Universidad Central Marta Abreu de las Villas
Facultad de Matemática - Física - Computación
Licenciatura en Ciencias de la Computación

TRABAJO DE DIPLOMA

APLICACIÓN DE MEDIDAS DE SIMILITUD Y ALGORITMOS DE AGRUPAMIENTO A LA DETECCIÓN DE GENES ORTÓLOGOS

Reinier Millo Sánchez

Tutora:

Msc. Deborah Galpert Cañizares

Consultante:

Dr. Robersy Sánchez Rodríguez

Año 54 de la Revolución
Santa Clara
Junio 2012

DICTAMEN

El que suscribe, Reinier Millo Sánchez, hago constar que el trabajo titulado "Aplicación de medidas de similitud y algoritmos de agrupamiento a los algoritmos de detección de genes ortólogos" fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de los estudios de la especialidad de Ciencia de la Computación, autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

Firma del autor
Reinier Millo Sánchez

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

Firma del tutor
Msc. Deborah Galpert Cañizares

Firma del jefe del seminario
Dra. Gladys Casas Cardoso

Fecha

"Mil rutas se apartan del fin elegido, pero hay una que llega a él"

Michel Eyquem de Montaign

A la memoria de mi abuela Nena...
A la memoria de mi abuelo Marcelo...
A mis padres...
A mi familia...

A mi tutora Msc. Deborah Galpert por todo su apoyo y dedicación.

A mi familia en especial mis padres y mi hermana, por quererme y tener siempre todo su apoyo...
... a mis abuelos y tíos, por su cariño ...
... a todos ellos por hacer de mi la persona que soy.

A Claudia por brindarme todos su amor y cariño, por estar siempre a mi lado y ser mi motivo de inspiración.

A Marlene Brito por abrirme las puertas de su casa y recibirme como un miembro más de la familia.

A Zenaida Martínez a quien quiero y estimo mucho, por brindarme todo su cariño y considerarme otro nieto más.

A Fernando Rosales, Jarvin Antón, Osmany Morffa, Abel Cepero, Claudia Companioni, Karelia Díaz, Marilyn Bello, José Manuel por soportarme y compartir juntos todos los momentos de la carrera.

Al Guille por su amistad, los momentos compartidos y su ayuda para el desarrollo de este trabajo.

A la Dra. María Matilde García por su preocupación y recibirnos en su casa.

A la Dra. Gladys Casas y al Dr. Ricardo Grau por su apoyo, dedicarme su tiempo y su experiencia.

Al Dr. Robersy Sánchez, por ayudarnos al desarrollo de este trabajo.

Al Msc. Mario Pupo por todas las dudas aclaradas en el transcurso de la investigación.

Al Msc. Oscar Borroto por sus oportunas sugerencias en cada seminario y con quien compartí el tiempo de cómputo.

Al Lic. Jorge Enrique Moreira a quien agradezco todo el conocimiento aportado en los eventos de la ACM y su ayuda para la realización de este trabajo.

A todo el colectivo del grupo de Bioinformática por todo su apoyo y oportunas sugerencias.

A todo el claustro de profesores por el conocimiento que me han brindado en estos cinco años de la carrera.

A todas las personas que de una forma u otra han colaborado con la realización de este trabajo.

A la Revolución por permitirme estudiar y formarme como profesional.

A todos muchas gracias

RESUMEN

En el presente trabajo se realiza un estudio sobre las diferentes medidas de similitud que caracterizan la relación entre los pares de secuencias de dos genomas, los algoritmos de detección de genes ortólogos, los algoritmos de agrupamientos sobre grafos aplicados a la detección de genes ortólogos y las medidas de validación de agrupamientos. Se presenta un nuevo algoritmo para la detección de ortólogos, usando el algoritmo de agrupamiento *MCL* sobre el grafo bipartido de la similitud de las secuencias. La implementación del algoritmo se distribuye entre *MATLAB* y *Java*, aprovechando las ventajas que brinda cada uno.

Se exponen estrategias de poda del grafo bipartido usando un umbral para la homología de las secuencias, la similitud de las secuencias y la eliminación de ambigüedades del grafo. Con el objetivo de disminuir la relación de falsos positivos al tener en cuenta las relaciones de ortólogos muchos a muchos, se define una nueva política para la asignación de ortólogos a partir de los grupos de homología obtenidos del agrupamiento del grafo.

Los resultados del algoritmo se validan con los genomas de *Saccharomyces Cerevisiae* en y el *Schizosaccharomyce Pombe* usando la lista de genes ortólogos obtenida por el algoritmo de *INPARANOID 7.0*, con la medida de validación externa *ARI*. Los resultados se comparan con resultados de modelos de regresión obtenidos con ayuda del *SPSS*.

ABSTRACT

This thesis paper studies various similarity measures featuring the relationship between pairs of sequences of two genomes, different ortholog detection algorithms, graph clustering algorithms applied to ortholog detection problems, and clustering validation measures. A novel ortholog detection algorithm with the *MCL* clustering over the bipartite similarity graph is also presented. This algorithm implemented on *MATLAB* and *Java* benefits from both of these languages.

Some pruning strategies are described for the bipartite graph. One strategy is based on the threshold of the sequence homology, another one on the sequence similarity and the third one on the deletion of the ambiguities in the graph. A new ortholog assigning policy is defined over the homology groups obtained from the *MCL* clustering, in order to decrease the obtained false positive rate taking into to account the many-to-many ortholog relationships.

The algorithm results were validated by using the external validation measure adjusted *Rand* index with *Saccharomyces Cerevisiae* and *Schizosaccharomyce Pombe* genomes against the ortholog list reported by the *INPARANOID 7.0* algorithm. The final results were compared with the results of some logistic regression models constructed with the *SPSS*.

TABLA DE FIGURAS

TABLA DE CONTENIDOS

TABLA DE ACRÓNIMOS

ARI Índice de Rand ajustado, acrónimo del inglés ***Adjusted Rand Index***

BBH Mejores correspondencias bidireccionales, acrónimo del inglés ***Bidirectional Best Hit***

BLAST Herramienta de búsqueda de regiones similares entre secuencias biológicas, acrónimo del inglés ***Basic Local Alignment Search Tool***

BLOSUM Acrónimo del inglés ***BLOcks SUbstitution Matrix***

BUS Mejores subconjuntos no ambiguos, acrónimo del inglés ***Best Unambiguous Subsets***

CITMA Acrónimo de *Centro de Investigación Tecnología y Medio Ambiente*

COG Acrónimo del inglés ***Cluster Ortholog Groups***

CTWC Acrónimo del inglés ***Coupled Two-Way Clustering***

DCC Acrónimo del inglés ***Double Conjugated Clustering***

EGO Base de datos de genes ortólogos eucariotas, acrónimo del inglés ***Eukaryotic Gene Orthologs***

FASTA Acrónimo del inglés ***FAST All***

FLOC Acrónimo del inglés ***FLexible Overlapped biClustering***

ITWC Acrónimo del inglés ***Interrelated Two-Way Clustering***

KOG Base de datos de grupos ortólogos de organismos eucariotas, acrónimo del inglés ***euKaryotic Orthologous Group***

LCB Acrónimo del inglés ***Locally Collinear Blocks***

LPC Codificación predictiva lineal, acrónimo del inglés ***Lineal Predictive Coding***

MATLAB Acrónimo del inglés ***MATrix LABoratory***

MCL Acrónimo del inglés ***Markov CLustering***

MCODE Acrónimo del inglés ***Molecular COmplex DEtection***

MUSCLE Acrónimo del inglés ***MUltiple Sequence Comparison by Log-Expectation***

NCBI Acrónimo del inglés ***National Center for Biotechnology Information***

OWA Acrónimo del inglés ***Ordered Weighted Average operator***

PAM Acrónimo del inglés ***Percent Accepted Mutations***

PCBBH Pares de mejores correspondencia bidireccionales cercanos, acrónimo del inglés ***Pairs of Close Bidirectional Best Hits***

RBH Mejores correspondencias recíprocas, acrónimo del inglés ***Reciprocal Best Hit***

RNSC Acrónimo del inglés ***Restricted Neighborhood Search Clustering***

ROCK Acrónimo del inglés ***RObust Clustering using linKs***

RSD Menores distancias recíprocas, acrónimo del inglés ***Reciprocal Smallest Distance***

SAMBA Acrónimo del inglés ***Statistical-Algorithmic Method for Bicluster Analysis***

SPC Acrónimo del inglés ***Super Paramagnetic Clustering***

UCLV Acrónimo de *Universidad Central "Marta Abreu" de Las Villas*

UML Acrónimo del inglés ***Unified Modeling Language***

XMFA Acrónimo del inglés ***eXtended Multi-FastA***

INTRODUCCIÓN

Cada día la ciencia de la computación se hace partícipe de nuestra vida cotidiana, desde las aplicaciones comunes para el trabajo diario, hasta el impulso de novedosas investigaciones en el hallazgo y análisis de secuencias moleculares de diversas enfermedades, y el desarrollo de fármacos efectivos para combatirlas. Conocida como Bioinformática, en esta confluyen disciplinas tales como la biología, computación y tecnología de la información, para facilitar la solución de problemas, el análisis de datos, simulación de sistemas o mecanismos, todos ellos de índole biológico, usualmente en el nivel molecular, pero no de forma exclusiva (Altman, 2006). Los teóricos de este campo afirman que las técnicas usadas en la bioinformática se basan en la utilización de recursos computacionales, para solucionar o investigar problemas sobre escalas de tal magnitud que sobrepasan el discernimiento humano (Werner, 2005).

La investigación en la biología computacional se solapa a menudo con la biología de sistemas. En ella los principales esfuerzos incluyen el alineamiento de secuencias, la predicción de genes, el alineamiento estructural de proteínas, la predicción de estructuras de proteínas, así como el estudio de las interacciones proteína-proteína, entre otras (Kanehisa and Bork, 2003).Las interacciones entre proteínas desempeñan un papel fundamental en varios aspectos de la organización estructural y funcional de la célula. El análisis detallado, a nivel atómico, de la relación entre proteínas se realiza a partir de datos de complejos macromoleculares y su reconocimiento está determinado por las propiedades físicas y químicas, las cuales han sido caracterizadas en términos de su geometría y de su naturaleza química.

La detección de genes ortólogos es un área dentro de la Bioinformática que aunque ha sido ampliamente trabajada, aún requiere de algoritmos más precisos (Gabaldón et al., 2009) por su valor para la predicción de la función de las proteínas. Este problema se presenta en la bibliografía como un problema de aprendizaje no supervisado, específicamente, como un problema de agrupamiento de genes ortólogos pertenecientes a genomas que se quieran comparar.

Los genes ortólogos son aquellos que evolucionan a partir de un ancestro común en un proceso de especiación. Tienen como contraparte los genes parálogos que también son semejantes en su secuencia pero son resultado de la duplicación. Los genes homólogos divergen en cuanto a su función. Es tarea de la detección de ortólogos distinguir los genes que son ortólogos a partir de los homólogos en cuanto a la secuencia. Otros genes

son ortólogos ya que sus productos proteicos preservan su función aunque no conservan la similitud de la secuencia.

Los algoritmos de detección de ortólogos como *PhyOP* (Goodstadt and Ponting, 2006) se basan en el enfoque de árbol filogenético para la clasificación de los genes. Inicialmente agrupan los homólogos, generan los alineamientos múltiples correctos para cada grupo de dominio de homólogos y construyen el árbol filogenético para cada grupo. Finalmente extraen los ortólogos de estos árboles. Los métodos de árbol típicamente reconcilian el árbol de genes con el de especies para distinguir los nodos de duplicación y especiación.

Una alternativa al enfoque de árbol filogenético es el enfoque basado en grafo o de comparación par a par de genes. Estos algoritmos parten de construir un grafo bipartido con la similitud par a par de secuencias de los genes de dos genomas en comparación y luego aplican heurísticas como *RBH* (Tatusov et al., 1997) o *RSD* (Wall et al., 2003) para podar el grafo antes del agrupamiento.

Diversas bases de datos han sido construidas a partir de la predicción basada en grafo como por ejemplo: *NCBI KOG* (Tatusov, 2003), *INPARANOID* (O'Brien et al., 2005), *OrthoMCL-DB* (Chen et al., 2006), *MultiParanoid* (Alexeyenko et al., 2006), *EGO* (Lee et al., 2002), *COG* (Tatusov, 2003) y más recientemente *INPARANOID 7.0* (Östlund et al., 2010).

Existen otros algoritmos como *SOAR* (Chen et al., 2005), *MSOAR* (Fu et al., 2007) y *Multi-BUS* (Rasmussen and Kellis, 2005) que además de tener en cuenta la similitud entre las secuencias, tienen en cuenta los reordenamientos globales de genes y los bloques de genes que conservan el orden para estimar la distancia evolutiva. Enfoques híbridos (Towfic et al., 2009) han explorado las propiedades de la red de interacción de proteínas en combinación con la similitud de las secuencias.

El algoritmo *SOAR* (Chen et al., 2005) comienza identificando las familias de genes homólogos y la correspondencia entre estas familias en los dos genomas comparados realizando una búsqueda de homología. Las familias son tratadas como copias de los genes y la asignación de ortólogos se formula como el problema de optimización, el cual consiste en reordenar un genoma como una secuencia de genes posiblemente duplicados en el otro genoma con el número mínimo de inversiones. Dicho problema consiste específicamente en calcular la distancia de inversión con signo y con duplicados entre los dos genomas. *SOAR* sólo puede procesar genomas de un cromosoma y no tiene en cuenta los genes parálogos, es decir, asume que no hubo duplicaciones de genes después de la especiación (Fu et al., 2007). Devuelve las relaciones uno a uno de ortólogos y

todos los genes son forzados a formar pares de ortólogos.

En cambio *MSOAR* (Fu et al., 2007) se basa en un enfoque de máxima parsimonia que combina los eventos de reordenamiento incluyendo inversiones, translocaciones, fusiones de genes y fisiones de genes con los eventos de duplicación. En este caso se minimiza el número de eventos calculando la distancia de reordenamiento-duplicación. *MSOAR* a diferencia de *SOAR* procesa genomas con múltiples cromosomas y realiza mejoras a las técnicas empleadas en SOAR. Incorpora un nuevo paso de post-procesamiento para eliminar pares de genes con "ruido" que son más propensos a constituir parálogos.

Antecedentes

En el Laboratorio de Bioinformática de la *UCLV* se desarrolló el trabajo de diploma "Herramientas Computacionales para la Comparación de Genomas" (Estopiñales, 2009) donde se define una nueva distancia local-global para la comparación de genes entre dos genomas cercanos en la evolución. En este trabajó se usó el enfoque de grafo construyendo el grafo bipartido a partir del alineamiento par a par de secuencias, empleando el algoritmo de *Needleman–Wunsch* (Needleman and Wunsch, 1970) implementado en *MATLAB 7.4.* Sobre la base del esquema del algoritmo *BUS* (Kamvysselis, 2003) se combinó la similaridad de secuencias con la información de los bloques de orden conservado y se aplicaron políticas de poda (Kamvysselis, 2003) al grafo. Los experimentos con el cromosoma 5 de *Saccharomyces Cerevisiae* y el genoma completo de *Saccharomyces Bayanus* arrojaron resultados prometedores.

Con la nueva distancia local-global y utilizando la posibilidad de combinar rasgos, se implementó un algoritmo paralelo en el *MATLAB 9.0*, para la comparación par a par de genes empleando el algoritmo de *Needleman–Wunsch* y las estructuras algebraicas del código genético definidas en el Laboratorio de Bioinformática de la *UCLV* (Sánchez, 2006). Los experimentos realizados con el genoma *Saccharomyces Cerevisiae* (Goffeau, 1996) y el genoma *Schizosaccharomyces Pombe* (Wood, 2002) se validaron con la lista de ortólogos curada manualmente de *SANGER* (Wood, 2006), sugiriéndose una mejora en cuanto a la fase de poda y agrupamiento del algoritmo general de detección de ortólogos.

Tomando como referencia el grafo bipartido de los experimentos realizados con el genoma *Saccharomyces Cerevisiae* y el genoma *Schizosaccharomyces Pombe* se desarrolló el trabajo de diploma "Agrupamiento en grafos bipartitos y optimización de parámetros basada en enjambre de partículas, aplicación en la detección de genes

ortólogos" (Labrada, 2011) donde se realizó un estudio de diferentes técnicas de agrupamiento y co-agrupamiento sobre grafos bipartidos. El grafo bipartido se podó por umbral y se empleó el algoritmo de particionamiento multinivel *METIS* (Karypis and Kumar, 1998b) a la fase de agrupamiento del algoritmo de detección de ortólogos. Los resultados se validaron con la lista de ortólogos curada manualmente de *SANGER*, obteniéndose mejores resultados de calidad a los obtenidos con el algoritmo *BUS* (Kamvysselis, 2003), pero aún es necesario mejorar la fase de agrupamiento.

A partir de los resultados obtenidos y siguiendo la tendencia de combinar rasgos de similitud de las secuencias se desarrolló la tesis de maestría "Herramientas computacionales para la comparación de genomas y detección de genes ortólogos con un enfoque de grafo bipartido" (Fernández, 2012), donde se combinaron medidas de distancia basadas en la información del alineamiento local de secuencias, la longitud de las secuencias, la relación evolutiva según el modelo de las cinco bases (Sánchez and Grau, 2009) y la pertenencia a regiones conservadas teniendo en cuenta los reordenamientos globales en los genomas. Las diferentes distancias fueron combinadas usando el operador *OWA* (Lindahl, 2008) y la media aritmética. Aplicando el algoritmo de agrupamiento *BUS*, y comparando con la lista de ortólogos del algoritmo *INPARANOID* (Östlund et al., 2010) para los genomas del *Saccharomyces Cervisiae* y *Schizosaccharomyces Pombe*, se logró obtener un $85,24\,\%$ de clasificación. El alto número de falsos positivos obtenidos sugiere una mejora en el algoritmo de agrupamiento y en los criterios usados para la selección de los pares de genes ortólogos. De todo lo planteado anteriormente surge la necesidad de realizar este trabajo de diploma.

Planteamiento del problema

Es preciso elevar la exactitud de los algoritmos de detección de genes ortólogos.

Hipótesis de investigación

Si se combinan las medidas de similitud de las secuencias con medidas de comparación del perfil físico-químico de las proteínas para conformar el grafo bipartido, y se le aplica un algoritmo de agrupamiento a este grafo entonces se mejora la exactitud de los algoritmos de detección de genes ortólogos.

Objetivo general

Desarrollar un algoritmo basado en grafos que combine las medidas de similitud de las secuencias con medidas de comparación del perfil físico-químico de las proteínas y utilice un algoritmo de agrupamiento para la detección de genes ortólogos.

Objetivos específicos

1. Realizar un estudio crítico de las medidas de similitud de las secuencias y los algoritmos de detección de genes ortólogos basados en grafo.
2. Diseñar un algoritmo que combine las medidas de similitud y use un algoritmo de agrupamiento para la detección de ortólogos.
3. Implementar el algoritmo para la detección de genes ortólogos basado en grafo.
4. Validar los resultados del algoritmo con los datos de *Saccharomyces Cerevisiae* y *Schizosaccharomyces Pombe* utilizando como referencia la lista de ortólogos obtenida por el algoritmo *INPARANOID 7.0*.

Justificación

El presente trabajo es parte de los proyectos de investigación del Laboratorio de Bioinformática de la *UCLV*:

1. **Título del proyecto:** Métodos estadísticos y de Inteligencia Artificial para el análisis de secuencias de ADN
   Jefe del Proyecto: Dra. Gladys Casas Cardoso
2. **Título del proyecto:** Desarrollo y Aplicación de la Arquitectura de los genomas mediante la aplicación de las estructuras algebraicas
   Jefe del Proyecto: Dr. Robersy Sánchez Rodríguez

Además el trabajo que se propone forma parte de las investigaciones para optar por el grado de doctor en ciencias de la tutora, en el tema aprobado por el CITMA en marzo del 2010 “Métodos Computacionales para la Comparación de Genomas e Identificación de Genes Ortólogos”.

Dados los objetivos anteriores la tesis se divide en tres capítulos. Primeramente en el capítulo 1 “Marco teórico”, se hace un estudio de los algoritmos de detección de genes ortólogos, las medidas de similitud de secuencias, los algoritmos de agrupamiento sobre grafos y las medidas de validación de agrupamientos. En el capítulo 2 “Diseño e implementación”, se analiza el procesamiento de los datos de las especies a comparar y la implementación de las medidas de similitud empleadas, así como las medidas de validación. En el capítulo 3 “Experimentación y resultados”, se analizan y validan los resultados obtenidos luego de aplicar el algoritmo de agrupamiento al grafo bipartido formado por las medidas de similitud.

CAPÍTULO 1

MARCO TEÓRICO

El presente capítulo, abarca elementos relacionados con la homología de secuencias y los conceptos de ortología y paralogía. Se hace un estudio de los algoritmos de detección de genes ortólogos con un enfoque de grafo, y se analiza los aspectos relacionados con las medidas de similitud entre secuencias empleadas por estos algoritmos. Además se incluye un estudio sobre los algoritmos de agrupamientos sobre grafos, y las medidas de validación de agrupamientos, haciendo énfasis en en las medidas de validación externas.

1.1. Comparación de genomas

La genómica comparativa se dedica el estudio de las semejanzas y diferencias entre genomas de diferentes organismos. Se aprovecha tanto de las similitudes como de las diferencias entre las proteínas y regiones reguladoras de diferentes organismos para inferir cómo la selección natural ha actuado sobre los genomas. Durante la evolución de los genomas se producen cambios genéticos, que son clasificados en nivel bajo y alto. En el nivel bajo se producen las mutaciones puntuales que afectan a los aminoácidos, mientras que en el nivel alto se producen cambios entre segmentos como pueden ser la duplicación, transferencia horizontal, inversión y transposición.

La homología de secuencias se refiere a las secuencias de dos o más proteínas que son similares entre sí, debido a que presentan un mismo origen evolutivo. La homología no es un criterio medible, las secuencias son homólogas o no lo son, aunque usualmente se asume que dos secuencias son homólogas si estas dos presentan un alto grado de similitud[1] (Webber and Chris, 2004). Un alto grado de similitud entre dos secuencias puede estar dado simplemente al azar, como sucede en ocasiones con secuencias de poco tamaño.

[1]Para secuencias de proteínas los valores mínimos más usados están entre un 30 % y un 35 % de similitud

Dentro de la homología de secuencia se distinguen dos tipos de homología: la ortología y la paralogía. Los genes ortólogos son genes homólogos de especies diferentes que evolucionaron a partir de un ancestro común en un proceso de especiación, mientras que los parálogos son resultado de la duplicación. Los genes ortólogos proveen información útil en estudios de taxonomía, estudios filogenéticos y estudios de las funciones conservadas de los genes entre los genomas.

1.2. Definición operacional de ortología

La definición operacional de ortología en muchas ocasiones es usada como el primer paso para encontrar posibles ortólogos, teniendo en cuenta algunas de las mejores coincidencias entre genomas completos de dos especies. Estos métodos son conocidos como "vecino más cercano". Se usan para podar el grafo y luego aplicar técnicas de particionado. El algoritmo de comparación más usado por estos métodos es el *BLAST* (Altschul et al., 1990), a pesar de que puede devolver una secuencia con el mayor grado de similitud que no es la más cercana en el árbol filogenético de las secuencias comparadas (Koski and Golding, 2001).

El método *RBH* busca las mejores correspondencias recíprocas de todas las secuencias de un genoma en otro[2] (Tatusov et al., 1997). Sean las secuencias X y Y pertenecientes a los genomas G_1 y G_2 respectivamente, se dice que X y Y son *RBH* si Y es la secuencia con mayor similitud al comparar X con el genoma G_2 y X es la secuencia con mayor similitud al comparar Y con el genoma G_1 (Johnson, 2008). El método es vulnerable a excluir genes ortólogos debido a la identificación de genes parálogos en una de las direcciones de comparación del algoritmo *BLAST* (Wall et al., 2003).

Un método que corrige la principal deficiencia del *RBH* es el método *RSD* (Wall et al., 2003). En un primer momento se obtiene el conjunto H de los grados de semejanza, obtenidos de comparar la secuencia X con todas las secuencias Y del genoma G_2, tomando solo las que superan un umbral preestablecido[3]. Luego usando *ClustalW*[4] (Thompson et al., 2000) cada secuencia Y es alineada con la secuencia X, si poseen un grado de semejanza en H. Cuando la región alineable de Y y X supera una fracción de la longitud total del alineamiento[5], se calcula el número de sustituciones de aminoácidos que separa las secuencias (Jones et al., 1992). En un segundo momento el método *RSD* calcula la distancia recíproca, de forma análoga al primer momento. Si las secuencias X

[2]El método puede ser generalizado para n genomas
[3]El autor del artículo define como ejemplo $\varepsilon < 10^{-20}$
[4]Herramienta para el alineamiento múltiple de secuencias
[5]El autor en su artículo define el umbral como 0.8 de la longitud total del alineamiento

y Y tienen un grado de semejanza y el menor valor de distancia en las dos direcciones de búsqueda, entonces X y Y son *RSD* y son considerados ortólogos.

El método *BBH* es el método más usado para determinar posibles pares de genes ortólogos (Hulsen et al., 2006). Dos secuencias X y Y pertenecientes a los genomas G_1 y G_2 son llamadas *BBH* si existe un valor de similitud $S_{X,Y}$ significativo[6] entre ellas, además no existe una secuencia Z del genoma G_2 cuyo valor de similitud con X sea superior a $S_{X,Y}$ y no existe una secuencia W del genoma G_1 cuyo valor de similitud con Y sea superior a $S_{X,Y}$ (Overbeek et al., 1999). De esta forma, no se dan garantía de que las mejores coincidencias uno a uno representen relaciones de pares de genes ortólogos, estableciéndose incorrectas relaciones entre pares de genes (Kamvysselis, 2003).

Basados en el método *BBH* se define el método *PCBBH* basados en bloques de pares genes cercanos. Dos pares de genes *BBH* (X, Y) y (W, Z) se definen como *PCBBH* si las secuencias X y W se encuentran sobre la misma hebra[7] y el número de pares de bases que las separan no supera los 300 pares de bases; y las secuencias Y y Z se encuentran sobre la misma hebra y el número de pares de bases que las separan no supera los 300 pares de bases (Overbeek et al., 1999).

1.3. Algoritmos de detección de genes ortólogos

La detección de genes ortólogos frecuentemente está relacionada con los árboles filogenéticos. La construcción de los árboles filogenéticos requiere de la generación de alineamientos múltiples para cada grupo de dominios homólogos, el cual es un proceso que demanda gran cantidad de recursos computacionales (Remm et al., 2001). Los métodos basados en grafos, a diferencia de estos no necesitan generar alineamientos múltiples, por lo que son más computacionalmente ligeros y más sencillos de implementar.

Los algoritmos de detección de genes ortólogos basados en grafos se basan en la similitud entre pares de secuencias. Inicialmente hacen un filtrado de los datos, en el que generalmente eliminan secuencias muy pequeñas, para luego calcular las comparaciones par a par de las secuencias usando *BLAST* (Altschul et al., 1990). La similitud entre pares de secuencias se define a partir de la puntuación obtenida en cada comparación o por su porciento de similitud (Remm et al., 2001). Una vez conformado el grafo basado en la similitud de las secuencias se aplica alguna definición operacional de ortología. Las

[6] El autor define la similitud significativa cuando las puntuaciones obtenidas por el algoritmo *FASTA3* son menores que 10^{-5}

[7] En inglés se conoce como ***strand***

definiciones de ortología más usadas son *RBH* y *BBH*. Luego de determinar los posibles pares de genes ortólogos se aplica un algoritmo de agrupamiento sobre el grafo para conformar los grupos de pares de genes (Li et al., 2003), y a partir de estos determinar mediante diversos criterios los pares de genes ortólogos.

1.3.1. Algoritmo INPARANOID

El algoritmo *INPARANOID* permite clasificar pares de genes de dos especies en ortólogos o in-parálogos. El algoritmo toma como entrada inicial los archivos en formato *FASTA* de los dos genomas G_1 y G_2. Halla las correspondencias par a par entre los pares de genomas obtenidos de las permutaciones de los genomas G_1 y G_2, o sea, G_1 con G_1, G_1 con G_2, G_2 con G_1 y G_2 con G_2. Todos los pares de correspondencia cuya puntuación no supera los 50 bits y la región alineada no rebasa el 50 % de las longitudes de las secuencias, son podados. Las puntuaciones obtenidas para los pares de secuencias (X, Y) y (Y, X) pueden ser asimétricas, por lo que sus valores de puntuación son promediados (Remm et al., 2001).

Luego de podar las correspondencias entre pares de genes se buscan todos los pares (X, Y) *BBH* entre el genoma G_1 y G_2, los cuales son marcados como posibles genes ortólogos. Las correspondencias del genoma G_1 con G_1 y G_2 con G_2 cuya puntuación es superior a la puntuación del par *BBH* son marcadas como posibles in-parálogos. A cada in-parálogo se le calcula un valor de confianza para determinar cuán semejante es respecto a cada par de genes marcado como ortólogos. Basado en este valor de confianza se resuelven los problemas de solapamiento de grupos de ortólogos (Remm et al., 2001).

1.3.2. Algoritmo OrthoMCL

Al algoritmo *OrthoMCL* al igual que la mayoría de los algoritmos inicia buscando las correspondencias entre todos los pares de secuencias del genoma G_1 con las secuencias del genoma G_2, usando el algoritmo *BLAST* (Altschul et al., 1990). Las correspondencias *RBH* son marcadas como posibles relaciones de pares de genes ortólogos. Para cada una de estas correspondencias marcadas como posibles ortólogos, se buscan los posibles parálogos, como las secuencias de un mismo genoma para las cuales su mejor grado de similitud se encuentra con una secuencia de ese mismo genoma y no con una secuencia del otro genoma en comparación.

Una vez seleccionados los posibles genes ortólogos y parálogos, se construye un grafo,

donde los nodos representan las secuencias de proteínas de cada uno de los genomas, y todos los posibles pares de ortólogos y parálogos se enlazan por una arista pesada por el grado de la relación. Los pesos iniciales de las aristas entre cada par de secuencias se calculan como $promedio_{score} - log_{10}(P\_value^{8})$, donde $promedio_{score}$ es el promedio de las puntuaciones obtenidas por el *BLAST* para cada par de secuencias. Debido al alto grado de similitud de los genes parálogos respecto a los ortólogos, el agrupamiento pueder ser polarizado en un sólo sentido, por lo que los pesos de las aristas son normalizados (Li et al., 2003).

El grafo resultante se representa como una matriz simétrica a la cual se le aplica el algoritmo de agrupamiento *MCL* (Enright et al., 2002). Un parámetro importante del algoritmo *MCL* es el valor de inflación, el que refleja cuán compactos deben ser los grupos obtenidos. *OrthoMCL* define el valor de inflación como 1,5. El algoritmo *MCL* es analizado con mayor profundidad en la sección 1.6.

OrthoMCL produce resultados similares a los obtenidos por el algoritmo de *INPARANOID* (O'Brien et al., 2005) cuando se comparan dos genomas, y además brinda la posibilidad de comparar más de dos genomas. La identificación de grupos con *OrthoMCL* a través de múltiples genomas revela patrones filogenéticos entre proteínas de diferentes familias. Los grupos obtenidos son coherentes con los grupos obtenidos por *EGO* (Lee et al., 2002), extendiendo estos con la adición de los parálogos recientes. Esta estrategia es útil para el análisis de genomas eucariotas dónde la duplicación de genes puede evitar la identificación de pares de genes *RBH* (Li et al., 2003).

1.3.3. Algoritmo BUS

El algoritmo de detección de genes ortólogos *BUS* se basa en las debilidades del método *BBH* (Overbeek et al., 1999) y la base de datos *COG*[9] (Tatusov, 2003) para resolver los problemas de correspondencia entre las especies (Kamvysselis, 2003). Representa la mejor coincidencia de cada gen como un conjunto de genes, en vez de una sola correspondencia, lo que le permite ser más robusto sobre la presencia de ligeras diferencias en la similitud de las secuencias (Kamvysselis, 2003). El algoritmo *BUS* proporciona una buena solución para determinar las correspondencias de genes ortólogos entre dos genomas, trabajando en un rango de distancias evolutivas (Kamvysselis, 2003).

[8] *P_value* es un valor de corte definido por el algoritmo *BLAST*, y basado en estudios empíricos el autor lo define como 10^{-5}

[9] Base de datos de grupos de ortólogos, que no tiene en cuenta la duplicación de genes

Inicialmente se construye el grafo bipartido, donde los nodos representan los genes de los genomas comparados. Todos los nodos que representan genes de un genoma G_1 se enlazan por una arista pesada con todos los nodos que representan genes del genoma G_2. El peso de cada arista se define como la similitud del alineamiento entre el par de secuencias que representa cada gen. Luego de construido el grafo, se podan todas las aristas de un nodo, cuyo peso es inferior al 80 % del peso máximo de las aristas que conectan este nodo. El algoritmo usa la información de los bloques de orden conservado[10] para eliminar ambigüedades adicionales en el grafo bipartido (Kamvysselis, 2003).

El grafo es dividido en subgrafos aproximadamente del mismo tamaño, de manera que existan pocas conexiones en cada subgrafo. Las conexiones dentro de cada subgrafo conectan los verdaderos pares de genes ortólogos. Busca los subconjuntos de genes que son óptimos a nivel local, de forma que las mejores coincidencias de los genes dentro de cada subgrafo están contenidas dentro del subconjunto. Los mejores subconjunto no ambiguos obtenidos aseguran que el grafo es separable máximamente, a la vez que mantiene todas las posibles relaciones uno a uno de ortólogos (Kamvysselis, 2003).

1.4. Medidas de similitud

La similitud de secuencias es tomada como base para la clasificación de genes o proteínas. La relación entre dos genes X y Y, puede estar dada por una medida de similitud $S_{X,Y}$, o una medida de distancia $D_{X,Y}$. Similitud y distancia son dos conceptos totalmente opuestos, la similitud entre dos genes refleja cuánto se parecen los genes X y Y, y la distancia cuánto se diferencian uno del otro. Las medidas de distancia deben garantizar que se cumplan las propiedades básicas de las distancias: no negatividad, identidad, simetría y desigualdad triangular (Deza and Deza, 2006).

$$D_{X,Y} \geq 0 \tag{1.1}$$

$$D_{X,Y} = 0 \Longleftrightarrow X = Y \tag{1.2}$$

$$D_{X,Y} = D_{Y,X} \tag{1.3}$$

$$D_{X,Y} + D_{Y,Z} \geq D_{X,Z} \tag{1.4}$$

La similitud entre genes puede ser convertida en una medida de distancia y viceversa, como se muestra en las ecuaciones 1.5 (Deza and Deza, 2006). Para que una medida de

[10] En inglés se conocen como ***synteny blocks***

similitud pueda ser convertida en una medida de distancia debe cumplir las propiedades de no negatividad, identidad, simetría, debe cumplirse que $S_{X,Y} \leq S_{X,X}$ para todos los valores de X y Y, con igualdad sólo cuando $X = Y$, y $S_{X,Y} \in [0, 1]$.

$$
\begin{aligned}
D_{X,Y} &= 1 - S_{X,Y} \\
D_{X,Y} &= \sqrt{1 - S_{X,Y}} \\
D_{X,Y} &= \sqrt{2 \times (1 - S^2_{X,Y})} \\
D_{X,Y} &= -ln(S_{X,Y}) \\
D_{X,Y} &= arccos(S_{X,Y})
\end{aligned} \tag{1.5}
$$

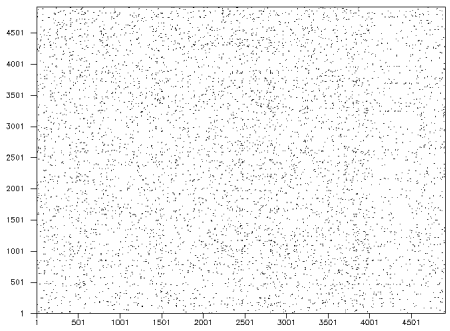
Una de las primeras medidas de relación entre dos secuencias es la distancia de edición (Levenshtein, 1965) entre dos cadenas o secuencias X y Y, $D_{X,Y}$ se define como el menor costo de insertar, eliminar o reemplazar caracteres que transformen la secuencia X en la secuencia Y.

1.4.1. Medida basada en el alineamiento de secuencias

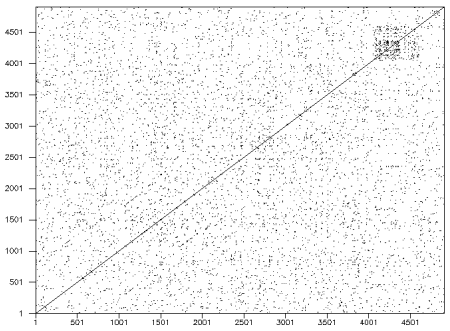
La distancia de edición está directamente relacionada al problema de alineamiento de secuencias. Ambos problemas en esencia son equivalentes pero ofrecen diferentes formas de ver la relación entre dos secuencias. El alineamiento de secuencias es una forma de representar y comparar dos o más secuencias para resaltar las regiones de mayor similitud, que podrían indicar relaciones funcionales o evolutivas entre los genes o proteínas comparadas. Puede analizar cambios genéticos a pequeña escala como son la inserción, eliminación o sustitución de aminoácidos, mientras que cambios como: la inversión, duplicación o reordenamiento de aminoácidos, no pueden ser analizadas a través del alineamiento de secuencias (Kamvysselis, 2003). Algunos de los métodos de alineamiento de pares de secuencias son: las matrices de puntos, los algoritmos de programación dinámica y los métodos de las k-tuplas (Mount, 2004b).

El alineamiento de secuencias basado en una matriz de puntos es uno de los métodos de alineamiento más sencillos. Produce una familia de alineamientos para regiones individuales de las secuencias, los que son analizados visualmente a través de una gráfica, para identificar determinadas características de las secuencias, como pueden ser las inserciones, eliminaciones, repeticiones y repeticiones invertidas de aminoácidos (Gibbs and McIntyre, 1970). Algunas implementaciones varían el tamaño o la intensidad de los puntos en función del grado de similitud de los dos aminoácidos,

para tener en cuenta las sustituciones conservadas (Maizel and Lenk, 1981). El método de la matriz de puntos también pueden utilizarse sobre una sola secuencia, y las regiones que comparten similitudes significativas aparecen en la gráfica como líneas fuera de la diagonal principal, como se muestra en la figura 1.1.



(a) Matriz de puntos obtenida para el alineamiento de la proteína *YLR106C* del *Saccharomyces Cerevisiae* con la proteína *SPAC23G3.02c* del *Schizosaccharomyces Pombe*



(b) Matriz de puntos obtenida del alineamiento de la proteína *YLR106C* del *Saccharomyces Cerevisiae* con la propia proteína *YLR106C*

Figura 1.1.: Ejemplos de alineamientos basados en matrices de puntos, generados con DOTTUP

Cuando se comparan grandes cantidades de secuencias, los métodos de alineamiento basados en matriz de puntos dejan de ser efectivos, por lo que se necesita recurrir a otros métodos no gráficos que permitan asignar un valor numérico a los alineamientos. Uno de los métodos más usados es la programación dinámica, que permite calcular el alineamiento óptimo entre pares de secuencias, asignando un valor de puntuación[11] para el alineamiento obtenido. El alineamiento calculado brinda información útil para establecer interrelaciones entre las secuencias y hacer predicciones funcionales, estructurales y evolutivas de las secuencias (Mount, 2004b).

Los algoritmos de alineamiento basados en programación dinámica usan las matrices de sustitución (Durbin et al., 1998), para asignar un valor de penalización diferente para cada sustitución de un aminoácido por otro, o por él mismo; y una función $\gamma(\alpha)$ de penalización para las inserciones y eliminaciones de aminoácidos en las secuencias. Para representar las inserciones o eliminaciones en el alineamiento de dos secuencias son introducidos pequeños saltos o *"gaps"*[12], que representan una eliminación en una de las secuencias, o una inserción en la otra secuencia. Las matrices de sustitución y las funciones de penalización son analizadas en la sección 1.4.1.3. Sean las secuencias $X = (X_1, X_2, \ldots, X_n)$ y $Y = (Y_1, Y_2, \ldots, Y_m)$, $s(X_i, Y_j)$ el valor asignado por la matriz de sustitución para el reemplazo del aminoácido X_i por Y_j y $\gamma(\alpha)$ la función de

[11] En inglés es conocido como ***score***

[12] Los *"gaps"* o huecos son representados por el guión "-"

penalización para α inserciones o eliminaciones, entonces, la matriz de programación dinámica F es generada desde $F(0,0)$ en la esquina superior izquierda, hasta la esquina inferior derecha $F(n,m)$ de forma recursiva, como se muestra en la figura 1.2. El alineamiento óptimo es calculado a partir de la posición $F(n,m)$, donde se encuentra la puntuación óptima del alineamiento, haciendo una búsqueda hacia atrás en la matriz F hasta la posición $F(0,0)$.

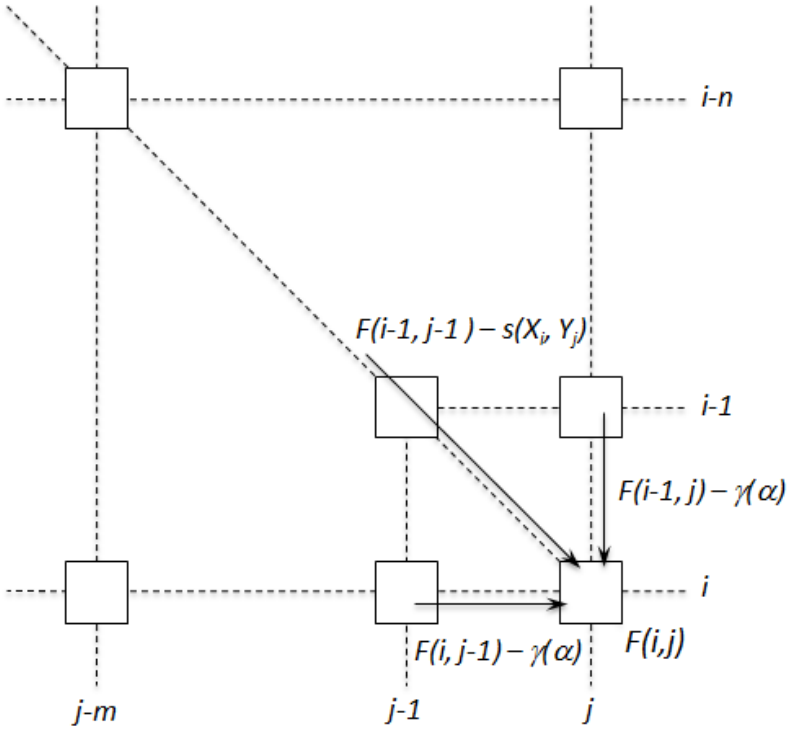


Figura 1.2.: Descripción gráfica del algoritmo de programación dinámica

$$F(i,j) = max \left\{ \begin{array}{l} F(i-1,j-1) + s(X_i, Y_j), \\ F(i-1,j) - \gamma(\alpha), \\ F(i,j-1) - \gamma(\alpha) \end{array} \right\}, \begin{array}{l} \forall i \in \{1,2,\ldots,n\} \\ \forall j \in \{1,2,\ldots,m\} \end{array} \tag{1.6}$$

Los algoritmos de alineamiento de secuencias basados en programación dinámica son clasificados en globales o locales, en dependencia de la porción de las secuencias que abarca el alineamiento. El alineamiento global busca la mejor correspondencia de las secuencias en su totalidad, mientras el local busca la mejor correspondencia de una subcadena en las secuencias. Usualmente el alineamiento local es más significativo que el global, ya que identifica patrones conservados entre las secuencias comparadas, con una cantidad mínima de *"gaps"* (Mount, 2004b). Calcular un alineamiento global o local, no solo depende del algoritmo empleado, también está estrechamente relacionado con la combinación de matriz de sustitución y valores de penalización empleados

(Mount, 2004b). La combinación de estos valores es analizada en la sección 1.4.1.3. El alineamiento global puede producir un alineamiento incorrecto con grandes cantidades de *"gaps"* cuando la diferencia entre las longitudes de las secuencias es alta, por lo que se recomienda emplearlo cuando el grado de similitud entre las secuencias comparadas es medio o alto y la diferencia entre las longitudes de las secuencias es pequeña, en otros casos es recomendable emplear algoritmos de alineamiento local (Keith et al., 2008).

Por otra parte los métodos de las k-tuplas son métodos heurísticos que no garantizan encontrar una solución óptima del alineamiento de secuencias, pero son significativamente más eficientes que los algoritmos de programación dinámica. Por lo general en estos métodos el usuario puede definir el valor de k para definir la longitud de la palabra con la cual buscar en la base de datos. Valores pequeños de k producen una solución más sensible pero más lenta y son preferibles para búsquedas que impliquen secuencias muy cortas (Mount, 2004a).

Herramientas de búsquedas sobre bases de datos de secuencias como *FASTA* (Lipman and Pearson, 1985) y *BLAST* (Altschul et al., 1990) se basan en estos métodos ya que son especialmente útiles en búsquedas sobre bases de datos de secuencias a gran escala, donde se asume que una gran cantidad de secuencias no tendrán coincidencias significativas con la secuencia empleada para buscar. Estos métodos identifican en la secuencia que se emplea para buscar, una serie de subsecuencias cortas, de longitud k, que son usadas para buscar las secuencias de la base de datos que tiene coincidencias más significativas con estas subsecuencias (Mount, 2004a).

1.4.1.1. Algoritmo de Needleman-Wunsch

El algoritmo de *Needleman-Wunsch* (Needleman and Wunsch, 1970) es un algoritmo de programación dinámica para hallar el alineamiento global de dos secuencias. La idea esencial del algoritmo es ir construyendo el alineamiento óptimo de las secuencias a partir del alineamiento óptimo de subsecuencias más pequeñas. La figura 1.3 muestra el alineamiento global de dos secuencias de proteínas, donde se obtuvo un 28 % de identidad y un 40 % de similitud entre las proteínas comparadas.

El algoritmo se basa en una matriz de programación dinámica F, indexada por i y j, un índice para cada secuencia. El valor $F(i, j)$ representa la puntuación del mejor alineamiento del segmento inicial $X_{1...i}$ de la secuencia X con el segmento $Y_{1...j}$ de la secuencia Y. La matriz F es construida de forma recursiva por la ecuación 1.6. El valor óptimo del alineamiento para cada posición es calculado sumando el valor de

```
Identities = 11/40 (28%), Positives = 16/40 (40%)
MVKLTSIAAGVAAIA-ATASA-TTTL-AQSD--ER-----
||  : ||  ::  |   |:  |    :| |  ||
MVVASEIAKVASKTARDIAGCFTCQCGTQFDNVERIVQHF
```

Figura 1.3.: Alineamiento global de las secuencias de proteínas "*MVKLTSIAAGVAAIAA-TASATTTLAQSDER*" y "*MVVASEIAKVASKTARDIAGCFTCQCGTQFDNVE-RIVQHF*" usando el algoritmo de *Needleman-Wunsch*, con matriz de sustitución BLOSUM50 y penalización de *"gaps"* constante -8.

la sustitución correspondiente a las bases comparadas o restando la penalización por inserciones o eliminaciones, al valor óptimo calculado con anterioridad.

1.4.1.2. Algoritmo de Smith-Waterman

El algoritmo de *Smith-Waterman* (Smith and Waterman, 1981) es un algoritmo de programación dinámica para hallar el alineamiento local óptimo de dos secuencias. Se emplea para encontrar regiones conservadas entre secuencias cuando estas regiones son sólo una fracción de las longitudes de las secuencias, las secuencias tienen longitudes significativamente diferentes[13], se superponen, o cuando una secuencia es un fragmento o subsecuencia de la otra secuencia (Mount, 2004b). La figura 1.4 muestra el alineamiento local de dos secuencias de proteínas, para las cuales se obtuvo un 38 % de identidad y un 77 % de similitud, valores que son mayores a los obtenidos por el alineamiento global de estas secuencias, mostrado en la figura 1.3.

```
Identities = 5/13 (38%), Positives = 10/13 (77%)
IAAGVAAIAA-TA
:|: :| :|: ||
VASEIAKVASKTA
```

Figura 1.4.: Alineamiento local de las secuencias de proteínas "`MVKLTSIAAGVAAIAATASATTTLAQSDER`" y "*MVVASEIAKVASKTARDIAGCFTCQCGTQFDNVERIVQHF*" usando el algoritmo de *Smith-Waterman*, con matriz de sustitución *BLOSUM50* y penalización de *"gaps"* constante -8.

La esencia del algoritmo es similar al algoritmo de *Needleman-Wunsch*, pero a diferencia de este el menor valor que puede tomar $F(i, j)$ es 0 y el valor óptimo puede estar ubicado en cualquier posición de la matriz de programación dinámica, la cual se construye de forma recursiva transformando la ecuación 1.6 en 1.7. El alineamiento óptimo es calculado a partir de la posición $F(i, j)$ que contiene la mayor puntuación, haciendo

[13]Las longitudes son significativamente diferentes en dependencia del problema en que se aplica el algoritmo.

una búsqueda hacia atrás en la matriz F hasta encontrar una posición con valor cero.

$$F(i,j) = max \left\{ \begin{array}{l} 0, \\ F(i-1,j-1) + s(X_i, Y_j), \\ F(i-1,j) - \gamma(\alpha), \\ F(i,j-1) - \gamma(\alpha) \end{array} \right\}, \begin{array}{l} \forall i \in \{1,2,\ldots,n\} \\ \forall j \in \{1,2,\ldots,m\} \end{array} \tag{1.7}$$

1.4.1.3. Matrices de sustitución y penalización de *"gaps"*

En proteínas relacionadas de diferentes especies es común encontrar la sustitución de un aminoácido por otro, y a pesar de la sustitución las proteínas conservan su función. En estos casos en que las proteínas conservan su función, se dice que la sustitución de estos aminoácidos es compatible con la función y estructura de las proteínas, pero pueden ocurrir sustituciones en que no lo son. Las matrices de sustitución contienen valores de puntuación que reflejan como un aminoácido puede ser aparejado o sustituido por otro aminoácido en el alineamiento de secuencias.

Debido a que usualmente no se conoce el ancestro de un aminoácido en el árbol filogenético, se asume que la probabilidad de cambiar un aminoácido A por un aminoácido B, es la misma que la de cambiar un aminoácido B por un aminoácido A. Adicionalmente el valor de la puntuación también depende de la frecuencia de ocurrencia de los dos aminoácidos y de sus similitudes físicas y químicas. Algunos modelos plantean que la frecuencias de los aminoácidos no cambian durante la evolución (Dayhoff, 1978). Las matrices de sustitución más conocidas son las matrices de la familia *PAM* y las matrices de la familia *BLOSUM*.

Las matrices de la familia *PAM* se calculan observando las sustituciones compatibles en proteínas muy cercanas en la evolución, con al menos un 85 % de similitud (Dayhoff, 1978). Estas matrices se numeran desde 10 hasta 500, incrementando en 10, lo que significa el porciento de sustituciones esperadas en un período de tiempo, respecto a la matriz *PAM1* tomada como base para el cálculo del resto de las matrices. Una de las matrices más usada de esta familia es la *PAM250* y produce buenos alineamientos para secuencias distantes en la evolución cuya similitud se encuentra entre el 14 % y el 27 % (George et al., 1990). Otras matrices como la *PAM120*, *PAM80* y *PAM60* deben ser usadas para el alineamiento de secuencias que tiene un 40 %, 50 % y 60 % de similitud respectivamente.

A diferencia de las matrices de la familia *PAM* las *BLOSUM* se basan en alineamientos observados, y no son extrapoladas de comparaciones de proteínas cercanamente

relacionadas. La matrices *BLOSUM* surgen a partir del análisis de regiones conservadas de gran cantidad de familias de proteínas y la comprobación de las frecuencias relativas de aparición de los aminoácidos, así como las probabilidades de sustitución entre ellos (Henikoff and Henikoff, 1992). Las matrices se basan en el mínimo porcentaje de identidad de los alineamientos usados para calcularlas (Mount, 2004b). Estas matrices se numeran desde 30 hasta 90, incrementando en 5 e incluyendo la matriz *BLOSUM62*. Esta última es la más usada y ha sido tomada como estándar de muchos de los programas de alineamiento.

Existen importantes diferencias en la forma en que son calculadas las matrices *PAM* y *BLOSUM*, y esas diferencias indican como interpretar los resultados obtenidos del alineamiento usando estas matrices (Mount, 2004b). Las matrices *PAM* están diseñadas para tener en cuenta el camino evolutivo del origen de las secuencias, mientras que las matrices *BLOSUM* están diseñadas para encontrar los dominios conservados de las secuencias. *BLOSUM* ha demostrado actuar mejor en la puntuación de secuencias distantemente relacionadas que las matrices *PAM* (Mount, 2004b).

Por otra parte la penalización de las inserciones y eliminaciones, conocidas como *"gaps"*, es necesaria para obtener un buen alineamiento entre dos secuencias. Si se emplea un valor de penalización relativamente alto respecto a las puntuaciones de la matriz de sustitución entonces se obtendrá un alineamiento sin *"gaps"*, y por el contrario, si se usa un valor relativamente pequeño se obtendrá un alineamiento con gran cantidad de *"gaps"*, lo cual puede llevar a que siempre se obtenga un alineamiento global de las secuencias, aunque se emplee un algoritmo de alineamiento local (Mount, 2004b).

El modelo de penalización de *"gaps"* más usado es el modelo de penalización extendida[14], donde se le asigna un valor de penalización g para la primera inserción o eliminación, conocido como valor de penalización de abertura[15], y un valor de penalización r para las sucesivas inserciones o eliminaciones, conocido como valor de penalización extendida[16] (Mount, 2004b). La función de penalización $\gamma(\alpha)$ se define como:

$$\gamma(\alpha) = g + r \times (\alpha - 1) \tag{1.8}$$

Cuando se combina un algoritmo de alineamiento de programación dinámica con la penalización extendida, la ecuación 1.6, empleada para calcular la matriz $F(i, j)$ de programación dinámica, se transforma en la ecuación (Durbin et al., 1998):

[14]En inglés es conocido como ***affine gap penalty model***
[15]En inglés es conocido como ***gap open penalty value***
[16]En inglés es conocido como ***extended gap penalty value***

$$
\begin{aligned}
F(i,j) &= max \begin{cases} F(i-1,j-1) + s(X_i, Y_j), \\ G_x(i-1,j-1) + s(X_i, Y_j), \\ G_y(i-1,j-1) + s(X_i, Y_j) \end{cases} \\
G_x(i,j) &= max \begin{cases} F(i-1,j) - g, \\ G_x(i-1,j) - r \end{cases} \\
G_y(i,j) &= max \begin{cases} F(i,j-1) - g, \\ G_y(i,j-1) - r \end{cases}
\end{aligned} \tag{1.9}
$$

El modelo de penalización extendida penaliza con mayor fuerza la apertura de una nueva inserción o eliminación, no siendo así con la extensión, lo que significa que el valor de penalización de abertura debe ser mayor o igual que el valor de penalización extendida (Mount, 2004b). Una caso específico de este modelo es el modelo de penalización constante, donde $g = r$. En este caso la ecuación 1.8 del valor de penalización $\gamma(\alpha)$ se transforma en la ecuación:

$$\gamma(\alpha) = g \times (\alpha) \tag{1.10}$$

Este modelo es más sencillo de implementar que el modelo extendido, ya que mantiene la ecuación 1.6 para calcular la matriz F de programación dinámica. Existen otros modelos de penalización de *"gaps"* más complejos y con un menor uso (Miller and Myers, 1988).

Como se mencionó anteriormente los algoritmos de alineamiento local por si sólos no garantizan un alineamiento local, dependen de las matrices de sustitución y los valores de penalización empleados. Estudios de alineamientos de secuencias, sobre la combinación de matrices de sustitución y valores de penalización han permitido determinar valores aceptables[17] para obtener buenos alineamientos locales (Altschul and Gish, 1996). La combinación de estos parámetros se muestran en el anexo A.

1.4.2. Medida basada en la información de los LCB

La recombinación de genes puede causar cambios a gran escala como pueden ser pérdida, duplicación o reordenamiento de genes, lo cual puede provocar que las regiones de genes ortólogos puedan encontrarse reordenadas o invertidas respecto a un genoma

[17] El autor aclara que estos valores no son los valores óptimos, pero producen buenos alineamientos, determinado por los valores estadísticos K, λ y H.

en comparación. El *Mauve* (Darling et al., 2004) es un software que durante el proceso de alineamiento, identifica las regiones conservadas que aparentan no haber sido alteradas por el reordenamiento, las que son referidas como *LCB* (Darling et al., 2004). Los *LCB* definidos en el alineamiento final del *Mauve* son aquellas regiones consideradas verdaderamente homólogas (Darling et al., 2004), y genes que se encuentran en un mismo *LCB* son más propensos a ser ortólogos que los que están en diferentes *LCB*.

Para el cálculo de los *LCB* el *Mauve* tiene dos algoritmos: el algoritmo original del *Mauve* y el algoritmo progresivo(Darling et al., 2010). Ambos algoritmos se basan en la técnica de alineamiento anclado para un rápido alineamiento de los genomas comparados. El algoritmo original trabaja de forma óptima sobre genomas relativamente cercanos en la evolución, pero no alinea grandes regiones conservadas, el tamaño mínimo de los bloques *LCB* debe ser determinado de forma manual y presenta dificultades alineando genomas con gran cantidad de regiones duplicadas. Por otra parte el algoritmo progresivo a pesar de que es un poco más lento y consume más memoria que el algoritmo original, corrige todas las deficiencias del algoritmo original, y además permite el alineamiento de un número mayor de genomas, permitiendo alinear genomas más distantes en la evolución con menos de un 50 % de identidad.

1.4.3. Medida basada en el perfil físico-químico de las proteínas

En ocasiones secuencias de proteínas que tienen un alto grado de similitud funcional y estructural, la similitud de sus secuencias de aminoácidos no supera el 20 % o el 30 %. Estas secuencias se dicen que están en la zona límite[18] de la homología de secuencias, y en la mayoría de los casos son clasificadas de forma incorrecta (Carpio-Muñoz and Carbajal, 2002). El análisis de la periodicidad de las propiedades físico-químicas de los aminoácidos que constituyen la estructura primaria de las proteínas permite detectar la similitud en la dimensión espectral de las secuencias (Carpio-Muñoz and Yoshimori, 2002), por lo que debe mejorar el proceso de clasificación.

El cálculo del perfil físico-químico de las proteínas se basa en la transformación de la secuencia de aminoácidos en una representación espectral, la cual puede ser representada gráficamente por una línea de frecuencia. Esta representación espectral puede ser analizada basándose en *LPC* o usando la transformada rápida de *Fourier* (Deza and Deza, 2006). Una vez calculado el perfil de las secuencias, la comparación de dos secuencias se reduce al alineamiento de la representación espectral de la estructura primaria de las proteínas (Carpio-Muñoz and Carbajal, 2002).

[18]En inglés es conocida como ***twilight zone***

El primer paso es representar las secuencias de aminoácidos como una serie numérica, asignando a cada aminoácido alguna propiedad físico-química como puede ser: hidrofobicidad, energía de contacto, polaridad ionización, volumen, superficie, accesibilidad, etc. (Carpio-Muñoz and Yoshimori, 2002). La serie numérica obtenida que representa el perfil físico-químico se maneja como una onda, de forma similar a las señales digitales. En la figura 1.5 muestra la representación espectral del perfil físico-químico de la proteína *"YLR106C"* del *Saccharomyces Cerevisiae*, al sustituir cada uno de sus aminoácidos por su energía de contacto.

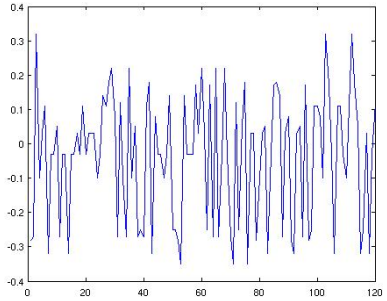


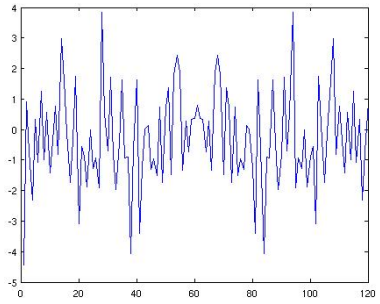
Figura 1.5.: Perfil físico-químico de la proteína *YLR106C* del *Saccharomyces Cerevisiae* usando la energía de contacto

El análisis basado en la transformada de *Fourier* transforma el perfil físico-químico en un ámbito de frecuencia, para modelar la relación entre la representación primaria y terciaria de la proteína. Sea $X = (X_1, X_2, \ldots, X_N)$ la representación del perfil físico-químico de una proteína de N aminoácidos, el *cepstrum* C del perfil se define aplicando la transformada rápida de *Fourier*, la que se reduce a la ecuación:

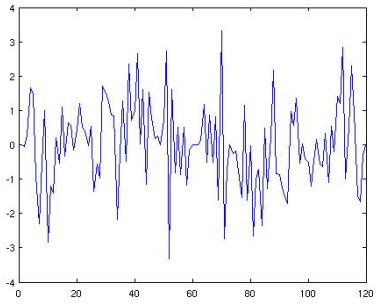
$$C_k = \sum_n \left(X_n \times exp\left(-j \times \frac{2 \times \pi \times k \times n}{N} \right) \right) \tag{1.11}$$

El cepstrum se divide en su parte real y su parte imaginaria, a las cuales se les aplica filtros, como puede ser la eliminación de algunos puntos máximos que no rebasan un valor establecido, y operaciones de compensación, para luego aplicar la transformada inversa de *Fourier* y la transformada de *Fourier* (Carpio-Muñoz and Carbajal, 2002). En la figura 1.6 se muestra la parte real y la parte imaginaria del perfil obtenido en la figura 1.5. La similitud entre la representación espectral de las secuencias es calculada de forma similar a los algoritmos de alineamiento basados en programación dinámica, pero se introduce un gradiente para que las correspondencias de las representaciones continúen diagonalmente sin la presencia de "gaps" (Carpio-Muñoz and Carbajal, 2002).

El análisis basado en *LPC*, modela el espectro como la combinación lineal de muestras anteriores, similar a un proceso autoregresivo (Deza and Deza, 2006). Una vez obtenido la representación del perfil físico-químico de la proteínas, para representar el espectro se le calcula la media móvil para un tamaño de ventana n, con el objetivo de minimizar la discontinuidad en los bordes (Achelis, 1995). Sea $X = (X_1, X_2, \ldots, X_N)$



(a) Parte real de la transformada de Fourier para el perfil físico-químico de la figura 1.5



(b) Parte imaginaria de la transformada de Fourier para el perfil físico-químico de la figura 1.5

Figura 1.6.: Representación del perfil físico-químico luego de aplicada la transformada de Fourier

la representación del perfil físico-químico de una proteína de N aminoácidos, se define la media móvil $MX = (MX_1, MX_2, MX_{N-n+1})$ como:

$$MX_j = \frac{\sum_{k=0}^{n-1} (X_{j+k})}{n} \tag{1.12}$$

La similitud entre los coeficientes espectrales obtenidos para cada subsecuencia es calculada mediante un coeficiente de correlación (Deza and Deza, 2006). Uno de los coeficiente de correlación más usados es el coeficiente de correlación de *Pearson*, el cual verifica si dos variables cuantitativas están linealmente relacionadas entre sí (Achen, 1982). Los valores obtenidos por el coeficiente de correlación se encuentran en el intervalo $[-1, 1]$, e indican el grado de relación directa si es positivo o inversa si es negativo. Una relación es directa si a medida que aumentan los valores en un espectro también aumentan en el otro, y es inversa si a medida que aumentan los valores en un espectro disminuyen en el otro espectro (Orfanidis, 1996).

1.5. Construcción y poda del grafo del grafo de similitud de secuencias

El grafo de similitud entre dos genomas se define como el grafo bipartido completo no dirigido $G = (V, E)$ (Bondy and Murty, 1976) que describe la similitud entre los genes de dos genomas G_1 y G_2 de las dos especies comparadas, donde cada gen de cada genoma representa un vértice del grafo. El conjunto de vértices V se define como la unión de los vértices que representan genes del genoma G_1 con los vértices que

representan genes del genoma G_2 (Diestel, 2000).

$$\begin{aligned} V_{G_1} &= \{X_1, X_2, \ldots, X_n\} \\ V_{G_2} &= \{Y_1, Y_2, \ldots, Y_m\} \\ V_{G_1} \cap V_{G_2} &= \{\phi\} \\ V = V_{G_1} \cup V_{G_2} &= \{X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m\} \end{aligned} \tag{1.13}$$

Todos los vértices que representan genes del genoma G_1 se encuentran relacionados por una arista con cada uno de los vértices del genoma G_2, y viceversa. Cada arista que conecta cada par de vértices es pesada por la similitud entre los genes que representan los vértices conectados por la arista.

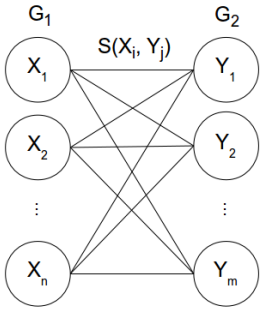


Figura 1.7.: Grafo bipartido completo

La poda del grafo bipartido completo reduce drásticamente la conectividad del grafo, eliminando todas las aristas no óptimas del grafo bipartido dirigido correspondiente. La mayor parte de las aristas podadas en este proceso, se debe a la existencia de áreas de proteínas similares entre proteínas lejanamente relacionadas, o proteínas con funciones similares, pero cuya separación precede a la divergencia de las especies. Seleccionando un umbral próximo al peso óptimo de cada nodo del grafo bipartido G, se puede asegurar que el algoritmo funciona en un ámbito de distancias evolutivas (Kamvysselis, 2003)[19].

Sea G el grafo bipartido ponderado completo que describe la similitud entre dos conjuntos de genes X y Y, y E el conjunto de aristas que conecta cada nodo $x_i \in X$ con cada nodo $y_j \in Y$, se construye el grafo G_1 como la versión dirigida del grafo G, sustituyendo cada arista no dirigida de la forma (x_i, y_j) por dos aristas dirigidas de la forma (x_i, y_j) y (y_j, x_i), ambas con el mismo peso de la arista no dirigida (Bondy and Murty, 1976). Las aristas óptimas de un nodo x del grafo bipartido G_1 se definen como aquellas aristas de salida del nodo, que su peso es relativamente cercano en un rango el peso óptimo del nodo x, definido como el mayor peso de todas las aristas de salida del nodo (Kamvysselis, 2003). Una vez podado el grafo bipartido dirigido G_1 se transforma en un grafo bipartido no dirigido.

[19]El autor define el umbral de la poda como el 80 % del valor del peso óptimo del nodo.

1.6. Algoritmo de agrupamiento MCL

El agrupamiento es una técnica de aprendizaje no supervisado que divide un conjunto de objetos en grupos (Kleinberg and Lawrence, 2001). Cada grupo está formado por objetos que son similares entre ellos y distintos a los que forman el resto de los grupos. Los problemas de agrupamiento se pueden dividir en dos grupos: agrupamiento duro[20] y agrupamiento blando[21]. Se dice que un agrupamiento es duro cuando un objeto puede pertencer a un único grupo, mientras que cuando el agrupamiento es blando puede pertenecer a más de un grupo (Kleinberg and Lawrence, 2001).

El procedimiento general de los métodos de agrupamiento sobre grafos, consiste en construir un grafo y luego aplicarle técnicas de particionado. El particionado de un grafo, es el proceso que divide el grafo en partes aproximadamente del mismo tamaño, de manera que prevalezcan pocas conexiones entre los grupos creados. En el caso de un grafo ponderado, la partición se encamina a minimizar la suma de los pesos de las aristas implicadas en el corte (Kleinberg and Lawrence, 2001). Algunos algoritmos de particionado sobre grafo fusionan los grupos en base a la interconexión relativa y la cercanía relativa de los nodos, como es el caso de *Chamelon* (Karypis et al., 1999), *ROCK* (Guha, 1999), *MCODE* (Bader and Hogue, 2003), *RNSC* (King et al., 2004), *SPC* (Brohue and Helden, 2006), entre otros.

Los algoritmos anteriormente mencionados están orientados al agrupamiento de genes de un organismo según un conjunto de características. El problema abordado por este trabajo intenta agrupar genes ortólogos en base al grado de similitud entre las secuencias de genes que pertenecen a diferentes genomas. Para dar solución a este problema es necesario agrupar los genes por las dos dimensiones de la matriz de adyacencia del grafo bipartido de forma simultánea.

Los algoritmos de co-agrupamiento[22] son algoritmos de agrupamiento sobre grafos que permiten agrupar las filas y columna de la matriz de adyacencia de forma simultánea (Mirkin, 1996). Los algoritmos pueden ser divididos en cinco grupos basándose en el procedimiento empleado para calcular los grupos: combinación iterativa de las filas y columnas, divide y vencerás, búsqueda iterativa, enumeración exhaustiva y la identificación de la distribución de parámetros. Varios algoritmos de co-agrupamiento como *CTWC* (Getz et al., 2000), *ITWC* (Tang et al., 2001), *DCC* (Busygin et al., 2002), *FLOC* (Yang et al., 2002) y *SAMBA* (Tanay et al., 2002) han sido empleados en agrupamiento de secuencias sobre un conjunto de características (Madeira and Oliveira,

[20]En inglés es conocido como ***hard clustering*** o ***crisp clustering***
[21]En inglés es conocido como ***soft clustering*** o ***fuzzy clustering***
[22]En inglés es conocido como ***biclustering***

2004).

El co-agrupamiento por lo general trabaja directamente sobre el grafo original para producir los grupos, lo que se convierte en un proceso altamente costoso en dependencia de la cantidad de nodos del grafo. Los algoritmos *METIS* (Karypis and Kumar, 1998a) y *ParMETIS* (Karypis et al., 2003) son algoritmos de particionado multinivel. Primeramente, reducen el tamaño del grafo, y aplican el agrupamiento sobre un grafo más pequeño, para luego reconstruir los grupos para el grafo original (Karypis and Kumar, 1998b), permitiendo trabajar sobre grafos con una mayor cantidad de nodos.

Al algoritmo de particionado *MCL* agrupa los nodos de un grafo mediante simulación, calculando la probabilidad de que estos pertenezcan al grupo (van Dongen, 2000). El algoritmo se basa en las matrices de *Markov*, que aplican el concepto de caminos aleatorios[23] en el grafo. Sea M la matriz cuadrada de orden N, la matriz de adyacencia de un grafo dirigido, se define la matriz de *Markov* como:

$$\begin{aligned} \tau_G = \quad & M \times d^{-1} \\ & d_{kk} = \sum_{i=1}^{N} M_{ik} \\ & d_{ij} = 0; i \neq j \end{aligned} \tag{1.14}$$

Un camino aleatorio se define como una cadena de *Markov* usando las matrices de probabilidades en cada transición de un estado al otro (van Dongen, 2000). En las matrices que conforman la cadena de *Markov*, el peso de los las aristas se hace mayor en los enlaces que pertenecen a un mismo grupo y menor entre los enlaces de grupos diferentes.

El algoritmo *MCL* se basa en la alternación de las operaciones de expansión e inflación sobre la matriz de *Markov*, hasta que se obtiene una matriz idempotente. La expansión consiste en calcular la potencia de la matriz de transición de *Markov*, la cual es responsable de permitir las conexiones entre diferentes regiones del grafo, mientras el operador de inflación es el encargado de fortalecer o debilitar las conexiones entre las diferentes regiones del grafo. Sea la matriz de *Markov* inicial $\tau_1 = \tau_G$, r el operador de inflación, en la iteración k del algoritmo se calculan las matrices τ_{2k} como resultado

[23]En inglés es conocido como ***random walks***

de la expansión y la matriz τ_{2k+1} como resultado de la inflación.

$$\tau_{2k} = \tau_{2k-1} \times \tau_{2k-1} \tag{1.15}$$

$$\tau_{2k+1} = (\Gamma_r \tau_{2k})_{pq} = (\tau_{2k}[p,q])^r / \sum_{i=1}^{N} (\tau_{2k}[i,q])^r \tag{1.16}$$

El algoritmo *MCL* tiene una complejidad temporal de $O(k \times N^3)$, donde N es el número de nodos del grafo (van Dongen, 2000). En la práctica el algoritmo converge a una matriz idempotente en k iteraciones, y por lo general $k \in [10, 100]$. La operación de inflación tiene una complejidad de $O(N^2)$, mientras que la expansión es el proceso más costoso $O(N^3)$. Usando matrices esparcidas y criterios de poda, algunas implementaciones del algoritmo MCL logran disminuir su complejidad computacional a $O(k \times N \times n^2)$ (ZOEM, 2011), donde $n \ll N$ es el número máximo de nodos adyacentes a un nodo cualquiera del grafo.

1.7. Validación de los resultados

Con la construcción del grafo bipartido a partir de diferentes combinaciones de las medidas de similitud entre los dos genomas, el algoritmo de agrupamiento produce resultados diferentes, de los cuales algunos pueden ser no relevantes desde el punto de vista de la calidad del agrupamiento. Uno de los objetivos de las técnicas de agrupamiento es incluir alguna medida de calidad, para desechar resultados poco relevantes.

Los índices o medidas de calidad del agrupamiento pueden ser clasificados en externos, relativos e internos (Brun et al., 2007). Los índices externos evalúan el desempeño del agrupamiento, al comparar los resultados obtenidos contra clasificaciones de referencia[24] realizadas sobre el mismo conjunto de datos. Un agrupamiento obtenido por un clasificador es mejor en la medida que este se parece más a las clasificaciones de referencia. Entre los índices externos más usados se encuentran la medida F[25] (van Rijsbergen, 1979), el índice de *Rand* (Rand, 1971), el índice de *Jaccard* (Ben-Hur et al., 2002), el índice de *Rand* ajustado (Hubert and Arabie, 1985) y el índice de *Minkowski* (Jardine and Sibson, 1971).

Los índices relativos se basan en el cálculo de la consistencia de los clasificadores,

[24] Llámese clasificaciones de referencias a las clasificaciones obtenidas por bases de datos internacionales que son usadas para la validación de algoritmos.

[25] En inglés es conocida como ***F-measure***.

comparando los agrupamientos obtenidos a partir del mismo algoritmo bajo condiciones diferentes, con el objetivo de decidir cuál revela mejor las características reales de los objetos. Al no necesitar información externa, muchas veces son tratados como índices internos, pero necesitan de la aplicación repetida del algoritmo de agrupamiento para comparar los resultados. En estos casos en que se desconoce el resultado esperado, la validación se hace de acuerdo al cumplimiento de algunas propiedades (Brun et al., 2007).

Por otra parte los índices internos se basan solamente en la información del propio agrupamiento, y a diferencia de los índices externos y relativos, no necesitan información adicional ni la repetición del proceso de agrupamiento. Un gran número de estos índices asumen que los agrupamientos obtenidos deben cumplir que los objetos en un mismo agrupamiento deben estar más cerca que los objetos de agrupamientos diferentes, por lo que se basan en la evaluación de la compacidad u homogeneidad de cada agrupamiento y la separación entre cada uno de los agrupamientos. Entre los índices internos más usados se encuentran el índice de validación *SD* (Halkidi et al., 2001), los índices basados en teoría de grafos (Bezdek and Pal, 1998), el índice de *Davies-Bouldin* (Davies and Bouldin, 1979) y otros índices que evalúan la conectividad y la densidad esperada (Stein and Niggemann, 1999).

El problema abordado en este trabajo se presenta como un problema de predicción de dos clases o clasificación binaria, en la que los pares de genes p_k se agrupan en dos clases: V_1 cuando se clasifican en ortólogos y V_0 cuando se clasifican en no ortólogos. Sea S un conjunto de N pares de genes $\{s_1, s_2, \ldots, s_N\}$ y $U = \{U_1, U_0\}$, un agrupamiento sobre el conjunto S que será tomado como referencia, donde U_1 es el conjunto de pares de genes ortólogos y U_0 el conjunto de pares de genes no ortólogos. Para cada agrupamiento $V = \{V_1, V_0\}$ del grafo bipartido, se obtiene una tabla de contingencia para analizar la relación entre los agrupamientos U y V, donde $n_{ij} = |U_i \cap V_j|, \forall i, j \in \{1, 0\}$ (Vinh et al., 2009).

Tabla 1.1.: Tabla de contingencia

$V \backslash U$	U_1	U_0	Totales
V_1	n_{11}	n_{10}	a_1
V_0	n_{01}	n_{00}	a_2
Totales	b_1	b_2	$\sum_{ij} n_{ij} = N$

Como se observa en la tabla de contingencia existen cuatro posibles relaciones entre la clasificación obtenida V y la clasificación tomada como referencia U. Si un par de

genes p_k es clasificado como ortólogo en V y U entonces $p_k \in n_{11}$, y son nombrados verdaderos positivos(VP); si por el contrario p_k es clasificado como ortólogo en V pero no en U, entonces $p_k \in n_{10}$, y son nombrados falsos positivos(FP). De igual forma se definen los falsos negativos(FN) cuando p_k no es clasificado como ortólogo por V pero si por U, por lo que $p_k \in n_{01}$; y los verdaderos negativos(VN) cuando $p_k \in n_{00}$, lo cual indica que p_k no es clasificado como ortólogo en V ni en U.

La tabla de contingencia puede proporcionar varias medidas de evaluación (Fawcett, 2004) como pueden ser la precisión de la clasificación, la exactitud, la sensibilidad y la especificidad. La precisión y la exactitud indican el porciento de pares de genes clasificados como ortólogos correctamente, la precisión a diferencia de la exactitud, no tiene en cuenta los pares de genes clasificados como ortólogos incorrectamente. La sensibilidad brinda la probabilidad de que un par de genes ortólogos sea clasificado correctamente, mientras que la especificidad muestra la probabilidad de clasificar un par de genes no ortólogos incorrectamente.

$$Precision(U,V) = \frac{VP}{VP+FP} = \frac{n_{11}}{n_{11}+n_{10}} \tag{1.17}$$

$$Exactitud(U,V) = \frac{VP+VN}{P+N} = \frac{n_{11}+n_{00}}{b_1+b_2} \tag{1.18}$$

$$Sensibilidad(U,V) = \frac{VP}{VP+FN} = \frac{n_{11}}{n_{11}+n_{01}} \tag{1.19}$$

$$Especificidad(U,V) = \frac{VN}{VN+FP} = \frac{n_{00}}{n_{00}+n_{10}} \tag{1.20}$$

1.7.1. Medida F

La medida F es una de los índices externos más sencillos, y está estrechamente relacionada con la precisión y la exactitud del agrupamiento V (Larsen and Aone, 1999). La medida devuelve un valor en el rango $[0, 1]$ que indica cuánto se parece el agrupamiento V al agrupamiento U. Usando la misma notación empleada en la sección anterior, el valor correspondiente a la medida F se calcula por la ecuación:

$$F\_measure(U,V) = \frac{2 \times Precision(U,V) \times Exactitud(U,V)}{Precision(U,V) + Exactitud(U,V)} \tag{1.21}$$

1.7.2. Índice de Rand

El índice de *Rand* (Rand, 1971) es una medida similitud simple entre dos agrupamientos U y V. El agrupamiento U representa la clasificación tomada como referencia, y V representa la clasificación obtenida por el algoritmo de agrupamiento. El índice devuelve un valor en el rango $[0, 1]$ que indica cuánto se parecen el agrupamiento V al agrupamiento U. Usando la misma notación empleada en la sección anterior, el valor correspondiente al índice de *Rand* se calcula por la ecuación:

$$\begin{aligned} RI(U,V) &= \frac{2 \times \sum_{ij} C_2^{n_{ij}} + C_2^N - \sum_i C_2^{a_i} - \sum_j C_2^{b_j}}{C_2^N} \\ &= \frac{n_{11} + n_{00}}{n_{11} + n_{00} + n_{10} + n_{01}} \end{aligned} \tag{1.22}$$

1.7.3. Índice de Jaccard

El índice de *Jaccard* (Ben-Hur et al., 2002) es una medida de similitud muy semejante al índice de *Rand*, sólo que no tiene en cuenta los elementos que no son clasificados correctamente en ninguna de las particiones. Usando la misma notación empleada en la sección anterior, el valor del índice de *Jaccard* se calcula por la ecuación:

$$JI(U,V) = \frac{n_{11}}{n_{11} + n_{10} + n_{01}} \tag{1.23}$$

1.7.4. Índice de Rand ajustado

Esta medida es la versión corregida del índice de *Rand*. Una deficiencia del índice de *Rand* es que el valor esperado del índice no puede tomar valores constantes, como por ejemplo cero. La forma general de un índice con valor esperado constante, el cual se encuentra entre $[0, 1]$ y toma valor cero cuando el índice es igual al valor esperado, como se muestra en la ecuación (Hubert and Arabie, 1985):

$$\frac{\acute{\imath}ndice - valor_{esperado}}{\acute{\imath}ndice_{máximo} - valor_{esperado}} \tag{1.24}$$

El índice de *Rand* ajustado usa una distribución hipergeométrica generalizada como modelo de aleatoriedad (Hubert and Arabie, 1985), por lo que el valor esperado puede

ser expresado como:

$$valor_{esperado}(C_2^{n_{ij}}) = \frac{\sum_i C_2^{a_i} \times \sum_j C_2^{b_j}}{C_2^N} \tag{1.25}$$

Usando la misma notación empleada en la sección anterior, y combinando la ecuación 1.24 con la ecuación 1.25 se obtiene (Hubert and Arabie, 1985):

$$\begin{aligned} ARI(U,V) &= \frac{\sum_{ij} C_2^{n_{ij}} - \frac{\sum_i C_2^{a_i} \times \sum_j C_2^{b_j}}{C_2^N}}{\frac{\sum_i C_2^{a_i} + \sum_j C_2^{b_j}}{2} - \frac{\sum_i C_2^{a_i} \times \sum_j C_2^{b_j}}{C_2^N}} \\ &= \frac{n_{11} - \frac{(n_{11}+n_{10}) \times (n_{11}+n_{01})}{n_{11}+n_{01}+n_{10}+n_{00}}}{\frac{(n_{11}+n_{10})+(n_{11}+n_{01})}{2} - \frac{(n_{11}+n_{10}) \times (n_{11}+n_{01})}{n_{11}+n_{01}+n_{10}+n_{00}}} \end{aligned} \tag{1.26}$$

Consideraciones finales del capítulo

En este capítulo se presentaron los conceptos de homología, ortología y paralogía. Se hizo un estudio sobre los algoritmos de detección de genes ortólogos y algunas medidas que caracterizan las secuencias de genes. Los algoritmos de detección de genes ortólogos basados en grafo utilizan la información del alineamiento de secuencias para construir el grafo y podarlo usando un definición operacional de ortología para aplicar un algoritmo de agrupamiento. El capítulo finaliza estudiando algoritmos de agrupamientos sobre grafos y las medidas de validación de los agrupamientos.

CAPÍTULO 2

DISEÑO E IMPLEMENTACIÓN

En este capítulo se hace un análisis del algoritmo propuesto y su implementación. Se presentan las medidas de similitud de las secuencias empleadas en la solución propuesta. Se realiza un análisis de la implementación de cada uno de los segmentos de la solución y se hace un análisis de escalabilidad para el cálculo de los alineamientos en paralelo.

2.1. Lenguage de programación y herramientas empleadas

En el área de la bioinformática son muchos los lenguajes de programación empleados y los ambientes de programación, unos con un mayor rendimiento que otros. Los lenguajes de programación más usados hasta el momento son: *C*, *C++*, *C#*, *Java*, *Perl* y *Python* (Fourment and Gillings, 2008); y los ambientes más usados son: *MATLAB* (MathWorks, 2009b) y *R* (Core-Team, 2001). La mayoría de los lenguajes disponen de paquetes de código para el desarrollo de aplicaciones para la bioinformática.

Perl incluye una colección de módulos, *BioPerl* (Stajich et al., 2002), que facilitan el desarrollo de scripts para aplicaciones bioinformáticas, mientras que *Python* dispone de un proyecto internacional de código abierto, *BioPython* (Cock et al., 2009), que provee un conjunto de módulos para resolver problemas bioinformáticos en *Python*. *BioPython* y *BioPerl* incluyen módulos para el manejo de los diferentes formatos de archivos, alineamiento de secuencias, manipulación de estructuras moleculares y permiten interactuar con otras herramientas como:*BLAST* (Altschul et al., 1990) y *ClustalW* (Larkin et al., 2007).

2.1.1. MATLAB

MATLAB es un ambiente de programación fácil de usar, donde problemas y soluciones son expresados en notaciones matemáticas conocidas. Integra la manipulación de matrices, gráfico de funciones y de conjuntos de datos, permite la implementación sencilla de algoritmos, la creación de interfaces de usuarios y brinda interfaces para interactuar con programas escritos en otros lenguajes como: *C*, *C++*, *Java* y *Fortran*. Además permite el procesamiento en paralelo y la ejecución en grupos de computadoras.

MATLAB es un lenguaje de alto nivel donde los arreglos son el tipo de dato fundamental, con sentencias de control de flujo, funciones, estructuras de datos y manejo de ficheros. Es un lenguaje de programación donde se combina el paradigma de programación estructurada con la programación orientada a objetos. Las operaciones y funciones básicas soportan las operaciones sobre arreglos sin necesidad de recorrer los arreglos (MathWorks, 2009b).

Aunque *MATLAB* fue pensado de forma general para trabajos de cálculo numérico, presenta una familia de herramientas o paquetes[1] para resolver problemas específicos. Estos paquetes son una colección de funciones que extienden el ambiente del *MATLAB* para solucionar clases particulares de problemas. El *MATLAB* dispone de paquetes para el procesamiento de señales, el control de sistemas, trabajo con redes neuronales entre otros. La mayor parte de estas funciones se encuentran implementadas en el propio lenguaje del *MATLAB*, lo que permite el acceso a su código fuente (MathWorks, 2009b).

MATLAB dispone de un paquete bioinformático[2] que lo convierte en un ambiente integrado para el análisis de genomas y proteomas. Usando las funciones básicas provistas por el paquete bioinformático se pueden implementar algoritmos y aplicaciones más complejos. este paquete permite el manejo de los diferentes formatos de datos de los genomas de forma local o mediante el acceso a bases de datos de genomas en la web. Tiene implementados los algoritmos de alineamiento global y local de *Needleman-Wunsch* y *Smith-Waterman* respectivamente, entre otras características que pueden ser consultadas en la ayuda oficial del *MATLAB* (MathWorks, 2009c).

2.1.2. Java

Java es un lenguaje de programación de alto nivel, orientado a objetos, sencillo, robusto y seguro. Los programas de *Java* son compilados en un código intermedio, que

[1] En inglés se conocen como ***toolboxes***
[2] En inglés se conoce como ***Bioinformatics Toolbox***

posteriormente es interpretado por la máquina virtual, lo que permite la portabilidad del programa. La portabilidad permite que un programa escrito en *Java* se ejecute de forma similar en cualquier sistema operativo e independiente del hardware (Schildt, 2001).

El estándar de *Java* dispone de un conjunto de clases que permiten el manejo de archivos binarios, texto plano y *xml*, manejo de excepciones, tipos de datos, algoritmos de ordenamiento y búsqueda, entre otros. También dispone de clases para el manejo de diferentes estructuras de datos como pueden ser: pilas, colas, listas, árboles y diccionarios.

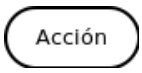
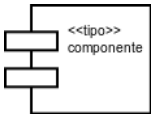
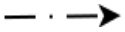
La comunidad de *Java* dispone del paquete *BioJava* (Holland et al., 2008), surgido con el objetivo de proveer un conjunto de clases para la representación y manipulación de información biológica. *BioJava* dispone de clases para el manejo de los diferentes formatos de archivos usados en la notación de secuencias, algoritmos genéticos y alineamientos, distribuciones estadísticas, así como para la manipulación de árboles filogenéticos, entre otros (Holland et al., 2008).

En comparación con el *MATLAB* el acceso a los archivos de datos es más rápido y eficiente. Permite el análisis de archivos de texto de forma más rápida y sencilla que el *MATLAB*. Por la facilidades que brinda *Java* para el manejo de las diferentes estructuras de datos, la disponibilidad del paquete *BioJava* y el paquete *OrthoXML* y *SeqXML* (Schmitt et al., 2011) se decide usar *Java* para desarrollar parte de la aplicación correspondiente a esta investigación, fundamentalmente el manejo y preparación de los datos, la construcción y procesamiento del grafo bipartido y la validación de los resultados.

2.1.3. DIA, herramienta de diagramas UML

El diagrama estructurado de procesos o de flujo, es una técnica apropiadas para la descripción gráfica de algoritmos. La herramienta *DIA* (Breit et al., 2009) es una aplicación de software libre para crear diagramas técnicos sencillos y es muy útil para dibujar diagramas *UML*, mapas de redes y gráficos de flujo. La tabla 2.1 muestra los estereotipos empleados para representar los diagramas de este trabajo empleando *DIA* versión 0.97.2.

Tabla 2.1.: Estereotipos empleados en los diagramas

Representación	Descripción
	Indica el comienzo de una actividad
	Indica la culminación de una actividad
	Tarea a realizar por el algoritmo
	Datos consultados o modificados por las tareas
	Indica el paso de una acción a otra
	Indica que un archivo es consultado o modificado
	Componentes de la aplicación
	Indica la dependencia de un componente de otro

2.1.4. Mauve

El *Mauve* (Darling et al., 2009) es una herramienta de alineamiento anclado. El alineamiento obtenido por el *Mauve* permite determinar los bloques *LCB*, los cuales son usados para calcular una medida de similitud entre las secuencias. Para ejecutar el *Mauve* se empleó el algoritmo progresivo por ser más eficiente que el algoritmo original como se planteó en el capítulo anterior. Por defecto para el algoritmo progresivo, el *Mauve* selecciona un conjunto de parámetros apropiados para el alineamiento de genomas relativamente cercanos, aunque algunos parámetros pueden ser ajustados para obtener mejores resultados (Darling et al., 2009). La figura 2.1 muestra los parámetros utilizados para la ejecución del *Mauve* usando el algoritmo progresivo.

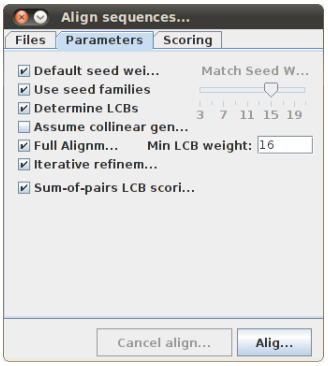


Figura 2.1.: Parámetros de ejecución del *Mauve*

El parámetro "*Match seed weight*" indica el peso mínimo que debe tener el patrón de correspondencia usado para calcular los alineamiento múltiples durante la primera

iteración del alineamiento anclado. En genomas divergentes se recomienda usar un valor pequeño, pero el *Mauve* requiere que el patrón sea único en cada genoma, por lo que un valor pequeño también puede reducir la sensibilidad. Si se selecciona el parámetro "*Default seed weight*" el *Mauve* selecciona el peso mínimo del patrón de correspondencia en dependencia de la longitud de las secuencias que se quieren alinear. El valor del peso aumenta en correspondencia a la longitud de los genomas, por ejemplo genomas de $1MB$ tienen un peso cercano a 11 y genomas de $5MB$ tienen un peso cercano a 15. En algunos casos estos altos valores reducen el ruido en la correspondencia y pueden conducir a mejores resultados del alineamiento. Directamente relacionado con el patrón de correspondencia se encuentra el parámetro "*Use seed families*", con el cual el *Mauve* usa tres patrones de correspondencias en lugar de uno, lo que permite usar pesos mayores en organismos relativamente cercanos sin perder sensibilidad.

Cuando se selecciona el parámetro "*Determine LCBs*" el *Mauve* identifica las correspondencias entre genomas y calcula los bloques *LCB*, en otro caso solo determina las correspondencias. Para determinar la pertenencia de una secuencia a un *LCB* por similitud se usa el parámetro "*Min LCB Weight*". Por defecto el *Mauve* usa 3 veces el peso mínimo del patrón de correspondencia, pero se recomienda seleccionar este valor de forma manual (Darling et al., 2009). Para la presente investigación se decidió usar el valor 16, el cual se corresponde con la tercera parte de la longitud mínima de las secuencias en nucleótidos.

Existen otros parámetros como "*Assume collinear genomes*", con el cual se asume que no existen reordenamientos en los genomas comparados. Usando el parámetro "*Full Alignment*" el *Mauve* realiza una búsqueda anclada recursiva y se calcula el alineamiento completo usando *MUSCLE* (Edgar, 2004), y cuando se combina con "*Iterative refinement*", *MUSCLE* refina el alineamiento, evitando la polarización de la inferencia filogenética con un sólo árbol guía. El parámetro "*Sum-of-pairs LCB scoring*" aún se considera experimental en el *Mauve* y se recomienda no emplearlo cuando se comparan genomas colineales (Darling et al., 2009).

Para la ejecución del *Mauve* fue necesario combinar los archivos *GenBank* de los cromosomas en un sólo archivo *GenBank* para cada genoma. Entre los archivos de salida del *Mauve* se encuentra el archivo del árbol guía, el *backbone* y el archivo principal de la salida en formato *XMFA*, el cual es de interés para el cálculo de los límites de cada *LCB*. Directamente el *Mauve* no determina los límites de los bloques *LCB*, por lo que se recurrió a la herramienta ***makeBadgerMatrix*** (Darling et al., 2008), a la cual se le pasa el archivo de la salida en formato *XMFA* y devuelve un archivo con los límites de cada *LCB*. En total de obtuvieron 463 *LCBs* con una longitud mínima de

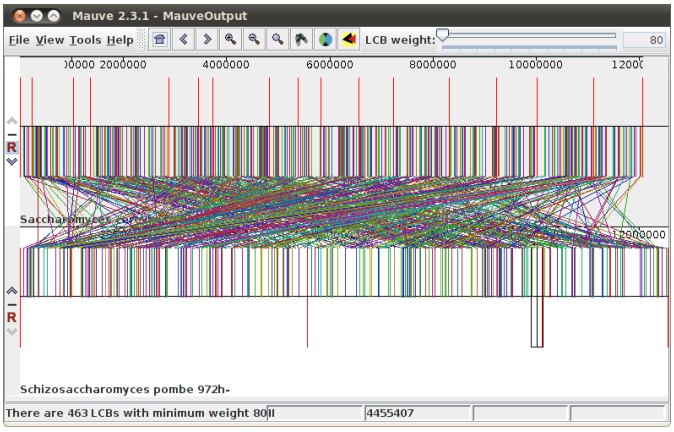
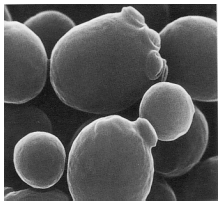


Figura 2.2.: Representación de los *LCB* obtenidos por el *Mauve*

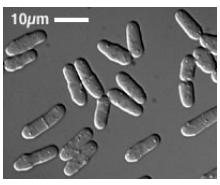
80 nucleótidos, lo cual se corresponde por los reportados por el *Mauve* en la figura 2.2.

2.2. Manejo de los datos de experimentación

Las levaduras son organismos muy estudiados y existe un amplio conocimiento funcional sobre estos, que permite validar resultados de un algoritmo con resultados ya existentes. Son de un tamaño de genoma mucho más pequeño que el genoma humano, lo cual permite el análisis comparativo de las secuencias con un costo razonable de tiempo y espacio. Por estas razones se decidió emplear los genomas de las levaduras del *Saccharomyces Cerevisiae* (Goffeau, 1996) y *Schizosaccharomyces Pombe* (Wood, 2002) para validar los resultados del algoritmo propuesto en este trabajo (ver figura 2.3). Los datos de los genomas fueron obtenidos desde el sitio oficial de *INPARANOID* (Östlund et al., 2010), así como la lista de pares de genes ortólogos obtenida por el algoritmo de *INPARANOID 7.0* para este par de genomas. Se decidió emplear la lista de pares de genes ortólogos obtenida por *INPARANOID* porque es uno de los algoritmos de detección de genes ortólogos más reconocidos. Además las listas de ortólogos curadas manualmente no brindan los genomas empleados para su confección.



(a) *Saccharomyces Cerevisiae*



(b) *Schizosaccharomyces Pombe*

Figura 2.3.: Genomas usados en la experimentación

Los archivos en formato *XML* obtenidos de *INPARANOID* contienen el nombre del gen, la descripción y la secuencia de aminoácidos que representa la proteína. Para obtener la información de las posiciones iniciales y finales de cada gen dentro del genoma, el cromosoma al cual pertenece y si está anotado en complemento reverso o no, se usaron los archivos *GenBank* de cada cromosoma de los genomas, los cuales fueron tomados del sitio oficial de *NCBI*. La información de los datos de *INPARANOID* y los archivos *GenBank* de cada genoma se combinaron en un archivo de datos en formato *FASTA*. Para este propósito se desarrolló la aplicación **XML2FASTA** en *Java*, la cual toma los archivos *XML* de *INPARANOID* y combina la información de estos con la información de los archivos *GenBank* de *NCBI*, como muestra la figura 2.4. Las secuencias de los archivos *XML* no encontradas en los archivos *GenBank* fueron desechadas por pertenecer al cromosoma mitocondrial. La lista de los genes desechados de cada genoma puede ser consultada en el anexo C. En la tabla 2.2 se muestra un resumen de los datos de los genomas luego de eliminar los pares desechados.

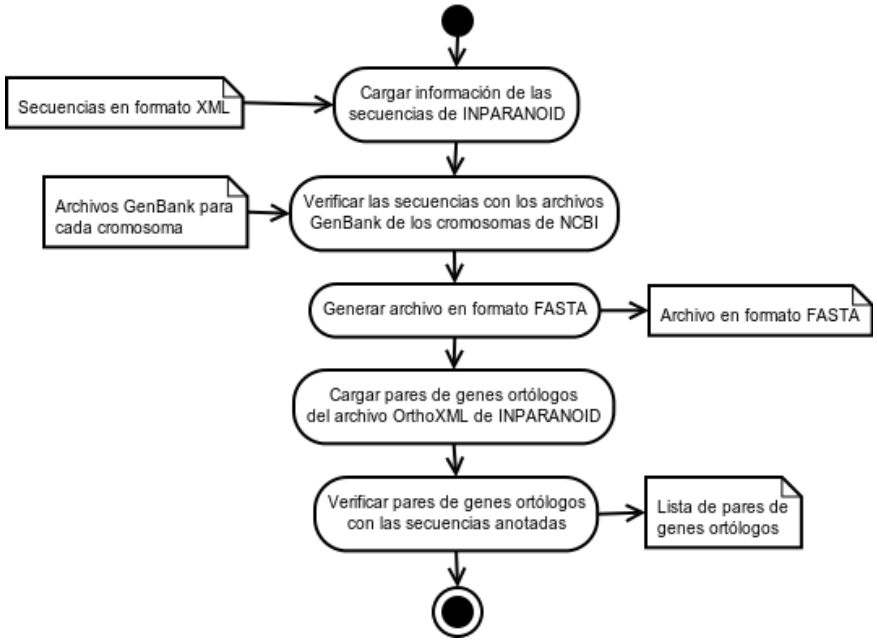


Figura 2.4.: Esquema general del manejo de los datos

La lista de pares de genes ortólogos dada por *INPARANOID* en formato *OrthoXML* es transformada a una lista de pares de índices de genes usando la aplicación **OrthoXML2List** desarrollada en *Java*. Los pares de índices se definen como I_{G1} I_{G2}, donde I_{G1} representa el índice de la secuencia según el orden de las secuencias en el archivo *FASTA* del primer genoma, y I_{G2} representa el índice de la secuencia según el orden de las secuencias en el archivo *FASTA* del segundo genoma. Fueron desechados los pares de genes ortólogos donde al menos una de las secuencias que lo componen

Tabla 2.2.: Resumen de los genomas

	Saccharomyces Cerevisiae	*Schizosaccharomyces Pombe*
Cromosomas	16	3
Genes	5861	5006
Genes desechados	23	16
Longitud Míx.	16	24
Longitud Máx.	4910	4924
Longitud Promedio	495	470

es una de las secuencias desechadas en alguno de los genomas, quedando 3807 pares de genes ortólogos luego de eliminar 12 pares. Los pares de genes ortólogos desechados pueden ser consultados en el anexo C.

2.3. Definición general del algoritmo

El algoritmo de detección de genes ortólogos propuesto en este trabajo parte de los archivos en formato *FASTA* de los dos genomas a comparar y devuelve una lista de pares de genes ortólogos. Un primer paso del algoritmo es calcular las medidas de similitud entre cada par de genes entre ambos genomas para construir el grafo bipartido completo. A continuación se le aplican técnicas de poda al grafo bipartido para reducir la conectividad de los nodos, eliminando aristas de baja similitud, y después de podado, se transforma en una matriz de adyacencia cuadrada para ser agrupada por el algoritmo *MCL*.

Una vez agrupado el grafo bipartido usando el algoritmo *MCL*, se aplica una política de asignación de genes ortólogos a los grupos obtenidos, para conformar la lista de pares de genes ortólogos devuelta por el algoritmo. Esta lista de ortólogos es validada tomando como referencia la lista de genes ortólogos obtenida por el algoritmo de *INPARANOID*. Para esta validación se usan las medidas de validación externas, siendo la medida *ARI* la medida de mayor importancia en la validación. La figura 2.5 muestra los pasos del algoritmo propuesto de forma general.

El *MATLAB* fue de gran utilidad para la implementación de una parte del algoritmo, pues con el paquete bioinformático se realizaron los alineamientos locales con la función ***swalign*** y los alineamientos globales con la función ***nwalign***, así como el cálculo del perfil físico-químico a partir de los alineamientos obtenidos. Los alineamientos de pares de secuencias se implementaron de forma paralela empleando la instrucción ***parfor***

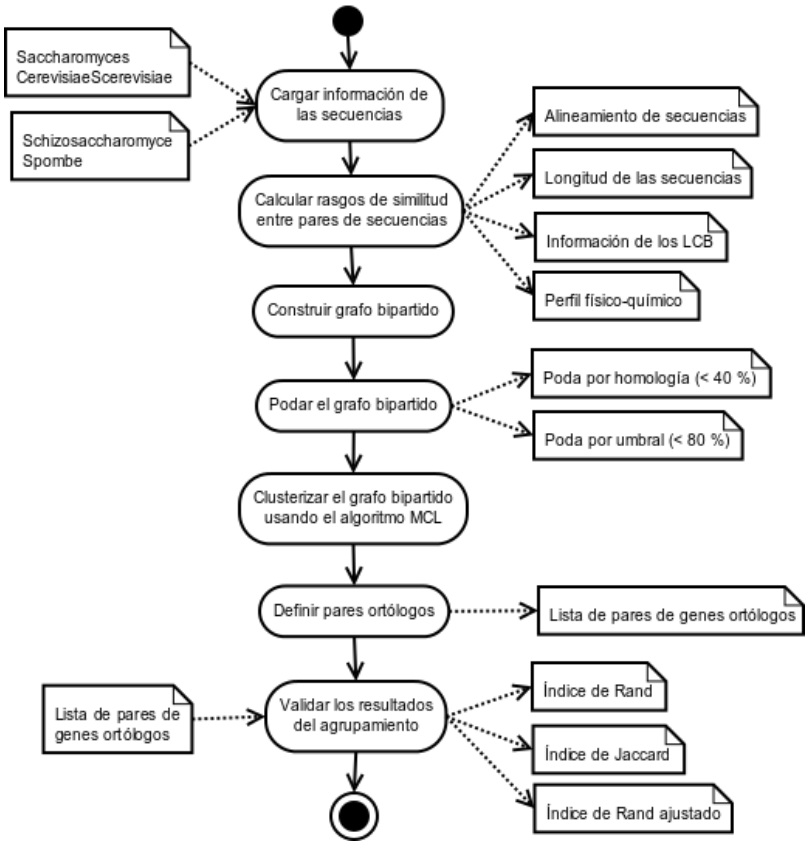


Figura 2.5.: Esquema general del algoritmo

(MathWorks, 2009a) que provee el paquete de procesamiento en paralelo[3] del *MATLAB*. Se implementaron las funciones ***calculate_alignment_features*** para el cálculo de los alineamiento par a par de todas las secuencias, y ***calculate_profile_features*** para el cálculo de la medida de similitud basada en los perfiles físico-químicos de cada par de secuencias, como se muestra en la figura 2.6.

Usando las facilidades que brinda *Java* se implementó un conjunto de clases para desarrollar una parte del algoritmo. Estas clases se agruparon en paquetes de clases teniendo en cuenta la funcionalidad general de cada clase. La figura 2.7 muestra la relación entre cada uno de los paquetes de clases implementados. De forma general se programaron dos paquetes básicos, el paquete ***core*** y el paquete ***io***, y cinco paquetes de aplicaciones para el procesamiento de los datos del algoritmo, los cuales dependen de los paquetes básicos y de los paquetes *BioJava* y *OrthoXML*.

El paquete ***core***, contiene las clases para representar los elementos básicos usados por el algoritmo, como son las secuencias, los bloques de orden conservado, los bloques *LCB* y los pares de genes. Por su parte el paquete ***io***, contiene las clases empleadas para realizar la lectura y escritura de los elementos básicos en ficheros de texto plano,

[3]En inglés es conocido como ***parallel computing toolbox***

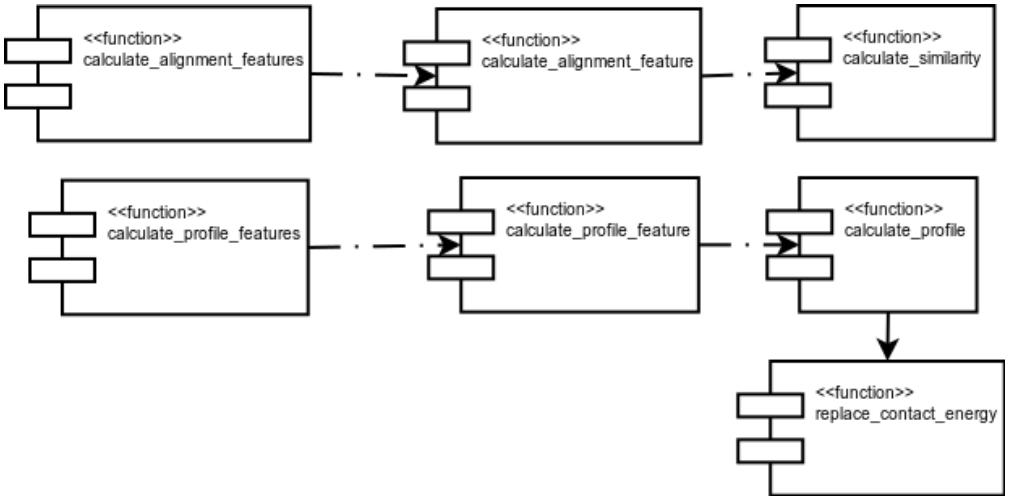


Figura 2.6.: Esquema general de los elementos del algoritmo desarrollados en *MATLAB*

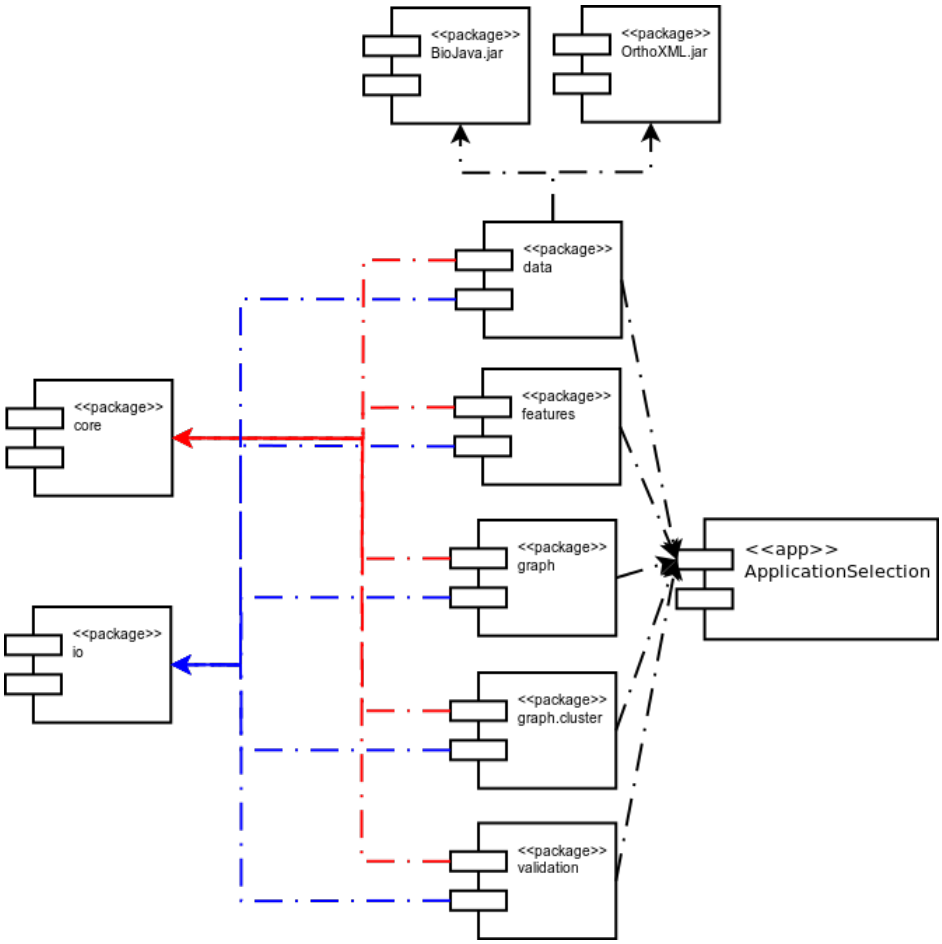


Figura 2.7.: Esquema general de los elementos del algoritmo desarrollados en *Java*

así como la lectura y escritura de matrices de datos.

Las aplicaciones para el manejo de los datos del algoritmo se agruparon en el paquete ***data***, y las aplicaciones para las operaciones sobre las medidas de similitud, como

son el cálculo de algunas medidas, la normalización y la agregación de medidas, fueron agrupadas en el paquete ***features***. Por otra parte, los paquetes ***graph*** y ***graph.cluster*** contienen las aplicaciones para el manejo del grafo bipartido como matriz de adyacencia. Estos contienen aplicaciones para las operaciones de poda, transformación, asignación de genes ortólogos e implementaciones de los algoritmos de detección de genes ortólogos basados en *BUS* y *RBH*. En el paquete ***validation*** se incluyen las aplicaciones para el cálculo de las medidas de validación implementadas. Para ver una información más detallada de las clases implementadas consultar el anexo B.

2.4. Medidas de similitud

En este trabajo se emplean cuatro medidas de similitud que tienen en cuenta diferentes aspectos relacionados con las secuencias comparadas. Estos aspectos son: el alineamiento de las secuencias, la longitud de las secuencias, la pertenencia de las secuencias a los bloques *LCB*, y la información del perfil físico-químico de las proteínas. Las medidas de similitud entre pares de secuencias se representan mediante una matriz, donde cada fila representa genes del primer genoma, las columnas genes del segundo genoma y cada celda contiene el grado similitud entre los genes que corresponden a la fila y la columna de la celda.

Los valores de las medidas de similitud son normalizados en el intervalo $[0, 1]$, dividiendo por el máximo valor de similitud obtenido. En los casos en que el grado de similitud es negativo, se considera una similitud no significativa, y se asume como valor nulo el cero. Para normalizar y corregir los valores negativos de una matriz de similitud, se implementó en *Java* la aplicación ***CalculateNormalizedMeasure***, la cual se le indica una matriz de similitud y se normalizan sus valores dividiendo entre el máximo valor de similitud que alcanza la matriz, y se reemplazan las similitudes negativas por el valor nulo.

2.4.1. Similitud basada en el alineamiento de secuencias

Como se analizó en el capítulo anterior, el alineamiento de secuencias brinda una de las medidas más influyentes en la clasificación de genes ortólogos. Hasta el momento los trabajo realizados en el laboratorio de Bioinformática de la *UCLV* han trabajado esta medida de similitud basada en la puntuación del alineamiento local o global de secuencias. El alineamiento local brinda la relación funcional entre las secuencias,

mientras que el global brinda la relación estructural. En este trabajo se propone basar la medida de similitud en una combinación de la puntuación del alineamiento local y global a partir de la agregación de las dos medidas. El objetivo es tener en cuenta la relación funcional y estructural de las secuencias en comparación.

Sean los genomas $G_1 = (X_1, X_2, \ldots, X_n)$ y $G_2 = (Y_1, Y_2, \ldots, Y_m)$, con n y m secuencias respectivamente, la medida de similitud basada en la puntuación del alineamiento local entre cada par de secuencias de ambos genomas se define como:

$$S_{local}(X_i, Y_j) = \begin{cases} ca\_local(X_i, Y_j) & , ca\_local(X_i, Y_j) > 0 \\ 0 & , ca\_local(X_i, Y_j) \leq 0 \end{cases} \tag{2.1}$$

$$ca\_local = \frac{swalign(X_i, Y_j)}{\text{máx}(swalgin(X_k, Y_l))} \quad \forall k \in [1, n]; \quad \forall l \in [1, m]$$

De forma similar se define la similitud basada en la puntuación del alineamiento global entre cada par de secuencias, sustituyendo la función de alineamiento local ***swalign*** por la función de alineamiento global ***nwalign***:

$$S_{global}(X_i, Y_j) = \begin{cases} ca\_global(X_i, Y_j) & , ca\_global(X_i, Y_j) > 0 \\ 0 & , ca\_global(X_i, Y_j) \leq 0 \end{cases} \tag{2.2}$$

$$ca\_global = \frac{nwalign(X_i, Y_j)}{\text{máx}(nwalgin(X_k, Y_l))} \quad \forall k \in [1, n]; \quad \forall l \in [1, m]$$

Adicionalmente con la similitud del alineamiento global de cada par de secuencias se calcula el porciento de similitud usando la representación del alineamiento, el cual es empleado posteriormente para realizar operaciones de poda sobre el grafo bipartido. Sea $A = (A_1, A_2, \ldots, A_w)$ la cadena de correspondencias[4] resultante del alineamiento global de las secuencias X_i y Y_j, el porciento de similitud entre X_i y Y_j se define según la ecuación 2.3, donde $Identidad(A)$ es la cantidad de veces que aparece el caracter "|" en A y $Similitud(A)$ es la cantidad de veces que aparece el caracter ":".

$$PS(A) = \frac{100 \times (Identidad(A) + Similitud(A))}{w} \tag{2.3}$$

[4]La cadena de correspondencia esta compuesta por "|" para indicar identidad, ":" para indicar alta similitud, y " " para otros casos

La medida de similitud S_1 basada en la puntuación del alineamiento, propuesta en este trabajo como la combinación de la medida de similitud basada en la puntuación local y global, se define usando la media aritmética como:

$$S_1(X_i, Y_j) = \frac{S_{local}(X_i, Y_j) + S_{global}(X_i, Y_j)}{2} \tag{2.4}$$

El algoritmo 1 muestra el pseudocódigo de la función ***calculate_alignment_feature*** implementada en el *MATLAB*, para el cálculo de la medida de similitud combinando la puntuación de los alineamientos locales y globales. Esta función recibe como parámetros las secuencias de los dos genomas, la matriz de sustitución y valores de penalización de *"gaps"* para calcular los alineamientos par a par. El porciento de similitud se calcula usando la función ***calculate_similarity***, también programada en *MATLAB*. Como salida produce tres archivos, la matriz de puntuación del alineamiento local, la matriz de puntuación del alineamiento global y la matriz de porciento de similitud, como se muestra en la figura 2.8. Las matrices de puntuación obtenidas por el *MATLAB* son agregadas en una sola matriz de puntuación usando la media aritmética, a través de la aplicación ***AggregateMeasures*** desarrollada en *Java*, a la cual se le indican las matrices de entrada y produce como salida la nueva matriz agregada.

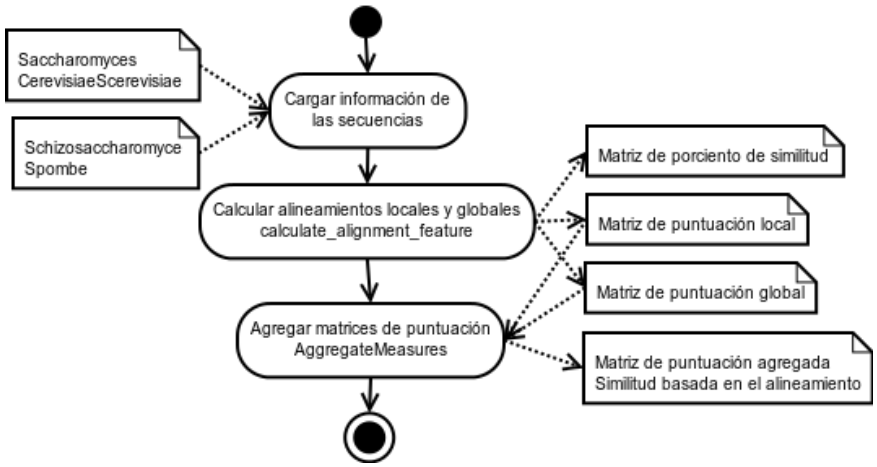


Figura 2.8.: Esquema general del cálculo de la similitud basada en el alineamiento

2.4.2. Similitud basada en la longitud de las secuencias

La medida de similitud basada en la longitud de las secuencias es calculada a partir de la distancia de las diferencias renormalizadas (Duch, 2000). Esta distancia se transforma en medida de similitud, expresando la similitud en función de la distancia en la primera ecuación de 1.5, por lo que la medida de similitud S_2 basada en la longitud de las

Algoritmo 1 Cálculo de los alineamientos locales y globales

Input: G_1, G_2, $scoreMatrix$, $open$, $extend$
  for all $s_i \in G_1$ **do**
    $local \leftarrow array()$
    $global \leftarrow array()$
    $sim \leftarrow array()$
    parfor $s_j \in G_2$ **do**
      $local[j] \leftarrow swalign(s_i, s_j, scoreMatrix, open, extend)$
      $global[j] \leftarrow nwalign(s_i, s_j, scoreMatrix, open, extend)$
      $sim[j] \leftarrow calculate\_similarity(global[j])$
    end for
    $saveScore(local$, "PuntuaciónLocal.out")
    $saveScore(global$, "PuntuaciónGlobal.out")
    $save(sim$, "PorcientoSimilitud.out")
  end for

secuencias se define como:

$$S_2(X_i, Y_j) = 1 - \frac{|longitud(X_i) - longitud(Y_j)|}{\text{máx}(longitud(Z_k)) - \text{mín}(longitud(Z_k))} \quad (2.5)$$
$$Z = X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_m$$
$$\forall k \in [1, n+m]$$

Si una secuencia X_i de un genoma tiene una longitud muy pequeña y otra secuencia Y_j del otro genoma tiene una secuencia relativamente grande, entonces es probable que tengan un alto grado de similitud, dado por la puntuación del alineamiento local. Esto puede provocar que secuencias poco semejantes se clasifiquen incorrectamente. Por otra parte la similitud basada en sus longitudes tiende a ser un valor pequeño, el cual sirve de contrapartida al alto grado de similitud obtenido por el alineamiento.

Para el cálculo de la medida de similitud basada en la longitud de las secuencias se emplea la aplicación ***CalculateLengthMeasure***, desarrollada en *Java*. A esta se le indican los archivos en formato FASTA de los dos genomas, y produce como salida la matriz de similitud entre los dos genomas, calculada según la ecuación 2.5.

2.4.3. Similitud basada en la pertenencia a los bloques LCB

La similitud basada en la pertenencia a los bloques *LCB* obtenidos por el *Mauve*, se calcula a partir de las distancia entre los pares de secuencias, teniendo en cuenta la pertenencia a los bloques *LCB*. Esta distancia se transforma en medida de similitud,

expresando la similitud en función de la distancia en la primera ecuación de 1.5, por lo que la medida de similitud S_3 basada en la pertenencia a los bloques LCB se define como:

$$S_3(X_i, Y_j) = 1 - dlcb(X_i, Y_j) \tag{2.6}$$

Sean los genomas $G_1 = (X_1, X_2, \ldots, X_n)$ y $G_2 = (Y_1, Y_2, \ldots, Y_m)$, $LCB[k, l]$ la matriz de la cantidad de aminoácidos de cada secuencia en cada LCB, donde $LCB[k, 1 \ldots n]$ es la cantidad de aminoácidos de las secuencias del genoma G_1 en el bloque k y $LCB[k, n+1 \ldots n+m]$ es la cantidad de aminoácidos de las secuencias del genoma G_2; entonces la distancia entre dos secuencias X_i y Y_j del genoma G_1 y G_2 se define como:

$$\begin{aligned} dlcb(X_i, Y_j) &= \frac{1}{P} \times \sum_{k=1}^{cantidadLCB} dlcb(k, X_i, Y_j) \\ dlcb(k, X_i, Y_j) &= \begin{cases} 0, si\ \text{máx}(LCB[k,l]) = \text{mín}(LCB[k,l]) \\ \dfrac{|LCB[k,i] - LCB[k,n+j]|}{\text{máx}(LCB[k,l]) - \text{mín}(LCB[k,l])} \end{cases} \\ & \forall l \in [1, n+m]; k = 1 \ldots cantidadLCB \end{aligned} \tag{2.7}$$

donde P es la cantidad de LCB donde una o ambas secuencias del par contiene al menos un aminoácido. En la figura 2.9 se indica la secuencia de pasos a seguir para obtener esta medida de similitud. La aplicación ***CalculateLCBMeasure***, implementada en *Java*, recibe los archivos en formato *FASTA* de los dos genomas en conjunto con el archivo de los límites de cada LCB obtenidos con ***makeBadgerMatrix***, y produce como salida la matriz de similitud entre los dos genomas calculada según la ecuación 2.6, el archivo con los límites de cada LCB y el archivo con la cantidad de aminoácidos de cada secuencia en cada LCB.

El archivo de los límites de los LCB, contiene una línea con cuatro elementos por bloque, donde los dos primeros representan las posiciones inicial y final dentro del primer genoma; y los dos últimos las posiciones dentro del segundo genoma. La cantidad de aminoácidos de cada secuencia en cada LCB, se guarda en forma matricial, guardando una línea para cada fila, donde las columnas son separadas por espacios. Primero se guardan las k filas correspondientes al primer genoma y luego las k filas del segundo genoma, siendo k la cantidad de bloques LCB.

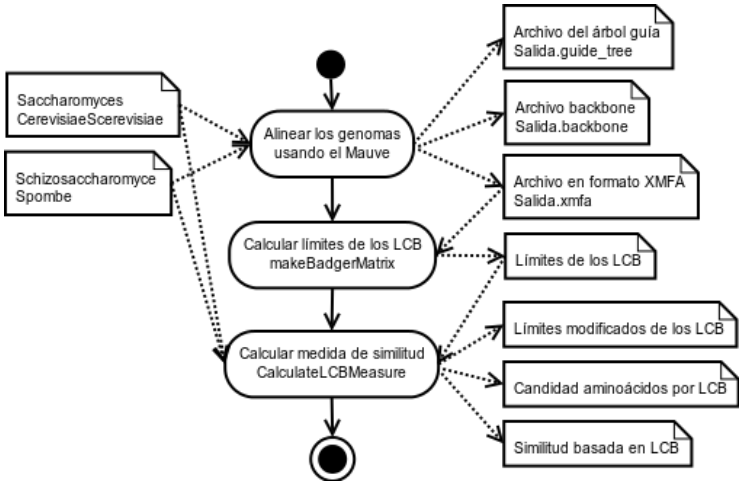


Figura 2.9.: Esquema general del cálculo de la similitud basada en la pertenencia a los *LCB*

2.4.4. Similitud basada en el perfil físico-químico de las proteínas

Como se estudió en el capítulo anterior el perfil físico-químico de las proteínas puede ser analizado mediante la transformada rápida de *Fourier* o por *LPC*, siendo esta última la seleccionada para el desarrollo de este trabajo. A diferencia de los métodos tradicionales, el análisis del perfil físico-químico propuesto en este trabajo no se basa en el alineamiento de la representación espectral de las secuencias, si no, en el cálculo de la representación espectral de las secuencias a partir del alineamiento global de ambas secuencias.

Tomando como referencia el alineamiento global de dos secuencias, se buscan las regiones de correspondencia sin *"gaps"* y se sustituye cada aminoácido por su energía de contacto, para luego calcular la relación entre las dos representaciones espectrales usando el coeficiente de correlación de *Pearson*. La siguiente tabla muestra la energía de contacto de cada aminoácido.

Tabla 2.3.: Energía de contacto correspondiente a cada aminoácido

A	**R**	**N**	**D**	**C**	**Q**	**E**	**G**	**H**	**I**
-0.03	0.08	0.12	0.18	-0.35	0.14	0.22	0.05	-0.03	-0.32
L	**K**	**M**	**F**	**P**	**S**	**T**	**W**	**Y**	**V**
-0.1	0.32	-0.28	-0.35	0.17	0.11	0.03	-0.28	-0.25	-0.27

Una vez sustituidos cada uno de los aminoácidos por su energía de contacto para determinar la representación espectral de la región, se calculan las medias móviles para

cada uno de los espectros usando la ecuación 1.12. A partir de los nuevo espectros obtenidos se calcula el coeficiente de correlación de *Pearson*, donde la correlación es significativa si el valor de significación obtenido es menor que 0,05, por lo que el grado de similitud de una región sin *"gaps"* se define como:

$$Corr_{Pearson}(MX, MY) = \begin{cases} corr(MX, MY) & , sig \leq 0{,}05 \\ 0 & , sig > 0{,}05 \end{cases} \tag{2.8}$$

Una vez calculada la similitud de cada una de las R regiones sin *"gaps"*, es necesario combinar estas similitudes para determinar la similitud global de las dos secuencias comparadas. Las similitudes de cada una de las regiones son agregadas hallando la relación del grado de similitud en función de la longitud de la región, para la longitud del alineamiento sin *"gaps"*, por lo que la medida de similitud S_4 basada en la información del perfil físico-químico de las proteínas se define como:

$$S_4(X_i, Y_j) = \frac{\sum_{k=0}^{R} Corr_{Pearson}(MX_{ik}, MY_{jk}) \times long_k}{\sum_{k=0}^{R} long_k} \tag{2.9}$$

Como se observa en el algoritmo 2, el cálculo de la similitud basada en el perfil físico-químico de las proteínas parte del alineamiento global obtenido por la función ***nwalign*** del *MATLAB*, en el cual se buscan las regiones de correspondencias sin *"gaps"*, y se sustituyen los aminoácidos por su energía de contacto usando la función ***replace_contact_energy***, programada en el *MATLAB*. Una vez obtenida la representación espectral de la región, se calcula la media móvil usando la función ***tsmovavg*** del *MATLAB*, para un tamaño de ventana indicado como parámetro. Finalmente, se calcula el coeficiente de correlación de *Pearson* usando la función ***corr*** del *MATLAB*, la cual devuelve el valor de correlación y la significación estadística. Se desechan las correlaciones no significativas. Una vez calculados los coeficientes de correlación para cada una de las regiones, se combinan según la ecuación 2.9.

Las operaciones sobre arreglos hacen que el cálculo de la similitud sea un proceso computacionalmente muy costoso. Para ahorrar tiempo de cómputo y evitar realizar cálculos innecesarios se decidió calcular esta medida de similitud de forma independiente a la similitud basada en el alineamiento, aunque es necesario recalcular una parte de los alineamientos globales. La similitud se calcula solamente en aquellos pares de secuencias que no resultan podados por la homología de las secuencias (ver sección 2.5).

El cálculo de la medida de similitud se implementó con la ayuda de *Java* y *MATLAB*. En *Java* se programaron las aplicaciones ***MatrixMask2List*** y ***List2Matrix***, donde la primera toma una máscara de poda y crea una lista con los pares de genes no

Algoritmo 2 Cálculo de la similitud basada en el perfil físico químico de las proteínas

Input: G_1, G_2, $pares$, $scoreMatrix$, $open$, $extend$, $wndSize$

```
simProfile ← array()
parfor (s_i, s_j)_k ∈ pares do
    global ← nwalign(s_i, s_j, scoreMatrix, open, extend)
    rLen ← array()
    rCorr ← 0
    for all r_l ∈ regionesSinGaps(global) do
        rLen[l] ← len(r_l)
        ceS_1 ← replace_contact_energy(r_l.seq1)
        ceS_2 ← replace_contact_energy(r_l.seq2)
        maS_1 ← tsmovavg(ceS_1,'s', wndSize)
        maS_2 ← tsmovavg(ceS_2,'s', wndSize)
        profile ← corr(maS_1', maS_2','type','Pearson')
        if profile.sig ≤ 0,05 then
            rCorr ← rCorr + profile.corr × rLen[l]
        end if
    end for
    simProfile[k] ← rCorr / sum(rLen)
end for
save(simProfile, "SimilitudPerfil.out")
```

podados, y la segunda toma las lista de los pares no podados con su grado de similitud y construye una matriz de adyacencia de un grafo bipartido. A partir de la lista de pares obtenida por ***MatrixMask2List***, la función ***calculate_profile_feature***, programada en *MATLAB*, calcula la similitud basada en el perfil físico-químico para las secuencias presente en la lista. Los alineamientos de los pares de secuencias son calculados de forma paralela usando la instrucción ***parfor*** del *MATLAB*. Esta función devuelve una lista con la similitud de cada par de secuencias, la cual se transforma en la matriz de similitud usando la aplicación ***List2Matrix***. La figura 2.10 muestra los pasos necesarios para el cálculo de la similitud basada en el perfil físico-químico de las proteínas.

2.5. Construcción y poda del grafo similitud

El grafo bipartido de la similitud entre las secuencias se construye a partir de la agregación de las diferentes medidas de similitud usando la media aritmética mostrada en la ecuación 2.10. El grafo se representa como una matriz de adyacencia, donde cada fila representa una secuencias del *Saccharomyces Cerevisiae*, cada columna una secuencia del *Schizosaccharomyces Pombe* y una celda representa la similitud entre las

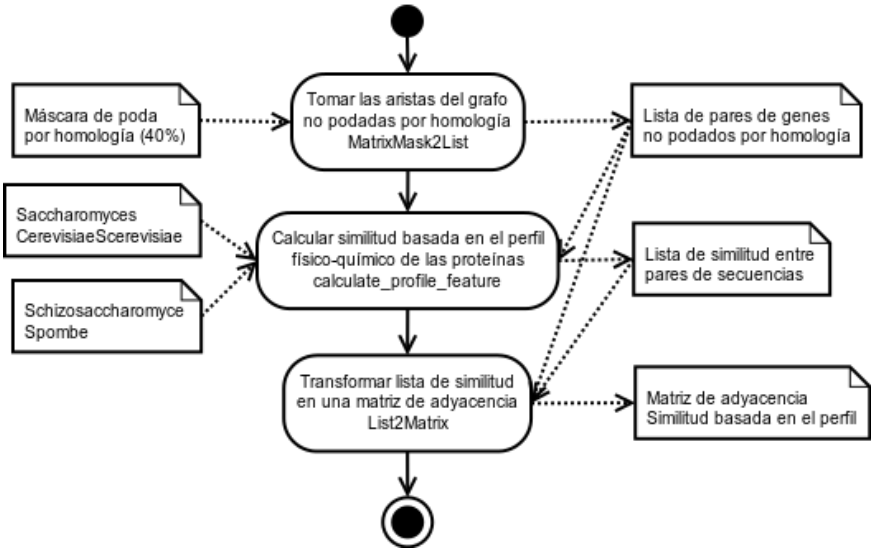


Figura 2.10.: Esquema general del cálculo de la similitud basada en el perfil físico-químico de las proteínas

secuencias representadas por la fila y la columna.

$$S(X_i, Y_j) = \frac{1}{4} \times \sum_{k=1}^{4} S_k(X_i, Y_j) \quad i = 1 \ldots n; j = 1 \ldots m; \tag{2.10}$$

Como se estudió en el capítulo anterior, el grafo bipartido completo es muy denso, lo que hace muy costoso su procesamiento. Con el objetivo de reducir la complejidad computacional sin perder información, se aplican políticas de poda para reducir la conectividad del grafo. En este trabajo se aplica una política de poda por homología de las secuencias, poda por umbral, poda para eliminar las ambigüedades del grafo teniendo en cuenta los bloques de orden conservado y poda para la asignación de los genes ortólogos a partir de los grupos obtenidos por el algoritmo de agrupamiento. Esta última política de poda es analizada en la sección 2.6.

La poda por homología de las secuencias elimina los pares de secuencias que no son homólogos entre sí, basándose en el porciento de similitud del alineamiento global de las dos secuencias calculado empleando la ecuación 2.3. Esta política de poda reduce significativamente la conectividad del grafo y determina los pares de genes a los que se les calcula la medida del perfil físico-químico de las proteínas. En la literatura referenciada en el capítulo anterior se consideran secuencias homólogas cuando tiene al menos un 35 % de similitud, valor que puede variar en dependencia de la distancia evolutiva que separa las especies comparadas. Como las especies empleadas para la

validación en este trabajo son relativamente cercanas en la evolución, se decidió emplear como límite un 40 % de similitud.

La poda por umbral, como se estudió en el capítulo anterior, busca eliminar todos los pares que su grado de similitud es muy pequeño respecto a los mejores grados de similitud del grafo. Se podan los valores por fila y columna, buscando las mejores correspondencias por cada fila y cada columna, para eliminar todos los pares que no rebasan el umbral de la mejor correspondencia por ambas dimensiones. En la literatura el umbral se define como el 80 % de las mejores correspondencias, valor que será tomado para el desarrollo de los experimentos de este trabajo.

El proceso de eliminación de las ambigüedades del grafo bipartido parte del cálculo de los bloques de orden conservado (Kamvysselis, 2003), usando las posiciones de las secuencias dentro de cada uno de los genomas. Inicialmente se buscan las correspondencias uno a uno dentro del grafo bipartido. Dos secuencias X_i y Y_j son correspondencias uno a uno, si existe una arista entre los nodos que representan las secuencias y no existe otra arista de esos nodos hacia otros nodos del grafo bipartido.

Una vez obtenidas las correspondencias uno a uno, se determinan las correspondencias cercanas entre sí, y se conforman los bloques de orden conservado. Dos relaciones se consideran cercanas si las secuencias que los componen están en el mismo cromosoma de cada genoma, en la misma hebra y la distancia intergénica entre las secuencias de cada genoma no supera los 20 000 pares de bases (Kamvysselis, 2003). Además una relación pertenece a un bloque de orden conservado si es cercana al menos a una relación de las que se encuentran en el bloque. Los bloques de orden conservado que tengan menos de tres pares de genes, no son tomados en cuenta, para evitar la confusión por el reordenamiento de genes aislados.

Una vez que se calculan todos los bloques de orden conservado se eliminan las ambigüedades del grafo bipartido. Para cada gen de ambos genomas se buscan los bloques de orden conservado a los cuales el gen es cercano, y se eliminan las aristas que conectan este gen con genes no cercanos a estos bloques.

Para la construcción del grafo bipartido se implementó en *Java* la aplicación ***AggregateMeasures***, que toma un conjunto de medidas de similitud y usando la media aritmética las agrega en una sola matriz de adyacencia. La mayor parte de los algoritmos realizan la poda directamente sobre el grafo, lo cual no permite reutilizar la información calculada en cada paso. En la solución propuesta la poda no se realiza directamente sobre el grafo, se crea una máscara de poda, que no es más que una matriz donde se marcan los pares podados, y luego se

combina con la matriz de adyacencia del grafo bipartido usando la aplicación ***Matrix2UndirectedPrunedMatrix*** programada en *Java*.

Para crear máscaras de poda por homología de las secuencias se usa la aplicación ***Matrix2MaskNonHomologs*** y para crear la máscara de poda por umbral se usa la aplicación ***Matrix2UndirectedMask***, ambas programadas en *Java*. A estas aplicaciones se les indica la matriz de adyacencia del grafo y los valores de umbral en cada caso, para obtener una máscara de poda. Los elementos podados en la máscara se marcan con el valor 2. Lo mismo ocurre cuando se combina la máscara con el grafo bipartido no podado. Las ambigüedades del grafo bipartido pueden ser podadas a través de la aplicación ***MatrixResolveAmbiguities*** o en el momento de transformar el grafo para ser agrupado, a través de la aplicación ***Matrix2MCL***, ambas desarrolladas en *Java*.

2.6. Agrupamiento sobre grafo bipartido y política de asignación de genes ortólogos

Para realizar el agrupamiento sobre el grafo bipartido se usa la implementación del algoritmo *MCL* disponible en los repositorios de *Ubuntu*. Como se estudió en el capítulo anterior este algoritmo depende del valor de inflación, que se indica con $-i$. Para esta investigación al igual que el algoritmo de *OrthoMCL* se tomó como valor de inflación 1.5, que según la bibliografía consultada produce buenos resultados.

Para poder agrupar el grafo usando el algoritmo *MCL* es necesario transformar la matriz de adyacencia del grafo bipartido pesado en un formato que acepte la implementación del algoritmo. Para esto se utiliza el formato de archivo de etiquetas expuesto en el manual de ayuda del *MCL* (van Dongen, 2011), donde cada par de genes se representa en una línea con tres elementos separados por espacios. El primero y segundo elemento representan los genes de cada uno de los genomas, enlazados por una arista, cuyo peso es indicado en el tercer elemento. Para diferenciar los genes de cada uno de los genomas se le añade el prefijo $G1\_$ a las etiquetas del genoma del *Saccharomyces Cerevisiae* y $G2\_$ a las etiquetas del *Schizosaccharomyces Pombe*. En *Java* se programó la aplicación ***Matrix2MCL*** la cual toma la matriz de adyacencia de un grafo bipartido pesado y la transforma en el formato antes explicado, excluyendo las aristas de los pares podados. En el momento en que se transforma la matriz de adyacencia, se puede pasar como parámetro que se eliminen las ambigüedades del grafo bipartido.

El algoritmo de agrupamiento devuelve un archivo, donde cada línea representa un

grupo y contiene los genes de ambos genomas separados por espacios que pertenecen a este grupo. Usando los genes que pertenecen a cada grupo se forman los pares de genes posibles, desechando los pares de genes podados antes del agrupamiento, y de esta forma se obtienen los grupos de homología. Sobre los grupos de homología compuestos por más de un par de genes se aplica la política de selección de genes ortólogos para determinar los pares que son ortólogos o no. Si no se usa política, todos los pares de genes son considerados ortólogos, lo que puede influir significativamente en el número de falsos positivos reportados por el algoritmo.

La política de asignación de genes ortólogos propuesta en este trabajo se basa en la poda de los pares de genes de los grupos de homología obtenidos por el algoritmo de agrupamiento. Los pares son podados teniendo en cuenta la longitud de las secuencias y el grado de similitud entre las secuencias que lo componen. Pares de secuencias muy cortas pueden ser beneficiados con un alto grado de similitud y pueden ser clasificados como ortólogos cuando no lo son. La política de asignación elimina los pares de secuencias donde una de sus secuencias no supere el 10 % de la longitud promedio de las secuencias y los pares cuyo grado de similitud no alcanza el 20 % del mayor grado de similitud del grupo de homología.

En *Java* se programó la aplicación ***MCL2GenesPair*** que transforma los grupos obtenidos con la salida del *MCL* en una lista de pares de genes ortólogos. A partir de cada grupo de homología se definen los pares de genes ortólogos usando la política de asignación antes mencionada. El uso de la política de asignación es un parámetro opcional que se le indica a la aplicación. La lista de pares de genes ortólogos se devuelve en un archivo de texto donde cada línea representa un par de genes, el primer elemento es un gen del *Saccharomyces Cerevisiae* y el segundo es un gen del *Schizosaccharomyces Pombe*.

2.7. Validación

Una vez obtenido el resultado del algoritmo de agrupamiento es necesario usar una medida de validación para determinar cuán semejante es la clasificación obtenida por el algoritmo, respecto a la clasificación obtenida por el algoritmo de *INPARANOID*. Para realizar esta validación se emplean las medidas de validación externas, a partir de la construcción de la tabla de contingencia como se definió en la tabla 1.1. Para la validación de los resultados de este trabajo se hace énfasis en el porciento de pares clasificados correctamente y en la medida de validación del índice de *Rand* ajustado, medida sensible a los falsos positivos.

En *Java* se programó la clase ***IndexMetrics***, que construye la tabla de contingencia con la lista de pares obtenidas por el algoritmo propuesto y la lista de pares del algoritmo de *INPARANOID*. Automáticamente se calculan diversas medidas de validación, entre las que se encuentran el índice de *Rand*, el índice de *Rand* ajustado, el índice de *Jaccard*, la precisión, la exactitud, el porciento de clasificación correcta, la sensibilidad, y la especificidad, todas definidas en la sección 1.7 del capítulo anterior. Se implementó la aplicación ***Validate2OrthoList*** en *Java*, para tomar la lista de genes obtenida de la salida del algoritmo de agrupamiento *MCL* y con la clase ***IndexMetrics*** calcular las medidas antes mencionadas.

2.8. Análisis del rendimiento del algoritmo

Para analizar el rendimiento del algoritmo propuesto, se debe tener en cuenta que el algoritmo sólo tiene dos secciones que ejecutan de forma paralela, por lo que sólo se analiza el rendimiento de estas dos secciones. La primera se corresponde con el cálculo de los alineamientos par a par de todas las secuencias usando el algoritmo 1, y la segunda con el cálculo de la similitud basada en el perfil físico-químico de las proteínas usando el algoritmo 2. A diferencia de los algoritmos secuenciales donde su rendimiento se evalúa por su tiempo de ejecución en función de la dimensión del problema, los algoritmos paralelos deben tener en cuenta una serie de factores como son el tamaño del problema, el número de procesadores y ciertos parámetros de comunicación de la plataforma sobre la que se ejecuta el algoritmo.

En (Kumar et al., 2003) se plantea que el rendimiento de un algoritmo paralelo puede ser analizado con una mayor exactitud cuando se compara éste con su mejor versión secuencial. Para la realización de este trabajo no se dispone de la mejor versión secuencial del algoritmo, por lo que se decidió emplear la versión secuencial del mismo en el propio *MATLAB*. En base al tiempo de ejecución de la versión secuencial T_S y el tiempo de la versión paralela T_P se usan las medidas: aceleración[5] o ganancia y eficiencia, definidas en (Kumar et al., 2003). La aceleración S calcula la relación entre los tiempos de ejecución de ambas versiones del algoritmo e indica cuán rápido se ejecuta el algoritmo paralelo respecto a su mejor versión secuencial usando P procesadores idénticos.

$$S = \frac{T_S}{T_P} \tag{2.11}$$

[5]En inglés es conocida como ***speedup***.

Cuando se calcula la aceleración del algoritmo paralelo respecto al algoritmo secuencial, usado valores de tiempo tomados de la experimentación, no se tienen en cuenta todos los aspectos involucrados, como la cantidad de P procesadores empleados en la ejecución. La eficiencia E calcula la relación entre la aceleración S y la cantidad de P procesadores usados en la ejecución del algoritmo. Esta métrica da una medida del grado de aprovechamiento de los recursos disponibles en el sistema.

$$E = \frac{S}{P} = \frac{T_S}{P \times T_P} \tag{2.12}$$

En la figura 2.11(a) y la figura 2.11(b) se muestran las versiones secuencial y paralela para el cálculo de los alineamientos par a par de secuencias. Las funciones ***swaling*** y ***nwalign*** del *MATLAB* tienen una complejidad computacional de $O(m \times n)$, donde m y n son las longitudes máximas de las secuencias de cada genoma, y como la longitud máxima que puede tener el alineamiento de las secuencias es $m + n$ entonces la función ***calculate_similarity*** tiene una complejidad de $O(m+n)$, siendo entonces la complejidad computacional de la versión secuencial $O(N \times m \times n)$. Los tiempos secuencial y paralelo correspondientes al cálculo de los alineamiento serán referenciados como T_{S1} y T_{P1}.

```
for j = 1:N
  d_nw = swalign(seq, seqs2(j).seq,'ScoringMatrix',scoring_matrix,...
      'GapOpen',gap_open,'ExtendGap',gap_extend);
  score_sw(counter,j)=d_nw;
  [d_nw, align_value] = nwalign(seq, seqs2(j).seq, ...
      'ScoringMatrix',scoring_matrix,'GapOpen',gap_open,...
      'ExtendGap',gap_extend);
  score_nw(counter,j)=d_nw;
  similarity_nw(counter,j) = calculate_similarity(align_value);
end;
```

(a) Cálculo de los alineamientos de forma secuencial

```
parfor j = 1:N
  d_nw = swalign(seq, seqs2(j).seq,'ScoringMatrix',scoring_matrix,...
      'GapOpen',gap_open,'ExtendGap',gap_extend);
  score_sw(counter,j)=d_nw;
  [d_nw, align_value] = nwalign(seq, seqs2(j).seq, ...
      'ScoringMatrix',scoring_matrix,'GapOpen',gap_open,...
      'ExtendGap',gap_extend);
  score_nw(counter,j)=d_nw;
  similarity_nw(counter,j) = calculate_similarity(align_value);
end;
```

(b) Cálculo de los alineamientos de forma paralela

Figura 2.11.: Versión secuencial y paralela del cálculo de los alineamientos de secuencias

En la figura 2.12(a) y la figura 2.12(b) se muestran las versiones secuencial y paralela para el cálculo del perfil físico-químico de las proteínas. La función ***nwalign*** del

MATLAB tiene una complejidad computacional de $O(m \times n)$, y como la longitud máxima que puede tener el alineamiento de las secuencias es $m + n$ entonces la función ***calculate_profile_feature*** tiene una complejidad de $O((m+n)^2)$, siendo entonces la complejidad computacional de la versión secuencial $O(N \times (m+n)^2)$. Los tiempos secuencial y paralelo correspondientes al cálculo del perfil físico-químico serán referenciados como T_{S2} y T_{P2}.

```
for k = 1:N
  p = pares_genes(k,:);
  s1 = p(1);
  s2 = p(2);
  [d_nw, align_value] = nwalign(seqs1(s1).seq, seqs2(s2).seq,...
      'ScoringMatrix',scoring_matrix,'GapOpen',gap_open,...
      'ExtendGap',gap_extend);
  seq1 = align_value(1,1:end);
  seq2 = align_value(3,1:end);
  profile = calculate_profile_feature(seq1, seq2, [3 5 7]);
  profile3(k) = profile(1);
  profile5(k) = profile(2);
  profile7(k) = profile(3);
end;
```

(a) Cálculo del perfil de forma secuencial

```
parfor k = 1:N
  p = pares_genes(k,:);
  s1 = p(1);
  s2 = p(2);
  [d_nw, align_value] = nwalign(seqs1(s1).seq, seqs2(s2).seq,...
      'ScoringMatrix',scoring_matrix,'GapOpen',gap_open,...
      'ExtendGap',gap_extend);
  seq1 = align_value(1,1:end);
  seq2 = align_value(3,1:end);
  profile = calculate_profile_feature(seq1, seq2, [3 5 7]);
  profile3(k) = profile(1);
  profile5(k) = profile(2);
  profile7(k) = profile(3);
end;
```

(b) Cálculo del perfil de forma paralela

Figura 2.12.: Versión secuencial y paralela del cálculo del perfil físico-químico de las proteínas

Teniendo en cuenta la complejidad computacional y la estructura de los algoritmos secuenciales, los tiempos de ejecución secuencial se pueden estimar como:

$$T_{S1} = N \times (2 \times m \times n + m + n) \times t_c \tag{2.13}$$

$$T_{S2} = N \times (m \times n + (m+n)^2) \times t_c \tag{2.14}$$

donde t_c indica el tiempo requerido para realizar las operaciones aritméticas. Para el desarrollo de este trabajo asumiremos el valor de t_c óptimo, $t_c = 1$. En (Kumar et al., 2003) se define el tiempo de ejecución paralelo T_P en función del tiempo de cálculo y

el tiempo de comunicación entre los procesadores. En el caso de la versión paralela el tiempo de cálculo viene dado por la distribución de los N ciclos entre los P procesadores empleados. Con la distribución de los ciclos, pueden darse los casos en que: todos los procesadores realizan $\lfloor N/P \rfloor$ iteraciones, o $p < P$ procesadores realizan $\lfloor N/P \rfloor + 1$ iteraciones; por lo que tomando el peor de los casos, el tiempo de cálculo de cada una de las versiones paralelas se puede estimar como:

$$T_{Cal1} = (\lfloor N/P \rfloor + 1) \times (2 \times m \times n + m + n) \times t_c \tag{2.15}$$

$$T_{Cal2} = (\lfloor N/P \rfloor + 1) \times (m \times n + (m + n)^2) \times t_c \tag{2.16}$$

El tiempo de comunicación entre los procesadores está directamente relacionado con la información accedida en la sección secuencial del algoritmo. En los dos casos que se analizan, en cada iteración cada uno de los P procesadores necesita recibir la información de las secuencias que se alinean. Siendo m y n las longitudes de las dos secuencias, el tiempo de comunicación se expresa como:

$$T_{Com} = (\lfloor N/P \rfloor + 1) \times (t_s + t_w \times P \times (m + n)) \tag{2.17}$$

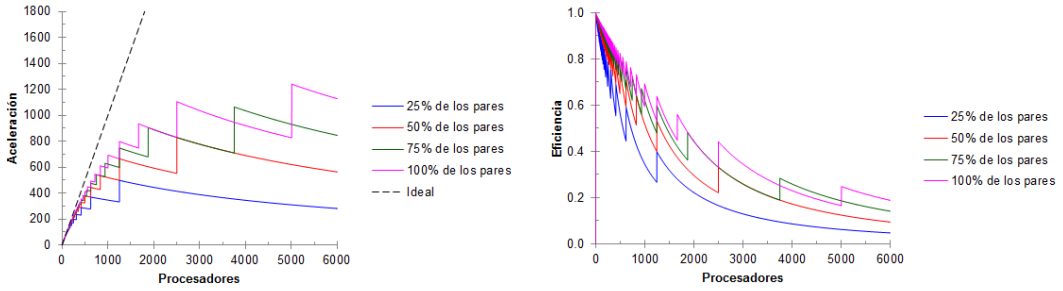
donde t_s representa el tiempo necesario para establecer la comunicación y preparar la información a enviar; y t_w el tiempo necesario para enviar un valor numérico. Para el desarrollo de este trabajo asumiremos estos valores óptimos, $t_s = 0$ y $t_w = 1$. Combinando la información de las ecuaciones 2.15 y 2.17 se obtiene una estimación del tiempo de ejecución paralelo para el cálculo del alineamiento y combinando las ecuaciones 2.16 y 2.17 se obtiene una estimación del tiempo de ejecución paralelo para el cálculo del perfil.

$$T_{P1} = (\lfloor N/P \rfloor + 1) \times (2 \times m \times n + (m + n) \times (P + 1)) \tag{2.18}$$

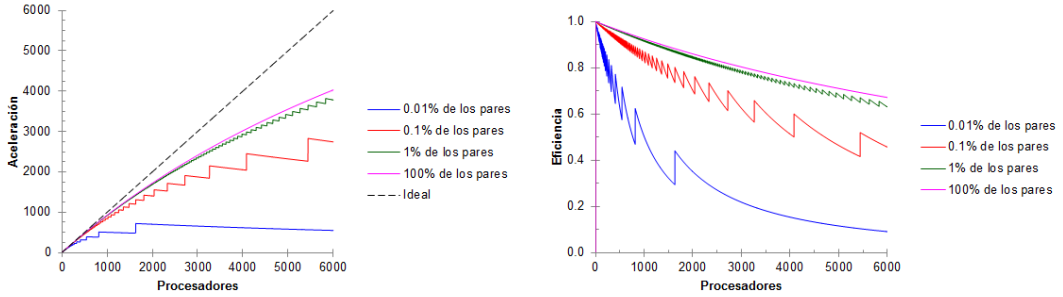
$$T_{P2} = (\lfloor N/P \rfloor + 1) \times (m \times n + (m + n) \times (m + n + P)) \tag{2.19}$$

Empleando las expresiones T_S y T_P para cada uno de los algoritmos, se realizó un análisis del comportamiento de la aceleración y la eficiencia para diferentes muestras de N, variando la cantidad de procesadores entre 1 y 6000. En la figura 2.13 se muestran las gráficas de los resultados obtenidos. Para el caso del alineamiento, el 100 % de la muestra está dado por las 5006 secuencias del genoma del *Schizosaccharomyces Pombe*, y para el perfil, por 16324500 pares de secuencias. Se puede observar que al mantener constante el tamaño de la muestra y aumentar la cantidad de procesadores, el valor de la aceleración y la eficiencia tienden a disminuir. No siendo así cuando se mantiene

constante la cantidad de procesadores y se aumenta la muestra. Cuando la cantidad de procesadores alcanza el tamaño de la muestra, la aceleración y la eficiencia comienzan a decrecer hasta alcanzar un valor constante.



(a) Cálculo del alineamiento de secuencias



(b) Cálculo del perfil físico-químico

Figura 2.13.: Comportamiento estimado de la aceleración y la eficiencia en el cálculo paralelo de los alineamientos par a par de secuencias y el perfil físico-químico de las proteínas

En ocasiones en que es casi imposible la ejecución secuencial del algoritmo, estos valores estimados pueden dar un valor cercano a la realidad. El cálculo de las medidas de similitud empleando ambos algoritmos paralelos fue realizado usando 6 procesadores, por lo que el algoritmo paralelo para el alineamiento se ejecutó aproximadamente $5,98$ veces más rápido que la versión secuencial, con un eficiencia de $0,9968$; y el algoritmo paralelo para el perfil se ejecutó aproximadamente $5,9971$ veces más rápido que la versión secuencial, con una eficiencia de $0,9995$.

A partir de los tiempos de ejecución en paralelo de las ecuaciones 2.18 y 2.19, se pueden estimar las complejidades computaciones de cada uno de los algoritmos cuando su ejecución es en paralelo. En ambos casos, la expresión $(\lfloor N/P \rfloor + 1)$ se puede tomar como N/P. En el caso del algoritmo 1, los elementos $2 \times m \times n$ y $(m + n) \times (P + 1)$ no se pueden comparar debido a la presencia de P, omitiendo los valores de las constantes la complejidad temporal del algoritmo 1 está dada

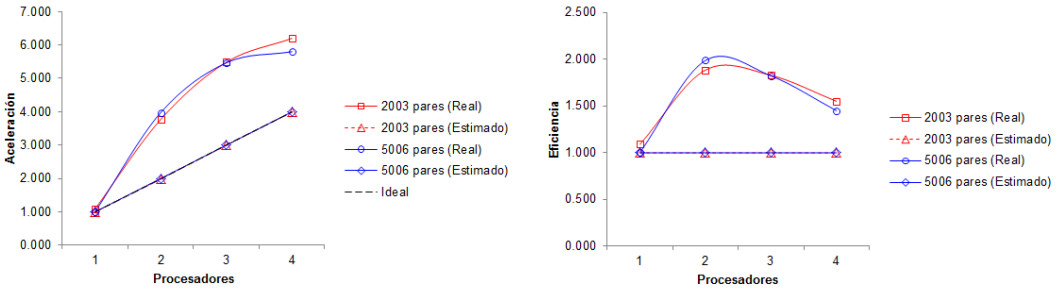
como $O\left(\frac{N \times m \times n}{P} + N \times (m+n)\right)$, la cual es menor que $O(N \times m \times n)$, como se muestra en la ecuación 2.20. Por otra parte en el algoritmo 2 se tiene que $m \times n \leq (m+n) \times (m+n+P)$, por lo que la complejidad temporal del algoritmo 2 es $O\left(\frac{N \times (m+n)^2}{P} + N \times (m+n)\right)$, la cual es menor que $O(N \times (m+n)^2)$ como se muestra en la ecuación 2.21.

$$\frac{\frac{N \times m \times n}{P} + N \times (m+n)}{N \times m \times n} = \frac{1}{P} + \frac{m+n}{m \times n} \leq 0 \tag{2.20}$$

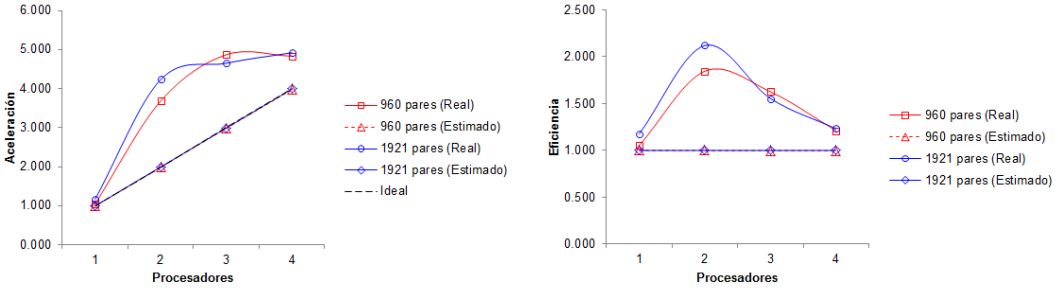
$$\frac{\frac{N \times (m+n)^2}{P} + N \times (m+n)}{N \times (m+n)^2} = \frac{1}{P} + \frac{1}{m+n} \leq 0 \tag{2.21}$$

Usando las versiones secuenciales y paralelas de cada uno de los algoritmos, se calculó el tiempo de ejecución secuencial y los tiempos de ejecución en paralelo usando 1, 2, 3 y 4 procesadores. Para esto se utilizó una computadora con procesador *Intel® Core™ i3 CPU M380* a 2.53 *GHz*, con una memoria *RAM DDR3* de 4,0 *Gb*, con *MATLAB(R2010a)* sobre el sistema operativo *Windows 7 de 64Bits*. Los experimentos fueron realizados sobre el 50 % de los pares y el conjunto total de los pares. Para el caso del cálculo del perfil se realizó una selección aleatoria de 1921 pares que representan el 100 %. La figura 2.14 muestra los resultados obtenidos. Como se observa en esta figura los valores estimados tienen un comportamiento similar entre sí, al igual que los valores reales calculados a partir de los tiempos de ejecución. Los mejores valores de aceleración se obtiene para le ejecución sobre 4 procesadores, pero con una mayor eficiencia cuando la ejecución fue sobre 2 procesadores.

Teóricamente el valor de la aceleración se encuentra acotado por la cantidad de procesadores empleados en la ejecución, y la eficiencia oscila en el intervalo $[0, 1]$. En este caso la aceleración es superior a la cantidad de procesadores empleados en cada ejecución y el valor de eficiencia es superior a 1. A este fenómeno se le conoce como aceleración súper lineal, es poco común, y se produce porque cada elemento de procesamiento consume un tiempo inferior a la razón T_S/P. Esto puede estar debido a que la versión paralela realiza menos trabajo que la versión secuencial, o a factores relacionados con los recursos que se emplean, como por ejemplo, la memoria caché.



(a) Cálculo del alineamiento de secuencias



(b) Cálculo del perfil físico-químico

Figura 2.14.: Comportamiento experimental de la aceleración y la eficiencia en el cálculo paralelo de los alineamientos par a par de secuencias y el perfil físico-químico de las proteínas

Consideraciones finales del capítulo

En este capítulo fue analizada la transformación realizada sobre los datos obtenidos del sitio de *INPARANOID*, para su empleo por el algoritmo propuesto. Se conformó un archivo *FASTA* para el genoma de *Saccharomyces Cerevisiae* con 5861 genes y un archivo con 5006 genes del *Schizosaccharomyces Pombe*. La lista de índices de pares de genes ortólogos contiene 3807 relaciones de pares ortólogos.

Se definieron cuatro medidas de similitud empleadas para la construcción del grafo bipartido, la medida basada en el alineamiento de secuencias, la basada en la longitud de las secuencias, la basada en la pertenencia a los bloques *LCB* y la basada en el perfil físico-químico de las proteínas. La medida basada en el alineamiento de secuencias se definió como la combinación de las puntuaciones del alineamiento local y el alineamiento global. Además se detallaron las políticas de poda realizadas sobre el grafo bipartido y los elementos necesarios para la ejecución del algoritmo de agrupamiento *MCL*, así como las medidas de validación del agrupamiento.

Con el uso de la instrucción ***parfor*** para el cálculo en paralelo de los alineamientos par a par de secuencias y el perfil físico-químico de las proteínas, se logra

reducir la complejidad computacional del cálculo de estas medidas. La complejidad computacional del cálculo del alineamiento de secuencias se reduce de un $O(N \times m \times n)$ a un $O\left(\frac{N \times m \times n}{P} + N \times (m+n)\right)$; y la complejidad computacional del cálculo del perfil físico-químico se reduce de un $O(N \times (m+n)^2)$ a un $O\left(\frac{N \times (m+n)^2}{P} + N \times (m+n)\right)$.

CAPÍTULO 3

EXPERIMENTACIÓN Y RESULTADOS

En este capítulo se describen los experimentos realizados para la validación del algoritmo propuesto, con los datos obtenidos del sitio de *INPARANOID*. Se analizan los resultados obtenidos en cada uno de los experimentos teniendo en cuenta el porciento de clasificación correcta y la medida de calidad *ARI*. También se comparan los resultados con modelos de regresión logística obtenidos en el *SPSS*.

3.1. Diseño de experimentos

Para la validación del algoritmo propuesto en este trabajo, usando los datos del *Saccharomyces Cerevisiae* y *Schizosaccharomyces Pombe*, se diseñaron cuatro modelos de alineamiento. Los modelos de alineamiento se diferencian en los parámetros empelados para el cálculo de los alineamiento locales y globales. En la tabla 3.1 se muestran los modelos definidos y los valores de matrices de sustitución y penalización de *"gaps"* empleados para el cálculo de los alineamientos.

Tabla 3.1.: Modelos de alineamiento para la ejecución del algoritmo

Nombre del modelo	Matriz de sustitución	Penalización de *"gaps"* (abertura)	Penalización de *"gaps"* (extendido)
blosum50	BLOSUM50	15	8
blosum62_1	BLOSUM62	8	7
blosum62_2	BLOSUM62	12	6
pam250	PAM250	10	8

3.1.1. Alineamiento de secuencias

Para verificar la validez de la medida de similitud basada en el alineamiento de secuencias definida en el capítulo anterior, se realizaron una serie de experimentos con las medidas del alineamiento de secuencias para observar su influencia en la clasificación de genes ortólogos. Se definieron seis rasgos basados en el alineamiento de secuencias: puntuación del alineamiento global, puntuación del alineamiento local, porciento de similitud global, porciento de similitud local y la combinación de las puntuaciones y los porcientos de similitud. A cada uno de estos rasgos se le aplicaron algoritmos de detección de genes ortólogos basados en *RBH* y *BUS*, implementados en *Java*, así como el algoritmo propuesto en este trabajo. La tabla 3.2 muestra los porcientos de clasificación y valores de la medida de calidad *ARI*, obtenidos para los alineamientos del modelo *blosum50*.

Tabla 3.2.: Experimentos usando alineamientos del modelo *blosum50*

	ARI			% de clasificación		
Rasgo	RBH	BUS	MCL	RBH	BUS	MCL
Alineamiento global	0,712	0,760	0,762	57,81 %	71,32 %	74,36 %
Alineamiento local	0,795	0,668	0,711	73,86 %	91,10 %	96,19 %
Similitud global	0,736	0,506	0,010	64,17 %	80,48 %	0,66 %
Similitud local	0,000	0,000	0,002	0,13 %	0,13 %	0,32 %
Alineamiento global+local	0,786	0,672	0,773	90,54 %	90,54 %	95,35 %
Similitud global+local	0,304	0,195	0,008	24,8 %	32,20 %	0,47 %

Como se puede observar la puntuación del alineamiento local es el rasgo que produce un mayor porciento de clasificación, seguido del rasgo donde se combina la puntuación del alineamiento local y global. La diferencia de pares clasificados por ambos rasgos no alcanza el 1 %, con lo que se puede afirmar que existe una diferencia de menos de 38 pares de secuencias. Observando los valores obtenidos de la medida de calidad se puede ver que la combinación de las puntuaciones tiene un mejor desempeño en la clasificación de forma general, debido a que se reduce considerablemente la cantidad de falsos positivos.

En la tabla 3.3 se muestran los valores del porciento de clasificación y la medida de calidad luego de aplicar los algoritmos de detección de ortólogos al alineamiento obtenido del modelo *blosum62_1*. Se puede ver que en este modelo se tiene un comportamiento similar al anterior, donde los rasgos más significativos continúan siendo la puntuación del alineamiento local y la combinación de ésta con la puntuación del alineamiento global. En este caso la diferencia de pares clasificados sigue siendo inferior

a 38, pues no se alcanza un 1 % de diferencia.

Tabla 3.3.: Experimentos usando alineamientos del modelo *blosum62_1*

	ARI			% de clasificación		
Rasgo	RBH	BUS	MCL	RBH	BUS	MCL
Alineamiento global	0,720	0,754	0,762	59,78 %	73,71 %	76,73 %
Alineamiento local	0,797	0,659	0,712	73,92 %	91,15 %	95,74 %
Similitud global	0,739	0,502	0,007	63,75 %	80,12 %	0,39 %
Similitud local	0,000	0,000	0,002	0,13 %	0,16 %	0,26 %
Alineamiento global+local	0,790	0,663	0,769	73,31 %	90,33 %	95,17 %
Similitud global+local	0,297	0,189	0,002	23,85 %	31,47 %	0,11 %

Los resultados obtenidos de aplicar los algoritmos de detección de genes ortólogos al modelo *blosum62_2* se muestran en la tabla 3.4, teniendo el mismo comportamiento que los casos anteriores. La diferencia de pares clasificados sigue siendo inferior a 38 pares de genes.

Tabla 3.4.: Experimentos usando alineamientos del modelo *blosum62_2*

	ARI			% de clasificación		
Rasgo	RBH	BUS	MCL	RBH	BUS	MCL
Alineamiento global	0,711	0,759	0,760	57,11 %	70,19 %	73,21 %
Alineamiento local	0,797	0,671	0,725	74,07 %	91,04 %	95,88 %
Similitud global	0,737	0,499	0,010	63,44 %	79,67 %	0,68 %
Similitud local	0,000	0,000	0,001	0,13 %	0,13 %	0,08 %
Alineamiento global+local	0,789	0,674	0,764	73,39 %	90,41 %	95,17 %
Similitud global+local	0,306	0,190	0,004	24,56 %	31,60 %	0,21 %

Los resultados obtenidos de la clasificación sobre el modelo *pam250* sigue teniendo el mismo comportamiento que el caso anterior, como se puede observar en la tabla 3.5. La diferencia de pares clasificados continúa siendo inferior a los 38 pares de genes.

En todos los casos sólo resultó significativo el rasgo basado en la puntuación del alineamiento local y el rasgo que combina la puntuación local con la global. La diferencia del porciento de clasificación estuvo dada en los cuatro casos por menos de 38 pares de genes, mientras, que la diferencia de la medida de calidad fue más significativa, producto de la reducción de los falsos positivos usando la segunda medida. Por lo tanto se puede afirmar que la combinación de las puntuaciones del alineamiento local y global puede conducir a obtener mejores resultados en los algoritmos de detección de genes ortólogos.

Tabla 3.5.: Experimentos usando alineamientos del modelo *pam250*

	ARI			% de clasificación		
Rasgo	RBH	BUS	MCL	RBH	BUS	MCL
Alineamiento global	0,688	0,623	0,699	59,86 %	74,10 %	76,57 %
Alineamiento local	0,801	0,646	0,700	74,00 %	91,59 %	95,98 %
Similitud global	0,690	0,445	0,006	57,29 %	72,66 %	0,37 %
Similitud local	0,001	0,002	0,013	0,60 %	1,31 %	1,34 %
Alineamiento global+local	0,789	0,653	0,761	73,13 %	91,28 %	95,27 %
Similitud global+local	0,356	0,244	0,000	28,97 %	40,35 %	0,00 %

3.1.2. Perfil físico-químico de las proteínas

Para el cálculo de la medida de similitud basada en el perfil físico-químico de las proteínas, no se dispone de un valor óptimo o recomendado para el tamaño de ventana. Por este motivo se decidió calcular la medida de similitud para los tamaños de ventana 3, 5 y 7. Para determinar cuál valor resulta más significativo en la clasificación de genes ortólogos se realizaron dos pruebas de verificación para cada tamaño de ventana.

La primera prueba de verificación realizada sobre los valores calculados fue la aplicación de el algoritmo propuesto, para determinar el tamaño de ventana que produce mejores resultados en la clasificación de genes ortólogos. La figura 3.1 muestra los porcientos de pares de genes ortólogos clasificados correctamente usando los tamaños de ventana en cada uno de los modelos de alineamiento.

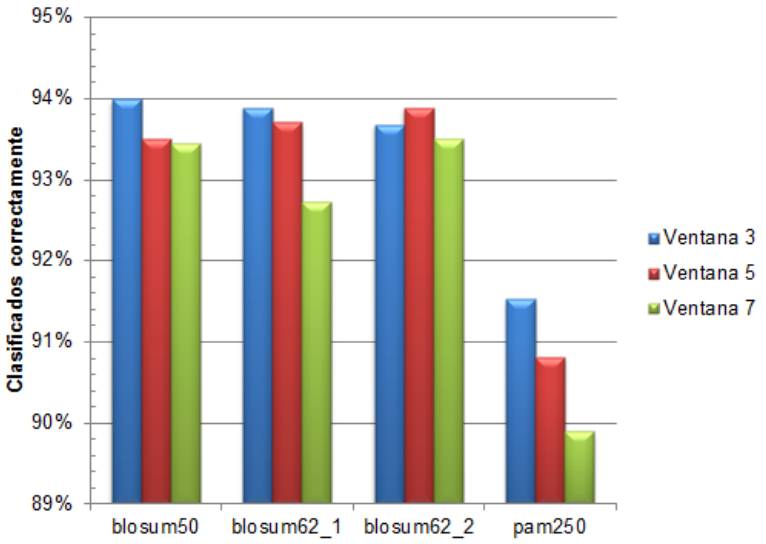


Figura 3.1.: Porciento de clasificación correcta de los pares de genes ortólogos para la selección del tamaño de ventana del perfil, primera prueba

Como se puede observar, los mejores resultados se obtienen cuando se usa un tamaño de ventana 3, excepto en modelo *blosum62_2* que se obtienen para un tamaño de ventana 5. Las diferencia de pares clasificados es inferior a 38 pares, debido a que la diferencia no alcanza el 1 %. Analizando los valores obtenidos de la medida de calidad mostrados en la figura 3.2, se observa que se obtiene un mejor balance de falsos positivos cuando se usa un tamaño de ventana 3.

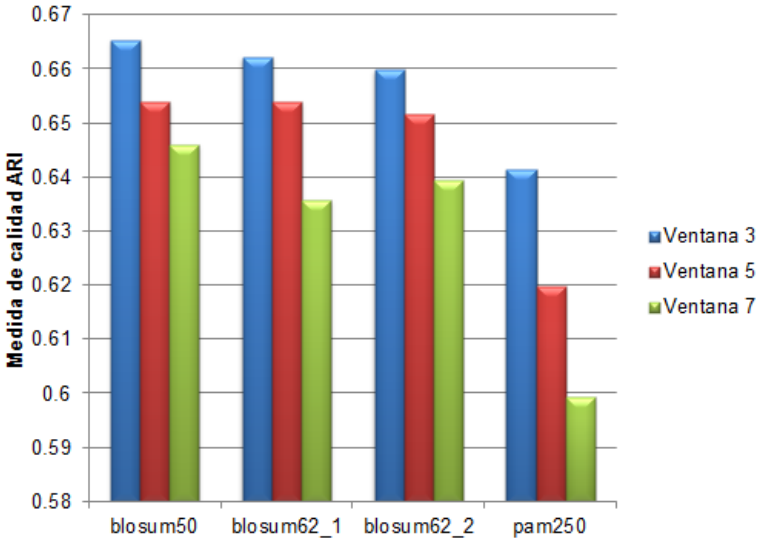


Figura 3.2.: Medida de calidad obtenida para la selección del tamaño de ventana del perfil, primera prueba

La segunda prueba aplicada para la selección del tamaño de ventana del perfil, fue la combinación de la medida de similitud para cada tamaño de ventana con la medida de similitud del alineamiento, la aplicación de una máscara de poda por homología al 40 % y una poda por umbral de un 80 % a la medida de similitud del alineamiento de las secuencias para conformar el grafo bipartido. La tabla 3.6 muestra los resultados obtenidos de la clasificación aplicando el algoritmo de agrupamiento *MCL*.

Tabla 3.6.: Resultados del agrupamiento para diferentes tamaños de ventana

	Ventana 3		Ventana 5		Ventana 7	
	%	ARI	%	ARI	%	ARI
blosum50	94,56	0,6785	94,35	0,6678	93,93	0,6570
blosum62_1	94,51	0,6748	93,96	0,6667	93,72	0,6501
blosum62_2	94,38	0,6731	94,19	0,6624	94,46	0,6551
pam250	92,72	0,6681	92,43	0,6522	91,26	0,6193

Como se puede observar se obtiene un mejor porciento de clasificación de pares de genes ortólogos para el tamaño de ventana 3, excepto en el modelo *blosum62_2*, donde

se obtiene para el tamaño de ventana 7. La figura 3.3 muestra la representación de los porcientos de clasificación obtenidos para cada tamaño de ventana. Analizando los valores de la tabla anterior se puede notar que la diferencia de clasificación en el modelo *blosum62_2* usando tamaño de ventana 3 y 7 es inferior a 1 %, por lo que la cantidad de pares de diferencia es inferior a 38 pares.

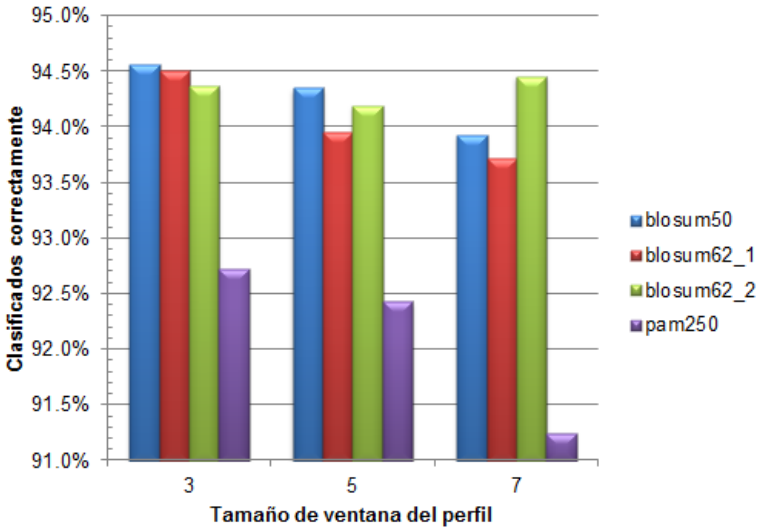


Figura 3.3.: Porciento de clasificación correcta de los pares de genes ortólogos para la selección del tamaño de ventana del perfil, segunda prueba

Analizando los valores obtenidos de la medida de calidad mostrados en la figura 3.4 se observa que se obtiene mejor balance de falsos positivos cuando se usa un tamaño de ventana 3.

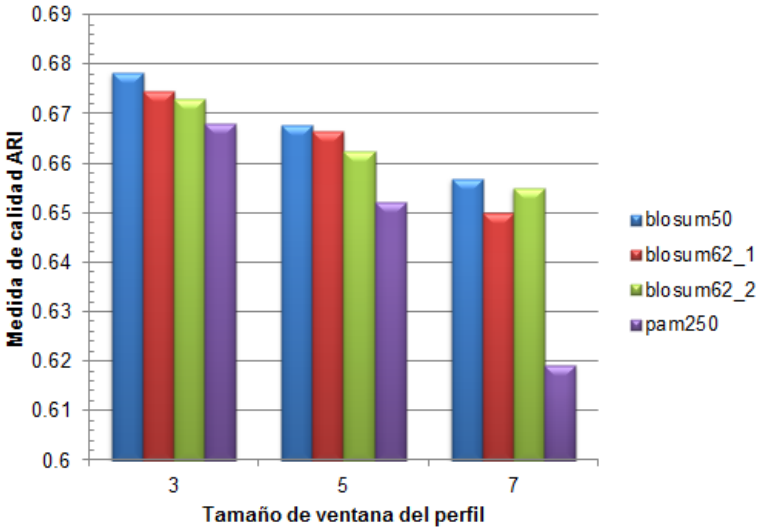


Figura 3.4.: Medida de calidad obtenida para la selección del tamaño de ventana del perfil, segunda prueba

Basados en los resultados obtenidos de las dos pruebas analizadas, se puede concluir que el tamaño de ventana 3 para el cálculo del perfil debe producir resultados más significativos que el tamaño de ventana 5 y 7. Por lo que se decide emplear éste para la validación del algoritmo empleando los experimento que se exponen en la siguiente sección.

3.2. Experimentos

Para el análisis de los datos, se realizaron ocho experimentos, sobre cada uno de los modelos de alineamiento. En los experimentos se combinan las medidas de similitud definidas en el capítulo anterior, usando la media aritmética. Las medidas de similitud combinadas son la medida de similitud basada en el alineamiento de secuencias, la basada en la longitud de las secuencias, la basada en la pertenencia a los bloques LCB y la basada en la información del perfil físico-químico de las proteínas. A continuación se explican la estructura de cada experimento y se analizan los resultados obtenidos de la clasificación a través del porciento de pares de genes ortólogos clasificados correctamente y la medida de validación ARI.

3.2.1. Experimento 1

El experimento 1 se basa en la medida de similitud basada en el alineamiento de secuencias. Se aplica una máscara de poda por homología al 40 % y una poda por umbral de un 80 %. La tabla 3.7 muestra los resultados obtenidos luego de aplicar el algoritmo de agrupamiento y seleccionar los pares de genes ortólogos.

Tabla 3.7.: Tabla de contingencia del experimento 1

	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3630	1951	177	29334408	0,7733
blosum62_1	3623	1994	184	29334365	0,7689
blosum62_2	3623	2057	184	29334302	0,7637
pam250	3627	2097	180	29334262	0,7611

Como se puede observar, los cuatro modelos clasifican aproximadamente la misma cantidad de pares de genes con una diferencia máxima de 7 pares. El modelo *blosum50*, es el que produce mejores resultados, clasificando correctamente 3630 pares de genes ortólogos, dejando de clasificar 177, y clasificando incorrectamente 1951 pares de genes.

En la figura 3.5 se muestra el porciento de clasificación correcta de los pares de genes ortólogos para cada uno de los modelos.

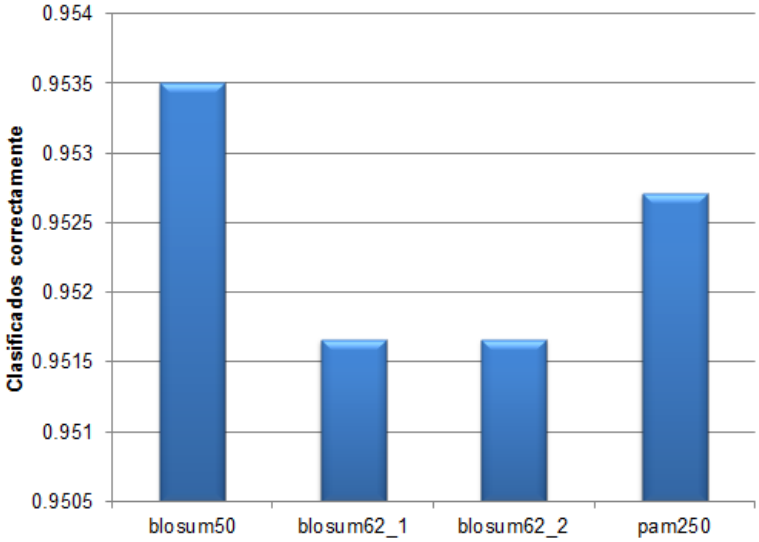


Figura 3.5.: Porciento de clasificación correcta de los pares de genes ortólogos en el experimento 1

3.2.2. Experimento 2

El experimento 2 se basa en la combinación de la medida de similitud basada en el alineamiento de secuencias con la medida de similitud basada en el perfil físico-químico de las proteínas. Se aplica una máscara de poda por homología al 40 % y una poda por umbral de un 80 % a la medida de similitud del alineamiento de las secuencias. La tabla 3.8 muestra los resultados obtenidos luego de aplicar el algoritmo de agrupamiento y seleccionar los pares de genes ortólogos.

Tabla 3.8.: Tabla de contingencia del experimento 2

	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3600	3204	207	29333155	0,6785
blosum62_1	3598	3258	209	29333101	0,6748
blosum62_2	3593	3275	214	29333084	0,6731
pam250	3530	3230	277	29333129	0,6681

Como se puede observar, el modelo *blosum50* obtiene los mejores resultados, pero no superan los valores obtenidos en el experimento 1. La figura 3.6 muestra el porciento de clasificación correcta de los pares de genes ortólogos para cada uno de los modelos.

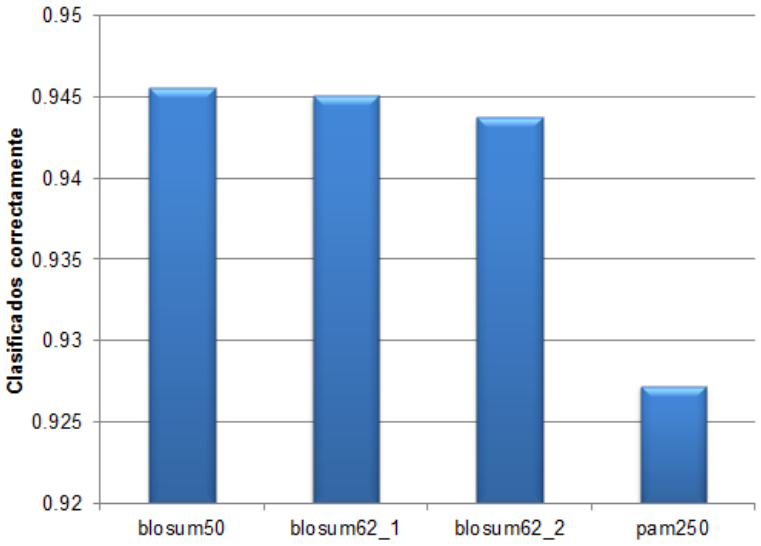


Figura 3.6.: Porciento de clasificación correcta de los pares de genes ortólogos en el experimento 2

3.2.3. Experimento 3

El experimento 3 se basa en la combinación de la medida de similitud basada en el alineamiento de secuencias con la medida de similitud basada en la longitud de las secuencias. Se aplica una máscara de poda por homología al 40 % y una poda por umbral de un 80 % a la medida de similitud del alineamiento de las secuencias. La tabla 3.9 muestra los resultados obtenidos luego de aplicar el algoritmo de agrupamiento y seleccionar los pares de genes ortólogos.

Tabla 3.9.: Tabla de contingencia del experimento 3

	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3361	6778	446	29329581	0,4819
blosum62_1	3317	7331	490	29329028	0,4588
blosum62_2	3352	6916	455	29329443	0,4762
pam250	3195	7343	612	29329016	0,4453

Como se puede observar, el modelo *blosum50* obtiene los mejores resultados, pero no superan los valores obtenidos en el experimento 1 y 2. La figura 3.7 muestra el porciento de clasificación correcta de los pares de genes ortólogos para cada uno de los modelos.

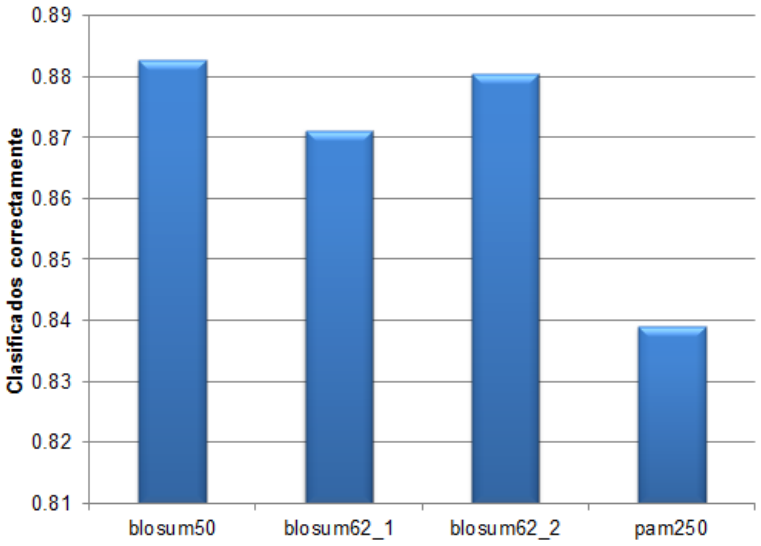


Figura 3.7.: Porciento de clasificación correcta de los pares de genes ortólogos en el experimento 3

3.2.4. Experimento 4

El experimento 4 se basa en la combinación de la medida de similitud basada en el alineamiento de secuencias con la medida de similitud basada en la pertenencia a los bloques *LCB*. Se aplica una máscara de poda por homología al 40 % y una poda por umbral de un 80 % a la medida de similitud del alineamiento de las secuencias. La tabla 3.10 muestra los resultados obtenidos luego de aplicar el algoritmo de agrupamiento y seleccionar los pares de genes ortólogos.

Tabla 3.10.: Tabla de contingencia del experimento 4

	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3207	1929	600	29334430	0,7172
blosum62_1	3211	2119	596	29334240	0,7028
blosum62_2	3251	1931	556	29334428	0,7233
pam250	3019	2188	788	29334171	0,6698

Como se puede observar, en este caso el modelo *blosum62_2* obtiene los mejores resultados, pero son inferiores a los resultados de los experimentos anteriores. La figura 3.8 muestra el porciento de clasificación correcta de los pares de genes ortólogos para cada uno de los modelos.

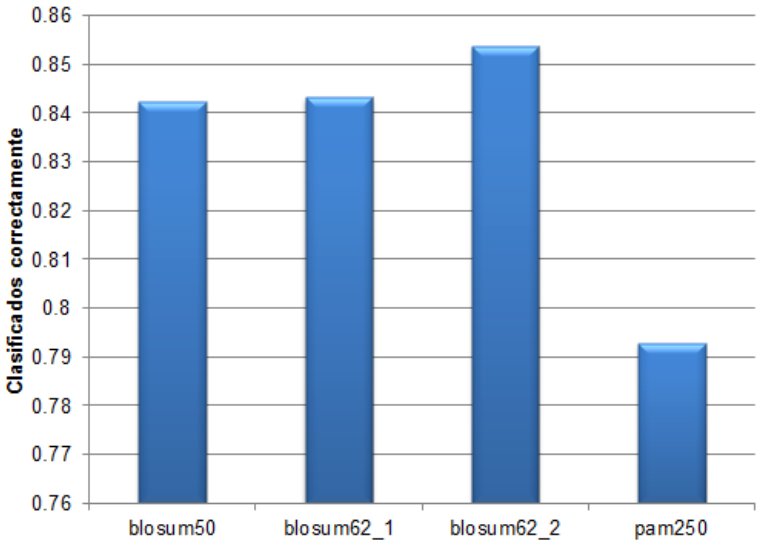


Figura 3.8.: Porciento de clasificación correcta de los pares de genes ortólogos en el experimento 4

3.2.5. Experimento 5

El experimento 5 se basa en la combinación de la medida de similitud basada en el alineamiento de secuencias con la medida de similitud basada en el perfil físico-químico de las proteínas y la medida de similitud basada en la longitud de las secuencias. Se aplica una máscara de poda por homología al 40 % y una poda por umbral de un 80 % a la medida de similitud del alineamiento de las secuencias. La tabla 3.11 muestra los resultados obtenidos luego de aplicar el algoritmo de agrupamiento y seleccionar los pares de genes ortólogos.

Tabla 3.11.: Tabla de contingencia del experimento 5

	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3523	6288	284	29330071	0,5173
blosum62_1	3472	6886	335	29329473	0,4901
blosum62_2	3495	6469	312	29329890	0,5075
pam250	3375	7086	432	29329273	0,4730

Como se puede observar, el modelo *blosum50* vuelve a obtener los mejores resultados, pero continúa sin superar los valores obtenidos en los experimentos anteriores. La figura 3.9 muestra el porciento de clasificación correcta de los pares de genes ortólogos para cada uno de los modelos.

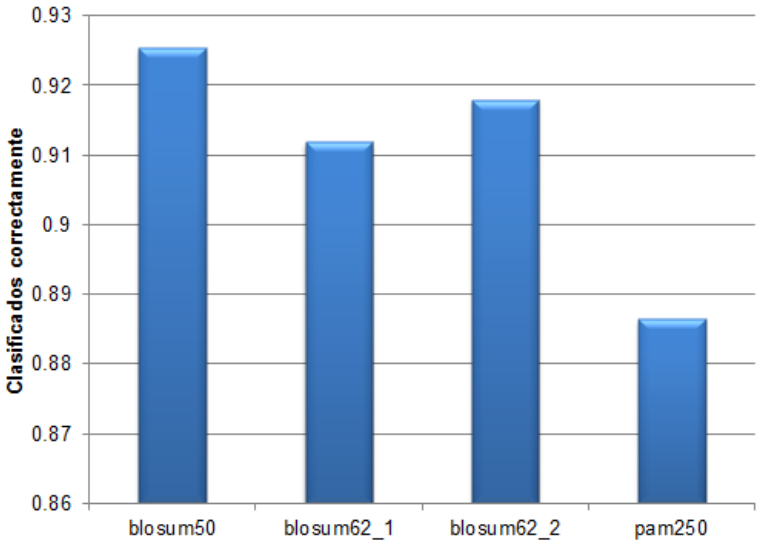


Figura 3.9.: Porciento de clasificación correcta de los pares de genes ortólogos en el experimento 5

3.2.6. Experimento 6

El experimento 6 se basa en la combinación de la medida de similitud basada en el alineamiento de secuencias con la medida de similitud basada en el perfil físico-químico de las proteínas y la medida de similitud basada en la pertenencia a los bloques *LCB*. Se aplica una máscara de poda por homología al 40 % y una poda por umbral de un 80 % a la medida de similitud del alineamiento de las secuencias. La tabla 3.12 muestra los resultados obtenidos luego de aplicar el algoritmo de agrupamiento y seleccionar los pares de genes ortólogos.

Tabla 3.12.: Tabla de contingencia del experimento 6

	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3457	2709	350	29333650	0,6932
blosum62_1	3424	2841	383	29333518	0,6799
blosum62_2	3443	2649	364	29333710	0,6956
pam250	3318	2940	489	29333419	0,6593

Como se puede observar, el modelo *blosum50* obtiene los mejores resultados, pero no superan los valores obtenidos en los experimentos anteriores. La figura 3.10 muestra el porciento de clasificación correcta de los pares de genes ortólogos para cada uno de los modelos.

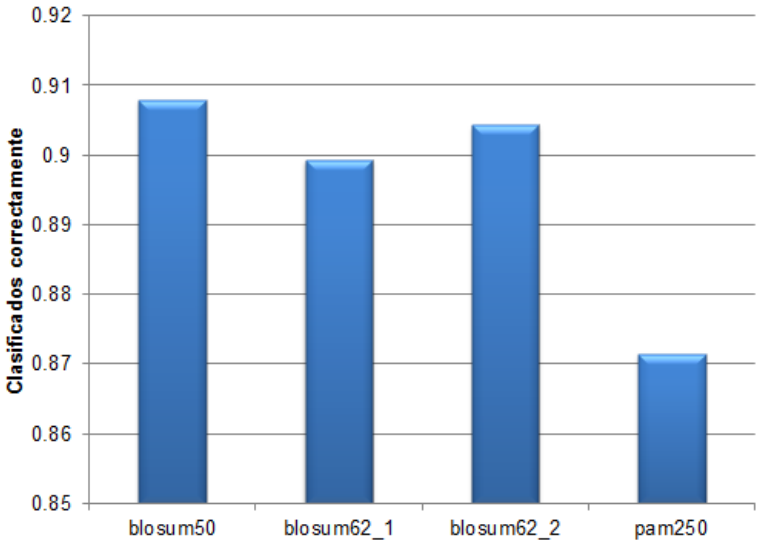


Figura 3.10.: Porciento de clasificación correcta de los pares de genes ortólogos en el experimento 6

3.2.7. Experimento 7

El experimento 7 se basa en la combinación de la medida de similitud basada en el alineamiento de secuencias con la medida de similitud basada en el perfil físico-químico de las proteínas, la medida de similitud basada en la longitud de las secuencias y la medida de similitud basada en la pertenencia a los bloques *LCB*. Se aplica una máscara de poda por homología al 40 % y una poda por umbral de un 80 % a la medida de similitud del alineamiento de las secuencias. La tabla 3.13 muestra los resultados obtenidos luego de aplicar el algoritmo de agrupamiento y seleccionar los pares de genes ortólogos.

Tabla 3.13.: Tabla de contingencia del experimento 7

	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3506	6233	301	29330126	0,5176
blosum62_1	3470	6621	337	29329738	0,4993
blosum62_2	3503	6451	304	29329908	0,5090
pam250	3397	6849	410	29329510	0,4834

Como se puede observar, el modelo *blosum50* obtiene los mejores resultados, pero no superan los valores obtenidos en los experimentos anteriores. La figura 3.11 muestra el porciento de clasificación correcta de los pares de genes ortólogos para cada uno de los modelos.

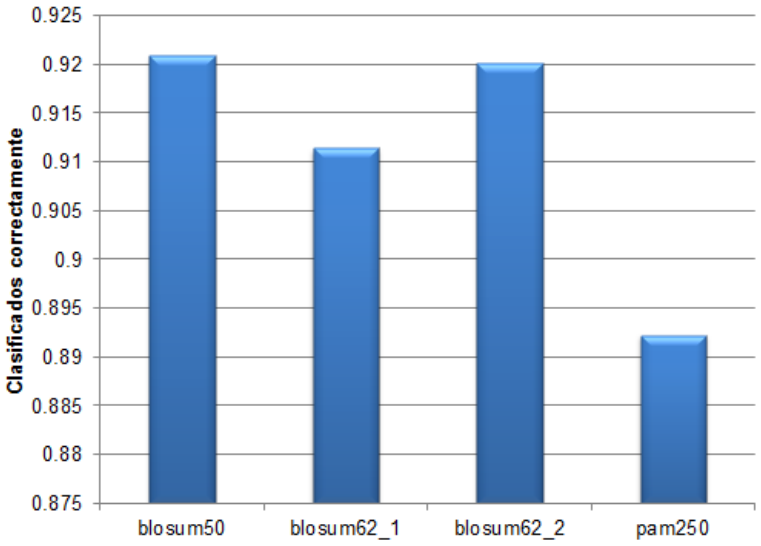


Figura 3.11.: Porciento de clasificación correcta de los pares de genes ortólogos en el experimento 7

3.2.8. Experimento 8

El experimento 8 se basa en la combinación de la medida de similitud basada en el alineamiento de secuencias con la medida de similitud basada en la longitud de las secuencias y la medida de similitud basada en la pertenencia a los bloques *LCB*. Se aplica una máscara de poda por homología al 40 % y una poda por umbral de un 80 % a la medida de similitud del alineamiento de las secuencias. La tabla 3.14 muestra los resultados obtenidos luego de aplicar el algoritmo de agrupamiento y seleccionar los pares de genes ortólogos.

Tabla 3.14.: Tabla de contingencia del experimento 8

	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3376	6619	431	29329740	0,4891
blosum62_1	3354	7011	453	29329348	0,4732
blosum62_2	3393	6813	414	29329546	0,4842
pam250	3249	7110	558	29329249	0,4586

Como se puede observar, en este caso el modelo *blosum62_2* obtiene los mejores resultados, pero continúa sin superar los valores obtenidos en los experimentos anteriores. La figura 3.12 muestra el porciento de clasificación correcta de los pares de genes ortólogos para cada uno de los modelos.

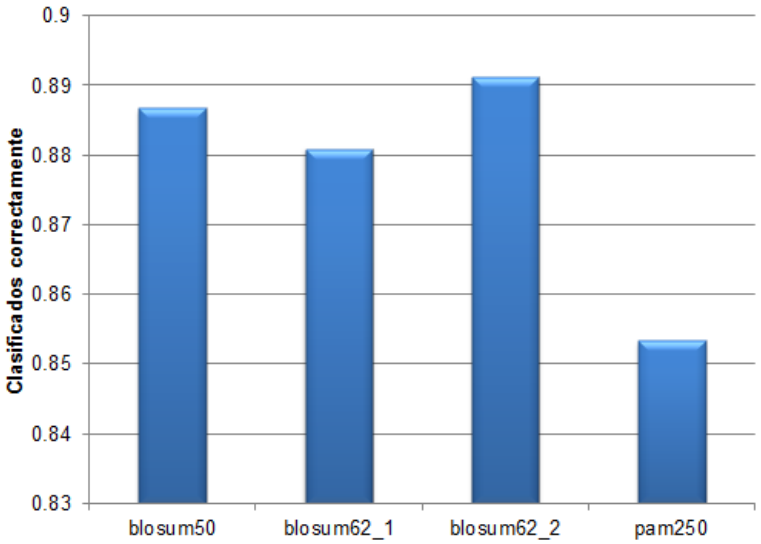


Figura 3.12.: Porciento de clasificación correcta de los pares de genes ortólogos en el experimento 8

3.2.9. Resultados generales de los experimentos

De los resultados expuestos de cada uno de los experimentos realizados se puede concluir que los mejores resultados en la clasificación se obtuvieron en el experimento 1 donde sólo se empleó la medida de similitud basada en el alineamiento. Este fue el mejor experimento en cada uno de los modelos de alineamientos.

La figura 3.13 muestra los porcientos de clasificación de los experimentos para cada uno de los modelos. Como se puede ver el experimento 1 alcanza el mayor porciento de clasificación en cada uno de los modelos. Seguido de éste se encuentra el experimento 2 donde se combina la medida de similitud del alineamiento de secuencias con la medida del perfil físico-químico. La diferencia de clasificación entre estos dos experimentos no supera los 38 pares de genes, excepto para el modelo *pam250*. Por lo que fue necesario recurrir a la medida de calidad para tener en cuenta la cantidad de falsos positivos clasificados incorrectamente.

Como se puede consultar en las tablas 3.7 y 3.8 la cantidad de falsos positivos clasificados en el experimento 2 aumenta considerablemente respecto al experimento 1. Cuando analizamos los valores de la medida de calidad para cada experimento en los modelos, la diferencia entre los experimentos se hace más notable, siendo indiscutiblemente el experimento 1 el de mejores resultados. La figura 3.14 muestra los valores de la medida de calidad para los experimentos.

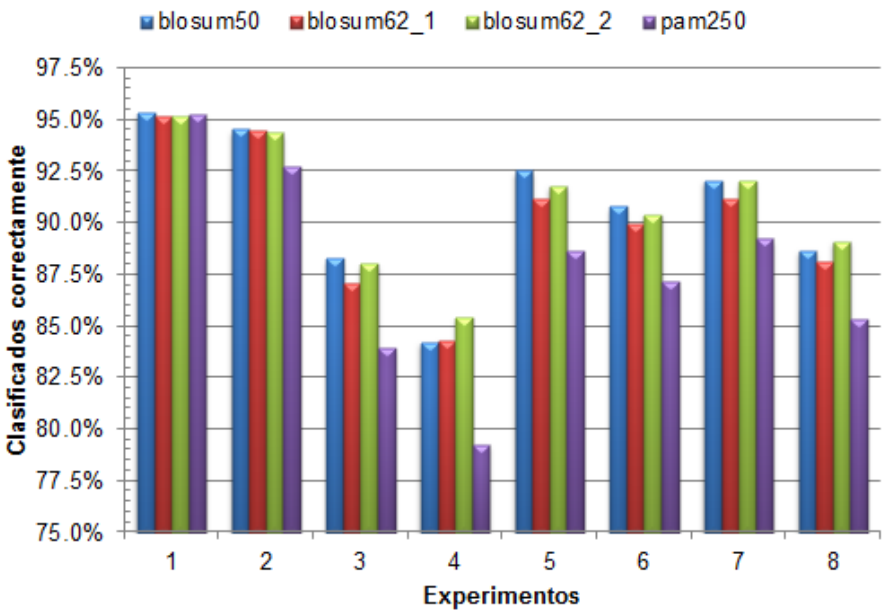


Figura 3.13.: Porciento de clasificación correcta de los experimentos para cada uno de los modelos de alineamiento

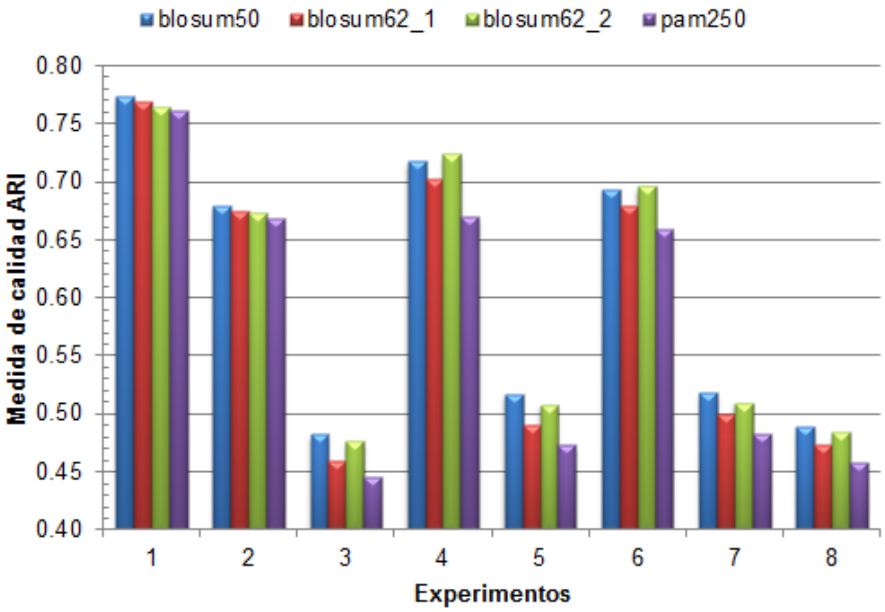


Figura 3.14.: Medida de calidad de los experimentos para cada uno de los modelos de alineamiento

3.3. Política de asignación de genes ortólogos

Para determinar si la política de asignación de genes ortólogos propuesta en este trabajo disminuye significativamente la cantidad de falsos positivos clasificados por el algoritmo, se ejecutaron los ocho experimentos sobre cada uno de los modelos. En un

primer momento se ejecutó usando la política para asignar los pares de genes ortólogos, y seguidamente se ejecutó sin usar la política de asignación. Los resultados de cada uno de los modelos se muestran en la figura 3.15.

Como se puede observar el algoritmo tiene un mejor porciento de clasificación cuando no se usa la política de asignación de genes ortólogos, pero existe un mejor balance de falsos positivos cuando se usa la política de asignación de genes ortólogos. Las diferencias de clasificación más significativas están dadas en el experimento 4 cuando se combina el rasgo basado en la longitud de las secuencias con el rasgo de la pertenencia a los *LCB* y en el experimento 6 que también se combina con el rasgo del perfil físico-químico de las proteínas.

En ambos casos, cuando se usa la política se asignación y cuando no se usa, resultan más significativos los experimento 1 y 2. Entre ambos experimentos la diferencia de pares clasificados correctamente no rebasa los 83 pares de genes, siendo mucho más notable la diferencia de falsos positivos clasificados en cada uno de los casos, con valores superiores a 1929 pares de genes.

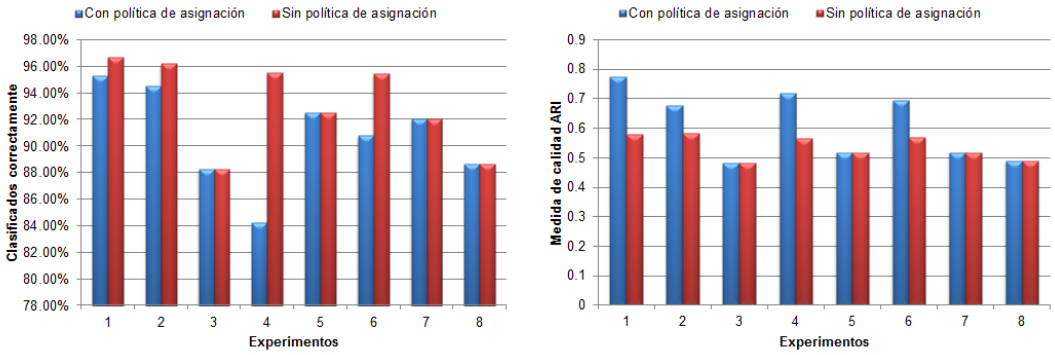
3.4. Regresión logística

Usando el *SPSS* se confeccionó un conjunto de datos para cada modelo de alineamiento, todos con la misma estructura. En cada conjunto de datos se definió la variable *"Clasificacion"* para almacenar la clasificación obtenida por el algoritmo de *INPARANOID*. Las variables *"Alineamiento"*, *"Longitud"* y *"LCB"* continene los valores de similitud de los rasgos basados en el alineamiento de secuencia, longitud de secuencias y pertenencia a los bloques LCB de cada par de secuencias. Además se definieron las variables *"Perfil_3"*, *"Perfil_5"* y *"Perfil_7"* para los valores de similitud del perfil físico-quimico para tamaño de ventana 3, 5 y 7.

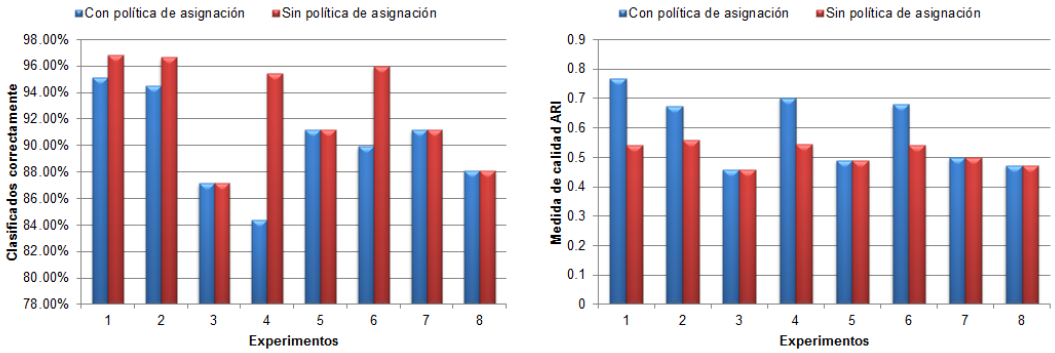
3.4.1. Modelo *blosum50*

Los datos del modelo *blosum50* están totalmente desbalanceados con 3766 pares de genes ortólogos y 8633798 pares de genes no ortólogos. El archivo de datos fue balanceado seleccionando todos los pares de genes ortólogos y aproximadamente la misma cantidad de pares de genes no ortólogos. Finalmente, el conjunto de datos quedó conformado por 3766 pares de genes ortólogos y 3751 pares de genes no ortólogos.

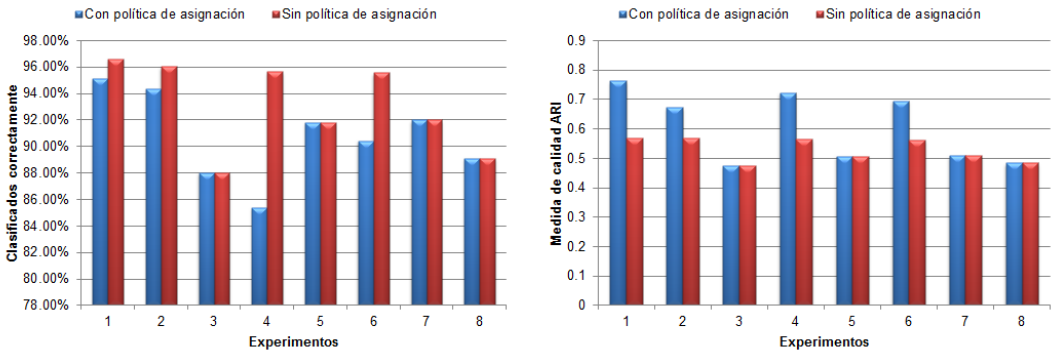
En la figura 3.16 se muestran los resultados obtenidos en el modelo de regresión logística



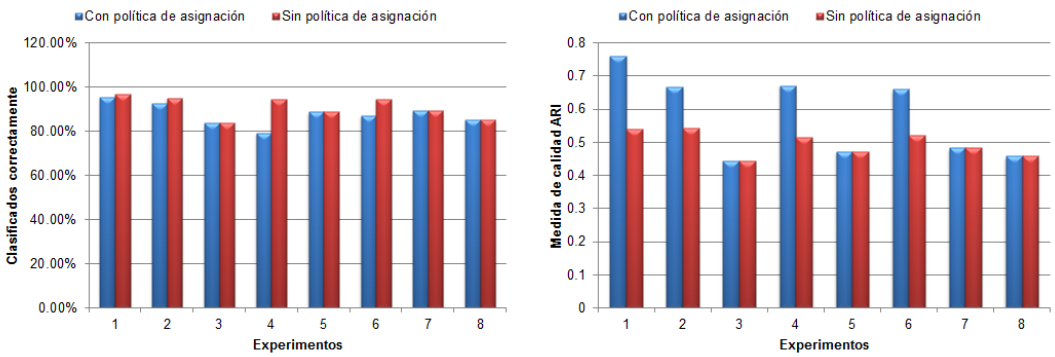
(a) Modelo *blosum50*



(b) Modelo *blosum62_1*



(c) Modelo *blosum62_2*



(d) Modelo *pam250*

Figura 3.15.: Comparación de los experimentos usando y sin usar la política de asignación de genes ortólogos

Classification Table[a]

			Predicted		
			Clasificación de los pares		
	Observed		0	1	Percentage Correct
Step 1	Clasificación de los pares	0	3554	197	94,7
		1	436	3330	88,4
		Overall Percentage			91,6
Step 2	Clasificación de los pares	0	3744	7	99,8
		1	26	3740	99,3
		Overall Percentage			99,6
Step 3	Clasificación de los pares	0	3746	5	99,9
		1	22	3744	99,4
		Overall Percentage			99,6

a. The cut value is ,500

Variables in the Equation

		B	S.E.	Wald	df	Sig.	Exp(B)
Step 1[a]	Perfil_3	20,158	,525	1476,385	1	,000	5,683E8
	Constant	-6,124	,149	1688,125	1	,000	,002
Step 2[b]	Alineamiento	2127,661	129,338	270,614	1	,000	.
	Perfil_3	6,506	2,118	9,434	1	,002	669,156
	Constant	-11,992	,680	310,724	1	,000	,000
Step 3[c]	Alineamiento	2162,078	134,006	260,311	1	,000	.
	Perfil_3	7,105	2,158	10,841	1	,001	1218,050
	Longitud	26,627	8,570	9,653	1	,002	3,663E11
	Constant	-38,198	8,558	19,921	1	,000	,000

a. Variable(s) entered on step 1: Perfil_3.

b. Variable(s) entered on step 2: Alineamiento.

c. Variable(s) entered on step 3: Longitud.

Figura 3.16.: Modelo de regresión logística obtenido del modelo de alineamiento *blosum50*

estimado. Como se puede observan el modelo clasifica correctamente el 99,6 % de los pares de la muestra. Las medidas más significativas para la clasificación en este modelo son: el alineamiento de secuencias, el perfil para tamaño de ventana 3 y la longitud de las secuencias.

Usando los coeficientes del modelo de regresión, la ecuación de la regresión logística se plantea como se muestra en la ecuación 3.1, donde X_1 representa el valor de la medida de similitud del alineamiento, X_2 representa el valor de la medida de similitud del perfil para tamaño de ventana 3 y X_1 representa el valor de la medida de similitud de la longitud de las secuencias. El punto de corte del modelo se definió como 0,5 lo que significa que si $P > 0,5$ entonces el par de genes se considera ortólogo.

$$P(Y = 1) = \frac{1}{1 + e^{(38,198 - 2162,078 \times X_1 - 7,105 \times X_2 - 26,627 \times X_3)}} \tag{3.1}$$

3.4.2. Modelo *blosum62_1*

Los datos del modelo *blosum62_1* al igual que el caso anterior, están totalmente desbalanceados con 3793 pares de genes ortólogos y 13510857 pares de genes no ortólogos. El archivo de datos fue balanceado seleccionando todos los pares de genes ortólogos y aproximadamente la misma cantidad de pares de genes no ortólogos. Finalmente, el conjunto de datos quedó conformado por 3792 pares de genes ortólogos y 3763 pares de genes no ortólogos.

En la figura 3.17 se muestran los resultados obtenidos en el modelo de regresión logística estimado. En este caso el modelo de regresión obtenido tiene cuatro pasos. Como se observa en la figura, en el cuarto paso del modelo entra la variable del perfil para tamaño de ventana 5, pero como se desea usar un sólo tamaño de ventana, y la diferencia de este paso respecto al paso 3 es solamente de un par de genes clasificados, tomaremos el modelo de regresión hasta el tercer paso. En este caso el modelo clasifica correctamente el 99,4 % de los pares de la muestra. Las medidas más significativas para la clasificación en este modelo son: el alineamiento de secuencias, el perfil para tamaño de ventana 3 y la longitud de las secuencias.

Usando los coeficientes del modelo de regresión en el tercer paso, la ecuación de la regresión logística se plantea como se muestra en la ecuación 3.2, donde X_1 representa el valor de la medida de similitud del alineamiento, X_2 representa el valor de la medida de similitud del perfil para tamaño de ventana 3 y X_1 representa el valor de la medida de similitud de la longitud de las secuencias. El punto de corte del modelo se definió como 0,5 lo que significa que si $P > 0,5$ entonces el par de genes se considera ortólogo.

$$P(Y = 1) = \frac{1}{1 + e^{(27,113 - 1421,615 \times X_1 - 8,402 \times X_2 - 17,591 \times X_3)}} \tag{3.2}$$

3.4.3. Modelo *blosum62_2*

Los datos del modelo *blosum62_2* al igual que en los casos anteriores están totalmente desbalanceados, con 3774 pares de genes ortólogos y 9439714 pares de genes no ortólogos. El archivo de datos fue balanceado seleccionando todos los pares de genes ortólogos y aproximadamente la misma cantidad de pares de genes no ortólogos. Finalmente, el conjunto de datos quedó conformado por 3774 pares de genes ortólogos y 3725 pares de genes no ortólogos.

En la figura 3.18 se muestran los resultados obtenidos en el modelo de regresión

Classification Table[a]

Observed			Predicted		
			Clasificación de los pares		
			0	1	Percentage Correct
Step 1	Clasificación de los pares	0	3562	201	94,7
		1	430	3363	88,7
		Overall Percentage			91,6
Step 2	Clasificación de los pares	0	3754	9	99,8
		1	33	3760	99,1
		Overall Percentage			99,4
Step 3	Clasificación de los pares	0	3755	8	99,8
		1	36	3757	99,1
		Overall Percentage			99,4
Step 4	Clasificación de los pares	0	3755	8	99,8
		1	35	3758	99,1
		Overall Percentage			99,4

a. The cut value is ,500

Variables in the Equation

		B	S.E.	Wald	df	Sig.	Exp(B)
Step 1[a]	Perfil_3	20,585	,539	1458,259	1	,000	8,711E8
	Constant	-6,054	,146	1708,664	1	,000	,002
Step 2[b]	Alineamiento	1418,995	73,812	369,578	1	,000	.
	Perfil_3	8,589	1,671	26,415	1	,000	5369,719
	Constant	-10,091	,499	408,191	1	,000	,000
Step 3[c]	Alineamiento	1421,615	75,049	358,821	1	,000	.
	Perfil_3	8,402	1,640	26,252	1	,000	4454,214
	Longitud	17,591	5,920	8,829	1	,003	4,362E7
	Constant	-27,113	5,813	21,752	1	,000	,000
Step 4[d]	Alineamiento	1407,760	75,289	349,616	1	,000	.
	Perfil_3	5,122	2,270	5,093	1	,024	167,686
	Perfil_5	4,088	1,800	5,158	1	,023	59,601
	Longitud	16,675	5,952	7,849	1	,005	1,746E7
	Constant	-26,420	5,838	20,480	1	,000	,000

a. Variable(s) entered on step 1: Perfil_3.

b. Variable(s) entered on step 2: Alineamiento.

c. Variable(s) entered on step 3: Longitud.

d. Variable(s) entered on step 4: Perfil_5.

Figura 3.17.: Modelo de regresión logística obtenido del modelo de alineamiento *blosum62_1*

Classification Table[a]

Observed			Predicted: Clasificación de los pares		
			0	1	Percentage Correct
Step 1	Clasificación de los pares	0	3526	199	94,7
		1	424	3350	88,8
		Overall Percentage			91,7
Step 2	Clasificación de los pares	0	3716	9	99,8
		1	32	3742	99,2
		Overall Percentage			99,5
Step 3	Clasificación de los pares	0	3717	8	99,8
		1	26	3748	99,3
		Overall Percentage			99,5

a. The cut value is ,500

Variables in the Equation

		B	S.E.	Wald	df	Sig.	Exp(B)
Step 1[a]	Perfil_3	19,997	,518	1491,130	1	,000	4,839E8
	Constant	-6,103	,148	1709,059	1	,000	,002
Step 2[b]	Alineamiento	1959,770	110,547	314,279	1	,000	.
	Perfil_3	8,197	1,797	20,816	1	,000	3629,900
	Constant	-11,347	,589	370,715	1	,000	,000
Step 3[c]	Alineamiento	1959,436	111,026	311,466	1	,000	.
	Perfil_3	8,943	1,771	25,511	1	,000	7657,091
	Longitud	27,117	8,296	10,683	1	,001	5,979E11
	Constant	-37,983	8,250	21,197	1	,000	,000

a. Variable(s) entered on step 1: Perfil_3.

b. Variable(s) entered on step 2: Alineamiento.

c. Variable(s) entered on step 3: Longitud.

Figura 3.18.: Modelo de regresión logística obtenido del modelo de alineamiento *blosum62_2*

logística estimado. Al igual que en los casos anteriores se obtiene un alto porciento de clasificación y las medidas más significativas para la clasificación continúan siendo: el alineamiento de secuencias, el perfil para tamaño de ventana 3 y la longitud de las secuencias.

Usando los coeficientes del modelo de regresión, la ecuación de la regresión logística se plantea como se muestra en la ecuación 3.3, donde X_1 representa el valor de la medida de similitud del alineamiento, X_2 representa el valor de la medida de similitud del perfil para tamaño de ventana 3 y X_1 representa el valor de la medida de similitud de la longitud de las secuencias. El punto de corte del modelo se definió como 0,5 lo que significa que si $P > 0,5$ entonces el par de genes se considera ortólogo.

$$P(Y = 1) = \frac{1}{1 + e^{(37,983 - 1959,436 \times X_1 - 8,943 \times X_2 - 27,117 \times X_3)}} \tag{3.3}$$

3.4.4. Modelo *pam250*

Los datos del modelo *pam250* al igual que en los casos anteriores están totalmente desbalanceados, con 3802 pares de genes ortólogos y 16320698 pares de genes no ortólogos. El archivo de datos fue balanceado seleccionando todos los pares de genes ortólogos y aproximadamente la misma cantidad de pares de genes no ortólogos. Finalmente, el conjunto de datos quedó conformado por 3802 pares de genes ortólogos y 3828 pares de genes no ortólogos.

En la figura 3.19 se muestran los resultados obtenidos en el modelo de regresión logística estimado. Al igual que en los casos anteriores se obtiene un alto porciento de clasificación y las medidas más significativas para la clasificación continúan siendo: el alineamiento de secuencias, el perfil para tamaño de ventana 3 y la longitud de las secuencias.

Usando los coeficientes del modelo de regresión, la ecuación de la regresión logística se plantea como se muestra en la ecuación 3.4, donde X_1 representa el valor de la medida de similitud del alineamiento, X_2 representa el valor de la medida de similitud del perfil para tamaño de ventana 3 y X_1 representa el valor de la medida de similitud de la longitud de las secuencias. El punto de corte del modelo se definió como 0,5 lo que significa que si $P > 0,5$ entonces el par de genes se considera ortólogo.

$$P(Y = 1) = \frac{1}{1 + e^{(38,638 - 1246,028 \times X_1 - 7,610 \times X_2 - 28,797 \times X_3)}} \tag{3.4}$$

3.4.5. Validación de los modelos de regresión logística

Cada uno de los modelos de regresión logística obtenidos, fueron programados en Java, y para su validación se probó todo el conjunto de datos de cada uno de los modelos de alineamiento. En la tabla 3.15 se muestran los resultados obtenidos de la clasificación usando cada uno de los modelos de regresión logística.

Tabla 3.15.: Tabla de contingencia de los modelos de regresión logística

	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3744	26168	63	29310191	0,2219
blosum62_1	3757	27693	50	29308666	0,2129
blosum62_2	3748	24324	59	29312035	0,2350
pam250	3763	28938	44	29307421	0,2060

Classification Table[a]

Observed			Predicted		
			Clasificación de los pares		
			0	1	Percentage Correct
Step 1	Clasificación de los pares	0	3599	229	94,0
		1	498	3304	86,9
		Overall Percentage			90,5
Step 2	Clasificación de los pares	0	3817	11	99,7
		1	39	3763	99,0
		Overall Percentage			99,3
Step 3	Clasificación de los pares	0	3818	10	99,7
		1	39	3763	99,0
		Overall Percentage			99,4

a. The cut value is ,500

Variables in the Equation

		B	S.E.	Wald	df	Sig.	Exp(B)
Step 1[a]	Perfil_3	19,248	,492	1527,965	1	,000	2,286E8
	Constant	-5,834	,138	1776,041	1	,000	,003
Step 2[b]	Alineamiento	1262,231	67,141	353,430	1	,000	.
	Perfil_3	6,949	1,568	19,650	1	,000	1041,816
	Constant	-10,721	,503	454,814	1	,000	,000
Step 3[c]	Alineamiento	1246,028	67,960	336,163	1	,000	.
	Perfil_3	7,610	1,625	21,924	1	,000	2018,000
	Longitud	28,797	6,124	22,115	1	,000	3,210E12
	Constant	-38,638	6,066	40,574	1	,000	,000

a. Variable(s) entered on step 1: Perfil_3.

b. Variable(s) entered on step 2: Alineamiento.

c. Variable(s) entered on step 3: Longitud.

Figura 3.19.: Modelo de regresión logística obtenido del modelo de alineamiento *pam250*

En la figura 3.20 se muestra un resumen de forma gráfica de la clasificación obtenida usando cada uno de los modelos de regresión logística. Como se puede observar se logran obtener altos porcientos de clasificación de pares de genes ortólogos respecto a la lista de ortólogos de *INPARANOID*. Pero aún existen un alto número de falsos positivos, los cuales se reflejan en la medida de calidad *ARI*, la cual tiene valores inferiores a 0,25. Los valores de clasificación son superiores a los valores obtenidos por el algoritmo propuesto, pero se clasifican muchos más falsos positivos con relación al algoritmo propuesto en este trabajo.

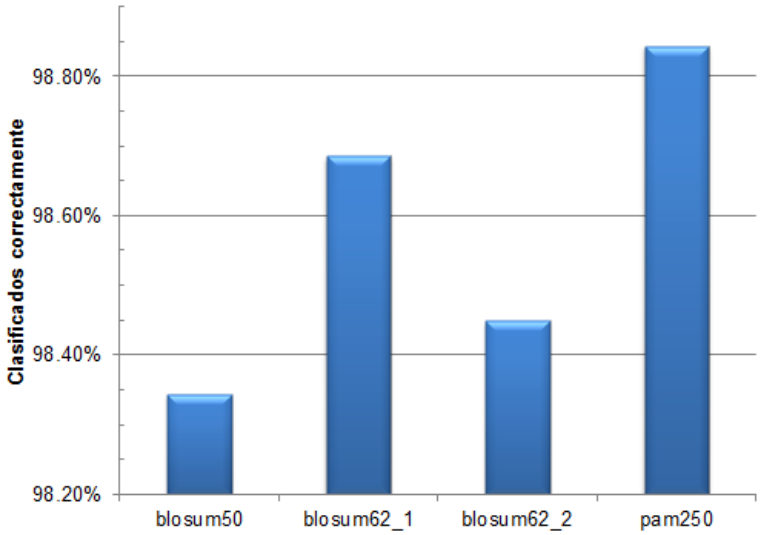


Figura 3.20.: Porciento de clasificación de pares de genes ortólogos de los modelos de regresión logística

Se puede concluir que los modelos de regresión logística producen buenos resultados de clasificación, pero clasifican un alto número de falsos positivos, que se hace necesario disminuir.

Consideraciones finales del capítulo

En este capítulo se realizaron experimentos sobre los resultados del alineamiento local y global de secuencias, donde se obtuvo que la puntuación del alineamiento local y la combinación de ésta con la puntuación global son las dos medidas obtenidas del alineamiento que tienen una mayor influencia en la clasificación de genes ortólogos para los algoritmos de agrupamiento *RBH*, *BUS* y *MCL*. Cuando se combinan las puntuaciones locales y globales se obtiene un mejor balance de falsos positivos para estos tres algoritmos de agrupamiento.

Para la selección del tamaño de ventana para el perfil físico-químico de las proteínas se realizaron experimentos para tamaño de ventana 3, 5 y 7. Los mejores resultados se obtienen para tamaño de ventana 3, con un mejor porciento de clasificación y mejor balance de falsos positivos.

Se realizaron ocho experimentos para validar los resultados obtenidos del algoritmo propuesto en este trabajo, aplicando la política de asignación de genes ortólogos definida en sección 2.6, con la clasificación obtenida de *INPARANOID* para los genomas del *Saccharomyces Cerevisiae* y *Schizosaccharomyces Pombe*. Los mejores resultados se obtienen cuando se emplea la medida de similitud del alineamiento de secuencias y cuando se combina ésta con la medida del perfil físico-químico para tamaño de ventana 3. En todos los modelos de alineamiento los resultados son similares, siendo el modelo *blosum50* el de mayor porciento de clasificación y un mejor balance de falsos positivos. Se logra obtener un 95,35 % de clasificación correcta de pares de ortólogos usando la medida del alineamiento de secuencias y un 94,56 % combinando el alineamiento de secuencias con la medida del perfil físico químico para tamaño de ventana 3. Estos resultados superan el 85,4 % obtenido en (Fernández, 2012).

Se realizaron los mismos ocho experimentos sin aplicar la política de asignación de genes ortólogos definida en este trabajo para determinar si se disminuye o no la cantidad de falsos positivos clasificados. En estos experimentos los más significativos son los mismos experimentos obtenidos aplicando la política de asignación. Usando la política de asignación se logra disminuir significativamente la cantidad de falsos positivos clasificados, a pesar de que disminuyen los pares de ortólogos clasificados correctamente, en el peor de los casos en 83 pares.

Finalmente, se estimaron modelos de regresión logística para analizar la influencia de cada una de las medidas de similitud en la clasificación de genes ortólogos de referencia. Las medidas de similitud más influyentes en la clasificación fueron la medida de similitud basada en el alineamiento, la medida de similitud basada en el perfil físico-químico para tamaño de ventana 3 y la medida de similitud basada en la longitud de la secuencia. En la validación de los modelos se obtuvieron porcientos de clasificación correcta de genes ortólogos superiores al 98,2 %, pero con una mayor cantidad de falsos positivos en relación a los obtenidos con el algoritmo propuesto.

CONCLUSIONES

Los algoritmos de detección de genes ortólogos basados en grafo parten del alineamiento de secuencias y utilizan una definición operacional de ortología para podar el grafo de similitudes y luego aplicar un algoritmo de agrupamiento. Se evidencia en la bibliografía la tendencia a combinar el alineamiento con información de bloques de orden conservado y reordenamientos globales.

En este trabajo se combinó la información de cuatro rasgos de similitud: el alineamiento de secuencias, la longitud de las secuencias, la pertenencia a los bloques *LCB* y la información del perfil físico-químico de las proteínas.

Mediante experimentos sobre los resultados del alineamiento local y global de secuencias se obtuvo que la puntuación del alineamiento local y la combinación de ésta con la puntuación global son las dos medidas obtenidas del alineamiento que tienen una mayor influencia en la clasificación de genes ortólogos para los algoritmos de agrupamiento `RBH`, *BUS* y *MCL*.

Las políticas de poda sobre el grafo de similitudes sugeridas en la literatura unidas al algoritmo de agrupamiento MCL y la política de asignación definida en el trabajo conforman el algoritmo propuesto en este trabajo con niveles adecuados de clasificación.

Los experimentos realizados con la medida de similitud basada en el perfil físico-químico de las proteínas para diferentes tamaños de ventana indican que el tamaño de ventana 3 debe producir mejores resultados de clasificación. Esto fue corroborado con la construcción de modelos de regresión logística.

Los resultados obtenidos de la clasificación fueron validados con la clasificación de *INPARANOID* para los genomas del *Saccharomyces Cerevisiae* y *Schizosaccharomyce Pombe*. Los mejores resultados se obtienen cuando se emplea la medida de similitud del alineamiento de secuencias y cuando se combina ésta con la medida del perfil físico-químico para tamaño de ventana 3. Se logra obtener un 95,35 % de clasificación correcta de pares de ortólogos usando la medida del alineamiento de secuencias y un 94,56 % combinando el alineamiento de secuencias con la medida del perfil físico químico para tamaño de ventana 3. Estos resultados superan el 85,4 % obtenido en (Fernández, 2012).

El uso de la medida de similitud basada en el perfil físico-químico, en el algoritmo propuesto, mejora los resultados de la clasificación con relación a trabajos anteriores, sin embargo, no supera la clasificación obtenida al combinar las puntuaciones del

alineamiento local con las puntuaciones del alineamiento global.

Usando la política de asignación se logra disminuir significativamente la cantidad de falsos positivos clasificados a pesar de que disminuyen los pares de ortólogos clasificados correctamente.

Las medidas de similitud más influyentes en la clasificación, según modelos de regresión logística, fueron: la basada en el alineamiento, la basada en el perfil físico-químico para tamaño de ventana 3 y la basada en la longitud de la secuencia. En la validación de los modelos de regresión se obtuvieron porcientos de clasificación correcta de genes ortólogos superiores al 98,2 %, pero con una mayor cantidad de falsos positivos en relación a los obtenidos con el algoritmo propuesto.

La implementación paralela en MATLAB del cálculo de los alineamientos de secuencias y el perfil físico-químico, mejora la complejidad computacional de la versión secuencial en ambos cálculos. La complejidad computacional del cálculo del alineamiento se reduce de un $O(N \times m \times n)$ a un $O\left(\frac{N \times m \times n}{P} + N \times (m+n)\right)$; y la complejidad computacional del cálculo del perfil se reduce de un $O(N \times (m+n)^2)$ a un $O\left(\frac{N \times (m+n)^2}{P} + N \times (m+n)\right)$.

RECOMENDACIONES

1. Realizar el cálculo de la medida de similitud del perfil físico-químico de las proteínas usando la transformada de Fourier.
2. Estudiar otros rasgos que caractericen a los genes con vistas a mejorar la precisión de los algoritmos.
3. Realizar la validación con otras clasificaciones incluyendo las listas curadas manualmente.
4. Continuar estudios sobre modelos de regresión logística aplicándolos a la clasificación de genes ortólogos.

REFERENCIAS BIBLIOGRÁFICAS

Achelis, Steven B. (1995): *Technical Analysis from A to Z.* McGraw-Hill.

Achen, C. H. (1982): *Interpreting and using regression.* London: Sage.

Alexeyenko, Andrey; I. Tamas; Gang Liu; and Erik L. Sonnhammer (2006): Automatic clustering of orthologs and inparalogs shared by multiple proteomes. *Bioinformatics*, 22(14):e9–e15.

Altman, B. (2006): Guide to Bioinformatics at Stanford University.

Altschul, S. F. and G. Gish (1996): Local alignment statistics. *Methods Enzymology*, 266:460–480.

Altschul, S. F.; W. Gish; W. Miller; W. Myers; and D. J. Lipman (1990): Basic local alignment search tool. *Jorunal Molecular Biology*, 215:403–410.

Bader, G. and C. Hogue (2003): An automated method for finding molecular complexes in large protein interaction networks. *BMC BioInformatics.*

Ben-Hur, A.; A. Elisseeff; and I. Guyon (2002): A stability based method for discovering structure in clustered data. In: *Pacific Symposium on Biocomputing.* pp. 6–17.

Bezdek, J. C. and N. Pal (1998): Some new indexes of cluster validity. *IEEE Transactions Syst. Man Cybernet*, 28:301–315.

Bondy, John Adrian and U. S. R. Murty (1976): *Graph Theory with Applications.* North-Holland.

Breit, Kevin; Henry House; Judith Samson; Alan Horkan; Thomas Harding; and Mark Dexter (2009): *Dia manual.*

Brohue, S. and J. Van Helden (2006): Evaluation of clustering algorithms for protein-protein interaction networks. *BMC BioInformatics.*

Brun, M.; C. Sima; J. Hua; J. Lowey; B. Carroll; E. Suh; and E. R. Dougherty (2007): Model-based evaluation of clustering validation measures. *Pattern Recognition*, 40:807–824.

Busygin, Stanislav; Gerrit Jacobsen; and Ewald Kramer (2002): Double conjugated clustering applied o leukemia microarray data. In: *In Proceedings of the 2nd*

SIAM International Conference on Data Mining, Workshop on Clustering High Dimensional Data.

Carpio-Muñoz, Carlos Adriel Del and Julio Cesar Carbajal (2002): Folding Pattern Recognition in Proteins Using Spectral Analysis Methods. *Genome Informatics*, 13:163–172.

Carpio-Muñoz, Carlos Adriel Del and A. Yoshimori (2002): *Protein structure prediction: Bioinformatic approach*, International University Line Publishers (IUL), chap. Fully automated protein tertiary structure prediction using Fourier transform spectral methods, pp. 171–200.

Chen, Feng; Aaron J. Mackey; Christian J. Stoeckert; and David S. Roos (2006): OrthoMCL-DB: querying a comprehensive multi-species collection of ortholog groups. *Nucleic Acids Research*, 34(Database issue):D363–D368.

Chen, Xin; Jie Zheng; Zheng Fu; Peng Nan; Yang Zhong; Stefano Lonardi; and Tao Jiang (2005): Assignment of Orthologous Genes via Genome Rearrangement. *IEEE-ACM transactions on computational biology and bioinformatics*, 2(4):302–315.

Cock, Peter J. A.; Tiago Antao; Jeffrey T. Chang; Brad A. Chapman; Cymon J. Cox; Andrew Dalke; Iddo Friedberg; Thomas Hamelryck; Frank Kauff; Bartek Wilczynski; and Michiel J. L. de Hoon (2009): Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423.

Core-Team (2001): What is R? *R News*, 1(1):2–3.

Darling, Aaron C.E.; Paul Infield-Harm; and Anna Rissman (2009): *Mauve User Guide.*

Darling, Aaron C.E.; Bob Mau; and Frederick R. Blattner (2004): Mauve: Multiple Alignment of Conserved Genomic Sequence With Rearrangements. *Genome Research*, 14(7):1394–1403.

Darling, Aaron C.E.; Istvan Miklos; and Mark A. Ragan (2008): Dynamics of genome rearrangement in bacterial populations. *PLoS Genetics. In press.*

Darling, Aaron E.; Bob Mau; and Nicole T. Perna (2010): progressiveMauve: Multiple Genome Alignment with Gene Gain, Loss and Rearrangement. *PLoS ONE*, 5(6).

Davies, D.L. and D.W. Bouldin (1979): A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Learning*, 1(2).

Dayhoff, M.O. (1978): *Survey of new data and computer methods of analysis*, vol. 5 of *Atlas of protein sequence and structure*. National Biomedical Research Foundation, Georgetown University, Washington, D.C.

Deza, Elena and M. Deza (2006): *Dictionary of Distances*. Elsevier.

Diestel, Reinhard (2000): *Graph Teory*. Springer.

Duch, Włodzisław (2000): Similarity-based methods: a general framework for classification, approximation and association. *Control and Cybernetics*, 29(4):1–30.

Durbin, Richard I.; S. R. Eddy; A. Krogh; and G. J. Mitchison (1998): *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, Cambridge, United Kingdom.

Edgar, Robert C. (2004): MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797.

Enright, A. J.; S. Van Dongen; and C. A. Ouzounis (2002): An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30:1575–1584.

Estopiñales, Michel (2009): *Herramientas Computacionales para la Comparación de Genomas*. Trabajo de diploma.

Fawcett, T. (2004): *ROC Graphs: Notes and Practical Considerations for Researchers*. Tech. rep., Palo Alto, USA: HP Laboratories.

Fernández, Miguel A. (2012): *Herramientas computacionales para la comparación de genomas y detección de genes ortólogos con un enfoque de grafo bipartido*. Tesis de maestría, Universidad Central "Marta Abreu"de Las Villas.

Fourment, Mathieu and Michael R. Gillings (2008): A comparison of common programming languages used in bioinformatics. *BMC BioInformatics*, 9(82):1471–2105.

Fu, Zheng; Xin Chen; Vladimir Vacic; Peng Nan; Yang Zhong; and Tao Jiang (2007): MSOAR: A High-Throughput Ortholog Assignment System Based on Genome Rearrangement. *Journal of Computational Biology*, 14:16.

Gabaldón, Toni; Christophe Dessimoz; Julie Huxley-Jones; Albert J. Vilella; Erik LL. Sonnhammer; and Suzanna Lewis (2009): Joining forces in the quest for orthologs. *Genome Biology*, 10(403).

George, D. G.; W. C. Barker; and L. T. Hunt (1990): Mutation data matrix and its uses. *Methods Enzymology*, 183:333–351.

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

Gibbs, A.J. and G. A. McIntyre (1970): The diagram, a method for comparing sequences. Its use with amino acid and nucleotide sequences. *Journal Biochemical*, 16:1–11.

Goffeau, A. (1996): Life with 6000 genes. *Science*, 274(546):563–567.

Goodstadt, L. and C. P. Ponting (2006): Phylogenetic Reconstruction of Orthology, Paralogy, and Conserved Synteny for Dog and Human. *PLoS Computational Biology*, 2(9).

Guha, S. (1999): ROCK: A RobustClustering Algorithm for Categorical Attributes. In: *l Conf. Data Eng.* ., IEEE CS Press, Los Alamitos,California, pp. 512–521.

Halkidi, M.; Y. Batistakis; and M. Vazirgiannis (2001): On clustering validation techniques. *Intelligent Informatic System*, 17:107–145.

Henikoff, S. and J. G. Henikoff (1992): Automated assembly of protein blocks for database searching. In: *National Academic Science*. vol. 89, p. 10915–10919.

Holland, R. C. G.; T. A. Down Down; M. Pocock; A. Prlić; D. Huen; K. James; S. Foisy; A. Dräger; A. Yates; M. Heuer; and M. J. Schreiber (2008): BioJava: an open-source framework for bioinformatics. *Bioinformatics*, 24(18):2096–2097.

Hubert, L. and P Arabie (1985): Comparing partitions. *Journal of Classification*, pp. 193–218.

Hulsen, Tim; Martijn A. Huynen; Jacob de Vlieg; and Peter MA. Groenen (2006): Benchmarking ortholog identification methods using functional genomics data. *Genome Biology*, 7(4).

Jardine, N. and R. Sibson (1971): *Mathematical Taxonomy*. John Wiley & Sons.

Johnson, Toby (2008): Reciprocal best hits are not a logically sufficient condition for orthology.

Jones, D. T.; W. R. Taylor; and J. M. Thornton (1992): The rapid generation of mutation data matrices from protein sequences. *Computer Application in Biosciences*, 8:275–282.

Kamvysselis, Manolis (Kellis) (2003): *Computational comparative genomics genes, regulation, evolution*. Doctor of philosophy in computer science.

Kanehisa, M. and P. Bork (2003): Bioinformatics in the post-sequence era. *Nature Genetics*, 33.

Karypis, George; Eui-Hong Han; and Vipin Kumar (1999): Chamelon: Hierarchical clustering using dynamic modeling. *IEEE Data Mining*, pp. 68–75.

Karypis, George and Vipin Kumar (1998a): *MeTiS: A software package for partitioning unstructured graphs, partitioning meshes, and computing reducing orderings of sparse matrices.* Tech. rep., University of Minnesota, Department of Computer Science and Engineering.

Karypis, George and Vipin Kumar (1998b): Multilevel algorithms for multi-constraint graph partitioning.

Karypis, George; Kirk Schloegel; and Vipin Kumar (2003): *ParMETIS Parallel Graph Partitioning and Sparse Matrix Ordering Library.* Tech. rep., University of Minnesota, Department of Computer Science and Engineering.

Keith, Jonathan M.; Faisal Ababneh; Timothy L. Bailey; Robert G. Beiko; Martin Bishop; Guillaume Bourque; Paul M.E. Bunje; Julian M. Catchen; John S. Conery; Aaron E. Darling; Steffen Durinck; Richard Friedberg; Midori A. Harris; Yoshihide Hayashizaki; Mirko Hennig; Jaap Heringa; Andrea Ilari; Vivek Jayaswal; Lars Sommer Jermiin; Hideya Kawaji; Namshin Kim; Christopher Lee; Pietro Liò; Tim Massingham; Alice Carolyn Mchardy; Ilene Karsch Mizrachi; Arfst Nickelsen; José M. Peregrín-Alvarez; Walter Pirovano; John H. Postlethwait; Mark A. Ragan; John Robinson; Carmelinda Savino; Lincoln G. Scott; Till Tantau; Glenn Tesler; Georg F. Weiller; Simon Whelan; Thierry Wirth; Yi Xing; and Sophia Yancopoulos (2008): *Bioinformatics Data, Sequence Analysis and Evolution*, vol. I of *Methods in Molecular Biology.* Humana Press.

King, A.; N. Przulj; and I. Jurisica (2004): Protein complex predictionvia cost-based clustering. *Bioinformatics.*

Kleinberg, J.M. and S. Lawrence (2001): The structure of the Web. *Science*, 294(5548):1849–1850.

Koski, L. B. and G. B. Golding (2001): The closest BLAST hit is often not the nearest neighbor. *Journal Mol. Evol.*, 52:540–542.

Kumar, V.; G. Karypis; A. Grama; and A. Gupta (2003): *Introduction to Parallel Computing.* Addison Wesley, England, second edition edn.

Labrada, Maray Montes de Oca (2011): *Agrupamiento en grafos bipartitos y optimización de parámetros basada en enjambre de partículas, aplicación en la detección de genes ortólogos.* Trabajo de diploma.

Larkin, M. A.; G. Blackshields; N. P. Brown; R. Chenna; P. A. McGettigan; H. McWilliam; F. Valentin; I. M. Wallace; A. Wilm; R. Lopez; J. D. Thompson; T. J. Gibson; and D. G. Higgins (2007): Clustal W and Clustal X version 2.0. *Bioinformatics*, 23:2947–2948.

Larsen, B. and Ch. Aone (1999): Fast and Effective Text Mining Using Linear-time Document Clustering. In: *KDD-99 Workshop*. San Diego, CA, USA.

Lee, Yuandan; Razvan Sultana; Geo Pertea; and Jennifer Cho (2002): Cross-referencing eukaryotic genomes: TIGR Orthologous Gene Alignments (TOGA). *Genome Research*, 12(3):493–502.

Levenshtein, V. I. (1965): Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163:845–848.

Li, Li; Christian J. Stoeckert; and David S. Roos (2003): OrthoMCL: Identification of Ortholog Groups for Eukaryotic Genomes. *Genome Research*, 13:2178–2189.

Lindahl, J. M. M. (2008): *Nuevas extensiones a los operadores OWA y su aplicación a los métodos de decisión.* Tesis de doctorado, Universidad de Barcelona.

Lipman, David J. and William R. Pearson (1985): Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441.

Madeira, Sara C. and Arlindo L. Oliveira (2004): Biclustering Algorithms for Biological Data Analysis: A Survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45.

Maizel, J. V. and R. P. Lenk (1981): Enhanced graphic matrix analysis of nucleic acid and protein sequences. In: *National Academic Science*. vol. 78, p. 7665–7669.

MathWorks (2009a): *MATLAB(R2009a) - Parallel Computing Toolbox.* The MathWorks, Inc.

MathWorks (2009b): *MATLAB(R2009a) - Product Overview.* The MathWorks, Inc.

MathWorks (2009c): *MATLAB(R2009a) Bioinformatics Toolbox - Product Overview.* The MathWorks, Inc.

Miller, W. and E. W. Myers (1988): Sequence comparison with concave weighting functions. *Mathematical Biology*, 50:97–120.

Mirkin, B. (1996): *Mathematical Classification and Clustering*. Kluwer Academic Publishers.

Mount, D. W. (2004a): *Bioinformatics Sequence and Genome Analysis*, Cold Spring Harbor Laboratory Press, chap. 7, pp. 281–335. 2nd edn.

Mount, David W. (2004b): *Bioinformatics Sequence and Genome Analysis*, Cold Spring Harbor Laboratory Press, chap. 3, pp. 51–137. 2nd edn.

Needleman, Saul B. and Christian D. Wunsch (1970): A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Jorunal Molecular Biology*, 48(3).

O'Brien, Kevin P.; Maido Remm; and Erik L. Sonnhammer. (2005): Inparanoid: a comprehensive database of eukaryotic orthologs. *Nucleic Acids Research*, 33(Database issue):D476–D480.

Orfanidis, S. J. (1996): *Optimum Signal Processing. An Introduction*. Prentice-Hall, Englewood Cliffs.

Östlund, Gabriel; Thomas Schmitt; Kristoffer Forslund; and Tina Köstler (2010): InParanoid 7: new algorithms and tools for eukaryotic orthology analysis. *Nucleic Acids Research*, 38(Database issue):D196–D203.

Overbeek, Ross; Michael Fonstein; Mark D'Souza; Gordon D. Pusch; and Natalia Maltsev (1999): The use of gene clusters to infer functional coupling. *Proceedings of the National Academy of Sciences of the United States of America*, 96:2896–2901.

Rand, W. M. (1971): Objective criteria for the evaluation of clustering methods. *American Statistical Association*, 66(336):846–850.

Rasmussen, M. and M. Kellis (2005): Multi-BUS: An algorithm for resolving multi-species gene correspondence and gene family relationships.

Remm, Maido; Christian E. V. Storm; and Erik L. L. Sonnhammer (2001): Automatic Clustering of Orthologs and In-paralogs from Pairwise Species Comparisons. *Jorunal Molecular Biology*, 314:1041–1052.

Schildt, Hebert (2001): *Java 2 Manual de Referencia*. McGraw-Hill.

Schmitt, Thomas; David N. Messina; Fabian Schreiber; and Erik L.L. Sonnhammer (2011): Letter to the Editor: SeqXML and OrthoXML: standards for sequence and orthology information. *Briefings in Bioinformatics*, 12(5):485–488.

Smith, T. F. and M. S. Waterman (1981): Identification of common molecular subsequences. *Jorunal Molecular Biology*, 147:195–197.

Stajich, J.; D. Block; K. Boulez; S. Brenner; S. Chervitz; C. Dagdigian; G. Fuellen; J. Gilbert; I. Korf; H. Lapp; H. Lehväslaiho; C. Matsalla; C. Mungall; B. Osborne; M. Pocock; P. Schattner; M. Senger; L. Stein; E. Stupka; M. Wilkinson; and E. Birney (2002): The Bioperl toolkit: Perl modules for the life sciences. *Genome Research*, 12(10):1611–1618.

Stein, Benno and Oliver Niggemann (July 1999): *Workshop on Graph Theory*, Springer, Ascona, Italy, chap. On the Nature of Structure and its Identification.

Sánchez, Robersy (2006): A Novel DNA Sequence Vector Space over an extended Genetic Code Galois Field MATCH. *Commun. Math. Comput. Chem.*, 56:5–20.

Sánchez, Robersy and Ricardo Grau (2009): An algebraic hypothesis about the primeval genetic code architecture. *Mathematical Biosciences*, 221:60–76.

Tanay, Amos; Roded Sharan; and Ron Shamir (2002): Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18(1):S136–S144.

Tang, Chun; Li Zhang; Idon Zhang; and Murali Ramanathan (2001): Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. In: *In Proceedings of the 2nd IEEE International Symposium on Bioinformatics and Bioengineering*. pp. 41–48.

Tatusov, Román L. (2003): The COG database: an updated version includes eukaryotes. *BMC Bioinformatics*, 4(41).

Tatusov, Román L.; Eugene V. Koonin; and David J. Lipman (1997): A genomic perspective on protein families. *Science*, 278(5338).

Thompson, J.D.; F. Plewniak; J. C. Thierry; and O. Poch (2000): DbClustal: Rapid and reliable global multiple alignments of protein sequences detected by database searches. *Nucleic Acids Research*, 28:2919–2926.

Towfic, Fadi; M. Heather West Greenlee; and Vasant Honavar (2009): Detection of Gene Orthology Based On Protein-Protein Interaction Networks. pp. 48–53.

van Dongen, Stijn (2011): *mcxio - File Formats*. Ubuntu Manual Pages.

van Dongen, Stijn Marinus (2000): *Graph Clustering by Flow Simulation*. Ph.D. thesis, Faculteit Letteren, Universiteit Utrecht.

van Rijsbergen, Cornelius Joost (1979): *Information Retrieval*. Butterworths, 2nd edn.

Vinh, Nguyen Xuan; Julien Epps; and James Bailey (2009): Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary? In: *Proceedings of the 26 th International Conference on Machine Learning*.

Wall, D. P.; H. B. Fraser; and A. E. Hirsh (2003): Detecting putative orthologs. *Bioinformatics*, 19(13):1710–1711.

Webber, C. A. P. and P. Chris (2004): Genes and Homology. *Current Biology*, 14(R332).

Werner, E. (2005): The Future and Limits of Systems Biology. *Science Signaling*, 278.

Wood, V. (2002): The genome sequence of Schizosaccharomyces pombe. *Nature*, 415:871–880.

Wood, V. (2006): Schizosaccharomycespombe comparative genomics; from sequence to systems. *Comparative Genomics using fungi as models*, 15:233– 285.

Yang, Jiong; Wei Wang; Haixun Wang; and Philip Yu (2002): δ-clusters: Capturing subspace correlation in a large data set. In: *In Proceedings of the 18th IEEE International Conference on Data Engineering*. pp. 517–528.

ZOEM (2011): *MCL*. Ubuntu Manual Pages.

Anexos

ANEXO A

Parámetros recomendados para el alineamiento local de secuencias, tomados de (Altschul and Gish, 1996)

Matriz de sustitución	Penalización de gaps (abertura)	Penalización de gaps (extendido)	K	λ	H
BLOSUM50	∞[1]	0 a ∞[a]	0,232	0,11	0,34
BLOSUM50	15	8 a 15	0,09	0,222	0,31
BLOSUM50	11	8 a 11	0,05	0,197	0,21
BLOSUM50	11	1	-[2]	-[b]	-[b]
BLOSUM62	∞[a]	0 a ∞[a]	0,318	0,13	0,40
BLOSUM62	12	3 a 12	0,1	0,305	0,38
BLOSUM62	8	7 a 8	0,06	0,270	0,25
BLOSUM62	7	1	-[b]	-[b]	-[b]
PAM250	∞[a]	0 a ∞[a]	0,229	0,09	0,23
PAM250	15	5 a 15	0,06	0,215	0,20
PAM250	10	8 a 10	0,031	0,175	0,11
PAM250	11	1	-[b]	-[b]	-[b]

ANEXO B

Lista de las clases implementadas en Java y paquete al cuál pertenece, con una breve descripción de la clase.

Clase	Paquete	Descripción
GenesPair	core	Representación de los pares de genes
LCBBounds	core	Representación de los límites de un bloque LCB
Sequence	core	Representación de una secuencias
SyntenyBlock	core	Representación de un bloque de orden conservado
FASTAReader	io	Clase para leer archivos en formato FASTA
FASTAWriter	io	Clase para escribir archivos en formato FASTA
FileIOManager	io	Clase para manipular y verificar los archivos de entrada/salida
GenesPairReader	io	Clase para leer archivos de pares de genes
GenesPairWriter	io	Clase para escribir archivos de pares de genes
LCBReader	io	Clase para leer archivos de los límites de los bloques LCB
LCBWriter	io	Clase para escribir archivos de los límites de los bloques LCB

continúa . . .

... continuación

Clase	Paquete	Descripción
MatrixReader	io	Clase para leer archivos de matrices de datos
MatrixWriter	io	Clase para escribir archivos de matrices de datos
Reader	io	Interfaz para la lectura de archivos
SyntenyBlockReader	io	Clase para leer archivos de bloques de orden conservado
SyntenyBlockWriter	io	Clase para escribir archivos de bloques de orden conservado
Writer	io	Interfaz para la escritura de archivos
OrthoXML2List	data	Aplicación para transformar archivos OrthoXML a una lista de pares de genes ortólogos
XML2FASTA	data	Aplicación para transformar archivos XML de INPARANOID a archivos en formato FASTA
AggregateMeasures	features	Aplicación para agregar medidas usando la media aritmética
CalculateLCBMeasure	features	Aplicación para calcular la medida de similitud basada en la pertenencia a los bloques LCB
CalculateLengthMeasure	features	Aplicación para calcular la medida de similitud basada en la longitud de las secuencias

continúa ...

... continuación

Clase	Paquete	Descripción
CalculateNormalisedMeasure	features	Aplicación para normalizar medidas de similitud
List2Matrix	graph	Aplicación para transformar una lista de similitudes a una matriz de similitud
Matrix2MaskNonHomologs	graph	Aplicación para crear una máscara de poda por homología
Matrix2MCL	graph	Aplicación para transformar una matriz de similitud al formato de datos del algoritmo MCL
Matrix2UndirectedMask	graph	Aplicación para crear una máscara de poda por umbral
Matrix2UndirectedPrunedMatrix	graph	Aplicación para combinar una máscara de poda con una matriz de similitud
MatrixMask2List	graph	Aplicación para obtener los pares no podados de una máscara de poda
MatrixResolveAmbiguities	graph	Aplicación para eliminar las ambigüedades de una matriz de similitud usando los bloques de orden conservado
MCL2GenesPair	graph	Aplicación para transformar el archivo de salida del algoritmo MCL en un archivo de pares de genes ortólogos

continúa ...

... continuación

Clase	Paquete	Descripción
BUSSimulator	graph.cluster	Aplicación que implementa un algoritmo de detección de genes ortólogos basado en BUS
OrthologousAssignment	graph.cluster	Clase de la política de asignación de genes ortólogos
RBHSimulator	graph.cluster	Aplicación que implementa un algoritmo de detección de genes ortólogos basado en RBH
Synteny	graph.cluster	Clase para calcular los bloques de orden conservado
IndexMetrics	validation	Clase para calcular las métricas de validación usando una tabla de contingencia
Validate2OrthoList	validation	Aplicación para validar una lista de pares ortólogos obtenida por el algoritmo contra una lista de pares tomada como referencia

ANEXO C

Lista de genes desechados en el proceso de transformación de los datos tomados del sitio de INPARANOID.

Saccharomyces Cerevisiae	Schizosaccharomyce Spombe	Pares de genes ortólogos
Q0045	SPBC14C8.08c	Q0045 SPMIT.01
Q0050	SPMIT.01	Q0105 SPMIT.05
Q0055	SPMIT.02	Q0120 SPMIT.05
Q0060	SPMIT.03	Q0050 SPMIT.06
Q0065	SPMIT.04	Q0055 SPMIT.06
Q0070	SPMIT.05	Q0250 SPMIT.11
Q0075	SPMIT.06	Q0275 SPMIT.04
Q0080	SPMIT.07	Q0085 SPMIT.07
Q0085	SPMIT.08	Q0130 SPMIT.10
Q0105	SPMIT.09	Q0060 SPMIT.03
Q0110	SPMIT.10	Q0065 SPMIT.03
Q0115	SPMIT.11	Q0080 SPMIT.09
Q0120	SPMTR.01	
Q0130	SPMTR.02	
Q0140	SPMTR.03	
Q0160	SPMTR.04	
Q0250		
Q0255		
Q0275		
R0010W		
R0020C		
R0030W		
R0040C		

ANEXO D

Tabla de contingencia de los ocho experimentos realizados sin usar la política de asignación de genes ortólogos definida en este trabajo.

Experimento 1					
	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3682	5250	125	29331109	0,5780
blosum62_1	3686	6098	121	29330261	0,5423
blosum62_2	3679	5474	128	29330885	0,5677
pam250	3677	6115	130	29330244	0,5407

Experimento 2					
	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3663	5133	144	29331226	0,5812
blosum62_1	3680	5703	127	29330656	0,5579
blosum62_2	3656	5406	151	29330953	0,5681
pam250	3613	5898	194	29330461	0,5425

Experimento 3					
	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3361	6784	446	29329575	0,4817
blosum62_1	3317	7336	490	29329023	0,4587
blosum62_2	3352	6921	455	29329438	0,4760
pam250	3195	7352	612	29329007	0,4451

Experimento 4					
	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3637	5410	170	29330949	0,5658
blosum62_1	3634	5945	173	29330414	0,5429
blosum62_2	3641	5441	166	29330918	0,5649
pam250	3600	6557	207	29329802	0,5155

Experimento 5					
	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3523	6294	284	29330065	0,5171
blosum62_1	3472	6891	335	29329468	0,4900
blosum62_2	3495	6474	312	29329885	0,5073
pam250	3375	7095	432	29329264	0,4727

Experimento 6					
	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3636	5316	171	29331043	0,5699
blosum62_1	3653	6005	154	29330354	0,5425
blosum62_2	3638	5497	169	29330862	0,5621
pam250	3597	6381	210	29329978	0,5218

Experimento 7					
	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3506	6239	301	29330120	0,5173
blosum62_1	3470	6626	337	29329733	0,4991
blosum62_2	3503	6456	304	29329903	0,5088
pam250	3397	6858	410	29329501	0,4830

Experimento 8					
	n_{11}	n_{10}	n_{01}	n_{00}	ARI
blosum50	3376	6625	431	29329734	0,4889
blosum62_1	3354	7016	453	29329343	0,4731
blosum62_2	3393	6818	414	29329541	0,4840
pam250	3249	7119	558	29329240	0,4583