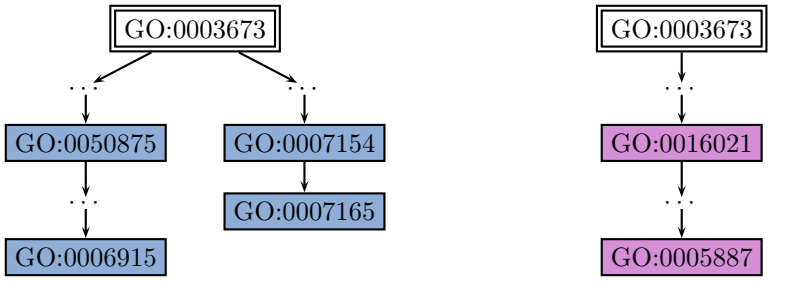
Tabla 5.3: Ejemplos de intersección de clusters

- Integran la membrana plasmática o, en un caso más general, la membrana, es decir, penetrando por lo menos en una bicapa fosforolípida de una membrana o de la membrana del plasmática. También se refiere al estado de ser enterrado en la bicapa sin exposición fuera de la misma.
- Trabajan en el proceso fisiológico celular o, en un caso más específico, en *apoptosis*[4]. Corresponde a los procesos pertinentes a la función integrada de una célula y, en el caso específico de apoptosis, a una forma de muerte programada de la célula inducida por las señales externas o internas que

[4]La muerte celular programada o *apoptosis* es el conjunto de reacciones bioquímicas que ocurren en las células cuando se diferencian y ejercen funciones normales, concluyendo tras un cierto número de divisiones celulares con la muerte celular de una forma ordenada y silenciosa. Por esta razón se conoce a la apoptosis como muerte celular programada.

#ClusterB	Proceso biológico	Función molecular	Componente celular
179	GO:0006915 apoptosis (level: 6)		GO:0005887 integral to plasma membrane (level: 4)
536	GO:0007165 signal transduction (level: 4)		GO:0016021 integral to membrane (level: 3)
759	GO:0007165 signal transduction (level: 4)		GO:0005887 integral to plasma membrane (level: 4)
89	GO:0007154 cell communication (level: 3)		GO:0016021 integral to membrane (level: 3)
256	GO:0007154 cell communication (level: 3) GO:0050875 cellular physiological process (level: 3)		GO:0016021 integral to membrane (level: 3)
380	GO:0007165 signal transduction (level: 4) GO:0050875 cellular physiological process (level: 3)		GO:0016021 integral to membrane (level: 3)
607		GO:0004871 signal transducer activity (level: 2)	GO:0016021 integral to membrane (level: 3)

Tabla 5.4: Clusters derivados de la información de GO por CC-EMO (ClustersB) que intersecan de manera significativa con el cluster 13 obtenido con la información de expresión (ClustersA).



(a) Proceso biológico (b) Componente celular

Figura 5.12: Sección de las jerarquías de GO.

accionan la actividad de ciertas enzimas, cuyas acciones desmontan la célula y tienen como resultado la muerte de la misma.

- Pertenecen a un proceso de la comunicación de la célula, que puede ser cualquier proceso que medie interacciones entre una célula y sus alrededores. Abarcan interacciones tales como señalización o enlace entre una célula y otra, entre una célula y una matriz extracelular, o entre una célula y cualquier otro aspecto de su entorno. También pueden señalar, de manera más específica, el proceso de transducción, el cual produce una cascada de los procesos por los cuales una señal obra recíprocamente con un receptor causando un cambio en el nivel o en la actividad de un segundo mensajero u otro blanco en sentido descendiente, y en última instancia, efectuar un cambio en el funcionamiento de la célula.

- Tienen una actividad de transducción de una señal, lo que significa que media la transferencia de una señal del exterior al interior de una célula por otros medios que la introducción de la molécula de la señal en sí misma en la célula.

En la Figura 5.13 se muestran los gráficos de niveles de expresión correspondientes a los genes que intersecan al cluster 13 de ClustersA. Cada subgráfico del árbol corresponde al conjunto de genes que se encuentran modelados por el ClusterB que se indica en la esquina superior derecha de cada gráfico. La jerarquía entre gráficos viene dada por su inclusión teniendo en cuenta las descripciones de GO asociadas a cada subestructura. También se puede ver que el cluster 380 de ClustersB es hijo del cluster 256 de ClustersB. Esto es debido a que, como puede observarse en la Tabla 5.4, la descripción para el cluster 380 es igual a la del cluster 256 salvo por un código de GO que, como puede verse en la Figura 5.12, es descendiente de aquel con el cual difiere del grupo 256. Por lo tanto, se puede concluir que el cluster 380 es descendiente del cluster 256.

Siguiendo el proceso de compactación explicado en la Sección 3.6, podemos eliminar el cluster 256 de ClustersB y quedarnos solamente con el cluster 380 de ClustersB, debido a que el primero es dominado por el segundo teniendo en cuenta los genes que componen el cluster 13 de ClustersA. Por otro lado, seleccionamos un cluster de cada rama, escogiendo el cluster 89 de ClustersB en la tercera, el cual es el cluster con una descripción más general.

Ahora estudiaremos el cluster 17 de ClustersA. En la Tabla 5.3(b) se muestran todos los clusters de ClustersB que intersecan con el cluster 17 de ClustersA con un p-value menor a $3{,}10^{-4}$, mientras que en la Tabla 5.5 se muestran las descripciones para cada una de las intersecciones. De las descripciones de los clusters de ClustersB que intersecan con el cluster 17 de ClustersA, obtenidas de la Tabla 5.5 y la Figura 5.14, se puede inferir que los genes cubiertos tienen una o varias de las siguientes propiedades:

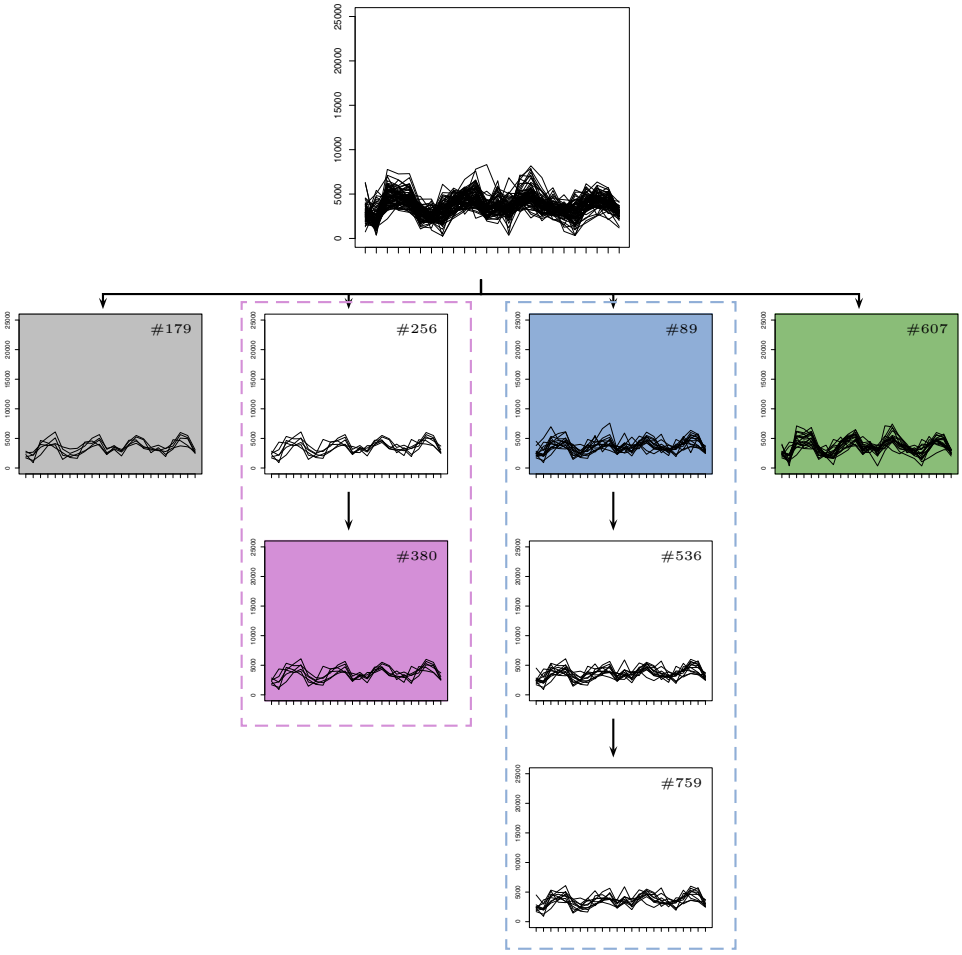
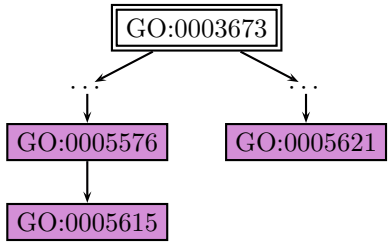


Figura 5.13: Expresión del cluster 13 de ClustersA y su relación con los ClustersB con los cuales interseca. Los ClustersB representados corresponden a la intersección entre éstos y el cluster 13 de ClustersA. La jerarquía viene dada por la relación entre los términos de GO de cada cluster conceptual. Los gráficos remarcados corresponden a aquellos clusters que contienen los mismos genes.

- Pertenecen a la región extracelular o al espacio extracelular. En el primer caso, esto significa que pertenecen al espacio externo a la estructura exterior de una célula. Para las células sin estructuras externas de encapsulado o protectoras, esto se refiere al espacio fuera de la membrana plasmática. Para el segundo caso, esto significa que pertenecen a la parte de un organismo multicelular fuera de la célula, tomadas generalmente para estar

#ClusterB	Proceso biológico	Función molecular	Componente celular
500		GO:0005102 receptor binding (level: 3)	
363			GO:0005576 extracellular region (level: 2)
824			GO:0005615 extracellular space (level: 3)
340	GO:0006954 inflammatory response (level: 5)		GO:0005621 intracellular (level: 3)

Tabla 5.5: Clusters derivados de la información de GO por CC-EMO (ClustersB) que intersecan de manera significativa con el cluster 17 obtenido con la información de expresión (ClustersA).



(a) Componente celullar

Figura 5.14: Sección de la jerarquía de GO

fuera de las membranas plasmáticas, y ocupadas por líquido.

- Son intracelulares, lo que significa que pertenecen al contenido vivo de una célula. Más detalladamente, la materia contenida dentro (pero no incluyendo) a la membrana plasmática, tomada generalmente para excluir las vacuolas y las masas grandes del material secretor o ingerido. En eucariotas, incluye el núcleo y el citoplasma. Además, tienen un proceso biológico de la respuesta inflamatoria, es decir, la reacción defensiva inmediata (por el tejido vertebrado) a la infección o lesión causada por los agentes químicos o físicos. El proceso se caracteriza por la vasodilatación local, la extravasación del plasma en espacios intercelulares y la acumulación de las células de sangre y de los macrófagos blancos.

Recordemos de la descripción del experimento biológico sobre el cual estamos trabajando, que éste estudia la respuesta inflamatoria a una endotoxina. Por lo

tanto, el conjunto de clusters de ClustersB extraídos por CC-EMO asociados al cluster 17 de ClustersA podría llegar a ser relevante para aquellos biólogos que han diseñando y desarrollado el experimento. Estamos, de hecho, mostrando que existe una relación entre la expresión de un grupo de genes y sus propiedades (procesos biológicos, función molecular y componente celular).

En la Figura 5.15 se muestran los gráficos de niveles de expresión correspondientes a los genes que intersecan al cluster 17 de ClustersA. Cada subgráfico del árbol corresponde al conjunto de genes que se encuentran modelados por el ClusterB que se indica en la esquina superior derecha de cada gráfico. La jerarquía entre gráficos viene dada por su inclusión teniendo en cuenta las descripciones de GO asociadas a cada subestructura. Por lo tanto, considerando esta nueva información externa de control, podemos reducir el conjunto de subestructuras extraídas por CC-EMO eliminando aquellos clusters que se encuentran dominados dentro de la intersección. Nuevamente, no es posible realizar esto sin información externa de control ya que los tres grupos no son dominados, con lo cual se deduce que involucran a un conjunto de elementos cuya intersección no es significativa con respecto al total de genes.

5.6. Predicción

Las subestructuras obtenidas por CC-EMO, y posteriormente depuradas durante la compactación mediante el uso de una base de datos con información externa, no sólo nos proveen de buenas descripciones de clusters comprensibles por el ser humano, sino que también proporcionan nuevas anotaciones para la base de datos original. Es decir, mientras que las anotaciones de esta base de datos original se basan en procesos biológicos, funciones moleculares y componentes celulares *realizadas de manera independiente*, nuestra propuesta logra definir nuevas anotaciones *que utilizan todos estos datos en forma conjunta*. Para ilustrar este hecho, en la Figura 5.16 se muestra un ejemplo de una nueva anotación basada en el cluster 380 de ClustersB. Como se puede ver gráficamente, en este caso la nueva anotación utiliza el conocimiento de la jerarquía de GO correspondiente a proceso biológico y a componente celular, generando una anotación compuesta, la cual permite definir de manera más específica y, por lo tanto, más informativa, a un conjunto de genes.

Una vez concluido el proceso de clustering y la compactación de los clusters generados, pueden clasificarse nuevas instancias en los diferentes conjuntos obtenidos. Dado que los clusters no son disjuntos, puede que una instancia dada pertenezca a uno o más conjuntos. Tal como se explica en la Sección 3.7, nuestra metodología utiliza un clasificador difuso basado en el k-prototipo más cercano. El cálculo del grado de pertenencia de la observación x_q con el conjunto I de subestructuras previamente identificadas se realiza mediante la ecuación 3.7.

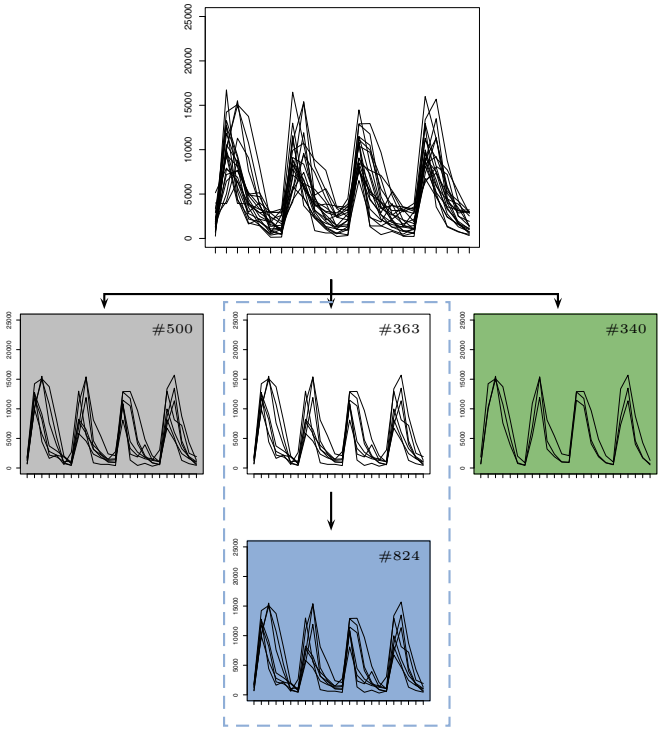


Figura 5.15: Expresión del cluster 17 de ClustersA y su relación con los ClustersB con los cuales interseca. Los ClustersB representados corresponden a la intersección entre éstos y el cluster 17 de ClustersA. La jerarquía viene dada por la relación entre los términos de GO de cada cluster conceptual. Los gráficos remarcados corresponden a aquellos clusters que contienen los mismos genes.

Nombre	Proceso biológico	Función molecular	Componente celular
`212659_s_at`	GO:0006954	GO:0005152	GO:0005615

Tabla 5.6: Descripción del gen `212659_s_at`.

Por ejemplo, queremos clasificar el gen `212659_s_at` (ver Tabla 5.6). De todos los clusters generados por CC-EMO y depurados en el proceso de validación, `212659_s_at` tiene un grado de pertenencia mayor a cero a los clusters 500, 340 y 824 de ClusterB (ver Figura 5.15). Estos valores son 0,27670, 0,34812 y 0,37518, respectivamente. Por lo tanto, clasificaremos a `212659_s_at` como perteneciente al cluster 824 de ClustersB:

$$knn(x_q = \texttt{212659\_s\_at}, V_{\#500}, V_{\#340}, V_{\#824}) = V_{\#824} \ /$$

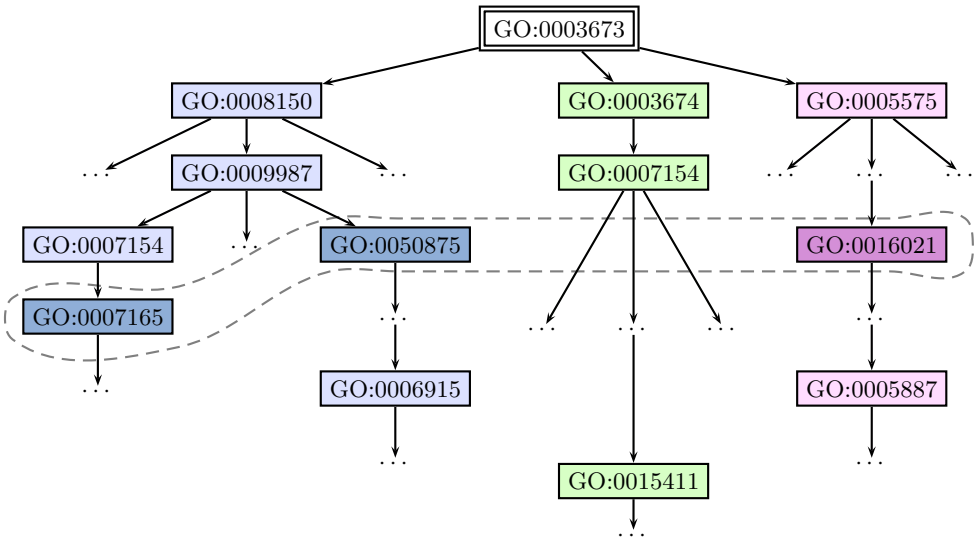


Figura 5.16: Ejemplo de una nueva anotación para el dominio de ontología de genes basado en el cluster 380 de ClustersB.

$$\mu_{i,q} = max\{\mu_{\#500,q}, \mu_{\#340,q}, \mu_{\#824,q}\} = \\ = max\{0, 27670, 0, 34812, 0, 37518\} = 0, 37518$$

Como puede verse en la Figura 5.17, su expresión es muy similar a aquellas de los genes pertenecientes al mismo conjunto.

5.7. Comentarios finales

En este capítulo se ha aplicado la metodología propuesta a un problema de regulación genética en organismos eucariotas, consistente en el estudio de la respuesta genética a procesos inflamatorios en seres humanos [79].

Para la construcción de la base de datos, se ha utilizado un repositorio estructurado llamado *Gene Ontology (GO)*. Este repositorio almacena información, en diversas jerarquías y a diferentes niveles, que provee un buen modelado de los genes estudiados en el experimento.

La aplicación de la metodología propuesta a este problema generó resultados superiores a los conseguidos por los otros dos enfoques –APRIORI y SUBDUE– utilizados en la comparativa.

El proceso de compactación basado en información externa, en este caso, datos sobre los niveles de expresión de cada gen, provee al usuario de una explicación de cada conjunto de genes que se expresan en forma similar. Esta explicación se realiza en base a los términos de GO. Las conceptos aprendidos

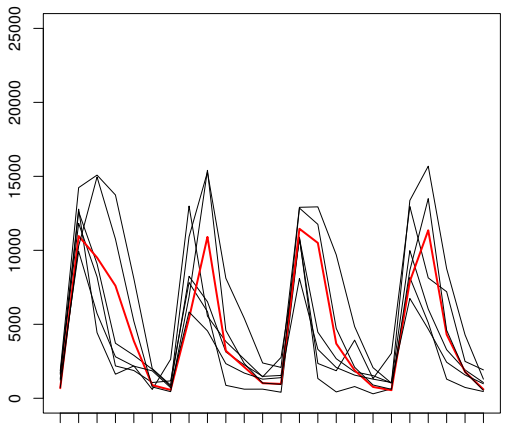


Figura 5.17: Expresión del cluster 824 de ClustersB, donde se clasifica el gen `212659_s_at`. La instancia clasificada se muestra en color rojo.

pueden contener información de las diferentes jerarquías de GO y, por lo tanto, permiten generar nuevas anotaciones de los genes utilizando una conjunción de diversos términos.

Por último, es importante destacar la capacidad de la metodología para predecir el comportamiento de nuevos genes a partir de los conceptos aprendidos y depurados, como se observa en el ejemplo de predicción de la Sección 5.6.

Comentarios finales

Dedicaremos esta sección a resumir brevemente los resultados obtenidos y a destacar las conclusiones que esta memoria puede aportar. Además, comentaremos algunos aspectos sobre trabajos futuros que siguen la línea aquí expuesta y sobre otras líneas de investigación que se pueden derivar.

I. Resumen y conclusiones

En esta memoria, hemos presentado una metodología general que permite la extracción de conocimiento interpretable de una base de datos estructurada. Esto se ha realizado bajo la perspectiva de obtener una metodología robusta y, a la vez, flexible y adaptable a diferentes problemas biológicos, e incluso de otras áreas. El eje central de la propuesta se basa en el modelado de una base de datos estructurada y en la utilización de la optimización multiobjetivo para conseguir distintas soluciones que permitan al usuario seleccionar aquellas de su preferencia y retroalimentar a la metodología haciendo uso de esta información. En concreto, hemos considerado el uso de los algoritmos evolutivos multiobjetivo como técnica para la extracción de las subestructuras de la base de datos.

Los siguientes apartados resumen brevemente los resultados obtenidos y presentan algunas conclusiones sobre los mismos. Aquellos resultados que ya han sido publicados van acompañados de las referencias bibliográficas correspondientes.

I.1. Metodología

A continuación se resumirán las características que distinguen a la metodología CC-EMO de otros métodos de minería de datos:

- CC-EMO considera varios criterios de clustering simultáneamente, a diferencia de una agregación en un único criterio de optimización. De esta

manera, es capaz de encontrar las soluciones óptimas en múltiples criterios (optimalidad de Pareto [33]), lo cual evita los sesgos que pueden resultar de utilizar un esquema específico de pesos [80]. Esto permite detectar clusters cohesivos con un soporte reducido que agrupan un número pequeño de genes, que no pueden detectarse por métodos que optimizan únicamente el número de instancias en una subestructura [2].

- CC-EMO permite la pertenencia de una instancia a más de una subestructura utilizando un clasificador flexible [18, 26, 40]. Por lo tanto, trata explícitamente a las subestructuras como hipótesis, que pueden ser luego validadas y refinadas [66]. Esto distingue a CC-EMO de otros enfoques de clustering que prematuramente fuerzan a las instancias a pertenecer a clusters disjuntos.

- CC-EMO realiza una selección de características local para cada subestructura, dado que no todas las características son relevantes para todos los grupos [58], y a priori no se conoce qué característica será relevante para un conjunto dado de instancias. Por tanto, es una filosofía muy diferente a los enfoques que filtran o reducen características para todos los posibles clusters [91]. Esto se ha logrado mediante el uso de la programación genética [59], la cual permite construir conceptos incrementalmente, incorporando sólo aquellas características que sean relevantes para el cluster que cada concepto define.

- CC-EMO tiene una naturaleza multimodal que permite la descripción alternativa de un sistema brindando de varias soluciones adecuadas [33, 82], recuperando soluciones localmente óptimas, lo cual puede llegar a ser relevante al usuario. Esto diferencia a CC-EMO de otros métodos que están focalizados en un único óptimo.

- CC-EMO difiere de los métodos de aprendizaje supervisado, que agrupan las características e instancias en base a la definición explícita de variables dependientes. CC-EMO considera los datos externos como una característica independiente y, por ello, permite la clasificación de instancias en su ausencia. En otras palabras, no realiza un agrupamiento basado en una clase dada por un experto. Más aún, CC-EMO minimiza el número de subestructuras utilizando una estrategia de compactación flexible que agrupa subestructuras similares basada en la capacidad de describir las instancias utilizando características independientes a las aplicadas en el proceso de aprendizaje de las subestructuras, en contraposición a otros enfoques que utilizan una compactación de base de datos irreversible [54].

I.2. Diseño de modelos de objetos biológicos

Para poder aplicar nuestra metodología a distintos problemas en el área de la bioinformática, es necesario diseñar bases de datos estructuradas de instancias biológicas. Para ello, se requiere la extracción de características de estos objetos a través de modelos de los mismos. A continuación, se resumirán las técnicas utilizadas para obtener los distintos modelos generados:

- Se han modelado diferentes características de las regiones reguladoras de un gen mediante técnicas de clustering y modelado difuso [99]. Para ello, se han utilizado características propias de la regulación transcripcional, incluyendo características *cis* (por ejemplo, factores de transcripción, distancias entre ellos, etc.) y perfiles de expresión. El procedimiento general para el modelado de estos objetos biológicos consta de tres fases:

  - Modelar las instancias mediante un conjunto de atributos, siguiendo varios pasos, entre los cuales se incluyen: (1) analizar los niveles de expresión de los genes, obtenidos a través de experimentos de microarray. Esto se lleva a cabo identificando las características *cis* de las secuencias de ADN, a partir de las bases de datos disponibles; y (2) realizar un modelado inicial de cada característica, para luego examinar las regiones promotoras del genoma detectando estos atributos, aceptando múltiples ocurrencias de una misma característica.
  - Realizar un clustering de los resultados en subgrupos, modelando las características, y generando atributos compuestos por medio de la descomposición o combinación de diferentes características individuales. De esta forma se logra aumentar la confianza en ellas, en base a una mayor cantidad de atributos compartidos.
  - Describir las regiones promotoras utilizando los modelos resultantes.

- En particular, se ha aplicado un modelado difuso para la búsqueda y reconocimiento del sitio de *binding* de la ARN polimerasa (la enzima que permite la transcripción de un gen), conocido con el nombre de *promotor* [5]. Para ello, se ha utilizado diferentes técnicas para llevar a cabo la búsqueda, algoritmos genéticos [77, 95] y un sistema híbrido redes neuronales-algoritmos genéticos [29].

  Esto se ha conseguido al utilizar sistemas genéticos difusos para solucionar el problema del reconocimiento de promotores, un problema de optimización multimodal y multiobjetivo. El método de reconocimiento propuesto, validado mediante la predicción de promotores del organismo *E. coli*, combina las ventajas de la representación de características basadas en conjuntos difusos y las capacidades de búsqueda de los algoritmos evolutivos multiobjetivo para obtener soluciones precisas y a la vez interpretables [77, 78]. En particular, esta clase de soluciones ayuda a descubrir posibles

sitios de transcripción (sitios putativos) mediante la detección de múltiples instancias de promotores en una misma secuencia. Esto provee una descripción completa de diversas posibilidades de regulación que pueden ocurrir en las regiones intergénicas de un genoma, permitiendo la predicción de distintas actividades reguladoras, tanto de activación como de represión.

El proceso de reconocimiento y los modelos difusos obtenidos se han optimizado mediante un algoritmo de ajuste basado en algoritmos genéticos en [95], extendido posteriormente utilizando un algoritmo de ajuste multiobjetivo también basado en algoritmos genéticos. El ajuste multiobjetivo, a diferencia del monobjetivo, produjo un conjunto resultados posibles al problema planteado, en lugar de una única solución. De esta manera, se provee al usuario de diferentes opciones y éste selecciona aquella que más se ajusta a sus necesidades. Para nuestra aplicación se optó por una solución global que mejor generalizaba la base de datos.

I.3. Aplicación a organismos procariotas

Hemos aplicado nuestra metodología al problema de regulación genética en organismos procariotas. En la primera etapa de la metodología, se ha modelado la información obtenida de la base de datos en bruto utilizando técnicas basadas en lógica difusa, aprendizaje automático y algoritmos genéticos multiobjetivo. Gracias a este proceso, se han podido detectar, en los genomas de *E. coli* y *Salmonella*, secuencias con características similares a las de los modelos obtenidos, que no habían sido detectadas anteriormente. Estas nuevas instancias de los modelos se han verificado biológicamente mediante diversas técnicas de biología molecular (por ejemplo, *footprinting in vivo*) [100]. De esta manera, se ha logrado ampliar la base de datos existente, mejorando, a su vez, el proceso de clustering posterior.

Al finalizar la aplicación de las distintas etapas de la metodología, se logró obtener como resultado un conjunto de clusters que representaban conceptos diversos y agrupaban a los distintos genes de la base de datos en base a la combinación de las diferentes características modeladas. Gracias a trabajos realizados por otros autores sobre estos genes [100, 89], pudimos comprobar que los conceptos inferidos por nuestra metodología son correctos y cohesivos, ayudando a describir grupos de genes que se expresan en forma similar en base a propiedades obtenidas de sus regiones reguladoras. De esta manera, se provee al usuario de una herramienta de caja blanca robusta y, a la vez, flexible a sus necesidades.

I.4. Aplicación a organismos eucariotas

Hemos utilizado nuestra metodología en el estudio de la respuesta genética a procesos inflamatorios en seres humanos. Para ello, modelamos las instancias de la base de datos utilizando la información provista por la base de datos "Gene Ontology" [8]. Esta base de datos estructurada provee información sobre las distintas propiedades de las secuencias biológicas, en base a diferentes niveles de especificidad. La aplicación de la metodología CC-EMO logra identificar clusters conceptuales y clasificar un grupo de genes co-regulados en base a múltiples características, incluyendo descripciones funcionales, procesos moleculares y componentes celulares. CC-EMO genera subestructuras que agrupan genes que comparten conjuntos de características [76]. Las subestructuras generadas son tratadas como hipótesis, que luego se emplean para conducir búsquedas en genomas completos. Esto produce anotaciones en diferentes tipos de características en múltiples niveles de especificidad, para capturar miembros adicionales de una subestructura, consiguiendo así caracterizaciones de perfiles de genes más apropiadas.

CC-EMO utiliza un entorno multivariable y multinivel, donde las subestructuras se descubren en base a diversos tipos de características jerárquicas. Por ejemplo, las subestructuras extraídas de la base de datos de "Gene Ontology" incluyen características o términos derivados de diferentes tipos de información. Más aún, cada característica está definida a distintos niveles de especificidad en un estructura basada en grafos. Esto distingue nuestra metodología de los enfoques previos, donde las características se tratan individualmente en un tipo de información particular, y el nivel de especificidad se selecciona a priori [4].

II. Líneas de investigación futuras

A continuación, consideraremos algunas extensiones posibles sobre la metodología propuesta en esta memoria y discutiremos varias líneas de investigación inmediatas.

Con respecto al modelado de objetos biológicos correspondiente a las características de la región reguladora de organismos procariotas, se plantean varias líneas futuras posibles:

- Sería interesante realizar un análisis exhaustivo de diferentes métodos –redes neuronales, modelos ocultos de Markov, redes bayesianas, etc.– que permita mejorar el modelado de los distintos objetos biológicos y, por lo tanto, su posterior reconocimiento. Mediante este análisis se podría llegar a realizar la búsqueda de los diferentes objetos biológicos, cada uno utilizando el método de reconocimiento que mejor se adapte a sus necesidades.

- El desarrollo de una metodología que permita el reconocimiento de diferentes clases de factores σ y su relación con el resto de los componentes de la

región reguladora de un gen también sería una línea de acción interesante.

En relación con el modelado de objetos biológicos correspondiente a las características, en términos de la base de datos "Gene Ontology", que presentan las secuencias biológicas eucariotas, se propone una extensión del modelado presentado en esta memoria que permita un emparejamiento difuso de las subestructuras y las instancias cubiertas por las mismas. Utilizando un modelado difuso se puede llegar mejorar la representación, con la consecuente ventaja en la aplicación al dominio eucariota que beneficiaría a sus usuarios.

Con respecto a la etapa central de la metodología propuesta, el clustering conceptual multiobjetivo, se propone la posibilidad de incorporar la capacidad de descubrir diferentes explicaciones de un mismo cluster, aún cuando una de ellas supere a las otras en todos los objetivos a optimizar. Para ello, se necesita incorporar a CC-EMO la capacidad de crear nichos en el espacio genotípico. Esto implica comparar no sólo si dos subestructuras representan a las mismas instancias de la base de datos, sino también si genotípicamente son similares, es decir, si están compuestas por un mismo subconjunto de características. Esto brindaría al usuario más de una descripción posible para un mismo cluster, lo que permitiría un mejor sistema de retroalimentación, evitando proveer de soluciones que no sean las requeridas por el usuario.

En referencia a las posibles aplicaciones de la metodología a otros dominios, existen diversos problemas biológicos que guardan información en bases de datos estructuradas con los que podríamos trabajar directamente. Un ejemplo es la base de datos BIND (Biomolecular Interaction Network Database) [6], que acumula información sobre *interacciones proteína-proteína*. Esta base de datos es una colección de información sobre interacciones moleculares, cuyos contenidos incluyen información recuperada de la literatura científica y de datos obtenidos mediante experimentos biológicos de gran escala. En el repositorio BIND se acumulan asociaciones moleculares con tres clasificaciones: moléculas que se asocian unas con otras para formar interacciones, complejos moleculares que están conformadas por una o más interacciones y caminos que están definidas por una secuencia específica de una o más interacciones. La aplicación de nuestra propuesta es sencilla de realizar, solamente se necesitaría un correcto modelado de la base de datos en bruto y, posiblemente, pequeños cambios en las funciones objetivo del algoritmo evolutivo utilizado en la etapa de clustering conceptual.

A corto plazo, se pretende aplicar la metodología propuesta a otras bases de datos, en particular, *Ingenuity* [51] y *ResNet y ResArt* de *Pathway Assist* [70].

Finalmente, señalaremos que la metodología propuesta no es de uso exclusivo para dominios biológicos y puede ser utilizada en una gran variedad de dominios con información estructurada, tal como se mostró para el dominio geométrico.

Apéndice A

Tablas y figuras adicionales

En este apéndice se recogen los resultados publicados por Harley & Reynolds [48], indicando la ubicación de los promotores, originales y alternativos, de cada secuencia, y los resultados completos para CC-EMO aplicado al problema de organismos eucariotas.

secuencia	TTGACA	TATAAT	Promotor
trpS	15	38	CGGCGAGGCTATCG ATCTCA GCCAGCCT GATGTAATT TATCAG TCTATAAATGACC
trpS	12	33	CGGCGAGGCTA TCGATC TCAGCCA GCCTGATG TAATTT ATCAGTCTATAAATGACC
trxA	15	39	CAGCTTACTATTGC TTTACG AAAGCGTAT CCGGTGAAA TAAAGT CAACTAGTTGGTTAA
tufB	15	38	ATGCAATTTTTTAG TTGCAT GAACTCGC ATGTCTCCA TAGAAT GCGCGCTACTTGATGCC
tyrT	15	37	TCTCAACGTAACAC TTTACA GCGGCGCG TCATTTGA TATGAT GCGCCCCGCTTCCCGAT
tyrT/109	15	39	ACAGCGCGTCTTTG TTTACG GTAATCGAA CGATTATTC TTTAAT CGCCAGCAAAAATAA
tyrT/140	15	39	TTAAGTCGTCACTA TACAAA GTACTGGCA CAGCGGGTC TTTGTT TACGGTAATCG
tyrT/140	13	39	TTAAGTCGTCAC TATACA AAGTACTGGC ACAGCGGGTC TTTGTT TACGGTAATCG
tyrT/178	13	34	TGCGCGCAGGTC GTGACG TCGAGAA AAACGTCT TAAGTC GTGCACTATACA
tyrT/178	13	33	TGCGCGCAGGTC GTGACG TCGAGAA AAACGTC TTAAGT CGTGCACTATACA
tyrT/212	2	24	C ATGTCG ATCATACC TACACAGC TGAAGA TATGATGCGCGCAGGTCGTGACG
tyrT/6	13	35	ATTTTTCTCAAC GTAACA CTTTACAG GCGCGTCA TTTGAT ATGATGCGCCCGCTTC
tyrT/6	20	40	ATTTTTCTCAACGTAACAC TTTACA GGCGCGT CATTTGA TATGAT GCGCCCCGCTTC
tyrT/77	13	38	ATTATTCTTTAA TCGCCA GCAAAAATA ACTGGTTACC TTTAAT CCGTTACGGATGAAAAT
tyrT/77	32	54	ATTATTCTTTAATCGCCAGCAAAAATAACTG GTTACC TTTAATCC GTTACGGAT GAAAAT
uncI	15	37	TGGCTACTTATTGT TTGAAA TCACGGGG GCGCACCG TATAAT TTGACCGCTTTTTGAT
uncI	14	37	TGGCTACTTATTG TTTGAA ATCACGGGG GCGCACCG TATAAT TTGACCGCTTTTTGAT
uvrB-P1	15	38	TCCAGTATAATTTG TTGGCA TAATTAAG TACGACGAG TAAAAT TACATACCTGCCCGC
uvrB-P2	15	39	TCAGAAATATTATG GTGATG AACTGTTTT TTTATCCAG TATAAT TTGTTGGCATAATTAA
uvrB-P3	15	38	ACAGTTATCCACTA TTCCTG TGGATAAC CATGTGTAT TAGAGT TAGAAAACACGAGGCA
uvrC	15	38	GCCCATTTGCCAGT TTGTCT GAACGTGA ATTGCAGAT TATGCT GATGATCACCAAGG
uvrD	15	37	TGGAAATTTCCCGC TTGGCA TCTCTGAC CTCGCTGA TATAAT CAGCAAATCTGTATAT
434PR	15	38	AAGAAAAACTGTAT TTGACA AACAAGAT ACATTGTAT GAAAAT ACAAGAAAGTTTGTTGA
434PRM	15	38	ACAATGTATCTTGT TTGTCA AATACAGT TTTTCTTGT GAAGAT TGGGGGTAAATAACAGA

Tabla A.1: Descripción de los promotores de la base de datos utilizada en la experimentación.

secuencia	TTGACA	TATAAT	Promotor
aceEF	13	36	ACGTAGACCTGT CTTATT GAGCTTTC CGGCGAGAG TTCAAT GGGACAGGTCCAG
aceEF	9	31	ACGTAGAC CTGTCT TATTGAGC TTTCCGGC GAGAGT TCAATGGGACAGGTCCAG
ada	15	38	AAGATTGTTGGTTT TTGCGT GATGGTGA CCGGGCAGC CTAAAG GCTATCCTTAACC
ada	15	39	AAGATTGTTGGTTT TTGCGT GATGGTGA CCGGGCAGCC TAAAGG CTATCCTTAACC
ada	23	46	AAGATTGTTGGTTTTTGCGTGA TGGTGA CCGGGCAGC CTAAAGGC TATCCT TAACC
alaS	15	39	AACGCATACGGTAT TTTACC TTCCCAGTC AAGAAAACT TATCTT ATTCCCACTTTTCAGT
ampC	15	37	TGCTATCCTGACAG TTGTCA CGCTGATT GGTGTCGT TACAAT CTAACGCATCGCCAATG
ampC/C16	7	30	GCTATC TTGACA GTTGTCAC GCTGATTGG TATCGT TACAATCTAACGTATCG
araBAD	15	37	TTAGCGGATCCTAC CTGACG CTTTTTAT CGCAACTC TCTACT GTTTCTCCATACCCGTT
araBAD	15	39	TTAGCGGATCCTAC CTGACG CTTTTTAT CGCAACTCTC TACTGT TTCTCCATACCCGTT
araC	15	38	GCAAATAATCAATG TGGACT TTTCTGCC GTGATTATA GACACT TTTGTTACGCGTTTTTG
araE	12	37	CTGTTTCCGAC CTGACA CCTGCGTGA GTTGTTCACG TATTTT TTCACTATGTCTTACTC
araI(c)	13	35	AGCGGATCCTAC CTGGCG CTTTTTAT CGCAACTC TCTACT GTTTCTCCATACCCGTT
araI(c)X(c)	13	37	AGCGGATCCTAC CTGGCG CTTTTTATC GCAACTCTC TACTAT TTCTCCATACCCGTTTT
argCBH	15	39	TTTGTTTTTCATTG TTGACA CACCTCTGG TCATGATAG TATCAA TATTCATGCAGTATT
argCBH	15	36	TTTGTTTTTCATTG TTGACA CACCTCTGG TCATGA TAGTAT CAATATTCATGCAGTATT
argCBH-P1/6-	15	36	TTTGTTTTTCATTG TTGACA CACCTCT GGTCATAA TATTAT CAATATTCATGCAGTAT
argCBH-P1/LL	15	36	TTTGTTTTTCATTG TTGACA CACCTCT GGTCATGA TATTAT CAATATTCATGCAGTAT
argE-P1	15	38	TTACGGCTGGTGGG TTTTAT TACGCTCA ACGTTAGTG TATTTT TATTCATAAATACTGCA
argE-P2	15	38	CCGCATCATTGCTT TGCGCT GAAACAGT CAAAGCGGT TATGTT CATATGCGGATGGCG
argE-P2	14	38	CCGCATCATTGCT TTGCGC TGAAACAGT CAAAGCGGT TATGTT CATATGCGGATGGCG
argE/LL13	15	38	CCGCATCATTGCTT TGCGCT GAAACAGT CAAAGCGGT TATATT CATATGCGGATGGCG
argF	15	38	ATTGTGAAATGGGG TTGCAA ATGAATAA TTACACATA TAAAGT GAATTTTAATTCAATAA
argI	7	30	TTAGAC TTGCAA ATGAATAA TCATCCATA TAAATT GAATTTTAATTCATTGA
argR	12	35	TCGTCGCCGCG TTGCAG GAGCAAGG CTTTGACAA TATTAA TCAGTCTAAAGTCTCGG
argR	11	36	TCGTCGCCGC GTTGCA GGAGCAAGG CTTTGACAAT ATTAAT CAGTCTAAAGTCTCGG
aroF	15	37	TACGAAAATATGGA TTGAAA ACTTTACT TTATGTGT TATCGT TACGTCATCCTCGCTG
aroG	15	38	AGTGTAAAACCCCG TTTACA CATTCTGA CGGAAGATA TAGATT GGAAGTATTGCATTCA
aroH	15	37	GTACTAGAGAACTA GTGCAT TAGCTTAT TTTTTTGT TATCAT GCTAACCACCCGGCGAG
bioA	15	39	GCCTTCTCCAAAAC GTGTTT TTTGTTGTT AATTCGGTG TAGACT TGTAAACCTAAATCT
bioA	25	47	GCCTTCTCCAAAAACGTGTTTTTTG TTGTTA ATTCGGTG TAGACTTG TAAACC TAAATCT
bioB	15	38	TTGTCATAATCGAC TTGTAA ACCAAATT GAAAAGATT TAGGTT TACAAGTCTACACCGAA
bioP98	15	38	TTGTTAATTCGGTG TAGACT TGTAAACC TAAATCTTT TAAATT TGGTTTACAAGTCGAT
C62.5-P1	14	37	CACCTGCTCTCGC TTGAAA TTATTCTC CCTTGTCCC CATCTC TCCCACATCCTGTTTT
C62.5-P1	13	34	CACCTGCTCTCG CTTGAA ATTATTCTC CCTTGT CCCCAT CTCTCCCACATCCTGTTTT
carAB-P1	15	38	ATCCCGCCATTAAG TTGACT TTTAGCGC CCATATCTC CAGAAT GCCGCCGTTTGCCAGA
carAB-P2	15	39	TAAGCAGATTTGCA TTGATT TACGTCATC ATTGTGAAT TAATAT GCAAATAAAGTGAG
cat	13	36	ACGTTGATCGGC ACGTAA GAGGTTCC AACTTTCAC CATAAT GAAATAAGATCACTACC
cat	15	36	ACGTTGATCGGCAC GTAAGA GGTTCCA ACTTTCAC CATAAT GAAATAAGATCACTACC
cat	23	46	ACGTTGATCGGCACGTAAGAGG TTCCAA CTTTCACC ATAATGAAA TAAGAT CACTACC
cit.util-379	13	37	AAACAGGCGGGG GTCTCA GGCGACTAA CCCGCAAAC TCTTAC CTCTATACATAATTCTG
cit.util-379	22	46	AAACAGGCGGGGGTCTCAGGC GACTAA CCCGCAAAC TCTTACCTC TATACA TAATTCTG
cit.util-431	14	38	GACAGGCACAGCA TTGTAC GATCAACTG ATTTGTGCC AATAAT TAAATGAAATCAC
CloDFcloacin	15	37	TCATATATTGACAC CTGAAA ACTGGAGG AGTAAGGT AATAAT CATACTGTGTATATAT
CloDFcloacin	8	31	TCATATA TTGACA CCTGAAAA CTGGAGGAG TAAGGT AATAATCATACTGTGTATATAT
CloDFnaI	15	39	ACACGCGGTTGCTC TTGAAG TGTGCGCCA AAGTCCGGC TACACT GGAAGGACAGATTTGG
colE1-B	15	36	TTATAAAATCCTCT TTGACT TTTAAAA CAATAAGT TAAAAA TAAATACTGTAA
colE1-B	9	31	TTATAAAA TCCTCT TTGACTTT TAAAACAA TAAGTT AAAAATAAATACTGTAA
colE1-C	15	37	TTATAAAATCCTCT TTGACT TTTAAAAC AATAAGTT AAAAAT AAATACTGTACATATAA
colE1-P1	15	38	GGAAGTCCACAGTC TTGACA GGGAAAAT GCAGCGGCG TAGCTT TTATGCTGTATATAAAA
colE1-P2	15	37	TTTTTAACTTATTG TTTTAA AAGTCAAA GAGGATTT TATAAT GGAAACCGCGGTAGCGT
colE1-P2	12	37	TTTTTAACTTA TTGTTT TAAAAGTCAAA GAGGATTT TATAAT GGAAACCGCGGTAGCGT
colE110.13	13	37	GCTACAGAGTTC TTGAAG TAGTGGCCC GACTACGGC TACACT AGAAGGACAGTATTTGG
colicinE1 P3	15	37	TTTTTAACTTATTG TTTTAA AAGTCAAA GAGGATTT TATAAT GGAAACCGCGGTAGCGT
crp	15	38	AAGCGAGACACCAG GAGACA CAAAGCGA AAGCTATGC TAAAAC AGTCAGGATGCTACAG
cya	15	38	GTAGCGCATCTTTC TTTACG GTCAATCA GCAAGGTGT TAAATT GATCACGTTTTAGACC
dapD	15	39	AAGTGCATCAGCGG TTGACA GAGGCCCTC AATCCAAAC GATAAA GGGTGATGTGTTTACTG
deo-P1	14	39	CAGAAACGTTTTA TTCGAA CATCGATCT CGTCTTGTGT TAGAAT TCTAACATACGGTTGC
deo-P2	10	35	TGATGTGTA TCGAAG TGTGTTGCG GAGTAGATGT TAGAAT ACTAACAAACTCGCAA
deo-P3	15	37	ACACCAACTGTCTA TCGCCG TATCAGCG AATAACGG TATACT GATCTGATCATTTAAA
divE	15	38	AAACAAATTAGGGG TTTACA CGCCGCAT CGGGATGTT TATAGT GCGCGTCATTCCGGAAG
dnaA-1p	15	39	TGCGGCGTAAATCG TGCCCG CCTCGCGGC AGGATCGTT TACACT TAGCGAGTTCTGGAAA
dnaA-1p	14	39	TGCGGCGTAAATC GTGCCC GCCTCGCGGC AGGATCGTT TACACT TAGCGAGTTCTGGAAA
dnaA-2p	15	38	TCTGTGAGAAACAG AAGATC TCTTGCGC AGTTTAGGC TATGAT CCGCGGTCCCGATCG
dnaK-P1	15	39	TTTGCATCTCCCCC TTGATG ACGTGGTTT ACGACCCCA TTTAGT AGTCAACCGCAGTG
dnaK-P1	14	34	TTTGCATCTCCCC CTTGAT GACGTGG TTTACGA CCCCAT TTAGTAGTCAACCGCAGTG
dnaK-P2	15	37	ATGAAATTGGGCAG TTGAAA CCAGACGT TTCGCCCC TATTAC AGACTCACAACCACA
dnaK-P2	14	33	ATGAAATTGGGCA GTTGAA ACCAGA CGTTTCG CCCCTA TTACAGACTCACAACCACA
dnaQ-P1	15	37	GCCAGCGCTAAAGG TTTTCT CGCGTCCG CGATAGCG TAAAAT AGCGCCGTAACCCC
Fpla-oriTpX	15	38	GAACCACCAACCTG TTGAGC CTTTTTGT GGAGTGGGT TAAATT ATTTACGGATAAAG
Fplas-traM	15	38	ATTAGGGGTGCTGC TAGCGG CGCGGTGT GTTTTTTTA TAGGAT ACCGCTAGGGGCGCTG
Fplas-traM	12	38	ATTAGGGGTGC TGCTAG CGGCGCGGTGT GTTTTTTTA TAGGAT ACCGCTAGGGGCGCTG
Fplas-traY/Z	14	37	GCGTTAATAAGGT GTTAAT AAAATATA GACTTTCCG TCTATT TACCTTTTCTGATTATT
Fplas-traY/Z	15	37	GCGTTAATAAGGTG TTAATA AAATATA GACTTTCCG TCTATT TACCTTTTCTGATTATT
Fplas-traY/Z	3	26	GC GTTAAT AAGGTGTT AATAAAATA TAGACT TTCCGTCTATTTACCTTTTCTGATTATT
frdABCD	12	34	GATCTCGTCAA ATTTCA GACTTATC GATCAGAC TATACT GTTGTACCTATAAAGGA
frdABCD	14	36	GATCTCGTCAAAT TTCAGA CTTATCGA TCAGACTA TACTGT TGTACCTATAAAGGA
fumA	15	38	GTACTAGTCTCAGT TTTTGT TAAAAAAG TGTGTAGGA TATTGT TACTCGCTTTTAACAGG
fumA	17	38	GTACTAGTCTCAGTTT TTGTTA AAAAAGTG TGTAGGA TATTGT TACTCGCTTTTAACAGG
γ-δ-tnpA	15	38	ACACATTAACAGCA CTGTTT TTATGTGT GCGATAATT TATAAT ATTTCGGACGGTTGCA
γ-δ-tnpR	14	36	ATTCATTAACAAT TTTGCA ACCGTCCG AAATATTA TAAATT ATCGCACACATAAAAAC
γ-δ-tnpR	15	36	ATTCATTAACAATT TTGCAA CCGTCCG AAATATTA TAAATT ATCGCACACATAAAAAC

Tabla A.2: Descripción de los promotores de la base de datos utilizada en la experimentación.

secuencia	TTGACA	TATAAT	Promotor
gal-P1	15	38	TCCATGTCACACTT TTCGCA TCTTTGTT ATGCTATGG TTATTT CATACCATAAG
gal-P1	6	28	TCCAT GTCACA CTTTTCGC ATCTTTGT TATGCT ATGGTTATTTCATACCATAAG
gal-P2	15	37	CTAATTTATTCCAT GTCACA CTTTTCGC ATCTTTGT TATGCT ATGGTTATTTCATACC
gal-P2	13	37	CTAATTTATTCC ATGTCA CACTTTTCG CATCTTTGT TATGCT ATGGTTATTTCATACC
gal-P2/mut-1	14	36	TAATTTATTCCAT GTCACA CTTTTCGC ATCTTTGT TATACT ATGGTTATTTCATAC
gal-P2/mut-1	14	41	TAATTTATTCCAT GTCACA CTTTTCGCATCTTTGTTATAC TATGGT TATTTCATAC
gal-P2/mut-2	14	36	TAATTTATTCCAT GTCACA CTTTTCGC ATTTTTGT TATGCT ATGGTTATTTCATAC
glnL	15	40	CAATTCTCTGATGC TTCGCG CTTTTTATC CGTAAAAAGC TATAAT GCACTAAATGGTGC
gln	15	38	TAAAAAACTAACAG TTGTCA GCCTGTCC CGCTTATAA GATCAT ACGCCGTTATACGTT
gltA-P1	15	37	ATTCATTCGGGACA GTTATT AGTGGTAG ACAAGTTT AATAAT TCGGATTGCTAAGTA
gltA-P1	9	34	ATTCATTC GGGACA GTTATTAGTG GTAGACAAG TTTAAT AATTCGGATTGCTAAGTA
gltA-P2	15	39	AGTTGTTACAAACA TTACCA GGAAAAGCA TATAATGCG TAAAAG TTATGAAGTCGGT
gltA-P2	7	30	AGTTGT TACAAA CATTACCAG GAAAAGCA TATAAT GCGTAAAAGTTATGAAGTCGGT
glyA	15	38	TCCTTTGTCAAGAC CTGTTA TCGCACAA TGATTCGGT TATACT GTTCGCCGTTGTCC
glyA/geneX	15	39	ACACCAAAGAACCA TTTACA TTGCAGGGC TATTTTTTA TAAGAT GCATTTGAGATACAT
gnd	15	38	GCATGGATAAGCTA TTTATA CTTTAATA AGTACTTTG TATACT TATTTGCGAACATTCCA
groE	11	34	TTTTTCCCCC TTGAAG GGGCGAAG CCATCCCCA TTTCTC TGGTCACCAGCCGGGAA
groE	10	29	TTTTTCCCC CTTGAA GGGGCG AAGCCAT CCCCAT TTCTCTGGTCACCAGCCGGGAA
gyrB	11	38	CGGACGAAAA TTCGAA GATGTTTACCGTGGAAAAGGG TAAAAT AACGGATTAACCCAAGT
his	14	38	ATATAAAAAGTTC TTGCTT TCTAACGTG AAAGTGGTT TAGGTT AAAAGACATCAGTTGAA
hisA	15	38	GATCTACAAACTAA TTAATA AATAGTTA ATTAACGCT CATCAT TGTACAATGAACTGTAC
hisA	15	41	GATCTACAAACTAA TTAATA AATAGTTA ATTAACGCTCAT CATTGT ACAATGAACTGTAC
hisA	23	46	GATCTACAAACTAATTAATAAA TAGTTA ATTAACGC TCATCATTG TACAAT GAACTGTAC
hisBp	15	38	CCTCCAGTGCGGTG TTTAAA TCTTTGTG GGATCAGGG CATTAT CTTACGTGATCAG
hisJ(St)	15	37	TAGAATGCTTTGCC TTGTCG GCCTGATT AATGGCAC GATAGT CGCATCGGATCTG
hisJ(St)	10	34	TAGAATGCT TTGCCT TGTCGGCCT GATTAATGG CACGAT AGTCGCATCGGATCTG
hisS	15	38	AAATAATAACGTGA TGGGAA GCGCCTCG CTTCCCGTG TATGAT TGAACCCGCATGGCTC
htpR-P1	15	38	ACATTACGCCACTT ACGCCT GAATAATA AAAGCGTGT TATACT CTTTCCTGCAATGGTT
htpR-P2	15	39	TTCACAAGCTTGCA TTGAAC TTGTGGATA AAATCACGG TCTGAT AAAACAGTGAATG
htpR-P2	1	28	TTCACA AGCTTGCATTGAACTTGTGGA TAAAAT CACGGTCTGATAAAACAGTGAATG
htpR-P3	15	38	AGCTTGCATTGAAC TTGTGG ATAAAATC ACGGTCTGA TAAAAC AGTGAATGATAACCTCGT
ilvGEDA	15	38	GCCAAAAAATATCT TGTACT ATTTACAA AACCTATGG TAACTC TTTAGGCATTCCTTCGA
ilvGEDA	14	38	GCCAAAAAATATC TTGTAC TATTTACAA AACCTATGG TAACTC TTTAGGCATTCCTTCGA
ilvGEDA	22	46	GCCAAAAAATATCTTGTACTA TTTACA AAACCTATG GTAACTCTT TAGGCA TTCCTTCGA
ilvIH-P1	14	37	CTCTGGCTGCCAA TTGCTT AAGCAAGA TCGGACGGT TAATGT GTTTTACACATTTTTTC
ilvIH-P2	15	38	GAGGATTTTATCGT TTCTTT TCACCTTT CCTCCTGTT TATTCT TATTACCCCGTGT
ilvIH-P2	15	38	GAGGATTTTATCGTTTC TTTTCA CCTTTCCT CCTGTTTAT TCTTAT TACCCCGTGT
ilvIH-P3	18	41	ATTTTAGGATTAA TTAAAA AAATAGAG AAATTGCTG TAAGTT GTGGGATTCAGCCGATT
ilvIH-P4	15	38	TGTAGAATTTTATT CTGAAT GTCTGGGC TCTCTATTT TAGGAT TAATTAAAAAAATAGAG
ISlins-PL	15	37	CGAGGCCGGTGATG CTGCCA ACTTACTG ATTTAGTG TATGAT GGTGTTTTTGAGGTGCT
ISlins-PR	13	36	ATATATACCTTA TGGTAA TGACTCCA ACTTATTGA TAGTGT TTTATGTTCAGATAAT
ISlins-PR	18	41	ATATATACCTTATGGTA ATGACT CCAACTTA TTGATAGTG TTTTAT GTTCAGATAAT
IS2I-II	7	30	GATGTC TGGAAA TATAGGGG CAAATCCAC TAGTAT TAAGACTATCACTTATT
lacI	15	38	GACACCATCGAATG GCGCAA AACCTTTC GCGGTATGG CATGAT AGCGCCCGGAAGAGAGT
lacP1	15	39	TAGGCACCCCAGGC TTTACA CTTTATGCT TCCGGCTCG TATGTT GTGTGGAATTGTGAGC
lacP115	14	37	TTTACACTTTATG CTTCCG GCTCGTAT GTTGTGTGG TATTGT GAGCGGATAACAATTT
lacP115	15	37	TTTACACTTTATGC TTCCGG CTCGTAT GTTGTGTGG TATTGT GAGCGGATAACAATTT
lacP115	1	25	TTTACA CTTTATGC TTCCGGCTCG TATGTT GTGTGGTATTGTGAGCGGATAACAATTT
lacP2	15	38	AATGTGAGTTAGCT CACTCA TTAGGCAC CCCAGGCTT TACACT TTATGCTTCCGGCTCG
lacP2	13	38	AATGTGAGTTAG CTCACT CATTAGGCA CCCCAGGCTT TACACT TTATGCTTCCGGCTCG
lacP2	22	45	AATGTGAGTTAGCTCACTCAT TAGGCA CCCCAGGC TTTACACTT TATGCT TCCGGCTCG
lambdac17	15	38	GGTGTATGCATTTA TTTGCA TACATTCA ATCAATTGT TATAAT TGTTATCTAAGGAAAT
lambdacin	15	38	TAGATAACAATTGA TTGAAT GTATGCAA ATAAATGCA TACACT ATAGGTGTGGTTTAAT
lambdaL57	14	37	TGATAAGCAATGC TTTTTT ATAATGCC AACTTAGTA TAAAAT AGCCAACCTGTTCGACA
lambdaPI	15	38	CGGTTTTTTCTTGC GTGTAA TTGCGGAG ACTTTGCGA TGTACT TGACACTTCAGGAGTG
lambdaPL	15	38	TATCTCTGGCGGTG TTGACA TAAATACC ACTGGCGGT GATACT GAGCACATCAGCAGGA
lambdaPo	15	38	TACCTCTGCCGAAG TTGAGT ATTTTTGC TGTATTTGT CATAAT GACTCCTGTTGATAGAT
lambdaPR	15	38	TAACACCGTGCGTG TTGACT ATTTTACC TCTGGCGGT GATAAT GGTTGCATGTACTAAG
lambdaPR'	15	38	TTAACGGCATGATA TTGACT TATTGAAT AAAATTGGG TAAATT TGACTCAACGATGGGTT
lambdaPRE	15	39	GAGCCTCGTTGCGT TTGTTT GCACGAACC ATATGTAAG TATTTC CTTAGATAACAAT
lambdaPRE	13	36	GAGCCTCGTTGC GTTTGT TTGCACGAA CCATATGT AAGTAT TTCCTTAGATAACAAT
lambdaPRM	15	38	AACACGCACGGTGT TAGATA TTTATCCC TTGCGGTGA TAGATT TAACGTATGAGCACAA
lep	15	37	TCCTCGCCTCAATG TTGTAG TGTAGAAT GCGGCGTT TCTATT AATACAGACGTTAAT
leu	2	25	G TTGACA TCCGTTTT TGTATCCAG TAACTC TAAAAGCATATCGCATT
leultRNA	15	37	TCGATAATTAACTA TTGACG AAAAGCTG AAAACCAC TAGAAT GCGCCTCCGTGGTAGCA
lex	15	38	TGTGCAGTTTATGG TTCCAA AATCGCCT TTTGCTGTA TATACT CACAGCATAACTGTAT
livJ	15	38	TGTCAAAATAGCTA TTCCAA TATCATAA AAATCGGGA TATGTT TTAGCAGAGTATGCT
lpd	7	30	TTGTTG TTTAAA AATTGTTA ACAATTTTG TAAAAT ACCGACGGATAGAACGA
lpp	15	38	CCATCAAAAAAATA TTCTCA ACATAAAA AACTTTGTG TAATAC TTGTAACGCTACATGGA
lpp	15	39	CCATCAAAAAAATA TTCTCA ACATAAAA AACTTTGTGT AATACT TGTAACGCTACATGGA
lpp/P1	13	37	ATCAAAAAAATA TTCTCA ACATAAAAA ACTTTGTGT TATACT TGTAACGCTACATGGA
lpp/P2	13	37	ATCAAAAAAATA TTCTCA ACATAAAAA ACTTTGTGT TATAAT TGTAACGCTACATGGA
lpp/R1	13	36	ATCAAAAAAATA TTCACA ACATAAAA AACTTTGT GTAATA CTTGTAACGCTACATGGA
lpp/R1	15	37	ATCAAAAAAATA TTCACA ACATAAAA AACTTTGTG TAATAC TTGTAACGCTACATGGA
Mlrna	15	38	ATGCGCAACGCGGG GTGACA AGGGCGCG CAAACCCTC TATACT GCGCGCCGAAGCTGACC
mac11	14	38	CCCCCGCAGGGAT GAGGAA GGTGGTCGA CCGGGCTCG TATGTT GTGTGGAATTGTGAGC
mac12	14	38	CCCCCGCAGGGAT GAGGAA GGTCGGTCG ACCGGCTCG TATGTT GTGTGGAATTGTGAGC
mac21	14	38	CCCCCGCAGGGAT GAGGAA GGTCGACCT TCCGGCTCG TATGTT GTGTGGAATTGTGAGC
mac3	14	37	CCCCCGCAGGGAT GAGGAA GGTCGGTC GACCGCTCG TATGTT GTGTGGAATTGTGAGCG
mac31	14	37	CCCCCGCAGGGAT GAGGAA GGTCGGTC GACCGCTCG TATATT GTGTGGAATTGTGAGCG
malEFG	15	37	AGGGGCAAGGAGGA TGGAAA GAGGTTGC CGTATAAA GAAACT AGAGTCCGTTTAGGTGT
malK	15	37	CAGGGGGTGGAGGA TTTAAG CCATCTCC TGATGACG CATAGT CAGCCCATCATGAATG

Tabla A.3: Descripción de los promotores de la base de datos utilizada en la experimentación.

secuencia	TTGACA	TATAAT	Promotor
malPQ	15	38	ATCCCCGCAGGATG AGGAAG GTCAACAT CGAGCCTGG CAAACT AGCGATAACGTTGTGT
malPQ/A516P1	12	34	ATCCCCGCAGG ATGAGG AGCCTGGC AAACTAGC GATGAT AACGTTGTGTTGAA
malPQ/A516P2	15	39	ATCCCCGCAGGAGG ATGAGG AGCCTGGCA AACTAGCGA TAACGT TGTGTTGAAAA
malPQ/A517/A	15	37	CCCCGCAGGATGAG GTCGAG CCTGGCAA ACTAGCGA TAACGT TGTGTTGAAAA
malPQ/Pp12	14	37	ATCCCCGCAGGAT GAGGAA GGTCAACA TCGAGCCTG GAAAAC TAGCGATAACGTTGTGT
malPQ/Pp12	14	38	ATCCCCGCAGGAT GAGGAA GGTCAACA TCGAGCCTGG AAAACT AGCGATAACGTTGTGT
malPQ/Pp13	14	38	ATCCCCGCAGGAT TAGGAA GGTCAACAT CGAGCCTGG CAAACT AGCGATAACGTTGTGT
malPQ/Pp13	15	38	ATCCCCGCAGGATT AGGAAG GTCAACAT CGAGCCTGG CAAACT AGCGATAACGTTGTGT
malPQ/Pp14	14	37	ATCCCCGCAGGAT GAGGAA GGTCAACA TCGAGCCTG GAAACT AGCGATAACGTTGTGT
malPQ/Pp15	14	38	ATCCCCGCAGGAT GAGAAA GGTCAACAT CGAGCCTGG CAAACT AGCGATAACGTTGTGT
malPQ/Pp16	15	38	ATCCCCGCAGGATA AGGAAG GTCAACAT CGAGCCTGG CAAACT AGCGATAACGTTGTGT
malPQ/Pp18	15	38	ATCCCCGCAGGATG GGGAAG GTCAACAT CGAGCCTGG CAAACT AGCGATAACGTTGTGT
malT	15	37	GTCATCGCTTGCAT TAGAAA GGTTTCTG GCCGACCT TATAAC CATTAATTACG
malT	9	32	GTCATCGC TTGCAT TAGAAAGGT TTCTGGCC GACCTT ATAACCATTAATTACG
manA	15	38	CGGCTCCAGGTTAC TTCCCG TAGGATTC TTGCTTTAA TAGTGG GATTAATTTCCACATTA
manA	10	33	CGGCTCCAG GTTACT TCCCGTAG GATTCTTGC TTTAAT AGTGGGATTAATTTCCACATTA
metA-P1	15	38	TTCAACATGCAGGC TCGACA TTGGCAAA TTTTCTGGT TATCTT CAGCTATCTGGATGT
metA-P2	15	38	AAGACTAATTACCA TTTTCT CTCCTTTT AGTCATTCT TATATT CTAACGTAGTCTTTTCC
metA-P2	17	40	AAGACTAATTACCATT TTCTCT CCTTTTAG TCATTCTTA TATTCT AACGTAGTCTTTTCC
metBL	12	35	TTACCGTGACA TCGTGT AATGCACC TGTCGGCGT GATAAT GCATATAATTTTAACGG
metBL	20	44	TTACCGTGACATCGTGTAA TGCACC TGTCGGCGT GATAATGCA TATAAT TTTAACGG
metF	8	31	TTTTCGG TTGACG CCCTTCGG CTTTTCCTT CATCTT TACATCTGGACG
micF	15	37	GCGGAATGGCGAAA TAAGCA CCTAACAT CAAGCAAT AATAAT TCAAGGTTAAAATCAAT
micF	27	50	GCGGAATGGCGAAATAAGCACCTAAC ATCAAG CAATAATA ATTCAAGGT TAAAAT CAAT
motA	15	39	GCCCCAATCGCGCG TTAACG CCTGACGAC TGAACATCC TGTCAT GGTCAACAGTGGA
MuPc-1	6	33	AAATT TTGAAA AGTAACTTTATAGAAAAGAAT AATACT GAAAAGTCAATTTGGTG
MuPc-1	6	30	AAATT TTGAAA AGTAACTTT ATAGAAAAG AATAAT ACTGAAAAGTCAATTTGGTG
MuPc-1	18	40	AAATTTTGAAAAGTAAC TTTATA GAAAAGAA TAATACTG AAAAGT CAATTTGGTG
MuPc-2	9	32	GGAACACA TTTAAA AACCCTCC TAAGTTTTG TAATCT ATAAAGTTAGCAATTTA
MuPc-2	9	30	GGAACACA TTTAAA AACCCTC CTAAGTTT TGTAAT CTATAAAGTTAGCAATTTA
MuPe	15	38	TACCAAAAAGCACC TTTACA TTAAGCTT TTCAGTAAT TATCTT TTTAGTAAGCTAGCTA
NR1rnaC	15	39	GTCACAATTCTCAA GTCGCT GATTTCAAA AAACTGTAG TATCCT CTGCGAAACGATCCCT
NR1rnaC	16	39	GTCACAATTCTCAAG TCGCTG ATTTCAAA AAACTGTAG TATCCT CTGCGAAACGATCCCT
NR1rnaC/m	15	38	TCACAATTCTCAAG TTGCTG ATTTCAAA AAACTGTAG TATCCT CTGCGAAACGATCCCT
NVP1rna100	11	35	GGAGTTTGTC TTGAAG TTATGCACC TGTTAAGGC TAAACT GAAAGAACAGATTTTGT
nusA	7	30	CAGTAT TTGCAT TTTTTACC CAAAACGAG TAGAAT TTGCCACGTTTCAGGCG
ompA	12	34	GCCTGACGGAG TTCACA CTTGTAAG TTTTCAAC TACGTT GTAGACTTTAC
ompA	13	34	GCCTGACGGAGT TCACAC TTGTAAG TTTTCAAC TACGTT GTAGACTTTAC
ompA	19	41	GCCTGACGGAGTTCACAC TTGTAA GTTTTCAA CTACGTTG TAGACT TTAC
ompC	15	38	GTATCATATTCGTG TTGGAT TATTCTGC ATTTTTGGG GAGAAT GGACTTGCCGACTG
ompF	7	30	GGTAGG TAGCGA AACGTTAG TTTGAATGG AAAGAT GCCTGCAGACACATAAA
ompF	22	45	GGTAGGTAGCGAAACGTTAGT TTGAAT GGAAAGAT GCCTGCAGA CACATA AA
ompF/pKI217	3	26	GG TAGCGA AACGTTAG TTTGCAAGC TTTAAT GCGGTAGTTTATCAC
ompF/pKI217	18	41	GGTAGCGAAACGTTAGT TTGCAA GCTTTAAT GCGGTAGTT TATCAC
ompR	15	36	TTTCGCCGAATAAA TTGTAT ACTTAAG CTGCTGTT TAATAT GCTTTGTAACAATTT
ompR	15	39	TTTCGCCGAATAAA TTGTAT ACTTAAGCTG CTGTTTAA TATGCT TTGTAACAATTT
p15primer	15	38	ATAAGATGATCTTC TTGAGA TCGTTTTG GTCTGCGCG TAATCT CTTGCTTGAAAACGAAA
p15rnaI	15	39	TAGAGGAGTTAGTC TTGAAG TCATGCGCC GGTTAAGGC TAAACT GAAAGGACAAGTTTTG
P22ant	15	38	TCCAAGTTAGTGTA TTGACA TGATAGAA GCACTCTAC TATATT CTCAATAGGTCCACGG
P22mnt	15	38	CCACCGTGGACCTA TTGAGA ATATAGTA GAGTGCTTC TATCAT GTCAATACACTAACTT
P22PR	15	37	CATCTTAAATAAAC TTGACT AAAGATTC CTTTAGTA GATAAT TTAAGTGTTCTTTAAT
P22PRM	9	32	AAATTATC TACTAA AGGAATCT TTAGTCAAG TTTATT TAAGATGACTTAACTAT
P22PRM	15	38	AAATTATCTACTAA AGGAAT CTTTAGTC AAGTTTATT TAAGAT GACTTAACTAT
P22PRM	24	47	AAATTATCTACTAAAGGAATCTT TAGTCA AGTTTATT TAAGATGAC TTAACT AT
pBR313Htet	12	35	AATTCTCATGT TTGACA GCTTATCA TCGATAAGC TAGCTT TAATGCGGTAGTTTAT
pBR322bla	15	38	TTTTTCTAAATACA TTCAAA TATGTATC CGCTCATGA GACAAT AACCCTGATAAATGCT
pBR322P4	15	42	CATCTGTGCGGTAT TTCACA CCGCATATGGTGCACTCTCAG TACAAT CTGCTCTGATGCCGCAT
pBR322primer	15	38	ATCAAAGGATCTTC TTGAGA TCCTTTTT TTCTGCGCG TAATCT GCTGCTTGCAAACAAAA
pBR322tet	15	38	AAGAATTCTCATGT TTGACA GCTTATCA TCGATAAGC TTTAAT GCGGTAGTTTATCACA
pBRH4-25	4	27	TCG TTTTCA AGAATTCA TTAATGCGG TAGTTT ATCACAGTTAA
pBRP1	15	42	TTCATACACGGTGC CTGACT GCGTTAGCAATTTAACTGTGA TAAACT ACCGCATTAAAGCTTA
pBRRNAI	15	39	GTGCTACAGAGTTC TTGAAG TGGTGGCCT AACTACGGC TACACT AGAAGGACAGTATTTG
pBRtet-10	15	38	AAGAATTCTCATGT TTGACA GCTTATCA TCGATGCGG TAGTTT ATCACAGTTAA
pBRtet-15	15	38	AAGAATTCTCATGT TTGACA GCTTATCA TCGGTAGTT TATCAC AGTTAAATTGC
pBRtet-22	15	39	AAGAATTCTCATGT TTGACA GCTTATCAT CGATCACAG TTAAAT TGCTAACGCAG
pBRtet/TA22	10	33	TTCTCATGT TTGACA GCTTATCA TCGATAAGC TAAATT TTATATAAAATTTAGCT
pBRtet/TA33	10	33	TTCTCATGT TTGACA GCTTATCA TCGATAAGC TAAATT TATATAAAATTTTATAT
pColViron-P1	15	38	TCACAATTCTCAAG TTGATA ATGAGAAT CATTATTGA CATAAT TGTTATTATTTTAC
pColViron-P2	13	35	TGTTTCAACACC ATGTAT TAATTGTG TTTATTTG TAAAAT TAATTTTCTGACAATAA
pEG3503	6	30	CTGGC TGGACT TCGAATTCA TTAATGCGG TAGTTT ATCACAGTTAA
phiXA	15	38	AATAACCGTCAGGA TTGACA CCCTCCCA ATTGTATGT TTTCAT GCCTCCAAATCTTGGA
phiXB	15	39	GCCAGTTAAATAGC TTGCAA AATACGTGG CCTTATGGT TACAGT ATGCCCATCGCAGTT
phiXD	15	39	TAGAGATTCTCTTG TTGACA TTTTAAAAG AGCGTGGAT TACTAT CTGAGTCCGATGCTGTT
pori-I	15	38	CTGTTGTTCAGTTT TTGAGT TGTGTATA ACCCCTCAT TCTGAT CCCAGCTTATACGGT
pori-r	15	39	GATCGCACGATCTG TATACT TATTTGAGT AAATTAACC CACGAT CCCAGCCATTCTTCTGC
ppc	15	38	CGATTTCGCAGCAT TTGACG TCACCGCT TTTACGTGG CTTTAT AAAAGACGACGAAAA
pSC101oriP1	3	30	TT TTGTAG AGGAGCAAACAGCGTTTGCGA CATCCT TTTGTAATACTGCGGAA
pSC101oriP2	8	30	ATTATCA TTGACT AGCCCATC TCAATTGG TATAGT GATTAAAATCACCTAGA
pSC101oriP3	15	38	ATACGCTCAGATGA TGAACA TCAGTAGG GAAAATGCT TATGGT GTATTAGCTAAAGC
pyrB1-P1	15	37	CTTTCACACTCCGC CCTATA AGTCGGAT GAATGGAA TAAAAT GCATATCTGATTGCGTG
pyrB1-P1	3	26	CT TTCACA CTCCGCCC TATAAGTCG GATGAA TGGAATAAAATGCATATCTGATTGCGTG
pyrB1-P2	13	36	TTGCATCAAATG CTTGCG CCGCTTCT GACGATGAG TATAAT GCCGGACAATTTGCCGG

Tabla A.4: Descripción de los promotores de la base de datos utilizada en la experimentación.

secuencia	TTGACA	TATAAT	Promotor
pyrD	15	38	TTGCCGCAGGTCAA TTCCCT TTTGGTCC GAACTCGCA CATAAT ACGCCCCCGGTTTG
pyrE-P1	15	38	ATGCCTTGTAAGGA TAGGAA TAACCGCC GGAAGTCCG TATAAT GCGCAGCCACATTTG
pyrE-P2	14	38	GTAGGCGGTCATA CTGCGG ATCATAGAC GTTCCTGTT TATAAA AGGAGAGGTGGAAGG
R100rna3	15	39	GTACCGGCTTACGC CGGGCT TCGGCGGTT TTACTCCTG TATCAT ATGAAACAACAGAG
R100RNAI	15	38	CACAGAAAGAAGTC TTGAAC TTTTCCGG GCATATAAC TATACT CCCCGCATAGCTGAAT
R100RNAII	15	38	ATGGGCTTACATTC TTGAGT GTTCAGAA GATTAGTGC TAGATT ACTGATCGTTTAAGGAA
R1RNAII	15	37	ACTAAAGTAAAGAC TTTACT TTGTGGCG TAGCATGC TAGATT ACTGATCGTTTAAGGAA
recA	15	37	TTTCTACAAAACAC TTGATA CTGTATGA GCATACAG TATAAT TGCTTCAACAGAACAT
rnh	15	38	GTAAGCGGTCATTT ATGTCA GACTTGTC GTTTTACAG TTCGAT TCAATTACAGGA
rnh	12	33	GTAAGCGGTCA TTTATG TCAGACT TGTCGTTT TACAGT TCGATTCAATTACAGGA
rn(pRNaseP)	15	38	ATGCGCAACGCGGG GTGACA AGGGCGCG CAAACCCTC TATACT GCGCGCCGAAGCTGACC
rp1J	15	38	TGTAAACTAATGCC TTTACG TGGGCGGT GATTTTGTC TACAAT CTTACCCCCACGTATA
rpmH1p	15	38	GATCCAGGACGATC CTTGCG CTTTACCC ATCAGCCCG TATAAT CCTCCACCCGGCGCG
rpmH1p	16	38	GATCCAGGACGATCC TTGCGC TTTACCC ATCAGCCCG TATAAT CCTCCACCCGGCGCG
rpmH2p	15	38	ATAAGGAAAGAGAA TTGACT CCGGAGTG TACAATTAT TACAAT CCGGCCTCTTTAATC
rpmH3p	15	38	AAATTTAATGACCA TAGACA AAAATTGG CTTAATCGA TCTAAT AAAGATCCCAGGACG
rpoA	15	38	TTCGCATATTTTTC TTGCAA AGTTGGGT TGAGCTGGC TAGATT AGCCAGCCAATCTTT
rpoB	15	37	CGACTTAATATACT GCGACA GGACGTCC GTTCTGTG TAAATC GCAATGAAATGGTTTAA
rpoD-Pa	13	36	CGCCCTGTTCCG CAGCTA AAACGCAC GACCATGCG TATACT TATAGGGTTGC
rpoD-Pb	9	33	AGCCAGGT CTGACC ACCGGGCAA CTTTTAGAG CACTAT CGTGGTACAAAT
rpoD-Pb	12	36	AGCCAGGTCTG ACCACC GGGCAACTT TTAGAGCAC TATCGT GGTACAAAT
rpoD-Phs	13	36	ATGCTGCCACCC TTGAAA AACTGTCG ATGTGGGAC GATATA GCAGATAAGAA
rpoD-Phs/min	4	31	CCC TTGAAA AACTGTCGATGTGGGACGATA TAGCAG ATAAGAATATTGCT
rpoD-Phs/min	4	27	CCC TTGAAA AACTGTCGA TGTGGGAC GATATA GCAGATAAGAATATTGCT
rpoD-Phs/min	12	36	CCCTTGAAAAA CTGTCG ATGT GGGAC GATATA GCAGATAAGAATATTGCT
rrn4.5S	14	37	GGCACGCGATGGG TTGCAA TTAGCCGG GGCAGCAGT GATAAT GCGCCTGCGCGTTGGTT
rrnABP1	15	37	TTTTAAATTTCCTC TTGTCA GGCCGGAA TAACTCCC TATAAT GCGCCACCACTGACACG
rrnABP2	15	37	GCAAAAATAAATGC TTGACT CTGTAGCG GGAAGGCG TATTAT GCACACCCCGCGCCGC
rrnB-P3	14	40	CTATGATAAGGAT TACTCA TCTTATCCTT ATCAAACCGT TAAAAT GGGCGGTGTGAGCTTG
rrnB-P4	15	36	GCGTATCCGGTCAC CTCTCA CCTGACA GTTCGTGG TAAAAT AGCCAACCTGTTCGACA
rrnDEXP2	15	37	CCTGAAATTCAGGG TTGACT CTGAAAGA GGAAAGCG TAATAT ACGCCACCTCGCGACAG
rrnD-P1	15	37	GATCAAAAAAAATAC TTGTGC AAAAAATT GGGATCCC TATAAT GCGCCTCCGTTGAGACG
rrnE-P1	15	37	CTGCAATTTTTCTA TTGCGG CCTGCGGA GAACTCCC TATAAT GCGCCTCCATCGACACG
rrnG-P1	15	37	TTTATATTTTTCGC TTGTCA GGCCGGAA TAACTCCC TATAAT GCGCCACCACTGACACG
rrnG-P2	15	37	AAGCAAAGAAATGC TTGACT CTGTAGCG GGAAGGCG TATTAT GCACACCGCCGCGCCG
rrnX1	15	37	ATGCATTTTTCCGC TTGTCT TCCTGAGC CGACTCCC TATAAT GCGCCTCCATCGACACG
RSFprimer	15	38	GGAATAGCTGTTCG TTGACT TGATAGAC CGATTGATT CATCAT CTCATAAATAAAGAA
RSFrnaI	15	39	TAGAGGAGTTTGTC TTGAAG TTATGCACC TGTTAAGGC TAAACT GAAAGAACAGATTTTG
S10	15	37	TACTAGCAATACGC TTGCGT TCGGTGGT TAAGTATG TATAAT GCGCCGGCTTGTCGT
sdh-P1	14	37	ATATGTAGGTTAA TTGTAA TGATTTTG TGAACAGCC TATACT GCCGCCAGTCTCCGGAA
sdh-P2	15	37	AGCTTCCGCGATTA TGGGCA GCTTCTTC GTCAAATT TATCAT GTGGGGCATCCTTACCG
spc	15	38	CCGTTTATTTTTTC TACCCA TATCCTTG AAGCGGTGT TATAAT GCCGCGCCCTCGATA
spot42r	15	37	TTACAAAAAGTGCT TTCTGA ACTGAACA AAAAAGAG TAAAGT TAGTCGCGTAGGGTACA
spot42r	14	37	TTACAAAAAGTGC TTTCTG AACTGAACA AAAAAGAG TAAAGT TAGTCGCGTAGGGTACA
ssb	15	39	TAGTAAAAGCGCTA TTGGTA ATGGTACAA TCGCGCGTT TACACT TATTCAGAACGATTTT
str	15	38	TCGTTGTATATTTC TTGACA CCTTTTCG GCATCGCCC TAAAAT TCGGCGTCCTCATAT
sucAB	15	39	AAATGCAGGAAATC TTTAAA AACTGCCCC TGACACTAA GACAGT TTTAAAAGGTTCCTT
supB-E	15	38	CCTTGAAAAAGAGG TTGACG CTGCAAGG CTCTATACG CATAAT GCGCCCCGCAACGCCGA
T7-A1	15	38	TATCAAAAAGAGTA TTGACT TAAAGTCT AACCTATAG GATACT TACAGCCATCGAGAGGG
T7-A3	15	38	GTGAAACAAAACGG TTGACA ACATGAAG TAAACACGG TACGAT GTACCACATGAAACGAC
T7-C	15	38	CATTGATAAGCAAC TTGACG CAATGTTA ATGGGCTGA TAGTCT TATCTTACAGGTCATC
T7-D	15	38	CTTTAAGATAGGCG TTGACT TGATGGGT CTTTAGGTG TAGGCT TTAGGTGTTGGCTTTA
T7A2	15	39	ACGAAAAACAGGTA TTGACA ACATGAAGT AACATGCAG TAAGAT ACAAATCGCTAGGTAAC
T7E	11	34	CTTACGGATG ATGATA TTTACACA TTACAGTGA TATACT CAAGGCCACTACAGATA
TAC16	10	32	AATGAGCTG TTGACA ATTAATCA TCGGCTCG TATAAT GTGTGGAATTGTG
Tn10Pin	9	33	TCATTAAG TTAAGG TGGATACAC ATCTTGTCA TATGAT CAAATGGTTTCGCGAAA
Tn10Pin	15	38	TCATTAAGTTAAGG TGGATA CACATCTT GTCATATGA TCAAAT GGTTTCGGCGAAA
Tn10Pout	15	38	AGTGTAATTCGGGG CAGAAT TGGTAAAG AGAGTCGTG TAAAAT ATCGAGTTCGCACATC
Tn10tetA	15	39	ATTCCTAATTTTTG TTGACA CTCTATCAT TGATAGAGT TATTTT ACCACTCCCTATCAGT
Tn10tetR	15	39	TATTCATTTCACTT TTCTCT ATCACTGAT AGGGAGTGG TAAAAT AACTCTATCAATGATA
Tn10tetR*	11	34	TGATAGGGAG TGGTAA AATAACTC TATCAATGA TAGAGT GTCAACAAAAATTAGG
Tn10xxxP1	15	37	TTAAAATTTTCTTG TTGATG ATTTTTAT TTCCATGA TAGATT TAAAATAACATACC
Tn10xxxP2	15	38	AAATGTTCTTAAGA TTGTCA CGACCACA TCATCATGA TACCAT AAACATACTGACGG
Tn10xxxP3	11	38	CCATGATAGA TTTAAA ATAACATACCGTCAGTATGTT TATGGT ATCATGATGATGTGGTC
Tn10xxxP3	19	43	CCATGATAGATTTAAAAT AACATA CCGTCAGTA TGTTTATGG TATCAT GATGATGTGGTC
Tn2660bla-P3	15	38	TTTTTCTAAATACA TTCAAA TATGTATC CGCTCATGA GACAAT AACCCTGATAAATGCT
Tn2661bla-Pa	15	38	GGTTTATAAAATTC TTGAAG ACGAAAGG GCCTCGTGA TACGCT TATTTTTATAGGTTAA
Tn2661bla-Pb	5	28	CCTC GTGATA CGCTTATT TTTATAGGT TAATGT CATGATAATAATGGTTT
Tn501mer	14	39	TTTTCCATATCGC TTGACT CCGTACATG AGTACGGAAG TAAGGT TACGCTATCCAATTTC
Tn501merR	15	37	CATGCGCTTGTCCT TTCGAA TTGAAATT GGATAGCG TAACCT TACTTCCGTACTCA
Tn501merR	8	32	CATGCGC TTGTCC TTTCGAATT GAAATTGGA TAGCGT AACCTTACTTCCGTACTCA
Tn5TR	15	38	TCCAGGATCTGATC TTCCAT GTGACCTC CTAACATGG TAACGT TCATGATAACTTCTGCT
Tn5neo	15	38	CAAGCGAACCGGAA TTGCCA GCTGGGGC GCCCTCTGG TAAGGT TGGGAAGCCCTGCAA
Tn7-PLE	15	38	ACTAGACAGAATAG TTGTAA ACTGAAAT CAGTCCAGT TATGCT GTGAAAAAGCAT
tnaA	15	37	AAACAATTTCAGAA TAGACA AAAACTCT GAGTGTAA TAATGT AGCCTCGTGTCTTGCG
tonB	15	39	ATCGTCTTGCCTTA TTGAAT ATGATTGCT ATTTGCATT TAAAAT CGAGACCTGGTTT
trfA	15	39	AGCCGCTAAAGTTC TTGACA GCGGAACCA ATGTTTAGC TAAACT AGAGTCTCCTT
trfB	15	38	AGCGGCTAAAGGTG TTGACG TGCGAGAA ATGTTTAGC TAAACT TCTCTCATGTG
trp	15	38	TCTGAAATGAGCTG TTGACA ATTAATCA TCGAACTAG TTAACT AGTACGCAAGTTCACGT
trpP2	15	38	ACCGGAAGAAAACC GTGACA TTTTAACA CGTTTGTTA CAAGGT AAAGGCGACGCCGCCC
trpR	15	39	TGGGGACGTCGTTA CTGATC CGCACGTTT ATGATATGC TATCGT ACTCTTTAGCGAGTACA
trpR	11	34	TGGGGACGTC GTTACT GATCCGCAC GTTTATGA TATGCT ATCGTACTCTTTAGCGAGTACA

Tabla A.5: Descripción de los promotores de la base de datos utilizada en la experimentación.

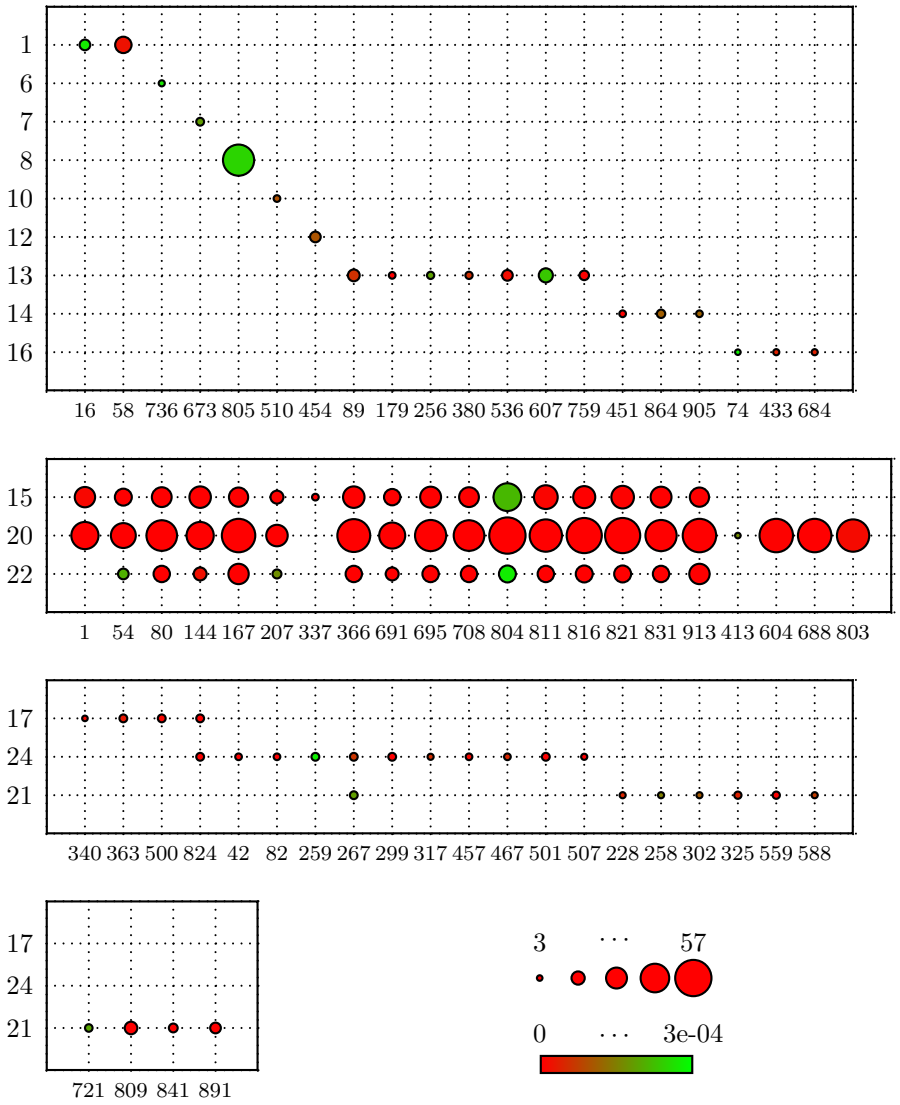


Figura A.1: Intersección de ClustersA y ClustersB para CC-EMO. El gráfico representa cada intersección como un círculo, cuyo tamaño crece con la cantidad de elementos en la intersección entre los clusters, mientras que el color muestra el p-value de la intersección.

Bibliografía

[1] R. Agrawal, T. Imielinski, y A. Swami. Mining association rules between sets of items in large databases. En P. Buneman y S. Jajodia, editores, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, páginas 207–216, Washington, D.C., 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H.Toivonen, y A.I. Verkamo. Fast discovery of association rules. En *Advances in Knowledge Discovery and Data Mining*, páginas 307–328. AAAI/MIT Press, 1996.

[3] A.V. Aho, J.E. Hopcroft, y J.D. Ullman. *Data Structures and Algorithms*. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley, 1982.

[4] F. Al-Shahrour, R. Díaz-Uriarte, y J. Dopazo. Fatigo: a web tool for finding significant associations of gene ontology terms with groups of genes. *Bioinformatics*, 20:578–580, 2004.

[5] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, y P. Walter. *Biología molecular de la célula. Cuarta Edición*. Omega, 2003.

[6] C. Alfarano, C.E. Andrade, K. Anthony, N. Bahroos, M. Bajec, K. Bantoft, D. Betel, B B. obechko, K. Boutilier, E. Burgess, K. Buzadzija, R. Cavero, C. D'Abreo, I. Donaldson, Dorair D. ajoo, M.J. Dumontier, M.R. Dumontier, V. Earles, R. Farrall, H. Feldman, E. Garderman, Gon Y. g, R. Gonzaga, V. Grytsan, E. Gryz, V. Gu, E. Haldorsen, A. Halupa, R. Haw, Hrvojic A., L. Hurrell, R. Isserlin, F. Jack, F. Juma, A. Khan, T. Kon, S. Konopinsky, V. Le, Lee E., S. Ling, M. Magidin, J. Moniakis, J. Montojo, S. Moore, B. Muskat, I.Ñg, Paraiso J.P., B. Parker, G. Pintilie, R. Pirone, J.J. Salama, S. Sgro, T. Shan, Y. Shu, J. Siew, Sk D. inner, K. Snyder, R. Stasiuk, D. Strumpf, B. Tuekam, S. Tao, Z. Wang, M. White, Willis R., C. Wolting, S. Wong, A. Wrong, C. Xin, R. Yao,

B. Yates, S. Zhang, K. Zheng, Pawson T., B.F.F. Ouellette, y C.W.V. Hogue. The Biomolecular Interaction Network Database and related tools 2005 update. *Nucl. Acids Res.*, 33(suppl_1):D418–424, 2005.

[7] D. Andre. Learning and upgrading rules for an OCR system using genetic programming. En *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, Orlando, Florida, USA, 27-29 Junio 1994. IEEE Press.

[8] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, y G. Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genet.*, 25:25–29, 2000.

[9] T.K. Attwood y D.J. Parry-Smith. *Introducción a la Bioinformática*. Prentice Hall, 2002.

[10] R. Babuska. *Fuzzy Modeling for Control*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[11] T. Back, D. Fogel, y Z. Michalewicz, editores. *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, 1997.

[12] W. Banzhaf, P. Nordin, R. Keller, y F. Francone. *Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications*. Morgan Kaufmann, Enero 1998.

[13] A. Barnard, A. Wolfe, y S. Busby. Regulation at complex bacterial promoters: how bacteria use different promoter organizations to produce different regulatory outcomes. *Curr Opin Microbiol*, 7:102–108, 2004.

[14] E. Benítez-Bellón, G. Moreno-Hagelsieb, y J. Collado-Vides. Evaluation of thresholds for the detection of binding sites for regulatory proteins in Escherichia coli K12 DNA. *Genome Biology*, 3(3):research0013.1 – 0013.16, 2002.

[15] W.A. Bennage y A.K. Dhingra. Single and multiobjective structural optimization in discrete-continuous variables using simulated annealing. *International Journal for Numerical Methods in Engineering*, 38:2753–2773, 1995.

[16] D.A. Benson, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, y D.L. Wheeler. GenBank. *Nucleics Acids Research*, 31(1):23–27, 2003.

[17] J. Berg, J. Tymoczko, y Lubert L. Stryer. *Bioquímica. Quinta Edición*. Reverte, 2003.

[18] J. Bezdek. Fuzzy clustering. En E. Ruspini, P. Bonissone, y W. Pedrycz, editores, *Handbook of Fuzzy Computation*. Institute of Physics Press, 1998.

[19] J.C. Bezdek y S.K. Pal, editores. *Fuzzy Models for Pattern Recognition*. IEEE Press, New York, 1992.

[20] D.J. Cavicchio. *Adaptive search using simulated evolution*. PhD thesis, University of Michigan, 1970.

[21] A. Charnes y W.W. Cooper. *Management Models and Industrial Applications of Linear Programming Vol. 1*. John Wiley & Sons, New York, 1961.

[22] C. Coello-Coello, D. Van Veldhuizen, y G. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer, 2002.

[23] J.L. Cohon. *Multiobjective Programming and Planning*. Academic Press, New York, 1978.

[24] F.S. Collins, M. Morgan, y A.A. Patrinos. The human genome project: Lessons from large-scale biology. *Science*, 300(5617):286–290, 2003.

[25] D. Cook, L. Holder, S. Su, R. Maglothin, y I. Jonyer. Structural mining of molecular biology data. *IEEE Engineering in Medicine and Biology, special issue on Advances in Genomics*, 4(20):67–74, 2001.

[26] O. Cordón, F. Herrera, F. Hoffmann, y L. Magdalena. *Genetic Fuzzy Systems. Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. Advances in Fuzzy Systems - Applications and Theory. Vol. 19. World Scientific, 2001.

[27] O. Cordón, E. Herrera-Viedma, y M. Luque. Evolutionary learning of boolean queries by multiobjective genetic programming. En *Seventh International Conference on Parallel Problem Solving from Nature PPSN VII 2002*, number 2439 in Lecture Notes in Computer Science, páginas 710–719, Granada, Spain, 2002.

[28] O. Cordón, E. Herrera-Viedma, y M. Luque. Improving the learning of boolean queries by means of a multiobjective IQBE evolutionary algorithm. *Information Processing and Management*, 2005. In press.

[29] V. Cotik, R. Romero-Zaliz, y I. Zwir. A hybrid promoter anaysis methodology for prokaryotic genomes. *Special issue on "Bioinformatics", Fuzzy Sets and Systems*, 152:83–102, 2005.

[30] J.M. Daida, T.F. Bersano-Begey, S.J. Ross, y J.F. Vesecky. Computer-assisted design of image classification algorithms: Dynamic and static fitness evaluations in a scaffolded genetic programming environment. En

John R. Koza, David E. Goldberg, David B. Fogel, y Rick L. Riolo, editores, *Genetic Programming 1996: Proceedings of the First Annual Conference*, páginas 279–284, Stanford University, CA, USA, 28–31 Julio 1996. MIT Press.

[31] C. Darwin. *On The Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.* John Murray, London, 1859.

[32] K. De Jong. *An analysis of the behaviour of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.

[33] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons, Inc., 2001.

[34] K. Deb, A. Pratap, S. Agarwal, y T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[35] G. Der y B.S. Everitt. *A handbook of statistical analyses using SAS.* CHAPMAN-HALL, 1996.

[36] R.O. Duda, P.E. Hart, y D.G. Stork. *Pattern Classification (2nd Edition).* Wiley-Interscience, 2000.

[37] T.E. Dunning y M.W. Davis. Evolutionary algorithms for natural language processing. En John R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1996 Conference Stanford University July 28-31, 1996*, páginas 16–23, Stanford University, CA, USA, 28–31 Julio 1996. Stanford Bookstore.

[38] C.M. Fonseca y P.J. Fleming. On the performance assesment and comparison of stochastic multiobjective optimizers. En *Fourth International Conference on Parallel Problem Solving from Nature (PPSN-IV)*, páginas 584–593, Berlin, Alemania, 1996. Springer.

[39] A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms.* Springer-Verlag, 2002.

[40] A.P. Gasch y M.B. Eisen. Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering. *Genome Biology*, 3, 2002.

[41] S.I. Gass y T.L. Saaty. The computational algorithm for the parametric objective function. *Naval Research Logistics Quarterly*, 2:39, 1955.

[42] D. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning.* Addison-Wesley, 1989.

[43] D. Goldberg y J.J. Richardson. Genetic algorithms with sharing for multimodal function optimization. En *Proceedings Second International Conference on Genetic Algorithm*, páginas 41–49, 1987.

[44] J. Grefenstette. *Genetic algorithms for machine learning*. Kluwer, 1993.

[45] D. Hand, H. Mannila, y P. Smyth. *Principles of Data Mining*. MIT Press, 2001.

[46] S. Handley. Automatic learning of a detector for alpha-helices in protein sequences via genetic programming. En *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, páginas 271–278. Morgan Kaufmann, 1993.

[47] A.E. Hans. Multicriteria optimization for highly accurate systems. *Multicriteria Optimization in Engineering and Sciences*, 19:309–352, 1988.

[48] C. Harley y R. Reynolds. Analysis of e.coli promoter sequences. *Nucleic Acids Research*, 15(5):2343–2361, 1987.

[49] J. Hernández, M. Ramírez, y C. Ferri. *Introducción a la Minería de Datos*. Pearson Prentice Hall, 2004.

[50] G. Hertz y G. Stormo. Identifiying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15:563–577, 1999.

[51] Ingenuity Systems. http://www.ingenuity.com.

[52] A Ishihama. Protein-protein communication within the transcription apparatus. *J Bacteriol*, 175:2483–2489, 1993.

[53] P. Jaccard. The distribution of flora in the alpine zone. *The New Phytologist*, 11(2):37–50, 1912.

[54] I. Jonyer, D. J. Cook, y L. B. Holder. Discovery and evaluation of graph-based hierarchical conceptual clusters. *Journal of Machine Learning Research*, 2:19–43, 2001.

[55] C. Kanz, P. Aldebert, N. Althorpe, W. Baker, A. Baldwin, K. Bates, P. Browne, A. van den Broek, M. Castro, G. Cochrane, K. Duggan, R. Eberhardt, N. Faruque, J. Gamble, F. García-Diez, N. Harte, T. Kulikova, Q. Lin, V. Lombard, R. Lopez, R. Mancuso, M. McHale, F.Ñardone, V. Silventoinen, S. Sobhany, P. Stoehr, M. Tuli, K. Tzouvara, R. Vaughan, D. Wu, W. Zhu, y R. Apweiler. The EMBL Nucleotide Sequence Database. *Nucleics Acids Research*, 33(suppl_1):D29–33, 2005.

[56] G. Klir y T. Folger. *Fuzzy sets, uncertainty, and information*. Prentice-Hall, Inc., 1987.

[57] W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. En *Advances in Knowledge Discovery and Data Mining*, páginas 249–271. MIT Press, 1996.

[58] R. Kohavi y G.H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997.

[59] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[60] F. Kursawe. A variant of evolution strategies for vector optimization. En *PPSN I: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, páginas 193–197, London, UK, 1991. Springer-Verlag.

[61] W. Langdon. *Genetic Programming + Data Structures = Automatic Programming!* Engineering and Computer Science. Kluwer, 1998.

[62] S. Lee y H. Wang. Modified simulated annealing for multiple objective engineering design optimization. *Journal of Intelligent Manufacturing*, 3:101–108, 1992.

[63] B. Lewin. *Genes VII*. Marbán, 2001.

[64] B. Masand. *Advances in Genetic Programming*, capítulo Optimising confidence of text classification by evolution of symbolic expressions, páginas 445–458. MIT Press, 1994.

[65] Z. Michalewicz y D.B. Fogel. *How to solve it: modern heuristics*. Springer-Verlag New York, Inc., New York, NY, USA, 2000.

[66] T. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[67] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report C3P Report 826, Caltech Concurrent Computation Program, 1989.

[68] I. Osman. An introduction to meta-heuristics. *Operational Research Tutorial Papers Series, Annal Conference OR37 - Canterbury*, 1995.

[69] A. Osyczka. An approach to multicriterion optimization problems for engineering design. *Computer Methods in Applied Mechanics and Engineering*, 15:309–333, 1978.

[70] Pathway Assit. http://www.ariadnegenomics.com/products/pathway.html.

[71] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[72] W. Pedrycz, P. Bonissone, y E. Ruspini. *Handbook of fuzzy computation.* Institute of Physics, 1998.

[73] Y. Pilpel, P. Sudarsanam, y G.M. Church. Identifying rgulatory networks by combinatorial analysis of promoter elements. *Nature Genet.*, 29:153–159, 2001.

[74] K. Robison, A.M. McGuire, y G.M. Church. A comprehensive library of DNA-binding site matrices for 55 proteins applied to the complete Escherichia coli K-12 genome. *J Mol Biol*, 284(2):241–54, 1998.

[75] K. Rodriguez-Vazquez, C. Fonseca, y P. Fleming. Multiobjective genetic programming: A nonlinear system identification application. En J. Koza, editor, *Late Breaking Papers at the 1997 Genetic Programming Conference*, páginas 207–212, Stanford University, CA, USA, 13–16 Julio 1997.

[76] R. Romero-Zaliz, O. Cordón, C. Rubio-Escudero, I. Zwir, y J.P. Cobb. A multi-objective evolutionary conceptual clustering methodology for gene annotation from networking databases. Sometido.

[77] R. Romero-Zaliz, I. Zwir, y F. Herrera. Búsqueda dispersa multiobjetivo de promotores en secuencias de ADN. En *MAEB'04 (Tercer congreso español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados)*, páginas 141–147, Córdoba, España, Febrero 2004.

[78] R. Romero-Zaliz, I. Zwir, y E. Ruspini. *Applications of Multi-Objective Evolutionary Algorithms*, capítulo Generalized Analysis of Promoters (GAP): A method for DNA sequence description, páginas 427–450. World Scientific, 2004.

[79] C. Rubio-Escudero, R. Romero-Zaliz, O. Cordón, J.P. Cobb, y I.Zwir. Identifying gene profiles by reverse problem solving: from grouping gene expressions to combining microarray analysis methods, 2005. Sometido.

[80] E. Ruspini. A new approach to clustering. *Information and Control*, 15(1):22–32, 1969.

[81] E. Ruspini. Recent developments in fuzzy clustering. En R. Yager, editor, *Fuzzy Sets and Possibility Theory, Recent Developments*, páginas 133–147. Pergamon Press, 1982.

[82] E. Ruspini y I. Zwir. Automated generation of qualitative representations of complex object by hybrid soft-computing methods. En S. Pal y A. Pal, editores, *Pattern Recognition: From Classical to Modern Approaches*, páginas 453–474, Singapore, 2001. World Scientific Company.

[83] Wrobel S. *Inductive logic programming for knowedge discovery in databases*, páginas 74–99. Springer-Verlag New York, Inc., New York, NY, USA, 2000.

[84] H. Salgado, A. Santos-Zavaleta, S. Gama-Castro, D. Millán-Zárate, E. Díaz-Peredo, F. Sánchez-Solano, E. Pérez-Rueda, C. Bonavides-Martínez, y J. Collado-Vides. RegulonDB (version 3.2): transcriptional regulation and operon organisation in escherichia coli k-12. *Nucleic Acids Research*, 29:72–74, 2001.

[85] J. Setubal y J. Meidanis. *Introduction to computational molecular biology.* PWS Publishing Company, 1997.

[86] N. Srinivas y K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[87] S. Tavazoie, J. Hughes, M. Campbell, R. Cho, y G. Church. Systematic determination of genetic network architecture. *Nature Genet.*, 22(3):281–285, 1999.

[88] A. Ulyanov y G. Stormo. Multi-alphabet consensus algorithm for identification of low specificity protein-DNA interactions. *Nucleic Acids Research*, 23(8):1434–1440, 1995.

[89] M.D. Winfield y E.A. Groisman. Phenotypic differences between Salmonella and Escherichia coli resulting from the disparate regulation of homologous genes. *Proc Natl Acad Sci U S A*, 101:17162–17167, 2004.

[90] M. Wong y K. Leung. *Data Mining Using Grammar-Based Genetic Programming and Applications.* Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[91] K.Y. Yeung y W.L. Ruzzo. Principal component analysis for clustering gene expression data. *Bioinformatics*, 17:763–774, 2001.

[92] L.A. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.

[93] L.A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, 8:119–249, 1975.

[94] L.A. Zadeh. Knowledge representation in fuzzy logic. En R.R. Yager y L.A. Zadeh, editores, *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, páginas 1–25. Kluwer, Boston, 1992.

[95] R. Romero Zaliz, O. Cordón, C. Rubio, y I. Zwir. A multiobjective evolutionary fuzzy system for promoter discovery in *e. coli.* En *I International Workshop on Genetic Fuzzy Systems*, páginas 68–75, Granada, España, Marzo 2005.

[96] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, Noviembre 1999.

[97] E. Zitzler, K. Deb, y L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[98] E. Zitzler y L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Noviembre 1999.

[99] I. Zwir, R. Romero-Zaliz, H. Huang, y E.A. Groisman. Extracting promoter features from genomic datasets: towards an annotation of genome regulatory regions. Sometido.

[100] I. Zwir, D. Shin, A. Kato, K.Ñishino, T. Latifi, F. Solomon, J.M. Hare, H. Huang, y E.A. Groisman. Dissecting the Phop regulatory network of Escherichia coli and Salmonella enterica. *PNAS*, 102(8):2862–2867, 2005.

Índice alfabético