Universidad Pablo de Olavide de Sevilla

Área de Lenguajes y Sistemas Informáticos
Escuela Politécnica Superior

Extracción y validación de biclusters a partir de bases de datos binarios

Tesis Doctoral

Domingo Savio Rodríguez Baena

Sevilla, Marzo de 2012

Universidad Pablo de
Olavide de Sevilla

Extracción y validación de biclusters a partir de bases de datos binarios

MEMORIA QUE PRESENTA
Domingo Savio Rodríguez Baena

PARA OPTAR AL GRADO DE DOCTOR POR LA
UNIVERSIDAD PABLO DE OLAVIDE DE SEVILLA

DIRECTOR
Jesús S. Aguilar-Ruiz

Área de Lenguajes y Sistemas Informáticos
Escuela Politécnica Superior

Marzo de 2012

D. Jesús S. Aguilar Ruiz, profesor Titular de Universidad adscrito al área de Lenguajes y Sistemas Informáticos de la Universidad Pablo de Olavide de Sevilla,

CERTIFICA QUE:

D. Domingo Savio Rodríguez Baena, Ingeniero Superior Informático por la Universidad de Sevilla, ha realizado bajo su supervisión el trabajo de investigación titulado:

EXTRACCIÓN Y VALIDACIÓN DE BICLUSTERS A PARTIR DE BASES DE DATOS BINARIOS

Una vez revisado, autoriza la presentación del mismo como tesis doctoral en la Universidad Pablo de Olavide de Sevilla y estima oportuna su presentación al tribunal que habrá de valorarlo. Dicha tesis ha sido realizada dentro del programa de doctorado: *Tecnología e Ingeniería del Software*, con mención de calidad MCD2005-00261, del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla.

Sevilla, Marzo de 2012.

D. Jesús S. Aguilar Ruiz, profesor adscrito al área de Lenguajes y Sistemas Informáticos de la Universidad Pablo de Olavide de Sevilla, como director de la tesis titulada

EXTRACCIÓN Y VALIDACIÓN DE BICLUSTERS A PARTIR DE BASES DE DATOS BINARIOS,

propone la siguiente composición del tribunal titular, a fin de que la Comisión de Doctorado designe al tribunal encargado de juzgar la tesis doctoral.

PRESIDENTE:	Dr. D. José C. Riquelme Santos
VOCALES:	Dr. D. Jose Carlos Clemente Litrán
	Dr. D. Antonio J. Pérez Pulido
	Dr. D. Jesús Cerquides Bueno
SECRETARIO:	Dr. Alicia Troncoso Lora
SUPLENTES:	Dr. D. Isabel Nepomuceno Chamorro
	Dr. D. Cristina Rubio Escudero

Para Irene, Gloria y Julia

Agradecimientos

Una persona es producto de sus experiencias y de las influencias de otras personas. En mi caso particular, me considero afortunado en ambos aspectos. Con respecto a mi vida profesional, he de hacer mención en primer lugar a mi director de tesis. En un principio, Jesús Aguilar me animó a que cambiara de ámbito profesional y apostara por la vida académica, después me dio toda su confianza contando conmigo para la aventura de la UPO y, finalmente, dirigió esta tesis doctoral, que ha resultado bastante complicada por muchos aspectos. Por todo ello, quiero darle mi más sincero agradecimiento. De él me quedo con el amor al trabajo bien hecho, al detalle y a la rigurosidad. Con Antonio J. Pérez tengo una gran deuda pendiente, ya que fue él quien me dio el último empujón necesario para acometer la última fase de la tesis. Gracias por solicitar mi colaboración, por las reuniones, por las revisiones, por las ideas y por tu compañerismo. No me he de olvidar de Jose Clemente, al que conocimos en Italia, fuimos a visitar a Japón y, si hay suerte, haremos venir de Estados Unidos. Es un honor haber contado con tu apoyo, con tu ejemplo y con tu amistad.

A continuación me he de acordar de todos mis compañeros, empezando por mi grupo de investigación, al que pertenecer es, cada día más, un orgullo. Pepe Riquelme siempre ha sabido conducir a este equipo con mano diestra y sabiduría. No puedo dejar de dedicar unas líneas a mis socios de pasillo en la UPO: Carlos, Roberto, Federico, Norberto, Paco y Jesús. Cuando tratas todos los días con tan buena gente, acabas cruzando la frontera de la amistad con pasmosa facilidad. Ellos me han visto en mis mejores y peores momentos, me han escuchado, me han ayudado y hemos compartido grandes ratos. Gracias de corazón. También he de acordarme del resto de compañeros: Fran, Miguel M., Miguel G., Alicia, Rául, Gualberto, Alfonso, Alejandro, etc. Trabajar con todos ellos es un placer. Continuo con Chari, quien me ha facilitado la vida con el papeleo y con su sonrisa. Y por supuesto no me olvido de Soraya, quien nos ha adoptado a todos con un cariño enorme. Un millón de gracias.

Mi familia es un pilar básico. Mi vida, mi educación, mi sentido de la responsabilidad, el aprecio por el esfuerzo, el deseo de ser una persona buena y honrada. Todo ello se lo debo a mis padres y hermanos. Para ellos, mi amor y mi cariño, al igual que para el resto de mi familia: sobrinos, tíos, primos, suegros, cuñados, etc. Por último, y más importante, nada de esto hubiera sido posible sin que la inmensa felicidad que me producen mi mujer y mis hijas me inundara el corazón cada mañana. Ellas son la más poderosa gasolina y a ellas le dedico esta tesis doctoral. Irene, Gloria y Julia, gracias

por hacerme mejor persona cada día y por aguantarme. Os quiero con todo mi corazón.

Tesis Doctoral parcialmente subvencionada por el Ministerio de Educación y Ciencia con el proyecto TIN2007-68084-C-00

MEC
TIN2007-68084-C-00

Abstract

Binary datasets represent a compact and simple way to store data about the relationships between a group of objects and their possible properties. In the last few years, different biclustering algorithms have been specially developed to be applied to binary datasets. Several approaches based on matrix factorization or divide-and-conquer techniques have been proposed to extract useful biclusters from binary data, and these approaches provide information about the distribution of patterns and intrinsic correlations.
We propose a novel approach to extracting biclusters from binary datasets, *BiBit*. The results obtained from different experiments with synthetic data reveal the excellent performance and the robustness of *BiBit* to density and size of input data. Also, *BiBit* is applied to a central nervous system embryonic tumor gene expression dataset to test the quality of the results. A novel gene expression pre-processing methodology, based on expression level layers, and the selective search performed by *BiBit*, based on a very fast bit-pattern processing technique, provide very satisfactory results in quality and computational cost. The power of biclustering in finding genes involved simultaneously in different cancer processes is also shown. Finally, a comparison with *Bimax*, one of the most cited binary biclustering algorithms, shows that *BiBit* is faster while providing essentially the same results.
Besides, in this work, we introduce a software tool, named *CarGene* (Characterization of Genes), that helps scientists to validate sets of genes using biological knowledge. The integration of huge databases with searching techniques in order to automatically validate results from different sources is a key factor in bioinformatics. Several tools have been developed for analysing gene–enrichment in terms. Most of them are Gene Ontology-based tools, i.e., these analyse gene-enrichment in GO annotations. *CarGene* uses metabolic pathways stored in the *Kyoto Encyclopedia of Genes and Genomes* (Kegg) and provides a friendly graphical environment to analyse and compare results generated by different clustering and/or biclustering techniques. *CarGene* is based on the degree of coherence of genes in (bi)clusters with

respect to metabolic pathways of organisms stored in Kegg, and provides an estimate of obtaining results by chance, including two statistical corrections (Bonferroni, andWestfall and Young). One of the most important features of *CarGene* is the possibility of simultaneously comparing and statistically analysing the information about many groups of genes in both visual and textual manner. Furthermore, it includes its own web browser to explore in detail the information extracted from Kegg.

Índice general

Índice de figuras

Índice de tablas

Parte I

Introducción

Capítulo 1

Introducción

El sabio no dice nunca todo lo que piensa,
pero siempre piensa todo lo que dice.
Aristóteles.

1.1. Motivación

La información es poder. Fue el filósofo Francis Bacon (1561–1626) al parecer el primero en expresar esta idea. Sin embargo, el manejo de la información siempre ha sido de gran importancia. Imaginémonos por ejemplo al Homo Ergaster hallando una fuente de alimentos y manteniéndola en secreto para él y su clan o una empresa multinacional recabando información de mercado para poder ampliar horizontes. La información se genera, se almacena, se vende, se compra, se publica, se esconde, etc. Todo ello impulsado hoy en día por el gran avance de las tecnologías de la información. Más que nunca, esta necesidad de conocer y manejar cantidades ingentes de datos se extiende a todos los ámbitos de nuestra sociedad. Y es esta necesidad, centrada en un área específica de la ciencia, la que ha sido motor impulsor de la tesis doctoral que se presenta en esta memoria.

En los últimos tiempos hemos asistido a una revolución tecnológica que ha permitido grandes avances en *genética* y *biología molecular*. Nuestras células pueden ser entendidas como bases de datos que guardan la información más trascendente de todas, aquélla en la que se basa la vida. Así, el proyecto Genoma Humano es el ejemplo más representativo de ello. Gracias a un gran esfuerzo económico y humano, se consiguió secuenciar todo el material genético del Homo Sapiens. La tecnología del microarray, es decir, la posibilidad de almacenar en un pequeño chip la respuesta de miles de genes ante ciertas condiciones de laboratorio supuso también nuevas vías

de investigación. En definitiva, la enorme cantidad de información disponible permite a biólogos y médicos afrontar nuevos retos e hipótesis, como por ejemplo, descubrir cómo se modifica la respuesta de un gen durante el transcurso de un proceso bioquímico, qué patrones de comportamiento de los valores de expresión de un gen son los responsables de ciertas enfermedades, cómo evoluciona el material genético en los seres vivos, y un largo etcétera.

Todas estas nuevas y apasionantes tareas requieren de herramientas especializadas. De esta forma, la *bioinformática* y la *biología computacional* surgen ante la necesidad de procesar toda la información biológica disponible. Estas nuevas ramas de la ciencia han de ser entendidas como un nexo de unión entre, por un lado, biólogos y médicos y, por el otro, todas aquellas disciplinas técnicas orientadas a analizar datos: ciencias de la computación, matemáticas, estadística, etc. A este respecto, es muy importante reducir al máximo la brecha tecnológica existente entre los usuarios finales y aquellos investigadores que diseñan las herramientas de análisis de la información, haciendo que éstas sean fáciles de manejar y que proporcionen, de manera eficiente, resultados con los que resulte sencillo trabajar a posteriori. Enmarcadas dentro de las ciencias de la computación, la *minería de datos* es un conjunto de técnicas orientadas a la extracción de conocimiento útil de grandes bases de datos. Este conocimiento generado nos permite no solo describir los datos, sino descubrir modelos de comportamiento capaces de ser predichos. De entre estas técnicas podemos destacar el clustering, el biclustering, la computación evolutiva o la generación de redes de conocimiento como las más utilizadas en bioinformática.

La información almacenada en las bases de datos biológicas es de muy diversa índole: expresión genética de un conjunto de genes ante un experimento concreto, relaciones entre genes y proteínas, secuencias de nucleótidos, relaciones entre factores de transcripción y motivos, etc. Muchas de estas bases de datos están expresadas en formato binario, es decir, codificadas mediante dos únicos valores: el 1 y el 0. Este formato presenta grandes ventajas ya que, por ejemplo, es una manera sencilla de almacenar relaciones entre distintos objetos y sus propiedades. Además, ocupa poco espacio y, a priori, son datos más fáciles de procesar. Sin embargo, en el ámbito de la bioinformática, son pocas las técnicas de minería de datos especialmente diseñadas para tratar con este tipo de información.

1.2. Planteamiento

Esta tesis doctoral plantea una nueva metodología no supervisada para la extracción de biclusters a partir de bases de datos binarios. Las técnicas de biclustering representan una alternativa a las muy utilizadas técnicas de clustering, debido al significado especial de los datos biológicos. En concreto, dentro del análisis de datos de expresión genética, el conocimiento que se tenía de la naturaleza de los genes hizo pensar que la búsqueda de un patrón local de comportamiento, objetivo del biclustering, se acercaría más a la realidad de los datos que la búsqueda de patrones globales, llevada a cabo por las técnicas de clustering. Así, se definió un bicluster como un subconjunto de genes que se comporta de la misma manera bajo un subconjunto de condiciones experimentales (Madeira y Oliveira, 2004). La obtención de estas submatrices implica el análisis simultáneo de todas las dimensiones de la base de datos.

El nuevo algoritmo presentado en esta memoria de tesis, *BiBit*, ha sido diseñado para ser independiente del tipo de base de datos en el que sea utilizado, sea de expresión genética o no. Ello implica una generalización del concepto de bicluster y una mayor robustez ante cambios de forma y tamaño de los datos de entrada. Además, se ha hecho especial hincapié en obtener un número coherente de resultados de la manera más sencilla y rápida posible, salvando así las dificultades que pueda tener un usuario final a la hora de utilizar esta nueva herramienta.

Una vez diseñado el algoritmo, los esfuerzos se han centrado en probar su comportamiento en distintos escenarios. De esta forma, y mediante el diseño de bases de datos sintéticas, se han elaborado multitud de pruebas para medir el rendimiento de nuestro método con respecto al tiempo de ejecución, número de resultados generados, velocidad a la hora de generar estos resultados, precisión en cuanto a encontrar biclusters concretos escondidos en los datos y escalabilidad. Y todo bajo condiciones cambiantes: distintos valores de los parámetros de entrada, distintos tamaños de la base de datos, distintas densidades de elementos iguales a 1 y distintos tipos de biclusters objetivo.

Posteriormente, y una vez conocidas las verdaderas cualidades de nuestra propuesta, fue necesario verificar la utilidad del algoritmo *BiBit* analizando una base de datos real. Para ello, se escogió una base de datos de expresión genética referente al estudio de tumores cerebrales infantiles desarrollado en el trabajo de Pomeroy et al. (Pomeroy *et al.*, 2002). Esta base de datos fue seleccionada por haber sido estudiada en una publicación de prestigio y por contar con acceso sencillo a todos los datos y resultados. Enmarcada en el proceso de análisis de la base de datos anteriormente comentada, presen-

tamos una nueva metodología para binarizar bases de datos de expresión genética, cuyo objetivo es proporcionar bases de datos binarias que, lejos de perder información, aportan mayor significado a los biclusters finales.
Los resultados obtenidos en las pruebas anteriormente descritas han sido comparados con una de las técnicas de biclustering más referenciadas y, además, especialmente diseñada para bases de datos binarios: *Bimax* (Prelic *et al.*, 2006). Las conclusiones obtenidas reflejan que nuestra metodología mejora con mucho el rendimiento de *Bimax*, obteniendo además resultados de igual o mayor calidad.
Finalmente, se propone una herramienta software para poder validar grupos de genes desde un punto de vista biológico. Este software, *CarGene*, lleva a cabo un análisis de enriquecimiento de rutas metabólicas a partir de un grupo de genes, resultado de cualquier método computacional. Las rutas metabólicas utilizadas constituyen información almacenada en un repositorio biológico público llamado *Kyoto Encyclopedia of Genes and Genomes* (KEGG). La aplicación se ha diseñado para ofrecer un interfaz lo más amigable posible, permitiendo comparar simultáneamente, y de manera gráfica y textual, grupos de genes obtenidos a partir de distintas técnicas de minería de datos, y validar posteriormente los resultados con técnicas estadísticas apropiadas.

1.3. Objetivos

Los objetivos que se plantean en esta tesis doctoral son los siguientes:

1. Diseñar una técnica de biclustering para ser aplicada en bases de datos binarios. Se ha pretendido desarrollar un método fácil de utilizar y eficaz. Fácil de utilizar, debido al reducido número de parámetros de entrada que afectan al resultado final. Eficaz, porque consigue un buen rendimiento con respecto a la calidad de los resultados obtenidos.

2. Otro aspecto muy importante a tener en cuenta es la eficiencia. Dado que el número de posibles resultados durante la creación de biclusters es normalmente muy elevado, se ha puesto especial cuidado en obtener un número coherente de biclusters con la calidad necesaria para obtener conocimiento útil de las bases de datos analizadas. Además, se ha planteando una técnica de procesamiento de secuencias binarias que muestra un magnífico rendimiento en cuanto al coste computacional se refiere.

3. Finalmente, nuestra propuesta de técnica de biclustering es robusta

frente a la estructura de la información: tamaño de las bases de datos, densidad de elementos iguales a 1, valores de los parámetros de entrada, distintas características de los biclusters objetivo, etc. Para verificar esta robustez, se ha diseñado una batería de pruebas basadas en bases de datos sintéticas con el objetivo de estudiar el comportamiento del algoritmo ante distintas situaciones.

4. Se ha planteado una nueva metodología de binarización de bases de datos que permite obtener, a partir de una base de datos basada en números reales o enteros, N bases de datos binarias que aportan más posibilidades de encontrar resultados interesantes y mayor semántica a los biclusters finales. Esta metodología ha sido aplicada al estudio de una base de datos real, basada en datos de expresión genética de tumores cerebrales infantiles, con el objetivo de aislar biclusters que representen a cada tumor y separar aquellos genes que intervienen en mayor medida en un proceso cancerígeno u otro.

5. Diseñar e implementar una herramienta software para la validación de resultados a través de repositorios biológicos públicos. El repositorio público utilizado en dicha herramienta es *Kyoto Encyclopedia of Genes and Genomes* (KEGG), y el objetivo es analizar el enriquecimiento de rutas metabólicas a partir de grupos de genes mediante procedimientos de contraste de hipótesis múltiple.

1.4. Principales contribuciones

Las contribuciones resultado de este trabajo de tesis son las siguientes:

1. Principales contribuciones:

   - A biclustering algorithm for extracting bit-patterns from binary datasets. Rodriguez-Baena DS, Perez Pulido AJ, Aguilar-Ruiz JS. Bioinformatics, Vol. 27(19), pp. 2738-2745, 2011.
   - CarGene: Characterisation of Set of Genes based on Metabolic Pathway Analysis. Aguilar-Ruiz JS, Rodriguez-Baena DS, Diaz-Diaz N, Nepomuceno-Chamorro IA. International Journal of Data Mining and Bioinformatics (IJDMB), Vol. 5(5), pp. 558-573, 2011.
   - Caracterización de un conjunto de genes basado en el análisis de Pathways metabólicos. Aguilar-Ruiz JS, Rodríguez-Baena DS, Díaz-Díaz N, Nepomuceno-Chamorro IA, Gómez-Vela F. Actas

del V Simposio de Teoría y Aplicaciones de Minería de Datos (TAMIDA), pp. 251-264, 2010.

- Discovering alpha-Patterns from Gene Expression Data. Rodriguez-Baena D, Díaz-Díaz N, Aguilar-Ruiz JS, Nepomuceno-Chamorro IA. International Conference on Intelligent Data Engineering and Automated Learning (IDEAL). Lecture Notes in Computer Science, Vol. 4881, pp. 831-839, 2007.
- Análisis de datos de expresión genética para la obtención de patrones ALFA. Rodríguez-Baena DS, Aguilar-Ruiz JS, García-Gutiérrez J. Actas del IV Simposio de Teoría y Aplicaciones de Minería de Datos (TAMIDA), pp. 265-272, 2007.
- Biclustering of gene expression data based on local nearness. Aguilar-Ruiz JS, Rodríguez-Baena DS, Simovici D. Actas de Extraction et Gestion Des Connaissances (EGC), Vol. 6, pp. 681-692, 2006.

2. Otras contribuciones relacionadas con la tesis:

- Gene association analysis: a survey of frequent pattern mining from gene expression data. Alves R, Rodríguez-Baena DS, Aguilar-Ruiz JS. Briefings in Bioinformatics, Vol. 11, pp 210-224, 2010.
- Gene Association Analysis and Frequent Pattern Mining definitions. Aguilar-Ruiz JS, Rodríguez-Baena DS, Alves R. Encyclopedia of Systems Biology, Springer ESB. In press. 2011
- Pattern Recognition in Biological Time Series. Gómez-Vela F, Martínez-Álvarez F, Barranco C, Díaz-Díaz N, Rodríguez-Baena DS, Aguilar-Ruiz JS. 14th Conference of the Spanish Association for Artificial Intelligence, CAEPIA. Lectures Notes in Computer Science, Vol. 6679, pp 164-172, 2011.
- Gene Regulatory Networks Validation Framework Based in KEGG. Díaz-Díaz N, Gómez-Vela F, Rodríguez-Baena DS, Aguilar-Ruiz JS. 6th International Conference on Hybrid Artificial Intelligent Systems (HAIS). Lectures Notes in Computer Science, Vol. 7023, pp 279-286, 2011.
- A Deterministic Model to Infer Gene Networks from Microarray Data. Nepomuceno-Chamorro IA, Aguilar-Ruiz JS, Díaz Díaz N, Rodriguez-Baena DS, García J. Lecture Notes in Computer Science Vol. 4881, pp. 850-859 (IDEAL), 2007.

- Software y validación de técnicas de conocimiento en Bioinformática. García-Gutiérrez J, Díaz-Díaz N, Rodríguez-Baena DS, Martínez-Álvarez F, Aguilar-Ruiz JS. 1th Workshop español sobre extracción y validación de conocimiento en bases de datos biomédicas (EVABIO), pp 75-84, 2007.
- Técnicas de aprendizaje no supervisado aplicadas a series temporales de datos de expresión genética. Martínez-Álvarez F, Rodríguez-Baena DS, Troncoso-Lora A, García-Gutiérrez J. 1th Workshop español sobre extracción y validación de conocimiento en bases de datos biomédicas (EVABIO), pp 55-64, 2007.
- Neighborhood-Based Clustering of Gene-Gene Interactions. Díaz-Díaz N, Rodríguez-Baena DS, Nepomuceno-Chamorro IA, Aguilar-Ruiz JS. International Conference on Intelligent Data Engineering and Automated Learning (IDEAL). Lecture Notes in Computer Science, Vol. 4224, pp 1111-1120, 2006.

3. Otras contribuciones:

- Clustering Main Concepts from E-mails. Aguilar-Ruiz JS, Rodríguez-Baena DS, Cohen PR, Riquelme JC. Lectures Notes in Computer Science, Vol. 3040, pp 231-240, 2004.
- Knowledge Extraction from E-mail Texts. Aguilar-Ruiz JS, Rodríguez-Baena DS, Riquelme JC. X Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA), pp 129-138, 2003.
- Minería de Textos y Datos aplicadas a la obtención de Ontologías a partir de mensajes de correo electrónico. Rodríguez-Baena DS, Aguilar-Ruiz JS, Riquelme JC. VIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD), pp 645-654, 2003.
- Evolutionary Neuroestimation of Fitness Functions. Aguilar-Ruiz JS, Mateos-García D, Rodríguez-Baena DS. Lectures Notes in Computer Science, Vol. 2902, pp 74-83, 2003.
- Local Nearest Neighbors By Competition. Ferrer-Troyano FJ, Giraldez-Rojo R, Aguilar-Ruiz JS, Rodríguez-Baena DS, et al. Frontiers in Artificial Intelligence and Applications, Vol. 82, pp 260-264, 2002.
- Discretization Oriented to Decision Rule Generation. Giraldez-Rojo R, Aguilar-Ruiz JS, Riquelme JC, Rodríguez-Baena DS, et al. Frontiers in Artificial Intelligence and Applications, Vol. 82, pp 275-279, 2002.

- Discretization by Maximal Global Goodness. Giraldez-Rojo R, Aguilar-Ruiz JS, Riquelme JC, Rodríguez-Baena DS. International Conference on Fuzzy Systems and Knowledge Discovery, pp 742-746, 2002.

1.5. Organización

Esta memoria de tesis doctoral está organizada de la siguiente forma: en primer lugar, es necesario proporcionar una serie de conceptos básicos sobre genética para poder comprender mejor los siguientes capítulos. Así, el capítulo 2 tiene como objetivo introducir al lector a la bioinformática, mostrando a la célula como un sistema de información propio, cuya base de datos es el material genético, y explicando de manera sencilla el funcionamiento de los procesos biológicos más importantes. Finalmente, se hará una descripción de la bioinformática, sus objetivos principales y se explicará en qué consiste la tecnología de microarray.
Los capítulos 3 y 4 corresponden al estado del arte. El capítulo 3 expone en qué se basan las técnicas de clustering y cómo se han adaptado a los distintos problemas biológicos. En concreto, al análisis de datos de expresión genética. Se hace una recopilación de las técnicas de clustering más representativas aplicadas en bioinformática. El capítulo 4 presenta a las técnicas de biclustering como una evolución necesaria del clustering debido a la naturaleza de los datos estudiados. Se formula el problema, se describen los distintos tipos de resultado, se clasifican los algoritmos de biclustering en función de la forma en la que se afrontan los objetivos y, de cada tipo de clase, se hace un resumen de la técnica más representativa. Finalmente, el capítulo se centra en aquellas técnicas de biclustering especialmente diseñadas para ser aplicadas a datos binarios.
Los capítulos 5, 6 y 7 están dedicados a la propuesta de esta tesis doctoral. En el capítulo 5 se expone un nuevo algoritmo de biclustering, *BiBit*, especialmente diseñado para bases de datos binarios. En primer lugar, mostramos las motivaciones que nos han llevado a desarrollar este trabajo. Posteriormente, se explica la metodología utilizada y finalmente se formaliza el algoritmo. El capítulo 6 recoge toda la batería de pruebas que se han llevado a cabo para analizar el comportamiento del algoritmo *BiBit*. Para ello, se han utilizado una serie de bases de datos sintéticas que ofrecen diferentes escenarios donde verificar la bondad de nuestra propuesta: tiempos de ejecución, número de resultados, precisión en la búsqueda de biclusters, escalabilidad, etc. Los resultados obtenidos han sido comparados con el algoritmo de biclustering *Bimax* (Prelic *et al.*, 2006). Una nueva herramienta

para la validación biológica de grupos de genes es la que ocupa el contenido del capítulo 7. Después de mostrar los motivos por los cuales es necesario este tipo de validación, se describe las distintas partes que forman la herramienta y se dan detalles de su implementación.
La parte experimental de este trabajo de tesis está recogida en el capítulo 8. Una base de datos real ha sido analizada en profundidad para demostrar la utilidad de la técnica presentada en esta memoria. Es una base de datos de expresión genética a la que se le aplicará, en primera instancia, una nueva metodología de binarización, fruto también de nuestro trabajo de investigación. Finalmente, indicar que en este capítulo también se establecen comparativas de resultados con el algoritmo de biclustering *Bimax* (Prelic *et al.*, 2006).
Las conclusiones y la recopilación de futuros trabajos será descrita en el capítulo 9.
Finalmente, los anexos incluyen datos de interés referentes a la experimentación llevada a cabo durante esta tesis doctoral, tanto con bases de datos sintéticas como con datos reales.

Capítulo 2

Preliminares en Biología y Bioinformática

Y ahora las declaraciones de Watson y Crick sobre el ADN. Esto es para mí la prueba verdadera de la existencia de Dios.
Salvador Dalí.

2.1. La célula como sistema de información

Se define 'Sistema' como un conjunto de elementos cooperantes entre sí que trabajan en pos de un objetivo común. En el caso de los sistemas de información, ese objetivo común es el de almacenar, procesar, usar y mantener un conjunto de datos provenientes de una fuente. El objetivo de este capítulo es dar las claves para comprender a las células que forman a los seres vivos como sistemas de información.

Este capítulo pretende además ser una breve y concisa introducción a la Biología Molecular y la Bioinformática, con el objeto de entender mejor algunos de los términos utilizados en la investigación propuesta en este documento.

2.1.1. La célula

La teoría celular establece que todos los organismos vivos están formados por células y, en general, se acepta que ningún organismo es un ser vivo si no consta al menos de una célula. Se enunció a mediados del siglo XIX y actualmente es un hecho aceptado. La célula es la unidad mínima de un organismo capaz de actuar de manera autónoma.

Hay células de formas y tamaños muy variados. Algunas de las células

bacterianas más pequeñas tienen forma cilíndrica de menos de una micra (una millonésima de metro) de longitud. En el extremo opuesto, se encuentran las células nerviosas, corpúsculos de forma compleja con numerosas prolongaciones delgadas que pueden alcanzar varios metros de longitud. A pesar de observarse gran diversidad en cuanto a tamaño, forma, localización y funcionalidad, todas ellas poseen unas características estructurales comunes.

La principal fuente de datos que encierra la célula está en el centro mismo de este sistema de información: el *núcleo*. Esta base de datos contiene la llamada información de la vida que determina las características de los seres vivos y es básica en la herencia o en la actividad celular, donde la información vital está contenida en macromoléculas llamadas *ADN* (Ácido desoxirribonucleico). Estas grandes moléculas se relacionan con ciertas proteínas formando una maraña de fibrillas y grumos que se tiñen con facilidad con colorantes básicos y que dan un aspecto reticular a dicho núcleo. Se trata de un material denominado *cromatina*. La cromatina se organiza, en ciertos momentos de la vida celular, en pequeños cuerpos en forma de bastoncillos en asa llamados *cromosomas*.

Son varias las actividades fundamentales en las células que requieren la participación de la información almacenada en el ADN (Griffiths *et al.*, 2005):

- Replicación: Copia de seguridad de datos.
- Transcripción: Copia de una parte de los datos para poner en marcha la construcción de una herramienta (proteína).
- Traducción: Construcción de la herramienta.
- Actividad Celular: Aplicación de la herramienta a un caso de uso concreto.
- Mutación: Transformación de la información que puede adaptarla a situaciones cambiantes.

Uno de los objetivos básicos de los seres vivos es la perpetuación de la especie a partir de la herencia del material genético. El ADN contiene información sobre las características concretas de un ser vivo. Esa información ha de ser replicada para que pase a futuras generaciones. Esta copia de datos se realiza en dos momentos fundamentales: al producirse un tipo de célula que asegurará la continuación de una especie de una generación a otra (gametos) y cuando la primera célula de un nuevo organismo (cigoto) comienza

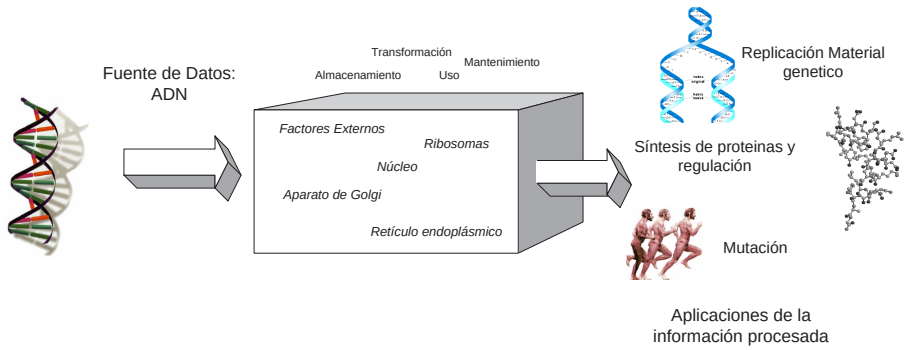

Figura 2.1: La célula como Sistema de Información

a dividirse para crear un organismo multicelular más complejo. Esta copia de seguridad comienza a realizarse en el interior del núcleo de la célula.

La actividad celular es el segundo uso importante de la información gestionada por la célula. Según su funcionalidad y los estímulos externos que recibe, la célula tendrá que desarrollar determinadas acciones. Para llevar a cabo dicha actividad es necesario el uso de las moléculas más abundantes en los seres vivos: las proteínas. Las proteínas son los materiales que desempeñan un mayor número de funciones en las células de todos los seres vivos. Por un lado, forman parte de la estructura básica de los tejidos (músculos, tendones, piel, uñas, etc.) y, por otro, desempeñan funciones metabólicas y reguladoras. La célula fabrica estas moléculas según conveniencia y las instrucciones de montaje de las mismas están contenidas en el ADN. La célula tiene que leer dicha información, procesarla, transportarla a la zona de fabricación de proteínas, sintetizar la proteína y utilizarla dentro de la célula o expulsarla de la misma para que realice su función en el exterior. Cada una de estas acciones se realiza en una parte de la célula y todas ellas cooperan entre sí para obtener el resultado final.

Finalmente, durante la mutación tiene lugar una transformación de los datos almacenados en la célula como una medida de adaptación de los seres vivos al entorno que, a largo plazo, es el punto de partida en la evolución de las especies.

En resumen, la célula es un sistema de información que contiene una fuente de datos, el ADN, almacenada en el núcleo y que, debido a la cooperación de factores internos (funcionalidad celular, partes de la célula: núcleo, ribosomas, retículos endoplasmáticos, aparato de golgi) y factores externos (condiciones ambientales, necesidades fisiológicas del ser vivo, enfermedades) procesan, transforman, usan y copian dicha información. Este resumen puede ser observado gráficamente en la figura 2.1.

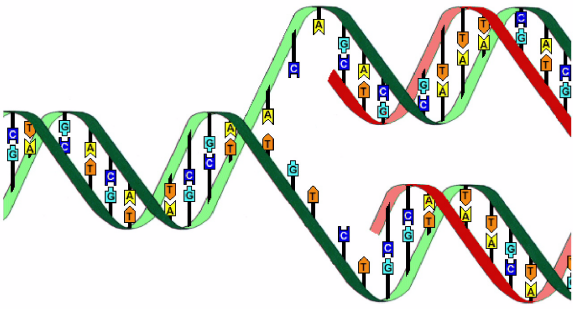

Figura 2.2: La doble hélice

2.1.2. Representación de los datos

Esta sección profundiza en la fuente de información que maneja la célula y que está contenida en las macromoléculas llamadas ADN. Y para comprender cómo están representados los datos dentro del ADN se ha de conocer en primer lugar su estructura.

La macromolécula de ADN se presenta como una estructura de doble hélice compuesta por dos cadenas entrelazadas, siendo cada una de ellas construida a partir de elementos básicos llamados *nucleótidos*. Estos nucleótidos son moléculas formadas por varios compuestos entre los que hemos de destacar las bases nitrogenadas (compuestos orgánicos cíclicos con dos o más átomos de nitrógeno). Encontramos en los nucleótidos cuatro tipos distintos de bases nitrogenadas: adenina (A), guanina (G), citosina (C) y timina (T). Las bases nitrogenadas son complementarias, es decir, forman parejas de igual manera que lo harían una llave y su cerradura. La adenina y la citosina son complementarias (A-C), y lo mismo ocurre con la guanina y la timina (G-T). De esta manera, las cadenas complementarias se entrelazan entre sí, tal y como muestra la figura 2.2.

La codificación de la información almacenada en la cadena de ADN viene dada por su longitud y la secuencia de aparición de los distintos nucleótidos. Así pues, en esta base de datos la unidad básica de información está representada por cada nucleótido. Existen, sin embargo, agrupaciones de estas unidades básicas que representan la misma información pero a más alto nivel. Un *gen* es una secuencia lineal de nucleótidos que cumple una función específica. La palabra *genética* viene de este término, y los genes son el centro de estudio de esta rama de la ciencia. Se llama *genoma* a todo el material genético contenido en los cromosomas de un organismo en particular. El descubrimiento de los genes y la comprensión de su estructura molecular y sus

funciones, han supuesto grandes avances a la hora de responder dos de los grandes misterios de la Biología:

- ¿Qué hace diferentes a las distintas especies de seres vivos?
- ¿Qué produce la variación de individuos dentro de una misma especie?

Los genes son los responsables de las características propias de una especie. La información contenida en estos genes se usa principalmente para producir proteínas, que son las moléculas más abundantes en los seres vivos. La periodicidad y la tasa de producción de las proteínas y otros componentes celulares corresponden a la acción de los genes en las células y a las condiciones externas en la que se desenvuelve un organismo. La respuesta a la segunda pregunta es que cada gen puede existir en distintas formas que difieren unas de otras (alelos), generalmente en pequeños aspectos. Este hecho provoca la variación de individuos dentro de una misma especie.

2.1.3. La Producción de Proteínas

Para entender gran parte de los términos que se utilizan en este documento, a continuación se explicará brevemente una de las aplicaciones más importantes de este sistema de información que representa la célula: el proceso de creación de una proteína.

El primer paso realizado por la célula para crear una proteína se lleva a cabo en el núcleo (transcripción del gen) y se trata de copiar o transcribir la secuencia de nucleótidos de un trozo de un gen en una secuencia de nucleótidos complementaria. Esta secuencia formará una molécula llamada *ARN* (Ácido Ribonucleico). Una vez creada, está transcripción sufre una serie de modificaciones para transformarse en una especie de molécula mensajera llamada *ARNm*. Una vez creado el ARNm, atraviesa el núcleo de la célula para dirigirse directamente a la maquinaria celular responsable de la fabricación de las proteínas. A esta acción se le denomina *traducción*.

La síntesis de proteínas tiene lugar en unos orgánulos celulares llamados *ribosomas*. El ARNm va pasando por el ribosoma proporcionando las instrucciones pertinentes para la creación de la proteína. Una proteína es una estructura molecular formada por un grupo de moléculas llamadas *aminoácidos*. En el ARNm está codificada la secuencia de aminoácidos necesaria para ir construyendo la proteína.

Por lo cual, y como se observa en la figura 2.3, la síntesis de proteínas se lleva a cabo en dos pasos principales: transcripción y traducción.

Un gen puede albergar la información para sintetizar más de una proteína. Y a veces, para crear sólo una de ellas, es necesaria la colaboración

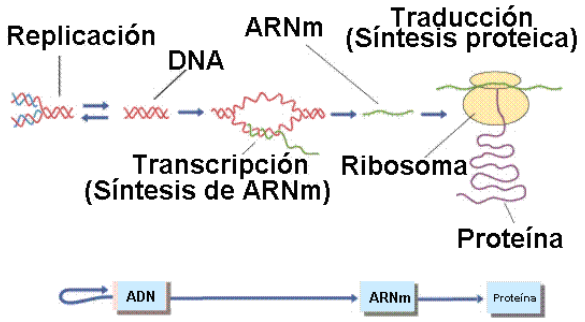

Figura 2.3: Esquema de la síntesis de Proteínas en la célula

de varios genes.

Se llama *expresión genética* a la transcripción de la información almacenada en un gen para crear una molécula de ARN. Cada célula puede contener en el núcleo entre 5000 y 60000 genes que codifican proteínas, pero, en un determinado instante, la célula sólo expresa un subconjunto de esos genes para llevar a cabo procesos celulares. ¿Qué es lo que regula el subconjunto de genes que se expresan en una célula en un momento determinado? Hay varios condicionantes a tener en cuenta: el tipo de célula, una determinada enfermedad, una respuesta dinámica a estímulos externos, la acción de otros genes, etc.

2.2. Introducción a la bioinformática

La bioinformática es una joven rama de la ciencia difícil de definir. A continuación se abordará el concepto de bioinformática desde distintos ángulos: su origen histórico, la necesidad a partir de la cual surgió y la revolución tecnológica que ha propiciado su crecimiento. Ésta es una tarea ardua ya que, además de estar en vías de desarrollo y de tener multitud de ramificaciones, la bioinformática consigue reunir a científicos de muy diversas materias: biólogos, físicos, matemáticos, estadísticos, informáticos, químicos, etc.

2.2.1. Origen histórico

Para entender el concepto de bioinformática es básico descubrir cuál es su sentido, es decir, la necesidad que fue punto de partida de esta rama de la ciencia.

La biología ha sufrido un cambio profundo en los últimos años. Antes, en un laboratorio de biología, podían encontrarse multitud de instrumentos de medición tales como microscopios o recipientes para el almacenamiento

Figura 2.4: Secuencia histórica de descubrimientos en genética desde Mendel hasta el genoma humano

de muestras, además de material para analizar y recopilar datos, maquetas con representaciones moleculares, etc. Pero actualmente, es necesario el uso de la informática, por lo que todo laboratorio ha de contar con ordenadores, robots de secuenciación, servidores, dispositivos de almacenamiento masivo, potentes bases de datos y, en definitiva, la más alta tecnología de la información. Si nos centramos en la rama de la genómica (el estudio del conjunto de genes de un organismo), el motivo de este cambio tiene sencilla explicación si se analiza la evolución de los descubrimientos llevados a cabo hasta ahora y mostrados esquemáticamente en la figura 2.4.

En 1866 Mendel descubre los genes. En principio no se sabía qué era un gen, dónde se encontraba y de qué estaba formado. Sin embargo, Mendel ya pudo intuir cuáles eran sus funciones y cómo influían en la herencia. Posteriormente, en 1871 se descubren los ácidos nucleicos: la gran molécula de la vida. Los primeros pasos en la genética fueron lentos y hasta el siglo siguiente no se hicieron descubrimientos nuevos, como por ejemplo en 1953, año en que se descubrió la estructura del ADN. A partir de este descubrimiento

se empezó a buscar a los genes en dicha estructura y, por consiguiente, la existencia de un código genético. Por fin, entre 1975 y 1979, se aísla el primer gen humano. Este hecho hace que la genómica dé un salto espectacular y se pase de estudiar un sólo gen a tener descifrados códigos genéticos sencillos pertenecientes a bacterias para, finalmente, llegar a conseguir la secuenciación completa del genoma humano. Es decir, se ha pasado de estudiar un sólo gen, ignorando el 99 % del restante material genético, a estudiar el genoma completo. ¿Por qué? La diferencia está en la cantidad de información disponible. Antes, los tímidos descubrimientos en genética proporcionaban escasos datos para la investigación. En los últimos años, los avances han sido tales que actualmente se cuentan con bases de datos formadas por millones y millones de entradas en las que se encuentran genomas completos. Se ha pasado de la nada al todo en algunas décadas.

Esta nueva situación impone un cambio en el paradigma de la genómica, y ese cambio está íntimamente relacionado con las tecnologías de la información. La única posibilidad de manejar cantidades tan monstruosas de datos es contar con el hardware y el software necesario, y de esta necesidad surge la bioinformática. Además, esta evolución en la biología molecular pudo llevarse a cabo gracias a una también importante evolución tecnológica. En conclusión, una gran cantidad de datos generados gracias a la tecnología y que necesita de ésta para poder ser manejada.

2.2.2. Definición de bioinformática

La bioinformática representa un punto de unión en las revoluciones científicas y tecnológicas del siglo veinte llevadas a cabo en los campos de la biología y la informática. El objetivo básico de esta nueva disciplina es el uso de bases de datos y algoritmos computacionales para analizar proteínas, genes y la completa colección de ADN que forma un organismo (genoma), generando herramientas software que ayudan a revelar los mecanismos fundamentales que hay detrás de problemas biológicos relacionados con la estructura y función de las macromoléculas, procesos bioquímicos, enfermedades, evolución, etc.

Según el *National Institute of Health* de Estados Unidos (Huerta *et al.*, 2000), la bioinformática es: *la investigación, desarrollo o aplicación de herramientas computacionales y propuestas científicas para extender y facilitar el uso de datos biológicos, médicos o sanitarios, incluyendo la adquisición, almacenamiento, organización, análisis y visualización de los mismos.* Existe una disciplina científica relacionada con la bioinformática, la biología computacional, que el mismo instituto define así: *el desarrollo y la aplica-*

ción de datos analíticos y métodos teóricos, modelado matemático y técnicas de simulación por ordenador para estudiar sistemas biológicos, de conducta y sociales. Aunque normalmente estas dos disciplinas son utilizadas indistintamente, la biología computacional es un campo más restringido basado en desarrollos matemáticos y métodos computacionales concretos.

La bioinformática representa un campo científico muy amplio que se puede resumir a partir de tres perspectivas distintas (Ideker *et al.*, 2001). La primera de sus perspectivas es la célula. El dogma central de la biología molecular es que el ADN es transcrito a ARN y transformado en proteínas. Uno de los principales objetivos de la bioinformática es pues el genoma (completa colección de ADN de un organismo), el transcriptoma (ARN generado a partir del ADN) y el proteoma (las secuencias de proteínas acumuladas hasta el momento). Estos millones de secuencias moleculares representan una gran oportunidad para aplicar algoritmos computacionales y bases de datos cuyo objetivo final será determinar la funcionalidad de ciertas células y las relaciones gen-proteína.

A partir de la célula se puede elevar el nivel de abstracción hasta los organismos individuales, los cuales representan la segunda perspectiva de la bioinformática. Cada organismo cambia a través de sus diferentes estados de desarrollo en diferentes regiones del cuerpo. Los genes, lejos de ser entidades estáticas, son regulados dinámicamente en respuesta al paso del tiempo, la región y el estado fisiológico. La expresión genética varía en estados de enfermedad o en respuesta a una gran variedad de señales, tanto intrínsecas del propio organismo como ambientales. Muchas de las herramientas generadas por la bioinformática son aplicadas a bases de datos que incluyen la expresión de una serie de genes, obtenida a partir de diferentes tejidos y condiciones en experimentos cuyos resultados se almacenan en chips capaces de albergar la expresión de cientos de genes.

Por último, en el más alto nivel de abstracción posible, aparece la tercera perspectiva de la bioinformática: el árbol de la vida. Hay millones de especies de seres vivos que pueden ser agrupadas en tres grandes ramas: bacterias, archaea y procariotas. La bioinformática ayuda en este nivel a estudiar las similitudes existentes entre los seres vivos a nivel molecular y en la comparación de genomas.

2.2.3. Principales áreas de investigación

A partir de estas tres perspectivas, las áreas de investigación más importantes de la bioinformática son resumidas a continuación. Con respecto a la perspectiva celular, la disciplina más importante es la *extracción y análisis*

de Secuencias de ADN, ARN y proteínas. Desde que en el 1977 se produjeran importantes avances en la secuenciación del ADN, más y más genomas se han ido secuenciando y almacenando en bases de datos. Uno de los ejemplos más importantes de nuestra era fue el proyecto Genoma Humano. Este proyecto comenzó en 1990 y su objetivo básico era la secuenciación de todo el genoma humano. Dos equipos compitieron por llevar a cabo esta gran empresa. El primer equipo era de carácter público y estaba formado por un consorcio de muchos países: *International Human Genome Sequencing Consortium*. El otro fue una iniciativa privada de la empresa *CELERA*. Para la secuenciación del genoma humano se utilizó la más alta tecnología: avanzados algoritmos de secuenciación (Genome Shot Gun Algorithm), multitud de potentes ordenadores formando clusters, robots de secuenciación, estaciones de trabajo para la composición de resultados, etc. Todo este dispositivo, de aproximadamente unos 3.000 millones de dólares, tuvo su recompensa cuando en el 2003 se completó la tarea. Aunque el resultado del consorcio fue más preciso, la empresa CELERA consiguió con creces su segundo objetivo encubierto: vender miles de robots ABI de secuenciación creados por ellos mismos para el proyecto.

Actualmente tres grandes bases de datos públicas almacenan grandes cantidades de secuencias de nucleótidos y proteínas: *GenBank*, en el Centro Nacional de Biotecnología de los Estados Unidos (http://www.ncbi.nih.gov), la *Base de Datos de ADN de Japón* (DDBJ, http://www.ddbj.nig.ac.jp/) y el *Instituto Europeo de Bioinformática* (EBI, http://www.ebi.ac.uk/) en Inglaterra. Estas tres instituciones intercambian sus secuencias diariamente como parte de una colaboración internacional.

Relacionada con la segunda de las perspectivas, el *análisis de datos de expresión genética* es el estudio de los ARNm transcritos por un conjunto de genes en distintas condiciones experimentales. El objetivo es intentar comprender la expresión de los genes en determinadas circunstancias para, por ejemplo, poder dar un diagnóstico precoz de una enfermedad. Esto se consigue comparando la situación diana de interés (células de un enfermo, tratadas con un medicamento, etc.) frente a una situación de control (células normales). Ésta es la disciplina sobre la que está centrado el ejemplo de aplicación de esta tesis doctoral.

Finalmente, con respecto a la biodiversidad de este planeta, la bioinformática juega también un papel importante. La reciente secuenciación de genomas de todas las ramas de seres vivos presenta una extraordinaria oportunidad en el ámbito de la biología. Como analogía, se puede asemejar este momento al vivido en el siglo XIX cuando se completó la tabla periódica de los elementos. Este hecho marcó el comienzo de nuevas tareas como la

de comprender la organización de los elementos en esa tabla e incluso la predicción de nuevos elementos por aparecer. Un estado similar vive la biología molecular en estos momentos. Unido a la gran cantidad de genomas secuenciados, comienza ahora el reto de comprender la organización de este conjunto de genomas. Ramas de la bioinformática como el *el estudio y medición de la biodiversidad* o la *comparación de genomas* intentan aplicar la tecnología computacional para recopilar información sobre las distintas especies y contrastarla.

2.3. La expresión genética

El ejemplo de aplicación de la metodología propuesta en esta tesis doctoral se va a centrar en el análisis de datos de expresión genética. La expresión de un gen se obtiene cuando una porción de ADN se transcribe para crear una molécula de ARN como primer paso en la síntesis de proteínas. Del numeroso grupo de genes que se encuentran en el núcleo y que codifican proteínas, sólo un subconjunto de ellos se expresará en un momento determinado. Esta expresión selectiva viene regulada por distintos aspectos: tipo de célula, fase de desarrollo del ser vivo, estímulo interno o externo, enfermedades, etc.

2.3.1. Introducción

La comparación entre distintos perfiles de expresión genética es una herramienta básica para poder responder a un gran número de cuestiones biológicas. De estas cuestiones cabe destacar la identificación de los genes de un organismo que se activan durante un ciclo celular o una producción de proteínas para el exterior. Cobran también mucha importancia las investigaciones realizadas sobre el efecto de las enfermedades en las expresiones genéticas de roedores, primates y humanos. En los últimos años, el estudio en los datos de expresión ha sido también de gran ayuda en las anotaciones realizadas sobre los genomas secuenciados hasta el momento. Cuando el ADN se secuencia, una de las tareas que cobra más importancia es la de detectar en qué tramos de dicha secuencia podemos encontrar los genes y cuál es la función de éstos.

Para poder llevar a cabo todos estos objetivos se han utilizado distintas tecnologías con el fin de obtener, almacenar y analizar esta información. Muchas de esas tecnologías se basaban en obtener los datos de expresión de un sólo gen en determinadas condiciones experimentales. Frente a estas técnicas han aparecido en los últimos años tecnologías que nos permiten obtener

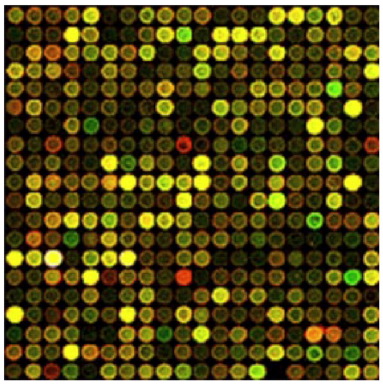

Figura 2.5: Sección de un microarray.

datos simultáneamente sobre una gran cantidad de genes. Esta generación masiva de datos ofrece las siguientes ventajas frente al estudio individual de los genes:

- Un estudio masivo de datos facilita la identificación de genes individuales que son expresados de manera desmesurada en algún estado biológico concreto.
- El análisis simultáneo de un conjunto de genes permite revelar patrones similares de comportamiento en determinadas condiciones experimentales, al igual que encontrar grupos de genes que reaccionen de forma inversa ante determinados estímulos.

Una de las tecnologías más utilizadas hoy en día de generación masiva de datos de expresión es la creación de *microarrays*.

2.3.2. Los microarrays

El uso de microarrays, cuya tecnología surgió del trabajo pionero de un grupo de científicos de Stanford y el NIH, entre otros (DeRisi *et al.*, 1996), ha emergido como una potente técnica para la medición de datos de expresión genética y la comparación de la abundancia relativa de ARN mensajero generado en distintas pruebas biológicas.

El microarray, también llamado *DNA chip* o *Biochip*, es un soporte sólido construido normalmente en cristal o en membrana de nylon. Son diversas las técnicas usadas para construir estos chips biológicos integrados: fotolitografía, robots piezoeléctricos, uso de haces de fibra óptica, etc.

¿Cuál es el proceso de creación de un microarray? En primer lugar se colocan trozos de ADN conocidos cuyos niveles de expresión pueden ser cuan-

tificados en una micromatriz del biochip. Después, se realizan experimentos bajo una serie de condiciones sobre células de distintos tejidos, extrayendo el ARN mensajero generado por los genes de dichas células. Estas moléculas de ARNm se marcan con fluidos fluorescentes o radiactividad y son transformadas a un tipo especial de ADN, ADNc, colocándose posteriormente en las micromatrices para comenzar el proceso de hibridación. La hibridación del ADN es el proceso más común para detectar un gen particular o un segmento de un ácido nucleico. El objetivo es crear cadenas dobles de ADN a partir de cadenas simples a sondear (las que fueron colocados inicialmente en las matrices) y otras cadenas simples marcadas (las obtenidas a partir de los experimentos y que son llamadas *sondas*). Después del proceso de hibridación, se eliminan todas las cadenas que no se han podido unir mediante lavados (sólo las moléculas que hibridan permanecerán en el biochip), y se procede al revelado mediante un escáner óptico o con microscopía láser confocal. El resultado final es una matriz en la que cada una de las celdas está marcada con un determinado color, como se puede observar en la figura 2.5. Las filas son genes y las condiciones experimentales las columnas. El color simboliza el grado de expresión genética de dicho gen frente a una determinada condición experimental.

La comunidad científica está entusiasmada con esta tecnología debido a las grandes ventajas que ofrece dentro de la investigación genética. Como ventaja más importante, ofrece la valiosa oportunidad de poder estudiar a la vez el comportamiento de una gran cantidad de genes bajo una serie de condiciones experimentales. Incluso genomas enteros pueden almacenarse en un biochip. Esta enorme base de datos de expresión representa un gran desafío con respecto a la obtención de información útil. ¿Qué genes se expresan de forma desmesurada, ya sea positiva o negativamente? ¿Qué patrones de comportamiento común existen entre las distintas agrupaciones de genes en la matriz? Para responder a estas preguntas, se aplican técnicas de estadística descriptiva y, referente a la informática y en concreto a esta tesis doctoral, distintas técnicas de minería de datos. Uno de los puntos negativos de esta tecnología lo constituye el hecho de que muchos científicos encuentran prohibitivo el coste de producir suficientes microarrays para sus experimentos.

La metodología propuesta en esta tesis doctoral se centra en estudiar la información obtenida a partir de estos microarrays mediante técnicas de minería de datos, y más concretamente, técnicas de biclustering. Las aplicaciones que puede llegar a tener la extracción de conocimiento útil de estas bases de datos es amplia y de gran relevancia:

- Estudio de la expresión genética ante sustancias tóxicas.
- Diagnóstico de enfermedades.
- Estudio de enfermedades genéticas complejas.
- Fisiología celular.
- Detectar polimorfismos y mutaciones.
- Análisis del comportamiento de la célula ante fármacos.
- Estudio de la expresión genética en el desarrollo.

2.4. Conclusiones

Este capítulo introductorio presenta a la célula, unidad autónoma más pequeña de un ser vivo, como un sistema de información cuya base de datos, el ADN, concentra la información más relevante para la vida. El ADN contiene las características concretas de un ser vivo y esa información es replicada para que se transmita a futuras generaciones. La fabricación de proteínas, piezas claves en la actividad celular, es otro de los usos más importantes de esta información. La base de datos que contiene el ADN se encuentra en el núcleo de la célula y tiene una estructura de doble hélice formada por cadenas complementarias de nucleótidos entrelazadas entre sí. Se define a un gen como una secuencia de nucleótidos que cumple funciones específicas, entre las que destaca la producción de proteínas y otros componentes celulares. La producción de proteínas es un proceso que se lleva a cabo en dos pasos: transcripción y traducción. La transcripción consiste en copiar o transcribir la secuencia de nucleótidos de un trozo de un gen a una molécula mensajera llamada ARNm. Esta molécula abandona el núcleo de la célula y se traduce en la información necesaria para fabricar una proteína.

Posteriormente, se define la bioinformática como aquella rama de la ciencia encargada del desarrollo de propuestas científicas para extender y facilitar el uso de datos biológicos o médicos. Así, la bioinformática nace ante la necesidad de procesar la ingente cantidad de información que se estaba generando en campos como la genética. Las siguientes disciplinas se pueden catalogar como principales áreas de investigación de la bioinformática: extracción y análisis de secuencias de ADN, ARN y proteínas, análisis de datos de expresión genética y, finalmente, estudio y medición de la biodiversidad. El estudio de la expresión genética cobra especial importancia en

este documento de tesis, ya que la experimentación con datos reales llevada a cabo se basa en este tipo de información. La capacidad de poder estudiar de manera simultánea el comportamiento de muchos genes tiene múltiples ventajas, entre las que destaca la posibilidad de revelar patrones similares de comportamiento de un conjunto de genes bajo ciertas condiciones experimentales. El análisis de la expresión genética no sería posible sin el uso de una nueva tecnología, la del microarray, que permite almacenar en un chip la respuesta de miles de genes ante ciertas condiciones de laboratorio.

Parte II

Estado del arte

Capítulo 3

Aplicación de técnicas de clustering sobre datos de expresión genética

Una síntesis vale por diez análisis.
Eugenio d'Ors.

Los grandes avances en la genética y en la tecnología puesta a su servicio han propiciado un aumento exponencial de la información disponible para los científicos. Prueba de esta revolución conjunta entre biología y tecnología es el uso de los microarrays para el almacenamiento de datos de expresión genética. En un pequeño chip se pueden almacenar las respuestas de miles de genes frente a un conjunto de condiciones experimentales, proporcionando una valiosa oportunidad de hacer un estudio simultáneo de la respuesta genética y de llevar a cabo la búsqueda de patrones de comportamiento.

Para llevar a cabo este gran reto se hace necesaria la aplicación de técnicas de *minería de datos* (data mining). Se llama minería de datos al conjunto de técnicas orientadas a la extracción de conocimiento útil de grandes bases de datos (KDD, Knowledge Discovery in Databases). Dentro de estas técnicas, y en referencia a las bases de datos de expresión genética, el *clustering* se ha consolidado como una de las técnicas mas utilizadas como primer paso en las tareas de descubrimiento de conocimiento. Posteriormente, el *biclustering* surgió para suplir algunas debilidades que el clustering presentaba frente a este tipo de bases de datos. Ambas técnicas serán introducidas y relacionadas en los dos siguientes capítulos, incluyéndose también una descripción de los algoritmos más conocidos.

3.1. Introducción

La predicción y la descripción de datos son dos de los objetivos principales que persigue la minería de datos. Dentro del ámbito de la descripción, las técnicas de clasificación permiten organizar un conjunto de datos mediante la asignación de clases. Una de las técnicas de clasificación no supervisada más importante es el *clustering*, cuyo objetivo es el de formar grupos o clases de datos, llamados *clusters*, de tal forma que los datos de un mismo grupo comparten una serie de características y similitudes. El hecho de que sea una clasificación no supervisada implica que en el clustering no se utilicen clases predefinidas o ejemplos de clasificación previos para poder realizar su labor.

Las técnicas de clustering han probado ser de gran utilidad a la hora de comprender la funcionalidad de los genes, su regulación, los procesos celulares y los distintos subtipos de células existentes. Una de las mayores tareas en el análisis de datos de expresión genética es la de descubrir grupos de genes que intervienen en una misma función celular o que están regulados de la misma manera, promoviendo incluso la comprensión de la funcionalidad de ciertos genes de los cuales no existía conocimiento previo (Eisen *et al.*, 1998) (Tavazoie *et al.*, 1999). Un primer paso consiste en agrupar los genes que tengan similares patrones de expresión (genes co-expresados). La búsqueda de secuencias comunes de ADN en las regiones organizadoras de los genes que se encuentran en un mismo cluster permite la identificación de los elementos reguladores dentro de dicho cluster (Brazma y Vilo, 2000) (Tavazoie *et al.*, 1999). Además, la inferencia de los procesos reguladores a través de la creación de clusters de datos de expresión genética genera interesantes hipótesis sobre las redes reguladoras de los procesos de transcripción (Dhaeseleer *et al.*, 1998). Finalmente, la aplicación de técnicas de clustering sobre las condiciones experimentales puede revelar subtipos de células que serían difíciles de descubrir utilizando los tradicionales métodos morfológicos (Alizadeh *et al.*, 2000) (Golub *et al.*, 1999).

3.2. Definiciones

3.2.1. Los datos

Mediante transformaciones matemáticas basadas en la aplicación de logaritmos, la información almacenada en los microarrays es transformada en matrices numéricas. Estos valores, números reales generalmente, representan la cantidad relativa de ARN mensajero expresado por un gen an-

te una determinada condición experimental. Así pues, los datos utilizado para el estudio de la expresión genética están organizados en matrices: $M = \{w_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$, en las que las filas representan a los genes, $G = \{g_1, g_2, \ldots, g_n\}$, y las columnas son condiciones experimentales, $C = \{c_1, c_2, \ldots, c_m\}$ y cada elemento w_{ij} es un número real que representa la cantidad relativa de ARNm expresado por un gen i ante una condición experimental j.

$$\mathbf{M} = \begin{pmatrix} w_{11} & w_{12} & \ldots & w_{1m} \\ w_{21} & w_{22} & \ldots & w_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ w_{n1} & w_{n2} & \ldots & w_{nm} \end{pmatrix}$$

La matriz original puede contener ruido, valores nulos y variaciones sistemáticas producidas durante la ejecución de los experimentos. Es por ello que el pre-procesado de estos datos es imprescindible a la hora de aplicar cualquier técnica de clustering. El problema del pre-procesamiento de los datos en una interesante área de investigación en sí misma. Por ejemplo, el problema de sustituir los valores nulos que aparecen en las matrices mediante estimación ha sido discutido en varios artículos (Troyanskaya *et al.*, 2001). La normalización de los datos es también una técnica de pre-procesamiento muy discutida y utilizada (Hill *et al.*, 2001) (Schuchhardt *et al.*, 2000). Además de los anteriores, existen más métodos de tratamiento de los datos de una matriz de expresión: filtrado de aquellos genes cuyo valor de expresión no cambie significativamente a lo largo de las condiciones experimentales, estandarizar cada fila de la matriz con una media de cero y una varianza de uno, discretización de la información, etc.

3.2.2. Medidas de proximidad

Las medidas de proximidad miden la similitud o distancia entre dos objetos. En este caso, los objetos serán o bien los genes (filas de la matriz) o bien las condiciones experimentales (columnas de la matriz). En cualquiera de los dos casos, estos objetos de expresión genética pueden ser representados en forma de vectores: $\vec{O}_i = \{o_{ij} \mid 1 \leq j \leq p\}$, donde o_{ij} es el valor de la característica j del objeto i, siendo p el número de características de dicho objeto. La proximidad entre dos objetos O_i y O_j es medida por una función de proximidad aplicada a sus vectores $\vec{O}_i$ y $\vec{O}_j$.

La *distancia euclídea* es uno de los métodos más comunes para medir

la distancia entre dos objetos. La distancia entre O_i y O_j en un espacio p-dimensional es medida por la siguiente fórmula:

$$Euclidea(O_i, O_j) = \sqrt{\sum_{d=1}^{p}(o_{id} - o_{jd})^2} \tag{3.1}$$

Sin embargo, en el caso de los datos de expresión genética, cobra mayor importancia la forma global de los patrones de expresión antes que el valor individual de cada característica. Éste es uno de los mayores problemas de la distancia euclídea, es decir, no identifica correctamente los fenómenos de desplazamiento y escalado que pueden aparecer entre los distintos patrones de comportamiento de la expresión genética, despreciándose así un buen número de agrupaciones correctas de genes o condiciones. Para solucionar este problema se han presentado distintas adaptaciones a la distancia euclídea (Wang *et al.*, 2002) o se ha aplicado un pre-procesamiento de los datos mediante la estandarización de los mismos (Tavazoie *et al.*, 1999) (Smet *et al.*, 2002) (Shamir y Sharan, 2000).

Una medida de proximidad alternativa es el *coeficiente de correlación de Pearson*, que mide la similitud existente entre las formas de dos patrones de expresión. Dados dos objetos, O_i y O_j, el coeficiente de correlación de Pearson se define como:

$$Pearson(O_i, O_j) = \frac{\sum_{d=1}^{p}(o_{id} - \mu_i)(o_{jd} - \mu_j)}{\sqrt{\sum_{d=1}^{p}(o_{id} - \mu_i)^2}\sqrt{\sum_{d=1}^{p}(o_{jd} - \mu_j)^2}} \tag{3.2}$$

donde μ_{oi} y μ_{oj} son las medias de $\vec{O}_i$ y $\vec{O}_j$, respectivamente. Este coeficiente entiende a cada objeto como una variable aleatoria con p observaciones y mide la similitud entre dos objetos calculando la relación lineal entre las distribuciones de sus dos variables aleatorias correspondientes.

El coeficiente de Pearson es ampliamente utilizado y bastante efectivo como medida de similitud entre los datos de expresión genética (Jiang *et al.*, 2003) (Tang *et al.*, 2001) (Tang y Zhang, 2002) (Yang *et al.*, 2002). Sin embargo, estudios empíricos han mostrado que no es robusto frente a los *outliers* (Heyer *et al.*, 1999), es decir, si dos patrones comparten un valle o un pico, la correlación estará dominada por esa característica aunque los patrones sean completamente distintos en el resto de las mismas. Este hecho provocó la aparición de otras medidas para solucionar este problema. Una de ellas fue la *jackknife's correlation* o correlación de la navaja (Efron, 1982) (Heyer *et al.*, 1999):

$$Jackknife(O_i, O_j) = min\{\rho_{ij}^{(1)}, \ldots, \rho_{ij}^{(l)}, \ldots, \rho_{ij}^{(p)}\} \tag{3.3}$$

donde $\rho_{ij}^{(l)}$ es el coeficiente de correlación de Pearson de los objetos O_i y O_j, sin tener en cuenta la característica l-ésima. De esta forma, se evita el efecto dominante de un sólo outlier común. Las formas más generales de la correlación de la navaja para soportar la acción de más de un outlier pueden ser derivadas de la fórmula anterior de forma sencilla, probando distintas combinaciones de eliminación de características. El problema es que dicha generalización implica, desde el punto de vista computacional, un gran aumento del tiempo de ejecución y de los recursos necesarios.

Otro problema del coeficiente de Pearson es que asume una aproximación de la distribución gaussiana para los puntos, por lo que no es robusto para distribuciones no gaussianas (David, 2001) (Dhaeseleer *et al.*, 1998). Para solucionar este contratiempo, se propuso la *correlación de Spearman*, que se diferencia de la de Pearson en que utiliza valores medidos a nivel de una escala ordinal. Es decir, cada valor numérico de expresión o_{id} se sustituye por la posición que ocupa dicho valor entre el resto, r_{id}. Por ejemplo, $o_{id} = 3$ si o_{id} es el tercer valor más grande entre o_{ik}, con $1 \leq k \leq p$. El coeficiente de Spearman no requiere que la distribución gaussiana sea asumida y es más robusto frente a los outliers. Sin embargo, a consecuencia del ranking de valores, se pierde información.

La mayoría de algoritmos de clustering mencionados en este capítulo utilizan la distancia euclídea o el coeficiente de correlación de Pearson como medidas de proximidad. Cuando se aplica la distancia euclídea, los datos de la matriz de expresión suelen estandarizarse. Supongamos que O_i' y O_j' son los objetos estandarizados de O_i y O_j, se demuestra que:

$$Pearson(O_i, O_j) = Pearson(O_i', O_j') \tag{3.4}$$

$$Euclidea(O_i', O_j') = \sqrt{2p}(\sqrt{1 - Pearson(O_i', O_j')}) \tag{3.5}$$

Estas dos ecuaciones revelan la consistencia entre el coeficiente de correlación de Pearson y la distancia euclídea, después de la estandarización de los datos. Es por ello que en este caso podemos esperar que la efectividad del algoritmo de clustering sea similar, independientemente de elegir una u otra medida de similitud.

3.2.3. Tipos de algoritmos de clustering

Normalmente, un microarray consta de entre 10^3 y 10^4 genes, y se espera que este número llegue al orden de 10^6. Sin embargo, el número de condi-

ciones experimentales es generalmente menor que 100. Esta configuración de los atributos y ejemplos hace que sea significativo aplicar las técnicas de clustering tanto a genes como a condiciones.

Por un lado, tenemos las técnicas de clustering basadas en genes. Los genes son agrupados en clusters dependiendo de la evolución de sus valores de expresión a lo largo de todas las condiciones experimentales. En el caso contrario, tenemos las técnicas de clustering basadas en condiciones, en las que éstas se agrupan en clusters similares. Cada cluster debería corresponder, en este caso, a un determinado fenotipo macroscópico, es decir, un tipo de cáncer, un tipo de tejido o un síndrome clínico determinado.

Algunos algoritmos de clustering, como el *K-means* o las técnicas jerárquicas, pueden ser usados indistintamente para agrupar genes o condiciones. Pero en la mayoría de los casos esta distinción implica cambios en las tareas a realizar para la agrupación de objetos similares.

Estas dos técnicas necesitan ser analizadas desde distinto ángulo y requieren técnicas computacionales bien diferenciadas.

3.3. Clustering sobre genes

3.3.1. Problemática

El principal objetivo de aplicar técnicas de clustering sobre los genes que forman un microarray es el de agrupar aquellos genes cuyos valores de expresión se comporten de manera similar a lo largo de las condiciones utilizadas en los experimentos. Se dice que los genes en los clusters así obtenidos se co-expresan, y esta característica es muy importante a la hora de la identificación de genes cooperantes en rutas metabólicas (*metabolic pathways*, relaciones entre los distintos componentes que intervienen en un proceso bioquímico). Revelar esta organización es crucial a la hora de obtener una visión global de la actividad celular. Y para ello, el detectar qué genes muestran valores de expresión similares bajo determinadas situaciones es un factor muy importante a tener en cuenta, aunque no el único. Es necesario además contar con otros datos, como por ejemplo, las interacciones entre los productos de dichos genes (Segal *et al.*, 2003).

Debido a las características especiales de las matrices de expresión genética y al carácter biológico de los estudios que se llevan a cabo, la aplicación de técnicas de clustering sobre genes traen consigo una problemática especial. En primer lugar, el clustering suele ser la primera fase en los procesos de minería de datos y obtención de conocimiento. El objetivo, en este caso particular, que persigue la creación de clusters es el de revelar la estructura

y distribución natural de los datos de expresión genética, y para dicha tarea no suele existir información de partida que pueda servir de ayuda. Por este motivo, un buen algoritmo de clustering debe depender lo menos posible de cualquier tipo de conocimiento previo al respecto. Por ejemplo, un algoritmo de clustering que pueda estimar el "verdadero" número de clusters existente en una base de datos será más favorable que otro que necesite un número predeterminado de clusters que crear. A este respecto, la existencia de parámetros de usuario constituye una problemática. Cuanto mayor es el número de parámetros más dependencia tendrá el resultado final de la distinta combinación de los valores de éstos y, por lo tanto, será más complejo seleccionar correctamente dichos valores.

Con respecto a los datos de entrada, los microarrays suelen tener una gran cantidad de ruido debido al complejo proceso de obtención de los mismos. Es por ello que los algoritmos de clustering que se dediquen al análisis de las matrices de expresión deben ser capaces de extraer conocimiento útil a pesar de la existencia de un gran nivel de ruido en la fuente de datos. Además, dicha fuente de datos suele constar de un gran número de atributos (genes). El tamaño de estas matrices implica que sea absolutamente necesario el minimizar los tiempos de ejecución de los programas resultantes y la cantidad de recursos hardware necesarios.

En relación a los resultados finales, hemos de llamar la atención sobre el hecho de que los destinatarios finales de estos algoritmos serán biólogos o médicos necesitados de la mayor cantidad de información posible. Por consiguiente, no sólo estarán interesados en conocer los clusters que se obtengan a partir de una determinada base de datos, sino que también sería interesante presentar las interrelaciones existentes entre los distintos clusters (qué clusters están más próximos unos de otros y cuáles más separados) y entre los genes de un mismo cluster. Además, la representación gráfica de los resultados se hace indispensable: visualización de los clusters obtenidos en el microarray, gráficas representando la evolución de los valores de expresión de los genes de un mismo cluster, etc (Shamir *et al.*, 2005) (Kano *et al.*, 2003).

A continuación, se expone una revisión de las principales técnicas de clustering aplicadas a la agrupación de genes. Para cada algoritmo se explicarán las ideas fundamentales y se repasarán sus características más trascendentes.

3.3.2. K-means

El algoritmo *K-means* (McQueen, 1967) es un típico método de clustering basado en división. Dado un determinado valor K, el algoritmo divide

los datos en K grupos disjuntos optimizando la siguiente función:

$$E = \sum_{i=1}^{K} \sum_{O \in C_i} |O - \mu_i|^2 \tag{3.6}$$

donde O es un objeto en el cluster C_i y μ_i es el centroide del cluster C_i, es decir, la media de todos sus objetos. Por lo tanto, la función E intenta minimizar la suma del cuadrado de la distancia de los objetos al centro de sus clusters.

El algoritmo K-means es simple y rápido. Su complejidad es $O(l * k * n)$, siendo l el número de iteraciones, k el número de clusters y n el número de objetos (genes). Además, este algoritmo converge normalmente en un reducido número de iteraciones.

A pesar de estas ventajas, esta técnica de clustering también cuenta con importantes inconvenientes a la hora de ser aplicada para la agrupación de genes. En primer lugar, el número de clusters de una matriz de expresión genética no es un dato que se pueda conocer de antemano. Para solucionar este problema, algunos usuarios ejecutan el algoritmo repetidas veces con distintos valores de k, comparan los resultados y se quedan con el más óptimo. Pero este método deja de ser práctico cuando estamos trabajando con una matriz de datos extensa formada por miles de genes. En segundo lugar, los datos de expresión contienen normalmente una gran cantidad de ruido y el K-means obliga a cada gen a estar en un cluster, lo que aumenta la sensibilidad a dicho ruido (Smet *et al.*, 2002) (Sherlock, 2000).

Para solucionar estos inconvenientes, se han propuesto trabajos que mejoran al algoritmo K-means en ciertos aspectos (RalfHerwig *et al.*, 1999) (Heyer *et al.*, 1999) (Smet *et al.*, 2002). Estas propuestas utilizan normalmente parámetros globales para controlar la calidad de los clusters resultantes, por ejemplo, el máximo radio de un cluster o la mínima distancia entre clusters.

3.3.3. Self-organizing Maps

Teuvo Kohonen, (Kohonen, 1997), presentó un modelo de red neuronal basado en el funcionamiento de neuronas biológicas. La red neuronal diseñada posee la capacidad de formar mapas de características. El objetivo de Kohonen era demostrar que un estímulo externo por sí solo, suponiendo una estructura propia y una descripción funcional del comportamiento de la red, era suficiente para forzar la formación de los mapas. El modelo tiene dos variantes: LVQ (*Learning Vector Quantization*) y TPM (*Topology Preserving Maps*) o SOM (*Self Organizing Maps*).

El modelo SOM está basado en redes neuronales de una sola capa. Los objetos están presentes en la entrada y las neuronas de salida están organizadas en una simple estructura bidimensional en forma de red. Cada neurona de la red está asociada a un vector de referencia con una serie de valores o características y cada objeto de entrada se asocia a la neurona cuyo vector se parezca más a dicho objeto.

En el algoritmo que implementa esta red, los vectores de referencia se inicializan con valores aleatorios. Cada objeto de entrada actúa como un dato de entrenamiento, modelando el contenido de dichos vectores dentro del espacio vectorial y haciendo que, gracias a ese entrenamiento, estos vectores se ajusten a la distribución de la información que la red recibe como entrada. Cuando el entrenamiento es completado, los clusters resultantes se identifican relacionando todos los objetos de entrada con las neuronas de salida, cada una de ellas representando una determinada clase. De esta forma, se genera una representación más o menos fidedigna de un conjunto de datos en un espacio de dos o tres dimensiones, colocando a los clusters resultantes muy cercanos entre sí.

El proceso de entrenamiento proporciona un método más robusto que el K-means a la hora de aplicar el clustering a un conjunto de datos con gran cantidad de ruido (Tamayo *et al.*, 1999) (Herrero *et al.*, 2001). Sin embargo, SOM necesita parámetros de usuario para indicar el número de clusters y la estructura de la red neuronal. Estos dos parámetros van a influir en todo el proceso de aprendizaje, por lo que una mala combinación de sus valores puede provocar una representación menos fiel de la estructura natural de los datos de entrada. Otro problema achacable a este método, cuando es aplicado a datos de expresión genética, se da en aquellos casos en los que los datos de entrada contienen gran cantidad de información no relevante, es decir, genes con patrones de comportamiento constantes o sin mucha variación. SOM generará una salida en la que este tipo de datos poblará la gran mayoría de los clusters (Herrero *et al.*, 2001). Así pues, en estos casos, SOM no es efectivo ya que muchos de los patrones de interés aparecerán en un reducido número de clusters, dificultando así su identificación.

3.3.4. Clustering jerárquico

Al contrario que los métodos de clustering basados en división, que descomponen los datos de entrada en un número determinado de particiones disjuntas, el clustering jerárquico genera una organización jerárquica de clusters anidados que es representada mediante una estructura de árbol llamada *dendograma*. Las ramas de este árbol proporcionan información de cómo

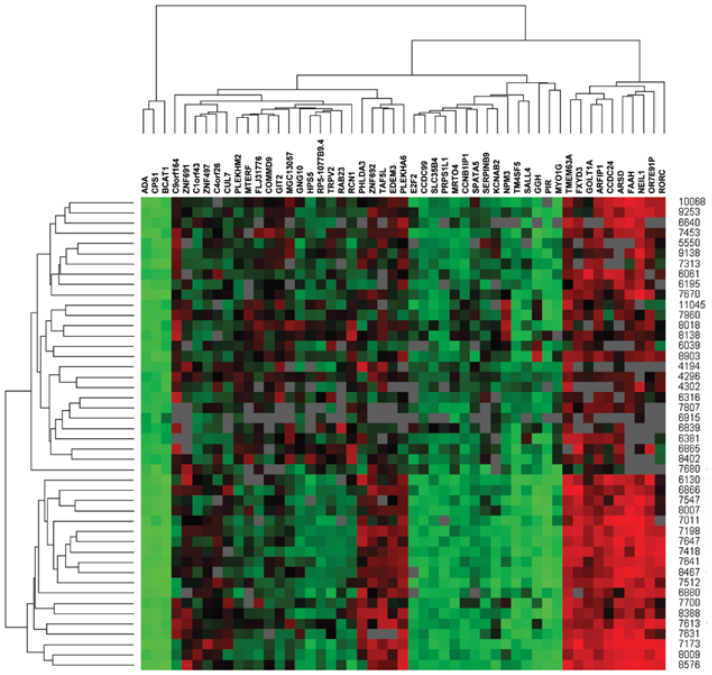

Figura 3.1: Ejemplo de una representación gráfica de clustering jerárquico a partir del método UPGMA. Esta imagen representa la agrupación de 51 genes con respecto a 46 tumores y ha sido extraída del trabajo de Karlsson et al. (Karlsson *et al.*, 2008) .

se formaron los clusters y la distancia entre las mismas muestran el grado de similitud entre los distintos grupos de objetos. Dependiendo del nivel del dendograma, se obtendrá un número u otro de clusters. Las técnicas de clustering jerárquico pueden ser divididas en dos categorías: basadas en la aglomeración y basadas en la división, y se diferencian en la forma en que el dendograma es creado. Los algoritmos aglomerativos utilizan una metodología *bottom-up*: al principio, cada objeto es asociado a un cluster individual y en cada paso se van uniendo aquellos clusters más cercanos hasta que todos los grupos acaban unidos en un sólo cluster. Con una aproximación contraria, *top-down*, los algoritmos basados en la división comienzan con un solo cluster en el que se encuentran todos los objetos. Mediante sucesivas divisiones se van creando distintos clusters hasta que cada objeto esté en un cluster distinto. Para el primer tipo de técnicas se utilizan distintas medidas de proximidad entre clusters (Dubes y Jain, 1988) (Kaufman y Rousseeuw, 1990): mínima varianza, single link (mínima distancia entre todos los pares de objetos de cada cluster), complete link (máxima distancia entre todos

los pares de objetos de cada cluster), etc. Para llevar a cabo las divisiones efectuadas sobre los clusters en el método *top-down* se pueden usar desde métodos heurísticos, por ejemplo jugando con la probabilidad de pertenecer a un nuevo cluster (Alon *et al.*, 1999), hasta otros basados en teoría de grafos.

Eisen et al. (Eisen *et al.*, 1998) utilizó un algoritmo basado en aglomeración llamado UPGMA (*Unweighted Pair Group Method with Arithmetic Mean*), adoptando también un método para representar gráficamente los clusters resultantes. En este método, cada celda del microarray es coloreada según su nivel de fluorescencia. Después de aplicarse el algoritmo de clustering, la matriz coloreada se ordena siguiendo unas reglas basadas en el dendograma obtenido. Al final, la matriz de expresión original es representada por una tabla coloreada en la que las zonas de un mismo color representan a grupos de genes que se expresan de la misma forma bajo una serie de condiciones experimentales (ver ejemplo en figura 3.1). Así, se concluye que el clustering jerárquico no sólo es un método para agrupar genes que se co-expresan, además proporciona una forma natural de representar gráficamente los datos de entrada.

Sin embargo, los métodos jerárquicos también tienen defectos. Uno de ellos es su falta de robustez (Tamayo *et al.*, 1999), es decir, una pequeña perturbación en los datos de entrada o una mala decisión tomada en los pasos iniciales puede modificar profundamente la estructura del dendograma final. Otra desventaja es la complejidad computacional que representa el hecho de construir de forma completa la estructura del árbol. Para crear un dendograma, en el que cada hoja representa un objeto, estando todos unidos en un sólo cluster en la raíz, es necesario llevar a cabo $\frac{n^2-n}{2}$ divisiones. La complejidad de un algoritmo jerárquico aglomerativo típico es de $O(n^2 logn)$ (Jain *et al.*, 1999).

3.3.5. Técnicas basadas en teoría de grafos

Dada una base de datos X, se puede construir una matriz de proximidad P en la que $P[i,j] = proximidad(O_i, O_j)$, y un grafo de proximidad $G(V,E)$ en el que cada dato se corresponde con un vértice.

Algunas técnicas de clustering representan los datos de expresión genética y los grados de proximidad entre las expresiones de los genes en forma de grafo para después aplicar técnicas asociadas a estas representaciones. A continuación se muestran dos de los métodos basados en grafos más conocidos.

CLICK

El algoritmo CLICK (*CLuster Identification via Connectivity Kernels*) (Shamir y Sharan, 2000) identifica como clusters aquellos componentes altamente relacionados en grafos de proximidad. CLICK juega con el supuesto probabilístico de que, tras la normalización de los datos, los valores de similitud entre ellos siguen una distribución normal. Siguiendo este supuesto, el peso w_{ij} de una arista se define como la probabilidad de que los vértices i y j estén en el mismo cluster.

CLICK sigue un proceso iterativo, al que le precede el pre-procesamiento de los datos: normalización y cálculo por parejas de la similitud. En este proceso iterativo, se eliminan aquellas aristas cuya probabilidad sea menor que un umbral prefijado. El resultado de este proceso es la división del grafo inicial en grupos de vértices conectados entre sí. Estos clusters son valorados y refinados al final de cada iteración. El proceso llega a su final cuando ningún sub-grafo pueda ser ya dividido.

En el trabajo de Shamir et al., los autores comparan los resultados obtenidos con CLICK en dos bases de datos públicas con aquellos obtenidos por un algoritmo basado en SOM (Tamayo *et al.*, 1999) y con el algoritmo jerárquico de Eisen (Eisen *et al.*, 1998), respectivamente. En los dos casos, los clusters obtenidos con CLICK demostraron tener un mayor grado de calidad en términos de homogeneidad y separación. Sin embargo, CLICK no da garantías de obtener siempre resultados coherentes, por ejemplo: es posible obtener un resultado tan extraño como una base de datos separada en dos grupos, uno en el que están todos los elementos y otro en el que encontramos sólo dos mínimos. Tampoco es sensible al solapamiento de los grupos de genes. Lo más probable es que CLICK represente dos grupos de genes solapados como un sólo grupo de genes conectados entre sí.

CAST

Ben-Dor et al. (Ben-Dor *et al.*, 1999) parte también de un determinado modelo probabilístico que asume que existe una verdadera partición biológica de los genes, en base a sus funcionalidades, que da lugar a clusters disjuntos. Cada cluster tiene un determinado patrón de expresión y los genes incluidos muestran pequeñas variaciones de ese patrón de comportamiento. Pero durante el proceso de extracción de las medidas de expresión se producen errores. Así pues, si se crea una matriz de similitud de todos los genes, el resultado mostrará una contaminación de las verdaderas relaciones entre dichos genes.

Los autores representan los datos de expresión mediante un grafo indi-

recto, G, en el que los nodos son los genes y las aristas conectan a los genes con similares patrones de expresión. Los verdaderos clusters, ocultos tras los errores de medición, se pueden expresar mediante un grafo, H, que es una unión disjunta de sub-grafos completos, representando cada uno de ellos a un cluster. El grafo de similitud G se puede derivar de H proporcionando a cada arista una probabilidad α.

CAST (*Cluster Affinity Search Technique*) es un eficiente algoritmo que obtiene un grafo G con muchas probabilidades de ser casi tan bueno como el grafo real H. La entrada es una matriz de similitud S con $S(i,j) \in [0,1]$ y un umbral de afinidad t. En cada iteración, el algoritmo busca un solo cluster, C_{open}. Cada elemento x tiene un valor de afinidad $\alpha(x)$ con respecto a C_{open}, $\alpha(x) = \sum_{y \in C_{open}} S(x,y)$. Un elemento x será considerado de gran afinidad si satisface la siguiente condición : $\alpha(x) \geq t \mid C_{open} \mid$. En otro caso, se dice que x tiene un pequeño nivel de afinidad. CAST va creando el cluster añadiendo elementos de gran afinidad y eliminando elementos de poca afinidad. Cuando el proceso se estabiliza, C_{open} es considerado un cluster completo. El algoritmo termina cuando cada uno de los elementos que forman los datos de entrada está asignado a un determinado cluster.

El umbral de afinidad t es en realidad la media de los valores de similitud entre los pares de elementos que forman el cluster. CAST especifica la calidad deseada de los clusters finales mediante t y lleva a cabo un proceso de búsqueda heurística para identificarlos. Por lo tanto, CAST no depende de un número de clusters predeterminado por el usuario y es capaz de tratar los outliers de forma efectiva. Sin embargo, el mayor problema es saber determinar un buen valor para el parámetro t.

3.3.6. Clustering basado en modelos

Las técnicas de clustering basadas en modelos (Fraley y Raftery, 1998) (Yeung *et al.*, 2001) (Ghosh y Chinnaiyan, 2002) (McLachlan *et al.*, 2002) proporcionan las herramientas estadísticas necesarias para modelar la estructura de los clusters de los datos de expresión genética. Estos métodos parten de la suposición de que los datos de entrada derivan de una mezcla finita de distribuciones probabilísticas, cuyos componentes se corresponden con diferentes clusters.

El objetivo es estimar los parámetros $\Theta = \{\theta_i \mid 1 \leq i \leq k\}$ y $\Gamma = \{\gamma_r^i \mid 1 \leq i \leq k, 1 \leq r \leq n\}$ que maximizan la probabilidad $L_{mix}(\Theta, \Gamma) = \sum_{i=1}^{k} \gamma_r^i f_i(x_r \mid \theta_i)$, donde n es el número de datos, k es el número de componentes, x_r es un dato, $f_i(x_r \mid \theta_i)$ es la función de densidad del dato x_r del componente C_i con un conjunto indeterminado de parámetros de modelado

θ_i, y γ_r^i representa la probabilidad de que x_r pertenezca a la componente C_i. Normalmente, los parámetros Γ y Θ son estimados por el algoritmo EM (Dempster *et al.*, 1977). Dicho algoritmo itera entre acciones de tentativa E y acciones de maximización M. En la parte de tentativa, los parámetros ocultos Γ son estimados condicionalmente a partir de los datos y de los ya estimados parámetros Θ. En los pasos M, los parámetros de modelado Θ son estimados para maximizar la probabilidad del total de los datos a partir de los parámetros estimados γ_r^i. Cuando el algoritmo EM converge, cada dato es asignado a un componente (cluster) con la máxima probabilidad.

Una importante ventaja de estas técnicas es que proporcionan una probabilidad estimada γ_r^i de que un objeto i pertenezca a un cluster k. Normalmente, en los datos de expresión genética existe gran conectividad, de tal forma que un solo gen puede tener un gran nivel de afinidad con más de un cluster. En estos casos, un modelo probabilístico puede ser muy útil. Sin embargo, estos modelos trabajan con la suposición de que los datos siguen una determina distribución y eso no tiene por qué ser cierto en todos los casos. El modelado de los datos de expresión genética es una labor en desarrollo por parte de muchos investigadores y actualmente no hay ningún modelo efectivo que represente estos datos.

3.4. Clustering sobre condiciones experimentales

3.4.1. Problemática

En una matriz de expresión genética, nos encontramos frecuentemente ciertos fenotipos macroscópicos o características comunes de las condiciones relacionadas con algunas enfermedades o drogas. El objetivo fundamental del clustering sobre las condiciones en las bases de datos de expresión es detectar aquellas condiciones con idéntico fenotipo o características.

Estudios realizados (Golub *et al.*, 1999) demuestran que las características comunes o fenotipo de las condiciones experimentales pueden ser solo discriminadas por un pequeño grupo de genes con niveles de expresión muy similares llamados *genes informativos*. El resto de genes son irrelevantes en cuanto al agrupamiento de las condiciones de interés y son considerados como ruido.

Aunque los métodos convencionales de clustering, como el K-means, SOM y técnicas jerárquicas pueden ser directamente aplicados para agrupar condiciones, utilizando a todos los genes como atributos, la proporción de genes informativos con respecto a los genes irrelevantes suele ser normalmente menor que 1:10 y, por consiguiente, puede degradar la calidad de los

resultados finales (Xing y Karp, 2001) (Tang *et al.*, 2003). Por ello, deberían aplicarse métodos concretos para identificar a los genes informativos y reducir la dimensionalidad de atributos irrelevantes de las bases de datos. Los métodos existentes de selección de genes informativos para obtener grupos de condiciones experimentales pueden dividirse en dos grandes categorías: *análisis supervisado* y *análisis no supervisado*.

3.4.2. Clustering basado en la selección supervisada de genes

Las técnicas supervisadas parten de la suposición de que en las bases de datos cada condición está asociada a información concerniente a su fenotipo, es decir, etiquetas como "enfermedad" o "normal". Partiendo de esta información, se puede crear un clasificador que utilice únicamente los genes informativos. Este clasificador es usado posteriormente para agrupar las condiciones experimentales en función de su fenotipo e incluso para predecir las etiquetas de futuras condiciones que se añadan a la base de datos. Los pasos principales para construir este clasificador son:

- *Entrenamiento para la selección de condiciones.* En este primer paso, se selecciona un subconjunto de condiciones para crear la base de datos de entrenamiento. Puesto que el número de condiciones es en la mayoría de ocasiones limitado (menos de 100), el subconjunto de entrenamiento normalmente será de un orden de magnitud similar al de la base de datos original.

- *Selección de genes informativos.* El objetivo es seleccionar aquellos genes cuyos patrones de expresión puedan diferenciar de forma clara a los fenotipos de las condiciones. Por ejemplo, la expresión de un gen es uniformemente alta en una condición y sin embargo, uniformemente baja en otra (Golub *et al.*, 1999). Se han desarrollado distintas técnicas para seleccionar los genes informativos: técnicas basadas en la vecindad (Golub *et al.*, 1999), métodos de aprendizaje supervisado como el SVM (*Support Vector Machine*) (Brown *et al.*, 2000), además de métodos basados en ranking (Ben-Dor *et al.*, 2001).

- *Clustering de condiciones y clasificación.* La última parte consiste en aplicar técnicas de clustering sobre todas las condiciones experimentales, utilizando para ello como atributos a los genes informativos seleccionados en la fase anterior. A pesar de que el número de estos atributos suele ser no muy grande, entre 50 y 200 (Golub *et al.*, 1999), algoritmos tales como el K-means o el SOM son usados en estos casos.

Las futuras incorporaciones de condiciones a la base de datos pueden ser clasificadas a partir de los genes informativos y métodos de aprendizaje supervisado.

3.4.3. Clustering no supervisado y selección de genes informativos

Los métodos de clustering no supervisados no presuponen una información sobre el fenotipo asociada a cada condición experimental. Llevan a cabo un descubrimiento automático de los fenotipos en las bases de datos, lo que contribuye de manera importante al análisis de datos de expresión y a la exploración de estructuras desconocidas en el dominio de las condiciones.

Estas técnicas son más complejas que las supervisadas, ya que no partimos de una base de datos de entrenamiento como guía para la selección de genes informativos. Además, existen dos factores importantes, que influyen en esta complejidad y que hay que tener muy en cuenta:

- Aunque el número de ejemplos (condiciones) es muy limitado, el volumen de atributos (genes) es muy grande. Esto hace que las bases de datos sean normalmente muy dispersas. Por ello, la estructura de clases de las condiciones no puede ser detectada de manera apropiada por las técnicas convencionales, como aquellas basadas en áreas de densidad.
- La mayoría de los genes no son de interés. Menos del 10 % manifiestan de forma comprensible los fenotipos ocultos, y por lo tanto, están inmersos en grandes cantidades de ruido (Golub *et al.*, 1999). La incertidumbre sobre qué genes son en verdad relevantes hace complicado seleccionar los llamados genes informativos.

Existen dos estrategias principales para solucionar estos problemas planteados:

Selección de genes no supervisada. Esta estrategia separa en procesos independientes la selección de genes con el clustering de condiciones, es decir, en primer lugar la dimensión de los atributos (genes) es reducida y posteriormente se aplican técnicas convencionales de clustering. Ya que no se parte de una base de datos de entrenamiento, la selección de genes ha de depender de modelos estadísticos que analicen la varianza de los datos de expresión genética.

Clustering Interrelacionado. Si prestamos atención a la selección de genes informativos y el clustering de condiciones, podremos observar que son

dos tareas muy relacionadas entre sí. Una vez que se han detectado los genes informativos es muy sencillo aplicar técnicas convencionales para agrupar las condiciones experimentales. Aunque también podríamos hacerlo al revés, es decir, partir de un agrupamiento correcto de las condiciones para, utilizando métodos supervisados como el *t-test scores* o *separation scores* (Thomas *et al.*, 2001), establecer un rango de los genes en función de su relevancia en las particiones realizadas. Los genes más relevantes serán considerados como genes informativos. En función de esta última observación, se puede considerar el hecho de utilizar de forma dinámica esta relación entre genes y condiciones e, iterativamente, combinar los procesos de clustering y de selección de genes. Aunque la partición exacta de las condiciones no se conozca por adelantado, en cada iteración podemos trabajar con una partición aproximada que se acerque al objetivo final. Esta aproximación de los clusters finales va a permitir la selección de un subconjunto, moderadamente bueno, de genes que acercará más a la partición aproximada al resultado final en la siguiente iteración, y así sucesivamente.

3.5. Conclusiones

En este capítulo se han revisado las técnicas más relevantes de clustering aplicadas a datos de expresión genética. Uno de los objetivos es el de agrupar genes que se co-expresan bajo una serie de condiciones experimentales, intentando evitar en lo posible el ruido. La detección de clusters de genes proporciona una útil base de conocimiento para las investigaciones sobre regulación y funcionalidad genética.

Inicialmente, algunos de los algoritmos de clustering convencionales fueron aplicados a estos datos, como el K-means, SOM o las técnicas jerárquicas (UPGMA), probando ser útiles pero no totalmente adecuados a este tipo de información tan peculiar. Por ello, aparecieron nuevas técnicas más adaptadas a las características de los datos de expresión, como CLICK y CAST.

Con respecto al clustering sobre condiciones experimentales, el objetivo principal es el de encontrar características o fenotipos comunes entre dichas condiciones. Los métodos existentes pueden ser divididos en dos grandes categorías:

- Técnicas de clustering basadas en selección supervisada de genes
- clustering y selección de genes no supervisadas: selección de genes no supervisada y clustering interrelacionado.

Debido a que el porcentaje de genes informativos en las bases de datos de expresión suele ser bajo, uno de los mayores retos de este tipo de técnicas es el de seleccionar de forma correcta cuáles son los genes relevantes que nos van a ayudar a establecer la estructura de fenotipos en el dominio de las condiciones experimentales.

Capítulo 4

Introducción al biclustering

Lo poco que he aprendido carece de valor,
comparado con lo que ignoro y no
desespero en aprender.
R. Descartes.

Este capítulo es una continuación natural del capítulo anterior y en él se introducen todos los aspectos relacionados con las técnicas de biclustering. En primer lugar, se explica su origen, el cual está relacionado con una debilidad de las técnicas de clustering cuando son aplicadas a bases de datos de expresión genética. Posteriormente, la problemática general es definida, al igual que los distintos tipos de biclusters que podemos encontrar. Se lleva a cabo una clasificación general de las técnicas existentes de biclustering y se presenta una breve introducción de las más relevantes. Para finalizar, el caso concreto de las técnicas de biclustering aplicadas a bases de datos binarios es analizado.

4.1. Introducción: clustering vs biclustering

Según lo visto en el capítulo anterior, los datos sobre expresión genética son organizados en matrices. Cada elemento de esa matriz representa el nivel de expresión de un gen bajo una determinada condición. Dicho nivel de expresión es representado a su vez por un número real, que es el logaritmo de la abundancia relativa de ARNm del gen bajo dicha condición.

Las matrices de expresión genética han sido analizadas en dos dimensiones de forma exhaustiva: en la dimensión de los genes y en la dimensión de las condiciones. Es decir, análisis de patrones de expresión de genes comparando filas en la matriz y análisis de patrones de expresión de condiciones comparando columnas. El objetivo, agrupar genes o condiciones que tengan

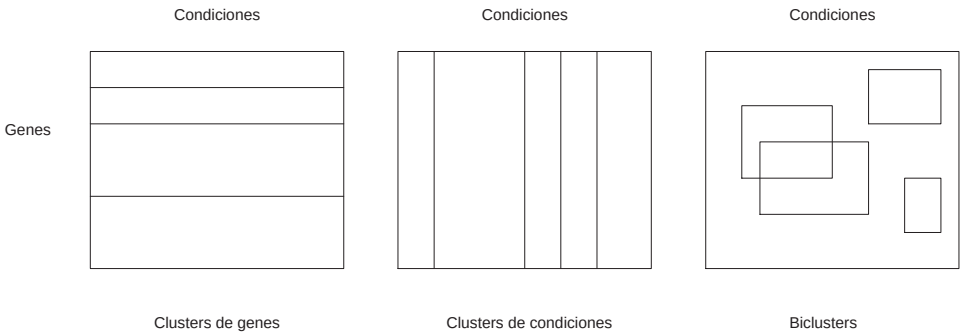

Figura 4.1: Clusters y biclusters de una matriz de expresión genética.

un significado biológico. Para llevar a cabo dicho objetivo se ha utilizado principalmente una de las técnicas más conocidas relacionadas con la minería de datos: el clustering. Sin embargo, la aplicación de algoritmos de clustering sobre datos de expresión genética presenta una importante dificultad. Muchos patrones de actividad de grupos de genes solo se presentan bajo un determinado conjunto de condiciones experimentales. De hecho, el conocimiento actual sobre los procesos celulares nos hace esperar que un subconjunto de genes sea co-regulado y co-expresado solo bajo ciertas condiciones experimentales, mientras que bajo otras condiciones estos genes pueden comportarse de forma independiente (Madeira y Oliveira, 2004). El descubrimiento de estos patrones locales de comportamiento puede ser la clave para descubrir rutas metabólicas que de otra manera serían difíciles de dilucidar. Por este motivo, el paradigma de las técnicas de clustering tuvo que evolucionar hacia métodos que permitieran descubrir patrones locales en datos de expresión genética (Ben-Dor *et al.*, 2002).

Las técnicas de clustering pueden ser aplicadas tanto a las filas como a las columnas de las matrices de expresión de forma separada. Sin embargo, las técnicas de biclustering pueden realizar el agrupamiento en las dos dimensiones de forma simultánea. Es decir, los métodos de clustering establecen un modelo global mientras que las técnicas de biclustering generan un modelo local. Cuando se aplica un algoritmo de clustering, cada gen, en un cluster de genes, es definido usando para ello todas las condiciones experimentales. De forma similar, cada condición, en un cluster de condiciones, se caracteriza mediante el análisis de todos los genes de la matriz. En el caso de los biclusters, cada gen es seleccionado utilizando solo para ello un subconjunto de condiciones y en la elección de cada condición de un bicluster solo intervienen un subgrupo de genes. Por lo tanto, el objetivo principal de las técnicas de biclustering consiste en identificar subgrupos de

genes y subgrupos de condiciones aplicando clustering sobre las filas y columnas de forma simultánea, en vez de analizar por separado cada una de estas dimensiones.

El concepto de bicluster establece un marco computacional más flexible. En la figura 4.1 vemos que los clusters forman grupos disjuntos tanto de filas como de columnas. En el caso de los biclusters, las restricciones son mucho menores ya que se tratan de submatrices que no tienen que ser ni exclusivas ni exhaustivas, es decir, un gen o una condición podría pertenecer a un bicluster, a más de uno o a ninguno. Así, la falta de reglas estructurales en los biclusters permite una gran libertad en detrimento de altas probabilidades de obtener resultados con un gran índice de solapamiento. Por eso, las técnicas de biclustering tienen que garantizar que los resultados obtenidos tengan cierta validez y significado. Por consiguiente, otro de los objetivos importantes de este tipo de técnicas es hacer acompañar a los resultados de medidas de calidad, tales como modelos estadísticos, puntuaciones heurísticas o análisis biológico de los resultados, que nos garanticen que los biclusters obtenidos sean significativos.

4.2. Definiciones y formulación del problema

4.2.1. Matriz de expresión genética y bicluster

El objeto de estudio de las técnicas de biclustering será una matriz n x m en la que cada elemento a_{ij} será, generalmente, un número real que representa el nivel de expresión del gen i bajo la condición experimental j.

Aunque la mayoría de algoritmos de biclustering son ideados para analizar datos de expresión genética, estas técnicas también tienen otras aplicaciones. Así, se considera el caso general de una matriz A, con un conjunto de filas X y un conjunto de columnas Y, en la que los elementos a_{ij} corresponden a un valor que representa la relación entre la fila i y la columna j.

Se puede definir la matriz A como un conjunto de filas $X = \{x_1, \ldots, x_n\}$ y un conjunto de columnas $Y = \{y_1, \ldots, y_m\}$. De esa forma, la notación (X, Y) se utilizará para hacer referencia a la matriz A. Si $I \subset X$ y $J \subset Y$ son subconjuntos de filas y columnas respectivamente, $A_{IJ} = (I, J)$ representa a la submatriz A_{IJ} de A, conteniendo únicamente los elementos a_{ij} pertenecientes al subconjunto de filas I y al subconjunto de columnas J.

Dada la matriz A, un cluster de filas es un subconjunto de filas que presentan un comportamiento similar bajo todas las columnas de la matriz. Se puede definir un cluster de filas como $A_{IY} = (I, Y)$, donde $I = \{i_1, \ldots, i_k\}$

es un subconjunto de filas ($I \subseteq X$ y $k \leq n$). Es decir, un cluster de filas (I, Y) puede ser definido también como una submatriz k x m de la matriz A. De las misma forma, un cluster de columnas puede ser definido como $A_{XJ} = (X, J)$, donde $J = \{j_1, \ldots, j_s\}$ es un subconjunto de columnas ($J \subseteq Y$ y $s \leq m$), o lo que es lo mismo, una submatriz n x k de la matriz de datos A.

Un bicluster $A_{IJ} = (I, J)$ es un subconjunto de filas y un subconjunto de columnas donde $I = \{i_1, \ldots, i_k\}$ es un subconjunto de filas ($I \subseteq X$ y $k \leq n$) y $J = \{j_1, \ldots, j_s\}$ es un subconjunto de columnas ($J \subseteq Y$ y $s \leq m$). Igual que en los casos anteriores, también podemos definir al bicluster como una submatriz kxs de la matriz A.

En conclusión, el problema específico al que se tienen que enfrentar los algoritmos de biclustering es el siguiente: dada una matriz A se desea identificar un grupo de biclusters, $B_k = (I_k, J_k)$, de tal forma que cada bicluster B_k satisfaga ciertas características de homogeneidad. Las características de homogeneidad son muy variadas y de ellas depende en gran medida el algoritmo de biclustering. Este hecho será estudiado más adelante.

4.2.2. Grafos bipartitos y matrices de datos

Se puede establecer una relación interesante entre las matrices de datos y la teoría de grafos. Una matriz de datos puede ser representada como un grafo bipartito, $G = (V, E)$, donde V es el conjunto de vértices del grafo y E es el conjunto de aristas. Se dice que un grafo es bipartito cuando sus vértices pueden ser particionados en dos conjuntos, L y R, de tal forma que cada arista en E tiene exactamente un extremo en L y el otro en R: $V = L \bigcup R$.

La matriz $A = (X, Y)$ se puede representar como un grafo bipartito en el que cada nodo $n_i \in L$ corresponde a una fila y cada nodo $n_j \in R$ corresponde a una columna. La arista entre n_i y n_j tiene un peso, a_{ij}, representando al elemento de la matriz que se encuentra en la fila i columna j (nivel de activación genética en las matrices de expresión).

Esta conexión entre el mundo de los grafos y las matrices ha derivado en algoritmos de biclustering muy interesantes que serán comentados más adelante.

4.2.3. Complejidad

Aunque la complejidad del problema de encontrar biclusters depende en gran parte de la formulación de dicho problema y en la función escogida para evaluar la calidad de cada bicluster, la mayoría de las variantes interesantes de dicho problema tienen una complejidad NP-completa.

En su forma más simple, la matriz A es una matriz binaria en la que cada elemento a_{ij} tiene el valor 0 ó 1. En este caso particular, un bicluster corresponde a un subgrafo en el correspondiente grafo bipartito. Encontrar un bicluster de tamaño máximo es, por lo tanto, equivalente a encontrar un subgrafo con el máximo número de aristas en un grafo bipartito, problema que tiene una complejidad NP-completa (Peeters, 2003).

En casos más complejos, en los que los valores numéricos de la matriz A son tomados en cuenta para determinar la calidad de un bicluster, la complejidad no es menor que la comentada anteriormente. Es por ello que la gran mayoría de algoritmos utilizan técnicas heurísticas para encontrar biclusters, en muchos casos precedidas de un pre-procesamiento de los datos con el fin de hacer más evidentes los patrones de interés. Otros métodos evitan utilizar heurísticas, pero pueden exhibir un tiempo de ejecución exponencial en el peor de los casos.

4.2.4. Problemática

Las técnicas de biclustering se enfrentan a una problemática similar a la de las técnicas de clustering, problemática derivada del tipo de información que se analiza. Es decir, datos que contienen mucho ruido debido a los complejos métodos a partir de los cuales son obtenidos. Además, estas bases de datos contienen un gran número de atributos, del orden de miles. Por ello, el algoritmo de biclustering ha de ser robusto al ruido encontrado en los datos, además de minimizar al máximo el tiempo de ejecución y los recursos hardware necesarios.

Además de estos problemas, la falta de restricciones estructurales en los biclusters proporciona más libertad a la hora de buscar posibles soluciones. Esta libertad provoca la posibilidad de obtener una gran cantidad de resultados, entre los que pueden existir gran cantidad de submatrices solapadas ó no relevantes desde el punto de vista de la información. Así, uno de los objetivos principales de estas técnicas tiene que ser el de definir un tipo concreto de bicluster objetivo y centrar la búsqueda en este tipo de resultados. Además, es necesario que incluyan alguna medida de calidad para desestimar resultados poco interesantes. En la mayoría de los casos estas medidas son estadísticas o matemáticas y son útiles para observar la coherencia numérica de un grupo de valores. Pero hemos de tener en cuenta que tratamos con datos de origen biológico y que los resultados finales han de ser relevantes o potencialmente interesantes para esta área de la ciencia. Es por ello que estas técnicas necesitan métodos de evaluación biológica.

Es importante resaltar el hecho de que, en referencia a los genes, se

tiene poca información sobre el mecanismo por el que éstos se rigen, por lo que no se pueden hacer suposiciones con este tipo de datos. Por lo tanto, los algoritmos de biclustering deberían tener el menor número de parámetros de usuario que condicionara los resultados y que ocultara la verdadera realidad de los datos de expresión genética.

Por último indicar que, como se ha comentado anteriormente, la complejidad de estos problemas suele ser NP-completa. Es por ello que se ha de intentar reducir el espacio de búsqueda a la hora de encontrar las soluciones si no queremos que el tiempo de ejecución o los recursos hardware necesarios sean demasiados costosos.

4.3. Características de los algoritmos de biclustering

4.3.1. Introducción

Desde que el análisis de matrices de expresión genética comenzó a ser un asunto de relevancia, son muchos los algoritmos que se han ideado para solucionar este problema. Estos algoritmos son muy variados y utilizan técnicas de muy diversa índole para solucionar el mismo dilema. Esta falta de unidad de criterios viene determinada por un desconocimiento de cómo han de ser los biclusters objetivo. Es decir, los resultados obtenidos son difíciles de calificar ya que van a formar parte de un mundo complejo, la genética, en el que no es fácil comprobar de forma experimental unos resultados teóricos y en el que faltan todavía muchas piezas del puzzle por completar. Por este motivo, todavía no hay una metodología y unos criterios únicos a seguir a la hora de encontrar subgrupos de genes que evolucionen de la misma manera bajo un subgrupo de condiciones experimentales.

Dado el amplio abanico de posibles técnicas a usar, es necesario recoger algunas de las características más importantes de los distintos tipos de algoritmos existentes:

- El tipo de bicluster objetivo de la búsqueda. A su vez, esta característica viene determinada por la función de calidad utilizada que define el tipo de homogeneidad buscada en los biclusters.

- La estructura de biclusters generada. Algunos algoritmos encuentran solo un bicluster, otros un determinado número de submatrices no solapadas y la mayoría encuentran muchos biclusters solapados entre sí. Hemos de decir que el fenómeno del solape no es algo que se tenga

que desdeñar, ya que tiene su equivalente en la naturaleza: genes que participan en más de un proceso biológico bajo diferentes condiciones.

- El tipo de técnica utilizada. Algunas propuestas son algoritmos voraces mientras que otras utilizan métodos estocásticos.
- El tipo de evaluación de los resultados. He aquí uno de los aspectos más importantes y que más polémica suscita en los distintos ámbitos de la bioinformática. Hay gran variedad de métodos de evaluación, unos más eficaces que otros.
- El ámbito de aplicación. El biclustering no solo es aplicable a datos extraídos de microarrays. Esta técnica también puede ser aplicada a sistemas de recomendación, marketing o análisis de resultados electorales.

Los dos primeros aspectos serán comentados en los siguientes apartados.

4.3.2. Tipos de biclusters

Un criterio muy interesante para evaluar los distintos tipos de algoritmos de biclustering existentes tiene que ver con el tipo de bicluster que dichos algoritmos son capaces de encontrar. Las principales clases de biclusters son las siguientes (ver Fig. 4.2):

- Biclusters formados por valores constantes.
- Biclusters con valores constantes en filas o columnas.
- Biclusters con valores coherentes.
- Biclusters con evolución coherente de sus valores.

Los algoritmos de biclustering más simples identifican biclusters formados por valores constantes. En matrices de datos de expresión genética, estos biclusters representan grupos de genes con valores de expresión idénticos bajo un subconjunto de condiciones experimentales. Un bicluster perfecto es una submatriz (I, J) en la que todos sus valores son iguales para todo $i \in I$ y todo $j \in J$: $a_{ij} = \mu$. Aunque este bicluster ideal puede ser encontrado en algunas matrices de datos, normalmente están enmascarados por el ruido. Esto significa que los valores a_{ij} que podemos encontrar en los llamados biclusters constantes se presentan generalmente con esta forma: $\eta_{ij} + \mu$, donde η_{ij} es el ruido asociado al valor real μ de a_{ij}. Cuando el objetivo es encontrar biclusters constantes, es natural el considerar formas de reordenar filas

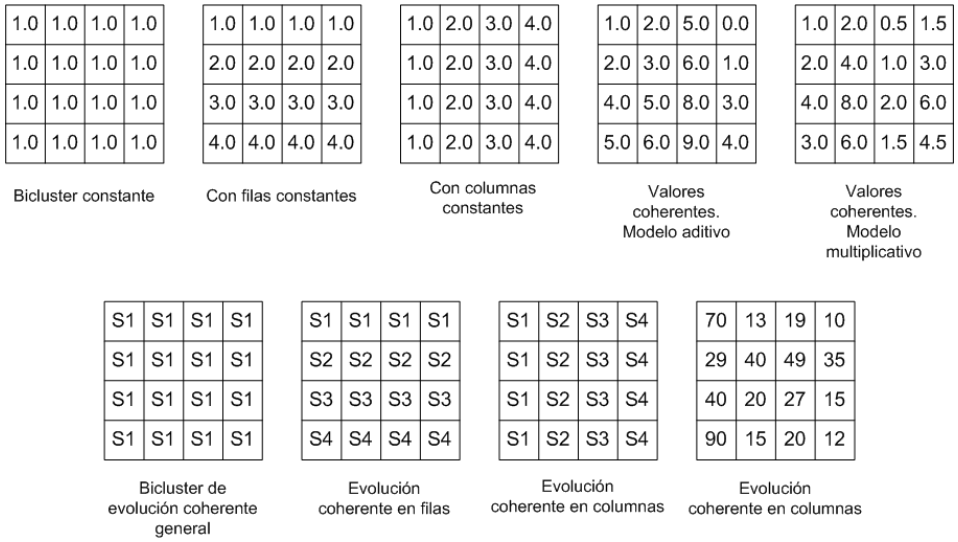

Figura 4.2: Distintos tipos de biclusters.

y columnas de la matriz de datos para agrupar aquéllas que sean similares. También es común el utilizar, como función de calidad, la varianza o alguna medida relacionada con ella.

Hay algoritmos que buscan subconjuntos de filas y subconjuntos de columnas con valores constantes en las filas o en las columnas. En el caso de matrices de expresión genética, un bicluster con valores constantes en las filas identifica un subconjunto de genes con expresiones similares a lo largo de un subconjunto de condiciones experimentales. El mismo razonamiento se puede aplicar para identificar un subconjunto de condiciones para las que un subconjunto de genes presentan valores de expresión similares, asumiendo que dichos valores cambian de condición a condición.

Un bicluster perfecto con valores constantes en sus filas es una submatriz (I, J), en la que todos sus valores pueden ser obtenidos utilizando una de las siguientes expresiones:

$$a_{ij} = \mu + \alpha_i \tag{4.1}$$

$$a_{ij} = \mu * \alpha_i \tag{4.2}$$

donde μ es el valor típico en el bicluster y α_i es la variación introducida por la fila i. Esta variación puede ser introducida a partir de un modelo aditivo o multiplicativo. La misma idea se puede aplicar a biclusters con valores constantes en las columnas (I, J), en los que los valores también siguen una de las siguientes ecuaciones:

$$a_{ij} = \mu + \beta_j \tag{4.3}$$

$$a_{ij} = \mu * \beta_j \tag{4.4}$$

donde en este caso β_j es el ajuste para la columna j. Un primer paso usual para identificar estos biclusters es normalizar las filas o las columnas de la matriz utilizando para ello la media de fila o de columna, respectivamente. Mediante esta práctica, este tipo de biclusters podrían transformarse en biclusters de valores constantes. En estos casos, la varianza no es suficiente para identificar de forma correcta estos resultados.

En el caso de matrices de expresión genética, el objetivo puede ser identificar biclusters más complejos en los que un subconjunto de genes y un subconjunto de condiciones tienen valores coherentes tanto en filas como en columnas. Este tipo de biclusters no se puede encontrar simplemente considerando que los valores en el bicluster son obtenidos a partir de un modelo aditivo o multiplicativo. Se requieren métodos más sofisticados en los que se utiliza el análisis de la varianza entre grupos y una forma de co-varianza entre filas y columnas del bicluster para evaluar la calidad de los resultados.

Podemos definir un bicluster perfecto (I, J) con valores coherentes en filas y columnas utilizando un modelo aditivo:

$$a_{ij} = \mu + \alpha_i + \beta_j \tag{4.5}$$

donde μ es el valor típico del bicluster, α_i es el ajuste para la fila $i \in I$ y β_j es el ajuste para la columna $j \in J$. También podemos utilizar un modelo multiplicativo para representar este tipo de biclusters:

$$a_{ij} = \mu' * \alpha'_i * \beta'_j \tag{4.6}$$

Esta aproximación es equivalente al modelo aditivo: $\mu = log\mu'$, $\alpha_i = \alpha'_i$ y $\beta_j = \beta'_j$.

Por último, están los biclusters basados en una evolución coherente de sus valores. Algunos algoritmos de biclustering intentan solucionar el problema de encontrar evoluciones coherentes a lo largo de las filas o columnas de una matriz sin tener en cuenta sus valores exactos. Para el caso concreto de matrices de expresión genética, se puede estar interesado en buscar evidencias de que un subconjunto de genes es regulado positiva o negativamente a lo largo de un subconjunto de condiciones experimentales, sin tener en cuenta sus valores de expresión en la matriz.

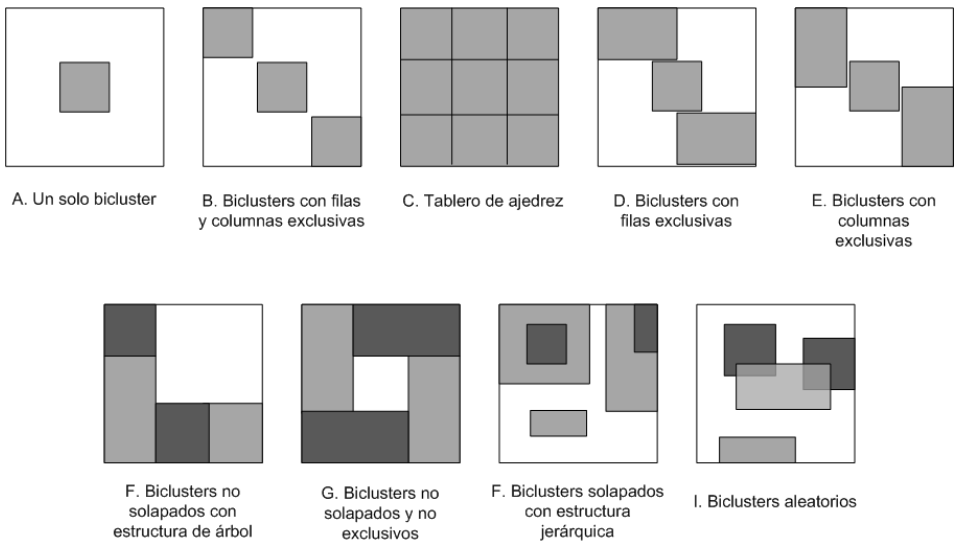

Figura 4.3: Posibles estructuras de biclusters.

4.3.3. Estructuras de biclusters

En los algoritmos de biclustering pueden plantearse dos tipos distintos de objetivos: encontrar un solo bicluster, tal y como aparece en la figura 4.3(A), o encontrar K biclusters, siendo K un número conocido o no de antemano.

Cuando los algoritmos asumen que existe más de un bicluster en la matriz de datos, como resultado se pueden obtener varios tipos de estructuras (ver figura 4.3):

- Biclusters con filas y columnas exclusivas (bloques diagonales de forma rectangular obtenidos tras una reordenación de filas y columnas).
- Biclusters sin solape con estructura de tablero de ajedrez.
- Biclusters con filas exclusivas.
- Biclusters con columnas exclusivas.
- Biclusters no solapados con estructura de árbol.
- Biclusters no solapados y no exclusivos.
- Biclusters solapados con estructura jerárquica.
- Biclusters arbitrarios.

Dependiendo de la técnica a utilizar, se podrá obtener un tipo u otro de estructura. Hay métodos que forman una imagen coloreada de la matriz A, en la que cada elemento a_{ij} tiene un tono de color asociado a su nivel de expresión. Posteriormente, se reordenan filas y columnas de tal forma que se va formando una imagen con bloques de datos del mismo color. Estos bloques son subconjuntos de filas y columnas con valores similares de expresión, es decir, biclusters. Una reordenación ideal de la matriz A produciría una imagen con cierto número K de bloques diagonales , tal y como aparece en la figura 4.3(B). Esta situación ideal corresponde a la existencia de K bloques exclusivos y exhaustivos, es decir, filas y columnas de la matriz pertenecen de forma exclusiva a uno de estos biclusters. Aunque ha quedado demostrado que este reordenamiento ideal rara vez se da en la naturaleza (Lazzeroni y Owen, 2000).

Así pues, la siguiente consideración a la hora de detectar biclusters sería partir del hecho de que filas y columnas pueden pertenecer a más de un bicluster y asumir una estructura de tablero de ajedrez en los datos (figura 4.3(C)). De esta forma, aceptamos la existencia de K biclusters no exclusivos y no solapados. Algunas técnicas asumen esta estructura (Klugar *et al.*, 2003).

Otros métodos asumen que las filas solo pueden pertenecer a un bicluster, mientras que las columnas pueden pertenecer a varios biclusters (figura 4.3(D)). Sin embargo, estos algoritmos también pueden obtener biclusters con columnas exclusivas, simplemente cambiando la orientación de la matriz de datos (figura 4.3(E)).

Las estructuras comentadas hasta el momento asumen que los biclusters son exhaustivos, es decir, cada fila y cada columna pertenecen al menos a un bicluster. Sin embargo, podemos considerar variaciones no exhaustivas de estas estructuras para hacer posible que alguna fila o columna se quede sin pertenecer a ninguna submatriz resultante (Hartigan, 1972) (Wang *et al.*, 2002). Un inconveniente de este tipo de aproximaciones es que ninguna de ellas acepta el solape, fenómeno que en el caso de los datos de expresión genética es muy usual, es decir, un gen participando en dos procesos celulares totalmente distintos.

Las estructuras anteriores son restrictivas en muchos aspectos. Por un lado, algunas de ellas asumen que, por motivos de visualización, todos los biclusters identificados deberían poder observarse directamente en la matriz de datos después de realizar un reordenamiento de filas y columnas. Por otro lado, está también la exhaustividad. Sin embargo, en los datos reales es más común que no existan restricciones, es decir, que alguna fila o columna no participe en el resultado final y que los biclusters se solapen. Aún así,

es posible habilitar estas dos propiedades, sin eliminar las propiedades de visualización, si se adopta la estructura jerárquica propuesta por Hartigan (Hartigan, 1972) (figura 4.3(H)). Esta estructura tiene el inconveniente de que o los biclusters son disjuntos o tienen que incluirse los unos a los otros. Podemos encontrar dos especializaciones de esta estructura en las figuras 4.3(F) y 4.3(G).

Finalmente, la última de las estructuras (figura 4.3(I)), representa el caso más general y con menos restricciones. Esta es la estructura más buscada por la mayoría de los algoritmos de biclustering. Se podría decir que es algo caótica, al permitir todo tipo de fenómenos: exhaustividad, no exhaustividad, solape, no solape, etc. Pero en principio es la más próxima a la realidad en cuanto a datos de expresión genética se refiere.

4.3.4. Técnicas de biclustering

La búsqueda de biclusters en matrices de expresión genética es un reto muy atractivo que en pocos años ha tenido muy buena respuesta por parte de la comunidad científica. Hay técnicas muy variadas que han resultado ser de gran efectividad, y todas ellas tienen características comunes y diferencias que las hacen únicas.

Debido a la gran variedad de técnicas propuestas, es necesario hacer una clasificación. Esta clasificación está basada en el tipo de método utilizado para encontrar submatrices en una matriz de datos:

- Divide y vencerás.
- Combinación de clustering sobre filas y columnas.
- Búsqueda voraz iterativa.
- Búsqueda exhaustiva.
- Identificación de parámetros de distribución.
- Búsqueda estocástica

Otro aspecto a tener muy en cuenta a la hora de valorar un algoritmo de biclustering es el método utilizado para comprobar la calidad de los resultados obtenidos. Existen alternativas muy diversas, desde valores estadísticos hasta visualizaciones gráficas, pasando por comprobación biológica de los datos. Se analizará la efectividad de estas funciones de calidad en datos obtenidos a partir de experimentos con microarrays.

4.4. Algoritmos de biclustering

El objetivo de este apartado es comentar, de cada tipo de técnica de biclustering expuesta en la sección 4.3.4, uno de los ejemplos más representativos que se encuentran en la literatura.

Previamente, es necesario presentar una serie de notaciones que pueden ser de utilidad a lo largo del estudio de los algoritmos de biclustering. Dada la matriz de datos $A = (X, Y)$, donde X es un conjunto de filas e Y un conjunto de columnas y un bicluster es una submatriz (I, J) donde $I \subset X$ e $J \subset Y$, nos referiremos con el símbolo a_{iJ} a la media de la fila i del bicluster. La media de la fila j del bicluster será representada por el símbolo a_{Ij}, y a_{IJ} será la media de todos los elementos del bicluster. Estos valores son definidos por las siguientes ecuaciones:

$$a_{iJ} = \frac{1}{\mid J \mid} \sum_{j \in J} a_{ij} \tag{4.7}$$

$$a_{Ij} = \frac{1}{\mid I \mid} \sum_{i \in I} a_{ij} \tag{4.8}$$

$$a_{IJ} = \frac{1}{\mid J \mid\mid I \mid} \sum_{i \in I, j \in J} a_{ij} = \frac{1}{\mid I \mid} \sum_{i \in I} a_{iJ} = \frac{1}{\mid J \mid} \sum_{j \in J} a_{Ij} \tag{4.9}$$

4.4.1. Divide y vencerás

A la hora de resolver un problema de gran dificultad, una de las iniciativas más utilizadas, no solo en programación si no en la mayoría de ámbitos, consiste en dividir ese problema complejo en una serie de sub-problemas más sencillos. El proceso de división finaliza cuando ya no es posible aplicar ninguna división más. Una vez solucionados estos sub-problemas, las distintas soluciones son combinadas para obtener la solución al problema original (Cormen *et al.*, 2001).

Esta sencilla y lógica forma de actuar también se ha aplicado con el objetivo de descubrir subgrupos de genes que evolucionan de forma similar bajo un conjunto de condiciones experimentales.

Direct Clustering

El trabajo de Hartigan (Hartigan, 1972) fue el comienzo de las tareas de búsquedas de biclusters. En este trabajo, se propone un algoritmo de clustering llamado *Direct Clustering*. Este algoritmo basado en divisiones tiene como objetivo dividir la matriz de datos original en submatrices.

El algoritmo Direct Clustering utiliza una estrategia *top-down*, es decir, se parte de un solo bicluster en el que se encuentran todos los datos, es decir, la matriz original A. Utilizando un proceso iterativo, se selecciona la fila o columna que produzca la mayor reducción de la varianza en el bloque (bicluster) de partida mediante la división de dicho bloque en dos trozos. Para encontrar la mejor división posible, las filas y columnas de la matriz son ordenadas en función de la media de las filas y la media de las columnas, respectivamente. Una vez seleccionado el punto de división, se generan dos bloques a partir del original y comienza una nueva iteración, partiendo de cada uno de los bloques generados anteriormente. El proceso de división continua hasta que un número K dado de bloques ha sido obtenido o la varianza en conjunto de todos los bloques llegue a un cierto umbral.

Hemos de destacar que la función de calidad utilizada por Hartigan para obtener los distintos bloques (I, J) es la varianza de sus datos:

$$VAR(I, J) = \sum_{i \in I, j \in J} (a_{ij} - a_{IJ})^2 \tag{4.10}$$

Al exigir la progresiva disminución de la varianza de las submatrices, se puede deducir que el objetivo de este algoritmo es el de descubrir biclusters formados por valores constantes. A pesar de esta limitación, el autor menciona la posibilidad de cambiar la función de calidad para hacer posible la búsqueda de biclusters con filas constantes, columnas constantes o ambas. De todas formas, es una de las primeras formas de validación conocida y se basa en una medida estadística de los datos.

Otro aspecto a tener en cuenta es el número K, que determina las divisiones máximas a aplicar a la matriz original, es decir, el número de biclusters resultantes. Este parámetro de usuario fue concebido como un valor necesario para evitar la degeneración de los resultados finales. Para Hartigan, el bicluster perfecto sería aquel cuya varianza fuese cero. El número K impide que el resultado final esté formado por tantos bloques como datos tenga la matriz de entrada. Además del número K, se utiliza un umbral prefijado de varianza que fija un límite en el análisis y la división de los datos. No obstante, el hecho de fijar un número máximo de resultados es una limitación, ya que puede ocultar la estructura natural de los datos de entrada.

El hecho de tener que establecer una estimación del número óptimo de divisiones supone una dificultad que algunos autores han intentado salvar. Duffy y Quiroz (Duffy y Quiroz, 1991) sugirieron el uso de tests basados en permutaciones para determinar cuándo una división no es significativa. Siguiendo esta misma línea, Tibshirani et al. (Tibshirani *et al.*, 1999) proponen un método de poda hacia atrás del proceso de división y una deducción

del número óptimo de biclusters, utilizando para ello permutaciones. En su algoritmo, las divisiones continúan hasta obtener un gran número de bloques. Algunos bloques son entonces recombinados hasta que se alcanza el número óptimo de submatrices. Tal y como se puede observar, esta forma de actuar es similar a la que se lleva a cabo en los árboles de decisión, en los que llegado a cierto nivel del árbol se realizan acciones de poda para dar el resultado final.

4.4.2. Combinación de clustering sobre filas y columnas.

Tal y como ha sido comentado en secciones anteriores, el biclustering es una técnica derivada del clustering. Por ello, parece lógico afrontar una agrupación bidimensional de los datos de una matriz aplicando las originales técnicas de clustering a cada una de las dimensiones, filas y columnas, para después combinar los resultados. Esta es una de las primeras formas utilizadas para obtener biclusters y, conceptualmente, la más obvia.

Al ser una adaptación de las técnicas ya existentes de clustering, estos métodos heredan tanto sus ventajas como sus limitaciones, ambos aspectos estudiados en el capítulo 3.

Coupled Two-Way Clustering

Getz et al. (Getz *et al.*, 2000) fueron de los primeros autores en aplicar este tipo de técnicas. El objetivo principal que persigue su algoritmo CTWC (*Coupled Two-Way Clustering*) es el de identificar parejas de pequeños subconjuntos de atributos (F_i) y objetos (O_j), pudiendo representar (F_i) y (O_j) tanto a filas como a columnas. Se parte de la suposición de que cuando solo los atributos en F_i son utilizados para agruparse con los objetos de O_j, se crean particiones de datos estables y significativas. Sin embargo, la existencia de múltiples combinaciones de posibles agrupaciones puede resultar un inconveniente. Para solventar esta dificultad, los autores utilizan un método heurístico para disminuir el espacio de búsqueda, considerando que solo los subgrupos de filas o columnas identificados como clusters estables en iteraciones anteriores son candidatos para un cluster combinado de filas y columnas. Se consideran estables a aquellos clusters que son significativos, estadísticamente hablando, según un determinado criterio.

El algoritmo CTWC establece un marco genérico en el que poder utilizar cualquier tipo de técnica de clustering para obtener submatrices significativas de datos. No obstante, los autores establecen una serie de criterios que hacen de filtro a la hora de seleccionar el algoritmo de clustering: el número de clusters ha de ser determinado por el propio algoritmo y no por

un parámetro externo (por lo que quedan fuera algoritmos tales como el K-means o el SOM, ver secciones 3.3.2 y 3.3.3), estables frente al ruido, que generen una jerarquía de datos (dendograma, ver Sec. 3.3.4) y que proporcionen un mecanismo para identificar clusters estables y robustos. En su artículo, Getz et al. utilizan una técnica jerárquica de clustering denominada SPC (*Super-Paramagnetic Clustering of data*) (Blatt *et al.*, 1996), que consta de un parámetro T que controla la progresiva división de los clusters y la estabilidad de los mismos.

El algoritmo tiene las siguientes entradas: una matriz de similitud entre las filas obtenidas a partir del conjunto de columnas, una matriz de similitud de las columnas obtenidas a partir del conjunto de filas y el parámetro T. La medida de similitud está basada en la distancia euclídea. El algoritmo comienza con $T = 0$ y un solo cluster que contiene a todas las filas y columnas (estrategia *top-down*), es decir, con el conjunto de todos los genes, g^0, y el conjunto de todas las condiciones, s^0. A partir de estos conjuntos, se obtienen una serie de clusters estables de genes g_n^1 y de condiciones s_m^1. Cada pareja de estos clusters forma una submatriz, o bicluster, que es utilizado como entrada de la siguiente iteración. A medida que T aumenta tienen lugar las fases de clustering, en las que los clusters se dividen en varios sub-clusters. El parámetro de control T se utiliza para proporcionar una medida de la estabilidad de un cluster particular a partir del rango de valores ΔT durante el cual dicho cluster permanece invariable. Un cluster estable es aquel que sobrevive sin cambios a un alto rango de valores de T. Durante la ejecución, el algoritmo mantiene de manera dinámica dos listas de clusters estables, una para clusters de filas y otra para clusters de columnas, además de una lista de clusters de filas y columnas. Los nuevos clusters estables de filas y columnas son añadidos a la lista, señalando qué submatriz ha sido su padre. El proceso iterativo continúa hasta que no se encuentren más clusters que satisfagan la propiedad de estabilidad o un tamaño crítico.

En conclusión, este algoritmo presenta un proceso iterativo en el que los resultados de aplicar técnicas de clustering sobre una matriz de entrada se combinan para crear submatrices (biclusters) que vuelven a ser entradas en la siguiente iteración. Como detalles importantes de este trabajo se puede destacar el uso de la distancia euclídea como medida de similitud entre los objetos de las distintas matrices usadas como entrada de las fases de clustering. La posibilidad del uso de distintos algoritmos de clustering proporciona mucha libertad a la hora de aplicar el método, no obstante los resultados pueden variar mucho dependiendo de la técnica utilizada, ya que en función de ella se define el concepto de estabilidad de los clusters. Otro aspecto discutible del método es la condición de parada del algoritmo, hecho que no

está claramente explicado por los autores (Getz *et al.*, 2000). Finalmente, indicar que los datos de entrada sufren una fase previa de pre-procesamiento en la que son normalizados.

Con respecto al método de evaluación de los resultados, los autores se centran en comparar los clusters obtenidos a partir de clasificación previamente conocida de los datos de entrada. Una de las técnicas que utilizan es la de identificar aquellos genes que dividen a las condiciones experimentales según una serie de clases conocidas a priori. Esta forma de comparar los resultados experimentales con conocimiento biológico previo será utilizada más adelante por muchos autores, siendo una de las mejores maneras de comprobar la calidad, desde el punto de vista biológico, de los resultados finales.

4.4.3. Búsqueda voraz iterativa

A estas clases de algoritmos (*Greddy Algorithms*) corresponden aquellos métodos que intentan obtener biclusters de la manera más rápida y simple posible. Para ello, aplican una meta-heurística basada en la maximización de criterios locales. Es decir, intentan simplificar la búsqueda tomando decisiones locales con la esperanza de encontrar la solución óptima global. Se pierde calidad de resultados pero se gana sencillez y velocidad a la hora de resolver los problemas.

Dentro de los algoritmos de biclustering que utilizan búsqueda voraz se encuentra uno de los más conocidos por haber sido el primer trabajo que utilizó el término de bicluster. Así pues, el trabajo de Cheng y Church (Cheng y Church, 2000) fue pionero en este área de la minería de datos y de gran trascendencia.

Cheng y Church

Cheng y Church (Cheng y Church, 2000) establecieron un método que basaba la búsqueda del resultado final en un problema de optimización. Parten de la siguiente suposición: para que un subgrupo de genes y condiciones sea un bicluster, sus valores han de evolucionar al unísono. Esta característica está representada por un valor estadístico: *Mean Squared Residue (MSR)*. Para llegar a entender el trabajo de Cheng y Church es necesario conocer a fondo esta medida.

Sea (I, J) un bicluster. El residuo R de un elemento e_{ij} del bicluster (I, J) es:

$$R(e_{ij}) = e_{ij} - e_{iJ} - e_{Ij} + e_{IJ} \tag{4.11}$$

donde e_{iJ} es la media de la fila i-ésima del bicluster, e_{Ij} la media de la columna j-ésima, y e_{IJ} es la media de todos los elementos del bicluster.

El MSR de un bicluster (I, J) se define como:

$$H(I,J) = \frac{1}{\mid I \mid\mid J \mid} \sum_{i \in I, j \in J} R^2(e_{ij}) \tag{4.12}$$

Este valor es considerado como un índice de calidad. El MSR es la varianza del conjunto de todos los valores del bicluster, más la varianza de fila y la varianza de columna. Es por ello que dicho valor mide el nivel de coherencia de los valores a lo largo de las filas y las columnas. A mayor valor del MSR, con menos coherencia evolucionan los valores de los genes a lo largo de las condiciones y, por consiguiente, menos calidad tendrá el bicluster.

Partiendo de esta definición, los autores establecieron el $\delta-$bicluster como aquel subconjunto de filas y columnas (I, J) cuyo valor de residuo no supera un determinado valor umbral: $H(I, J) \leq \delta$. Por consiguiente, el objetivo principal del primer algoritmo de biclustering, propiamente dicho, es el de obtener una submatriz del mayor tamaño posible y con un residuo no superior a un determinado umbral. Cheng y Church explican en su trabajo, haciendo uso de la teoría de grafos, que la búsqueda de un bicluster máximo se puede asemejar a la búsqueda de un subgrafo máximo en un grafo bipartito. Y dicho problema tiene una complejidad NP-completa (Peeters, 2003). Por ello, plantean un algoritmo voraz para converger de forma rápida a una submatriz máxima local que sea menor que un residuo dado, no teniendo que ser ésta la mejor solución posible.

El algoritmo de Cheng y Church puede ser entendido como un algoritmo de búsqueda local. Toma como entrada la matriz original de datos y un valor umbral de residuo. Su objetivo es encontrar un solo bicluster en cada ejecución y para ello lleva a cabo dos fases principales. En la primera fase, el algoritmo elimina filas y columnas de la matriz original. En cada paso, en el que la submatriz en construcción consta de I filas y J columnas, el algoritmo estudia el conjunto de posibles movimientos. Para cada fila, se calcula el valor de su residuo, $d(i) = \frac{1}{|J|} \sum_{j \in J} R(e_{ij})$, y lo mismo para cada columna, $d(j) = \frac{1}{|I|} \sum_{i \in I} R(e_{ij})$. El siguiente paso es seleccionar la fila o columna con mayor valor de residuo y eliminarla de la submatriz. El proceso finaliza cuando el residuo de la submatriz sea H(I,J)$< \delta$. Para implementar este borrado masivo, los autores se basan en la teoría de que aquellas filas o columnas con un alto valor de residuo pueden ser eliminadas con la garantía de una mejora en el residuo total de la submatriz resultante.

En la segunda fase del algoritmo, filas y columnas son añadidas a la submatriz resultante de la anterior fase. Se sigue un esquema similar al anterior,

pero esta vez buscando las filas y columnas de la matriz original con menores valores de residuo. El proceso finaliza cuando un posible aumento del tamaño de la submatriz hace que el valor de su residuo MSR cruce el umbral fijado en los parámetros de entrada. Finalmente, el resultado obtenido es la submatriz máxima que cumple la condición de no sobrepasar el valor umbral del MSR.

Con respecto al pre-procesamiento de los datos, los autores sustituyen los huecos en la matriz de entrada a partir de valores aleatorios.

La propuesta de Cheng y Church ha sido de gran relevancia al ser la primera que introducía el concepto de bicluster y que usaba un algoritmo original para su obtención. Sin embargo, el algoritmo propuesto tiene algunas lagunas. De hecho, ha sido fuente de múltiples estudios y muchos trabajos han sido generados con objetivo de mejorar sus debilidades (por ejemplo, el algoritmo FLOC (Yang y Wang, 2002) (Yang y Wang, 2003)). En primer lugar, el algoritmo tiene como objetivo en cada ejecución encontrar un solo bicluster. Sin embargo, los autores ofrecen la posibilidad de encontrar un número determinado K de biclusters siguiendo un método más que discutible: cada vez que una submatriz es generada, se sustituye en la matriz original por una serie de números aleatorios, generados a partir de un rango de valores que han de ser introducidos por el usuario, y vuelta a empezar. Esta modificación de la matriz original desvirtúa totalmente la naturaleza de los datos, confiriendo a los resultados finales dudosa credibilidad. En segundo lugar, en los experimentos llevados a cabo por los autores, se fija un valor de umbral igual a 300. La justificación de este valor nada tiene que ver con fijar una calidad determinada, sino con hacer interesante el experimento. El valor máximo del MSR supone una limitación desde el punto de vista de que cada base de datos debería contar con un valor aproximado que determinara la calidad de los biclusters deseados, algo que es verdaderamente complicado ya que no existe relación directa entre este valor y la calidad biológica de la agrupación de genes. A pesar de ello, este valor ha sido utilizado por multitud de investigadores en sus técnicas para establecer comparaciones de sus resultados. El trabajo de Aguilar (Aguilar, 2005) ha demostrado que el MSR no es una medida de calidad adecuada. En esta investigación, se presentan una serie de fenómenos que son muy frecuentes entre los biclusters y que pueden influir directamente en la evaluación final de los mismos.

El objetivo final de toda técnica de biclustering es encontrar un subgrupo de genes que se expresen de manera similar bajo un conjunto de condiciones experimentales. Se busca una similitud en el comportamiento, y en esa búsqueda hay que tener en cuenta dos aspectos importantes: los fenómenos

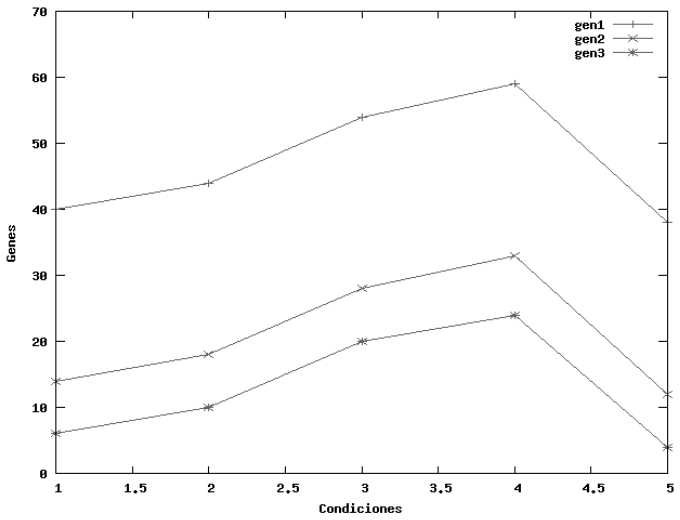

Figura 4.4: Ejemplo de desplazamiento en un bicluster de 3 genes

de escalado y desplazamiento. Es posible que un par de genes se comporten de la misma manera bajo un conjunto de condiciones, aunque sus valores de expresión se muevan en rangos muy distintos (desplazamiento, ver figura 4.4). En otras ocasiones, el aumento y disminución de sus valores de expresión se pueden llevar a cabo al unísono, sin embargo el porcentaje de subida o bajada puede no tener que ser el mismo (escalado, ver figura 4.5). En estas dos situaciones anteriores, los genes se comportan de la misma manera desde el punto de vista de la expresión genética. Así pues, existen técnicas de clustering y biclustering, basadas en utilizar la distancia como medidas de similitud, que no son capaces de detectar estos fenómenos, obviando así posible buenos resultados.

Aguilar (Aguilar, 2005), analizando estos fenómenos, demuestra que el MSR es muy útil para detectar patrones de desplazamiento, pero no es apropiado para detectar patrones de escalado. En este trabajo se expone que el residuo de Cheng y Church es altamente dependiente de la varianza del factor de escalado, lo que hace posible que un algoritmo basado en el MSR no tenga en cuenta este fenómeno cuando la varianza de los valores de expresión de los genes es demasiado alta. Mediante demostraciones matemáticas consigue descubrir que un desplazamiento (suma de una constante a los valores de la matriz) no afecta en lo más mínimo al valor del residuo, todo lo contrario que un escalado (multiplicación de los valores por una constante).

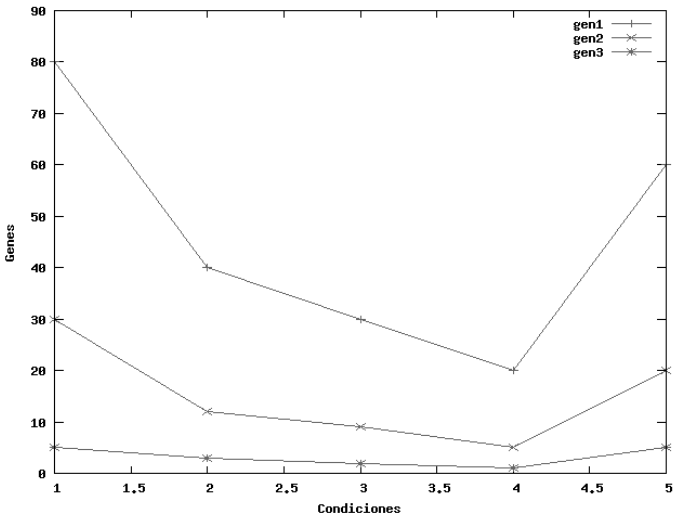

Figura 4.5: Ejemplo de escalado en un bicluster de 3 genes

4.4.4. Búsqueda exhaustiva

Algunos autores prefieren basar su búsqueda de biclusters en una enumeración exhaustiva de todas las posibles submatrices existentes en la matriz de datos. Por supuesto, se encuentran mejores resultados que en los dos tipos anteriores ya que se revisan todos los posibles biclusters. A pesar de todo, existe una gran desventaja: el gran coste computacional que supone este tipo de métodos. Para solucionar este problema, son numerosas las técnicas que añaden restricciones para reducir el espacio de búsqueda.

Por este motivo, lo que va a distinguir claramente a estas técnicas son la búsqueda de un tipo de bicluster determinado. Y esa caracterización del resultado vendrá dada por diversas variables: tamaño, número de condiciones, número de genes, MSR, etc.

SAMBA

Tanay et al. (Tanay *et al.*, 2002) utilizan una combinación muy interesante de teoría de grafos y de análisis estadístico en su trabajo. SAMBA (*Statistical-Algorithmic Method for Bicluster Analysis*) es un algoritmo cuyo objetivo es la identificación de un determinado número de biclusters significativos en una base de datos de expresión genética.

Los autores definen su objetivo particular como un subconjunto de genes que responden al unísono a una serie de condiciones experimentales. Dicha respuesta está a su vez basada en un aumento significativo de los valores de expresión de los genes con respecto a su nivel normal. Es decir, Tanay

et al. buscan un tipo especial de coherencia entre los datos de un bicluster basada en la activación (aumento de la cantidad relativa de ARNm, es decir, aumento del valor de expresión) de los genes a lo largo de una serie de condiciones. Además de esta caracterización del tipo de submatriz a buscar, se hace uso de la representación en forma de grafo bipartito de los datos de entrada (ver sección 4.2.2). Los dos conjuntos de vértices estarán formados por los genes y las condiciones. Si una arista une un gen con una condición, significará que ese gen ha respondido de forma positiva a esa condición, es decir, su nivel de expresión ha aumentado por encima de su nivel normal. Por lo cual, también habrá parejas de genes y condiciones para las que no exista arista. No obstante, a cada pareja de genes y condiciones se le asignará un peso.

Partiendo de este modelado de los datos de entrada, el problema aquí planteado será equivalente a encontrar el subgrafo más pesado dentro de un grafo bipartito, tarea que es NP-completa. Para reducir la complejidad computacional, se introduce una restricción que afecta al tamaño de los biclusters. Los autores solo se centrarán en aquellos subgrafos cuyos vértices genes tengan un grado (número de aristas incidentes en ellos) menor que un determinado valor d. La justificación es muy sencilla: los vértices genes con alto grado representan a genes que participan en la gran mayoría de procesos, por lo que no son significativos a la hora de detectar un efecto concreto.

SAMBA comienza con una discretización de los datos. En principio lo único que importa es que los valores de expresión estén activos bajo una determinada condición, no su valor concreto. Para ello, primero se estandariza la matriz de entrada (usando para ello un valor de la media igual 0 y un valor 1 para la varianza) y posteriormente se considerará que un gen está activo si su valor es superior a 1 y no activo si este valor es inferior a -1. El hecho de conocer el nivel de actividad de un gen no implica conocer si esa activación ha supuesto un aumento o disminución del valor de expresión con respecto a su estado anterior. Éste es un tema discutido por los autores en su artículo, en el que proponen aumentar la información del grafo para tener en cuenta las subidas o bajadas de los niveles de expresión. A cada arista, que representa una activación, se le añadiría un signo que indicaría un aumento o disminución de la expresión genética. Por lo tanto, el grafo tendría tres tipos de relaciones binarias: actividad positiva, actividad negativa y no actividad. De esta forma se podrá buscar un subgrafo en el que los genes tengan la misma tendencia o incluso la tendencia opuesta.

Partiendo de la información discretizada ya se puede formar el grafo bipartito, en el que un vértice gen y un vértice condición quedarían unidos por una arista si en la matriz estandarizada el valor correspondiente a ese

elemento es superior a 1. El siguiente paso es dotar a cada pareja (vértice gen- vértice condición) de un peso. Este peso será dependiente del modelo estadístico utilizado para representar al bicluster final. Los autores presentan dos de estos modelos y muestran cómo asignar los pesos a las parejas de vértices de tal forma que un subgrafo pesado se corresponda con el tipo de bicluster que estamos buscando.

La segunda fase tiene como objetivo encontrar los k subgrafos más pesados en el grafo. Para llevar a cabo esta búsqueda, Tanay et al. hacen uso de una tabla *hash* en la que se almacena el peso de los posibles sub-grafos existentes. Por supuesto, existen restricciones a la hora de calcular dichos subgrafos: solo se tienen en cuenta aquellos genes con un grado inferior a un parámetro dado, d.

Para finalizar, SAMBA lleva a cabo una heurística voraz en la que esos k subgrafos son depurados mediante la eliminación o adición de vértices. El algoritmo aplicará iterativamente la mejor modificación para cada bicluster hasta que no sea posible mejorar su peso. También se filtran aquellos subgrafos que son similares, es decir, que tienen conjunto de vértices (genes y condiciones) que difieren ligeramente.

Para la evaluación de los resultados obtenidos se utiliza de nuevo conocimiento previo sobre los datos analizados. Aún así, este trabajo es uno de los primeros relacionados con biclustering en usar una medida de calidad que cobrará gran importancia en posteriores investigaciones relacionadas: el *p-value*. El objetivo es representar la distribución del p-value de todos los biclusters obtenidos haciendo uso de una clasificación de las condiciones o los genes ya conocida. El p-value se calcula de la siguiente manera: suponemos la existencia de una clasificación previa formada por k clases,$(C_1, C_2, \ldots, C_k)$. Sea B un bicluster con b condiciones, de las que b_j pertenecen a la clase C_j. El p-value de B, asumiendo que la clase más abundante es C_i, es calculado como:

$$p(B) = \sum_{k=b_i}^{b} \frac{\binom{|C_i|}{k}\binom{m-|C_i|}{b-k}}{\binom{m}{b}} \tag{4.13}$$

Por lo tanto, el p-value mide la probabilidad de obtener b_i elementos de una clase dada en un conjunto de tamaño b. Esta medida se utiliza para comprobar el grado de azar que existe cuando se consigue un bicluster que contiene genes o condiciones de una clase dada.

Este tipo de evaluación basada en conocimiento previo tiene un problema: se podrían obviar aquellos biclusters que establezcan una clase desconocida que pudiera ser de importancia.

4.4.5. Identificación de parámetros de distribución

Existen técnicas que se ayudan de la estadística en determinadas fases de sus procesos de búsqueda. Sin embargo, hay algoritmos de biclustering en los que la estadística es el arma principal. En este tipo de investigaciones se intenta aproximar el concepto de bicluster a partir de un modelo estadístico. Por consiguiente, el objetivo más importante será encontrar los parámetros de distribución necesarios para generar las submatrices que se ajusten a dicho modelo estadístico. En la mayoría de los casos, será un proceso iterativo, en el que se intenta minimizar un determinado criterio, el encargado de encontrar el valor de esos parámetros.

Son técnicas con un gran contenido matemático y se distinguen por las distintas caracterizaciones que llevan a cabo de los biclusters. El álgebra lineal y la teoría matricial son de las herramientas matemáticas más utilizadas a este respecto.

Plaid Model

En el trabajo de Lazzeroni y Owen (Lazzeroni y Owen, 2000) la idea principal se basa en considerar a la matriz de genes y condiciones como una superposición de capas, siendo cada una de ellas un subconjunto de filas y columnas con unos valores de expresión concretos. Estos valores son representados en la matriz de entrada a partir de una determinada coloración en función del nivel de expresión, formándose así una matriz coloreada.

El orden de las filas y columnas de una matriz como la descrita anteriormente puede ser arbitrario. Aún así, es normal el considerar una reordenación para agrupar filas y columnas similares y así conseguir una matriz formada por bloques, cada uno de ellos constituidos por valores de un color similar. Un orden ideal de la matriz produciría una imagen formada por un número K de bloques rectangulares en la diagonal, estando el resto de la matriz formada por una serie de bloques con un color de fondo. Cada uno de estos bloques de la diagonal tendría un color prácticamente uniforme y corresponderían a biclusters exhaustivos y mutuamente excluyentes, es decir, cada gen en un bloque k sólo se expresa en las condiciones de dicho bloque (ver figura 4.3 (B)). De forma algebraica, este ideal estaría representado por la siguiente ecuación:

$$A_{ij} = \mu_0 \sum_{k=1}^{K} \theta_{ijk} \rho_{ik} \kappa_{jk} \tag{4.14}$$

donde μ_0 es un color de fondo general de la matriz, y $\theta_{ijk} = \mu_k + \alpha_{ik} + \beta_{jk}$

representa al color de un determinado bloque k, con μ_k siendo el color de fondo del bloque k y α y β colores de cada fila y columna que se suman al color inicial. Tal y como se puede apreciar, esta definición de bicluster formado por un valor típico y un ajuste específico para cada fila y columna se vio en el apartado 4.3.2 (ecuación 4.5). El color de fondo μ_k describirá una respuesta compartida por todos los genes de la capa, es decir, un comportamiento coherente. La estructura final que presentarán los biclusters está representada por los indicadores de pertenencia. Es decir, $\rho_{ik} \in \{0, 1\}$ será igual a 1 si el gen i pertenece al bloque k. Lo mismo para el indicador κ_{jk} con respecto a la columna k. Por ello, si se fija la restricción $\sum_k \rho_{ik} = 1$ para todas las filas i y $\sum_k \kappa_{jk} = 1$ para todas las columnas j, los biclusters resultantes sean exhaustivos y excluyentes, aunque en la naturaleza lo más normal es que estas agrupaciones de datos se solapen. Para permitir este fenómeno, dichas restricciones han de relajarse, es decir, $\sum_k \rho_{ik} \geq 2$ para alguna fila i y $\sum_k \kappa_{jk} \geq 2$ para alguna columna j. Incluso pueden existir genes o condiciones que no pertenezcan a ningún bicluster, por lo cual $\sum_k \rho_{ik} = 0$ y $\sum_k \kappa_{jk} = 0$ para alguna fila o columna, respectivamente.

Así, la matriz de entrada se considera como una suma de capas, siendo cada capa considerada como un bicluster. Los valores α_{ik} y β_{jk} determinarán el comportamiento de los genes y condiciones en dichas submatrices. Aquellos genes de la capa k en los que $\mid \mu_k + \alpha_{ik} \mid$ tenga un valor alto se verán más afectados por las condiciones experimentales de dicha capa que otros genes. Los niveles de actividad también tendrán mayor efecto en las condiciones con un alto valor de $\mid \mu_k + \beta_{jk} \mid$. Si $\mu_k + \alpha_{ik}$ tiene valor positivo para un gen i querrá decir que el gen está activo en la capa k, mientras que un valor negativo indicaría todo lo contrario.

Así, se puede concluir que el objetivo es encontrar los parámetros adecuados para que la matriz de entrada se ajuste lo mejor posible al modelo planteado. Es decir, en minimizar la siguiente ecuación:

$$\sum_{ij} [A_{ij} - \sum_{k=0}^{K} \theta_{ijk} \rho_{ik} \kappa_{jk}]^2 \tag{4.15}$$

Hemos de tener en cuenta que, para cada capa k, existen $(2^n - 1) * (2^m - 1)$ formas de determinar qué genes y condiciones participan en cada capa, siendo n el número de filas y m el número de columnas de la matriz original. Ello representa una gran complejidad computacional. Así pues, los autores proponen resolver este problema mediante una heurística iterativa en la que, en cada iteración , se añade una capa al modelo. Supongamos que hemos fijado las primeras $K - 1$ capas y estamos buscando la capa K para

minimizar la suma de errores cuadráticos. Sea

$$Z_{ij}^{K-1} = A_{ij} - \sum_{k=0}^{K-1} \theta_{ijk}\rho_{ik}\kappa_{jk} \tag{4.16}$$

la matriz residual después de quitar el efecto de las $K-1$ primeras capas. En la iteración K se intenta minimizar la siguiente ecuación:

$$Q = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{p}(Z_{ij} - \theta_{ijK}\rho_{iK}\kappa_{jK})^2 \tag{4.17}$$

Para ello, se intenta dar valores a los parámetros de θ, ρ y κ. En cada iteración s, los valores de θ^s son obtenidos a partir de aquellos encontrados en la iteración anterior, ρ^{s-1} y κ^{s-1}. A continuación, ρ^s se obtiene a partir de θ^s y κ^{s-1}, y finalmente κ^s se obtiene utilizando θ^s y ρ^{s-1}.

Para resolver los valores de θ_{ij}, dados los valores de ρ_i y κ_j, se utiliza el teorema de Lagrange y las siguientes ecuaciones:

$$\mu = \frac{\sum_i \sum_j \mu_i\kappa_i Z_{ij}}{(\sum_i \mu_i^2)(\sum_j \kappa_j^2)} \tag{4.18}$$

$$\alpha_i = \frac{\sum_j (Z_{ij} - \mu\rho_i\kappa_j)\kappa_j}{\rho_i \sum_j \kappa_j^2} \tag{4.19}$$

$$\beta_j = \frac{\sum_i (Z_{ij} - \mu\rho_i\kappa_j)\rho_i}{\kappa_j \sum_i \rho_i^2} \tag{4.20}$$

Una vez resueltos los valores de θ_{ij}, se obtiene los valores para ρ_i:

$$\rho_i = \frac{\sum_j \theta_{ij}\kappa_j Z_{ij}}{\sum_j \theta_{ij}^2\kappa_j^2} \tag{4.21}$$

y para κ_j:

$$\kappa_j = \frac{\sum_i \theta_{ij}\rho_i Z_{ij}}{\sum_i \theta_{ij}^2\rho_i^2} \tag{4.22}$$

Para establecer una condición de parada del algoritmo, los autores definen una medida, la *importancia* de una capa k como:

$$\sigma_k^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} \rho_{ik}\kappa_{jk}\theta_{ijk}^2. \tag{4.23}$$

El algoritmo acepta una capa si su importancia es mayor que la producida por la existencia de ruido. Para calcular el valor de σ_k^2 del ruido se sigue el siguiente proceso formado por T iteraciones: de forma aleatoria permutar

cada fila en Z de forma independiente y, posteriormente, hacer lo mismo con cada columna. Se aplica el algoritmo para encontrar una capa en la matriz resultante y se calcula su importancia.

En su artículo, los autores utilizan, como en muchas otras ocasiones, el conocimiento previo sobre la clasificación de los datos de entrada para evaluar la calidad de las capas resultantes.

Este método tiene varios inconvenientes. En primer lugar hay que fijar un número determinado de capas, por lo que la agrupación final va a depender de este parámetro. Cada elemento de la matriz tiene un color, que es la suma de un color de fondo más las aportaciones de color de la fila y la columna en la que se encuentren. Esos valores hay que modelarlos en la etapa inicial y dependiendo de los valores escogidos el resultado podrá variar. El mismo problema existe con los valores iniciales de ρ y κ. Los índices de pertenencia también nos muestran una forma de condicionar la estructura de las submatrices resultantes. En conclusión, hay demasiados factores que pueden afectar a la organización natural de los datos de las matrices de entrada.

4.4.6. Búsqueda Estocástica

En los últimos años, se están aplicando técnicas de búsqueda de biclusters relacionadas más directamente con la biología. Los algoritmos genéticos se encuadran dentro de la clase de algoritmos que presentan ciertas analogías con los procesos biológicos de la naturaleza.

Dentro de este campo, *la computación evolutiva* es un enfoque alternativo para abordar problemas complejos de búsqueda y aprendizaje a través de modelos computacionales de procesos evolutivos. El propósito genérico de los algoritmos evolutivos consiste en guiar una búsqueda estocástica, haciendo evolucionar a un conjunto de estructuras y seleccionando de modo iterativo las más adecuadas. Se define proceso estocástico como aquel proceso aleatorio que evoluciona con el tiempo. De esta manera, e imitando los procesos de selección natural de los seres vivos, estas técnicas utilizan procesos sencillos, fáciles de implementar e independientes de las distintas representaciones de datos, para encontrar un objetivo concreto. La búsqueda estocástica, además, es una manera de superar los problemas de localidad de ciertos algoritmos voraces, como el utilizado por Cheng y Church.

SEBI

En el trabajo de Aguilar y Divina (Aguilar y Divina, 2006), se propone un algoritmo evolutivo de búsqueda de biclusters llamado SEBI. El objetivo

de los autores es el de encontrar biclusters de tamaño máximo, con un valor del *Mean Squared Residue* (MSR, ver 4.4.3) inferior a un umbral dado, con un valor elevado de varianza de fila y con un bajo nivel de solapamiento entre las submatrices encontradas. Es decir, encontrar subconjuntos de genes que presenten comportamientos similares bajo una serie de condiciones y que dichos comportamientos sean los suficientemente variados para ser considerados interesantes. Así, se utiliza la combinación del residuo con la varianza para determinar estos objetivos. El algoritmo SEBI es un proceso iterativo cuyo número de repeticiones viene dado por una condición de parada. En cada iteración, se genera un bicluster máximo cuyo valor de residuo es menor que un umbral introducido como parámetro de entrada. Este bicluster es almacenado en una lista de resultados y el proceso comienza de nuevo.

Todo algoritmo evolutivo necesita de una población para comenzar. Esta población estará formada por una serie de individuos que son los que evolucionarán hasta llegar a alcanzar un objetivo concreto. Es importante la caracterización de estos individuos para que se ajusten al problema concreto. Esta flexibilidad es lo que va a distinguir a los algoritmos evolutivos, que son robustos y aplicables a cualquier tipo de problema pero a la vez débiles, ya que no se especializan en ninguno. En el caso de SEBI, la población estará formada por biclusters. Inicialmente, la población consistirá en biclusters conteniendo un solo elemento de la matriz de entrada, teniendo además la propiedad de poseer un valor de residuo $MSR = 0$. Los individuos de la población evolucionarán en un proceso de refinamiento hasta llegar a convertirse en las submatrices deseadas. La codificación de cada elemento vendrá determinada por cadenas binarias de longitud $N + M$, donde N y M son el número de filas (genes) y columnas (condiciones) de la matriz de entrada. Así pues, los primeros N bits de la cadena hacen referencia a las filas y los M siguientes a las columnas. Si un bit está a 1 significa que esa fila o columna pertenece al bicluster representado por el individuo.

En la naturaleza, los individuos más capacitados se aparean entre sí para ir obteniendo una versión mejorada y refinada de la especie. Esta forma de actuar es imitada por estos algoritmos. En primer lugar, se ha de seleccionar de la población a aquellas parejas de individuos que serán padres de la siguiente generación. En este algoritmo evolutivo se ha escogido la técnica del *torneo*. En esta técnica, un determinado número de individuos de la población es seleccionado de forma aleatoria y los mejores de entre ellos serán seleccionados como padres. Para determinar la calidad de un individuo se utiliza una función de ajuste que es combinación de varios criterios: residuo, volumen (número de filas x número de columnas), varianza de fila y nivel de solapamiento con otros biclusters.

Las parejas de padres seleccionados son combinados entre sí para producir nuevos individuos. Para combinar dos cadenas binarias se utilizan indistintamente tres métodos que se distinguen por el número de bits que se modifican en las cadenas padres. Mediante el primer método, *one-point crossover*, se selecciona de forma aleatoria un bit y los padres intercambian el valor de ese bit, creándose dos nuevos individuos. En el método *double-point crossover*, el número de bits seleccionado es 2 y en *multi-point crossover*, es un número m introducido por parámetro. En el proceso de combinación, también hay que tener en cuenta una probabilidad de engendrar descendencia, que en el caso del SEBI es de 0.9 por defecto.

Una vez obtenida la nueva prole, los individuos pueden estar sujetos a diversas mutaciones. Con una probabilidad de mutación de 0.1 por defecto, cada individuo puede cambiar en función de tres operadores de mutación: un operador estándar que cambia aleatoriamente el valor de un bit del individuo, un operador que añade una nueva fila o un tercero que añade una nueva columna. Dentro del proceso de recombinación y mutación, los mejores individuos pueden ser salvados de la evolución con una probabilidad de 0.75, es lo que se conoce como *elitismo*.

Al finalizar el proceso evolutivo, si el mejor individuo de la nueva generación es considerado como un bicluster válido, es decir, con un valor de residuo menor que un umbral, unos límites mínimos de tamaño establecidos y con un nivel de solapamiento aceptable, será devuelto como resultado por el algoritmo SEBI. Este proceso se repetirá un número máximo de veces.

Para evitar un alto nivel de solapamiento, los autores utilizan unos pesos asociados a los elementos de la matriz de entrada. Este peso depende del número de biclusters en la lista de resultados que contienen a dicho elemento.

Los algoritmos evolutivos son una aplicación muy interesante de lo que los seres humanos aprenden de la naturaleza, además de poderse aplicar a cualquier tipo de problema o dato. Lo único que se ha de escoger bien es la representación de los individuos y la función de ajuste adaptada a unos objetivos concretos. La representación aquí utilizada es muy apropiada al problema que se quiere abordar. Aún así, la función de ajuste se basa principalmente en un valor de MSR introducido como parámetro, heredándose así todos los problemas que ya se han comentado sobre esta forma de evaluar las submatrices de expresión genética (ver 4.4.3). Los autores comparan sus resultados con los obtenidos por el algoritmo de Cheng y Church (Cheng y Church, 2000) en base al MSR, al volumen de las submatrices y a las varianzas de filas. Otro aspecto destacable es el afán por evitar el solapamiento entre los resultados. Una de las ventajas es la de impedir generar resultados muy parecidos entre sí. No obstante, se está impidiendo un fenómeno que

en la naturaleza de los datos a analizar es bastante frecuente. Otro aspecto negativo de este tipo de algoritmos es el gran número de decisiones y de parámetros de entrada (valor de residuo, número máximo de generaciones, tamaño de la población, probabilidad de cruce, probabilidad de mutación, elitismo, pesos para los elementos de la matriz, etc.) que hacen que el resultado final sea muy voluble y complicado el hecho de acertar con los valores apropiados de la parametrización.

4.4.7. Conclusiones

Esta sección tiene como objetivo presentar una clasificación de los algoritmos de biclustering basada en la manera que tienen de afrontar el problema, además de ofrecer un resumen de una técnica representativa de cada clase. El número de publicaciones que tratan el problema del biclustering ha crecido enormemente desde la aparición del trabajo de Cheng y Church (Cheng y Church, 2000). Los tipos de técnicas han ido evolucionando, de la lógica aplicación de las técnicas existentes de clustering a las dos dimensiones de la matriz, hasta utilizar complejos modelos matemáticos o algoritmos genéticos. Incluso se han escrito trabajos que tienen en cuenta una dimensión más: la temporal. Así, hay algoritmos de biclustering que tratan con una componente temporal en la dimensión de las condiciones, exigiendo a sus biclusters que no puedan existir huecos en las condiciones que los forman (Madeira y Oliveira, 2009) o incluso introduciendo nuevos conceptos, como el de Triclustering (Mahanta *et al.*, 2011). Hoy en día, las soluciones que aportan los estudios de biclustering están más enfocadas a un tipo de problema biológico concreto, es decir, se están especializando y están buscando ser más útiles y prácticos. Además, la evaluación de los resultados es más sofisticada, no sólo desde el punto de vista biológico sino también desde el punto de vista de comparar distintas técnicas de biclustering entre sí (Prelic *et al.*, 2006) (Burton y Krishna, 2010).

A continuación, se presenta un problema concreto: el de encontrar biclusters en bases de datos binarios. Se definirán los aspectos básicos de esta especialización del biclustering y los trabajos más relevantes de la literatura serán comentados. Así, la siguiente sección sentará las bases para presentar la metodología que aporta esta tesis doctoral.

4.5. Biclustering sobre bases de datos binarios

4.5.1. Problemática

Las bases de datos binarios constituyen una forma compacta y simple de almacenar información acerca de las relaciones establecidas entre un grupo de objetos y sus posibles propiedades. Estas bases de datos son utilizadas en numerosos campos de la ciencia, como en data mining (Alqadah *et al.*, 2010) (Colantonio *et al.*, 2010) (Sun y Nobel, 2008), text mining (Mimaroglu y Simovici, 2007), bioinformática (Figueroa *et al.*, 2004) (Perco *et al.*, 2005) (Shmulevich y Zhang, 2002), ingeniería (Lu *et al.*, 2008) o paleontología (Puolamaki y Mannila, 2006), entre otros.

En el ámbito de la bioinformática, el significado de los valores 1 y 0 en estas bases de datos depende del contexto. Por ejemplo, cuando trabajamos con genes y proteínas, si el gen r codifica a la proteína de clase c, entonces $\langle r, c\rangle$ es igual a 1; en otro caso, es igual a 0 (Koyuturk *et al.*, 2004). Comúnmente, los valores binarios 1 y 0 implican que bajo la condición experimental c, el gen r está expresado o no, respectivamente (Prelic *et al.*, 2006) (Zhang *et al.*, 2010). En algunas bases de datos sobre la regulación de la transcripción, un elemento es igual a 1 si el factor de transcripción está asociado a un motivo que actúa sobre un gen concreto (Uitert *et al.*, 2008). Incluso la codificación binaria ha sido aplicada al alineamiento local múltiple de secuencias de ARN (Wang *et al.*, 2007). En este caso, el valor 1 implica que un alineamiento local de múltiples subsecuencias está presente en una secuencia concreta; 0 en caso contrario.

Como se puede observar, la codificación binaria se aplica a distintos problemas biológicos y es aquí donde radica el mayor atractivo de diseñar técnicas de biclustering para este tipo de datos. Más allá de las bases de datos de expresión genética, estas técnicas han de ser lo suficientemente flexibles para adaptarse a cualquier tipo de base de datos de entrada, con respecto al tamaño y a la forma, lo que puede conllevar una gran dificultad. Además, han de proporcionar los métodos de evaluación de resultados adecuados para cada caso. Sin embargo, el hecho de contar con solo dos valores diferentes en la información a analizar debe suponer una ventaja. Por ejemplo, con respecto a los distintos tipos de posibles biclusters que se pueden encontrar. Normalmente, son los biclusters constantes los más buscados y, en concreto, aquellos con todos los valores iguales a 1 (Prelic *et al.*, 2006) (Mimaroglu y Simovici, 2007). No obstante, son varias las técnicas que permiten la inclusión de ceros para flexibilizar los resultados finales y permitir la inclusión de ruido (Koyuturk *et al.*, 2004) (Uitert *et al.*, 2008).

Además, esta simple codificación debería ser aprovechada para simplificar las acciones de procesado de esta información. Con respecto a las acciones de pre-procesado, éstas no se pueden descartar. Si la base de datos de entrada no es binaria, se deben proporcionar métodos fiables de binarización que hagan perder la menor cantidad de información posible. Todas estas consideraciones, y otras más, son las que se deben tener en cuenta a la hora de analizar los distintos algoritmos de biclustering especialmente diseñados para las bases de datos binarios. A continuación, mostramos, utilizando la clasificación de tipos de algoritmos de biclustering de la sección 4.3.4 , los representantes más relevantes de este tipo de técnicas, teniendo en cuenta que su número es bastante reducido en comparación con las técnicas de biclustering convencionales.

4.5.2. Divide y vencerás

Binary inclusion-maximal biclustering algorithm (Bimax)

El algoritmo *Bimax* (Prelic *et al.*, 2006) es el representante más importante de los algoritmos de biclustering especialmente diseñados para bases de datos binarios, por ser uno de los más referenciados (Serin y Vingron, 2011) (Harpaz y Perez, 2010) (Bhattacharya y Rajat, 2009) (DiMaggio *et al.*, 2008). Curiosamente, fue presentado en un artículo cuyo principal objetivo era otro: proporcionar una comparación y evaluación sistemática de los algoritmos de biclustering más relevantes en cuanto a la clasificación de genes se refiere. En particular, los autores se centran en los siguientes cuestiones: ¿qué metodología de comparación y evaluación es adecuada para las técnicas de biclustering? ¿cómo de relevantes son los biclusters seleccionados por cada técnica? ¿cómo comparar métodos entre sí? Otra importante aportación de este trabajo fue una medida, el *Match Score*, que se utiliza para determinar el grado de precisión de una técnica de biclustering cuando intenta encontrar una serie de biclusters objetivo escondidos en una base de datos artificial. Esta medida será utilizada posteriormente, en el capítulo de experimentación de este documento de tesis (ver Cap. 6, Sec. 6.2). Así, dentro de este contexto, el algoritmo *Bimax* es utilizado como técnica de referencia, al proponer un modelo de datos muy sencillo que refleja la idea fundamental del biclustering.

El algoritmo *Bimax* presenta un modelo de datos relacionado con la expresión genética en el que se asumen sólo dos posibles valores para cada valor de expresión de un gen. De esta manera, un conjunto de m experimentos en microarrays aplicados a n genes pueden ser representados por una matriz binaria E^{nxm}, en la que un elemento e_{ij} es igual a 1 siempre que el

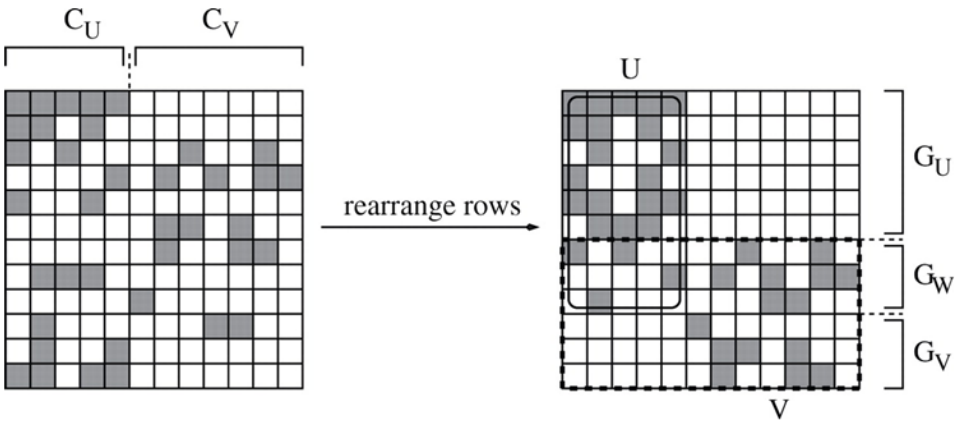

Figura 4.6: Proceso de división sobre la matriz de entrada llevado a cabo por el algoritmo *Bimax*.

gen i responda positivamente a la condición j; 0 en otro caso. Un bicluster *(G,C)* está formado por un conjunto de genes $G \subseteq \{1, \ldots, n\}$ que responden positiva y uniformemente a un conjunto de condiciones $C \subseteq \{1, \ldots, m\}$. En otras palabras, el par *(G,C)* define una submatriz de E en la que todos los elementos son iguales a 1. Sin embargo, según esta definición, cualquier elemento de la matriz e_{ij} que sea igual a 1 representa en sí mismo un bicluster. Para evitar este tipo de resultados sin interés, Prelic et al. define otro tipo de bicluster objetivo: el *bicluster de inclusión máxima*, que es aquél que no está contenido en su totalidad en otro bicluster. El hecho de enumerar todos los posibles biclusters de inclusión máxima puede llegar a ser una tarea de gran complejidad algorítmica. Dado un grafo bipartito G, definido a partir de la matriz binaria E entendida como matriz de adyacencia, la complejidad de utilizar dicho grafo para obtener los biclusters máximos puede llegar a ser $O(nm\beta \log \beta)$, siendo β el número de todos los posibles biclusters de inclusión máxima existentes en E^{nxm}.

Para reducir esta complejidad, los autores proponen un algoritmo muy sencillo y rápido basado en el método de divide y vencerás. La idea básica, tal y como se puede apreciar en la figura 4.6, es particionar la matriz de entrada en tres submatrices. Una de esas submatrices contendrá solo elementos iguales a cero y será descartada. Posteriormente, el algoritmo es aplicado recursivamente sobre las otras dos submatrices, U y V, hasta que encontremos una submatriz con todos sus elementos iguales a 1. Para dividir la matriz de entrada en tres submatrices el conjunto de columnas de la matriz es dividido en dos conjuntos: C_U y C_V, tomando como plantilla una fila (la primera fila en el ejemplo de la figura 4.6). Según el algoritmo propuesto

por los autores, dicha fila es elegida de manera aleatoria entre el conjunto de filas que contengan un número de unos que sea mayor que cero pero menor que el número de columnas de la matriz. El subconjunto C_U estará formado por aquellas columnas en las que la fila elegida tenga algún 1. El subconjunto C_V será su complementario. En la figura de ejemplo, la fila elegida tiene una disposición de unos tal que C_U y C_V se distinguen perfectamente. Posteriormente, las filas de la matriz son reorganizadas de tal manera que primero vendrán aquellos genes, G_U, que solo respondan positivamente las condiciones del conjunto C_U, después aquellos genes, G_W, que respondan a condiciones pertenecientes a C_U y C_V y, finalmente, aquellos genes, G_V, que solo respondan a las condiciones de C_V. Estos tres conjuntos de genes, G_U, G_W y G_V, definen en combinación con los subconjuntos de columnas, C_U y C_V, las submatrices resultantes, U y V, sobre las que se aplicará recursivamente el mismo procedimiento. Si estas submatrices no comparten ningún elemento, es decir, $G_W = \emptyset$, entonces se pondrán procesar de manera independiente. Sin embargo, si no es el caso, hay que tener cuidado de generar biclusters a partir del conjunto V solo si contienen al menos una columna del conjunto C_V.

Bimax tiene dos simples parámetros de entrada: número mínimo de filas y número mínimo de columnas permitidas en los biclusters finales. De esta manera, se reduce el espacio de búsqueda y se limita el número de resultados. La complejidad de *Bimax* es de $O(nm\beta min\{n, m\})$, siendo β el número total de biclusters de inclusión máxima existentes. Así, se puede concluir que es un algoritmo fácil de utilizar y rápido. Sin embargo, hay dos desventajas que pueden llegar a ser importantes. En primer lugar, al devolver la totalidad de biclusters de inclusión máxima, el número de resultados puede ser tal que sea inabarcable poder almacenarlos y/o procesarlos posteriormente. Y, en segundo lugar, un defecto asociado a las técnicas de divide y vencerás. El rendimiento de dichas técnicas depende, en gran medida, de poder balancear de manera equilibrada la carga de los distintos subproblemas en los que el problema original es dividido. Sin embargo, cuando la división es dependiente de los datos de entrada, el comportamiento es imprevisible. Por ejemplo, en el caso de *Bimax*, la división de la matriz de entrada depende de la fila que se utilice como plantilla. A todo esto hay que añadir la gran cantidad de recursos que se necesitan cuando se utiliza la recursividad. Estos factores son analizados en la comparativa que posteriormente se hará entre la propuesta de esta tesis doctoral y este algoritmo (ver Cap. 6, Sec. 6.1.3).

Los resultados de *Bimax* fueron evaluados mediante los procedimientos propuestos en este artículo. Primero, validación en bases de datos sintéticas, donde la medida Match Score fue utilizada para cuantificar la precisión

a la hora de localizar distintos tipos de biclusters (con distintos grados de solape, biclusters constantes, aditivos, etc.) escondidos en los datos de entrada. Posteriormente, distintas bases de datos biológicas fueron utilizadas, interviniendoc distintos repositorios de información biológica.

4.5.3. Búsqueda Exhaustiva

Text Biclustering Using Bit Sequences (TEBUBS)

El algoritmo que se describe a continuación, *TEBUBS* (Mimaroglu y Simovici, 2007), es el único incluido en este estado del arte que no está relacionado con bases de datos biológicas. En realidad, esta técnica ha sido diseñada para llevar a cabo minería textual o text mining. Por ejemplo, encontrar documentos en los que aparece un determinado término o agrupar documentos en función de una o más temáticas. En concreto, el objetivo es obtener biclusters a partir de una base de datos binarios que relacionan documentos y distintos términos.

Para los autores, la base de datos es un sistema documental constituido por la siguiente tupla: $S = (D, T, \mathbf{to})$, donde D y T son dos conjuntos finitos de documentos y términos, respectivamente, y $\mathbf{to} : DxT \rightarrow \{0, 1\}$ es una función denominada *ocurrencia de término*. Es decir, $\mathbf{to}(d, t) = 1$ si el término t aparece en el documento d; $\mathbf{to}(d, t) = 0$ en otro caso. A continuación, el bicluster objetivo de este algoritmo es definido. Sea λ un número tal que $1 \leq \lambda \leq \mid D \mid$, un *k-term λ–bicluster* es un par de conjuntos no vacíos, (I, J), tal que $I \subseteq D$, $J \subseteq T$ y $\mathbf{to}(i, j) = 1$ para cada $i \in I$ y $j \in J$, siendo además $\mid J \mid = k$ y $\mid I \mid \geq \lambda$. Es decir, estamos hablando de un bicluster constante formado por elementos iguales a 1 y que tiene un número de términos igual a k y un número de documentos, como mínimo, igual a un parámetro de entrada, λ.

El algoritmo exhaustivo diseñado por los autores, que recibe la base de datos binarios $S(D, T)$ y el mínimo número de documentos λ, consta de dos partes bien diferenciadas. En primer lugar, todos los *2-term λ–biclusters* son generados. Esta es la parte computacionalmente más compleja. Por ello, se lleva a cabo un filtrado previo de la base de datos y sólo son utilizados aquellos términos considerados válidos (*UsefulTerms*). Un término es válido cuando la columna que lo representa contiene un número de unos igual o mayor que λ. Esta medida es bastante lógica, ya que los términos descartados nunca podrían formar parte de un bicluster válido. El problema aquí es que los autores no indican en qué consiste este primer proceso de búsqueda.

La segunda parte del algoritmo es un proceso iterativo que consiste en aumentar la dimensión de los términos de los biclusters. Es decir, partimos

de un conjunto B con los *2-term λ–biclusters*. A continuación, por cada bicluster $(I, J) \in B$ recorremos todos los términos de la base de datos, t_x, tal que $t_x \in (UsefulTerms - J)$. Este término, t_x, pasará a formar parte del conjunto J del bicluster procesado si se comprueba que, para todos los términos $t_k \in J$, existe un *2-term λ–biclusters*, (I', J') con las siguientes características: $J' = \{t_x, t_k\}$ y $I \subseteq I'$. Así, tras procesar todos los biclusters iniciales del conjunto B se creará un conjunto de *3-term λ–biclusters*, que serán utilizados en la siguiente iteración para intentar aumentar en uno su número de términos. Es decir, para crear *n-term λ–biclusters* solo son utilizados los *(n-1)-term λ–biclusters* generados en la iteración anterior. De esta forma, evitamos generar resultados duplicados. Este proceso iterativo acaba cuando en una iteración no se haya conseguido ninguna expansión.

Este es un algoritmo muy sencillo que solo tiene un parámetro importante: λ, el cual restringe el espacio de búsqueda de tal manera que, a mayor valor de λ un número menor de biclusters será generado, como también disminuirá el tiempo de ejecución. Un problema que presenta esta técnica consiste en que no aprovecha el hecho de que los datos son binarios, es decir, este mismo método puede ser utilizado en cualquier tipo de base de datos, sustituyendo el hecho de que los valores del biclusters han de ser igual a 1 por cualquier otro criterio. Por ejemplo, que la distancia euclídea entre los valores sea menor que un determinado umbral, etc. Al no ser un algoritmo especialmente diseñado para bases de datos biológicas, no podemos saber cómo se comportará si es aplicado a, por ejemplo, una matriz de expresión genética formada por miles de genes. Además, los autores no explican cómo se desarrolla la primera parte de su algoritmo, que es la que más tiempo de ejecución requiere.

BiSim

BiSim, (Nighat y Muhammad, 2009), tiene como objetivo mejorar el rendimiento del algoritmo *Bimax* (ver Sec. 4.5.2). Los autores se basan en el hecho de que *Bimax* recorre innumerables veces, mediante una serie de llamadas recursivas, las mismas zonas de la matriz de entrada antes de dar el resultado final. Además, presentan una situación hipotética en la que, tras ser aplicado a una gran base de datos de entrada y tardar un tiempo considerable, *Bimax* es incapaz de devolver ningún resultado.

Para superar esta desventaja, los autores presentan un algoritmo que recorre una sola vez la matriz original y, posteriormente, combina esos resultados parciales para obtener los biclusters finales. En primera instancia, recorren cada fila de la matriz y detectan los llamados *row wise biclusters*,

para posteriormente almacenarlos en un vector. Estos biclusters parciales están formados por una sola fila y un conjunto de columnas contiguas para las que la matriz, en dicha fila, contiene elementos iguales a 1. Cada elemento del vector que almacena los *row wise biclusters* está formado por tres valores: el número de la columna de inicio, el de la columna de fin y el número de la fila. Sea M una matriz binaria de ejemplo:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Si aplicáramos el primer recorrido a esta matriz, obtendríamos los siguientes *row wise biclusters*:

- $(C_i, C_f, F) = (1,2,1)$
- $(C_i, C_f, F) = (5,5,1)$
- $(C_i, C_f, F) = (3,5,2)$
- $(C_i, C_f, F) = (1,1,3)$
- $(C_i, C_f, F) = (3,5,3)$
- $(C_i, C_f, F) = (1,4,4)$

donde C_i simboliza el número de la columna de inicio, C_f el número de la columna de fin y F el número de la fila. Posteriormente, comienza otro proceso que es independiente de la matriz original, ya que se basa únicamente en la información almacenada sobre estos biclusters iniciales. Este segundo proceso tiene como objetivo intentar combinar todos los *row wise biclusters* entre sí, buscando que de cada combinación surja un nuevo resultado. Los autores plantean 9 situaciones diferentes que se pueden dar durante una de estas combinaciones y, para cada una de ellas, se establece un resultado. En realidad estas combinaciones implican llevar a cabo la intersección de los distintos conjuntos de columnas contiguas que contiene cada bicluster. De esta intersección aparecerá un nuevo bicluster que se añadirá al vector original. A continuación se exponen dos ejemplos:

- Ejemplo 1: (3,5,2) vs (3,5,3) = (3,5,(2,3)). En esta ocasión, tanto las columnas de inicio como las de fin coinciden y se forma un bicluster mayor uniendo las filas. Los dos primeros biclusters son eliminados del vector.

- Ejemplo 2: (1,2,1) vs (1,4,4) = (1,2,(1,4)). El segundo bicluster incluye al primero, es decir, coincide la columna de inicio pero no la de fin. En este caso se añade al primer bicluster la fila del último. El nuevo bicluster se añade al vector pero los dos originales no son eliminados y seguirán estando en el vector para poder ser combinados posteriormente.

Este algoritmo plantea muchas dificultades que no están resueltas en el trabajo presentado por Nighat et al. En primer lugar, la creación de los biclusters fila iniciales depende de la ordenación de las columnas de la matriz. Así, los autores no dejan claro si el algoritmo es determinista, es decir, si siempre devuelve el mismo resultado independientemente del orden de las columnas. En segundo lugar, no se aclara cuál es la condición de parada del segundo proceso del algoritmo. Es decir, los biclusters parciales se van combinando todos con todos y en ese proceso se crean nuevos y se eliminan antiguos, con lo que el conjunto final va cambiando. Cada bicluster nuevo ha de ser combinado con todos los demás. Es de suponer que el algoritmo acabará cuando ya se no generen nuevos biclusters. Tampoco se habla de parámetros que limiten el espacio de búsqueda, por lo que es probable encontrar resultados finales formados por una sola fila y una sola columna. La complejidad del algoritmo es de $O(n * m)$, por lo que mejora a la del algoritmo *Bimax*. Sin embargo, la parte experimental de este trabajo es tan pobre que no aporta ningún dato fiable acerca de las pruebas comparativas realizadas: ni resultados obtenidos, ni bases de datos utilizadas, tiempo de ejecución, etc. En conclusión, es una idea interesante pero la investigación asociada no está bien concluida.

4.5.4. Identificación de parámetros de distribución.

Biclustering Gene-Feature Matrices for Statistically Significant Patterns

El trabajo de Koyuturk et al. (Koyuturk *et al.*, 2004) fue de los primeros en abordar la generación de biclusters en bases de datos binarios por medio de modelos estadísticos. Para ello, se definió el concepto de submatrices "inusualmente densas", se estableció una significatividad estadística como función a optimizar y se idearon rápidas heurísticas para afrontar el problema.

Dada una matriz G formada por M filas, N columnas y K elementos iguales a 1, el objetivo es encontrar subconjuntos de filas y columnas que determinen una submatriz lo suficientemente densa para ser considerada

significativa, desde el punto de vista estadístico. Se parte de la suposición de que en la matriz de entrada, la probabilidad de que un elemento sea igual a 1 es $\rho = \frac{K}{MN}$. Además, para un arbitrario conjunto de m filas y n columnas, se establece que la submatriz formada a partir de dichos conjuntos tiene un número de elementos iguales a 1 denominado k. Se considera que dicho número k proviene de una distribución binomial con los parámetros m, n y ρ, usando para ello el límite de Chernoff:

$$Pr\{k \geq mn\rho(1+\epsilon)\} \leq e^{\frac{-mn\rho\epsilon^2}{3}} \tag{4.24}$$

para $\epsilon > 0$. A partir de aquí, los autores indican que están interesados en descubrir todas las submatrices tal que la probabilidad de encontrar un número de unos igual a k sea menor que P^*, siendo P^* un valor escogido por el usuario. Así, a partir de la ecuación anterior de puede definir que un bicluster es significativo sí:

$$e^{\frac{-mn\rho\epsilon^2}{3}} \leq P* \tag{4.25}$$

Entonces,

$$\frac{-mn\rho\epsilon^2}{3} \leq -Ln(P*) \tag{4.26}$$

Si resolvemos esta última ecuación, con $E = -Ln(P*)$, tenemos que una submatriz de m filas, n columnas y k elementos iguales a 1 es significativa estadísticamente hablando si $k \geq mn\rho(1+\epsilon)$, donde

$$\epsilon \geq \sqrt{\frac{3E}{mn\rho}} \tag{4.27}$$

A partir de este resultado, los autores definen una función objetivo de la siguiente manera:

$$C(m,n,k) = k - mn\rho - \sqrt{3Emn\rho} \geq 0 \tag{4.28}$$

Teniendo en cuenta que cuanto mayor sea el valor de $C(m,n,k)$ más interesante será el bicluster, el resultado deseado serán aquellos biclusters para los que dicha función tenga un máximo local. Para alcanzar dicho objetivo, se define una heurística iterativa basada en proyecciones alternativas entre el espacio de filas y de columnas. El espacio de filas va a venir determinado por un vector x de dimensión M tal que $x(i) = 1$ sí y sólo sí la fila i-ésima de G pertenece a la submatriz bajo análisis. Para las columnas, existe un vector equivalente y de dimensión N.

El algoritmo propuesto por los autores recibe como entrada una matriz binaria G y el valor de significatividad deseada, *(-Ln(P*))*. Inicialmente se establece un vector y aleatorio. A continuación, se desarrolla un proceso iterativo en el que se ejecutan dos pasos. En primer lugar, se busca un vector x que maximice la función objetivo. Una vez obtenido dicho vector, se vuelve a depurar el vector y para maximizar de nuevo dicha función. El bucle finaliza cuando en la última iteración el valor de la función objetivo no haya sido modificado. Tal y como se puede observar, el algoritmo en cuestión devuelve un solo bicluster. Los autores proponen dos estrategias diferentes para encontrar todos los biclusters considerados interesantes:

1. Ejecutar el algoritmo varias veces para obtener un conjunto de biclusters. Eliminar posteriormente aquellos redundantes desde el punto de vista de la significatividad y el solape. Devolver los biclusters restantes ordenados a partir de su significatividad estadística.

2. Ejecutar varias veces el algoritmo hasta seleccionar un bicluster que tenga una significatividad máxima. Devolver dicho bicluster y utilizarlo como entrada del algoritmo para encontrar patrones significativos relacionados con él. Repetir el proceso hasta que no se encuentren más patrones interesantes.

Este trabajo define un tipo de bicluster binario formado por un deseado nivel de unos, pero también aceptando valores iguales a 0. En principio, los autores no definen el significado de esos ceros, representando probablemente cierto nivel de ruido. Aunque el algoritmo tiene pocos parámetros, el hecho de contar al comienzo con cierto grado de aleatoriedad y de generar un solo bicluster cada vez, dificulta su uso para obtener un conjunto definido de resultados. La propuesta, desde el punto de vista de la complejidad computacional, solo implica la multiplicación de vectores dispersos, la cual puede ser ejecutada en un tiempo $O(K)$, siendo K el número de elementos iguales a 1 en la matriz de entrada. Sin embargo, no está claro el número de veces que, en función de la estrategia escogida, se ha de ejecutar el algoritmo para obtener un conjunto final de resultados óptimo.

La nueva técnica es probada con dos bases de datos. La primera de ellas es parte de la competición de minería de datos KDD-cup del 2001, la cual incluye genes y una serie de atributos como clases de proteínas, fenotipos, motivos, etc. La otra base de datos es una matriz de expresión genética obtenida de la colección de datos GEO del NBCI (`http://www.ncbi.nlm.nih.gov/geo/gds/gds_browse.cgi`).

Biclustering Sparse Binary Genomic Data (BicBin)

Uitert et al. (Uitert *et al.*, 2008) diseñaron el algoritmo *BicBin* para solventar los problemas que, según ellos, ofrecían los dos únicos algoritmos de biclustering especialmente diseñados para datos binarios creados hasta la fecha: *Bimax* (Prelic *et al.*, 2006) y *Cmnk*, que es como llamaron a la solución ofrecida por Koyuturk et al. (Koyuturk *et al.*, 2004) y comentada en la sección anterior. El ámbito para el que *BicBin* está especialmente recomendado abarca cualquier base de datos binarios dispersa, es decir, con un número elevado de elementos iguales a 0 (gran cantidad de ruido). En este tipo de bases de datos, *Cmnk* y *Bimax* no se desenvuelven de manera satisfactoria: en los experimentos basados en datos artificiales, *Cmnk* no fue capaz de encontrar un bicluster escondido en un entorno ruidoso. El problema de *Bimax* deriva de la gran cantidad de resultados que devuelve cuando la dimensión de las columnas es muy grande, además de no poder devolver biclusters que permitan un cierto grado de error, es decir, algún elemento igual a 0.

La técnica expuesta en este trabajo consta de tres aportaciones principales: una función objetivo para evaluar las submatrices, un algoritmo capaz de restringir el espacio de búsqueda y otro algoritmo para obtener todos los biclusters de una forma ordenada. A continuación será definida la función objetivo. Al igual que en el trabajo de Koyurtk et al. (Koyuturk *et al.*, 2004), se considera que en una submatriz de $m \times n$ el número de elementos iguales a 1, $X_{m,n}$, sigue una distribución binomial en la que intervienen los parámetros m, n y ρ, siendo este último valor la proporción de unos. Como se tiene interés en aquellas submatrices suficientemente densas con elementos iguales a 1, se buscan en realidad aquellos bicluster con una baja probabilidad $P(X_{m,n} \geq k)$. Debido al alto nivel de dispersión de las bases de datos a utilizar, el parámetro ρ será muy bajo, al igual que la probabilidad anteriormente definida. Ante la imposibilidad de trabajar con estos valores, los autores utilizan la versión multiplicativa del límite de Chernoff para definir la probabilidad P como:

$$P(X_{m,n} > k) \leq e^{-\frac{(k-mn\rho^2)}{3mn\rho}}, k \in [mn\rho, 2mn\rho] \tag{4.29}$$

$$P(X_{m,n} > k) \leq e^{-\frac{(k-mn\rho^2)}{k+mn\rho}}, k > 2mn\rho \tag{4.30}$$

Este límite sólo es válido para $k \geq mn\rho$, es decir, para un número k mayor que el número de unos esperado en una submatriz $m \times n$. Así, en vez de buscar submatrices con una baja probabilidad, podemos buscar submatrices que maximicen el exponente:

$$\tilde{C}(m,n,k) = \frac{(k-mn\rho)^2}{3mn\rho}, k \in [mn\rho, 2mn\rho] \tag{4.31}$$

$$\tilde{C}(m,n,k) = \frac{(k-mn\rho)^2}{k+mn\rho}, k > 2mn\rho \tag{4.32}$$

En la anterior función objetivo, el número de las filas y las columnas de la submatriz no se tiene en cuenta de manera separada, es decir, solo el producto mn es tenido en cuenta. Además, el hecho de maximizar el exponente hace que se favorezcan las matrices de mayor tamaño. Así, para tener en cuenta el tamaño del bicluster, los autores normalizan la función $\tilde{C}(m,n,k)$ con los factores $m^\alpha n^\beta$, con α y β en el intervalo [0.5,1]. Así, la función objetivo final es:

$$C_{\alpha\beta} = \frac{\tilde{C}(m,n,k)}{m^\alpha n^\beta} = \frac{(k-mn\rho)^2}{m^\alpha n^\beta 3mn\rho}, k \in [mn\rho, 2mn\rho] \tag{4.33}$$

$$C_{\alpha\beta} = \frac{\tilde{C}(m,n,k)}{m^\alpha n^\beta} = \frac{(k-mn\rho)^2}{m^\alpha n^\beta (k+mn\rho)}, k > 2mn\rho \tag{4.34}$$

Con respecto al algoritmo de búsqueda, denominado A_1, los autores utilizan la misma estrategia que la definida en el trabajo de Koyurtk et al., es decir, llevar a cabo proyecciones alternativas entre el espacio de filas y de columnas. De esta manera, se obtiene una submatriz que maximiza la función de búsqueda. Sin embargo, un bicluster máximo puede tener un número de unos que varía entre $mn\rho$ y mn. Así, se introduce un parámetro que el usuario utilizará para determinar la proporción de unos que una submatriz ha de tener para ser considerada un bicluster máximo. Se define pues el parámetro p_1 y, por consiguiente, el p_1–bicluster como un bicluster con una proporción de unos que es igual o mayor que p_1. Para poder obtener este tipo de bicluster, se diseña un nuevo algoritmo, A_2, el cual consiste en ejecutar el algoritmo A_1, teniendo como entrada su propia salida, tantas veces como sea necesaria hasta obtener la proporción de unos deseada. Finalmente, para generar todos los p_1–biclusters máximos se establece el algoritmo A_3, en el cual se ejecuta el algoritmo A_2 hasta encontrar un bicluster máximo, después dicho bicluster es sustituido en la matriz original por elementos iguales a 0 y de nuevo se vuelve a ejecutar A_2 si la matriz de entrada tiene todavía algún elemento igual a 1.

Las ventajas de este estudio frente al de Koyurtk et al. consisten en una mayor facilidad a la hora de actuar sobre bases de datos dispersas, el poder controlar la proporción de unos que ha de tener un bicluster y, además, tener la capacidad de buscar biclusters de distintas formas, es decir, biclusters con más filas que columnas, más columnas que filas o dando a las dos dimensiones

la misma importancia. Esto último se consigue manipulando el valor de los parámetros α y β. Es decir, si α es mayor que β, a la hora de añadir una columna o una fila al bicluster final con el mismo número de unos, la función $C_{\alpha\beta}$ aumenta más si se añade la columna. Lo contrario pasa si β es mayor que α. Si ambos valores son iguales, tanto la adición de la fila como de la columna tendrán el mismo efecto en la función objetivo. Finalmente, los algoritmos para obtener el total de biclusters máximos están mejor descritos y presentan mayores facilidades.

Las desventaja principal de *BicBin*, sin embargo, consiste en que se introducen 3 nuevos parámetros: p_1, α, β. Cada uno de ellos afecta definitivamente al resultado final. Ello implica la dificultad de determinar qué valores son los apropiados para cada base de datos. Además, el hecho de sustituir los biclusters obtenidos en el algoritmo A_3 puede desvirtuar la matriz original, evitando solapes entre biclusters y enmascarando patrones más próximos a la realidad.

El algoritmo *BicBin* es comparado con *Bimax* y *Cmnk*, utilizándose para ello bases de datos artificiales. Se verifica que *BicBin* es más efectivo a la hora de localizar biclusters que contienen algún elemento igual a cero, es decir, algún ruido. Además, se utilizan bases de datos biológicas que contiene información acerca de factores de transcripción. Los resultados son validados a partir de la base de datos GO (Consortium, 2006) y STRING (Mering *et al.*, 2007).

Binary Matrix Factorization (BMF)

La factorización de matrices no negativas (*Non-negative Matrix Factorization (NMF)*), ha sido una herramienta muy útil para analizar este tipo de base de datos (Lee y Seung, 2001). Esta técnica factoriza una matriz, con la restricción de no contener números negativos, en otras dos matrices no negativas de menor rango. Si además se cuenta, como función de coste, con el error cuadrado medio, la NMF es equivalente a una versión menos estricta del algoritmo de clustering K-means (Ding *et al.*, 2005). Sin embargo, la factorización NMF es incapaz de generar de manera explícita la estructura de un bicluster. Por este motivo, en el trabajo de Zhang et al. (Zhang *et al.*, 2010) se introduce el nuevo concepto *Binary Matrix Factorization (BMF)*. Básicamente, la idea es tomar una matriz binaria de expresión genética, X, y descomponerla en dos matrices binarias, W y H. Estas dos matrices resultantes conservan la integridad de la matriz original y además determinan de manera explícita los miembros que formarán los clusters de genes y condiciones (Li, 2005).

La factorización de matrices no negativas (NMF) consiste en descomponer una matriz X de $m \times n$ en otras dos: W, de tamaño $n \times r$ y H de tamaño $r \times m$, tal que $X \approx WH$. Este proceso se traduce en la siguiente optimización:

$$min_{W \geq 0, H \geq 0} \parallel X - WF \parallel_F^2 = min_{W \geq 0, H \geq 0} \sum_{ij} (X - WH)_{ij}^2 \tag{4.35}$$

donde el valor r es un valor predefinido que satisface $r < \frac{nm}{(n+m)}$. En general, el proceso a seguir es muy similar a los ya estudiados en este tipo de algoritmos, es decir, se fija una de las dos componentes y la otra se va modificando para optimizar la función objetivo. Pero, para poder obtener biclusters a partir de esta factorización, es necesario aplicar cambios a este modelo. El primer problema que se encuentran los autores es que los valores que forman las matrices W y H han de estar limitados, de tal manera que no excedan la magnitud de la matriz de entrada. Para ello, a partir de un determinado teorema, demuestran que existe una matriz diagonal D de tal manera que:

$$X = WH = (WD)(D^{-1}H) = W^*H^* \tag{4.36}$$

para las que se cumple que $W_{ij}^* \leq 1$ y $H_{ij}^* \leq 1$. La ecuación anterior es un proceso de normalización que asegura que W y H estén en la misma escala. Una vez aplicada la normalización, Zhang et al. proponen dos maneras de afrontar este problema de optimización: la primera de ellas consiste en utilizar una determinada función de penalización y en la segunda manera el uso de umbrales es la base. Sirva el siguiente ejemplo para ilustrar como funciona el algoritmo BMF. Sea X la siguiente matriz de entrada:

$$\mathbf{X} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 1 & \mathbf{0} & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & \mathbf{0} & \mathbf{1} & 0 & 0 & 0 & 0 \end{pmatrix}$$

Se pueden observar, resaltados en negrita, dos biclusters claros, compuestos en su mayoría por elementos iguales a 1. El primero de ellos está en la esquina superior derecha y el otro en la esquina inferior izquierda. Si se apllica el modelo BMF, se obtendrán como resultado las siguientes matrices:

$$\mathbf{W} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\mathbf{H} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

De esta manera, se puede recuperar de una manera clara los dos biclusters objetivo:

$$\mathbf{WH} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \end{pmatrix}$$

Durante la fase experimental, el método BMF es comparado con otros algoritmos de biclustering desde dos puntos de vista. En primer lugar, utilizan bases de datos sintéticas y la medida Match Score introducida por Prelic et al. (Prelic *et al.*, 2006). Posteriormente, se analizan tres bases de datos reales sobre tumores: *AML/ALL data* y *Central Nervous System Tumor Data* (Brunet *et al.*, 2004), y *lung cancer data* (Gordon y et al., 2002). La validación de los resultados biológicos se lleva a cabo mediante un análisis de enriquecimiento de genes proporcionado por la herramienta Onto-Express (Draghici y et al., 2003).

Al igual que todos los algoritmos que se basan en métodos estadísticos, BMF es un método complejo de utilizar. En primer lugar, están los parámetros de entrada y el nivel de aleatoriedad que se introduce en este tipo de modelado. En segundo lugar, se no conoce a priori qué tipo de bicluster se va a obtener, ni tampoco su número. La información acerca de su complejidad y tiempos de ejecución no es aportada por los autores y, finalmente, el problema de factorizar requiere encontrar el número óptimo de factores k, un problema no trivial que ya ha sido estudiado con anterioridad (Brunet *et al.*, 2004) (Carmona-Saez *et al.*, 2006).

4.5.5. Conclusiones

Son pocos los algoritmos que han sido diseñados especialmente para extraer biclusters a partir de bases de datos binarios. En el repaso presentado, la mitad de dichos métodos plantean modelos estadísticos que definen los biclusters objetivo. En concreto, el trabajo de Koyuturk et al. (Koyuturk *et al.*, 2004) fue el primero en proponer el concepto de biclusters significativamente densos y establecer una función de optimización para poder obtenerlos. Más tarde, Uitert et al. (Uitert *et al.*, 2008) mejoró dicho concepto y lo adaptó para que fuera más eficiente en bases de datos dispersas, es decir, con muchos elementos iguales a 0. También, incluyó una serie de parámetros que permiten escoger la forma de los biclusters resultado, desde el punto de vista de facilitar que el bicluster final tenga más filas que columnas o viceversa. Aunque este último trabajo mejoró la aportación inicial de Koyuturk et al., siguió arrastrando una serie de problemas derivados de la propia metodología: en primer lugar, son algoritmos complejos de usar debido a cierto nivel de aleatoriedad que introducen y al efecto de sus parámetros. En segundo lugar, existe la limitación de que es necesario ejecutar varias veces los algoritmos para poder obtener un conjunto determinado de biclusters. Incluso, en el trabajo de Uitert et al., se propone reemplazar los biclusters obtenidos por ceros en la matriz original cada vez que se vaya a comenzar una nueva ejecución, lo que puede desvirtuar la propia naturaleza de los datos. Finalmente, los creadores de la técnica BMF (Zhang *et al.*, 2010) aportaron una nueva visión del problema y adaptaron la factorización de matrices no negativas al problema de los datos binarios. Sin embargo, también la factorización trae consigo dificultades, como por ejemplo escoger el número apropiado de factores (Brunet *et al.*, 2004) (Carmona-Saez *et al.*, 2006).

Otros algoritmos proponen una búsqueda exhaustiva. Las dos propuestas presentadas presentan curiosas similitudes. Tanto el algoritmo *TEBUBS* (Mimaroglu y Simovici, 2007) como el algoritmo *BiSim* (Nighat y Muhammad, 2009) hacen un primer recorrido de la matriz de entrada para encontrar resultados parciales. Posteriormente, estos resultados son utilizados en fases posteriores como única fuente de datos. En el caso de *TEBUBS*, se extraen en primera instancia biclusters con un número mínimo de filas, prefijado por el usuario, pero con sólo 2 columnas. En la siguiente fase del algoritmo, cada bicluster generado intenta aumentar su dimensión de las columnas añadiendo aquéllas que no contenía previamente, pero que aparecen junto a las que actualmente tiene en el resto de biclusters. En cada iteración, la dimensión de las columnas aumenta en 1 y es el nuevo conjunto de biclusters

generados el que es entrada para la siguiente iteración. *BiSim* extrae, por cada fila de la matriz, un conjunto de biclusters parciales formados por las columnas contiguas cuyo valor es igual a 1. Posteriormente, esos biclusters parciales se combinan para dar lugar a nuevos resultados. La similitud de estas dos técnicas va más allá y en ambos trabajos no se presenta información relevante. Como por ejemplo, cómo se generan los biclusters iniciales en el caso de *TEBUBS* o hasta cuándo se han de combinar los biclusters entre sí en el caso de *BiSim*. Sí existen diferencias en el ámbito de aplicación de las mismas, ya que *TEBUBS* es la única técnica que no ha sido diseñada para bases de datos biológicas.

Bimax (Prelic *et al.*, 2006), es la única técnica de divide y vencerás presente en este resumen. Este algoritmo de biclustering es uno de los más referenciados. Y no le sobran motivos: es un algoritmo sencillo de utilizar y rápido, y que además puede extraer el número de total de biclusters máximos que existen en una base de datos binarios. Sin embargo, existen dos dificultades principales. En primer lugar, el número de resultados puede ser tan grande que es muy difícil trabajar con ellos posteriormente. Este tema puede llegar a ser tan crítico, que los autores realizan una estimación del número de posibles resultados que podemos obtener, y el tiempo de ejecución asociado, para bases de datos con distintas densidades de elementos iguales a 1 (ver material suplementario de este trabajo). En segundo lugar, la complejidad del algoritmo puede verse incrementada si la carga del problema original, cuando éste se divide en subproblemas, no está balanceada. También, el hecho de contar con grandes bases de datos (sobre todo con un gran tamaño en la dimensión de la matriz utilizada para llevar a cabo la división de la misma) puede llegar a ser un inconveniente.

Con respecto al efecto del ruido, indicar que solo las técnicas de biclustering sobre datos binarios basados en modelos estadísticos permiten la inclusión, a veces controlada, de algún cero. El resto de técnicas siempre obtienen biclusters constantes con todos sus elementos iguales a 1.

Finalmente, indicar que la única técnica que propone un método de binarización es *Bimax*, el cual se basa en fijar un umbral para la expresión de los genes. Así, sí un gen tiene un valor de expresión bajo una condición experimental superior a dicho umbral, su valor equivalente en la matriz binaria será un 1; 0 en caso contrario.

Parte III

Propuestas

Capítulo 5

Extracción de biclusters en bases de datos binarios

If you can't do Bioinformatics, you can't do Biology
J.D. Tisdall.

En este capítulo se describe la metodología de extracción de biclusters en bases de datos binarios, aportación principal de esta tesis doctoral. En primer lugar, se presenta la problemática que supuso el punto de partida y la justificación de la investigación llevada a cabo. A continuación, la metodología propuesta es ampliamente descrita, utilizándose para ello un ejemplo claro y sencillo. Finalmente, el algoritmo de biclustering es formalizado y las conclusiones finales presentadas.

Este trabajo ha sido publicado durante el año 2011 en la revista *Bioinformatics*, con el título *"A biclustering algorithm for extracting bit-patterns from binary datasets"*, contando con la colaboración de Antonio Pérez Pulido y Jesús Aguilar Ruiz, ambos pertenecientes a la universidad Pablo de Olavide de Sevilla.

5.1. Problemática y justificación

Tal y como ya se comentó en el capítulo anterior (ver Sec.4.5), las bases de datos binarios constituyen una forma compacta y simple de almacenar información acerca de las relaciones establecidas entre un grupo de objetos y sus posibles propiedades y son utilizadas en numerosos campos de estudio. Desde el punto de vista de la Bioinformática, los valores 1 y 0 pueden encerrar distintos significados según el contexto: genes cuya respuesta es positiva o negativa frente a un estímulo, factores de transcripción asociados a

un motivo, etc.

Las técnicas de clustering son de las herramientas más populares de las que dispone la minería de datos y que se utilizan para detectar la existencia de patrones y correlaciones intrínsecas en grandes bases de datos (Kerr *et al.*, 2008). Sin embargo, la imposibilidad de estas técnicas para detectar patrones locales ha propiciado la popularización de las técnicas de biclustering (Madeira y Oliveira, 2004), capaces de analizar de manera simultánea todas las dimensiones de una base de datos y, consecuentemente, extraer patrones locales de comportamiento que proporcionan un mejor entendimiento de los fenómenos biológicos. Durante los últimos años, aunque son varias las técnicas de biclustering que han sido especialmente desarrolladas para las bases de datos binarios, su número no es demasiado elevado. Se puede afirmar pues, que ampliar la oferta de técnicas disponibles y, por consiguiente, ofrecer mayor variedad y calidad de soluciones, es uno de los mayores atractivos que tiene esta especialidad concreta del biclustering. Paradójicamente, *Bimax* (Prelic *et al.*, 2006) es uno de los algoritmos de biclustering más conocidos (Serin y Vingron, 2011) (Harpaz y Perez, 2010) (Bhattacharya y Rajat, 2009) (DiMaggio *et al.*, 2008) y está precisamente pensado para bases de datos binarios. *Bimax* tiene como objetivo obtener todos los biclusters máximos cuyos elementos son iguales a 1. Sin embargo, este método genera un número tan elevado de biclusters que la tarea de procesarlos resulta inabarcable. El algoritmo *Bimax* está basado en una técnica de divide y vencerás que proporciona una rápida respuesta. Aún así, el rendimiento de estas técnicas puede verse gravemente afectado si el número de llamadas recursivas es demasiado alto. Todo esto lleva a considerar que el objetivo de mejorar el rendimiento de uno de los algoritmos de biclustering más citados es otro importante atractivo de la investigación presentada en esta tesis doctoral. De igual forma, y tal y como se ha recopilado en el estado del arte (ver Sec.4.5), los otros algoritmos de biclustering sobre datos binarios existentes también plantean algunos inconvenientes susceptibles de mejora. Por ejemplo, el algoritmo *BicBin* (Uitert *et al.*, 2008) requiere parámetros de entrada complejos que dificultan su utilización. Además, cada vez que un bicluster se genera durante el proceso iterativo que dicho algoritmo plantea, todos sus elementos son sustituidos por ceros en la base de datos original, desvirtuándola. El método de biclustering basado en factorización binaria presentado por (Zhang *et al.*, 2010) requiere encontrar el número óptimo de factores k, un problema no trivial que ya ha sido estudiado con anterioridad (**?**) (Carmona-Saez *et al.*, 2006). Finalmente, otras técnicas no dejan totalmente claro cuales son los métodos seguidos (Mimaroglu y Simovici, 2007) (Nighat y Muhammad, 2009).

5.2. Propuesta de metodología

En esta tesis doctoral se presenta un nueva alternativa para extraer biclusters de bases de datos binarios: *BiBit* (Bit-Pattern Biclustering Algorithm). El objetivo es obtener biclusters máximos cuyos elementos son todos iguales a 1 (*bit–patterns*). La técnica de procesamiento de patrones de bits utilizada y la búsqueda selectiva hacen que esta nueva propuesta alcance velocidades de procesamiento muy altas, además de generar un número razonable de resultados. Sus escasos y simples parámetros de entrada y su determinismo hacen de *BiBit* un algoritmo fácil de utilizar, hecho muy importante si se tiene en cuenta el tipo de usuario al que va destinado (biólogos y médicos). Finalmente, la experimentación llevada a cabo ha demostrado que el algoritmo *BiBit* no se ve afectado por la forma, tamaño o densidad de las bases de datos de entrada y que es capaz de obtener, esencialmente, los mismos resultados que *Bimax* pero significativamente más rápido.

5.3. Metodología

La metodología que se presenta en esta tesis doctoral está compuesta de dos fases principales: codificación y búsqueda (ver Fig. 5.1). Los únicos parámetros de entrada necesarios son: la base de datos binaria, B, el mínimo número de filas, *mnr* y el mínimo número de columnas, *mnc*, permitidos en un bicluster final. Previo al análisis de cada una de las fases, se presentan a continuación una serie de definiciones importantes.

5.3.1. Definiciones

Definición 1.
Una matriz de entrada se define como la siguiente terna $B = (R, C, \ell)$*, en la que* R *y* C *son dos conjuntos finitos definidos como el* conjunto de filas *y el* conjunto de columnas*, respectivamente, y* $\ell : R \times C \longrightarrow \{0, 1\}$ *es una función binaria. Denominaremos el valor binario* $\ell(r, c)$ *como* $\langle r, c \rangle$.
La matriz binaria $B = (R, C, \ell)$, con $N = |R|$ y $M = |C|$, puede ser descompuesta en N conjuntos de M bits: $B = \{r_1, r_2, \ldots, r_N\}$, con $ri = \{b_{i1}, b_{i2}, \ldots, b_{iM}\}$, y siendo $b_{ij} = \{0, 1\}$. Sobre estos grupos de conjuntos de bits puede ser aplicada un álgebra Booleana $\beta(\wedge$ (AND),$\vee$ (OR), $'$ (NOT)) definida a partir de la función binaria ℓ.
Definición 2.
Un bit-pattern es un bicluster compuesto por un par de conjuntos no vacios (I, J)*, con* $I \subseteq R$ *y* $J \subseteq C$*. El conjunto de columnas* $J = \{c_1, c_2, \ldots, c_k\}$ *es*

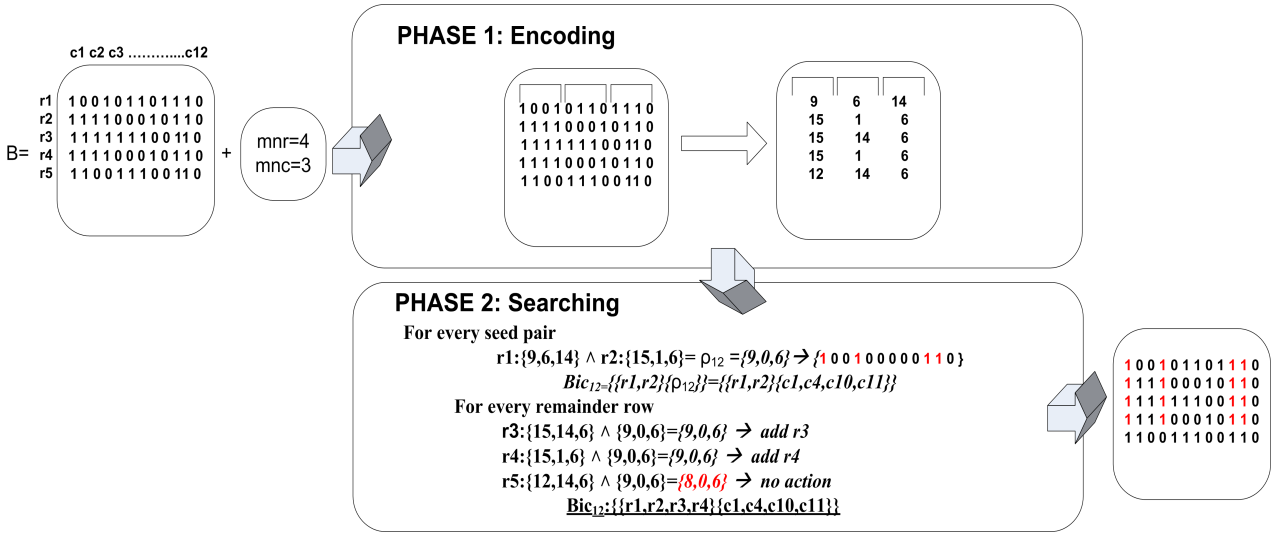

Figura 5.1: Representación esquemática de la metodología presentada en esta tesis doctoral y que, además, incluye un pequeño ejemplo. La matriz binaria B y los parámetros de usuario, definidos como *the minimum number of rows* (mnr) y *the minimum number of columns*(mnc), son las únicas entradas del método. El proceso de extracción de patrones de bits (bit–patterns) se divide en dos fases secuenciales. En la primera fase, un proceso de codificación reduce el tamaño de la base de datos. Durante dicho proceso, las columnas son divididas en palabras de bits de una cierta longitud (longitud igual a 4 en este ejemplo). Cada palabra de bits es traducida de manera independiente a su representación entera. La segunda fase está relacionada con el proceso de búsqueda de biclusters. Cada pareja de filas genera lo que llamamos un patrón ρ. Si el número de elementos de ρ es mayor o igual que el parámetro de entrada mnc y dicho patrón no ha sido procesado con anterioridad, se crea un bicluster inicial. Posteriormente, más filas serán añadidas a dicho bicluster. Al final del proceso, si el número de filas que incluye el bicluster final es mayor o igual que el parámetro de entrada mnr, dicho bicluster será considerado como un resultado válido.

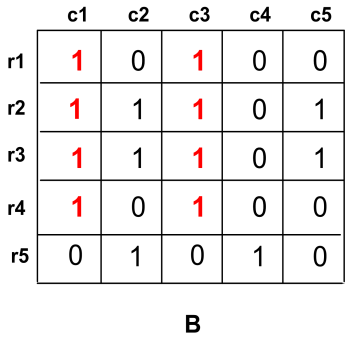

Figura 5.2: En esta matriz binaria, B, se destaca en rojo un bicluster tipo obtenido por nuestra metodología (bit–pattern).

lo que se conoce como un ***patrón*** *si por cada* $c_i \in J$ *y por cada pareja de filas* $r, r' \in I$*, se cumple que* $\langle r, c_i\rangle \wedge \langle r', c_i\rangle = 1$.
Por ejemplo, sea B una matriz binaria como la que aparece en la figura 5.2. Como se puede apreciar, el bit-pattern destacado en rojo aparece formado por las filas $\{r_1, r_2, r_3, r_4\}$ y las columnas $\{c_1, c_3\}$. Este último conjunto de columnas se conoce como *patrón*.
Definición 3.
El bit–pattern (I, J) *se considera un bicluster máximo si y sólo si es un bicluster y no está totalmente contenido en cualquier otro bicluster.*
Es importante mencionar que el objetivo de esta metodología no es el de extraer todos los posibles biclusters máximos de una base de datos. El algoritmo *BiBit* extrae un subconjunto del total de biclusters máximos. A continuación, se define el número máximo de componentes que puede tener dicho subconjunto. Consideremos el conjunto $L = \{(r_1, r_2), (r_1, r_3), \ldots, (r_N, r_{N-1})\}$ como el conjunto de todas las posibles parejas de filas, con $|L| = \frac{N!}{2!*(N-2)!}$. Si cada pareja (r_i, r_j) es considerada como una semilla potencial a partir de la cual un *bit–pattern* puede ser generado, el valor $|L|$ es el máximo número de biclusters máximos que pueden ser extraídos por el algoritmo *BiBit*. Es decir, cada pareja de filas crea un *patrón*, ρ (tal y como se puede observar en la figura 5.3), que será utilizado para crear un *bit–pattern*, añadiendo nuevas filas que sean compatibles con dicho patrón ρ.

En resumen, los biclusters objetivo de esta tesis serán submatrices máximas formadas por unos y creadas a partir de un patrón obtenido por la aplicación del operador booleano $\wedge$ a una pareja de filas. Además, el máximo número de biclusters máximos que pueden ser extraídos está limitado por el número de posibles parejas de filas.

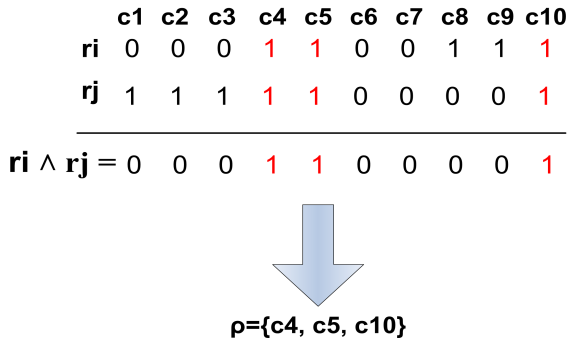

Figura 5.3: La pareja compuesta por las filas r_i y r_j crean el nuevo patrón $\rho = \{c_4, c_5, c_{10}\}$ a partir del cual un nuevo bit–pattern podrá ser creado.

5.3.2. Fase 1: Codificación.

El objetivo principal de esta fase de pre-procesamiento es reducir el tamaño de la base de datos y, consecuentemente, aumentar la eficiencia del algoritmo *BiBit*. Ello se consigue transformando la matriz binaria original en una matriz formada por números enteros. Tal y como se puede observar en nuestro ejemplo de la figura 5.1, cada fila, r_i, se divide en palabras de bits de un determinado tamaño w (fijado a 4 en este ejemplo). Este proceso de división se lleva a cabo de derecha a izquierda, completando con ceros la última palabra si ésta no llegara a alcanzar el tamaño prefijado. Después, cada palabra se traduce de manera individual a su representación entera. Así pues, la matriz $B = (R, C)$ vería reducida la dimensión de las columnas tantas veces como el valor del tamaño, w, escogido para las palabras: $\frac{|C|}{w}$. Se puede observar un ejemplo detallado del proceso de codificación de la fila 1 de la matriz ejemplo de la figura 5.1 en la figura 5.4. Este preprocesamiento disminuirá el número de operaciones necesarias en la siguiente fase, ayudando así a mejorar su rendimiento.

5.3.3. Fase 2: Búsqueda.

La búsqueda es un proceso iterativo que, por cada posible pareja de filas, lleva a cabo una serie de acciones. Para ilustrar su funcionamiento, a continuación se desarrollará la búsqueda completa de biclusters sobre la matriz de entrada B que aparece en la figura 5.1. Los parámetros de entrada restantes tomarán los siguientes valores: $mnr = 2$ y $mnc = 3$. La primera pareja en procesarse es la correspondiente a las filas r_1 y r_2. La primera acción que tiene lugar es la **creación del patrón**. Para ello, se lleva a cabo la operación *AND* ($\wedge$) entre las dos filas, generándose como resultado una

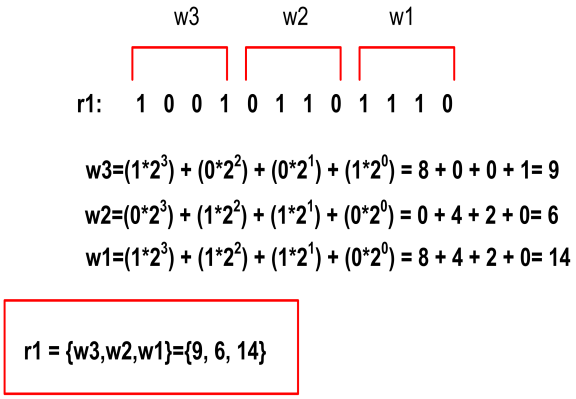

Figura 5.4: En esta figura se muestra el proceso de traducción de una secuencia de bits correspondiente a una fila de una matriz binaria en un conjunto de números enteros.

combinación concreta de bits (que se traduce en un subconjunto determinado de columnas de la matriz original) y un bicluster inicial:

$\longrightarrow$ **Pareja** (r_1, r_2)

Patrón:

$\rho_{12} = r_1 : \{9, 6, 14\} \wedge r_2 : \{15, 1, 6\} = \{9, 0, 6\} \rightarrow \{100100000110\}$

$Bic_{12} = \{\{r_1, r_2\}, \{\rho_{12}\}\} = \{\{r_1, r_2\}, \{c_1, c_4, c_{10}, c_{11}\}\}$

A continuación, y sólo en el caso en el que el patrón (ρ_{12}) no haya sido procesado con anterioridad y que el número de columnas que lo forman sea igual o mayor que el valor establecido por el parámetro de entrada *mnc* (fijado a 3), se comprueba si las filas restantes son compatibles o no con el patrón. Esta **verificación de la compatibilidad** se lleva a cabo aplicando de nuevo el operador *AND* ($\wedge$) a la fila analizada y al patrón en proceso, ρ_{12}. Si dicha operación *AND* da como resultado una combinación de bits exactamente igual que el patrón, significará que dicha fila puede ser añadida al bicluster en construcción. En caso contrario, implicaría la existencia, en la fila, de al menos un bit igual a 0 en la misma posición que ocupa un bit igual a 1 en el patrón, por lo que no se llevará a cabo ninguna acción y se procesará la fila siguiente:

Resto de filas:

$r_3 : \{15, 14, 6\} \wedge \rho_{12} : \{9, 0, 6\} = \{9,0,6\} \rightarrow$ Se añade la fila r_3

$r_4 : \{15, 1, 6\} \wedge \rho_{12} : \{9, 0, 6\} = \{9,0,6\} \rightarrow$ Se añade la fila r_4

$r_5 : \{12, 14, 6\} \wedge \rho_{12} : \{9, 0, 6\} = \{8,0,6\} \rightarrow$ NO compatible

Así, se obtiene un bicluster al final del proceso que será considerado válido si su número de filas es igual o mayor que el valor del parámetro *mnr* (fijado a 2). En esta ocasión, se genera el siguiente bicluster válido:

Bicluster final: $Bic_{12} = \{\{r_1, r_2, r_3, r_4\}, \{\rho_{12}\}\} = \{\{r_1, r_2, r_3, r_4\}, \{c_1, c_4, c_{10}, c_{11}\}\}$

A continuación, mostramos el procesamiento del resto de parejas del ejemplo:

$\longrightarrow$ **Pareja** (r_1, r_3)
Patrón:
$\rho_{13} = r_1 : \{9, 6, 14\} \wedge r_3 : \{15, 14, 6\} = \{9, 6, 6\} \rightarrow \{100101100110\}$
$Bic_{13} = \{\{r_1, r_3\}, \{\rho_{13}\}\} = \{\{r_1, r_3\}, \{c_1, c_4, c_6, c_7, c_{10}, c_{11}\}\}$
Resto de filas:
$r_4 : \{15, 1, 6\} \wedge \rho_{13} : \{9, 6, 6\} = \{9,0,6\} \rightarrow$ NO compatible
$r_5 : \{12, 14, 6\} \wedge \rho_{13} : \{9, 6, 6\} = \{8,6,6\} \rightarrow$ NO compatible
Bicluster final: $Bic_{13} = \{\{r_1, r_3\}, \{\rho_{13}\}\} = \{\{r_1, r_3\}, \{c_1, c_4, c_6, c_7, c_{10}, c_{11}\}\}$

$\longrightarrow$ **Pareja** (r_1, r_4)
Patrón:
$\rho_{14} = r_1 : \{9, 6, 14\} \wedge r_4 : \{15, 1, 6\} = \{9, 0, 6\} \rightarrow \{100101100110\}$ PATRÓN YA GENERADO

$\longrightarrow$ **Pareja** (r_1, r_5)
Patrón:
$\rho_{15} = r_1 : \{9, 6, 14\} \wedge r_5 : \{12, 14, 6\} = \{8, 6, 6\} \rightarrow \{100001100110\}$
Bicluster final: $Bic_{15} = \{\{r_1, r_5\}, \{\rho_{15}\}\} = \{\{r_1, r_5\}, \{c_1, c_6, c_7, c_{10}, c_{11}\}\}$

$\longrightarrow$ **Pareja** (r_2, r_3)
Patrón:
$\rho_{23} = r_2 : \{15, 1, 6\} \wedge r_3 : \{15, 14, 6\} = \{15, 0, 6\} \rightarrow \{111100000110\}$
$Bic_{23} = \{\{r_2, r_3\}, \{\rho_{23}\}\} = \{\{r_2, r_3\}, \{c_1, c_2, c_3, c_4, c_{10}, c_{11}\}\}$
Resto de filas:
$r_4 : \{15, 1, 6\} \wedge \rho_{23} : \{15, 0, 6\} = \{15,0,6\} \rightarrow$ Se añade la fila r_4
$r_5 : \{12, 14, 6\} \wedge \rho_{23} : \{15, 0, 6\} = \{12,0,6\} \rightarrow$ NO compatible
Bicluster final: $Bic_{23} = \{\{r_2, r_3\}, \{\rho_{23}\}\} = \{\{r_2, r_3, r_4\}, \{c_1, c_2, c_3, c_4, c_{10}, c_{11}\}\}$

$\longrightarrow$ **Pareja** (r_2, r_4)

Patrón:
$\rho_{24} = r_2 : \{15, 1, 6\} \wedge r_4 : \{15, 1, 6\} = \{15, 1, 6\} \rightarrow \{111100010110\}$
$Bic_{24} = \{\{r_2, r_4\}, \{\rho_{24}\}\} = \{\{r_2, r_4\}, \{c_1, c_2, c_3, c_4, c_8, c_{10}, c_{11}\}\}$
Resto de filas:
$r_5 : \{12, 14, 6\} \wedge \rho_{25} : \{15, 1, 6\} = \{12,0,6\} \rightarrow$ NO compatible
Bicluster final: $Bic_{24} = \{\{r_2, r_4\}, \{\rho_{24}\}\} = \{\{r_2, r_4\}, \{c_1, c_2, c_3, c_4, c_8, c_{10}, c_{11}\}\}$

$\longrightarrow$ **Pareja** (r_2, r_5)
Patrón:
$\rho_{25} = r_2 : \{15, 1, 6\} \wedge r_5 : \{12, 14, 6\} = \{12, 0, 6\} \rightarrow \{110000000110\}$
Bicluster final: $Bic_{25} = \{\{r_2, r_5\}, \{\rho_{25}\}\} = \{\{r_2, r_5\}, \{c_1, c_2, c_{10}, c_{11}\}\}$

$\longrightarrow$ **Pareja** (r_3, r_4)
Patrón:
$\rho_{34} = r_3 : \{15, 14, 6\} \wedge r_4 : \{15, 1, 6\} = \{15, 0, 6\} \rightarrow \{111100000110\}$
PATRÓN YA GENERADO

$\longrightarrow$ **Pareja** (r_3, r_5)
Patrón:
$\rho_{35} = r_3 : \{15, 14, 6\} \wedge r_5 : \{12, 14, 6\} = \{12, 14, 6\} \rightarrow \{110011100110\}$
Bicluster final: $Bic_{35} = \{\{r_3, r_5\}, \{\rho_{35}\}\} = \{\{r_3, r_5\}, \{c_1, c_2, c_5, c_6, c_7, c_{10}, c_{11}\}\}$

$\longrightarrow$ **Pareja** (r_4, r_5)
Patrón:
$\rho_{45} = r_4 : \{15, 1, 6\} \wedge r_5 : \{12, 14, 6\} = \{12, 0, 6\} \rightarrow \{110000000110\}$
PATRÓN YA GENERADO

Los siete biclusters que se han obtenido a partir de este sencillo ejemplo están representados en la figura 5.5.

5.4. Formalización y Algoritmo

A continuación, presentamos el algoritmo *BiBit* formalizado.
Los parámetros de entrada necesarios son los siguientes:

- Base de datos binaria, B.
- Mínimo número de filas que puede tener un bicluster final, *mnr*.
- Mínimo número de columnas que puede tener un bicluster final, *mnc*.

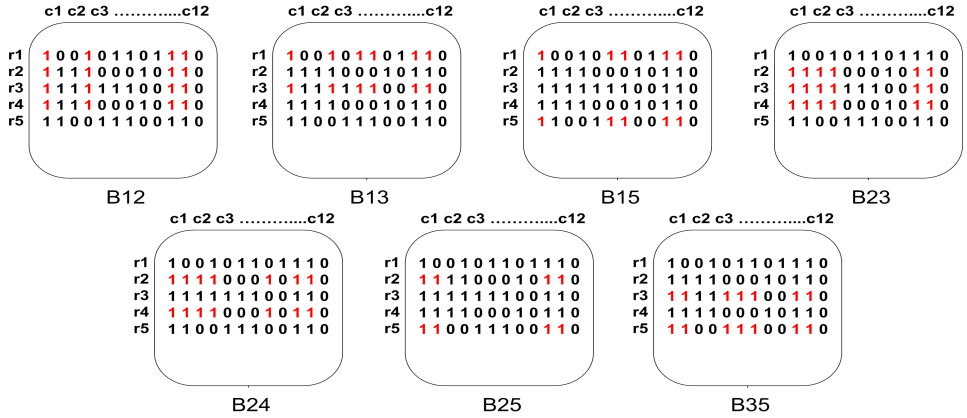

Figura 5.5: Los 7 biclusters que se generan en la fase de búsqueda. La matriz de entrada es la que aparece en el ejemplo de la figura 5.1

Como se puede observar en el pseudocódigo de la siguiente página, el algoritmo está formado por un bucle principal (línea 1) que recorre todas las posibles parejas de fila sin repetición. Por cada pareja de filas, r_i y r_j, se llevan a cabo dos acciones principales: creación del patrón y el bicluster inicial, y adición de filas hasta formar el bicluster final. La primera de estas acciones comienza en la línea 2, en la que un patrón se define a partir de la operación AND entre una pareja de filas. Este patrón no es más que un subconjunto de columnas determinado por aquellos bits iguales a 1 al final de dicha operación booleana. Una vez creado el patrón, ρ_{ij}, se han de llevar a cabo una serie de verificaciones (línea 3): si el mismo patrón ya ha sido procesado con anterioridad o el número de columnas que lo forman es menor que el valor del parámetro de entrada, *mnc*, entonces no será procesado. Una vez superadas estas dos verificaciones, se crea el bicluster inicial (línea 4) y comienza el segundo bucle (línea 5) cuyo objetivo es aumentar la dimensión de las filas del mencionado bicluster inicial. En este bucle se recorren el resto de filas, r_q, comprobándose si cada una de ellas es compatible con el patrón que está siendo procesado, ρ_{ij}. En la línea 6 aparece el método utilizado para llevar a cabo esta comprobación: si la operación AND entre la fila en proceso, r_q, y el patrón, ρ_{ij}, tiene como resultado una combinación de bits idéntica al patrón, entonces la fila podrá ser añadida al bicluster (línea 7). Una vez finalizado el segundo bucle, se comprueba (línea 10) si el bicluster final tiene un número de filas igual o mayor que el indicado por el parámetro de entrada *mnr*. En caso afirmativo, se habrá generado un nuevo *bit–pattern*, es decir, un bicluster máximo formado por unos.

Este algoritmo fue implementado utilizando el lenguaje de programación

Algorithm 1 Bit-patterns Biclustering Algorithm

Require: B: Encoded binary matrix
mnr: Minimum number of rows allowed
mnc: Minimum number of columns allowed
Ensure: X: List of final biclusters
1: **for** every rows pair (r_i, r_j) **do**
2: $\rho_{ij} = r_i \wedge r_j$
3: **if** ρ_{ij} is new and $|\rho_{ij}| \geq mnc$ **then**
4: $Bic_{ij} = \{I, J\} = \{\{r_i, r_j\}, \{\rho_{ij}\}\}$
5: **for** every remainder row r_q, with $q > j$ **do**
6: **if** $r_q \wedge \rho_{ij} = \rho_{ij}$ **then**
7: Add r_q to I
8: **end if**
9: **end for**
10: **if** $|I| \geq mnr$ **then**
11: Add Bic_{ij} to X
12: **end if**
13: **end if**
14: **end for**

Java y la plataforma *Eclipse* y recibe como entrada una base de datos en formato *ARFF*. Además, se tuvieron en cuenta algunas consideraciones relativas a la implementación que ayudaron a aumentar la eficiencia del algoritmo. Por ejemplo, no se procesan aquellas filas en las que el número de bits iguales a 1 es inferior al parámetro de entrada *mnc*. También son exentas de procesarse aquellas filas que son exactamente iguales a otras ya procesadas. Finalmente, se introdujo una mejora para facilitar el hecho de contar el número de bits iguales a 1 de cada patrón. Es decir, al trabajar con la versión entera de la base de datos binaria, en ciertos momentos (línea 3), necesitamos conocer el número de unos que tiene un cierto patrón. Para evitar transformar cada patrón de su representación entera a su representación binaria y, posteriormente, recorrer sus bits para contar aquellos con valor 1, esta información se tiene previamente calculada y almacenada. Para ello, y previo al comienzo del bucle principal, se cuentan el número de unos de todas las palabras de w bits, siendo w el tamaño prefijado para dividir las secuencias de bits durante la fase de codificación (ver apartado 5.3.2). Esta información es almacenada en un array de una sola dimensión en el que cada índice del mismo se corresponde con un posible valor entero generado por una palabra de w bits. Así, para contar el número de unos de un patrón formado, por ejemplo, por 3 números enteros, sólo hay que acceder 3 veces a dicho array e ir sumando su contenido.
Finalmente, indicar que éste es un algoritmo de búsqueda exhaustiva de complejidad: $O(N^2M')$, siendo N el número de filas y M' el número de columnas

tras la fase de codificación. A pesar de ser una complejidad cuadrática, la acción que más se repite durante el algoritmo, es decir, la operación AND entre dos secuencias de bits, se ejecuta a nivel de bits. Así pues, al ser una de las operaciones más básicas que un procesador actual es capaz de ejecutar, hace que la búsqueda de biclusters *bit–patterns* sea extremadamente rápida.

5.5. Conclusiones

En este capítulo se ha presentado una metodología, denominada *BiBit*, para la extracción de biclusters a partir de bases de datos binarias. Los biclusters objetivo de la metodología, llamados *bit–patterns*, son submatrices máximas con todos sus elementos iguales a 1. El algoritmo *BiBit* desarrolla una búsqueda selectiva basada en un procesamiento extremadamente rápido de secuencias de bits. Este hecho hace que se pueda obtener un número razonable de biclusters finales en muy poco tiempo. Además, sus sencillos parámetros de entrada hacen que sea una herramienta muy fácil de utilizar. Todas estas características vienen a mejorar el escaso número de algoritmos de biclustering especialmente diseñados para bases de datos binarias que existen en la actualidad. En especial, uno de los objetivos de esta tesis es mejorar los resultados del algoritmo *Bimax*, uno de los más referenciados. Así, los capítulos dedicados a la experimentación y aplicación de nuestra propuesta, irán enfocados también a la tarea de establecer comparaciones con *Bimax* y demostrar que podemos obtener iguales o mejores resultados que este famoso algoritmo, pero de una manera mucho más rápida.

En el siguiente capítulo, el algoritmo *BiBit* será analizado con el objetivo de comprender mejor cómo se comporta ante distintos escenarios. Durante este análisis, los resultados obtenidos serán comparados con el algoritmo *Bimax* (Prelic *et al.*, 2006).

Capítulo 6

Análisis del algoritmo *BiBit*

Antes pensábamos que nuestro futuro estaba en las estrellas. Ahora sabemos que está en nuestros genes
James Watson.

El objetivo principal de esta parte de la investigación consiste en analizar el comportamiento del algoritmo *BiBit* bajo diversas circunstancias. Para ello, se han diseñado bases de datos sintéticas a partir de las cuales analizar el efecto que tienen los parámetros de entrada en los resultados finales, establecer comparaciones con el algoritmo *Bimax* en términos de número de resultados generados, tiempo de ejecución y calidad de los resultados finales y finalmente verificar la escalabilidad de nuestra propuesta con respecto al tamaño de las bases de datos de entrada. Todos los experimentos se han llevado a cabo en una estación de trabajo HP Z600 workstation, con 12GB de memoria y procesador 2.40GHZ Intel Xeon Quad-Core.

6.1. Test de rendimiento

En este primer test se analizará el comportamiento de *BiBit* con diferentes valores de los parámetros de entrada, además de comparar el rendimiento de los algoritmos *BiBit* y *Bimax*. Finalmente, se pondrá a prueba la escalabilidad del algoritmo *BiBit*. A continuación, se describen las bases de datos utilizadas al igual que el algoritmo diseñado para generarlas.

6.1.1. Bases de datos sintéticas

El algoritmo *BiBit* ha sido diseñado para no verse influenciado por la forma o la densidad de unos de la base de datos de entrada. Por ello, se han utilizado como bases de datos sintéticas matrices binarias cuadradas

Algorithm 2 Matrix generation Algorithm

Require: N: Matrix size
P: Matrix density percentage
Ensure: B: A binary matrix of size NxN and with a density of 1's of $P\,\%$
1: Calculate the number of 1's for every row, $D = \frac{N*P}{100}$
2: **for** every row r_i **do**
3: $U = 0$, Number of 1's of the row
4: **for** every element of the row, r_{ij} **do**
5: r_{ij} = random(0,1)
6: **if** r_{ij}=1 **then**
7: $U = U + 1$
8: **end if**
9: **end for**
10: **while** $U > D$ **do**
11: Select randomly element equal to 1 and set them to 0 until $U = D$
12: **end while**
13: **while** $U < D$ **do**
14: Select randomly element equal to 0 and set them to 1 until $U = D$
15: **end while**
16: **end for**

de diferentes tamaños y densidades. En concreto, 20 diferentes tamaños de matrices , en el rango de 50×50 a 1000×1000, fueron utilizados con un incremento de 50 en ambas dimensiones. Para cada tamaño, se generaron 10 matrices con densidades de unos que variaron del 10 % al 100 %, con un incremento del 10 % en cada caso. En total, 200 matrices sintéticas.

De aquí en adelante se utilizará la siguiente notación para describir las características de cada matriz: $N \times M\_P\,\%$, siendo N el número de filas, M el número de columnas y P el porcentaje de unos que contiene la matriz.

Se ha diseñado un sencillo algoritmo (ver Algoritmo 2) para conseguir que la distribución de elementos iguales a 1 dentro de las matrices sintéticas sea lo más uniforme posible. Tomemos, como ejemplo de aplicación del mismo, una matriz de 50×50 que ha de ser generada con una densidad de unos del 10 %. La primera acción que se lleva a cabo (línea 1) es calcular el número de unos, D, que tendría que tener cada fila de dicha matriz para conseguir la densidad deseada. Así, para este caso concreto, tendríamos que: $D = \frac{50*10}{100} = 5$. A continuación, se procesa cada fila r_i, asignándose de manera aleatoria un valor de 0 ó 1 a cada uno de sus elementos, r_{ij} (líneas 2 a 5). Además, por cada fila, se cuenta el número de unos que han sido asignados, U (líneas 6-7). Al finalizar el recorrido de una fila, se comprueba si dicha cifra, U, coincide con el número de unos esperado, D. Si $U > D$, comienza un proceso iterativo en el que se seleccionan aleatoriamente elementos iguales a 1 y se cambia su valor a 0 hasta que $U = D$ (líneas 10-12). En caso contrario, si $U < D$,

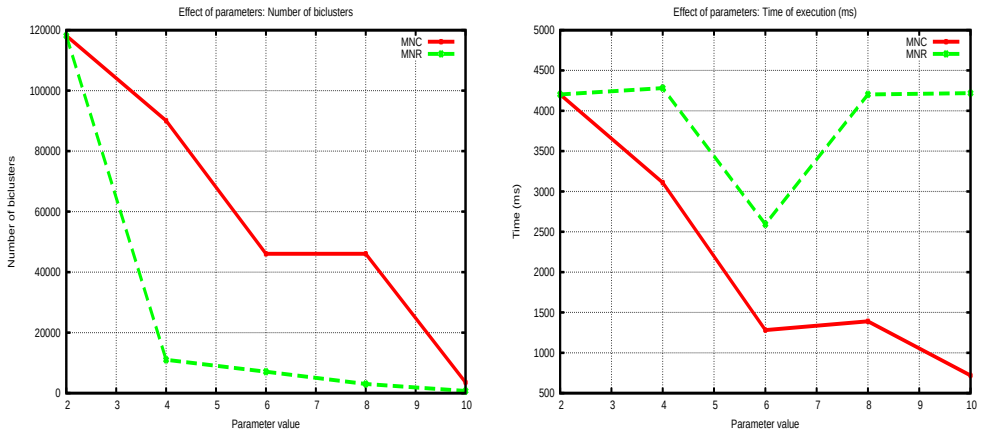

Figura 6.1: Comportamiento del algoritmo *BiBit* en función de los parámetros de entrada: *mnc* y *mnr*. El objetivo de esta figura es mostrar gráficamente el efecto que distintos valores de dichos parámetros provoca sobre el número de biclusters generados (gráfica de la izquierda) y el tiempo de ejecución (gráfica de la derecha). Ambos experimentos se llevaron a cabo con una matriz binaria $500 \times 500\_10\,\%$

son los elementos iguales a 0 los que son seleccionados aleatoriamente y ven cambiados su valor a 1 hasta que $U = D$. En conclusión, la uniformidad en la distribución de unos en las matrices sintéticas generadas se consigue aplicando dos veces la aleatoriedad.

6.1.2. Análisis del efecto de los parámetros de entrada: mnr y mnc

Los parámetros de entrada del algoritmo *BiBit*, *mnc* y *mnr*, son utilizados para delimitar los tamaños mínimos de los biclusters finales, tanto de las columnas como de las filas, respectivamente. Son parámetros muy sencillos de usar cuyo fin no es más que el de modificar el espacio de búsqueda, reduciendo o aumentado el número de biclusters finales que se pueden obtener. Para analizar el efecto de estos parámetros sobre el algoritmo *BiBit*, se utilizará una matriz binaria de entrada con las siguientes características: $500 \times 500\_10\,\%$, y varias combinaciones de los valores de los parámetros de entrada, a saber: En un caso, el parámetro *mnr* tendrá un valor fijo de 2 mientras que el parámetro *mnc* variará del valor 2 al 10 en el siguiente rango: $mnc = \{2, 4, 6, 8, 10\}$. En otro caso, será el parámetro *mnc* el que tenga el valor constante de 2 y *mnr* será el que varíe: $mnr = \{2, 4, 6, 8, 10\}$.

Tanto el número final de biclusters generados como el tiempo de ejecución empleado en milisegundos serán objeto de estudio en este análisis. Los resultados recogidos al respecto están reflejados en la figura 6.1. La gráfica de la izquierda representa los resultados en función del número de biclusters finales generados. En general, se puede observar que el número de resultados decrece a la vez que aumentan las restricciones impuestas por los parámetros. Sin embargo, esta tendencia es más marcada en el caso del parámetro *mnr*. Todos los biclusters, inicialmente, se crean a partir de una pareja de filas. Es por ello que, tal y como se muestra en la gráfica, el máximo número de resultados se alcance con $mnr = 2$. Sin embargo, el hecho de añadir más filas al bicluster inicial pasa, como se ha visto en secciones anteriores (ver Sec. 5.3), por superar una verificación de compatibilidad entre filas. Este hecho implica que sea más costoso encontrar biclusters cuando el parámetro *mnr* aumenta de valor. Así, la manera más efectiva de reducir el número de resultados es aumentar el valor del mínimo número de filas permitidas en los biclusters finales. Con respecto al tiempo de ejecución (ver gráfica de la derecha en figura 6.1), el parámetro más relevante es el relacionado con el mínimo número de columnas permitidas (parámetro *mnc*). La explicación es la siguiente: este parámetro toma efecto justo antes de la parte del algoritmo que consume más tiempo y recursos, es decir, buscar nuevas filas para incorporarlas a un bicluster. De esta forma, la vía más rápida para reducir el tiempo de ejecución del algoritmo *BiBit* es aumentar el valor del parámetro *mnc*.

En resumen, esta experimentación ha permitido verificar los distintos efectos que los parámetros de entrada tienen sobre el número de resultados y el tiempo de ejecución. Además, se han podido deducir las acciones a llevar a cabo para influir, de la manera más eficiente, en el comportamiento del algoritmo *BiBit*.

6.1.3. Comparativa de rendimiento: BiBit vs Bimax

El objetivo de esta sección es establecer una comparativa entre los algoritmos *BiBit* y *Bimax* para poder analizar sus diferencias de comportamiento al procesar las mismas bases de datos. Este análisis se ve facilitado por el hecho de que las dos propuestas reciben los mismos parámetros de entrada: números mínimos de filas y columnas.

Así, se aplicaron los dos algoritmos a las 200 bases de datos sintéticas creadas al efecto (ver Sec. 6.1.1). Tanto *BiBit* como *Bimax* han sido implementados en Java para la realización de las pruebas y, además, en los dos casos los parámetros que limitan el número de filas y el número de columnas

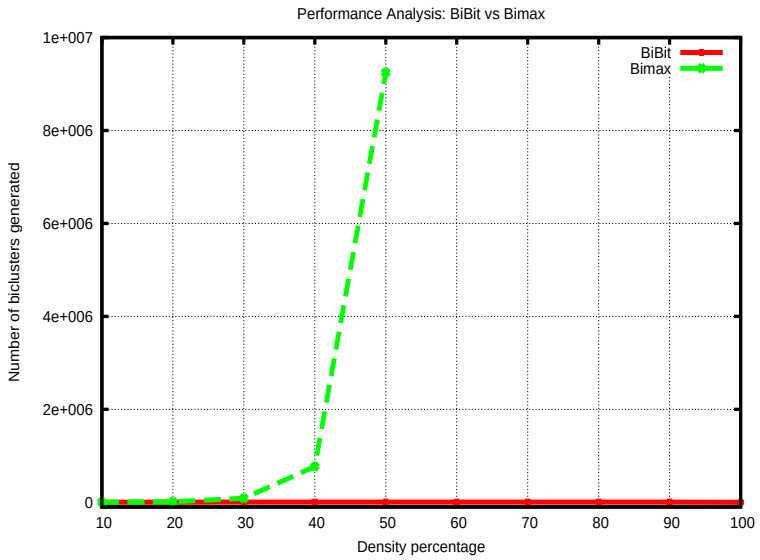

Figura 6.2: Comparativa entre *BiBit* y *Bimax*: Número de biclusters finales generados.

son fijados al valor 2, el menos restrictivo posible. Finalmente, indicar que solo se ha tenido en cuenta el tiempo de generación de los biclusters en la toma de tiempos.

El algoritmo *BiBit* tardó aproximadamente 19 minutos y 89 segundos en procesar las 200 bases de datos. Sin embargo, *Bimax* sólo fue capaz de procesar con éxito 12 de ellas. Los motivos que llevaron a *Bimax* a no poder completar el experimento estuvieron relacionados con continuos problemas de falta de memoria o tiempos de ejecución tan largos que obligaban a abortar la ejecución. Para establecer la comparativa entre ambas propuestas, 5 de esas 12 bases de datos serán utilizadas, concretamente las correspondientes a $100 \times 100\_D\,\%$, con $D \in \{10, 20, 30, 40, 50\}$, y los siguientes aspectos analizados: número de biclusters generados, tiempo de ejecución total y tiempo necesario para el procesamiento de un bicluster. Todas las mediciones obtenidas pueden ser consultadas en los apéndices de este documento (ver apéndice A). Con respecto al número final de biclusters obtenidos, los datos recogidos del experimento están representados en la figura 6.2. El menor número de biclusters que obtuvo *BiBit* fue de 947, siendo 4950 el mayor valor alcanzado en esta medición. Como se puede comprobar, existe una gran diferencia con el número de resultados que *Bimax* fue capaz de generar: entre 989 y 9246446 biclusters. Una importante diferencia es también la que reflejan los datos obtenidos al medir el tiempo de ejecución de ambos algoritmos (ver

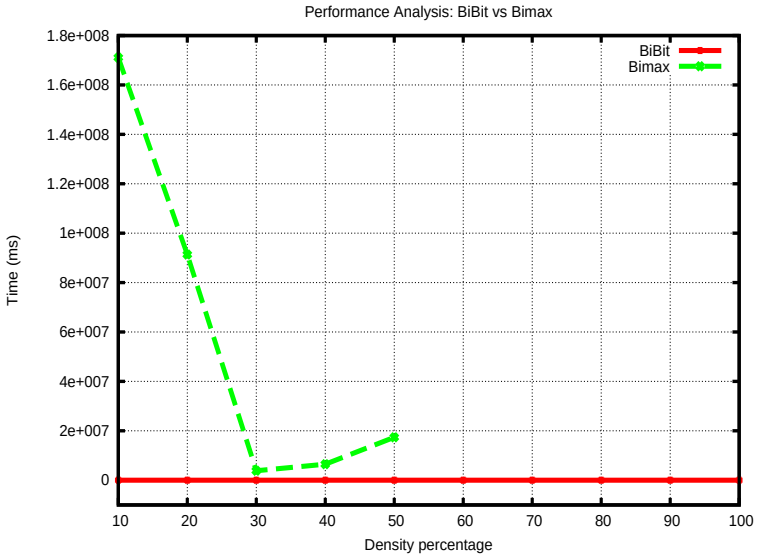

Figura 6.3: Comparativa entre *BiBit* y *Bimax*: Tiempo de ejecución en milisegundos.

figura 6.3). *BiBit* necesitó tiempos de ejecución que oscilaron entre los 47 y 63 milisegundos. Sin embargo, *Bimax* registró 3847704 y 171051625 milisegundos como menor y mayor tiempo de ejecución, respectivamente. En primera instancia, esta diferencia de tiempos se podría deber a que *Bimax* genera muchos más biclusters que *BiBit*. No obstante, podemos rebatir esta hipótesis utilizando para ello el siguiente ejemplo. Fijémonos en la base de datos $100 \times 100\_10\,\%$, para la cual los dos algoritmos generan un número de biclusters muy parecido (947 *BiBit* y 989 *Bimax*). Sin embargo, esta es la base de datos para la que *BiBit* necesitó menos tiempo y *Bimax* alcanzó su tiempo de ejecución más elevado: 47 milisegundos y 47.5 horas (171051625 milisegundos), respectivamente. Este hecho se puede explicar a partir de la necesidad de un número de recursos muy elevado por parte de *Bimax* para este caso concreto. Es decir, las técnicas de divide y vencerás dividen un problema en distintos subproblemas para poder abordarlos de manera más sencilla. En algunas ocasiones, como en el caso de *Bimax*, el número y tamaño de estas subdivisiones es dependiente de los datos de entrada y la eficiencia del algoritmo puede verse negativamente afectada si la carga de los subproblemas no está equilibrada. *Bimax* utiliza las columnas que contienen el valor 1 de una fila determinada como plantilla para establecer la división de la matriz en distintas submatrices. Para la base de datos en cuestión, $100 \times 100\_10\,\%$, el hecho de contar con un número de valores

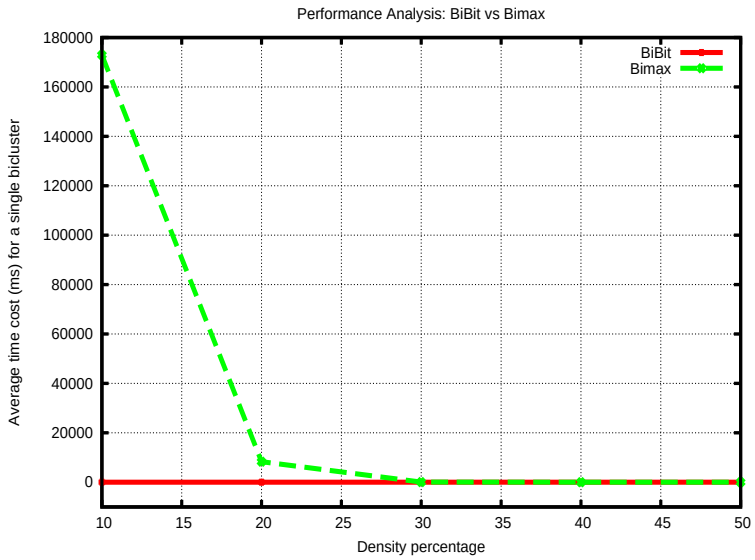

Figura 6.4: Comparativa entre *BiBit* y *Bimax*: Tiempo promedio por bicluster.

iguales a 1 tan reducido puede haber llevado a una división no balanceada del problema. Este hecho, unido a la forma cuadrada de la matriz, puede llevar a *Bimax* a comportarse con una complejidad de $O(n^3\beta)$, siendo β el número de todos los posibles biclusters máximos que existen en la matriz de entrada (vea material suplementario de (Prelic *et al.*, 2006)). Como muestra la figura 6.3, el tiempo de ejecución de *Bimax* decrece a medida que la densidad aumenta hasta el 30 % y aumenta a continuación debido al gran número de biclusters que se obtienen con densidades más elevadas (vea figura 6.2). Para reforzar la gran diferencia de rendimiento que existe entre los dos algoritmos, se aportan a continuación las mediciones realizadas sobre el tiempo en milisegundos necesario para generar un bicluster. Esta medida, $\frac{Time}{|Biclusters|}$, pretende establecer una comparativa de rendimiento más justa, sin tener en cuenta el número de resultados que cada algoritmo es capaz de obtener. Los datos obtenidos fueron los siguientes (ver figura 6.4): *BiBit* requiere como poco 0.01 ms/bicluster y como mucho 0.04 ms/bicluster, mientras que *Bimax* está entre los 1.88 milisegundos y 172.9 segundos por bicluster.

Como podemos observar, *BiBit* supera en gran medida al rendimiento de *Bimax* en cuanto a tiempo de ejecución y velocidad de generación de biclusters. A partir de este punto, es necesario demostrar también que se obtienen los mismos o mejores resultados en cuanto a calidad de los mismos

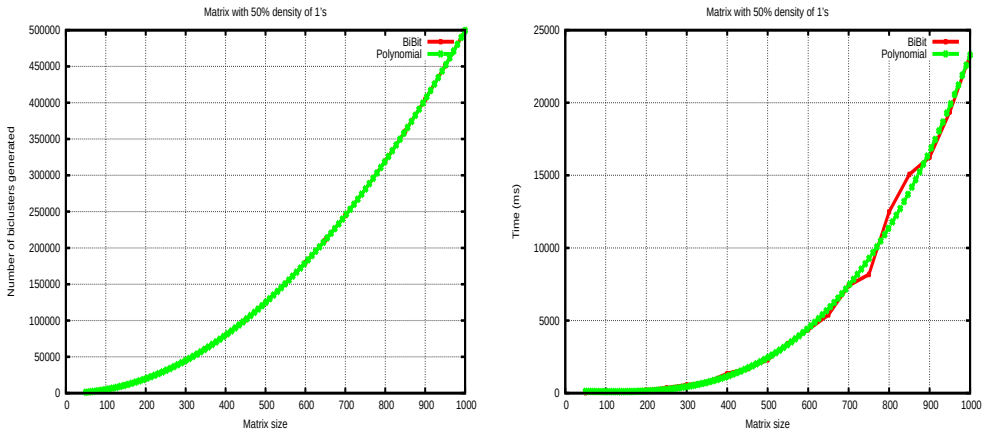

Figura 6.5: Impacto del tamaño sobre, de izquierda a derecha, (a) el número de biclusters generados, y (b) el tiempo de ejecución, para matrices con una densidad igual al 50 %.

se refiere. Este análisis será posteriormente desarrollado en la sección 6.2.

6.1.4. Escalabilidad del algoritmo BiBit

Los datos presentados anteriormente se centran en una base de datos de tamaño concreto y nos muestran el comportamiento de los algoritmos a medida que la densidad de dicha base de datos varía. El objetivo de este experimento consiste en comprobar cómo se comporta el algoritmo *BiBit* cuando se va aumentando el tamaño de la base de datos de entrada. Es decir, en qué proporción el tamaño de la base de datos influye en el número de biclusters obtenidos y en el tiempo de ejecución. Para ello, se ha escogido una densidad concreta, 50 %, y se ha ido modificando el tamaño de las bases de datos. Los resultados obtenidos están resumidos en las gráficas de la figura 6.5. Tanto en la gráfica de la izquierda (número de biclusters generados) como en la de la derecha (tiempo de ejecución en milisegundos) se ha incluido una función polinomial de tercer orden (en verde). Los resultados muestran cómo la evolución tanto del número de biclusters como del tiempo de ejecución es casi exacta a la que presenta la función polinomial. Este hecho demuestra, dentro del ámbito del experimento realizado, que el algoritmo *BiBit* es escalable, es decir, tiene un rendimiento óptimo a pesar del incremento del tamaño de la base de datos de entrada.

6.2. Match Score test

El fin de este experimento consiste en medir la precisión de *BiBit* y *Bimax* a la hora de localizar biclusters concretos. Para ello, se contará con la medida *match score*, comúnmente utilizada para evaluar el rendimiento de técnicas de biclustering. Los datos de entrada consisten en bases de datos sintéticas en las que ciertos biclusters han sido distribuidos de manera deliberada. Los algoritmos de biclustering que participan en el experimento han de localizar esos biclusters "ocultos" y la medida match score cuantifica la similitud existente entre los resultados obtenidos por los algoritmos y los biclusters objetivo. Se han llevado a cabo dos experimentos diferentes. En primer lugar, los algoritmos analizados tendrán que buscar biclusters con diferentes grados de solape en las bases de datos sintéticas. El segundo test tiene como función medir la precisión de *BiBit* y *Bimax* cuando se utilizan bases de datos de diferentes densidades. A continuación, se define de manera más concreta la medida match score.

6.2.1. Definición del Match Score

Esta medida ha sido diseñada para comparar 2 conjuntos de biclusters entre sí. En primer lugar, es necesario conocer cómo se mide la similitud de una pareja de biclusters:
Definición: Sean $G1, G2 \subseteq \{1, ..., n\}$ dos conjuntos de genes. El match score de G1 y G2 se obtiene a partir de la siguiente función, la cual caracteriza la correspondencia entre dos conjuntos de genes:

$$S_G(G_1, G_2) = \frac{\mid G_1 \cap G_2 \mid}{\mid G_1 \cup G_2 \mid} \tag{6.1}$$

Este match score, el cual se asemeja al coeficiente de Jaccard (Halkidi y et al., 2001), es simétrico, es decir, $S_G(G_1, G_2) = S_G(G_2, G_1)$, y sus posibles valores oscilan entre el 0 (los dos conjuntos son disjuntos) y el 1 (los dos conjuntos son iguales). De la misma manera se puede definir una medida match score para las condiciones experimentales, S_C. Sin embargo, sólo una dimensión, la de los genes, se ha tenido en cuenta a la hora de desarrollar esta experimentación.

En función de la anterior ecuación, el match score puede ser adaptado a dos conjuntos de biclusters:
Definición: Sean M_1,M_2 dos conjuntos de biclusters. El match score de M_1 con respecto a M_2 se define a partir de la función:

$$S(M_1, M_2) = \frac{\sum_{(G_1, C_1 \in M_1)} max_{(G_2, C_2 \in M_2)} S_G(G_2, G_1)}{|M_1|} \tag{6.2}$$

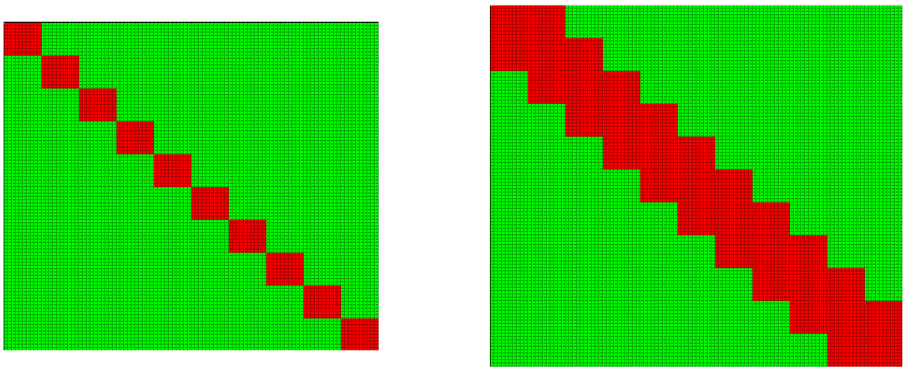

Figura 6.6: La matriz de la izquierda es la base de datos inicial utilizada en este experimento, con un grado de solape igual a 0. La matriz de la derecha tiene un grado de solape 10, lo que también implica que tiene 10 filas y 10 columnas más que la anterior. De ahí su diferencia de tamaño. En rojo aparecen destacados los biclusters objetivo.

Esta medida refleja el promedio de match scores máximos de todos los biclusters del conjunto M_1 con respecto a los biclusters de M_2. En esta ocasión, el match score no es simétrico y su significado cambia si intercambiamos de posición los dos conjuntos analizados. Así, se define el conjunto M_{opt} como los biclusters objetivo, es decir, aquellos biclusters que han sido incluidos de forma deliberada en las bases de datos sintéticas, y M como el conjunto de biclusters resultado de aplicar un algoritmo. De esta forma, el match score que se conoce como *bicluster relevance*, $S_G(M, M_{opt})$, nos informa hasta qué punto los biclusters resultado representan a los biclusters objetivo con respecto a la dimensión de los genes. Por otro lado, la medida denominada *module recovery*, $S_G(M_{opt}, M)$, cuantifica la precisión con la que los biclusters objetivo han sido recuperados por los algoritmos de biclustering bajo análisis. Ambas medidas alcanzan el valor máximo de 1 cuando $M = M_{opt}$ y serán las utilizadas en los experimentos que se describen a continuación.

6.2.2. Primer test: biclusters con distintos grado de solape.

Este primer experimento está relacionado con una de los tests incluidos en el trabajo de Prelic et al. (Prelic *et al.*, 2006) y tiene como finalidad medir la precisión de *BiBit* y *Bimax* cuando biclusters con diferentes grados de solapamiento son objetivo de búsqueda en bases de datos sintéticas.

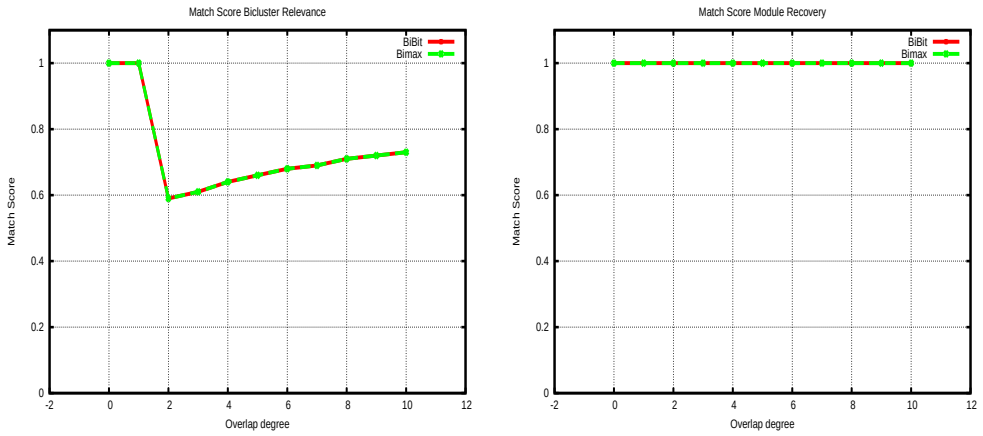

Figura 6.7: Los resultados del match score en la búsqueda de biclusters solapados se muestran en esta figura. Como se puede observar, tanto para la medida *bicluster relevance* (gráfica de la izquierda) como para la medida *module recovery* (gráfico de la derecha), *BiBit* y *Bimax* obtienen idénticos resultados.

El experimento comienza con una base de datos binario formada por una matriz de tamaño 100×100 que incluye 10 biclusters no solapados, formados por elementos iguales a 1, de tamaño 10x10 y distribuidos a lo largo de la diagonal principal. Para generar una nueva matriz, se aumenta el grado de solapamiento de los biclusters en una fila y en una columna. Así, al final son 11 las bases de datos que se utilizaron en este experimento, con grados de solape que van del 0 al 10. En la figura 6.6 están representadas como ejemplo la matriz de grado 0 de solape (en la izquierda) y la matriz de grado 10 de solape (a la derecha). Ésta última tiene un tamaño mayor ya que por cada grado más de solape se ha de añadir una fila y una columna nueva. En rojo aparecen destacados los biclusters objetivo.

Los algoritmos *BiBit* y *Bimax* fueron aplicados a las 11 bases de datos con los mismos parámetros de entrada. Los biclusters resultado y los objetivo fueron comparados mediante la medida match score. Los resultados del experimento aparecen resumidos en la figura 6.7. La gráfica de la izquierda, referente a la medida *bicluster relevance*, nos muestra que los dos algoritmos han obtenido resultados idénticos. En ambos casos, los mejores resultados se alcanzaron para grados de solape igual a 0 y 1. Con respecto a la medida *module recovery*, gráfica de la derecha, todos los valores recogidos son iguales a 1 tanto para *BiBit* como para *Bimax*. Ello implica que ambas propuestas fueron capaces de localizar a todos los biclusters objetivo. Es

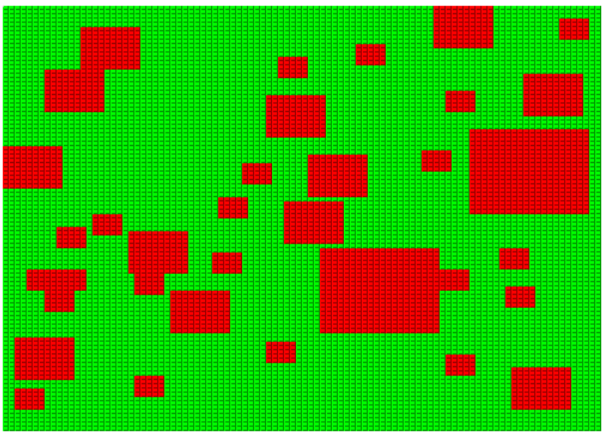

Figura 6.8: Ejemplo de una base de datos utilizada en el segundo experimento con match score. En este caso tenemos una matriz de tamaño 100x100 con un 25 % de densidad de elementos iguales a 1. Distintos biclusters no solapados, resaltados en rojo, han sido distribuidos de manera aleatoria por toda la matriz.

importante indicar que tanto *BiBit* como *Bimax* obtuvieron exactamente el mismo número de biclusters en todas las bases de datos utilizadas. En resumidas cuentas, en cuanto a la búsqueda de biclusters con distintos niveles de solape se refiere, tanto *BiBit* como *Bimax* presentan la misma precisión.

6.2.3. Segundo test: bases de datos con distintas densidades.

En el capítulo dedicado a la aplicación del algoritmo *BiBit* a un caso real, ver capítulo 8, se utiliza un novedoso método acumulativo de binarización para preprocesar la base de datos de entrada y producir un determinado número de bases de datos binarios con densidades crecientes que serán entrada del algoritmo. Así, lo que se pretende con el segundo experimento que se presenta a continuación es estudiar el comportamiento de la precisión de *BiBit* y *Bimax* cuando son aplicados a bases de datos con distintas densidades de elementos iguales a 1.

En esta ocasión, se han utilizado para las bases de datos binarios los siguientes tamaños: 50×50, 100×100 y 200×200. Para cada tamaño, se han generado 10 matrices sintéticas con una densidad de elementos iguales a 1 que varía del 5 al 50 %, con un incremento del 5 % en cada caso. En cada

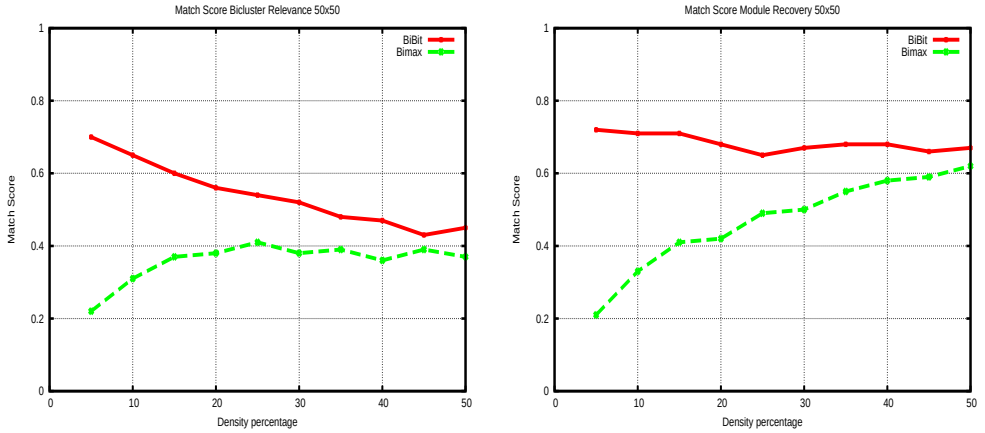

Figura 6.9: Resultados obtenidos para las medidas *bicluster relevance* (izquierda) y *module recovery* (derecha) en la base de datos de 50x50.

matriz, $N \times M\_P\,\%$, biclusters de diferentes tamaños: 5×5, 10×10 y 20×20, han sido aleatoriamente distribuidos sin solape hasta alcanzar aproximadamente una densidad de $P\,\%$ (véase un ejemplo en la figura 6.8). Además, por cada tipo de base de datos, $N \times M\_P\,\%$, se han creado 10 versiones diferentes y se ha calculado la media de las medidas recogidas para obtener los resultados finales, los cuales aparecen representados en las figuras 6.9 (para matrices de tamaño 50×50), 6.10 (para matrices de tamaño 100×100) y 6.11 (para matrices de tamaño 200×200). En todas las figuras, la medida *bicluster relevance* se muestra a la izquierda y *module recovery* a la derecha. Hay que indicar que el algoritmo *Bimax* no pudo procesar todas las matrices de tamaño 200×200 debido a continuos problemas de falta de memoria.

Como se puede apreciar en las gráficas, los datos obtenidos en las dos medidas para los distintos tamaños de bases de datos son muy similares. Sin embargo, hay una clara relación entre la densidad y los resultados del match score. El algoritmo *BiBit* localiza de manera correcta los biclusters objetivo de manera más frecuente si la base de datos es dispersa, es decir, si tiene una baja densidad de unos. *Bimax* presenta un comportamiento opuesto al ver aumentada su precisión con densidades más altas. No obstante, en todos los casos, los resultados obtenidos por *BiBit* son siempre mejores que los obtenidos por *Bimax*.

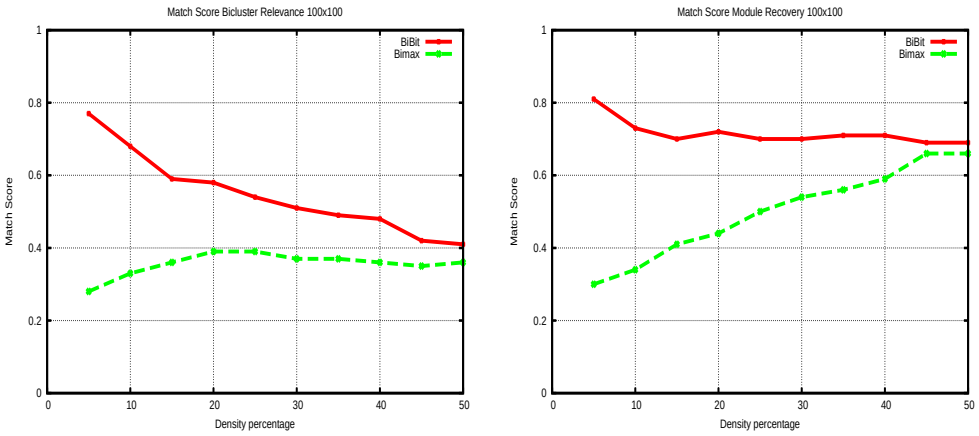

Figura 6.10: Resultados obtenidos para las medidas *bicluster relevance* (izquierda) y *module recovery* (derecha) en la base de datos de 100x100.

6.3. Conclusiones

Se han llevado a cabo dos tipos principales de experimentos: el primero de ellos intenta analizar el rendimiento, en cuanto a número de resultados y tiempo de ejecución, bajo diversas circunstancias (distintos valores de parámetros de entrada, distintos tamaños y distintas densidades de base de datos). El segundo experimento tiene como objetivo medir la precisión a la hora de buscar biclusters objetivo previamente distribuidos en las bases de datos de entrada. Ambos experimentos se han llevado a cabo con bases de datos sintéticas creadas al efecto. Además, en ambos casos se ha desarrollado una comparativa con el algoritmo de biclustering *Bimax*.

Con respecto al primer experimento se ha comprobado, en primer lugar, cuál es el efecto de los parámetros de entrada *mnr (minimum number of rows)* y *mnc (minimum number of columns)* sobre el número de resultados finales y el tiempo de ejecución. La forma más rápida de disminuir el número de resultados es aumentar la restricción impuesta por el parámetro *mnr*. El parámetro *mnc* es el que debemos ajustar para influir de manera más decisiva sobre el tiempo de ejecución (si aumentamos *mnc*, disminuirá el tiempo de ejecución). También hemos podido observar como el algoritmo *BiBit* presenta una muy buena escalabilidad con respecto al tamaño de la base de datos de entrada. Finalmente, se ha aplicado *BiBit* y *Bimax* sobre las 200 bases de datos sintéticas que se han creado para este experimento y se ha comparado su comportamiento. *BiBit* ha procesado todas ellas en apenas 20 minutos, alcanzando un número razonable de resultados finales y un mínimo

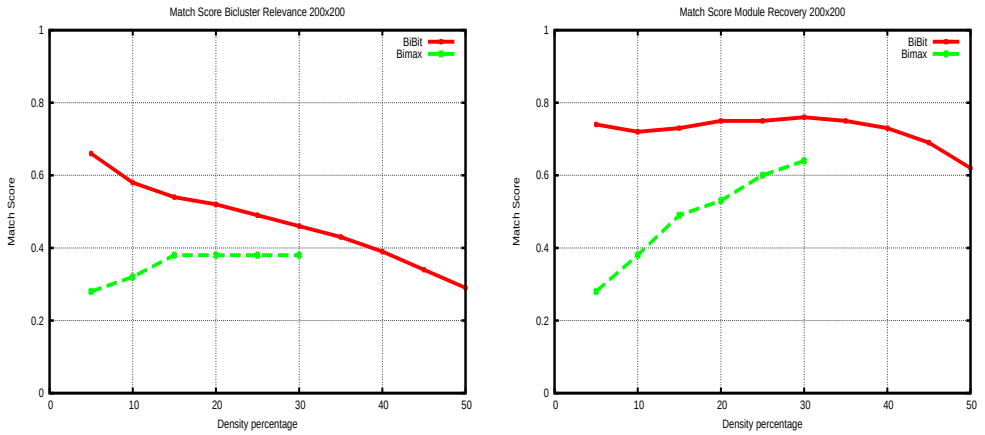

Figura 6.11: Resultados obtenidos para las medidas *bicluster relevance* (izquierda) y *module recovery* (derecha) en la base de datos de 200x200. En este caso, *Bimax* no pudo acabar el experimento debido a continuos problemas de falta de memoria.

tiempo de ejecución. *Bimax*, sin embargo, sólo ha podido concluir con éxito el procesamiento de 12 de las bases de datos y ha presentado tiempos de ejecución muy superiores. Para compensar el hecho de que *Bimax* obtiene un mayor número de resultados que *BiBit*, se ha establecido una medida, tiempo promedio por bicluster, que informa sobre el tiempo que tardan ambas propuestas en generar 1 sólo bicluster. Aquí también *BiBit* ha sido muy superior a *Bimax*.

Profundizando en el hecho de la gran diferencia en el número de biclusters obtenidos por los dos algoritmos bajo estudio, el objetivo del segundo experimento era comprobar si el hecho de que *BiBit* no generara ciertos biclusters influía negativamente en su precisión. Para ello, se diseñaron bases de datos sintéticas en las que se distribuyeron biclusters objetivo para que fuesen buscados tanto por *BiBit* como por *Bimax*. La medida match score, capaz de cuantificar la similitud existente entre un grupo de biclusters resultado y un grupo de biclusters objetivo, fue la elegida para medir esa precisión. En primer lugar, biclusters con distintos grados de solape fueron los objetivo de búsqueda. En esa ocasión, los dos algoritmos obtuvieron los mismos resultados y, curiosamente, el mismo número de biclusters. La segunda parte del experimento tenía como objetivo adicional estudiar la relación existente entre la precisión y la densidad de las bases de datos de entrada. En esta ocasión, *BiBit* y *Bimax* presentaron comportamiento antagónicos, mostrando mejores resultados cuando la densidad era baja y alta, respecti-

vamente. Sin embargo, en todas las pruebas la precisión de *BiBit* siempre fue superior a la de *Bimax*.

En conclusión, *BiBit* obtiene resultados mejores o iguales que *Bimax* pero empleando mucho menos tiempo y recursos y generando un número de biclusters más razonable.

Capítulo 7

Análisis de relevancia biológica: CarGene

Si quieres ganar un adepto para tu causa,
convéncelo primero de que eres su amigo sincero.
Abraham Lincoln.

La metodología propuesta en el capítulo 5 puede ser aplicada al análisis de datos de expresión genética, es decir, a los datos provenientes de experimentos con microarrays. Sin embargo, este análisis debe ser completado con la evaluación de los resultados obtenidos desde un punto de vista biológico. Esta verificación biológica engloba un amplio espectro de tareas que van desde el contraste de resultados con información biológica en repositorios conocidos hasta la verificación en laboratorio de las conclusiones obtenidas. El contraste de información puede considerarse una acción previa al laboratorio ya que ayuda a obtener conclusiones sobre la hipótesis de partida. Para llevar a cabo dicho contraste de información hemos diseñado una herramienta software denominada *CarGene*, la cual permite medir y comparar el enriquecimiento de varios conjuntos de genes de manera simultánea, ofreciendo un interfaz gráfico amigable y basándose para ello en la base de datos online KEGG (`http://www.genome.jp/kegg/pathway.html`).

Este trabajo ha sido publicado durante el año 2011 en la revista *International Journal of Data Mining and Bioinformatics*, con el título *"CarGene: Characterisation of Set of Genes based on Metabolic Pathway Analysis"*, en colaboración con Norberto Díaz Díaz, Isabel Nepomuceno Chamorro y Jesús Aguilar Ruiz.

7.1. Evaluación por contraste de información

Cuando se aplica un algoritmo a una base de datos de corte biológico, como por ejemplo una base de datos de expresión genética, es básico conocer si los resultados obtenidos están provistos o no de relevancia biológica. En este contexto, la gran cantidad de información sobre genes recopilada en repositorios públicos es un elemento clave para acometer dicho objetivo y por ello la integración de herramientas de búsquedas para evaluar automáticamente grupos de genes frente a estos repositorios son de gran relevancia. Existen diversas herramientas software que acometen dicho objetivo, como por ejemplo: FuncAssociate (Berriz *et al.*, 2009), GoSurfer (Zhong *et al.*, 2004), GO::TermFinder (Boyle *et al.*, 2004), GOstat (Falcon y Gentleman, 2007) y David (Huang *et al.*, 2007). Todas las herramientas mencionadas se basan principalmente en el repositorio Gene Ontology (GO) (Consortium, 2006). Otras herramientas se basan en el repositorio Kyoto Encyclopedia of Genes and Genomes, KEGG (Kanehisa *et al.*, 2008) tales como KCaM (Aoki *et al.*, 2004), PathwayVoyager (Altermann y Klaenhammer, 2005) y KGML-ED (Klukas y Schreiber, 2007) que ayudan en la visualización de rutas metabólicas de genomas concretos.

Una manera de evaluar resultados biológicos mediante el contraste de información consiste en calcular la relevancia biológica o enriquecimiento de anotaciones a través de repositorios de anotaciones de genes (Olson, 2006). Son numerosos los repositorios existentes que proporcionan dicha información y de entre ellos destacamos los siguientes:

1. Genbank: Contiene información de secuencia de nucleótidos de genes (`http://www.ncbi.nlm.nih.gov/Genbank/`).

2. UniGene: Organiza la información de Genbank en clusters de genes sin redundancia de información

3. Entrez Gene (LocusLink): Incluye información de secuencia, descripción y función de cada gen (`http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene`).

4. Ensembl Genome Browser: Contiene información actualizada de los genomas de organismos eucariotas (`http://www.ensembl.org/`).

5. Gene Ontology: Proporciona un vocabulario basado en ontologías que contiene información sobre los genes y sus productos (`http://geneontology.org/`).

6. KEGG: Proporciona información sobre rutas metabólicas (`http://www.genome.jp/kegg/pathway.html`). La base de datos KEGG contiene mapas de rutas metabólicas (metabolic pathways) que representan el conocimiento que actualmente se tiene sobre las redes de reacciones e interacciones moleculares sobre el metabolismo, el procesamiento de la información genética, el procesamiento de la información medioambiental, procesos celulares y enfermedades humanas.

Medir el enriquecimiento de los genes con respecto a una base de datos concreta implica calcular las coincidencias de esos genes con la información almacenada y, posteriormente, dar una medida de la relevancia de esas coincidencias utilizando para ello cálculos estadísticos. Un contraste de hipótesis o test de significatividad es una técnica de inferencia estadística utilizada para juzgar si una propiedad que se supone cumple una población es compatible con lo observado en una muestra de dicha población.

7.2. CarGene

CarGene es una herramienta software diseñada para extraer y procesar información obtenida a partir de la base de datos KEEG con el objetivo de evaluar, por contraste con información biológica, diversos grupos de genes obtenidos a partir de técnicas de inferencia. Con técnicas de inferencia nos referimos a técnicas computacionales que realicen análisis de microarray y que proporcionen listados, clusters, biclusters o redes de genes como resultado final. Una de las principales características de *CarGene* es la posibilidad de analizar y comparar entre sí, de manera simultánea, varios grupos de genes. Se ha puesto mucho esfuerzo en diseñar un interfaz amigable que proporcione información útil tanto de forma visual como textual. Además, incluye un navegador propio para explorar toda la información contenida en KEGG.

A continuación, la funcionalidad de *CarGene* será descrita en el siguiente orden: en primer lugar, extracción de información y consulta del repositorio KEGG; En segundo lugar, los test estadísticos implementados en la herramienta; En tercer lugar, cómo interpretar los resultados visuales así como usar el navegador; Finalmente, mencionaremos algunos detalles de implementación de la herramienta.

7.2.1. Extracción de información biológica

CarGene permite al usuario extraer información biológica de KEGG a través de dos tipos de consultas diferentes. El primer tipo de consulta se

realiza a partir de un organismo dado, proporcionando el listado de genes y rutas metabólicas o pathways asociados a dicho organismo. En el repositorio KEGG se puede encontrar la información genómica de una gran cantidad de organismos. Esta información se actualiza diariamente y se incrementa en unos diez organismos por mes. Utilizando este tipo de consultas, el usuario puede conocer el listado de genes involucrados en un cierto pathway o en qué pathways metabólicos participa un gen concreto.

El segundo tipo de consulta se basa en la verificación del grado de coherencia de un grupo de genes con respecto a la participación de los mismos en los pathways metabólicos almacenados en KEGG. Para utilizar esta funcionalidad, el usuario puede hacer referencia a un grupo de genes de dos maneras diferentes. La primera de ellas se basa en la utilización de un fichero para almacenar cada grupo de genes. Dicho fichero tiene el siguiente formato: el nombre tipo ORF de los genes organizados en una sola columna. La segunda de ellas permite almacenar varias grupos de genes en un solo fichero. El formato de este fichero es similar al anterior, solo que junto a cada gen añadimos el número del grupo al que pertenece. Con este último tipo de fichero, *CarGene* se adapta al formato de salida de la herramienta *Expander* (Shamir *et al.*, 2005), uno de los recursos bioinformáticos más utilizados para aplicar técnicas de biclustering a bases de datos de expresión genética. Finalmente, indicar que en ambos formatos de fichero no hay restricciones sobre el nombre o la extensión del mismo y que se aceptan los comentarios, precediendo con el símbolo % cada línea que quiera ser comentada.

El usuario puede seleccionar grupo de ficheros de manera individual o incluso, tiene la posibilidad de seleccionar una carpeta entera. Una vez seleccionados los grupos de genes a validar, CarGene consulta la base de datos KEGG, extrae información acerca de las rutas metabólicas en las que cada gen está involucrado y devuelve una medida estadística que nos informa acerca del nivel de enriquecimiento de cada grupo de genes con respecto a los datos consultados.

7.2.2. Medidas estadísticas

En cada ejecución, *CarGene* utiliza la información extraída sobre rutas metabólicas para medir el nivel de enriquecimiento de los grupos de genes bajo análisis. Para llevar a cabo esta medición, *CarGene* acepta o rechaza la siguiente hipótesis nula: "pertenecer a un grupo de genes Q es independiente de pertenecer a una ruta metabólica o pathway P". *CarGene* calcula la probabilidad de encontrar como mínimo y como máximo m genes de una consulta Q de longitud q en la ruta P si la hipótesis nula es cierta.

La probabilidad (p–value) se calcula por medio del Test de Fisher, (Rivals *et al.*, 2007), que emplea la probabilidad hipergeométrica exacta. Si el valor p–value asociado es igual o menor que un nivel de significatividad α, entonces la hipótesis nula se rechaza, es decir, el grupo de genes analizado Q está relacionado en cierto grado con la ruta metabólica P.

Escoger un valor α=0.05 implica contar con una probabilidad del 95 % de que la hipótesis nula sea cierta. Sin embargo, esta probabilidad decrece con el número de hipótesis a probar. Es decir, si se prueban dos hipótesis nulas independientes, entonces la probabilidad de encontrar que las dos son ciertas disminuye de la siguiente forma: $0.95 \times 0.95 = 0.90$. En el caso de *CarGene*, por cada consulta Q se ha de probar esta hipótesis para todas las rutas P detectadas, es decir, existe un problema de test de múltiples hipótesis. Para evitar el decrecimiento de esta probabilidad se utilizan correcciones para ajustar los p–values calculados.

La corrección de Bonferroni (Bland y Altman, 1995) es un procedimiento que ajusta el nivel de significatividad mediante la ecuación $\alpha' = \frac{\alpha}{k}$, siendo k el número de hipótesis. Tras esta corrección, una hipótesis nula será rechazada cuando el p–value sea menor a α'. Diversas herramientas utilizan como método de ajuste el proporcionado por Bonferroni o Holm. Sin embargo, si existen dependencias entre las hipótesis, en este caso concreto entre los pathways de KEGG, se recomienda utilizar otros procedimientos de ajuste (Berriz *et al.*, 2003; Gibbons y Roth, 2002). A diferencia de las correcciones de Bonferroni, en el que cada valor p–value se corrige de manera independiente, las correcciones proporcionadas por Westfall y Young (Westfall y Young, 1989) tienen en cuenta la dependencia entre los pathways por medio de simulaciones en las que se permutan los genes que pertenecen a cada pathway. Este procedimiento estima el p–value como resultado de 1000 simulaciones de consultas Q utilizando el procedimiento Monte Carlo (Kroese *et al.*, 2011). Tanto la corrección de Bonferroni como la de Westfall están implementadas en *CarGene*.

7.2.3. Representación de la información

Tanto la información extraída de KEGG como los resultados del análisis de enriquecimiento se presentan en *CarGene* de manera visual. El objetivo es proporcionar al usuario una forma gráfica de comparar el nivel de enriquecimiento de cada grupo de genes, de manera que se pueda rechazar fácilmente aquellos grupos no relevantes desde el punto de vista biológico. El usuario tienen la posibilidad de visualizar de manera individual, *single view mode*, o simultánea, *global view mode*, los resultados obtenidos para cada

grupo de genes. Además, la herramienta proporciona un informe completo de todas la ejecuciones con cada grupo de genes de manera textual.

Vista global

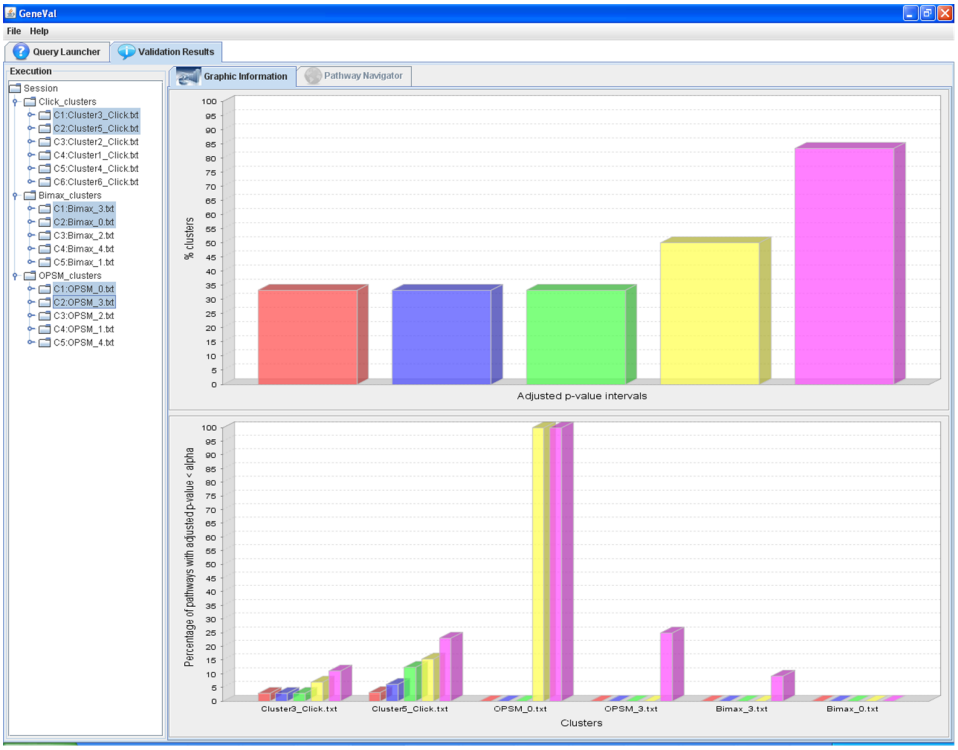

Figura 7.1: Vista global del análisis de enriquecimiento de pathways para los dos mejores grupos de genes proporcionados por tres métodos computacionales.

El objetivo de esta vista es mostrar en una única ventana todas las ejecuciones realizadas para diferentes grupos de genes, resultado de diversos métodos computacionales. En la figura 7.1 observamos tres partes diferenciadas. A la izquierda, el árbol de ejecución, en el que observamos los resultados de tres algoritmos distintos de clustering y biclustering sobre la misma base de datos (CLICK (Shamir y Sharan, 2000), BiMax (Prelic *et al.*, 2006) y OPSM (Ben-Dor *et al.*, 2002)). Cada técnica ha producido 6, 5, y 5 grupos de genes, respectivamente. El usuario puede elegir qué grupo de genes quiere analizar en detalle, teniendo en cuenta que éstos aparecen ordenados en la lista por orden de relevancia (en cuanto al análisis de enriquecimiento se

refiere). En el caso de la figura, el usuario ha seleccionado los dos primeros resultados de cada técnica. La vista global de la herramienta proporciona una comparación visual, organizada y rápida de los grupos de genes seleccionados. Para cada ejecución de un método computacional, se muestran los pathways con su p–value asociado de manera creciente, desde el pathway menos significativo hacia el pathway más significativo, estadísticamente hablando. Así por ejemplo, en la figura 7.1 y para el algoritmo CLICK, el grupo de genes denominado cluster 3 tiene un p–value menor que el cluster 5.

La gráfica superior derecha muestra el porcentaje de grupos de genes que contienen algún pathway con un p–value ajustado en los siguientes intervalos: [0,0.001),[0, 0.01),[0, 0.1), [0,1) y [0,5). Cada barra muestra el porcentaje de grupos cuyo p–value es menor al α del intervalo correspondiente. Así, por ejemplo, el 33 % de los grupos seleccionados por el usuario contienen pathways significativos para un (α <0.001). Las tres primeras barras muestran el mismo porcentaje, es decir, el 33 % de los grupos tienen al menos un pathway con un p–value ajustado menor que 0.1.

La gráfica inferior derecha muestra, por cada grupo de genes, el porcentaje de pathways con un p–value ajustado en alguno de los intervalos mencionados anteriormente. Por ello, cada grupo de genes tiene asociado cinco barras y los grupos que muestra son los seleccionados por el usuario en el árbol de ejecución.

CarGene proporciona más información sobre cada ejecución, además de la que se observa en la figura 7.1. Asociado con cada uno de los grupos de genes, tenemos un menú con opciones de visualización tales como la visualización en pantalla completa (porcentaje de genes por cada pathway, porcentaje de genes por cada pathway en grupos, porcentaje de pathways con un p–value ajustado $\leq \alpha$, porcentaje de conjuntos que contienen un p–value en el intervalo correspondiente) y finalmente un resumen textual.

Vista única

Esta vista proporciona dos tipos de información referente a un único grupo de genes. El primer tipo de información es referente a los p–value de cada pathway detectado en el grupo de genes. En este caso, se representa el porcentaje de pathways en el grupo de genes con un p–value ajustado en cada uno de los intervalos mencionados anteriormente. La herramienta también proporciona un resumen textual que incluye el nombre de los pathways, sus genes asociados, el p–value exacto y ajustado en cada caso, la fecha de ejecución, etc.

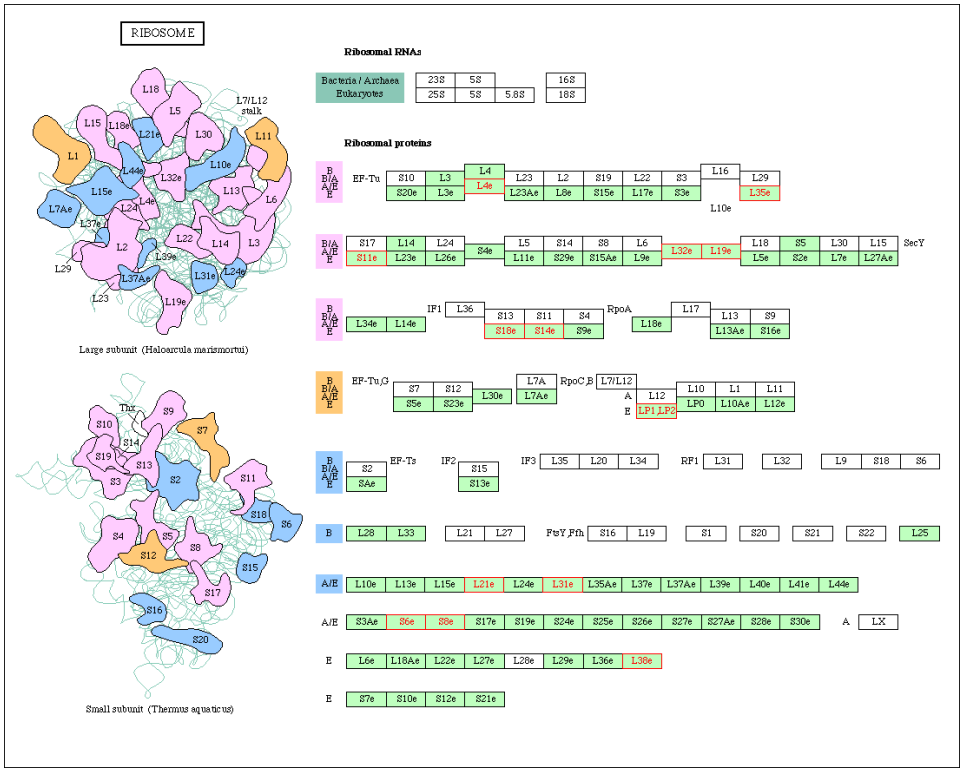

Figura 7.2: Representación de la proteína Ribosoma generada por KEGG y proporcionado por *CarGene*.

Con respecto al segundo tipo de información, *CarGene* proporciona un navegador para que el usuario pueda explorar la información relativa a cada uno de esos pathways que proporciona KEGG. Por ejemplo, la figura 7.2 representa el mapa asociado a la ruta metabólica que produce proteínas de Ribosoma. Este pathway aparece para el grupo de genes que forman el cluster 20 obtenido a partir del algoritmo CLICK. *CarGene* proporciona el mapa que representa al pathway seleccionado. En dicho mapa se marcan los genes pertenecientes al grupo de genes analizados en color rojo. Además, el usuario puede interactuar navegando por dicho mapa. Así, por ejemplo, el usuario puede acceder a la descripción detallada sobre el gen L21e (marcado en rojo en la Figura 7.2) como es el nombre del gen, motif, posición, secuencia, etc.

Información textual

CarGene ofrece además un informe en formato de texto con el análisis de enriquecimiento para los grupos de genes de entrada. De este modo, la herramienta obtiene para cada ejecución: a) el número de diferentes pathways detectados en todos los grupos de genes, de forma que se puede observar los diversos procesos biológicos detectados; b) porcentaje de grupos que presentan un p–value ajustado dado (esta información mide la calidad del grupo de genes en relación al proceso biológico).

7.2.4. Detalles de implementación

Java es el lenguaje de programación utilizado en el desarrollo de *CarGene*. Se ha llevado a cabo una implementación de tipo multihilo y para el intercambio de información estructurada XML se ha utilizado el Simple Object Access Protocol (SOAP). De esta forma, el usuario puede lanzar diversos análisis de resultados al mismo tiempo, pudiendo trabajar con la herramienta mientras ésta se encuentra ejecutando otras tareas. Por lo cual, es una herramienta especialmente pensada para plataformas multiprocesador. El núcleo estadístico está desarrollado en Java y utiliza JNI (Java Native Interface) para realizar llamadas a método nativos escritos en lenguaje C.

7.3. Conclusión

Este capítulo describe, en primer lugar, la necesidad de verificar la importancia biológica de los resultados obtenidos a partir de técnicas computacionales. Con respecto a aquellas técnicas que obtienen grupos de genes (clusters, biclusters, redes, etc.), la validación de resultados consiste en medir el grado de coherencia de los miembros de un mismo grupo llevando a cabo un análisis a partir de información biológica ya conocida. Esta información biológica aparece en numerosos e importantes repositorios online que permiten un acceso rápido a una gran cantidad de datos. Una de esas bases de datos es KEGG, la cual almacena información sobre las rutas metabólicas en las que los genes de los organismos se ven involucrados.

El objetivo de la herramienta *CarGene* es el de evaluar el enriquecimiento de un grupo de genes a partir de la información almacenada en KEGG. A parte de su interfaz gráfico amigable, uno de los mayores atractivos de *CarGene* es la posibilidad de comparar de manera simultánea la relevancia biológica de varios grupos de genes, tanto de manera visual como textual. Son varias las medidas estadísticas (test de Fisher, corrección de Bonferroni y corrección de Westfall-Young) las que se utilizan para medir la bondad de las

relaciones de los genes de un grupo concreto con los pathways almacenados en KEGG.

Parte IV

Resultados

Capítulo 8

Aplicación de *BiBit* a una base de datos real

Si buscas resultados distintos, no hagas siempre lo mismo.
Albert Einstein.

En el capítulo 6, el comportamiento del algoritmo *BiBit* fue estudiado en un entorno experimental basado en bases de datos sintéticas. El fin de este capítulo es mostrar los resultados obtenidos cuando una base de datos real es analizada con un objetivo concreto, para así verificar la utilidad de la técnica propuesta en esta tesis doctoral. Con este objetivo, se ha escogido una base de datos previamente estudiada en un trabajo de investigación de gran relevancia: la base de datos de expresión genética basada en tumores embrionarios del sistema nervioso central en la que se centra el trabajo de Pomeroy et al. (Pomeroy *et al.*, 2002). El objetivo de la experimentación que se presenta a continuación es obtener biclusters que aíslen las muestras de los distintos tipos de tumores que la base de datos incluye y así estudiar la relevancia biológica de los genes incluidos en dichos biclusters. Además, se lleva a cabo una comparativa con los resultados obtenidos por el algoritmo *Bimax* en el mismo estudio.

8.1. Descripción de la base de datos

La base de datos utilizada en esta experimentación se obtuvo tras un estudio clínico en el Children's Hospital de Boston en el que participaron niños y adultos jóvenes con distintos tipos de tumores relacionados con el sistema nervioso central: Medulloblastomas (tumor cerebral maligno que se origina en el cerebelo), AT/RT (Atypical teratoid rhabdoid tumor, tumor bastante

raro que se origina en el sistema nervioso central durante la infancia), renal/extrarenal rhabdoid tumors (neoplasma infantil maligno bastante raro que suele aparecer en los riñones y, en casos más raros aún (extrarenal), en el abdomen, pechos, pulmones, etc.) y supratentorial PNETs (otro infrecuente tumor infantil que afecta a la cresta neural). Todas las muestras tumorales fueron extraídas antes de iniciar algún tipo de tratamiento. Para recoger la expresión de un conjunto determinado de genes bajo esas muestras de tejido se utilizaron microarrays. Concretamente, biochips de la empresa Affymetrix (*www.affymetrix.com*) y de tipo HuGeneFL. Tras un determinado preprocesamiento, se obtuvieron tres bases de datos diferentes constituidas como matrices de números enteros: Dataset A (42 muestras que contienen: 10 medulloblastomas, 10 malignant gliomas, 5 AT/RT y 5 renal/extrarenal rhabdoid tumors, 8 supratentorial PNETs y 4 normal cerebella), Dataset B (34 muestras: 9 desmoplastic medulloblastoma y 25 classic medulloblastoma), y Dataset C (60 muestras referentes a 39 supervivientes de medulloblastoma y 21 tratamientos fallidos). Para nuestro experimento, se escogió una variante del Dataset A (la base de datos que más tipos diferentes de tumores incluye) denominada Dataset A1, la cual contiene 40 muestras en vez de 42. La diferencia radica en las muestras referentes a supratentorial PNETs, de las cuales solo se incluyen 6 muestras y no 8, excluyéndose 2 muestras de pineoblastomas que históricamente han sido clasificadas de manera errónea como PNETs. Así, la base de datos objeto de estudio es una matriz de números enteros con un tamaño de 7129 genes $\times$ 40 muestras tumorales (estas últimas aparecen descritas en el apéndice B, ver Tab. B.1).

8.2. Metodología

El objetivo de este estudio es localizar aquellos biclusters que sean capaces de aislar muestras de un tipo concreto de tumor y, posteriormente, analizar la relación existente entre los genes de dichos biclusters. Para llevar a cabo este objetivo se ha desarrollado una metodología, cuyo esquema aparece en la figura 8.1, que además incluye una novedosa técnica para obtener bases de datos binarios a partir de bases de datos previamente discretizadas. Esta metodología consta de tres fases principales: preprocesamiento, aplicación del algoritmo *BiBit* y, por último, análisis de los resultados. A continuación, se explican en detalle cada una de estas fases.

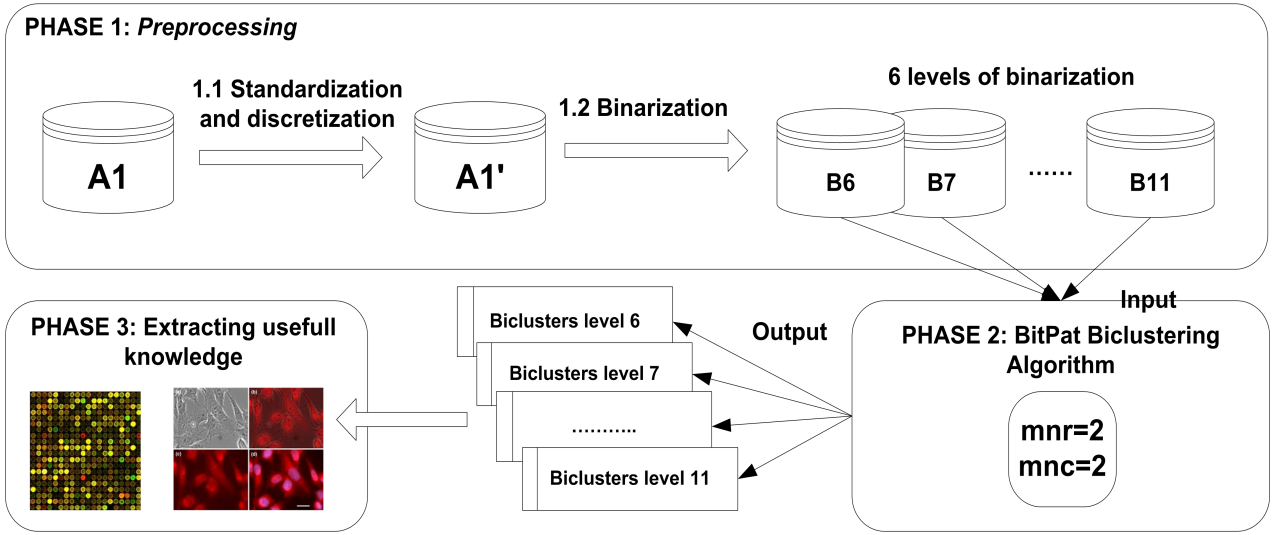

Figura 8.1: Esta imagen muestra un resumen esquemático del proceso de análisis de la base de datos de expresión genética. En la primera fase, la base de datos es preprocesada en 2 pasos. En el primer paso, los datos son estandarizados (media 0 y varianza 1) y a continuación discretizados en 12 diferentes niveles, con valores que oscilan entre el 0 y el 11. Cada uno de estos valores corresponde a un nivel de expresión genética. El segundo paso consiste en aplicar un novedoso método de binarización el cual generará un grupo de bases de datos binarios, teniendo cada una de ellas un nivel asignado i, con $6 \leq i \leq 11$ (solo se tienen en cuenta aquellos niveles en los que los genes están activos). En una base de datos de nivel i, los valores iguales a 1 serán aquellos que en la base de datos discretizada son mayores o iguales a i. El resto de valores serán iguales a 0. En la fase 2 de esta metodología, el algoritmo*BiBit* es aplicado a cada base de datos binarios, generando un conjunto de biclusters. En este experimento, el valor escogido para los parámetros de entrada de *BiBit*, *mnc* y *mnr*, es 2. Finalmente, en la fase 3, los resultados son analizados para extraer conocimiento útil.

8.2.1. Fase 1: Preprocesamiento

La fase de preprocesamiento consta a su vez de dos pasos fundamentales, durante los cuales la matriz original de números enteros se transforma en un conjunto de bases de datos binarios. El primer paso consiste en la estandarización y posterior discretización de la base de datos de entrada. En primer lugar, la base de datos es estandarizada, obteniéndose una matriz de números reales con media igual a 0 y varianza igual a 1. A continuación tiene lugar la discretización, cuyo objetivo es crear una base de datos en la que aparezcan representados por niveles los distintos valores de expresión de los genes. En concreto, el rango [-3.0, 3.0] fue dividido en 12 niveles diferentes, con una distancia de 0.5 entre un nivel y otro. Así, cada valor real es transformado en un valor discretizado d, con $d \in \{0, 1, 2, \ldots, 11\}$, según el nivel de expresión genética que represente. Los valores iguales o inferiores a -3.0 son considerados en el nivel 0 y los valores iguales o mayores a 3.0, en el nivel 11. Los niveles del 6 al 11 corresponden a genes expresados o activos y los niveles del 0 al 5 a genes no expresados o inactivos. En el estudio de Pomeroy et al. (Pomeroy *et al.*, 2002), el proceso de estandarización es llevado a cabo justo antes de aplicar la técnica de clustering conocida como *self-organizing maps* (Kohonen, 1997). Además, los mismos rangos de niveles de expresión fueron utilizados para representar gráficamente el comportamiento de los genes frente a los distintos tipos de tumores. En definitiva, como resultado de estas primeras acciones se obtiene una nueva base de datos, Dataset A1', que será entrada del siguiente paso del preprocesamiento.

Para completar el preprocesamiento, es necesario obtener bases de datos binarios que alimenten al algoritmo *BiBit*. Para desarrollar este proceso de binarización se ha diseñado un nuevo método el cual es capaz de obtener un conjunto de bases de datos binarios a partir de una base de datos discretizada. Como se mencionó en la sección 4.5, en las bases de datos de expresión genética los valores 1 y 0, bajo una cierta condición experimental c, implican que el gen en cuestión está o no expresado bajo dicha condición c, respectivamente. Son varios los métodos de binarización que se pueden aplicar a una base de datos de expresión genética. De todos ellos, uno de los más comunes es establecer un umbral de activación. Si un valor de la base de datos es igual o superior a este umbral, su valor binario correspondiente será un 1; 0 en caso contrario. Por ejemplo, en el trabajo en el que el algoritmo *Bimax* fue presentado, (Prelic *et al.*, 2006), se fijó el siguiente umbral de activación: $\underline{e} + \frac{(\bar{e}-\underline{e})}{2}$, con $\underline{e}$ y $\bar{e}$ como los valores mínimo y máximo de expresión genética en la base de datos, respectivamente. El nuevo método de binarización presentado en esta tesis doctoral obtiene una nueva base

Level	**Number of 1s**	**Density %**
11	6342	2.22
10	7855	2.75
9	10161	3.56
8	14444	5.1
7	24418	8.56
6	65594	23
5	272787	95.66
4	283838	99.63
3	284731	99.64
2	284983	99.93
1	285059	99.96
0	285160	100

Tabla 8.1: Por cada base de datos binarios generada, esta tabla almacena información acerca de su nivel de expresión, su número de elementos iguales a 1 y el porcentaje que dichos elementos representan con respecto al total (densidad).

de datos binarios por cada nivel de expresión mencionado anteriormente. Es decir, por cada valor discretizado d, con $d \in \{0, 1, 2, \ldots, 11\}$, una nueva matriz binaria B_d es generada, en la que se cumple que $B_d(x, y) = 1$ if $A1'(x, y) \geq d$; $B_d(x, y) = 0$ en otro caso. Como se puede observar, este método acumulativo proporciona más información en el sentido en que no solo la activación de los genes es tenida en cuenta. Además, el mínimo nivel de expresión genética en el que dicha activación tiene lugar es conocido y dicha información puede ser utilizada para obtener nuevas e interesantes conclusiones. Al final, se generaron 12 bases de datos binarios diferentes. Sin embargo, en este estudio solo se tuvo en cuenta a aquellas bases de datos binarios correspondientes a los niveles que van del 6 al 11, es decir, relativas a genes expresados.

La tabla 8.1 contiene información acerca de las 12 bases de datos binarios generadas. Debido al nuevo método acumulativo utilizado, el número de elementos iguales a 1 crecerá a medida que va disminuyendo el nivel de expresión y, por consiguiente, tendremos bases de datos con distintas densidades. Como se puede observar en la tabla, las bases de datos en las que se centra el estudio, niveles 6 al 11, tienen una densidad de elementos iguales a 1 muy pequeña comparada con la que presentan las bases de datos del otro rango de niveles de expresión (del 0 al 5). Según lo verificado durante el

análisis del algoritmo *BiBit* (ver sección 6.2.3), éste tiene una mayor precisión cuanto más dispersa es la base de datos, por lo que podemos esperar un comportamiento positivo de nuestra propuesta bajo esas seis bases de datos binarios.

8.2.2. Fase 2: Aplicación del algoritmo *BiBit*

El objetivo de esta fase consiste en obtener biclusters que cumplan con los siguientes dos requisitos: que incluyan el mayor número de muestras (columnas) y que además éstas sean del mismo tipo de tumor. De aquí en adelante, a este tipo de resultados se les denominará *tumor biclusters*. A las 6 bases de datos binarios obtenidas en la fase anterior, se han aplicado los algoritmos *BiBit* y *Bimax* y se ha establecido una comparativa de resultados.

Para el algoritmo *BiBit* se utilizaron los siguientes valores para los parámetros de entrada: mnr=2 y mnc=2. El mínimo número de columnas (muestras tumorales) fue fijado a 2 porque es el menor número de muestras referentes a un tipo concreto de tumor (AT/RT extra renal tumor) que se puede encontrar en la base de datos. El mínimo número de filas (genes) permitido en los biclusters fue fijado a 2 para poder extraer el máximo número de resultados posibles. Debido a la codificación a la que cada base de datos es sometida durante la ejecución de *BiBit*, la dimensión original de la matriz de entrada, 7129×40, es drásticamente reducida a 7129×3. Para el algoritmo *Bimax*, se utilizó el mismo valor referente al número mínimo de columnas: 2. Sin embargo, el mínimo número de filas tuvo que oscilar entre el valor 50 y el 150 para poder obtener algún resultado. Además, las bases de datos binarios de nivel 6 y 7 no pudieron ser procesadas debido a tiempos de ejecución excesivamente largos y continuos problemas de falta de memoria. La tabla 8.2 muestra información sobre el rendimiento que mostraron ambas propuestas durante su aplicación a las 6 bases de datos. Las mayores diferencias se encuentran en el número de biclusters generados y en el tiempo empleado para ello. Con respecto al número de biclusters, *BiBit* genera no más del 3 % de biclusters generados por *Bimax* (concretamente, el 2.93 % para el caso de la base de datos de nivel 11). A pesar de ello, el número de *tumor biclusters* obtenidos por *BiBit* es aproximadamente la mitad de aquellos obtenidos por *Bimax*. Además, mientras que *BiBit* sólo necesitó 2.5 minutos para procesar todas las matrices, *Bimax* requirió 14 horas para procesar sólo 4 de ellas.

8.2.3. Fase 3: Análisis de resultados

La última fase de la metodología consiste en analizar un determinado conjunto de *tumor biclusters* y obtener conocimiento útil analizando las re-

Level	BiBit Bic.	Bimax Bic.	BiBit T. Bic.	BiMax T. Bic.	BiBit Time(sg)	Bimax Time(sg)
11	12192	415709	278	636	1.19	575.1
10	16859	1966223	388	1213	1.52	47400.37
9	25682	1794909	438	1069	2.14	1677.35
8	49246	3600474	619	1305	4.29	939.21
7	135129	–	791	–	13.29	–
6	891373	–	1148	–	135.70	–

Tabla 8.2: Comparativa de rendimiento de *BiBit* y *Bimax* durante la ejecución de las 6 bases de datos binarios. La primera columna muestra el nivel de la base de datos binarios. Las dos siguientes columnas muestra el número de biclusters generados, pero la tercera y la cuarta únicamente se refieren a aquellos biclusters con todas sus muestras pertenecientes a un sólo tipo de tumor (tumor biclusters). Las últimas dos columnas presentan el tiempo de ejecución en segundos. No hay información disponible de *Bimax* para las bases de datos de nivel 7 y 6 debido a que no pudieron ser ejecutadas ya que se registraron tiempos de ejecución excesivamente largos y continuos problemas de falta de memoria.

laciones existentes entre los genes que los forman. Los criterios utilizados para seleccionar aquellos resultados susceptibles de ser analizados son los siguientes: se escogen aquellos *tumor biclusters* con el máximo porcentaje de muestras de un determinado tipo de tumor y con el mínimo número de genes. El sentido de esta última restricción radica en el hecho de buscar grupos de genes con el mayor grado de especialización posible. Consecuentemente, es importante encontrar biclusters que pertenezcan a los mayores niveles de expresión genética. Así, por cada tipo de tumor escogemos un *tumor bicluster* que lo represente. La tabla 8.3 recoge información sobre los *tumor biclusters* escogidos de entre los generados por *BiBit* y por *Bimax*, en particular: número de genes de los biclusters, porcentaje de muestras que contienen del tumor al que representan y nivel de la base de datos binarios de la que fueron extraídos. Hay que destacar el hecho de que, tanto en el caso de *BiBit* como en el de *Bimax*, los biclusters escogidos comparten un gran número de genes. Sin embargo, esos genes comunes son en su mayoría controles positivos de la base de datos (como era de esperar) o genes relacionados con proteínas ribosómicas, presentes de manera frecuente en los diferentes tumores. Por este motivo, aquellos genes comunes de cada bicluster (incluidos como mínimo en el 70 % del resto) son eliminados. Así, el valor que aparece en la tabla 8.3 hace referencia al número de genes tras esta labor de filtrado.

El algoritmo *BiBit* fue capaz de encontrar biclusters muy relacionados con cada tipo de tumor. En 4 de los 7 tipos de tumores diferentes se obtuvieron biclusters que contenían el 100 % de sus muestras. Además, la media del porcentaje de muestras incluidas es muy alto: 87 %. Merece la pena destacar el caso del *tumor bicluster* relacionado con el *Glioblastoma*, con 10 de sus 10 muestras incluidas en la base de datos. El *Glioblastoma* es el tumor que, junto con el *Medulloblastoma*, más muestras incluye. Otro importante detalle es el hecho de que todos los biclusters fueron extraídos de bases de datos con un alto nivel de expresión genética, la media es de 10, implicando que los genes involucrados en todos estos procesos carcinogénicos aparecen en general altamente expresados. Con respecto a los resultados obtenidos por *Bimax* hemos de indicar que son bastante similares. Los biclusters de *BiBit* contienen una media de 79 genes mientras que aquellos pertenecientes a *Bimax* tienen una media de 66.8. Los porcentajes de muestras de tumores incluidos en los *tumor biclusters* son iguales para las dos propuestas, excepto para el caso del *Medulloblastoma* (*BiBit* 80 % y *Bimax* 100 %) y el *Glioblastoma* (*BiBit* 100 % y *Bimax* 90 %). Finalmente, indicar que los genes de los biclusters de *BiBit* y *Bimax* referentes al mismo tipo de tumor coinciden en un porcentaje medio del 53.09 %.

Tumour	**BiBit Genes**	**Bimax Genes**	**BiBit %C.**	**Bimax %.**	**BiBit lev.**	**Bimax lev.**
Brain	101	86	80 %	80 %	10	9
Extra Renal	78	96	100 %	100 %	11	11
Renal	168	135	100 %	100 %	9	10
Glioblastoma	24	23	100 %	90 %	9	10
Medulloblastoma	87	68	80 %	100 %	8	10
Normal C.	31	36	100 %	100 %	11	11
PNET	64	24	50 %	50 %	9	11

Tabla 8.3: Características de *tumor biclusters* seleccionados para su estudio. La primera columna es el tipo de tumor que cada bicluster representa. La segunda y tercera columna hacen referencia al número de genes de los biclusters. La cuarta y quinta columna muestra el porcentaje de muestras del tipo concreto de tumor incluidas en cada bicluster. Finalmente, las dos últimas columnas contienen el nivel de la base de datos binarios de la que el bicluster fue extraido.

Tumor	**BiBit**	**Bimax**	**Common annotations**
Brain	90	65	60
Extra Renal	39	39	38
Renal	77	56	51
Glioblastoma	52	56	49
Medulloblastoma	57	47	43
Normal C.	7	2	0
PNET	61	50	38

Tabla 8.4: Número de anotaciones funcionales descubiertas en los tumor biclusters de BiBit (segunda columna) y Bimax (tercera columna). La última columna muestra el número de anotaciones comunes en cada caso.

De los genes incluidos en un bicluster se espera que estén involucrados en los mismos procesos biológicos, sirviendo dicho grado de relación como medida de calidad de los resultados obtenidos. Para verificar este hecho, se ha llevado a cabo un análisis del enriquecimiento de los genes incluidos en los *tumor biclusters*. En primer lugar, ha sido necesario utilizar una herramienta online, *Biomart Ensemble data integration system* (Smedley y et al., 2009), para convertir los identificadores de los genes incluidos en el chip *HuGeneFL* de *Affymetrix* en identificadores de tipo *Ensemble*. Posteriormente, esos nuevos identificadores de los genes fueron utilizados en la aplicación web *FuncAssociate* (Berriz *et al.*, 2009), a partir de la cual pudimos descubrir qué anotaciones funcionales de *Gene Ontology* (Consortium, 2006) aparecían enriquecidas en nuestros grupos de genes. *Gene Ontology* (GO, de aquí en adelante) es una base de datos online que proporciona un vocabulario estructurado el cual describe los genes, y sus productos, de cualquier organismo utilizando tres tipos diferentes de ontología: cellular component, molecular function y biological process. Medir el enriquecimiento de los genes con respecto a GO implica calcular las coincidencias de esos genes con los términos y categorías de GO y, posteriormente, dar una medida de la relevancia de esas coincidencias utilizando para ello cálculos estadísticos (p-value).

La tabla 8.4 muestra un resumen del número de anotaciones funcionales de GO encontrados en los *tumor biclusters* de *BiBit* y *Bimax*. Como se puede observar en la mayoría de casos, *BiBit* presenta más anotaciones funcionales que *Bimax* y además, se da la circunstancia de que esas anotaciones extra están generalmente relacionadas con el cancer. Por ejemplo, el bicluster obtenido por *BiBit* y relacionado con el *brain tumor* contiene anotaciones funcionales que no aparecen en el correspondiente *tumor bicluster* de *Bimax*

y que están relacionadas con procesos carcinogénicos, por ejemplo: replicación (spindle assembly, spindle organization, microtubule-based movement), inmunidad (cell killing, immune response, natural killer cell mediated immunity, lymphocyte mediated immunity, MHC protein binding, MHC protein binding, MHC class I protein complex, MHC class I protein complex, MHC class I protein binding) y transcripción (RNA binding, ribonucleoprotein complex, ribosomal small subunit biogenesis, etc.).

Con respecto a *BiBit*, la tabla 8.5 incluye, por cada uno de sus *tumor biclusters* representando a un tumor concreto, un ejemplo compuesto por tres procesos biológicos obtenidos durante el análisis de enriquecimiento de sus genes, junto con su p-value. Cada uno de los 7 biclusters contiene un alto grado de enriquecimiento y todos presentan anotaciones relacionadas con el cancer. A continuación se comentan algunos de estos resultados. En el trabajo de Pomeroy et al. (Pomeroy *et al.*, 2002) se incluye un estudio de la estructura intrínseca de los datos relacionados con el *medulloblastoma*, siendo una de sus conclusiones que los genes cuya correlación era más afín con este tipo de tumor eran principalmente codificadores de proteínas ribosómicas. En el bicluster obtenido por *BiBit* que representa al *medulloblastoma*, la mayoría de anotaciones están relacionadas con proteínas ribosómicas, como era de esperar. También cabe destacar el bicluster relacionado con el *glioblastoma*, en el cual encontramos anotaciones significativas relacionadas con células de tipo *natural killer* (células NK). Estas células son un tipo de linfocito perteneciente al sistema inmunitario cuya función está relacionada con la destrucción de células infectadas o cancerígenas. El *glioblastoma* es un tumor del cual se conoce que activa los receptores de estas células NK (Castriconi y et al., 2009). Además, la anotación *spindle assembly* detectada también en este bicluster está relacionada con el proceso de división celular inherente al cancer. Otro resultado del trabajo de Pomeroy et al. fue el de proporcionar una lista con los 10 genes que están más correlacionados con cada tipo de cancer. Como prueba de la utilidad de las técnicas de biclustering, nuestros resultados confirman esta clasificación para algunos de estos genes pero además, *BiBit* es capaz de detectar, en algunos casos, genes asociados con más de una clase. Por ejemplo, el gen D76435 es clasificado como *medulloblastoma*, sin embargo también está incluido en nuestro bicluster relacionado con *normal cerebellum*. Yokota, (Yokota y et al., 1996), probó que este gen en concreto aparece altamente expresado en el núcleo de neuronas del cerebelo pero también es detectado en los *medulloblastomas* (26 de los 29 casos estudiados), no apareciendo en ningún tipo de tumor adicional incluido en este estudio. El gen X86809 proporciona otro buen ejemplo ya que está asociado con *glioblastomas* malignos pero también aparece en nuestro

Tumour Type	**P-value**	**GO Attribute**
Brain	5.33E-14	MHC class I protein complex
	1.87E-08	cell killing
	5.10E-06	immune response
Extra Renal	7.94E-32	ribonucleoprotein complex
	1.06E-13	ribosomal small subunit biogenesis
	2.61E-06	antigen processing and presentation
Renal	1.04E-12	U4 snRNA binding
	6.50E-07	ribosomal large subunit biogenesis
	2.60E-05	microtubule-based movement
Glioblastoma	4.39E-18	natural killer cell mediated immunity
	5.18E-15	spindle assembly
	7.34E-15	MHC protein binding
Medulloblastoma	3.60E-13	ribosomal small subunit biogenesis
	2.04E-12	ribonucleoprotein complex biogenesis
	2.34E-06	ribosomal large subunit biogenesis
N. Cerebellum	3.30E-05	fructose-bisphosphate aldolase activity
	4.46E-05	homophilic cell adhesion
	4.68E-03	neuron projection morphogenesis
PNET	9.38E-14	leukocyte mediated cytotoxicity
	1.60E-09	large ribosomal subunit
	2.50E-05	negative regulation of RNA splicing

Tabla 8.5: Por cada tipo de tumor, esta table incluye algunas anotaciones de GO encontradas durante el proceso de análisis de enriquecimiento de los biclusters obtenidos por *BiBit*. La primera columna se refiere al bicluster relacionado con un tipo de tumor. La segunda columna es la medida estadística p-value y el atributo GO en cuestión aparece en la última columna

bicluster que representa al *normal cerebellum*. A pesar de ser un gen ubicuo, presenta altos niveles de expresión en los tejidos cerebrales, incluido el cerebelo (Estelles y et al., 1996). Finalmente, nuestros biclusters incluyen genes que no aparecen en el trabajo de Pomeroy et al. y que pueden ser considerados potencialmente interesantes para ser incluidos en algún estudio experimental. La lista de estos genes aparece en el apéndice B (ver tablas B.2 y B.3).

8.3. Conclusiones

Para evaluar la utilidad de nuestra propuesta se ha analizado una base de datos de expresión genética relacionada con tumores infantiles del sistema nervioso central. Previamente, estos datos habían sido estudiados en un prestigioso trabajo de investigación, (Pomeroy *et al.*, 2002). El objetivo era corroborar parte las conclusiones a las que se llegaron en dicho estudio y aportar nuevos conocimientos. Además, también se aplica el algoritmo *Bimax* sobre la misma información.

La metodología presentada incluye un nuevo método acumulativo de binarización que permite obtener no solo información acerca de que genes están expresados sino que además podemos conocer en que nivel de expresión tuvo lugar dicha activación. Los biclusters obtenidos por *BiBit* fueron capaces de clasificar los 7 tipos de tumores incluidos en la base de datos, con un 87 % medio de todas las muestras cubiertas. Además, los genes de estos biclusters contenían un alto grado de enriquecimiento, estando relacionados con un gran número de procesos biológicos asociados al cancer. La utilidad de las técnicas de biclustering fue demostrada mediante la detección de algunos genes que, corroborando la clasificación llevada a cabo por Pomeroy et al., además estaban presentes en otro tipo de tejidos. Finalmente, y con respecto a la comparativa con *Bimax*, indicar que *BiBit* fue capaz de obtener mejores resultados, en cuanto a anotaciones se refiere, necesitando para ello generar menos biclusters y consumir menos recursos.

Parte V

Conclusiones

Capítulo 9

Conclusiones y trabajos futuros

En mi final está mi comienzo.
Maria I de Escocia

El objetivo principal de esta tesis es presentar una alternativa simple y eficiente a la obtención de biclusters a partir de datos binarios. La codificación basada en los valores 1 y 0 ofrece una manera sencilla de almacenar las relaciones existentes entre un conjunto de objetos y sus posibles propiedades. Además de ocupar menos espacio, los datos binarios son, a priori, más sencillos de procesar. Sin embargo, en la actualidad, las propuestas existentes de biclustering especialmente diseñadas para este tipo de información son escasas y, o bien presentan modelos estadísticos complejos de utilizar o obtienen un número de resultados demasiado elevado, ofreciendo bajos rendimientos según qué tipo de base de datos se reciba como entrada.

9.1. Conclusiones

Nuestro algoritmo, *BiBit*, es una propuesta con las siguientes características principales:

- Es un algoritmo fácil de utilizar. Tiene un número muy limitado de parámetros, cuyo objetivo es simplemente reducir el espacio de búsqueda. Además, *BiBit* es determinista. Todo ello hace que cualquier biólogo o médico pueda hacer uso de esta técnica y conozca de forma clara qué se puede esperar de ella.
- Presenta una novedosa forma de trabajar con secuencias de bits. Esta técnica aprovecha las operaciones más primitivas que un procesador

actual puede llevar a cabo para procesar toda la información binaria de una base de datos. El resultado es un procesamiento muy rápido y eficiente.

- Genera un número coherente de resultados. Lejos de obtener todos los biclusters máximos existentes, como otras propuestas, el algoritmo *BiBit* obtiene un subconjunto de estos biclusters máximos. Así, es posible almacenar y pos-procesar los resultados finales, dando la posibilidad de analizar el conocimiento generado.

- Las pruebas realizadas con bases de datos sintéticas han demostrado que nuestra propuesta es escalable y que se adapta de manera eficiente al aumento del tamaño de la base de datos de entrada. Además, *BiBit* ofrece mejores resultados de precisión cuanto más dispersa es la base de datos, es decir, cuanto menor es la densidad de elementos iguales a 1. La forma de la base de datos tampoco afecta a nuestra técnica, pudiéndose aplicar a cualquier tipo de información biológica binaria, no sólo a bases de datos de expresión genética.

- El análisis de la base de datos real, basada en la expresión de genes en tumores cerebrales infantiles, ha corroborado la utilidad del algoritmo *BiBit*. Todos los tipos diferentes de tumores incluidos en la base de datos han sido aislados por los biclusters obtenidos. Además, los genes incluidos en los biclusters presentan un alto grado de enriquecimiento con respecto a funciones biológicas relacionados con procesos cancerígenos. Finalmente, hemos podido concluir que varios de los genes estudiados intervienen en más de un proceso diferente, pudiéndose verificar este hecho con la literatura existente.

Además de nuestra técnica de biclustering, durante el estudio del caso real, se ha propuesto una nueva metodología para binarizar bases de datos. Esta nueva técnica de preprocesamiento, basada en la división de los datos en capas o niveles, proporciona la posibilidad de obtener, a partir de una base de datos discretizada, N bases de datos binarios que, además de la información intrínseca a la base de datos, ofrecen nuevos datos interesantes. En el caso de la base de datos estudiada, podíamos discernir el nivel mínimo de expresión a partir del cual un gen estaba expresado. Así, pudimos descubrir que la mayoría de genes de los biclusters seleccionados se activaban en niveles de expresión muy altos.
Todas las pruebas realizadas, ya sea con datos artificiales o reales, han venido acompañadas de una comparativa con el referenciado algoritmo de biclustering *Bimax* (Prelic *et al.*, 2006). Los resultados demuestran que nuestra

propuesta es capaz de obtener resultados de una calidad igual o superior, pero en mucho menos tiempo y utilizando muchos menos recursos.
Finalmente, hemos presentado en esta tesis doctoral una nueva herramienta, *CarGene*, para validar, desde un punto de vista biológico, grupos de genes obtenidos a partir de distintas técnicas de computación. La validación efectuada se basa en medir la coherencia de un grupo de genes con respecto a su participación en las distintas rutas metabólicas almacenadas en la base de datos KEGG. Una de sus características más importantes consiste en la posibilidad de comparar, de manera simultánea y de forma gráfica y textual, los resultados de varios grupos de genes.

9.2. Trabajo futuro

Toda la investigación llevada a cabo no es más que el comienzo a partir del cual abordar objetivos más ambiciosos. Con respecto al trabajo futuro que se pretende llevar a cabo mostramos a continuación los puntos más importantes:

- Implementar un entorno gráfico online para facilitar el uso nuestro algoritmo *BiBit* por parte de la comunidad científica. Esta herramienta deberá proporcionar la posibilidad de almacenar los resultados, de elegir qué información es susceptible de ser generada y de incluso mostrar gráficamente la evolución de valores dentro de los biclusters.

- Mejorar nuestra técnica para permitir incluir cierto grado de error. Es decir, los biclusters obtenidos son constantes, es decir, están formados por elementos iguales a 1. Dando la posibilidad de incluir algún elemento igual a cero, ampliaremos el abanico de posibilidades y haremos más realista el análisis de la información. Dicho grado de error ha de ser parametrizado por el usuario.

- Utilizando la técnica de validación biológica presentada por Norberto Díaz (Diaz-Diaz y Aguilar-Ruiz, 2011), diseñar una metodología para validar biclusters provenientes de distintas técnicas y que para ello se tenga en cuenta no sólo una dimensión sino ambas. Se llevará a cabo un estudio comparativo entre varias técnicas de biclustering que incluirá, por supuesto, nuestro algoritmo.

- En contraste con el conocimiento que actualmente existe sobre los genes y su funcionalidad, la anotación de regiones reguladoras de genes

está actualmente en sus comienzos. La unión de sitios de transcripción (*Transcription Factors*, TF) a secuencias de ADN juega un papel muy importante en la regulación de la transcripción. Las secuencias a las cuales estos TFs se enlazan son muy cortas (entre 5 y 12 pares de bases) y muestran una variabilidad considerable. Así, podemos observar abundantes potenciales sitios de unión para TFs de los cuales sólo unos pocos son realmente útiles. Las secuencias reguladoras de la acción de los genes están compuestas normalmente por múltiples sitios de unión para múltiples factores de transcripción (TF). Debido a esta composición modular y su acción en cambios reguladores de tipo CIS, estas secuencias son llamadas *Cis-Regulatory Modules* (CRMs). La detección de CRMs identificados experimentalmente y la búsqueda de nuevos son tareas que actualmente tienen un gran atractivo para la comunidad científica. Hasta ahora, los métodos actuales se centraban en buscar en secuencias de ADN modelos de CRM creados a partir de muy diferentes técnicas. La idea es enfocar este problema de una forma diferente: ¿es posible que CRMs de la misma funcionalidad compartan sitios de transcripción también relacionados entre sí?, ¿mejoraría la búsqueda en secuencias de ADN el hecho de contar con una ventana más amplia de pares de bases?, ¿se podrían identificar CRMs nuevos e incluso predecir su funcionalidad? Parte del trabajo futuro se centrará en la búsqueda de respuestas a estas preguntas.

Parte VI

Apéndices

Apéndice A

Comparativa de rendimiento: BiBit vs Bimax

A.1. Algoritmo BiBit

A continuación se exponen cuatro tablas con los datos acerca de los biclusters generados, el tiempo de ejecución , el número de biclusters obtenidos en promedio por milisegundo y el tiempo en milisegundos empleado para generar 1 bicluster, respectivamente. Esta información corresponde a la ejecución de *BiBit* sobre 200 bases de datos binarias sintéticas de distintos tamaños y densidades de unos.

A.1.1. Número de biclusters generados

En la tabla A.1 podemos apreciar el número de biclusters que el algoritmo *BiBit* ha generado en la experimentación con bases de datos sintéticas. Para aquellas bases de datos con densidad 100 %, el número de biclusters obtenidos siempre será 1, ya que en estos casos toda la matriz representa a un solo bicluster. En el resto de casos, el número de biclusters aumenta con el tamaño y la densidad de las matrices. Es curioso observar que, dada una matriz concreta, se alcanza una densidad a partir de la cual siempre se va a obtener el mismo número de resultados.

A.1.2. Tiempo de ejecución en milisegundos.

La siguiente tabla, ver Tab. A.2, muestra el tiempo de ejecución en milisegundos del algoritmo *BiBit*. En esta medición de tiempos únicamente se ha tenido en cuenta el tiempo de obtención de los biclusters. Para un tamaño dado, en general, el tiempo de ejecución decrece a medida que la densidad

va aumentando. En el caso de un tamaño determinado de matriz, son muchos los casos en los que la densidad 10 % aparece como la más costosa en tiempo de procesar. Si sumamos todos los valores de la tabla tenemos que el algoritmo *BiBit* tardó unos 19.89 minutos en procesar las 200 bases de datos.

A.1.3. Velocidad de procesamiento de biclusters

Hemos recogido la misma información de dos maneras diferentes. En primer lugar, la tabla A.3 recoge el número de biclusters que se genera cada milisegundo durante el procesamiento de las 200 bases de datos sintéticas. La máxima velocidad fue alcanzada durante el procesamiento de la base de datos $200x200\_40\,\%$, la cual fue de 105.85 biclusters por milisegundo, es decir, 105850 Biclusters por segundo. Finalmente, la tabla A.4 muestra el tiempo en milisegundos que tardó *BiBit* en generar en promedio un sólo bicluster.

A.2. Algoritmo Bimax

En el caso del algoritmo *Bimax*, únicamente se pudieron procesar 12 bases de datos con éxito de las 200 debido a consumos excesivos de memoria o tiempos de ejecución extremadamente largos. Mostramos a continuación las mismas tablas que para el algoritmo *BiBit*.

A.2.1. Número de biclusters generados

Se puede observar en la tabla A.5 que el número de biclusters generados es mucho mayor en el caso del algoritmo *Bimax*. Concretamente, la máxima diferencia la encontramos para la base de datos $100x100\_50\,\%$, en la que el número de biclusters generados por *BiBit*, 4950, representa de manera aproximada el 0,05 % de aquellos obtenidos por *Bimax*, 9246446 biclusters.

A.2.2. Tiempo de ejecución en milisegundos.

Los tiempos de ejecución empleados por *Bimax*, ver tabla A.6, son muy altos en comparación con los que *BiBit* necesita. El ejemplo más claro se ve en la base de datos $100x100\_10\,\%$. Los dos algoritmos generan aproximadamente el mismo número de biclusters, 947 *BiBit* y 989 *Bimax*, pero utilizan para ello una cantidad de tiempo muy diferente: 47 milisegundos tarda *BiBit* frente a las 47.5 horas que necesita *Bimax* (171051625 milisegundos).

NxN vs P %	**10 %**	**20 %**	**30 %**	**40 %**	**50 %**	**60 %**	**70 %**	**80 %**	**90 %**	**100 %**
50x50	67	587	1112	1218	1225	1225	1225	1225	1221	1
100x100	947	4287	4939	4950	4950	4950	4950	4950	4950	1
150x150	3869	10951	11175	11175	11175	11175	11175	11175	11175	1
200x200	9829	19828	19900	19900	19900	19900	19900	19900	19900	1
250x250	19516	31105	31125	31125	31125	31125	31125	31125	31125	1
300x300	32997	44848	44850	44850	44850	44850	44850	44850	44850	1
350x350	50138	61073	61075	61075	61075	61075	61075	61075	61075	1
400x400	70294	79800	79800	79800	79800	79800	79800	79800	79800	1
450x450	92747	101025	101025	101025	101025	101025	101025	101025	101025	1
500x500	118171	124750	124750	124750	124750	124750	124750	124750	124750	1
550x550	146040	150975	150975	150975	150975	150975	150975	150975	150975	1
600x600	175919	179700	179700	179700	179700	179700	179700	179700	179700	1
650x650	208196	210925	210925	210925	210925	210925	210925	210925	210925	1
700x700	242557	244650	244650	244650	244650	244650	244650	244650	244650	1
750x750	279430	280875	280875	280875	280875	280875	280875	280875	280875	1
800x800	318547	319600	319600	319600	319600	319600	319600	319600	319600	1
850x850	360082	360825	360825	360825	360825	360825	360825	360825	360825	1
900x900	404011	404550	404550	404550	404550	404550	404550	404550	404550	1
950x950	450380	450775	450775	450775	450775	450775	450775	450775	450775	1
1000x1000	499270	499500	499500	499500	499500	499500	499500	499500	499500	1

Tabla A.1: Número de biclusters generados por *BiBit*.

NxN vs P %	10 %	20 %	30 %	40 %	50 %	60 %	70 %	80 %	90 %	100 %
50x50	16	32	31	31	31	31	47	31	32	0
100x100	47	62	62	63	62	63	63	63	62	0
150x150	93	109	125	109	109	110	109	125	125	0
200x200	141	219	203	188	203	203	203	203	203	0
250x250	343	422	328	328	360	312	359	344	406	15
300x300	672	625	515	594	563	500	500	531	718	15
350x350	1172	984	860	797	735	750	813	812	875	15
400x400	2062	1609	1422	1359	1344	1188	1407	1343	1328	15
450x450	3078	2188	2078	2063	1688	1735	2015	2078	2266	32
500x500	4172	2875	2234	2594	2297	2422	3000	2609	2641	31
550x550	5906	3734	3218	3109	3406	3156	3562	3453	4203	31
600x600	7610	4906	4641	4390	4375	4141	4063	4453	6000	31
650x650	9688	5906	5594	5125	5390	5562	5547	5344	6922	32
700x700	13265	6469	6922	6172	7406	6640	7140	6672	7359	31
750x750	9547	9094	7484	7734	8172	8015	8515	8047	10313	47
800x800	12046	14000	12031	11657	12485	10906	11907	10922	13781	47
850x850	14921	15843	14797	14641	15062	13860	14438	15844	15172	62
900x900	17484	15391	15454	15781	16250	17953	18032	17641	22047	62
950x950	20250	22969	20578	18781	19360	21360	22344	23328	25812	63
1000x1000	40735	22234	24390	24266	23265	24531	23859	27500	31625	78

Tabla A.2: Tiempo de ejecución en milisegundos empleado por *BiBit* por cada base de datos sintética.

NxN vs P %	10 %	20 %	30 %	40 %	50 %	60 %	70 %	80 %	90 %	100 %
50x50	4.19	18.34	35.87	39.29	39.52	39.52	26.06	39.52	38.16	0.00
100x100	20.15	69.15	79.66	78.57	79.84	78.57	78.57	78.57	79.84	0.00
150x150	41.60	100.47	89.40	102.52	102.52	101.59	102.52	89.40	89.40	0.00
200x200	69.71	90.54	98.03	105.85	98.03	98.03	98.03	98.03	98.03	0.00
250x250	56.90	73.71	94.89	94.89	86.46	99.76	86.70	90.48	76.66	0.07
300x300	49.10	71.76	87.09	75.51	79.66	89.70	89.70	84.46	62.47	0.07
350x350	42.78	62.07	71.02	76.63	83.10	81.43	75.12	75.22	69.80	0.07
400x400	34.09	49.60	56.12	58.72	59.38	67.17	56.72	59.42	60.09	0.07
450x450	30.13	46.17	48.62	48.97	59.85	58.23	50.14	48.62	44.58	0.03
500x500	28.32	43.39	55.84	48.09	54.31	51.51	41.58	47.82	47.24	0.03
550x550	24.73	40.43	46.92	48.56	44.33	47.84	42.38	43.72	35.92	0.03
600x600	23.12	36.63	38.72	40.93	41.07	43.40	44.23	40.35	29.95	0.03
650x650	21.49	35.71	37.71	41.16	39.13	37.92	38.03	39.47	30.47	0.03
700x700	18.29	37.82	35.34	39.64	33.03	36.84	34.26	36.67	33.25	0.03
750x750	29.27	30.89	37.53	36.32	34.37	35.04	32.99	34.90	27.24	0.02
800x800	26.44	22.83	26.56	27.42	25.60	29.30	26.84	29.26	23.19	0.02
850x850	24.13	22.78	24.39	24.64	23.96	26.03	24.99	22.77	23.78	0.02
900x900	23.11	26.28	26.18	25.64	24.90	22.53	22.44	22.93	18.35	0.02
950x950	22.24	19.63	21.91	24.00	23.28	21.10	20.17	19.32	17.46	0.02
1000x1000	12.26	22.47	20.48	20.58	21.47	20.36	20.94	18.16	15.79	0.01

Tabla A.3: Número de biclusters por milisegundo obtenidos en promedio por *BiBit*.

NxN vs P %	10 %	20 %	30 %	40 %	50 %	60 %	70 %	80 %	90 %	100 %
50x50	0.24	0.05	0.03	0.03	0.03	0.03	0.04	0.03	0.03	0.00
100x100	0.05	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00
150x150	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00
200x200	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00
250x250	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	15.00
300x300	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.02	15.00
350x350	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	15.00
400x400	0.03	0.02	0.02	0.02	0.02	0.01	0.02	0.02	0.02	15.00
450x450	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	32.00
500x500	0.04	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	31.00
550x550	0.04	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.03	31.00
600x600	0.04	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.03	31.00
650x650	0.05	0.03	0.03	0.02	0.03	0.03	0.03	0.03	0.03	32.00
700x700	0.05	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	31.00
750x750	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.04	47.00
800x800	0.04	0.04	0.04	0.04	0.04	0.03	0.04	0.03	0.04	47.00
850x850	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	62.00
900x900	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.05	62.00
950x950	0.04	0.05	0.05	0.04	0.04	0.05	0.05	0.05	0.06	63.00
1000x1000	0.08	0.04	0.05	0.05	0.05	0.05	0.05	0.06	0.06	78.00

Tabla A.4: Milisegundos que tarda *BiBit* en producir un sólo bicluster.

A.2.3. Velocidad de procesamiento de biclusters

Se ha comprobado que *BiBit* es más rápido que *Bimax*, pero hay que tener en cuenta que éste último genera un mayor número de resultados. La medida de la velocidad, es decir, número de biclusters generados por milisegundo, es una buena forma de establecer una comparativa justa entre los dos algoritmos. Y, tal y como se observa en la tabla A.7, *Bimax* alcanza unas velocidades muy pobres en comparación con *BiBit* (ver tabla A.3). También hemos recogido la misma información pero mostrando el tiempo en milisegundos que tarda *Bimax* en generar un sólo bicluster (ver tabla A.8).

NxN vs P %	**10 %**	**20 %**	**30 %**	**40 %**	**50 %**	**60 %**	**70 %**	**80 %**	**90 %**	**100 %**
50x50	68	737	3333	14533	69363	519723	-	-	-	1
100x100	989	10979	84618	768909	9246446	-	-	-	-	-

Tabla A.5: Número de biclusters generados por *BiMax*.

NxN vs P %	**10 %**	**20 %**	**30 %**	**40 %**	**50 %**	**60 %**	**70 %**	**80 %**	**90 %**	**100 %**
50x50	516	7891	4109	4266	8110	44594	-	-	-	0
100x100	171051625	91333094	3847704	6471453	17388575	-	-	-	-	-

Tabla A.6: Tiempo de ejecución en milisegundos empleado por *BiMax* para cada una de las 12 bases de datos que pudo procesar.

NxN vs P	**10 %**	**20 %**	**30 %**	**40 %**	**50 %**	**60 %**	**70 %**	**80 %**	**90 %**	**100 %**
50x50	0.13	0.09	0.81	3.41	8.55	11.65	-	-	-	0.00
100x100	0.00	0.00	0.02	0.12	0.53	-	-	-	-	-

Tabla A.7: Número de biclusters por milisegundo obtenidos en promedio por *Bimax*.

NxN vs P %	**10 %**	**20 %**	**30 %**	**40 %**	**50 %**	**60 %**	**70 %**	**80 %**	**90 %**	**100 %**
50x50	7.59	10.71	1.23	0.29	0.12	0.09	-	-	-	0.00
100x100	172954.12	8318.89	45.47	8.42	1.88	-	-	-	-	-

Tabla A.8: Milisegundos que tarda *Bimax* en producir um sólo bicluster.

Apéndice B

Aplicación de *BiBit* a una base de datos real

B.1. Listado de muestras del Dataset A1

En la tabla B.1 se recogen las 40 muestras que forman dicha base de datos, indicándose a que tipo de tumor pertenece cada una de ellas.

B.2. Listado de genes potenciales

A continuación recogemos aquellos genes potencialmente interesantes que han sido descubiertos en los 7 t*umor biclusters* obtenidos por *BiBit*. Para mayor comodidad, y debido al hecho de que son más de 200 genes, los hemos dividido en 2 tablas: B.2 y B.3

Sample Number	**Sample name**	**Type**
1	*Brain_MD_12*	Medulloblastoma
2	*Brain_MD_61*	Medulloblastoma
3	*Brain_MD_15*	Medulloblastoma
4	*Brain_MD_57*	Medulloblastoma
5	*Brain_MD_33*	Medulloblastoma
6	*Brain_MD_64*	Medulloblastoma
7	*Brain_MD_17*	Medulloblastoma
8	*Brain_MD_62*	Medulloblastoma
9	*Brain_MD_63*	Medulloblastoma
10	*Brain_MD_32*	Medulloblastoma
11	*Brain_MGlio_1*	Malignant Glioma
12	*Brain_MGlio_2*	Malignant Glioma
13	*Brain_MGlio_3*	Malignant Glioma
14	*Brain_MGlio_4*	Malignant Glioma
15	*Brain_MGlio_5*	Malignant Glioma
16	*Brain_MGlio_6*	Malignant Glioma
17	*Brain_MGlio_7*	Malignant Glioma
18	*Brain_MGlio_8*	Malignant Glioma
19	*Brain_MGlio_9*	Malignant Glioma
20	*Brain_MGlio_10*	Malignant Glioma
21	*Brain_Rhab_1*	AT/RT (Brain)
22	*Brain_Rhab_2*	AT/RT (Renal)
23	*Brain_Rhab_3*	AT/RT (Renal)
24	*Brain_Rhab_4*	AT/RT (Brain)
25	*Brain_Rhab_5*	AT/RT (Extra Renal)
26	*Brain_Rhab_6*	AT/RT (Extra Renal)
27	*Brain_Rhab_7*	AT/RT (Renal)
28	*Brain_Rhab_8*	AT/RT (Brain)
29	*Brain_Rhab_9*	AT/RT (Brain)
30	*Brain_Rhab_10*	AT/RT (Brain)
31	*Brain_Ncer_1*	Normal cerebellum
32	*Brain_Ncer_2*	Normal cerebellum
33	*Brain_Ncer_3*	Normal cerebellum
34	*Brain_Ncer_4*	Normal cerebellum
35	*Brain_PNET_1*	PNET
36	*Brain_PNET_2*	PNET
37	*Brain_PNET_3*	PNET
38	*Brain_PNET_4*	PNET
39	*Brain_PNET_5*	PNET
40	*Brain_PNET_6*	PNET

Tabla B.1: Por cada muestra del Dataset A1 mostramos su número, su nombre y el tipo de tumor al que corresponde

$AC002045\_xpt2\_s\_at$	$M77232\_rna1\_at$	$D87735\_at$	$U30521\_at$
$AC002115\_cds1\_at$	$M81757\_at$	$HG1153-HT1153\_at$	$U32944\_at$
$AFFX-HSAC07$	$M84332\_at$	$HG1322-HT5143\_s\_at$	$U44839\_at$
$D00017\_at$	$M84711\_at$	$HG1428-HT1428\_s\_at$	$U46025\_at$
$D13413\_rna1\_s\_at$	$M86400\_at$	$HG1515-HT1515\_f\_at$	$U46751\_at$
$D13639\_at$	$M94250\_at$	$HG1612-HT1612\_at$	$U48437\_at$
$D13748\_at$	$M94880\_f\_at$	$HG1980-HT2023\_at$	$U49869\_rna1\_at$
$D14710\_at$	$S45630\_at$	$HG2279-HT2375\_at$	$U54778\_at$
$D14812\_at$	$S54005\_s\_at$	$HG3431-HT3616\_s\_at$	$U58682\_at$
$D21261\_at$	$S72043\_rna1\_at$	$HG3510-HT3704\_at$	$U68105\_s\_at$
$D21267\_at$	$S79522\_at$	$HG3514-HT3708\_at$	$U70439\_s\_at$
$D25274\_at$	$S82297\_at$	$HG3543-HT3739\_at$	$U72511\_at$
$D26068\_at$	$U00947\_s\_at$	$HG384-HT384\_at$	$U73377\_at$
$D30655\_at$	$U02493\_at$	$HG417-HT417\_s\_at$	$U73824\_at$
$D31883\_at$	$U04241\_at$	$HG4319-HT4589\_at$	$U78027\_rna3\_at$
$D31890\_at$	$U07857\_at$	$HG4542-HT4947\_at$	$U90915\_at$
$D32129\_f\_at$	$U09813\_at$	$HG613-HT613\_at$	$U94586\_at$
$D49824\_s\_at$	$U09953\_at$	$HG658-HT658\_f\_at$	$U94855\_at$
$D50310\_at$	$U12404\_at$	$HG662-HT662\_at$	$U95040\_at$
$D61380\_at$	$U14968\_at$	$HG821-HT821\_at$	$V00572\_at$
$D63851\_at$	$U14970\_at$	$HG987-HT987\_at$	$V00594\_s\_at$
$D63874\_at$	$U14971\_at$	$J00105\_s\_at$	$V00599\_s\_at$
$D63878\_at$	$U14972\_at$	$J02783\_at$	$X01703\_at$
$D78611\_at$	$U15008\_at$	$J03040\_at$	$X02152\_at$
$D79205\_at$	$U24105\_at$	$J03191\_at$	$X02761\_s\_at$
$D86974\_at$	$U25789\_at$	$J03242\_s\_at$	
$D87465\_at$	$U29607\_at$	$X04106\_at$	

Tabla B.2: Genes potencialmente interesantes, parte 1.

$J03592\_at$	$X04347\_s\_at$	$M18000\_at$	$X62654\_rna1\_at$
$J03827\_at$	$X05196\_at$	$M19283\_at$	$X62691\_at$
$J04173\_at$	$X05610\_at$	$M19483\_at$	$X64707\_at$
$J04456\_at$	$X06700\_s\_at$	$M20471\_at$	$X67247\_rna1\_at$
$J04615\_at$	$X07979\_at$	$M21142\_cds2\_s\_at$	$X67951\_at$
$J04823\_rna1\_at$	$X12447\_at$	$M22382\_at$	$X69391\_at$
$K03515\_at$	$X12671\_rna1\_at$	$M23613\_at$	$X69654\_at$
$L04483\_s\_at$	$X13546\_rna1\_at$	$M24485\_s\_at$	$X70940\_s\_at$
$L06132\_at$	$X13794\_rna1\_at$	$M26708\_s\_at$	$X71428\_at$
$L06505\_at$	$X15183\_at$	$M26880\_at$	$X78136\_at$
$L10284\_at$	$X15341\_at$	$M27891\_at$	$X79234\_at$
$L10373\_at$	$X15880\_at$	$M31520\_at$	$X80818\_at$
$L11373\_at$	$X15940\_at$	$M32053\_at$	$X80909\_at$
$L11566\_at$	$X16560\_at$	$M32405\_at$	$Y00433\_at$
$L11672\_at$	$X17620\_at$	$M33336\_at$	$Y09836\_at$
$L19527\_at$	$X52851\_rna1\_at$	$M37457\_at$	$Z19554\_s\_at$
$L19686\_rna1\_at$	$X52882\_at$	$M55998\_s\_at$	$Z23090\_at$
$L25080\_at$	$X52966\_at$	$M58028\_at$	$Z25749\_rna1\_at$
$L26247\_at$	$X53331\_at$	$M60854\_at$	$Z28407\_at$
$L38941\_at$	$X53777\_at$	$M60858\_rna1\_at$	$Z29505\_at$
$M11353\_at$	$X55715\_at$	$M61832\_s\_at$	$Z37166\_at$
$M11718\_at$	$X55954\_at$	$M63138\_at$	$Z48501\_s\_at$
$M13577\_at$	$X56494\_at$	$M63379\_at$	$Z48950\_at$
$M13934\_cds2\_at$	$X56681\_s\_at$	$M64716\_at$	$Z49148\_s\_at$
$M14199\_s\_at$	$X56997\_rna1\_at$	$M69066\_at$	$Z84721\_cds2\_at$
$M14328\_s\_at$	$X57959\_at$	$M17886\_at$	
$M14483\_rna1\_s\_at$	$X60036\_at$	$X60489\_at$	

Tabla B.3: Genes potencialmente interesantes, parte 2.

Bibliografía

Aguilar, J. (2005) Shifting and scaling patterns from gene expression data. *Bioinformatics,* **21**, 3840–3845.

Aguilar, J. y Divina, F. (2006) Evolutionary computation for biclustering of gene expression. *IEEE Transactions on Knowledge and Data Engineering,* **18**, 590–602.

Alizadeh, A., Eisen, M., Davis, R. y et al. (2000) Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature,* **403**, 503–511.

Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D. y Levine, A. (1999) Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide array. *Proc. Natl. Acad. Sci. USA,* **96**, 6745–6750.

Alqadah, F., Bhatnagar, R. y Jegga, A. (2010) A novel framework for detecting maximally banded matrices in binary data. *Statistical Analysis and Data Mining,* **3**, 431–445.

Altermann, E. y Klaenhammer, T. (2005) Pathwayvoyager: pathway mapping using the kyoto encyclopedia of genes and genomes (kegg) database. *BMC Genomics,* **6** (1), 60–67.

Aoki, K., Yamaguchi, A., Ueda, N., Akutsu, T., Mamitsuka, H., Goto, S. y Kanehisa, M. (2004) Kcam (kegg carbohydrate matcher): a software tool for analyzing the structures of carbohydrate sugar chains. *Nucleic Acids Research,* **32**, 267–272.

Ben-Dor, A., Chor, B., Karp, R. y Yakhini, Z. (2002) Discovering local structure in gene expression data: the order preserving submatrix problem. In *6th International Conference on Computacional Biology, RECOMB.* pp. 49–57.

Ben-Dor, A., Friedman, N. y Yakhini, Z. (2001) Class discovery in gene expression data. In *5th International Conference on Computacional Biology, RECOMB.* pp. 31–38.

Ben-Dor, A., Shamir, R. y Yakhini, Z. (1999) Clustering gene expression patterns. *Journal of Computational Biology,* **6**, 281–297.

Berriz, G., Beaver, J., Cenik, C., Tasam, M. y Roth, F. (2009) Next generation software for functional trend analysis. *Bioinformatics,* **25**, 3043–3044.

Berriz, G., King, O., Bryant, B., Sander, C. y Roth, F. (2003) Characterizing gene sets with FuncAssociate. *Bioinformatics,* **19** (18), 2502–2504.

Bhattacharya, A. y Rajat, D. (2009) Bi-correlation clustering algorithm for determining a set of co-regulated genes. *Bioinformatics,* **25**, 2795–2801.

Bland, J. y Altman, D. (1995) Multiple significance tests - the bonferroni method. *Brittish Medical Journal,* **310**, 170.

Blatt, M., Wiseman, S. y Domany, E. (1996) Super-paramagnetic clustering of data. *Physical Review Letters,* **76**, 3251–3254.

Boyle, E., Weng, S., Gollub, J., Jing, H., Botstein, D., Cherry, J. y Sherlock, G. (2004) GO::TermFinder–open source software for accessing Gene Ontology information and finding significantly enriched Gene Ontology terms associated with a list of genes. *Bioinformatics,* **20** (18), 3710–3715.

Brazma, A. y Vilo, J. (2000) Minireview: gene expression data analysis. *Federation of European Biochemical societies,* **480**, 17–24.

Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., Furey, T., Ares, M. y Haussler, D. (2000) Knowledge-based analysis of microarray gene expression data using support vector machines. *Proc.Natl. Acad. Sci,* **97**, 262–267.

Brunet, J., Tamayo, P., Golub, T. y Mesirov, J. (2004) Metagenes and molecular pattern discovery using matrix factorization. *Proc Natl Acad Sci USA,* **101**, 4164–4169.

Burton, K. y Krishna, M. (2010) Differential co-expression framework to quantify goodness of biclusters and compare biclustering algorithms. *Algorithms for computational biology,* **5**.

Carmona-Saez, P., Pascual-Marqui, R., Tirado, F., Carazo, J. y Pascual-Montano, A. (2006) Biclustering of gene expression data by non-smooth non-negative matrix factorization. *BMC Bioinformatics,* **7** (1), 78. M3: 10.1186/1471-2105-7-78.

Castriconi, R., D. A. y et al. (2009) Nk cells recognize and kill human glioblastoma cells with stem cell-like properties. *The Journal of Immunology,* **182**, 3530–3539.

Cheng, Y. y Church, G. (2000) Biclustering of expression data. In *8th International Conference on Intelligent Systems for Molecular Biology,ISMB* pp. 93–103.

Colantonio, A., Di Pietro, R., Ocello, A. y Verde, N. V. (2010) Abba: adaptive bicluster-based approach to impute missing values in binary matrices. In *25th ACM Symposium on Applied Computing, SAC10* pp. 1026–1033.

Consortium, G. (2006) The gene ontology (go) project in 2006. *Nucleic Acids Research,* **34**, 322–326.

Cormen, T., Leiserson, C., Rivest, R. y Stein, C. (2001) Introduction to algorithms. *The MIT Electrical Engineering and Computer Science Series. The MIT Press, 2nd edition,* .

David, R. (2001) Robust cluster analysis of dna microarray data: an application of nonparametric correlation dissimilarity. In *Joint Statistical Meetings of the American Statistical Association (Biometrics Section).*

Dempster, A., Laird, N. y Rubin, D. (1977) Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society,* **39**, 1–38.

DeRisi, J., Penland, L., Brown, P., Bittner, M., Meltzer, P., Ray, M., Chen, Y., Su, Y. y Trent, J. (1996) Use of a cdna microarray to analyse gene expression patterns in human cancer. *Nature. Genetics,* **14**, 457–460.

Dhaeseleer, P., Wen, X., Furhman, S. y Somogyi, S. (1998) Mining the gene expression matrix: inferring gene relationships from large scale gene expression data. In *Information Processing in Cells and Tissues, IPCAT98* pp. 121–203.

Diaz-Diaz, N. y Aguilar-Ruiz, J. (2011) Go-based functional dissimilarity of gene sets. *BMC Bioinformatics,* **12**, 360.

DiMaggio, P., McAllister, D., Christodoulos, A., Xiao-Jiang, F., Joshua, D. y Herschel, R. (2008) Biclustering via optimal re-ordering of data matrices in systems biology: rigorous methods and comparative studies. *BMC Bioinformatics,* **9**.

Ding, C., He, X. y Simon, H. (2005) On the equivalence of nonnegative matrix factorization and spectral clustering. In *SIAM data mining conference.*

Draghici, S. y et al. (2003) Onto-tools, the toolkit of the modern biologist: onto-express, onto-compare, onto-design and onto-translate. *Nucleic Acids Research,* **31**, 3775–3781.

Dubes, R. y Jain, A. (1988) *Algorithms for Clustering Data.* Prentice-Hall.

Duffy, D. y Quiroz, A. (1991) A permutation based algorithm for block clustering. *Journal of Classification,* **8**, 65–91.

Efron, B. (1982) The jackknife, the bootstrap, and other resampling plans. In *CBMS-NSF Regional Conference Series in Applied Mathematics.*

Eisen, M., Spellman, P., Brown, P. y Botstein, D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA,* **92**, 14863–14868.

Estelles, A., Y. M. y et al. (1996) The major astrocytic phosphoprotein pea-15 is encoded by two mrnas conserved on their full length in mouse and human. *Journal of Biological Chemistry,* **271**, 14800–14806.

Falcon, S. y Gentleman, R. (2007) Using GOstats to test gene lists for GO term association. *Bioinformatics,* **23** (2), 257–258.

Figueroa, A., Borneman, J. y Jiang, T. (2004) Clustering binary fingerprint vectors with missing values for dna array data analysis. *Journal of Computational Biology,* **11**, 887–901.

Fraley, C. y Raftery, A. (1998) How many clusters? which clustering method? answers via model-based cluster analysis. *The Computer Journal,* **41**, 578–588.

Getz, G., Levine, E. y Domany, E. (2000) Coupled two-way clustering analysis of gene microarray data. In *Proceedings of the Natural Academy of Sciences USA* pp. 12079–12084.

Ghosh, D. y Chinnaiyan, A. (2002) Mixture modelling of gene expression data from microarray experiments. *Bioinformatics,* **18**, 275–286.

Gibbons, F. D. y Roth, F. P. (2002) Judging the quality of gene expression-based clustering methods using gene annotation. *Genome Research,* **12**, 1574–1581.

Golub, T., Slonim, D., Tamayo, P., Huard, C., Gassenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, D. y Lander, E. (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science,* **286**, 531–537.

Gordon, G. y et al. (2002) Translation ofmicroarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer Research,* **62**, 4963–4967.

Griffiths, A., Wessler, S., Lewontin, R., Gelbart, W., Suzuki, D. y Miller, J. (2005) *Introduction to Genetic Analysis.* W.H. Freeman.

Halkidi, M. y et al. (2001) On clustering validation techniques. *J. Intell. Inform. Systems,* **17**, 107–145.

Harpaz, R. y Perez, H. (2010) Biclustering of adverse drug events in the fdas spontaneous reporting system. *Clinical Pharmacology and Therapeutics,* **89**, 243–250.

Hartigan, J. A. (1972) Direct clustering of a data matrix. *Journal of the American Statistical Association (JASA),* **67**, 123–129.

Herrero, J., Valencia, A. y Dopazo, J. A. (2001) hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics,* **17**, 126–136.

Heyer, L., Kruglyak, S. y Yooseph, S. (1999) Exploring expression data: identification and analysis of coexpressed genes. *Genome Research,* **9**, 1106–1115.

Hill, A., Brown, E., Whitley, M., Tucker-Kellogg, G., Hunter, C. y Slonim, D. (2001) Evaluation of normalization procedures for oligonucleotide array data based on spiked crna controls. *Genome Biology,* **12**.

Huang, D. W., Sherman, B. T., Tan, Q., Kir, J., Liu, D., Bryant, D., Guo, Y., Stephens, R., Baseler, M. W., Lane, H. C. y Lempicki, R. A. (2007)

DAVID Bioinformatics Resources: expanded annotation database and novel algorithms to better extract biology from large gene lists. *Nucleic Acids Research,* **35** (suppl 2), 169–175.

Huerta, M., Haseltine, F., Liu, Y., Downing, G. y Seto, B. (2000) *NIH Working Definition of Bioinformatics and Computational Biology.* U.S National Institute of Health.

Ideker, T., Galitzki, T. y Hood, L. (2001) Integrated genomic and proteomic analyses of a sistematically perturbed metabolic network. *Science,* **292**, 929–934.

Jain, A., Murty, M. y Flynn, P. (1999) Data clustering: a review. *ACM Computing Surveys,* **31**, 254–323.

Jiang, D., Pei, J. y Zhang, A. (2003) Dhc: a density-based hierarchical clustering method for timeseries gene expression data. In *3rd IEEE International Symposium on Bioinformatics and Bioengineering, BIBE.*

Kanehisa, M., Araki, M., Goto, S., Hattori, M., Hirakawa, M., Itoh, M., Katayama, T., Kawashima, S., Okuda, S., Tokimatsu, T. y Yamanishi, Y. (2008) KEGG for linking genomes to life and the environment. *Nucleic Acids Research,* **36** (suppl 1), 480–484.

Kano, M., Nishimura, K., Tsutsumi, S., Aburatani, H., Hirota, K. y Hirose, M. (2003) Cluster overlap distribution map: visualization for gene expression analysis using immersive projection technology. *Presence: Teleoperators and Virtual Environments. MIT press,* **12**, 96–109.

Karlsson, E., Delle, U., Danielsson, A., Olsson, B., Abel, F., Karlsson, P. y Helou, K. (2008) Gene expression variation to predict 10-year survival in lymph-node-negative breast cancer. *BMC Cancer,* **8** (1), 254.

Kaufman, L. y Rousseeuw, P. (1990) *Finding Groups in Data: an Introduction to Cluster Analysis.* John Wiley and Sons.

Kerr, G., Ruskin, H. J., Crane, M. y Doolan, P. (2008) Techniques for clustering gene expression data. *Computers in biology and medicine,* **38** (3), 283–293.

Klugar, Y., Basri, R., Chang, J. y Gerstein, M. (2003) Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research,* **13**, 703–716.

Klukas, C. y Schreiber, F. (2007) Dynamic exploration and editing of KEGG pathway diagrams. *Bioinformatics,* **23** (3), 344–350.

Kohonen, T. (1997) *Self-Organizaing Maps.* Series in Information Sciences, Springer.

Koyuturk, M., Szpankowski, W. y Grama, A. (2004) Biclustering gene-feature matrices for statistically significant dense patterns. In *Computational Systems Bioinformatics Conference, International IEEE Computer Society* pp. 480–484.

Kroese, D., Taimre, T. y Botev, Z. (2011) *Handbook of Monte Carlo Methods.* John Wiley and Sons.

Lazzeroni, L. y Owen, A. (2000) Plaid models for gene expression data. *Technical report, Stanford University,* .

Lee, D. y Seung, H. (2001) Algorithms for non-negative matrix factorization. *Advances in neural information processing systems,* **13**.

Li, T. (2005) A general model for clustering binary data. In *11th ACM SIGKDD international conference* pp. 188–197.

Lu, H., Vaidya, J. y Atluri, V. (2008) Optimal boolean matrix decomposition: application to role engineering. In *IEEE 24th International Conference on Data Engineering* pp. 297–306.

Madeira, S. C. y Oliveira, A. L. (2004) Biclustering algorithms for biological data analysis: a survey. *IEEE Transactions on computational Biology and Bioinformatics,* **1**, 24–45.

Madeira, S. C. y Oliveira, A. L. (2009) A polynomial time biclustering algorithm for finding approximate expression patterns in gene expression time series. *Algorithms for Molecular Biology,* **4**, 1–39.

Mahanta, P., Ahmed, H., Bhattacharyya, D. y Kalita, J. (2011) Triclustering in gene expression data analysis: a selected survey. In *2nd National Conference on Emerging Trends and Applications in Computer Science, NCETACS* pp. 1–6.

McLachlan, G., Bean, R. y Peel, D. (2002) A mixture model-based approach to the clustering of microarray expression data. *Bioinformatics,* **18**, 413–422.

McQueen, J. (1967) Some methods for classification and analysis of multivariate observations. In *The Fifth Berkeley Symposium on Mathematical Statistics and Probability* pp. 281–297.

Mering, V., Jensen, L. y Khun, M. (2007) String 7: recent developments in the integration and prediction of protein interactions. *Nucleic Acids Research,* **35**, 358–362.

Mimaroglu, S. y Simovici, D. (2007) Bit sequences and biclustering of text documents. In *Seventh IEEE International Conference on Data Mining Workshops* pp. 51–56.

Nighat, N. y Muhammad, A. (2009) Bisim: a simple and efficient biclustering algorithm. In *2009 International Conference of Soft Computing and Pattern Recognition* pp. 1–6.

Olson, N. (2006) The microarray data analysis process: from raw data to biological significance. *NeuroRx,* **3**, 373–383.

Peeters, R. (2003) The maximum edge biclique problem is np-complete. *Discrete Applied Mathematics,* **131**, 651–654.

Perco, P., Kainz, A., Mayer, G., Lukas, A., Oberbauer, R. y Mayer, B. (2005) Detection of coregulation in differential gene expression profiles. *Biosystems,* **82** (3), 235–247.

Pomeroy, S., Tamayo, P. y et al. (2002) Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature,* **415**, 436–442.

Prelic, A., Bleuler, S., Zimmermann, P., Wille, A., Buhlmann, P., Gruissem, W., Hennig, L., Thiele, L. y Zitzler, E. (2006) A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics,* **22** (9), 1122–1129.

Puolamaki, K., F. M. y Mannila, H. (2006) Seriation in paleontological data using markov chain monte carlo methods. *PLoS Computational Biology,* **2**.

RalfHerwig, Poustka, A., Muller, C., Bull, C., Lehrach, H. y OBrien, J. (1999) Large-scale clustering of cdna-fingerprinting data. *Genome Research,* **9**, 1093–1105.

Rivals, I., Personnaz, L., Taing, L. y Potier, M. (2007) Enrichment or depletion of a go category within a class of genes: which test? *Bioinformatics,* **23** (4), 401–407.

Schuchhardt, J., Beule, D., Malik, A., Wolski, E., Eickhoff, H., Lehrach, H. y Herzel, H. (2000) Normalization strategies for cdna microarrays. *Nucleic Acids Research,* **28**.

Segal, E., Wang, H. y Koller, D. (2003) Discovering molecular pathways from protein interaction and gene expression data. *Bioinformatics,* **19**, 264–272.

Serin, A. y Vingron, M. (2011) Debi: discovering differentially expressed biclusters using a frequent itemset approach. *Algorithms for Molecular Biology,* **6**.

Shamir, R., Maron-Katz, A., Tanay, A., Linhart, C., Steinfeld, I., Sharan, R., Shiloh, Y. y Elkon, R. (2005) EXPANDER – an integrative program suite for microarray data analysis. *BMC Bioinformatics,* **6** (1), 232.

Shamir, R. y Sharan, R. (2000) Click: a clustering algorithm for gene expression analysis. In *8th International Conference on Intelligent Systems for Molecular Biology.*

Sherlock, G. (2000) Analysis of large-scale gene expression data. *Curr. Opin. Immunology,* **12**, 201–205.

Shmulevich, I. y Zhang, W. (2002) Binary analysis and optimization-based normalization of gene expression data. *Bioinformatics,* **18** (4), 555–565.

Smedley, D., H. S. y et al. (2009) Biomart: biological queries made easy. *BMC Genomics,* **10**.

Smet, F., Mathys, J., Marchal, K., Thijs, G., Moor, B. y Moreau, Y. (2002) Adaptive quality-based clustering of gene expression profiles. *Bioinformatics,* **18**, 735–746.

Sun, X. y Nobel, A. (2008) On the size and recovery of submatrices of ones in a random binary matrix. *Journal of Machine Learning Research,* **9**, 2431–2453.

Tamayo, P., Solni, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. y Golub, T. (1999) Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci. USA,* **96**, 2907–2912.

Tanay, A., Sharan, R. y Shamir, R. (2002) Discovering statistically significant biclusters in gene expression data. *Bioinformatics,* **18**, 136–144.

Tang, C., Zhang, A., y Pei, J. (2003) Mining phenotypes and informative genes from gene expression data. In *9th ACM SIGKDD International Conference on Knowledge Discovery andData Mining, SIGKDD.*

Tang, C. y Zhang, A. (2002) An iterative strategy for pattern discovery in high-dimensional data sets. In *11th International Conference on Information and Knowledge Management.*

Tang, C., Zhang, L.and Zhang, A. y Ramanathan, M. (2001) Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. In *2nd IEEE International Symposium on Bioinformatics and Bioengineering, BIBE* pp. 41–48.

Tavazoie, S., Hughes, D., Campbell, M., Cho, R. y Church, G. (1999) Systematic determination of genetic network architecture. *Nature Genetics,* , 281–285.

Thomas, J., Olson, J., Tapscott, S. y Zhao, L. (2001) An efficient and robust statistical modeling approach to discover differentially expressed genes using genomic expression profiles. *Genome Research,* **11**, 1227–1236.

Tibshirani, R., Hastie, T., Eisen, M., Ross, D., Botstein, D. y Brown, P. (1999) *Clustering methods for the analysis of DNA microarray data.* Technical report, Department of Health Research and Policy, Department of Genetics and Department of Biochemestry, Stanford University.

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D. y R., A. (2001) Missing value estimation methods for dna microarrays. *Bioinformatics,* **17**, 520–525.

Uitert, M. v., Meuleman, W. y Wessels, L. (2008) Biclustering sparse binary genomic data. *Journal of Computational Biology,* **15** (10), 1329–1345.

Wang, H., Wang, W., Yang, J. y Yu, P. S. (2002) Clustering by pattern similarity in large data sets. In *ACM SIGMOD International Conference on Management of Data* pp. 394–405.

Wang, S., Gutell, R. R. y Miranker, D. P. (2007) Biclustering as a method for rna local multiple sequence alignment. *Bioinformatics,* **23** (24), 3289.

Westfall, P. y Young, S. (1989) p Value Adjustments for Multiple Tests in Multivariate Binomial Models. *Journal of the American Statistical Association,* **84** (407), 780–786.

Xing, E. y Karp, R. C. (2001) Clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. *Bioinformatics,* **17**, 306–315.

Yang, J. y Wang, W. (2002) Improving performance of bicluster discovery in a large data set. In *6th ACM International Conference on Research in Computational Molecular Biology*.

Yang, J. y Wang, W. (2003) Enhanced biclustering on expression data. In *3rd IEEE Conference on Bioinformatics and Bioengineering* pp. 321–327.

Yang, J., Wang, W., Wang, H. y Yu, P. (2002) δ-cluster: capturing subspace correlation in a large data set. In *18th International Conference on Data Engineering*.

Yeung, K., Fraley, C., Murua, A., Raftery, A. y Ruzz, W. (2001) Model-based clustering and data transformations for gene expression data. *Bioinformatics,* **17**, 977–987.

Yokota, N., A. J. y et al. (1996) Predominant expression of human zic in cerebellar granule cell lineage and medulloblastoma. *Cancer Research,* **56**, 377–383.

Zhang, Z.-Y., Li, T., Ding, C., Ren, X.-W. y Zhang, X.-S. (2010) Binary matrix factorization for analyzing gene expression data. *Data Mining and Knowledge Discovery,* **20** (1), 28–52.

Zhong, S., Storch, F., Lipan, O., Kao, M., Weitz, C. y Wong, W. (2004) GoSurfer: a graphical interactive tool for comparative analysis of large gene sets in Gene Ontology space. *Applied Bioinformatics,* **3** (4), 1–5.