Departamento de computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

IDENTIFICACIÓN DE DETALLES ESTRUCTURALES EN PROTEÍNAS MEDIANTE EL MÉTODO DE BÚSQUEDA DE SUBGRAFOS ISOMORFOS FRECUENTES

A Reum Lee (LU. 606/01)

Director: Adrián Turjanski
Director asistente: Cristian S. Rocha

Buenos Aires, Mayo 2008

Resumen

Las proteínas son macromoléculas que participan en casi todos los procesos celulares. Las cadenas de polipéptidos que forman una proteína adquieren una conformación tridimensional específica que determina su función. Uno de los problemas centrales de la biología es el análisis de la información presente en estas complejas estructuras como ser la búsqueda de patrones o detalles estructurales conservados. Una simplificación del problema es representar las proteínas como grafos.[39] En esta tesis se propone desarrollar una metodología para encontrar detalles estructurales comunes entre un conjunto proteínas. Para ello adaptamos el algoritmo AcGM[37] que permite realizar una búsqueda completa de subgrafos inducidos en una base de grafos. De esta manera se encara el problema de identificar los detalles estructurales característicos de un conjunto de proteínas determinando un conjunto de subgrafos isomorfos compartidos en un grupo de estructuras. Finalmente aplicamos la metodología para predecir la familia de una proteína analizando la presencia o ausencia de detalles estructurales específicos en ella.

Abstract

The proteins are macromolecules that take place in most of cellular processes. The polypeptide chains which define the proteins primary structures acquire a specific tridimensional conformation that determines their biological function. One of the most important problems in biology is the analysis of the information in these complex structures, like finding patterns or structural details. A possible approach which simplifies the problem is representing the protein structures as graphs. [39] This work introduces a new method to find common structural details among a set of proteins. We use AcGM (Apriori-based algorithm for mining frequent substructures from Graph Data), a novel algorithm which achieves the complete search of frequent connected induced subgraphs in a labeled graph dataset. [37] We identify characteristic structural details in a set of proteins using this algorithm to get a set of isomorphic subgraphs shared by this set of proteins represented as graphs. Finally, the set of isomorphic subgraphs obtained is used to predict the family of a protein analyzing the presence of them in it.

Agradecimientos

Mis agradecimientos especiales
a mis viejos, el amor que ustedes me brindan supera los 19448km de distancia.
(엄마 아빠 사랑해요.)
A Capi, sos el mejor hermano del mundo, y único para mí.

Gracias amigos, con ustedes aprendí lo que es una verdadera amistad.
Gracias Auri, Fer, Sole, Gus, Cris, Lu, Lili, Anita y SW.
Les agradezco por escucharme y hacerme el aguante cada vez que quise bajar los brazos… Y un agradecimiento especial a Lau que me ayudó muchísimo…

También quiero agradecer a mi director, Adrián, por su confianza en mí.
A mi co-director, Cristian y mi compañero del laboratorio, Esteban, por su gran ayuda que me brindaron para el desarrollo de esta tesis.

Y por último, gracias a Dios, sé que siempre estás ahí.

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Estructuras de proteínas [40, 41]

Las proteínas son las macromoléculas más abundantes en las células animales y constituyen alrededor del 50% de su peso seco. Dentro de las células se las encuentra en formas muy variadas ya sea como constituyente de las membranas biológicas y las diferentes organelas celulares, como catalizadores de reacciones químicas (enzimas), interactuando con los ácidos nucleicos o con neurotransmisores y hormonas (receptores) o participando de la transducción de señales entre otras. Prácticamente, no existe proceso biológico en el que no participe por lo menos una proteína, como ser la duplicación celular, la restauración celular o la apoptósis.

Las proteínas son polímeros de aminoácidos unidos por enlaces peptídicos, cada aminoácido está formado por un grupo amino y un grupo ácido carboxílico, unidos a un átomo de carbono central o carbono alfa, el que además tiene unido un átomo de hidrógeno y una cadena lateral de características variables.

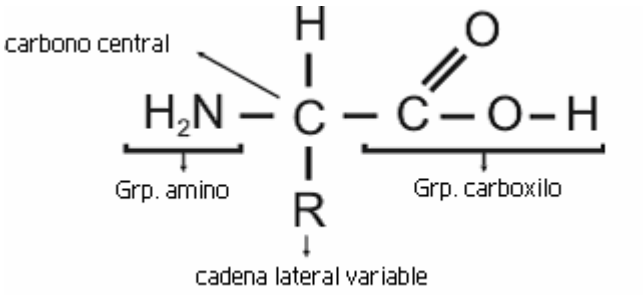


Fig. 1.1 Fórmula general de un aminoácido

Como muestra la fórmula (Fig 1.1), el carbono central se encuentra unido a un grupo variable (R). Es en dichos grupos R, donde las moléculas de los veinte aminoácidos que forman parte de las proteínas se diferencian unas de otras. En la glicina, el más simple de los aminoácidos, el grupo R se compone de un único átomo de hidrógeno. En otros aminoácidos el grupo R es más complejo, conteniendo carbono e hidrógeno, así como oxígeno, nitrógeno y azufre.

De acuerdo con la naturaleza del ”R” podemos clasificar a los aminoácidos (Fig. 1.2) en polares, básicos(con carga positiva), ácidos (con carga negativa) y aminoácidos no polares.

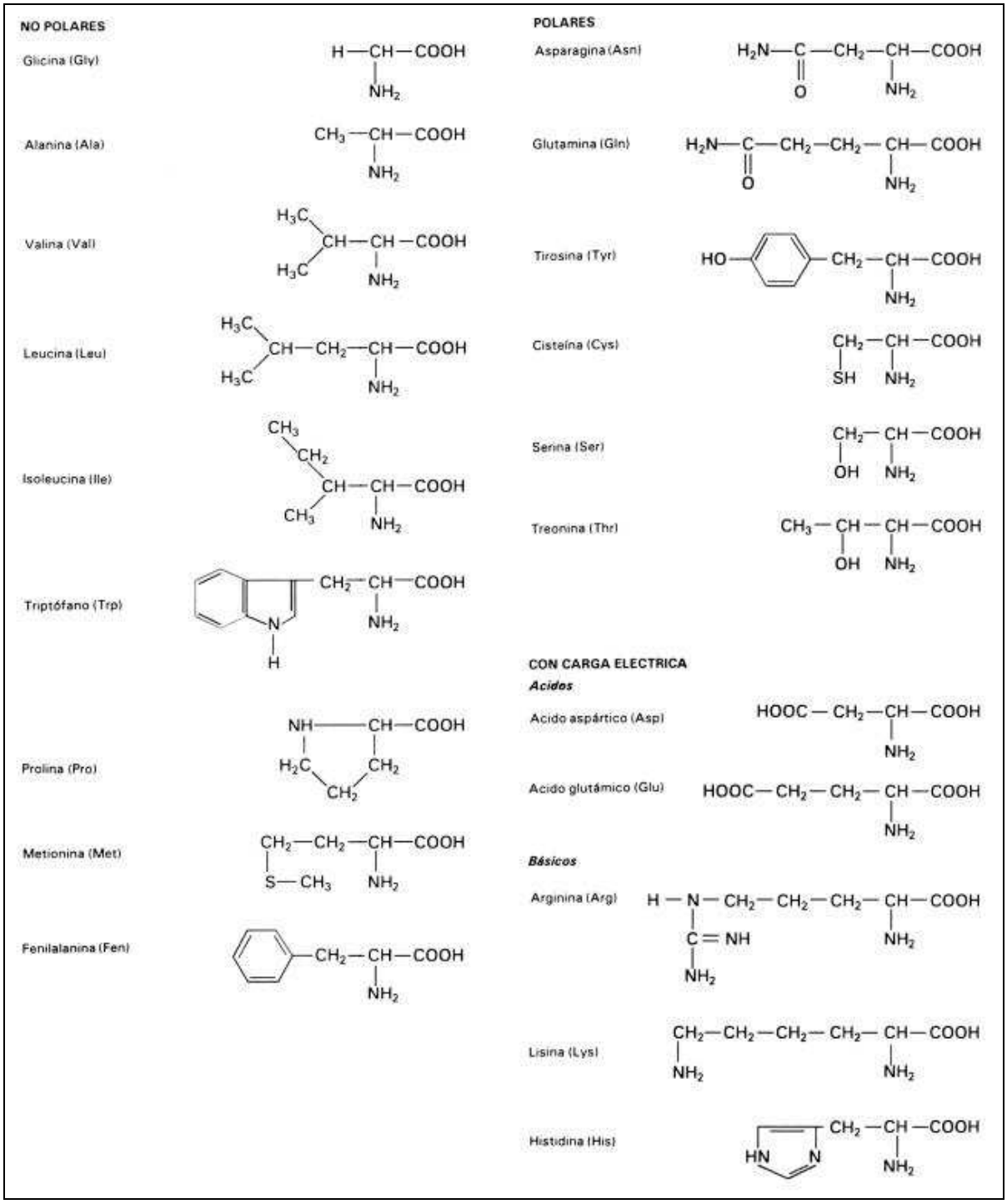


Fig. 1.2 Estructura química de los veinte aminoácidos clasificados en ácidos, básicos, neutros polares y neutros no polares. [41]

La clasificación de los aminoácidos en estos grupos está relacionada con la capacidad de interactuar con otras moléculas, en particular con el agua, el componente mayoritario en las células. Los aminoácidos con cadenas laterales no polares son hidrófobos, en tanto que aquéllos con cadenas

laterales polares son hidrófilos. Los aminoácidos ácidos tienen cadenas laterales con un grupo carboxilo y al pH celular, el grupo carboxilo se disocia de manera que el grupo R tiene una carga negativa. Los aminoácidos básicos tienen carga positiva debido a la disociación del grupo amino en su cadena lateral. Las cadenas laterales ácidas o básicas son iónicas y por tanto, son hidrófilas. Es importante remarcar que en las células se pueden producir modificaciones de los aminoácidos que en general son catalizadas por enzimas, por ejemplo la tirosina, la treonina o la serina pueden ser fosforiladas, adquirir un grupo fosfato (PO_3^-), mediante la acción de proteínas kinasas.

Los aminoácidos se combinan entre si enlazando el carbono del grupo carboxilo de una molécula con el nitrógeno del grupo amino de otra. El enlace covalente que une dos aminoácidos se denomina enlace peptídico. Cuando dos aminoácidos se combinan, se forma un dipéptido; una cadena más larga recibe el nombre polipéptido. Fig 1.3.

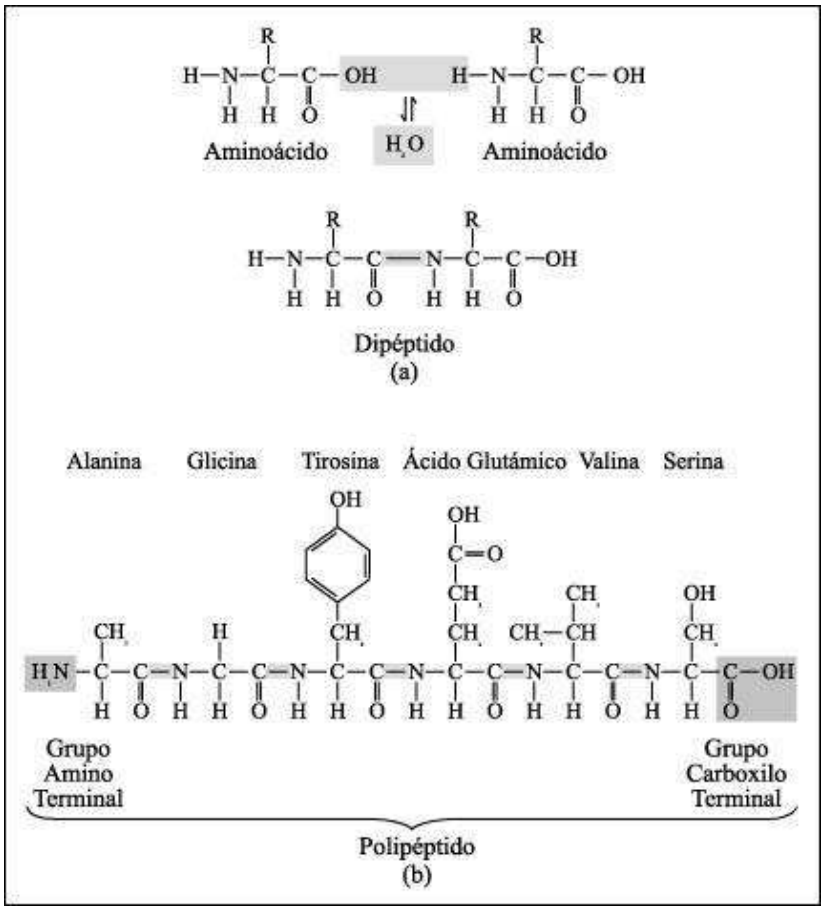


Fig. 1.3 Formación del enlace peptídico.

Un polipéptido puede contener cientos de aminoácidos unidos en un orden lineal específico y una proteína esta constituida por una o varias cadenas de polipéptidos, de esta manera pueden formarse una variedad casi infinita de moléculas proteínicas. Debe aclararse que una proteína difiere de otra en cuanto al número, tipo y secuencia de los aminoácidos que la conforman. Los veinte tipos que se encuentran en las proteínas biológicas podrían considerarse como letras de un alfabeto de proteínas, de manera que cada proteína sería una palabra formada por distintas letras.

1. Introducción

Las cadenas de polipéptidos que forman una proteína se pueden enrollar o plegar mediante interacciones específicas entre los aminoácidos de modo de adquirir una conformación tridimensional específica. Esta conformación determina la función de la proteína. Por ejemplo, la conformación única de una enzima le permite "identificar" y actuar sobre su sustrato, sustancia que dicha enzima regula. La forma de una hormona le permite combinarse con su receptor en el sitio de la célula blanco. No todas las proteínas adquieren una forma tridimensional en solución, en esta tesis trabajaremos únicamente con las proteínas que adquieren una conformación definida y cuya estructura ha sido determinada. Hay varios niveles de organización en una molécula proteínica: primario, secundario, terciario y cuaternario.

La secuencia de aminoácidos en una cadena de proteína determina su estructura primaria. Esta secuencia, esta determinada por la información genética. La estructura primaria es fundamental para la forma tridimensional que tendrá la proteína. Cualquier modificación en la secuencia de aminoácidos podría ocasionar un cambio en la estructura tridimensional y afectará la función biológica de la proteína. En la cadena polipetídica un punto importante que determina su conformación es la rotación en los enlaces covalentes del Carbono alfa de cada aminoácido. Cada unidad puede rotar en enlaces C-alfa-C=O (ángulo *psi*) y N-C-alfa (ángulo *phi*). De esta manera cada aminoácido trae asociado dos ángulos conformacionales; como estos son los puntos más importantes de rotación en la cadena polipetídica, la conformación de toda la cadena es determinada casi completamente por estos dos ángulos. Sin embargo, la mayoría de las combinaciones posibles de ángulos *phi* y *psi* no están permitidas, esto se debe a colisiones estéricas y a interacciones entre la cadena principal y la cadena lateral. La preferencia de ángulos *phi* y *psi* para un dado aminoácido se grafican generalmente en un diagrama llamado *Ramachandran plot*, un gráfico de contorno en dos dimensiones con *phi* y *psi* como ejes, y el contorno es una medida de la preferencia por los diversos ángulos. Regiones formadas en este gráfico son denominadas de acuerdo a la conformación que adopta la estructura primaria cuando los ángulos *phi* y *psi* correspondientes se repiten sucesivamente a lo largo de la cadena principal. A medida que la cadena de aminoácidos se va ensamblando, empiezan a tener lugar interacciones entre los diversos aminoácidos de la cadena. Pueden formarse por ejemplo puentes de hidrógeno entre el hidrógeno del amino de un aminoácido y el oxígeno del carboxilo de otro. A causa de estas y otras interacciones la cadena polipeptídica se pliega, adoptando mayoritariamente dos posibles configuraciones espaciales ordenadas que constituyen lo que se conoce como estructura secundaria de una proteína. Estas configuraciones son las llamadas alfa-hélice y hoja plegada beta y no son las únicas que pueden adoptar las proteínas ya que en realidad cada proteína adopta una forma característica que depende de la secuencia lineal, o estructura primaria. Sin embargo las configuraciones antes mencionadas son las más frecuentes. La hélice alfa es una estructura geométrica muy uniforme y en cada giro se encuentran 3,6 aminoácidos. La estructura helicoidal se determina y mantiene mediante puentes de hidrógeno entre los aminoácidos en los giros sucesivos de la espiral. En la estructura alfa-helicoidal, los puentes de hidrógeno ocurren entre aminoácidos que se distancian poco en secuencia, interacciones locales (Fig. 1.4). Otro tipo de estructura secundaria es la denominada lámina plegada beta. En éstas los puentes de hidrógeno pueden ocurrir entre la misma cadena y entre diferentes cadenas polipeptídicas (lámina intercatenaria); cada cadena en forma de zigzag está completamente extendida y los enlaces de hidrógeno ocasionan la formación de la estructura en forma de lámina. Pero también se pueden formar láminas plegadas entre regiones diferentes de una misma cadena peptídica (lámina intracatenaria). Fig 1.4. Son posibles dos formas laminares, según el alineamiento de las diferentes cadenas o segmentos: si éstos se alinean en la misma dirección (de extremo N- a

C-terminal, por ej.) la disposición es una lámina beta paralela, en tanto que si están alineados en sentido opuesto, la lámina es beta antiparalela. Si bien ambos casos ocurren en la naturaleza, la estructura antiparalela es más estable porque los dipolos C=O y N-H están mejor orientados para una interacción óptima.

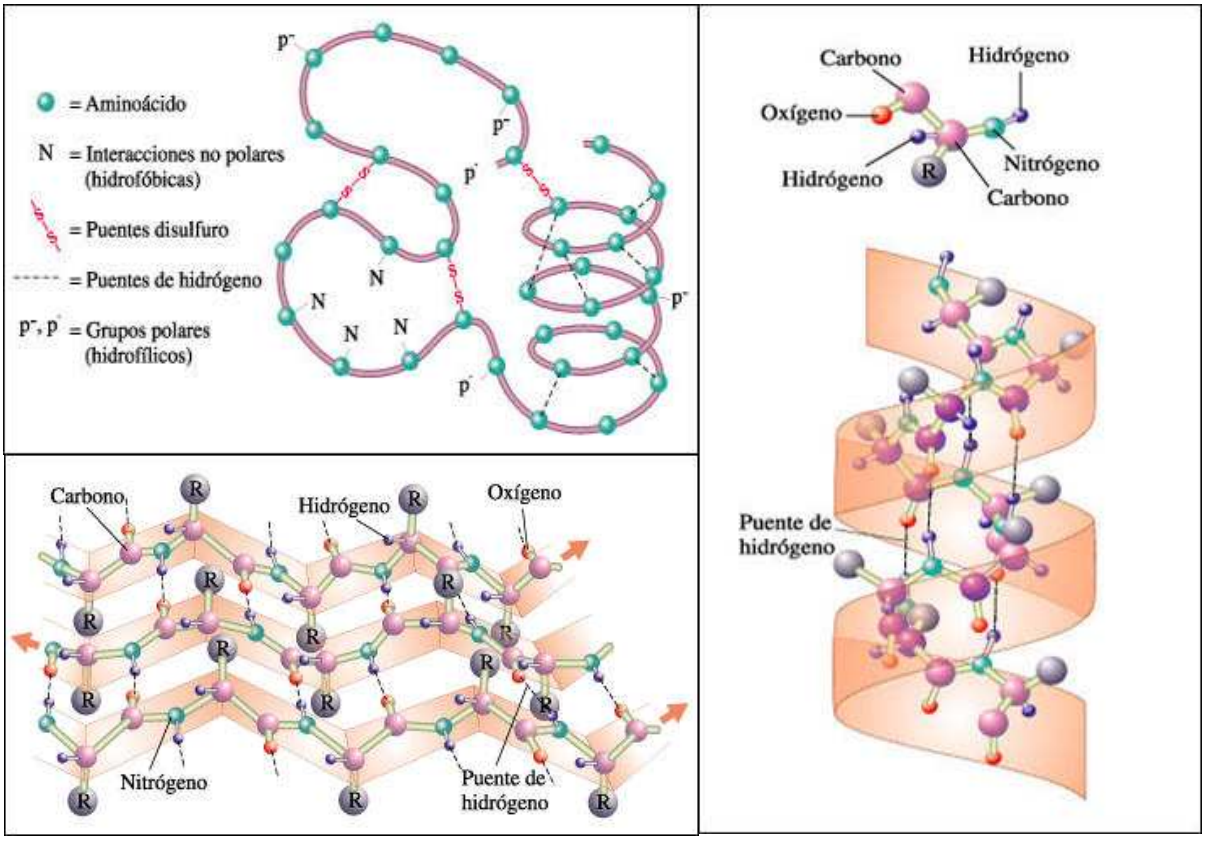


Fig. 1.4 Los distintos tipos de estructuras secundarias de una proteína.
La primera imagen (arriba izquierda) muestra interacciones típicas entre aminoácidos. La segunda imagen (abajo izquierda) muestra una lámina beta y la tercera (derecha), una hélice alfa.

Las proteínas casi siempre tienen una porción que no es ni helicoidal ni laminar, denominada aleatoria (zonas de conexión). Las proteínas se pueden clasificar por ser parcialmente helicoidales y aleatorias, parcialmente laminares y aleatorias, totalmente aleatorias o una mezcla variable de partes de ordenamiento helicoidal, laminar y aleatorio.

Un concepto muy utilizado en la estructura tridimensional de una proteína es el dominio, que corresponde a una zona de la molécula que tiene características estructurales definidas. En una molécula de proteína puede haber más de un dominio y este hecho está relacionado con la función de la misma. Los dominios suelen ser conservados a lo largo de la evolución, las proteínas con funciones parecidas suelen tener dominios similares.

La estructura terciaria de una proteína está determinada por la forma en que se interconectan las diversas estructuras secundarias de la cadena polipeptídica para generar un dominio. Así como la estructura secundaria esta principalmente determinada por interacciones entre átomos de la cadena

carbonada de los aminoácidos (no la cadena lateral), la estructura tridimensional está determinada por las diversas interacciones entre los grupos de la cadena lateral entre sí y con el agua:

1. Puentes de hidrógeno entre los grupos R de aminoácidos.

2. Atracción iónica entre los grupos R con cargas positivas y aquéllos con cargas negativas.

3. Interacciones hidrófobas derivadas de la tendencia de los grupos R no polares para asociarse hacia el centro de la estructura proteica, lejos del agua que los rodea.

4. Los enlaces disulfuro, que son covalentes (-S-S-), unen los átomos de azufre de dos subunidades de cisteína. Estos enlaces pueden unir dos porciones de una misma cadena o dos cadenas distintas.

Las proteínas compuestas de dos o más cadenas de polipéptidos adquieren una estructura cuaternaria: cada cadena muestra estructuras primaria, secundaria y terciaria, y es la interacción entre dos o más cadenas lo que determina la estructura cuaternaria. La hemoglobina, proteína de los glóbulos rojos encargada del transporte de oxígeno, es un ejemplo de proteína globular con estructura cuaternaria. La hemoglobina está compuesta por aminoácidos dispuestos en cuatro cadenas polipeptídicas: dos cadenas alfa idénticas y dos cadenas beta idénticas entre sí. (Fig. 1.5)

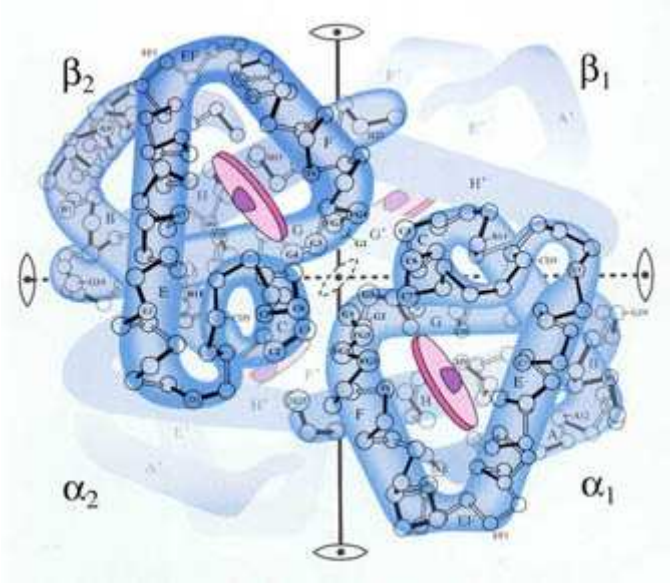


Fig. 1.5 Estructura cuaternaria de la hemoglobina

La estructura de las proteínas determina la actividad biológica de éstas. De entre las innumerables conformaciones teóricamente posibles de una proteína, generalmente hay una que predomina. Esta conformación es generalmente la más estable y en ese caso se dice que la proteína se encuentra en estado nativo (proteína nativa).

La actividad biológica de una proteína puede ser afectada por cambios en la secuencia de aminoácidos o en la conformación de la proteína. Cuando ocurre una mutación (cambio químico en un gen) que ocasiona un cambio en la secuencia de aminoácidos de una proteína, puede producirse un trastorno en las funciones celulares asociado a la alteración de su actividad biológica.

Las enzimas. Son los catalizadores de las reacciones de los sistemas biológicos. Tienen un gran poder catalítico, a menudo muy superior al de los catalizadores sintéticos. Poseen un elevado grado de especificidad respecto a sus sustratos, aceleran reacciones químicas específicas y funcionan en soluciones acuosas en condiciones determinadas de temperatura y pH. Hay pocos catalizadores no biológicos que tengan todas estas propiedades.

Las enzimas constituyen una de las claves para conocer de qué modo sobreviven y proliferan las células. Actuando en secuencias organizadas catalizan cientos de reacciones consecutivas en las rutas metabólicas mediante las que se degradan nutrientes, se conserva y transforma la energía química y se fabrican las macromoléculas biológicas a partir de precursores sencillos. Algunas de las enzimas que participan en el metabolismo son enzimas reguladores que pueden responder a diversas señales metabólicas cambiando adecuadamente su actividad catalítica. A través de la acción de las enzimas reguladoras los sistemas enzimáticos están altamente coordinados, proporcionando una armoniosa influencia recíproca entre la multitud de actividades metabólicas diferentes que son necesarias para la vida.

El estudio de las enzimas también tiene una importancia práctica inmensa. En algunas enfermedades, especialmente en las que son genéticamente heredables, puede haber una deficiencia, o incluso una ausencia total, de una o más enzimas en los tejidos. También se pueden producir situaciones anormales por la actividad excesiva de un enzima específico. La medición de la actividad enzimática en el plasma sanguíneo, eritrocitos o muestras de tejido es importante en el diagnóstico de enfermedades. Las enzimas se han convertido en herramientas prácticas importantes, no sólo en medicina sino también en la industria química, en el tratamiento de los alimentos y en la agricultura. Con la excepción de un pequeño grupo de moléculas de ARN catalítico (“ribozimas”), todas las enzimas son proteínas. Su actividad catalítica depende de la integridad de su conformación proteica nativa. Si se desnaturaliza o disocia un enzima en sus subunidades, se pierde normalmente la actividad catalítica. Si se descompone un enzima en sus aminoácidos constituyentes, siempre se destruye su actividad catalítica. Así, las estructuras primaria, secundaria, terciaria y cuaternaria de las proteínas enzimáticas son esenciales para su actividad catalítica.

Uno de los problemas centrales de la biología es la determinación y comprensión de las estructuras de las diversas proteínas. La biología estructural ha tenido un gran auge en las últimas décadas con el desarrollo de las técnicas que permiten determinar con precisión la estructura de proteínas, como ser Rayos X y Resonancia Magnética Nuclear. Hoy en día se han determinado más de 50000 estructuras proteicas. La clasificación y caracterización de las diversas estructuras ha dado origen a un área nueva de la biología computacional que se dedica al estudio de las estructura de las macromoléculas biológicas.

1.2 Bioinformática estructural [40]

La Bioinformática estructural es una rama de la bioinformática que está relacionada con el análisis y predicción de la estructura tridimensional de macromoléculas biológicas como las proteínas.

Los problemas más característicos estudiados por la bioinformática estructural son aquellos relacionados con la búsqueda de patrones en proteínas y ADN; la predicción de la estructura

tridimensional de las proteínas, y el almacenamiento y búsqueda, es decir la gestión de las estructuras de proteínas y ácidos nucleicos en bases de datos.

De estos problemas surge la necesidad de consultar una base que tenga almacenados estos datos en forma consistente. Desde la primera proteína secuenciada, la insulina bovina en 1956 con un total de 51 aminoácidos, las bases de datos han ido creciendo en tamaño y cantidad año tras año. En 1972 se crea el *Protein Data Bank* (Base de datos de estructuras de proteínas) que contenía las estructuras de las proteínas hasta entonces conocidas.

Por la alta tasa de producción de datos y la necesidad de que los investigadores tengan acceso rápido a datos nuevos, surgieron bases de datos públicas que han ido adquiriendo un rol fundamental en el desarrollo de las ciencias biomédicas modernas. Estas bases públicas, y los datos que éstas proveen, son recursos muy importantes para la bioinformática. Sin embargo, los servicios de datos públicos sufren continuamente demandas de la comunidad científica que crecen día a día en especificaciones y detalles que es imposible de abarcar por única base de datos. Es muy importante tener en cuenta desde el principio que las bases de datos nunca van a satisfacer completamente la necesidad de la comunidad. Junto con las bases de datos se han ido desarrollando una gran variedad de algoritmos que permiten clasificar y estudiar las estructuras. A continuación se exponen algunas bases públicas que merecen ser mencionadas. [1]

1.2.1 Bases de datos

Bases de datos de Secuencias de Proteínas

- UniProt: Ésta combina en una sola base la información que proveen varias bases de datos internacionales (European Bioinformatics Institute (EBI), Cambridge, UK; Protein Information Resource (PIR) - Georgetown University Medical Center (GUMC) & National Biomedical Research Foundation (NBRF), Washington, D.C.; and Swiss Institute of Bioinformatics (SIB) - Geneva, Switzerland.) The Universal Protein Resource (UniProt) provee a la comunidad científica una fuente de secuencias y funciones de proteínas única y centralizada.
- PIR. Es una base de datos que contiene datos de secuencias de proteínas que fue incorporada a un sistema de base de conocimiento íntegro y contiene diversas herramientas analíticas. La Protein Information Resource, es una de las fuentes más grandes de información sobre proteínas en EEUU.
- Swiss-Prot. Es la base de datos de secuencias de proteínas más grande de Europa, con anotaciones elaboradas por Swiss Institute of Bioinformatics. Swiss-Prot se dedica a proveer anotaciones de alta calidad (como descripción de función de una proteína, su dominio estructural, modificaciones post-traducción, variantes, etc.), con un nivel de redundancia mínimo y un alto nivel de integración con otras bases de datos.

Base de datos de clasificación y estructuras de proteínas

- PROSITE. Es una base de datos de familias de proteínas y dominios.

- MEROPS - the Protease Database. Esta base de datos provee un catálogo y clasificación de péptidos basado en estructuras. Contiene datos de proteínas que son de importancia para medicina y biotecnología.
- Protein Data Bank (PDB). Un repositorio internacional para el procesamiento y distribución de datos de estructuras de macromoleculares tridimensionales principalmente determinadas por cristalografía de rayos X y RMN.
- SCOP Esta base apunta a proveer una descripción detallada y comprensiva de relaciones estructurales y evolutivas de todas las proteínas de las que se conoce su estructura. Esta base clasifica a las proteínas en varios niveles organizados jerárquicamente. Los niveles más importantes son los siguientes:
    1. Familia: Proteínas que fueron clasificadas en la misma familia son aquellas que están claramente relacionadas. Esto significa que la identidad secuencial entre cada par de proteínas son mayor a 30% (alta homología) y cuya función sea similar.
    2. Superfamilia: Es una clasificación de familias. Las familias clasificadas en una superfamilia contienen proteínas que probablemente tienen un origen común.

1.2.4 Motivos (detalles) estructurales [43]

Como vimos en la sección anterior, actualmente hay una vasta cantidad de bases de datos e información disponible, que está aumentando cada vez más rápido, lo que implica una mayor cantidad de bases de datos (y a su vez bases más grandes) que necesitan ser examinadas para encontrar elementos y conexiones interesantes entre las diversas estructuras. Estos datos "interesantes" son muy difíciles de representar y clasificar, y de ahí que la bioinformática estructural tiene todavía mucho por desarrollar.

La predicción de la función biológica de una proteína a partir de su estructura se ha vuelto en uno de los objetivos más desafiantes de la biología estructural y estas funciones pueden ser predichas identificando motivos estructurales conservados. Identificar motivos estructurales en las estructuras proteicas complejas es una de las tareas de la bioinformática estructural. Los motivos o detalles estructurales conservados pueden tener diferentes roles, desde ser primordiales para la estabilización de la estructura tridimensional hasta formar parte del sitio catalítico responsable de la función. Definimos un Detalle Estructural (DE) como un conjunto de átomos organizados en el espacio que se encuentra conservado en un conjunto de proteínas. Los motivos estructurales en proteínas consisten en una cantidad pequeña de aminoácidos, grupos funcionales o átomos funcionales que pueden estar involucrados en la función de la proteína. Por ejemplo, la tríada catalítica constituida por serina, histidina y ácido aspartico que se encuentra en las estructuras de proteínas como la tripsina y la subtilisina, tiene una conformación particular única conservada en la familia de proteínas proteaseas.

Es interesante notar que aún cuando dos genes den lugar elementos estructurales similares, no necesariamente la secuencia aminoacídica codificada debe ser la misma. Esto puede ocurrir no sólo por la relación complicada entre las estructuras primarias y terciarias (dos proteínas de distinta estructura primaria pueden adoptar una conformación tridimensional similar), sino también porque el tamaño de elementos varía de una proteína a la otra. En otras palabras, un motivo estructural no necesita ser asociado con un motivo secuencial. Además, la existencia de un motivo secuencial no necesariamente implica una estructura distintiva.

1.3 Uso de grafos para representar proteínas y estado del arte de isomorfismo de grafos (subgraph isomorphism)

La Teoría de Grafos es una herramienta natural y conveniente para trabajar con los objetos que pueden ser vistos como un conjunto de subobjetos más elementales. Tales objetos pueden ser representados como grafos. En palabras simples, los grafos son conjuntos de nodos conectados por ejes. Este tipo de formalización es especialmente clara en casos importantes de estructuras moleculares.[1] En el caso más simple de estructuras químicas, los átomos son considerados como nodos, y las uniones químicas como ejes. Tanto los átomos como las uniones tienen propiedades que deberían ser asignadas a nodos y ejes correspondientes. Los grafos que contienen distintos tipos de nodos y/o ejes, son llamados etiquetados y pueden ser construidos sobre objetos mas sofisticados, como por ejemplo sobre elementos de estructuras secundarias de un plegamiento de una proteína (nodos con propiedades) con ejes midiendo la distancia entre nodos y sus orientaciones mutuas (cf. [2,3]). Habiendo construido una representación de grafos de un objeto en cuestión, uno puede utilizar un conjunto de herramientas para medir similitudes entre mismos tipos de objetos. Esto se logra haciendo matching de grafos, un procedimiento para identificar subgrafos comunes, o subconjuntos idénticos de nodos en grafos comparados, conectados por el subconjunto idéntico de ejes. La medida de similitud es luego dada por el tamaño del subgrafo maximal común (uno que contiene un número máximo de nodos) y generalmente por propiedades de sus nodos y ejes. Esta técnica en diversos problemas, como por ejemplo docking de proteína-ligando[4], reconocimiento de la estructura tridimensional[2,3,5–7], modelado de sitios activos [8,9], interpretación de espectro molecular [10,11], predicción de actividad biológica [12], búsqueda en base de datos [13,14] y muchos otros usos, incluyendo aquellos fuera de la química y la bioinformática (por ejemplo, reconocimiento de imágenes [15–19]).

El Matching de grafo es conocido por ser un procedimiento caro computacionalmente, lo que limita la mayoría de sus aplicaciones. Un número de algoritmos de matching tanto óptimos como aproximados fueron propuestos a lo largo de las últimas tres décadas (revisiones en [20,21]). Los Algoritmos óptimos son aquellos que garantizan encontrar la mejor solución, mientras que los algoritmos aproximados ofrecen soluciones subóptimas, usualmente con un costo computacional considerablemente bajo. La mayoría de los algoritmos óptimos para encontrar subgrafos comunes isomorfos (CSI), usados en varias aplicaciones, están basados en la detección de cliques maximales como fue primeramente propuesto en [22,23]. En cambio, los algoritmos de backtracking de CSI [24] son raramente usados por el costo computacional muy alto que está involucrado. Los problemas de detección de subgrafos exactos (ESI, un caso especial de CSI donde el subgrafo común maximal coincide con uno de los grafos de input) se resuelven mucho más efectivamente mediante algoritmos de backtracking, estos últimos han sido desarrollados por Ullman [25].

1.4 Teoría de grafos y estructuras proteicas [26-27]

Las estructuras proteicas pueden ser representadas por grafos donde los aminoácidos son los nodos y los ejes son las interacciones entre ellos. Las interacciones de los residuos de las proteínas entre y dentro de sitios funcionales son cruciales para las actividades proteicas. Sabiendo que se pueden predecir las funciones de las proteínas identificando motivos estructurales conservados, se puede aprovechar la posibilidad de representar las estructuras proteicas con grafos y aplicar métodos de detección de subgrafos isomorfos para reconocer estos motivos estructurales. En las proteínas, los

motivos estructurales usualmente consisten de solo algunos elementos, un ejemplo para un plegamiento es el motivo hélice-vuelta-hélice (*'helix-turn-helix')* que tiene solo tres elementos. Nosotros nos centraremos en motivos entre aminoácidos de una misma cadena polipeptídica.

1.4.1 Complejidad de detección de subgrafos isomorfos

El problema principal de la detección de subgrafos isomorfos es que este problema es NP-completo. Es decir, el tiempo para detectar un subgrafo isomorfo entre dos grafos, en el peor de los casos, es exponencial en el número de vértices de aquellos grafos. La técnica más usada para establecer un subgrafo isomorfo está basada en el backtracking en un árbol de búsqueda. (Efficient Subgraph Isomorphism Detection: A Decomposition Approach Bruno T. Messmer and Horst Bunke,
Member, IEEE Computer Society) **[25].**

2. Objetivo de este trabajo

El objetivo de la presente tesis es desarrollar una metodología que permita encontrar detalles estructurales comunes entre proteínas. Proponemos representar las proteínas como grafos usando la información estructural resuelta experimentalmente y así reducir el complejo problema original al de la búsqueda de subgrafos comunes conservados entre los diferentes grafos de un grupo de proteínas. Para ello inicialmente armamos una base de datos para almacenar y organizar las estructuras de proteínas y para poder calcular propiedades de manera sencilla y consistente. Luego generamos un grafo por cada estructura a estudiar usando los aminoácidos como nodos y la distancia entre ellos como enlaces. Usamos el algortimo *Apriori-based connected Graph Mining* (AcGM), que se explicará en detalle mas adelante, con el objetivo de encontrar subgrafos isomorfos comunes entre los grafos calculados previamente. Estos subgrafos comunes corresponden entonces a los detalles estructurales. Mostramos una aplicación a la búsqueda de detalles en un grupo de proteínas kinasas con el objeto de mostrar que pueden ser usados parar clasificar proteínas en familias. Finalmente, dado que el problema a evaluar tiene características particulares presentamos una evaluación de la eficiencia del algoritmo AcGM. La evaluación de eficiencia se presenta para proponer soluciones que permitan una mayor eficiencia de la metodología desarrollada, ya sea adaptar nuestro problema al algoritmo o realizar posibles modificaciones.

3. Materiales

En esta sección se mencionan las herramientas más importantes, bases de datos y programas, preexistentes utilizadas durante la tesis.

3.1 PDB

Esta herramienta que fue mencionada en la sección 1.2.1, es utilizada en esta tesis para obtener los datos estructurales de las proteínas seleccionadas para el experimento. Se obtienen además de las informaciones sobre las secuencias de las proteínas, las coordenadas de cada átomo para poder calcular propiedades estructurales de las mismas.

3.2 SCOP

Esta herramienta que fue mencionada en la sección 1.2, es utilizada en esta tesis con el objetivo de identificar y agrupar proteínas de una familia. Un conjunto de estructuras identificadas como una familia a través de esta herramienta será ingresado a la herramienta AcGM.

3.3 BLAST

BLAST (Basic Local Alignment Search Tool) encuentra regiones de similitud local entre secuencias. El programa compara secuencias de proteínas o nucleótidos con una base de secuencias y calcula la significancia estadística de las coincidencias encontradas. El BLAST puede ser utilizado tanto para inferir relaciones funcionales y evolutivas entre secuencias como para ayudar a identificar miembros de familias genéticas.
Esta herramienta es utilizada en esta tesis para comparar las secuencias de las proteínas utilizadas e identificar similitudes y diferencias. Esto nos permite agrupar y distinguir los diferentes grupos de proteínas que fueron identificadas previamente por su estructura en SCOP.

3.4 VMD

VMD es un programa de visualización molecular para mostrar, animar y analizar sistemas biomoleculares grandes usando gráficos 3D. Esta herramienta es utilizada en esta tesis para visualizar las estructuras elegidas en 3D. También se realizó un módulo para visualizar en 3D los centros geométricos de los aminoácidos que componen una estructura y enlaces (ver sección 4.3.2) y de esta manera visualizar los grafos obtenidos.

3.5 ClustalW

ClustalW es un programa de alineamiento de multiples secuencias de proteínas o ADN's. Esta herramienta calcula mejores matchs para las secuencias seleccionadas y las alinea para que puedan ser visualizadas las identidades, similitudes y diferencias entre las mismas. Esta es utilizada en esta tesis para calcular distancias entre las familias seleccionadas para probar el algoritmo AcGM.

4. Método

En esta sección se presenta el algoritmo principal utilizado para la aplicación de la propuesta. Si bien el problema de encontrar los subgrafos isomorfos frecuentes en una base de grafos es resuelto por este algoritmo, se necesitan además, otros procesos antes y después de la ejecución del algoritmo como adaptación de estructuras proteicas en grafos, creación de base de grafos, análisis de las estructuras para la generación de grafos, la selección de subgrafos característicos entre los subgrafos obtenidos por el algoritmo, evaluación de los subgrafos y visualización de los resultados. El resto de procesos y el flujo de los mismos serán presentados en la sección siguiente.

Nuestro objetivo es encontrar subgrafos isomorfos comunes en grafos de proteínas para poder identificar detalles estructurales compartidos por un conjunto de proteínas. Los métodos estudiados en el problema matemático de isomorfismo de grafo, como CSI, no son directamente aplicables a nuestro caso, porque estos métodos sirven solo para decidir si dos grafos determinados son isomorfos [33]. Inokuchi [30] introduce una representación de un grafo matemático en "matriz de adyacencia" y la manera de combinarla mediante una búsqueda "por nivel" (*levelwise search*) de codificación de matriz canónico frecuente (*frequent canonical matrix code*) [32]. La búsqueda por nivel es un concepto introducido por Agrawal usado en el algoritmo Apriori de análisis de *basket* (*basket analysis*) [31, 44]. Este algoritmo es el que ha sido utilizado en la tesis y se detalla a continuación.

4.1 Algoritmo A priori

El algoritmo Apriori presentado por Agrawal [31] fue desarrollado principalmente para descubrir reglas de asociación entre ítems en una base de datos grande de transacciones de ventas. El descubrimiento de reglas de asociación busca relaciones o afinidades entre conjuntos de artículos (*item sets*). Un **conjunto de artículos** se define como cualquier combinación formada por dos o más ítems diferentes de todos los artículos disponibles. Una **regla de asociación** se forma con dos conjuntos: la premisa y la conclusión. La conclusión se restringe a un solo elemento. Para describir la relación entre conjuntos de ítems se definen dos estadísticos: soporte (*support*) y confianza (*confidence*). La búsqueda exhaustiva usando reglas de asociación consideraría simplemente todas las combinaciones posibles de elementos, poniéndolas como premisas y conclusiones, entonces se evaluaría el soporte y confianza de cada regla. El número de combinaciones en este tipo de procedimientos crece rápidamente con el número de elementos. El algoritmo Apriori busca resolver el problema reduciendo el número de conjuntos considerados en cada nivel. El usuario define un valor de soporte mínimo (*min_sup*). Este valor sirve de umbral inferior en el momento de generar las reglas nuevas para decidir si quedarse con esta para los siguientes cálculos o no. El método comienza generando todas las reglas posibles de un solo elemento (nivel 1), luego va generando reglas de tamaño mayor aumentando de a un elemento. Este hecho nos garantiza que para el nivel n, todas las reglas de ese nivel tengan n elementos. Las reglas nuevas de nivel n son generadas uniendo dos reglas unificables de nivel n-1. Dos reglas son unificables si y solo si difieren en un solo ítem. Entonces, la regla generada a partir de dos reglas unificables (de tamaño n-1) tiene todos los elementos que se encuentran en ambas reglas del nivel inferior (en total n-2 elementos) mas dos elementos distintos, uno proveniente de una regla y otro de otra regla (y con estos dos (n-2)+2 = n elementos para la regla generada). Si una regla no tiene otro par unificable a sí mismo, ésta queda

sin pasar al nivel siguiente. Una vez que se obtienen todas las combinaciones posibles de reglas del nivel inferior Apriori cuenta la cantidad de ocurrencias de las mismas en los *item sets*, y compara la misma con el soporte mínimo. Si esta es menor al soporte mínimo la regla generada es desechada. Así en los próximos niveles estas reglas desechadas no son tenidas en cuenta para generar otras de longitud mayor. De esta manera Apriori reduce el espacio de búsqueda en cada nivel, no transfiriendo las reglas que no pueden ser unificadas con otras y descartando las reglas que no llegan a tener el soporte especificado. Esto evita la búsqueda exhaustiva de reglas de asociación.

4.2 AGM, AcGM

La minería de datos basado en estructuras de grafos es muy difícil de resolver en tiempo práctico porque la búsqueda de patrones de subgrafos posibles es combinatoriamente explosivo. La búsqueda de patrones de subgrafos posibles incluye el *matching* de subgrafos isomorfos que tiene complejidad NP-completo [27].

Para aliviar el tiempo computacional se hicieron propuestas que introdujeron conceptos de búsqueda aproximada de subgrafos. La programación de lógica inductiva (ILP) fue aplicada en este área [34] para dar una solución. Luego de algunas mejoras Dehaspe [28] propuso una manera de hacer minería de subestructuras frecuentes de componentes químicos usando el *frame work* de ILP, combinado con búsqueda por nivel para minimizar la cantidad de accesos a la base de datos. Sin embargo el espacio de búsqueda seguía siendo demasiado grande, por lo que el número de niveles debía ser limitado. Otras aproximaciones son aquellas que aplican búsqueda golosa como el algoritmo SUBDUE [29]. Sin embargo estas no son adecuadas para diferentes aplicaciones que necesitan hacer búsquedas completas y exactas. Otra propuesta fue limitar los grafos a buscar en clases más simples. El algoritmo de espacio de versiones por etiquetas propuesto por De Raedt limita el espacio de búsqueda a *frequent paths* (caminos frecuentes) para la derivación de resultados en un tiempo computacional razonable [35]. Sin embargo este algoritmo no es representativo ya que las clases de subestructuras son caminos y no subgrafos.

Para solucionar estas limitaciones se propuso un algoritmo llamado *Apriori-based Graph Mining* (AGM) en el cual la representación del conocimiento y operaciones de búsqueda de patrones se dedican a minería de grafos estructurados [36]. Este puede descubrir eficientemente todos los patrones frecuentes en términos de subgrafos inducidos que están contenidos en un *dataset* de grafos etiquetados.

En esta tesis se propone utilizar este algoritmo (AGM / AcGM) para resolver problemas de encontrar subgrafos isomorfos comunes para estudiar las estructuras de proteínas y se aplica como ejemplo a proteínas kinasas, mostrando que se puede predecir la identidad de una kinasa incógnita mediante el algoritmo.

4.2.1 Detalles del algoritmo AcGM

En esta sección se explicará en detalle el algoritmo AcGM. Previamente se presentarán las definiciones que ayudarán a dicho propósito.

4.2.1.1 Definiciones [37]

Un grafo en el que los nodos y ejes tienen etiquetas es matemáticamente definido como sigue:

Definición 1 (Grafos con etiquetas) Dado un conjunto de nodos $V(G)=\{v_1,v_2,\ldots,v_n\}$, un conjunto de ejes que conectan algún par de nodos *en* $V(G)$; $E(G)=\{e_h=(v_i;v_j)\,|\,v_i;v_j\in V(G)\}$ un conjunto de etiquetas de nodos $L(V(G))=\{lb(v_i)\,|\,\forall v_i\in V(G)\}$ y un conjunto de etiquetas de ejes $L(E(G))=\{lb(e_h)\,|\,\forall e_h\in E(G)\}$, entonces un grafo G se representa como

$$G=(V(G),E(G),L(V(G)),L(E(G)))$$

Este grafo *G* es representado por una matriz de adyacencia *X* el cual es una representación muy conocida en teoría de grafos [4]. La transformación de *G* a *X* no requiere de mucho esfuerzo computacional: $O(n^2)$.

Definición 2 (Grafos con etiquetas sin Self-loops) Esta definición es similar a la Definición 1 pero le agrega a la definición del conjunto de ejes del grafo, $E(G)$, una condición de que un nodo no puede estar unido a sí mismo por un eje. Por lo que la nueva definición de $E(G)$ es:

$$E(G)=\{e_h=(v_i,v_j)\,|\,v_i,v_j\in V(G),i\neq j\}$$

Si e_h no es dirigido, ambos (v_i,v_j) y (v_j,v_i) pertenecen al $E(G)$.

Definición 3 (Grafo Conectado) Un grafo no dirigido *G* es un grafo conectado, si existe un camino entre todo par de nodos en *G* .

Definición 4 (Grafo Semi-conectado) Un Grafo *G* es grafo semi-conectado si *G* es un grafo conectado o es un grafo que consiste en un subgrafo conectado y un nodo aislado.

Definición 5 (Matriz de Adyacencia) Dado un grafo $G=(V(G),E(G),L(V(G)),L(E(G)))$, la matriz de adyacencia X tiene siguiente elementos (i, j) ,

$$x_{ij}=\begin{cases} num(lb); & e_h=(v_i,v_j)\in E(G)\ \ y\ \ lb=lb(e_h) \\ 0; & (v_i,v_j)\notin E(G) \end{cases}$$

donde $num(lb)$ es un entero arbitrariamente asignado a un valor de etiqueta *lb*. Además, un número $num(lb)$ esta asignado a la *i* -ésima fila (*i* -ésima columna) de una matriz donde v_i en $V(G)$ y $lb=lb(v_i)$.

Definición 6 (Tamaño de grafo) El tamaño de un grafo G es la cantidad de nodos en $V(G)$, i.e., k en Definición 1.

Definición 7 (Datos de grafos) Dados n grafos con etiquetas $G_i = (V(G_i), E(G_i), L(V(G_i)), L(E(G_i)))$, con $i = 1, \cdots n$, datos de grafos GD es un conjunto de grafos, donde $GD = \{G_1, G_2, \cdots, G_n\}$.

Cada elemento de una matriz de adyacencia en una definición estándar es 0 o 1, mientras que cada elemento en la definición 5 puede tener el número de una etiqueta de eje. Esta noción extendida de una matriz de adyacencia da una representación compacta de un grafo teniendo ejes etiquetadas, y permite una codificación eficiente de un grafo como se muestra mas tarde.
La representación de una matriz de adyacencia depende de una asignación de cada nodo a una fila *i* -ésima (columna *i*-ésima). Para reducir las variantes de las representaciones e incrementar la eficiencia del *matching* del código que se describirá mas tarde, los nodos son ordenados de acuerdo a los números de las etiquetas. La matriz de adyacencia de un grafo de tamaño es *k*, se anota como X_k y el grafo como $G(X_k)$.

Definición 8 (Matriz de Adyacencia ordenado por nodos) La matriz de adyacencia X_k de un grafo $G(X_k)$ es ordenado por nodos si

$$num(lb(v_i)) \leq num(lb(v_{i+1})) \; para \; i = 1,2,\cdots,k-1$$

Definición 9 (Código de Matriz de Adyacencia) [1] En el caso de un grafo no dirigido, el código $code(X_k)$ de una matriz de adyacencia ordenado por nodos X_k ;

$$X_k = \begin{matrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,k} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,k} \\ x_{3,1} & x_{3,2} & x_{3.3} & \cdots & x_{3,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{k,1} & x_{k,2} & x_{k,3} & \cdots & x_{k,k} \end{matrix}$$

está definido como

$$code(X_k) = x_{1,1},\ x_{1,2},\ x_{2,2},\ x_{1,3},\ x_{2,3},\ x_{3,3},\ x_{1,4},\ \ldots,\ x_{k-1,k},\ x_{k,k},$$

donde los dígitos se obtienen escaneando los elementos a lo largo de columnas en la parte de triangulo superior de X_k .
Utilizando esta definición se define $CODE(X_k)$ incluyendo las etiquetas de nodos como sigue:

[1] En el análisis de *basket* estándar, items dentro de un conjunto de items se mantienen en un orden léxico-gráfico [1]. Esto permite un control eficiente de generación de conjunto de items candidatos. Pero, las matrices de adyacencia ordenadas por nodos no tienen tal orden lexicográfico. Por lo tanto, es necesario introducir un método de codificación de matrices de adyacencia.

$$CODE(X_k) = num(v_1) \cdots num(v_k)\, code(X_k)$$

Definición 10 (Forma Canónica) Un conjunto de matrices de adyacencia $\Gamma(G)$ que representa a un grafo idéntico G es definido como sigue:

$$\Gamma(G) = \{X_k \mid G(X_i) \text{ es conectado } \forall i = 1, \ldots k-1,\ G \equiv G(X_k)\}$$

donde X_i representa la submatriz $i \times i$, de izquierdo superior de X_k.

La matriz de adyacencia C_k del cual el CODE es mayor en $\Gamma(G)$ es llamado forma canónica.

$$C_k \text{ w.r.t } CODE(C_k) = \max_{X_k \in \Gamma(G)} CODE(X_k)$$

Definición 11 (Subgrafo inducido): Dado un grafo $G = (V(G), E(G), L(V(G)), L(E(G)))$, un subgrafo inducido de G, $G_s = (V(G_s), E(G_s), L(V(G_s)), L(E(G_s)))$, es un grafo que satisface las siguientes condiciones.

$$V(G_s) \subset V(G);\ E(G_s) \subset E(G);$$

$$\forall u, v \in V(G_s),\ (u,v) \subset E(G_s) \Leftrightarrow (u,v) \in E(G)$$

Cuando v es un subgrafo inducido de G, se denota como $G_s \subset G$.

Definición 12 (*Support*) Dado un grafo G_s, el *support* de G se define como

$$Sup(G_s) = \frac{numero\ de\ transacciones\ de\ grafos\ G\ donde\ G_s \subset G \in GD}{numero\ total\ de\ transacciones\ de\ grafos\ G \in GD}$$

Deficinión 13 (Frequent Induced Subgraph) Si el valor $Sup(G_s)$ es mayor que un valor de límite min_sup, G_s es llamado "subgrafo inducido frecuente" (*frequent induced subgraph*).

4.2.1.2 Algoritmo de AcGM

El algoritmo Apriori deriva todos los *item_sets* frecuentes en el orden ascendente en tamaño descartando aquellos *item_sets* que tienen el valor de *support* menor a un valor determinado en cada derivación. De la misma manera el AcGM deriva los subgrafos inducidos en el orden ascendente en cantidad de nodos en los subgrafos quedándose en cada derivación con aquellos subgrafos que tienen el valor de *support* mayor al *min_sup*.

4.2.1.2.1 Outline del algoritmo AcGM

GD es una base de datos de grafos.
F_k es un conjunto de matrices de adyacencia que representan grafos k-frecuentes.
C_k es un conjunto de matrices de adyacencia que representan grafos k-candidatos.
$\min\sup$ es el valor de *support* mínimo determinado por el usuario.

```
Main(GD, min sup){
   C_1 ← {todas las matrices de adyacencia que consisten de un nodo};
   k ← 1;
   while(C_k ≠ ∅){
      Count(GD, C_k); (Calculo de Frecuencia)
      F_k ← {c_k ∈ C_k | sup(G(c_k)) ≥ min sup};
      C_{k+1} ← Generate − Candidate(F_k);
      k ← k + 1;
    }
   return ∪_k {f_k ∈ F_k | f_k es canonico, G(f_k) es conexo};
}
```

Fig. 4.1 El outline del algoritmo AcGM

Generación de subgrafo inducido frecuente

Candidatos de subgrafos semi-conectados (inducidos) frecuentes de tamaño $\{k+1\}$ se generan con los subgrafos semi-conectados (inducidos) frecuentes de tamaño $\{k\}$ mediante la operación join.

Aunque los patrones frecuentes a ser derivados en los resultados finales de AcGM son grafos conectados, se requiere de grafos semi-conectados para realizar una búsqueda completa en los procesos intermedios de AcGM.

Condiciones de matrices unificables.

Dado dos matrices de adyacencia X_k y Y_k que representan a los subgrafos conectados (inducidos) frecuentes, estos pueden ser unificados (unificable) si y solo si cumplen con todas las condiciones siguientes:

Condición 1 X_k y Y_k son idénticos salvo por el *k*-ésima fila y *k*-ésima columna.

Condición 2 X_k es una forma canónica de $G(X_k)$.

Condición 3 $G(X_k)$ es un grafo conectado.

Condición 4 En el caso de que las etiquetas de *k*-ésimo nodos de $G(X_k)$ y $G(Y_k)$ son idénticos,

$$CODE(X_k) \geq CODE(Y_k) \quad (1)$$

En el caso de que ellos no sean idénticos,

$$num(lb(v_k \in V(G(X_k)))) > num(lb(v_k \in V(G(Y_k))))$$

o $G(Y_k)$ no es un grafo conectado. (2)

La condición 1 asegura que $G(X_k)$ y $G(Y_k)$ comparta las misma estructura salvo cada nodo correspondiente a la ultima fila y columna de las matrices de adyacencia que representan a los mismos. Como la forma canónica corresponde a una estructura de grafo en forma unívoca por la definición de *CODE*, la condición 2 evita operaciones redundantes de unión eficientemente. La condición 3 cumple un rol importante junto al uso de subgrafos conectados frecuentes para limitar la generación de grafos a grafos semi-conectados en AcGM. La condición 4 evita las uniones redundantes previniendo que si X_k es unificable con Y_k con X_k como primera matriz e Y_k segunda, no sean unificables estas mismas pero con el orden inverso, es decir, Y_k como primera y X_k como segunda. El $CODE(X_k)$ debería ser más grande que $CODE(Y_k)$ ya que $G(X_k)$ que tiene código más grande tiene posibilidad de tener más ejes con los elementos no-ceros en la matriz y por lo tanto ser una matriz conectada. Esto incrementa la posibilidad de satisfacer la condición 3.

Para que $G(Z_{k+1})$ sea un subgrafo frecuente, donde Z_{k+1} es la matriz resultante de unir X_k e Y_k, todos los subgrafos frecuentes inducidos de este deben ser también subgrafos frecuentes. Esta condición reduce la cantidad de candidatos.

Función Unión (Join / Generate-Candidate)

Como en el algoritmo de Apriori, la generación de subgrafo inducido frecuente se hace mediante búsqueda por nivel en términos del tamaño del subgrafo.

Sea X_k e Y_k matrices de adyacencia ordenadas por nodos (*vertex-sorted adjacency matrices*) de dos grafos inducidos frecuentes $G(X_k)$ y $G(Y_k)$ de tamaño *k*. Si ambos $G(X_k)$ y $G(Y_k)$ tienen los mismos elementos de matrices salvo la *k* -ésima fila y *k* -ésima columna entonces estos pueden ser unidos generando Z_{k+1} como se observa a continuación:

$$X_k = \begin{pmatrix} X_{k-1} & x_1 \\ x_2^T & x_{kk} \end{pmatrix}, \quad Y_k = \begin{pmatrix} X_{k-1} & y_1 \\ y_2^T & y_{k,k} \end{pmatrix}$$

$$Z_{k+1} = \begin{pmatrix} X_{k-1} & x_1 & y_1 \\ x_2^T & x_{k,k} & z_{k,k+1} \\ y_2^T & z_{k+1,k} & y_{k,k} \end{pmatrix} = \left(\begin{array}{cc|c} & & y_1 \\ & X_k & z_{k,k+1} \\ \hline y_2^T & z_{k+1,k} & y_{k,k} \end{array}\right),$$

$$lb(v_i; v_i \in V(G(X_k)) = lb(v_i; v_i \in V(G(Y_k)) = lb(v_i; v_i \in \mathrm{V(G(Z_{k+1}))}), \text{ con } i = 1, \cdots k-1$$

$$lb(v_k; v_k \in V(G(X_k)) = lb(v_k; v_k \in \mathrm{V(G(Z_{k+1}))}),$$

$$lb(v_k; v_k \in V(G(Y_k)) = lb(v_{k+1}; v_{k+1} \in \mathrm{V(G(Z_{k+1}))}) \quad (3)$$

donde X_{k-1} es la matriz de adyacencia que representa al grafo que tiene tamaño $k-1$, x_i e y_i $(i = 1, 2)$ son $(k-1)\times 1$ vectores de columna. X_k es llamado primera matriz (*first matrix*) e Y_k, segunda matriz (*second matrix*).

Los elementos $z_{k,k+1}$ y $z_{k+1,k}$ que surgen al unir las primera y segunda matriz, no están determinados por las matrices X_k e Y_k.

Las posibles estructuras de $G(Z_{k+1})$ son aquellas en las que existen un eje etiquetado entre el vértice *k* y el vértice *k+1*, y aquella en las que no lo hay. Entonces se generan $(|L_E|+1)$ matrices de adyacencia donde $|L_E|$ es la cantidad de etiquetas de ejes. Una matriz generada bajo estas condiciones se dice que está en forma normal.

En el análisis de *basket* estándar, el $(k+1)$ -*item_set* se transforma en un *item_set* candidato frecuente solo cuando todos los k -sub- *item_sets* se confirman que son *item_sets* frecuentes. Similarmente, el grafo G de tamaño $k+1$ es un candidato de subgrafos inducidos frecuente solo cuando todas las matrices de adyacencia generadas eliminando del grafo G el i -ésimo nodo v $(1 \le i \le k+1)$ y todos sus ejes conectados se confirman que son subgrafos inducidos frecuentes de tamaño *k*. Como este algoritmo genera solo matrices de adyacencia de forma normal en los niveles anteriores (*k* niveles), si la matriz de adyacencia del grafo generado eliminando el nodo i -ésimo v_i no esta en forma normal, este debería ser transformado a una forma normal para validar si este es igual a una matriz de forma normal encontrado anteriormente

Cálculo de Frecuencia (*Count*)

La frecuencia de cada subgrafo candidato inducido se cuenta escaneando la base de datos después de haber generado todos los subgrafos inducidos frecuentes y obtenido sus formas canónicas. Todos los grafos G en la base de datos pueden ser representados por una matriz de adyacencia X_k, pero esta no sería una matriz en forma normal en mayoría de los casos. Como los candidatos de subgrafos inducidos frecuentes están en forma normal, la normalización debe ser aplicado a X_k de cada G para verificar que los candidatos están contenidos en G. Como previamente se describió, el procedimiento de la normalización de X_k puede derivar forma normal de todos los subgrafos inducidos de G en los niveles intermedios. Por lo tanto la frecuencia de cada candidato se cuenta basándose sobre todas las formas normales de subgrafos inducidos de G. Cuando el valor de la cuenta excede el umbral inferior *min_sup*, el subgrafo es un subgrafo inducido frecuente.

La manera mas simple de hacer *matching* de subisomorfismo entre grafos de distintos tamaños (en nuestro caso el subgrafo candidato encontrado al cual se desea calcular la frecuencia en la base de datos, y cada grafo en la base de datos) puede ser realizado mapeando los nodos de un subgrafo al otro directamente. Pero el tiempo de cálculo de este método de subisomorfismo en el peor de los casos puede ser exponencial.

Una vez verificado que un subgrafo candidato $G(X_{k-1})$ es un subgrafo frecuente en la base de datos, y en el nivel siguiente se quiere calcular la frecuencia del subgrafo candidato $G(X_k)$, se podría aprovechar el cálculo de frecuencia realizado para $G(X_{k-1})$ ya que este es un subgrafo de $G(X_k)$.

Sea X_k un subgrafo candidato y X_{k-1} la primera matriz que fue utilizado para calcular X_k, se sabe que X_{k-1} ya es un subgrafo frecuente. Se graba la relación de $G(X_k)$ y $G(X_{k-1})$ de manera que a la hora de calcular la frecuencia de X_k se pueda reutilizar la búsqueda ya realizada para X_{k-1}. El reutilizar los resultados de niveles anteriores permite calcular la frecuencia más eficientemente que por el método de fuerza bruta.

Una vez que todos los subgrafos inducidos frecuentes son hallados se enumeran aquellos cuyos valores de *confidence* son más que el *confidence threshold* dado usando un algoritmo similar al de análisis de *basket* estándar.

4.2.1.2.2 Detalles de implementación

El AcGM realiza la búsqueda de subgrafos recorriendo el árbol de búsqueda en BFS (*Breadth First Search*). Cada nodo del árbol contiene una submatriz del tamaño que le corresponde a cada nivel y la raíz del árbol es una raíz *dummy*. De este nodo nacen otros que contiene submatrices de tamaño 1, luego va haciendo operación *join* para construir posibles subgrafos frecuentes de tamaño 2, y así sucesivamente. Para esto se crean dos archivos en la memoria principal. Uno contiene los grafos tal cual como están en el archivo de entrada. Este se crea, por única vez, al principio de la ejecución del algoritmo y queda fijo en la memoria principal hasta que la ejecución finaliza. Otro archivo contiene datos de subgrafos que se van creando a lo largo de la ejecución. Por lo que este archivo crece en tamaño constantemente hasta que termine de calcular las frecuencias de cada subgrafo encontrado. Como los dos archivos residen en la memoria principal a lo largo de toda la ejecución, los tamaños de estos archivos se ven limitados por el tamaño de la memoria virtual.

Como se puede observar en la explicación del método, este realiza un mecanismo de *prunning* para reducir el espacio de búsqueda. Luego de realizar *join* entre dos submatrices cuenta la frecuencia de la submatriz creada por la operación consultando al archivo que contiene los grafos. Si la frecuencia es menor al valor del *support* mínimo pedido, el algoritmo no va a utilizar este nodo del árbol para construir otras submatrices.

4.2.2.3 Ejemplo de construcción de subgrafos

Ejemplo de como se construye el subgrafo Gly-Phe1, Gly-Gap, Phe1-Phe2, Gap-Phe2., donde Phe1 y Phe2 corresponden a dos nodos diferentes con misma etiqueta.
Aquí se muestra un ejemplo claro y simple de cómo se construyen subgrafos isomorfos frecuentes.

Supongamos que la base de grafos contiene dos grafos:

size(1,4);
name(1, g2/trainsub28);
node(1,1,GLY);
node(1,2,PHE);
node(1,3,GAP);
node(1,4,PHE);
link(1,1,2,C);
link(1,1,3,C);
link(1,2,4,C);
link(1,3,4,C);
end
	size(2,5);
	name(2 g2/trainsub28);
	node(2,1,GLY);
	node(2,2,PHE);
	node(2,3,GAP);
	node(2,4,PHE);
	node(2,5,ALA);
	link(2,1,2,C);
	link(2,1,3,C);
	link(2,1,5,C);
	link(2,2,4,C);
	link(2,3,4,C);
	end

Fig. 4.2 Archivo de entrada conteniendo la información del grafo.
Con size(*x*,*y*) se indica que el grafo *x* tiene *y* cantidad de nodos, con name(*x*,*s*), que el nombre del grafo *x* es *s*, con node(*x*,*i*,*ln*), que la etiqueta del *i*-ésimo nodo del grafo *x* es *ln* y con link(*x*,*v*,*w*,*ll*), que la etiqueta del eje establecido entre los nodos *v* y *w* del grafo *x* es *ll.* Con end se indica el fin de un grafo. En este ejemplo las etiquetas de los nodos son los nombres de grupos de aminoácidos y la de ejes es *C* de contacto, la única en este ejemplo.

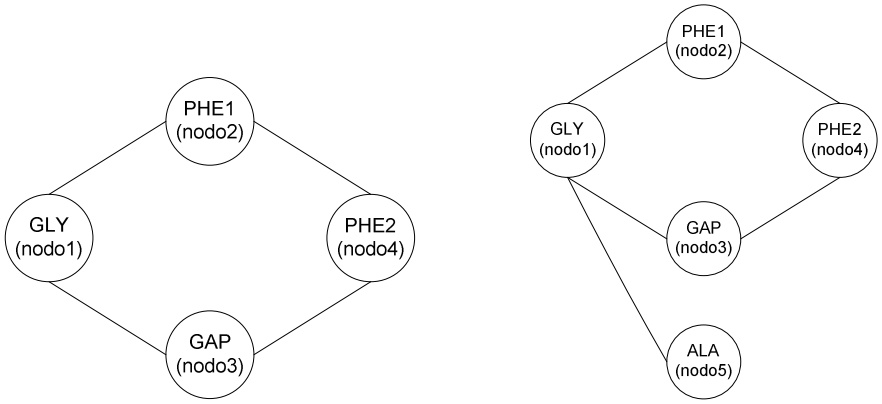


Fig. 4.3 Los dos grafos en el archivo de entrada para el algoritmo

Se puede observar que el subgrafo isomorfo maximal con *min_sup* (soporte mínimo) igual a 100% es el subgrafo siguiente:

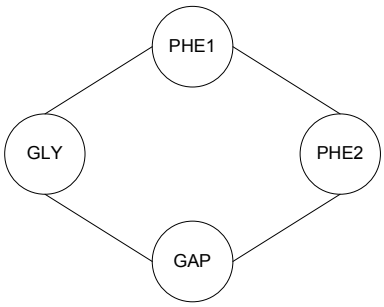


Fig. 4.4 El subgrafo isomorfo frecuente encontrado

Los números de las etiquetas se definen como sigue:

$$num(lb(GLY)) = 3$$
$$num(lb(GAP)) = 2$$
$$num(lb(PHE)) = 1$$

La etiqueta PHE es penalizado por la cantidad de apariciones en los grafos.
En el primer nivel de derivación se generan 5 subgrafos (matrices) candidatos.
$C_1^1 = GLY$, $C_1^2 = GAP$, $C_1^3 = PHE_1$, $C_1^4 = PHE_2$, $C_1^5 = ALA$ donde C_i^j indica el *j*-ésimo subgrafo candidato del nivel *i*.
Primero se elige el C_1^1 y C_1^2 para ver si es posible aplicar la función *join*. Para eso hay que verificar que se cumplan las cuatro condiciones expuestas en la sección 4.2.1.2.1 en estas dos matrices.

Las submatrices correspondientes son las siguientes:

$$C_1^1 = \begin{pmatrix} \cdot & \text{GLY} \\ \text{GLY} & - \end{pmatrix} \text{ y } C_1^2 = \begin{pmatrix} \cdot & \text{GAP} \\ \text{GAP} & - \end{pmatrix}$$

Como $num(lb(GLY)) > num(lb(GAP))$, según Eq. (2) el C_1^1 es elegido como la primera matriz (X_k con $k=0$) y el C_1^2 como la segunda matriz (Y_k con $k=0$), y se crea un nodo con la siguiente submatriz mediante la operación *join*:

$$\begin{pmatrix} . & \text{GLY} & \text{GAP} \\ \text{GLY} & - & - \\ \text{GAP} & C & - \end{pmatrix} \quad con \quad \text{CODE}(C_2^1) = 3\,2\,\text{C}$$

Así se generan todas las submatrices posibles según las condiciones que deben ser cumplidas. Una vez creada estas submatrices se accede a la base de grafos para contar las frecuencias.

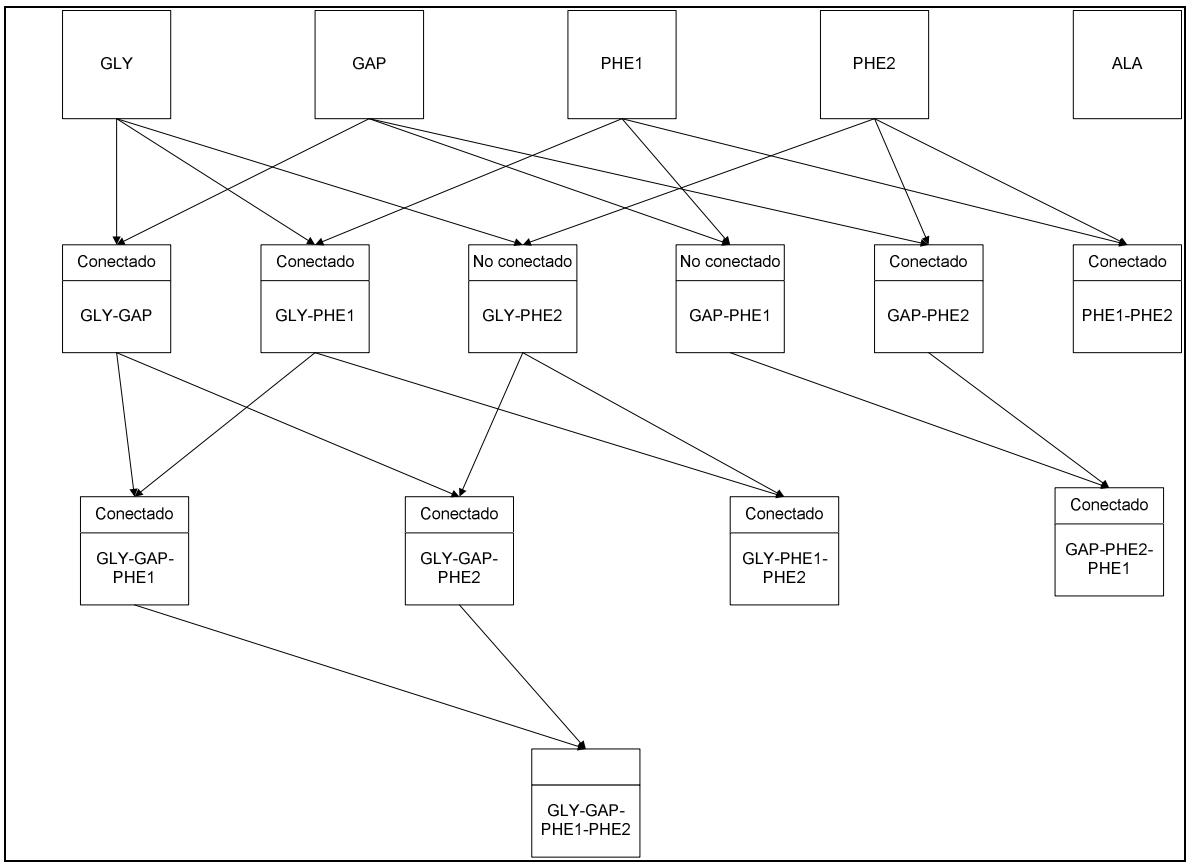


Fig. 4.5 Los subgrafos isomorfos por nivel

En la figura 4.5 se observa que dos de las submatrices construidas de tamaño 2 ($GLY - PHE_2$ *y* $GAP - PHE_1$) no son subgrafos conectados. Sin embargo, si la submatriz $GLY - PHE_2$ no estuviera en el árbol, un subgrafo posible como $GAP - PHE_2 - PHE_1$ no podría ser obtenido.

Podemos observar en la Fig. 4.5 que el resultado final es la siguiente matriz:

.	GLY	GAP	PHE_1	PHE_2
GLY	–	–	–	–
GAP	C	–	–	–
PHE_1	C	0	–	–
PHE_2	0	C	C	–

Se puede observar también que el nodo5 ALA queda podado ya que desde el principio su valor de *support* no supera el 100% (solo aparece en el grafo 2). De esta manera, el algoritmo reduce el espacio de búsqueda.

size=1 VertexLabels = GLY, code = count = 2 size=2 VertexLabels = GLY,GAP, code = C, count = 2 size=3 VertexLabels = GLY,GAP,PHE, code = C,C,0, count = 2 size=4 VertexLabels = GLY,GAP,PHE,PHE, code = C,C,0,0,C,C, count = 2 size=3 VertexLabels = GLY,GAP,PHE, code = C,0,C, count = 2 size=2 VertexLabels = GLY,PHE, code = C, count = 2	size=3 VertexLabels = GLY,PHE,PHE, code = C,0,C, count = 2 size=1 VertexLabels = GAP, code = count = 2 size=2 VertexLabels = GAP,PHE, code = C, count = 2 size=3 VertexLabels = GAP,PHE,PHE, code = C,0,C, count = 2 size=1 VertexLabels = PHE, code = count = 2 size=2 VertexLabels = PHE,PHE, code = C, count = 2

Fig. 4.6 El archivo de salida del ejemplo.

Se puede observar, como se muestra en la figura 4.6, que para encontrar un subgrafo maximal de 4 nodos (una matriz de tamaño 4x4) en la memoria principal quedan almacenados hasta el final 4 matrices de 1x1, 6 de 2x2 y 4 de 3x3. Este almacenamiento para la aplicación en cuestión es redundante ya que solo precisamos subgrafos maximales frecuentes. Si los subgrafos isomorfos maximales entre grafos son de tamaño grande termina generando y guardando una vasta cantidad de subgrafos de estos mismos en la memoria principal. Y pasado cierto tamaño este puede ocupar toda la memoria virtual, causando que el sistema operativo termine matando el proceso.

Para ver cómo puede obtenerse un grafo semiconectado a partir de dos grafos normales podemos suponer que los grafos de la base como PHE_1 y PHE_2 no están conectados. El algoritmo generaría la siguiente submatriz (que representa un subgrafo semi-conectado) para no descartar la posibilidad de que haya un nodo que esté conectado tanto a PHE_1 como a PHE_2:

$$C_2^1 = \begin{pmatrix} . & GAP & PHE_2 \\ GAP & - & - \\ PHE_2 & C & - \end{pmatrix} \quad \text{y} \quad C_2^2 = \begin{pmatrix} . & GAP & PHE_1 \\ GAP & - & - \\ PHE_1 & 0 & - \end{pmatrix}$$

Como C_2^2 no es conectado, ésta va a ser la segunda matriz. Y el resultado de aplicar la función join es el siguiente:

$$C_3^3 = \begin{pmatrix} . & GAP & PHE_2 & PHE_1 \\ GAP & - & - & - \\ PHE_2 & C & - & - \\ PHE_1 & 0 & 0 & - \end{pmatrix}$$

Y ahora, supongamos otra vez que, en los dos grafos de la base existen dos ejes PHE1-ALA y PHE2-ALA. Y que en el nivel 3 tenemos la siguiente submatriz:

$$C_3^4 = \begin{pmatrix} . & GAP & PHE_2 & ALA \\ GAP & - & - & - \\ PHE_2 & C & - & - \\ ALA & 0 & C & - \end{pmatrix}$$

Como C_3^4 es conectado y C_3^3 no, la primera sería Xk y la segunda Yk (por las condiciones 3 y 4) . Luego mediante la operación join se obtendría la siguiente submatriz:

$$C_4^2 = \begin{pmatrix} . & GAP & PHE_2 & ALA & PHE_1 \\ GAP & - & - & - & - \\ PHE_2 & C & - & - & - \\ ALA & 0 & C & - & - \\ PHE_1 & 0 & 0 & C & - \end{pmatrix}$$

Si no hubieramos generado la matriz C33 sería imposible encontrar la C24 que es un grafo conectado valido.

5. Aplicación a la búsqueda de detalles estructurales en familias de proteínas

Para poder aplicar el algoritmo a la búsqueda de Detalles Estructurales en una familia de proteínas y así encontrar aquellos representativos de la familia proponemos seguir los siguientes pasos:

1. Seleccionar proteínas y organización de las mismas en conjuntos según su clasificación de familias.
2. Generar grafos y archivos de entrada para el algoritmo de AcGM con las proteínas seleccionadas.
3. Ejecutar el algoritmo con las entradas generadas.
4. Seleccionar subgrafos frecuentes maximales entre todos los subgrafos producidos por el AcGM.

5.1 Selección y organización de proteínas

En nuestro caso, el AcGM fue aplicado para obtener los detalles estructurales comunes entre proteínas que se encuentran clasificadas en la misma familia en la base de datos SCOP.

Basado en estas clasificaciones, se seleccionaron varias proteínas las cuales fueron organizadas en dos niveles, donde el nivel alto es el grupo que corresponde a la superfamilia de SCOP y el nivel bajo correspondiente a la familia, que denominaremos clase. Una vez elegidas las proteínas se hizo una depuración de estas seleccionando aquellas que tengan una identidad de secuencia alta, de manera de elegir varias estructuras de una misma proteína. En este sentido definimos que dos estructuras corresponden a la misma proteína si tienen una identidad de secuencia mayor a 95%. Para alinear las proteínas y calcular la identidad de secuencia se utilizó el algoritmo BLAST.

5.2 Generación de grafos

Para poder generar grafos a partir de las proteínas debe decidirse qué atributos van a ser representados como nodos y qué interacciones entre esos atributos como ejes. Una vez decididas las unidades y tipo de interacción, se deben construir conjuntos de nodos posibles y caracterizar los ejes que van a ser establecidos entre ellos.

5.2.1 Selección de atributos

Se pueden construir varios grafos a partir de una proteína variando la porción de grafos que representa cada nodo y el tipo de interacciones, cada eje.

5. Aplicación a la búsqueda de detalles estructurales

Se propusieron varias unidades estructurales que podrían ser representadas como nodos:

1. átomos: en química molecular, esta es una opción válida ya que la cantidad de átomos que puede tener una molécula ronda los veinte y no supera los cientos. Sin embargo, una proteína está constituida por varios miles de ellos, por lo que puede traer graves problemas en el tiempo de cómputo y uso de recursos. Además, estos átomos que forman parte de aminoácidos no sufren grandes cambios en su estructura, por lo que tampoco sería una opción respaldada desde el punto de vista biología estructural.
2. aminoácidos: esta es una opción muy popular en este tipo de investigación. Generalmente, una proteína consta de alrededor de cientos de aminoácidos. Y como fue expuesto en la parte introductoria, los mayores cambios estructurales son generados por interacciones entre aminoácidos.
3. clases de aminoácidos: los aminoácidos pueden ser clasificados en varios grupos según las características que comparten. Esta opción es válida ya que en la evolución de la secuencia los aminoácidos suelen ser reemplazados por otro del mismo grupo.
4. detalles estructurales: esta es una opción que fue considerada para reducir el tiempo computacional y el uso de recursos. Pero no se pudieron elegir límites, en cuanto al tamaño, y las características que estas deberían tener. Además, esta opción implicaba tener detalles estructurales preparados, que en realidad es a lo que se quiere llegar.

Se decidió que las clases de aminoácidos serían representadas como nodos de grafos.

Los ejes deben representar algún tipo de interacción entre partículas de la unidad que se hayan elegidos para ser representados en nodos.

Una vez que fue elegido la unidad para los nodos, se propusieron varios tipos de interacciones:

1. enlace
2. contacto
3. ángulo *phi* y *psi*
4. puente hidrógeno
5. puente salino[2]

Por cuestión de simplicidad se eligió trabajar con enlaces y contactos en esta tesis. En trabajos futuros pretendemos utilizar otros atributos.

[2] Dos átomos con carácter iónico o iones (un anión y un catión) pueden permanecer unidos en virtud de la atracción electrostática que se establece entre ambos, al poseer carga opuesta. Decimos entonces que se ha formado un enlace iónico. Este tipo de enlaces se presentan comúnmente en los cristales de compuestos salinos como el cloruro de sodio y se conocen también como puentes salinos.

5.2.2.1 Nodos

Cada aminoácido de la proteína corresponde a un nodo del grafo y los ejes son establecidos entre estos nodos si hay una interacción entre ellos.

Hay aminoácidos que son muy similares entre sí en cuanto a sus características fisicoquímicas y pueden realizar interacciones similares con sus pares. De esta manera se simplifica el problema y además . Definimos el conjunto de etiquetas de los nodos *V(G)* considerandos los siguientes grupos: agrupamiento de la siguiente manera:

- VAI si el aminoácido correspondiente es Val, Leu o Ile ().
- ST si es Ser o Thr.
- GAN si es Gln o Asn
- GAP si es Glu o Asp
- LA si es Lys o Arg
- El resto (Ala, Pro, Gly, Cys, Tyr, Phe, Met,Trp, His) mantiene su nombre

Así la cantidad de etiquetas posibles de los nodos es 14 y no 20 como en el caso de usar directamente aminoácidos.

5.2.2.2 Ejes

Los ejes se establecen entre dos nodos si están a una distancia menor a un valor determinado, descartando los ejes triviales como los existentes entre aminoácidos consecutivos en la secuencia ya que los aminoácidos consecutivos en la cadena no aportan un conocimiento que no se encuentre ya codificado en la secuencia y no es lo que pretendemos con esta implementación. Como un aminoácido está formado por varios átomos, fue necesario definir un punto desde el cual calcular las distancias. La manera más simple, que no agrega cálculos extras, es tomar las coordenadas del carbono alfa del aminoácido. Pero esta manera no estaría representando la posición del aminoácido ya que el centro de coordenadas se ve corrido como consecuencia de los residuos de las cadenas de laterales. Una forma sencilla y adecuada es calcular el centro geométrico del aminoácido como el punto desde el cual tomar las distancias. Este consiste en calcular el promedio de cada coordenadas en *x* , *y* y *z*.

$$(c_x, c_y, c_z) = \left(\sum_{i=1}^{n} x_i \Big/ n \;,\; \sum_{i=1}^{n} y_i \Big/ n \;,\; \sum_{i=1}^{n} z_i \Big/ n \right)$$ donde *n* es la cantidad de átomos del aminoácido.

Flexibilidad de proteínas:

Una característica de las proteínas es que son flexibles, adquieren diferentes conformaciones al encontrarse en solución acuosa. Dada esta flexibilidad intrínseca la información estructural que se encuentra en el PDB para una misma proteína que fue cristalizada más de una vez, aunque sea la

secuencia idéntica, difiere en la posición relativa de los átomos que la componen. (Fig 5.1) Hay ciertas proteínas cuya estructura se determinó en diferentes condiciones (Con diferentes ligandos, con diferentes modificaciones en los aminoácidos, o con mutaciones puntuales) lo que genera aún una diferencia mayor. Es de esperar que estos cambios modifiquen las distancias entre los nodos y entonces se modifiquen el número de enlaces entre nodos, si usamos como propiedad una distancia particular entre enlaces. Distintas estructuras de una misma proteína entonces son una herramienta excelente para nuestro análisis ya que al querer comparar entre diferentes proteínas estos cambios son un problema, generando un ruido no deseado. Para analizar las estructuras y obtener los subgrafos importantes conservados en una misma proteína (Secuencia idéntica o casi idéntica) procedimos a analizar entonces primero en conjunto diferentes estructuras de la misma proteína, y así obtener subgrafos que no se modifican por pequeños cambios en la conformación de ellas.

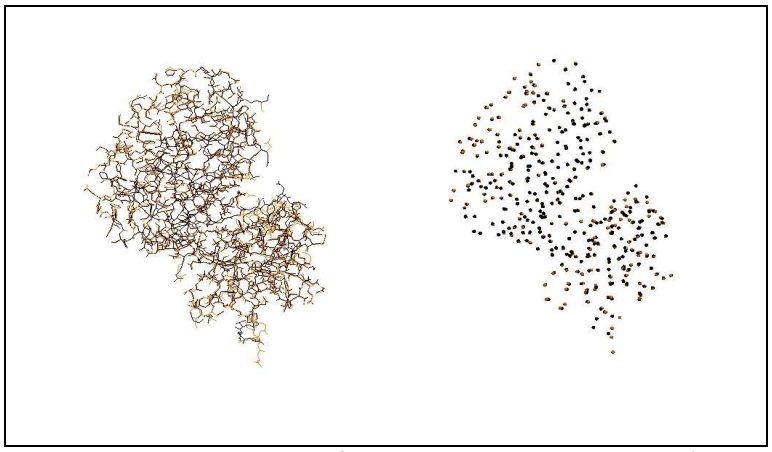


Fig. 5.1 Estructura de una proteína cristalizada de dos maneras diferentes donde se observa que hay diferencias en la posición de los átomos. Izquierda incluyendo todos los átomos y a la derecha solo los carbonos alfa.

Distancia entre nodos para determinar un enlace:

Luego procedimos a elegir una distancia que nos permita establecer si hay un enlace o no entre nodos. Para ello cargamos la base de datos con el conjunto de proteínas pertenecientes al TOP500 (http://kinemage.biochem.duke.edu/databases/top500.php) que son usados comúnmente para representar el espacio conformacional de todo el PDB ya que se supone que es un conjunto de proteínas lo suficientemente diversas para cubrir dicho espacio. Calculamos las distancias entre centros geométricos para todos los aminoácidos entre ellos. De esta manera encontramos que un primer máximo aparece ha distancias entre 4.0 y 7.0 Å (Ver los gráficos del apéndice). Decidimos usar, como primer aproximación, distancias que sean dependientes del tipo aminoácido, usar 5.5 Å. Sin embargo, esta decisión puede ser modificada en trabajos futuros.

Como se ve en la figura 5.2, a distancias cortas, 2 Å, no se forman enlaces, y se empiezan a formar generando grafos interconectados y con una densidad de enlaces razonables (alrededor de 3 enlaces por nodo) con distancias entre 5.0 y 6.0 Å.

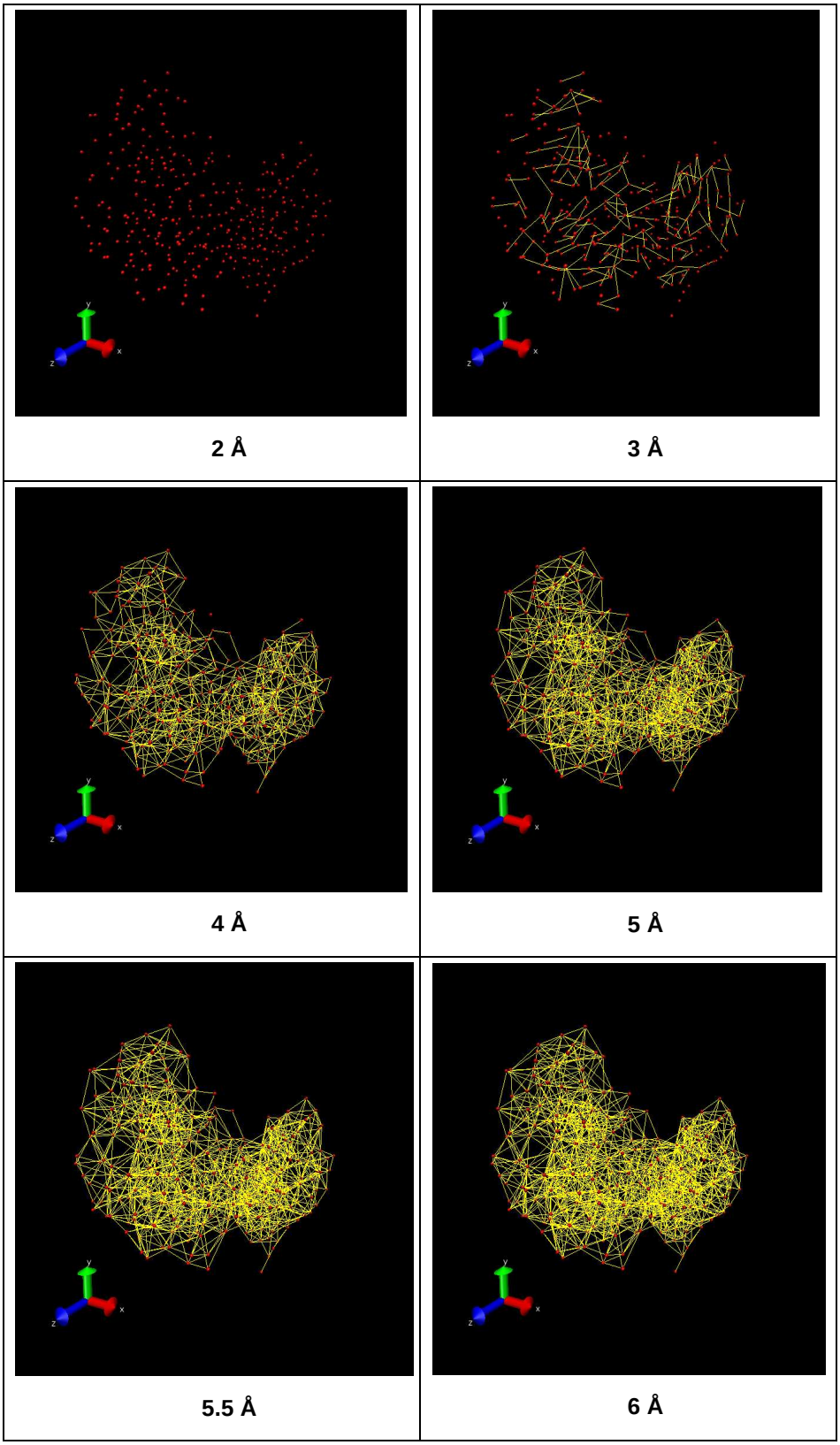


Fig. 5.2 Los centros geométricos de los aminoácidos de la proteína 1A9U
y los enlaces que se van agregando de acuerdo a las distancias entre estos centros geométricos.

5.2.3 Construcción de base de datos

Se construyó una base de datos para poder generar estos grafos de forma eficiente. Esta base contiene información obtenidas de Protein DataBank. La tabla construida está organizada de manera que cada entrada (registro) contenga datos de un átomo de una proteína. Entonces cada entrada tiene almacenada la siguiente información del átomo correspondiente:

- nombre de la estructura
- nombre del átomo
- nombre del aminoácido al que pertenece
- número de la secuencia del aminoácido
- el valor de coordenada en x
- el valor de coordenada en y
- el valor de coordenada en z

Con los datos provistos por esta tabla en la base se construyen archivos de entrada al AcGM.

Es un solo archivo de entrada para cada conjunto de grafos entre los cuales se desea encontrar subgrafos isomorfos.

5.2.4 Construcción de archivo de entrada

El archivo de entrada al AcGM debe contener todos los grafos entre los cuales se quiere encontrar subgrafos isomorfos. En nuestro caso son las estructuras seleccionadas y organizadas como se expuso en la sección 5.1.

Los nodos de un grafo llevan además del identificador, que en este caso es el número de secuencia dentro de la estructura, el nombre del grupo del aminoácido correspondiente a este número de secuencia.

Los ejes son agregados al archivo si los centros geométricos calculados con los datos provistos por la base de dos nodos están a menos de 5.5 A de distancia.

5.3 Ejecución del AcGM para los conjuntos de entrenamiento

El AcGM cuenta con dos parámetros:

1. *Support*: como se explicó en la parte de métodos, este parámetro establece un límite inferior del porcentaje de grafos en los cuales los subgrafos deben ser encontrados.
2. *Isubgraph*: este parámetro permite elegir tipos de subgrafos que uno quiera obtener. Con este parámetro establecido este el algoritmo devuelve solamente subgrafos inducidos.

En nuestro caso el valor de *support* lo fijamos en 100%. Las proteínas agrupadas en el archivo de entrada son aquellas que están filtradas por BLAST, por lo que se puede pedir el valor de *support* alto para encontrar subgrafos isomorfos comunes en todos los grafos de la familia.

También pedimos que los subgrafos resultantes sean subgrafos inducidos. Esta opción garantiza que los nodos de los subgrafos devueltos estén unidos exactamente con los mismos ejes.

Los pasos necesarios para tener un archivo de salida con los subgrafos que nos interesan son los siguientes:

1. Ejecutar el algoritmo AcGM con el valor de parámetro *support* mínimo e *isubgraph*.
2. Tomar los subgrafos resultantes de la salida de la ejecución del AcGM y encontrar las instancias de los mismos en cada uno de los grafos con el algoritmo SUBDUE y devolver un archivo para cada par subgrafo-instancias encontrada en un grafo.
3. tomando las salidas del ítem 2 organizarlos en una planilla para que se pueda visualizar todas las instancias de un subgrafo en todos los grafos de entrada.

Los ítems 2 y 3 son necesarios porque nos interesa poder analizar todas las instancias de los detalles estructurales encontrados en todas las proteínas. Como es de esperar, la salida del AcGM consiste en subgrafos isomorfos frecuentes encontrados y la cantidad de grafos en los que fueron encontrados cada subgrafo. El formato de un subgrafo en el archivo de salida es el siguiente:

```
size=4
VertexLabels  = CYS,TYR,ST,VAI,
code = C,C,0,0,0,C,
count = 21
```

Fig. 5.3 Un fragmento del archivo de salida

Este fragmento del archivo de salida representa un subgrafo isomorfo de 4 nodos que fue encontrado en 21 de los grafos que fueron ingresados en el archivo de la entrada. La forma matricial de este fragmento sería:

$$\begin{bmatrix} - & CYS & TYR & ST & VAI \\ CYS & - & - & - & - \\ TYR & C & - & - & - \\ ST & C & 0 & - & - \\ VAI & 0 & 0 & C & - \end{bmatrix}$$

Los C en las celdas significan que los dos nodos involucrados en esas celdas comparten un enlace en este subgrafo isomorfo frecuente encontrado.

5.4 Selección de subgrafos maximales

El AcGM brinda la información sobre las etiquetas de los nodos y tipo de relación entre cada par de nodos (0: no hay interacción, C: están a una distancia menor a un valor determinado) para subgrafo frecuente encontrado. Pero para nuestro caso también es interesante poder analizar cada instancia de estos subgrafos presentes en los grafos. Para eso se necesitan datos de números (posición) de la secuencia de los aminoácidos involucrados en cada subgrafo, cuántos del mismo se encuentra en el grafo, etc. Esta tarea es realizada por el algoritmo SUBDUE que dado dos grafos evalúa si un grafo es subgrafo del otro.

Otro asunto es que el AcGM devuelve todos los subgrafos frecuentes aún cuando uno de ellos es a la vez un subgrafo de otros subgrafos. Llamaremos subgrafo maximal de un conjunto de grafos a un subgrafo isomorfo frecuente que no es subgrafo de ningún otro subgrafo del mismo conjunto. Si fue encontrado un subgrafo maximal de tamaño 5, el AcGM devuelve también los subgrafos que fueron utilizados para construir este según el método explicado en la sección de métodos, por lo que la salida del AcGM va a contener además, subgrafos de tamaño 1 al 4, que a la vez son subgrafos de aquel subgrafo maximal. Este hecho de devolver la información de subgrafos no maximales además de las maximales trae una consecuencia grave en el performance del algoritmo a la que se hará mención en secciones posteriores.

Como en nuestro caso los únicos subgrafos que deseamos analizar son aquellos que son maximales, desarrollamos un filtro de subgrafos para seleccionar solo aquellos que son maximales. Este procedimiento es sencillo ya el AcGM devuelve los grafos ordenados por tamaño.

5.5 Ejecución de AcGM para la validación

Una vez obtenidos los resultados finales (subgrafos maximales frecuentes) del conjunto de grafos de entrenamiento es necesario ver si los subgrafos predichos son representativos para detectar una familia.

Se crearon para cada subgrafo maximal frecuente obtenido un archivo de entrada al algoritmo AcGM, agregando este subgrafo como un grafo más del conjunto de grafos de prueba (donde estos grafos de prueba son distintos a aquellos que fueron utilizados para obtener los subgrafos isomorfos). Luego se ejecuta el algoritmo con el archivo de entrada, con *support* 100%. Si en el resultado de esta ejecución está contenido el subgrafo maximal insertado, esto quiere decir que este subgrafo esta presente en todas las proteínas de la familia. Y este subgrafo estaría representando un motivo estructural de esta familia.

En el caso de que este subgrafo no sea recuperado en su totalidad, es decir, se pierden algunos nodos o ejes, el motivo estructural representado por este no es estable. Sin embargo, el subgrafo recuperado más chico sí, es un motivo estructural presente en todas las proteínas de la familia.

6. Resultados

6.1 Objetivo del experimento

El objetivo del experimento es comprobar cuan útil es esta herramienta para detectar motivos estructurales con el fin de aportar nueva información en la clasificación de estructuras proteicas en distintas familias.

Esta herramienta será utilizada para realizar tareas de aprendizaje supervisado sobre proteínas de algunas familias seleccionadas. Se eligieron proteínas con conocimiento previo sobre la familia a la que pertenece cada una. Se agruparon de acuerdo a estas familias y cada uno de estos grupos fue tomado como entrada a la herramienta. Los resultados fueron tomados y analizados desde distintos puntos de vista para comprobar la utilidad de esta herramienta.

6.2 Observaciones generales del conjunto de dominios elegidos para el experimento

Grupos de evaluación seleccionados

Se eligieron 4 dominios de kinasas para probar el poder de clasificación del algoritmo. Estos son: MAP kinasa P38 (P38), MAP kinasa ERK2 (ERK2), kinasa dependiente de ciclinas-2(CDK2) y c-src tyrosine kinase (C-SRC).

Cada grupo consiste de estructuras pertenecientes a uno de estos dominios. Las estructuras fueron elegidas de acuerdo a su dominio. Se seleccionaron aquellas estructuras que comparten más de 95% de la identidad de secuencias con el resto de las estructuras del mismo dominio para grupos de aprendizaje.

Las proteínas pertenecientes a la familia de las kinasas tienen estructuras tridimensionales parecidas, pero sin embargo tienen funciones bien diferentes dentro de la célula. Algunas kinasas, además de tener el dominio kinasa tienen otros dominios que ejercen otras funciones. Las proteínas p38, ERK2 y CDK2 tienen solo dominio kinasa, sin embargo c-src tiene además dos dominios SH1 y SH2, que han sido cristalizados en forma independiente. Para realizar nuestro trabajo elegimos en el grupo de c-src para el entrenamiento estructuras con el dominio SH2, esto es útil ya que luego usaremos en el grupo validación una estructura de c-src que contiene todos los dominios, lo cual es un desafío para el programa ya que el dominio kinasa no se encuentra en los subgrafos del grupo entrenamiento en c-src pero si en los otros tres.

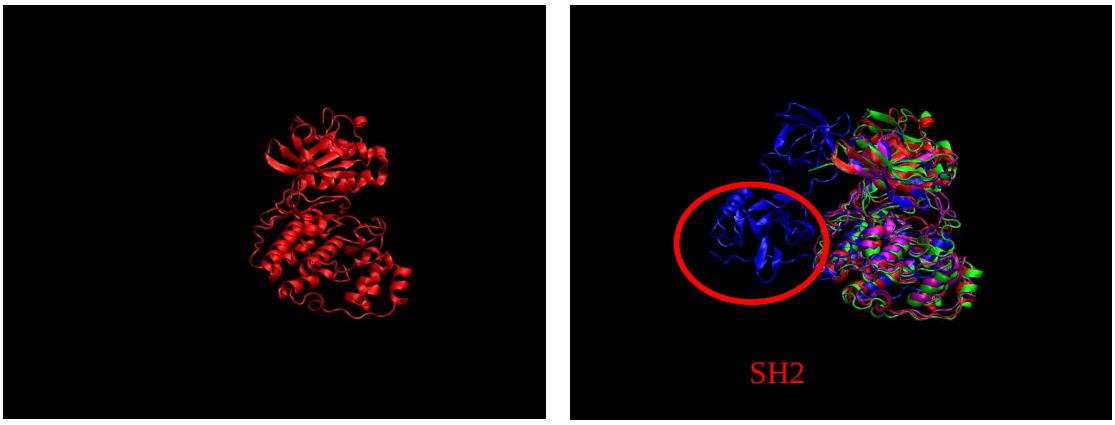


Fig. 6.1 Superposición de dominios.
La imagen de la izquierda muestra la estructura de ERK2 que consta de un dominio kinasa. La imagen de la derecha muestra la superposición de de las estructuras de ERK2, p38, CDK2 y C-SRC donde se pueden observar la similitud de los dominios kinasa y que C-SRC cuenta ademas con dos dominios más.

La figura 6.1 muestra la estructura de una kinasa, ERK2, y luego la superposición con las otras tres. En azul se muestra la estructura de c-src que tiene dos dominios extras, uno es el SH2 usado en el grupo entrenamiento para c-src; la estructura azul completa se usara en el grupo validación.

Los dominios tienen una identidad de secuencias menor a 50% entre ellos. La tabla 6.1 muestra los valores obtenidos para cada par de dominios seleccionados.

SeqA	Name	Len(aa)	SeqB	Name	Len(aa)	score(%)
1	P38	379	2	ERK2	365	47
1	P38	379	3	CDK2	298	37
1	P38	379	4	C-SRC	452	15
2	ERK2	365	3	CDK2	298	36
2	ERK2	365	4	C-SRC	452	12
3	CDK2	298	4	C-SRC	452	19

Tabla 6.1 La identidad de secuencias entre familias

Se puede observar que la identidad mas alta es entre P38 y ERK2 que tiene 47%. El dominio C-SRC es el que tiene la menor identidad para el resto de los dominios.

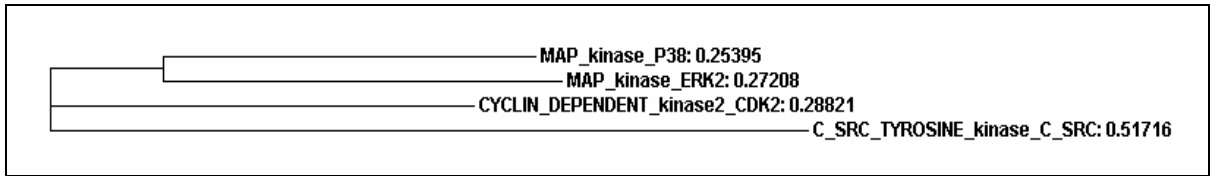


Fig. 6.2 La identidad de secuencias entre los dominios seleccionados.

En el árbol filogenético[1] de la figura 6.2 se observa la distancia medida en porcentaje de identidad, entre cada dominio utilizado en el experimento. El dominio C-SRC es el que más diferencia muestra con el resto de los dominios.

Se desea comprobar la potencia de esta herramienta a través de experimentos con dominios que son similares secuencialmente.

Observaciones de los resultados: El algoritmo devuelve en el archivo de salida una sola instancia por cada subgrafo isomorfo encontrado. Es decir, si el algoritmo encuentra un subgrafo isomorfo en la base de grafos no importa si este se repite dos o más veces en todas las proteínas de la base, devuelve una sola instancia. Como consecuencia, solo se encuentran 14 o menos subgrafos con un nodo en cada resultado, ya que la cantidad de grupos en los que se dividieron los aminoácidos es 14.

6.3 Resultados de experimentos por cada dominio

6.3.1 MAP Kinase P38

Fold: protein kinase-like(PK-like)
Superfamilia: protein kinase-like(PK-like)
Familia: Protein kinases, catalytic subunit
Dominio: MAP kinase P38
Especie: human
Cantidad de aminoácidos en las estructuras: entre 350 y 355
Cantidad de enlaces formados con 5.5 Å de distancia: alrededor de 450

6.3.1.1 Resultados de la ejecución

3410 subgrafos encontrados
subgrafo mas grande: 17 nodos
125 subgrafos maximales
Tiempo de ejecución: 36.360835 segundos

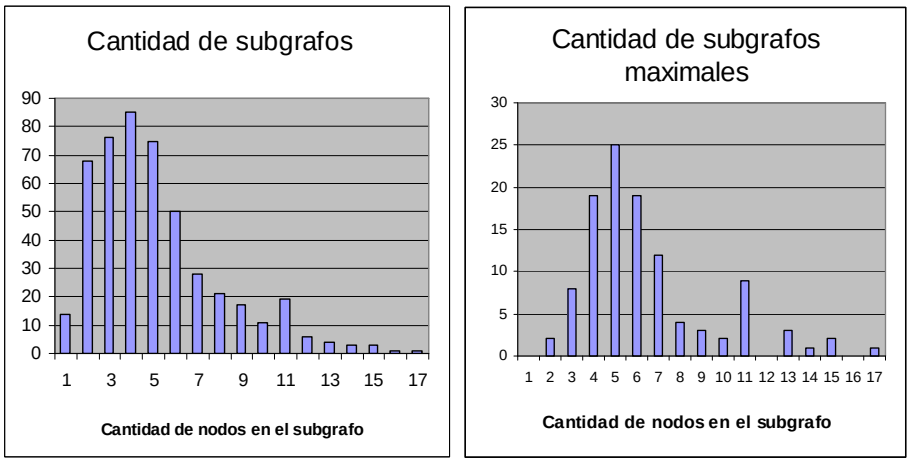


Fig. 6.3 Cantidad de subgrafos generados en cada nivel vs. Cantidad de subgrafos maximales generados en cada nivel en la familia p38

Se puede observar que la cantidad de subgrafos va creciendo hasta que la cantidad de nodos llega a ser 4 y luego disminuye. No es difícil descubrir la razón por la que la cantidad de subgrafos decrece cuando el subgrafo crece en tamaño, dado que es más difícil encontrar un subgrafo isomorfo de tamaño grande que aquellos de tamaño menor. Pero no es fácil notar por qué crece en algunos casos, por ejemplo el algoritmo encontró 68 subgrafos de 2 nodos y 76 de 3 nodos. Hay dos explicaciones: 1) porque el algoritmo imprime solo una instancia de cada conjunto de subgrafos isomorfos que encuentran en la base. 2) como consecuencia de *backtracking* que el algoritmo realiza en BFS, de un subgrafo de 2 nodos pueden derivar varios de 3 nodos. Por ejemplo, en el subgrafo mas grande encontrado en esta clase (figura 6.4), se puede observar que del subgrafo de dos nodos VAI(8)-VAI(13) pueden derivar dos subgrafos de 3 nodos: VAI-VAI-GAP y VAI-VAI-ST. Se puede observar que más de la mitad de los subgrafos de tamaño 1 a 12 no son subgrafos maximales.

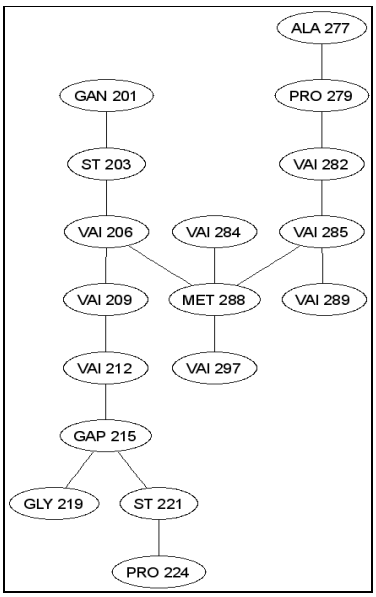


Fig. 6.4 Subgrafo de 17 nodos encontrado en p38.

La figura 6.4 muestra en grafo 2D uno de los subgrafos más grandes encontrados en la base de grafos de clase2. Se puede observar que la cantidad de ejes que une a cada nodo es pequeña. Tampoco se observa ningún ciclo dentro del subgrafo isomorfo frecuente obtenido.

6.3.2 MAP kinase ERK2

Fold: protein kinase-like(PK-like)
Superfamilia: protein kinase-like(PK-like)
Familia: Protein kinases, catalytic subunit
Dominio: MAP kinase ERK2
Especie: humano
Cantidad de aminoácidos en las estructuras: alrededor de 360 aminoácidos
Cantidad de enlaces formados con 5.5 Å de distancia: alrededor de 450 enlaces

6.3.2.1 Resultados de la ejecución

10903 subgrafos encontrados
subgrafo mas grande: 22 nodos
340 subgrafos maximales
Tiempo de ejecución: 15.450694 Segundo

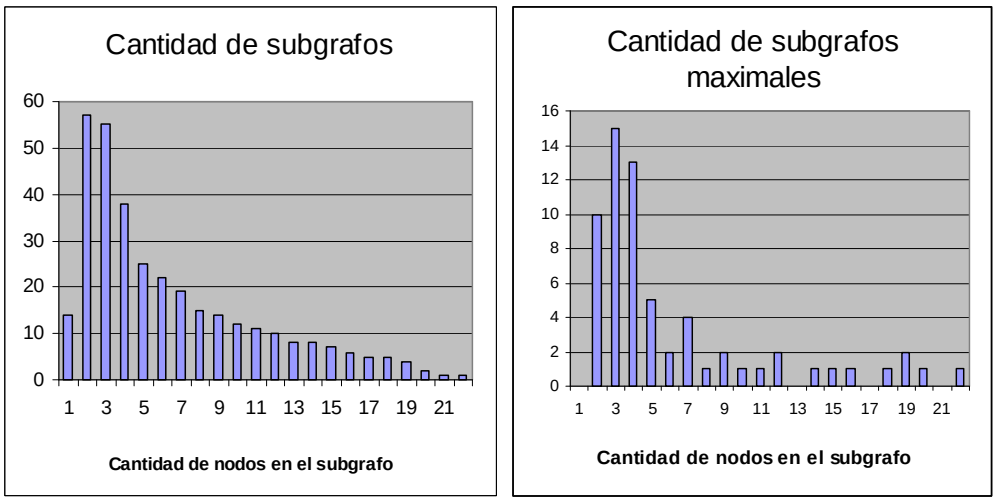


Fig. 6.5 Cantidad de subgrafos generados en cada nivel vs. Cantidad de subgrafos maximales generados en cada nivel en la familia ERK2

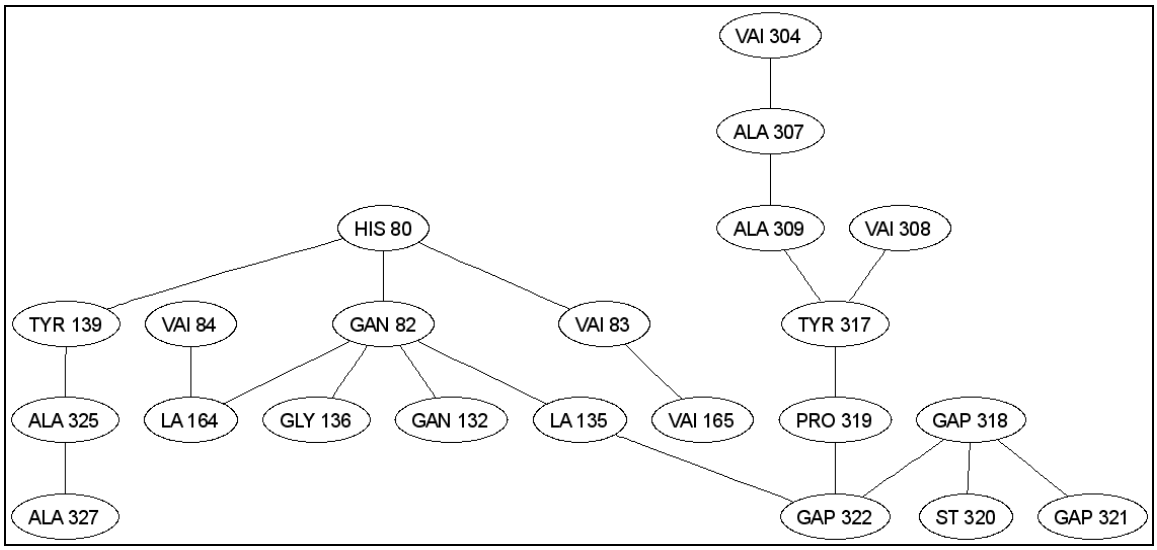


Fig. 6.6 Subgrafo de 22 nodos del grupo ERK2

El subgrafo más grande obtenido tiene 22 nodos. Se puede observar que el detalle estructural que encuentra en esta familia tiene información que involucra a varios aminoácidos que no se encuentran cerca en secuencia. Podemos observar como una zona geográfica de una estructura involucra aminoácidos que están secuencialmente lejos. Por ejemplo, podemos observar que un aminoácido que esta en la posición 139 está a menos de 5.5 A con un aminoácido que esta en la posición 325. Esto demuestra que el problema como está presentado y encarado, y el método utilizado para resolver el problema tiene éxito para identificar detalles estructurales en 3D y obtener informaciones sobre características que no son locales en secuencia.

6.3.3 Cyclin-dependent protein kinases, CDK2

Fold: protein kinase-like(PK-like)
Superfamilia: protein kinase-like(PK-like)
Familia: Protein kinases, catalytic subunit
Dominio: Cyclin-dependent PK, CDK2
Especie: human
Cantidad de aminoácidos en las estructuras: entre 294 y 298
Cantidad de enlaces formados con 5.5 Å de distancia: alrededor de 260

6.3.3.1 Resultados de la ejecución

562 subgrafos encontrados
subgrafo mas grande: 11 nodos
78 subgrafos maximales

Tiempo de ejecución: 2.926568 segundos

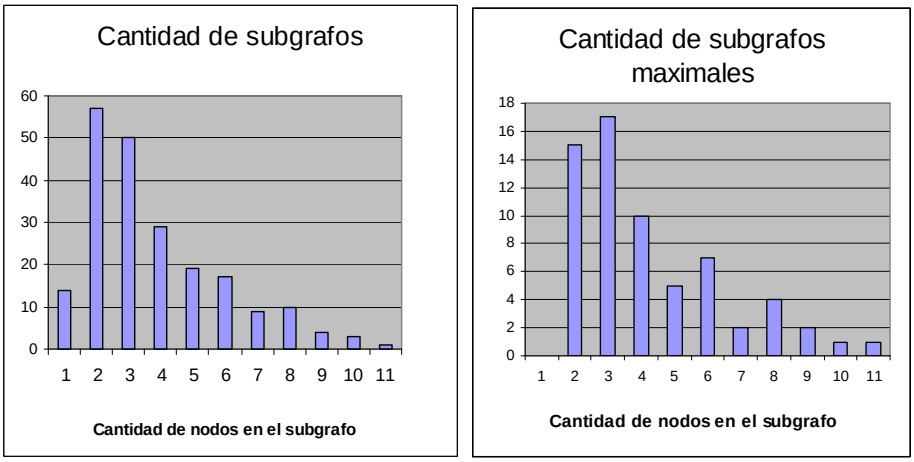


Fig. 6.7 Cantidad de subgrafos generados en cada nivel y Cantidad de subgrafos maximales generados en cada nivel en la familia CDK2

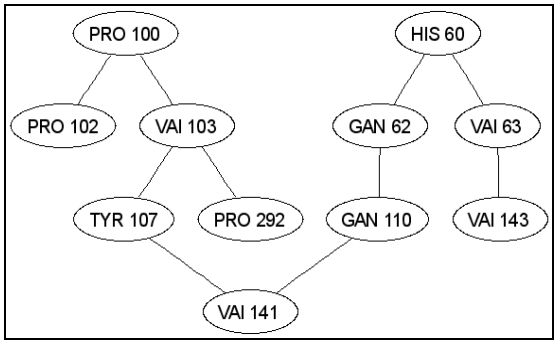


Fig. 6.8 Subgrafo de 11 nodos encontrado en cdk2.

Se puede observar como los aminoácidos de distintas partes de la estructura están combinados en un detalle estructural.

6.3.4 C-SRC Tyrosine kinase

Fold: protein kinase-like(PK-like)
Superfamilia: protein kinase-like(PK-like)
Familia: Protein kinases, catalytic subunit
Dominio: c-src tyrosine kinase
Especie: human
Cantidad de aminoácidos en las estructuras: entre 105 y 450

Cantidad de enlaces formados con 5.5 Å de distancia: entre 120 y 590
Cantidad de grafos en el archivo de entrada: 25

6.3.4.1 Resultados de la ejecución

30298 subgrafos encontrados
subgrafo mas grande: 22 nodos
428 subgrafos maximales
Tiempo de ejecución: 36.117565

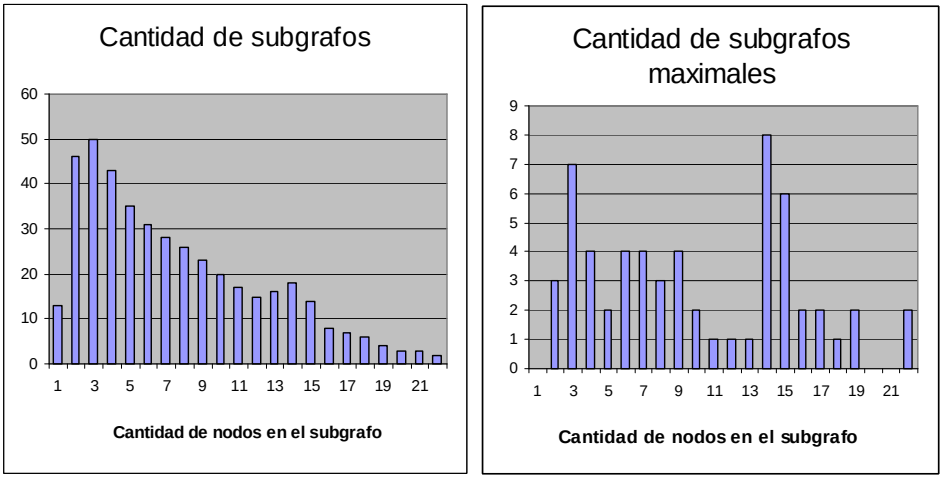


Fig. 6.9 Cantidad de subgrafos generados en cada nivel y cantidad de subgrafos maximales generados en cada nivel en la familia C-SRC

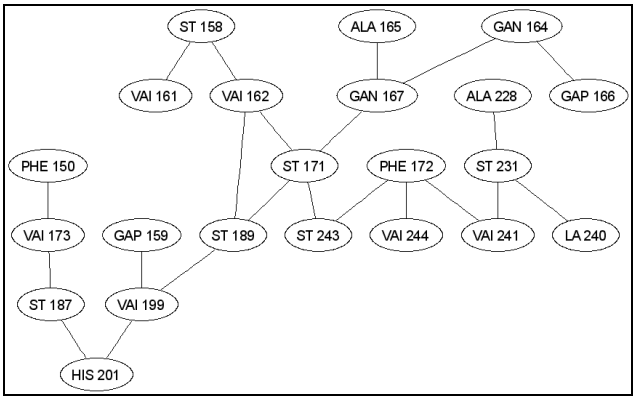


Fig. 6.10 Subgrafo de 22 nodos en 2SRC[3]

[3] Observación: se observan los números de aminoácidos mayores a 105 a pesar de que la cantidad de nodos la mayoría de las estructuras son alrededor de 105. Esto no es un error, la base PDB permite gaps en la secuencia. En el archivo de entrada los números de posiciones que no tienen ningún aminoácido asignado se le asigna la etiqueta ´NON´. Esta etiqueta no aparece en ningún subgrafo isomorfo frecuente ya que estos no tienen enlaces con ningún otro.

Se puede observar que el par mas alejados en la secuencia de este detalle estructural están a 94 aminoácidos.

6.3.5 Observaciones sobre tiempos de ejecución

Los tiempos de ejecución en los cuatros casos de evaluación son razonables teniendo en cuenta que la cantidad de subgrafos y los tamaños de los subgrafos isomorfos que encuentra. Estos valores, junto a la cantidad de nodos promedio de subgrafos son notablemente mayores a los presentados en el trabajo de Inokuchi [37]. Sin embargo el algoritmo pudo dar resultados correctos en un tiempo de ejecución razonable.

El tiempo de ejecución de la familia ERK2 es notablemente menor al de la ejecución con la familia de P38. Esto se debe a que la cantidad de grafos en la base es menor a un cuarto de la de la familia P38. Si bien el promedio de nodos en los grafos y la cantidad de ejes son parecidos, y aún más, la cantidad de nodos en los subgrafos encontrados es mayor en esta familia, el tiempo de buscar los subgrafos en mayor cantidad de grafos puede afectar enormemente en el tiempo de ejecución.

La familia CDK2 presenta una estructura más simple (menos cantidad de nodos y ejes) que las familias presentadas anteriormente. Este hecho reduce el espacio de búsqueda desde el comienzo de la ejecución disminuyendo considerablemente el tiempo de ejecución. Además se puede observar que el subgrafo más grande encontrado tiene 11 nodos. Es decir, la ejecución derivó en 12 niveles. Con la misma lógica se explica el tiempo de ejecución tardado para la familia C-SRC. Se puede observar que hubo 22 niveles de derivaciones y a diferencia de la familia P38, hay mayor cantidad de subgrafos en los niveles posteriores, lo cual complica el proceso de *join*.

6.4 Subgrafos obtenidos en otras familias

Es de esperar que haya motivos estructurales que esten presentes en diferentes proteínas y que haya motivos que sean caracteristicos de una sola proteína, ie que no esten presentes en otras. Nosotros tomamos un conjunto de subgrafos isomorfos frecuentes perteneciente a una proteína que ha sido cristalizada varias veces y evaluamos su tasa de recuperación en un grafo incognita como indicador de similitud entre el grafo y la proteína original. Otra implementación posible, más específica, pero que requiere un trabajo extra al realizado en esta tesis, podría ser la búsqueda de uno o varios subgrafos isomorfos frecuentes que tengan una baja tasa de recuperacion entre todas las otras posibles estructuras y una alta tasa de recuperacion en estructuras de la misma proteína. Estos subgrafos serían entonces una base de datos que se podría usar para diferenciar estructuras de proteínas.

Para eso se hizo un experimento con la misma herramienta utilizada para encontrar los subgrafos isomorfos dentro de un dominio de la siguiente manera:

1) De los subgrafos obtenidos del análisis anterior para los 4 grupos, se eligieron solamente los que tuviesen más de 8 nodos. Esto representa un número significativo de detalles estructurales que contienen un número significativo de nodos.

2) Se seleccionó un conjunto de estructuras de proteínas, mayoritariamente pertenecientes a los grupos anteriores o similares.
3) Se ejecutó nuestra implementación usando cada subgrafo del punto 1) contra cada grafo del punto 2. A la salida nuestro algoritmo nos dirá que porcentaje de un subgrafo del grupo 1 se encuentra en un grafo del punto 2.
4) Se calcula el promedio del porcentaje recuperado para todos los subgrafos pertenecientes a un mismo grupo. Este valor entonces será una medida de la similitud entre la proteína del punto 2 y la proteína perteneciente al grupo de donde se obtuvieron los subgrafos.

A continuación se muestran los resultados. Los números calculados se obtuvieron sacando el promedio de los porcentajes de la cantidad de nodos de los subgrafos recuperados contra la de subgrafos originales de un dominio.

6.4.1 Resultados

Se eligieron proteínas pertenecientes a diferentes familias y se realizó el análisis.

6.4.1.1 MAP KINASE P38 (P38)

Las estructuras elegidas para MAP KINASE P38 son los correspondientes a los ids de PDB 2EWA, 1P38, 1LEZ y 1LEW.

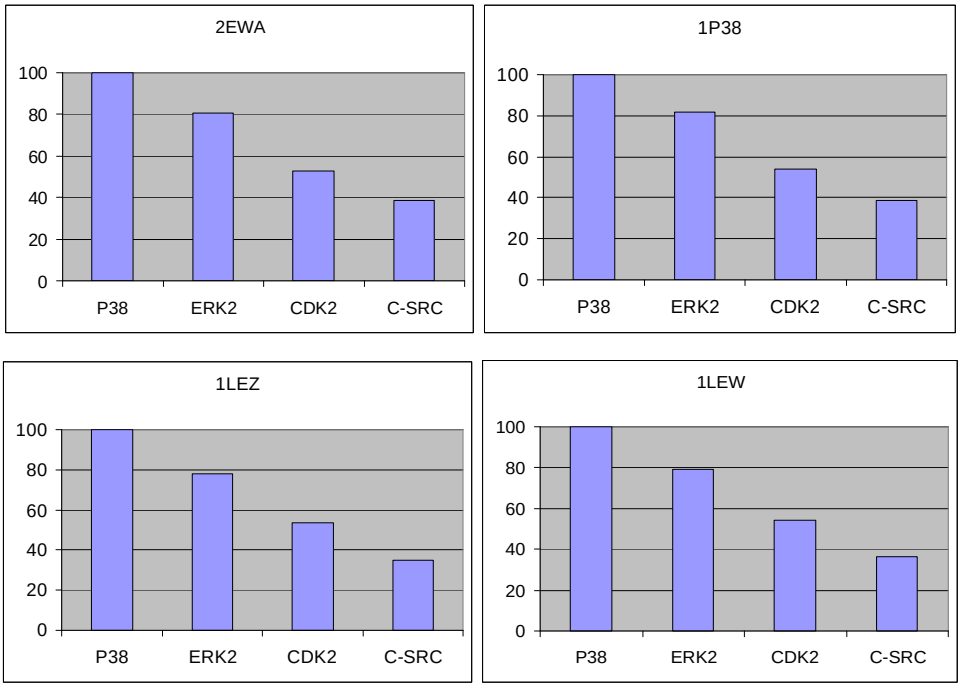


Fig. 6.11 Promedio de porcentajes de cantidad de nodos de subgrafos p38 que se recuperan en cada familia. En los ejes x de cada grafo se muestran los nombres de las familias de los subgrafos isomorfos y en los ejes y las tasas de recuperación en %.

Se puede observar que los promedios de tasas de recuperación de subgrafos originales de MAP Kinase P38 son 100% en los cuatro casos. Es decir, todos los subgrafos isomorfos frecuentes de MAP kinase P38 fueron encontrados en estas cuatro estructuras que también pertenecen al mismo dominio. Sin embargo, hay una presencia alta (alrededor de 80%, esto es alrededor de 20% de diferencia con la tasa de recuperación de los subgrafos del dominio P38) de partes de subgrafos originales del dominio P38 en el dominio ERK2. Los otros dominios, CDK2 y C-SRC, la tasa de recuperación no llega a los 60% en todos los casos. Esto es interesante dado que P38 y ERK2, ambas MAPK, tienen más similitud entre sí que con CDK2 y c-SRC. Se puede ver que claramente el programa identifica esta proteína como p38.

6.4.1.2 MAP KINASE ERK2 (ERK2)

Las estructuras seleccionadas para el dominio MAP KINASE ERK2 son 2ERK, 1GOL y 2GPH.

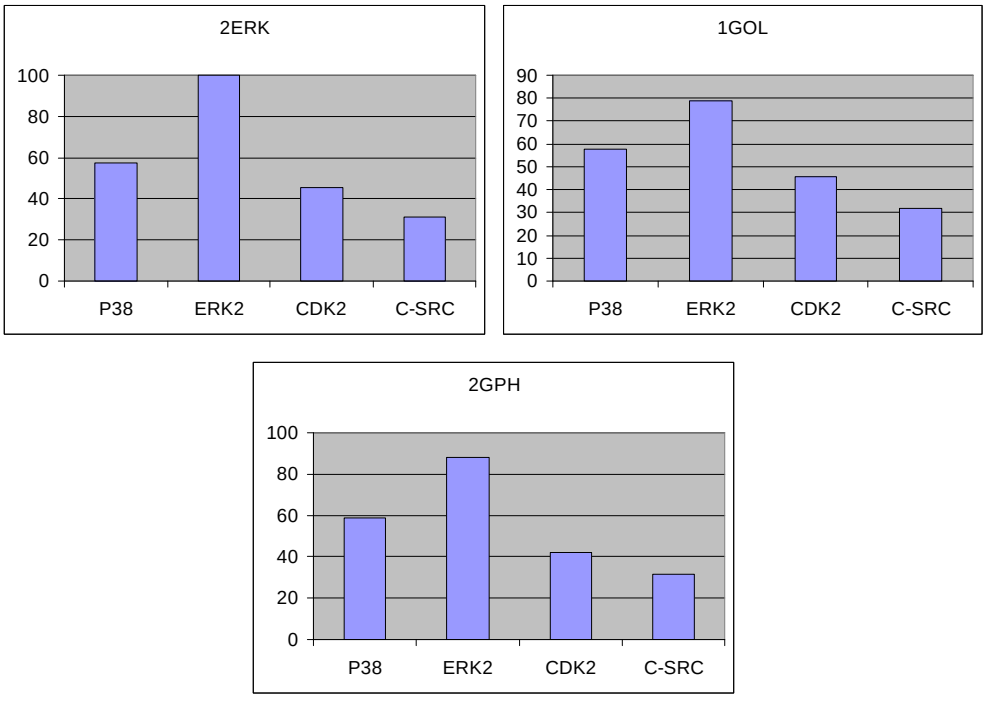


Fig. 6.12 Promedio de porcentajes de cantidad de nodos de subgrafos ERK2 que se recuperan en cada familia

Se puede observar que si bien no en todos los casos se recuperan el 100% de subgrafos en el dominio correspondiente a estas estructuras, MAP KINASE ERK2, es notable la diferencia de tasas de recuperación con el resto de los dominios. Otra vez se puede observar que hay una mayor similitud entre ERK2 y P38, sin embargo el porcentaje recuperado es menor que en el punto anterior, esto indicaría que P38 tiene detalles estructurales extra, no presentes en ERK2. .

6.4.1.3 CYCLIN-DEPENDENT KINASE 2 (CDK2)

Las estructuras elegidas para este dominio son 1H0W, 1KE7, 1OL1 y 1PXN. Este dominio tiene entre 19% y 37% de identidad de secuencias con el resto de los dominios.

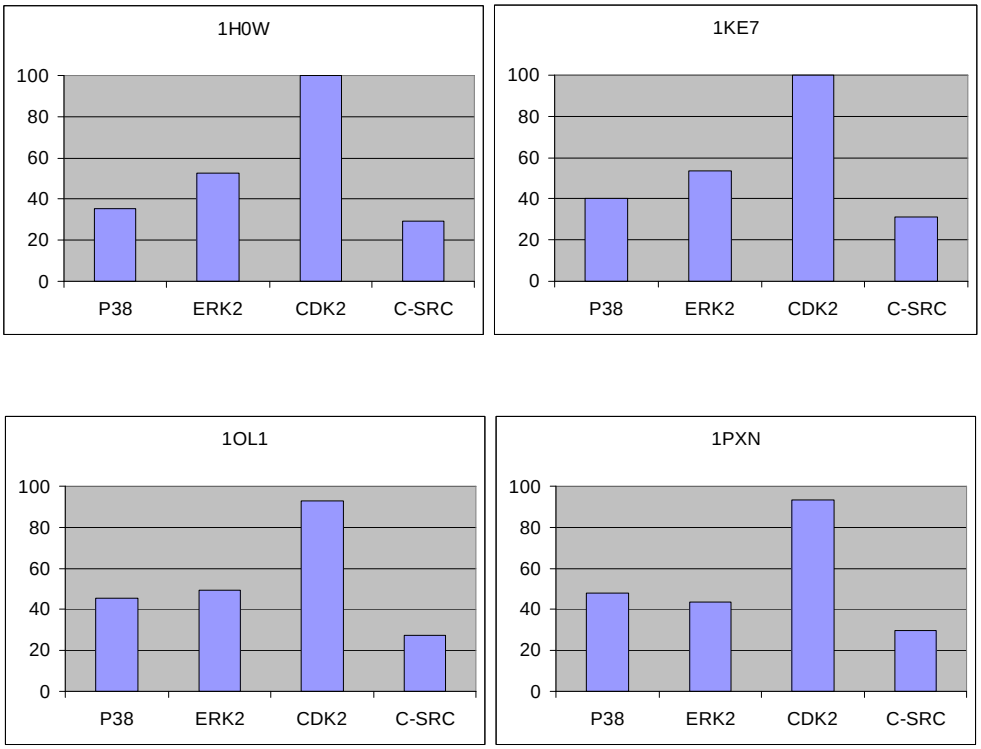


Fig. 6.13 Promedio de porcentajes de cantidad de nodos de subgrafos CDK2 que se recuperan en cada familia

Se puede observar que las tasas de recuperación para el dominio CDK2 es notablemente más alta que cualquier otro dominio, siendo estas más de 90% en las cuatro estructuras.

6.4.1.4 C-SRC Tyrosine KINASE (C-SRC)

Las estructuras elegidas para el dominio C-SRC Tyrosine Kinase son 1FMK, 1042, 104E y 1SHD.

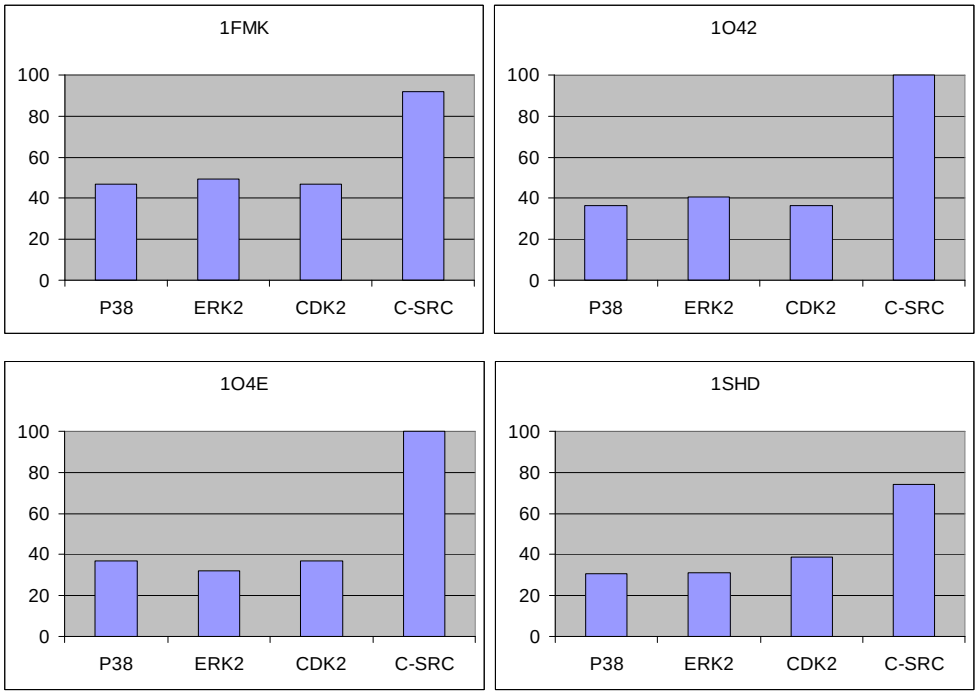


Fig. 6.14 Promedio de porcentajes de cantidad de nodos de subgrafos C-SRC que se recuperan en cada familia

Se puede observar que la presencia de partes de los subgrafos isomorfos frecuentes del dominio C-SRC Tyrosine Kinase no llega a los 50% en el resto de los dominios, y en mayoría menos de 40%.
Es interesante el caso de 1FMK que, como se dijo antes, fue seleccionada especialmente. Esta estructura cuenta con el dominio SH2, usado para la generación de subgrafos de C-SRC, y un dominio kinasa que está presente en los otros grupos. El programa reconoce un porcentaje mayor de subgrafos en P38, ERK2 y CDK2 que las otras tres estructuras que tienen solo SH2, lo cual indica que hay una mayor similitud por tener dominio kinasa pero igual es capaz de diferenciar ya que recupera mucho más en C-SRC.

En los cuatro dominios se pudo observar que las estructuras del experimento contienen los subgrafos isomorfos frecuentes encontrados en otras estructuras del mismo dominio y en el mismo sentido éstas contienen menor porcentaje de subgrafos isomorfos frecuentes de otros dominios. Teniendo en cuenta que los dominios seleccionados para el experimento son muy parecidos podemos decir que nuestro algoritmo es exitoso en diferenciar estructuras muy similares.

6.4.1.5 c-Jun N-Terminal Kinase (JNK)

Las estructuras 2P33, 2NO3, 2B1P y 2G01 pertenecen a proteínas de la familia de c-Jun N-Terminal Kinase (JNK). Ésta no pertenece a ninguna de las familias anteriores aunque como se

muestra en la tabla 6.2 comparten una identidad de secuencia bastante alta con las familias P38 y ERK2.

SeqA	Name	Len(aa)	SeqB	Name	Len(aa)	Score
1	2P33	365	5	P38	379	46
1	2P33	365	6	ERK2	380	39
2	2NO3	370	5	P38	379	44
2	2NO3	370	6	ERK2	380	36
3	2B1P	355	5	P38	379	47
3	2B1P	355	6	ERK2	380	40
4	2G01	370	5	P38	379	44
4	2G01	370	6	ERK2	380	36

Tabla 6.2 Identidades de secuencias de las estructuras de JNK con P38 y ERK2

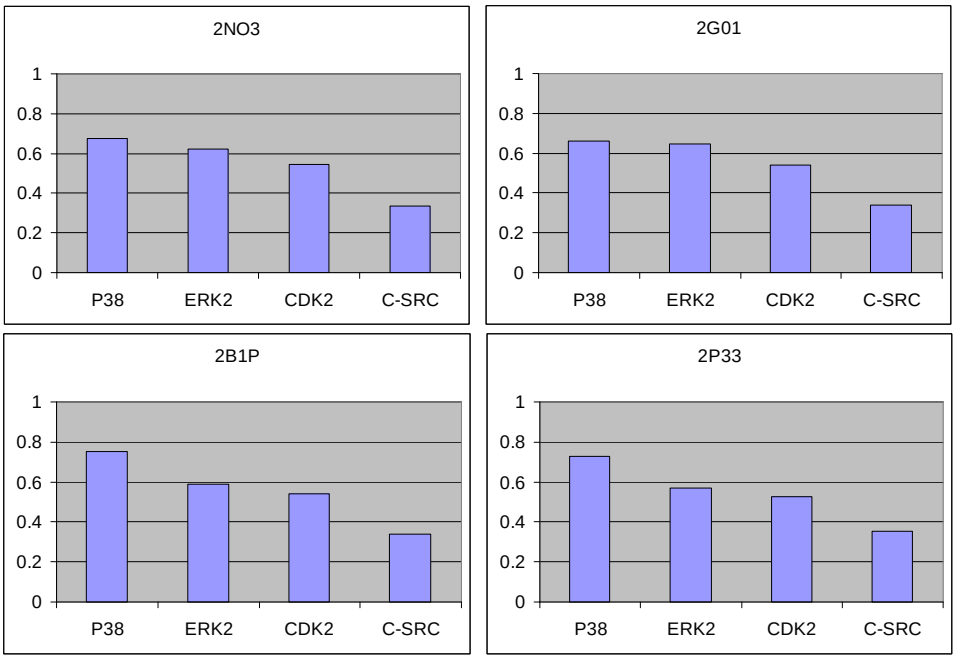


Fig. 6.15 Promedio de porcentajes de cantidad de nodos de subgrafos JNK que se recuperan en cada familia

Se puede observar que si bien se recuperan un porcentaje considerable de los subgrafos de la familia P38 y ERK2, por la alta identidad de secuencias, estas tasas no son suficientes como para identificar estas estructuras con las de P38. Esta decisión es respaldada con los resultados obtenidos en las secciones anteriores donde los subgrafos de P38 son recuperados casi en su totalidad en otras estructuras de la misma familia. Sin embargo no hemos desarrollado una manera cuantitativa que nos permita calcular en forma fehaciente el grado de similitud. En este sentido sería interesante extender este método para poder no solamente obtener los subgrafos isomorfos de una familia sino determinar un umbral inferior de tasas de recuperación de cantidad de nodos de los mismos con que se pueda decidir si una estructura pertenece o no a la familia en cuestión.

6.5 Evaluaciones de performance

Previamente se realizaron evaluaciones de performance del algoritmo AcGM. Estas evaluaciones consistieron en observar el tiempo de cómputo teniendo en cuenta cada variable de los grafos de entrada y parámetros, una por vez. La tabla 6.3 muestra los parámetros que se tuvo en cuenta para realizar estas evaluaciones.

Parametro	Definición	Default
D	cantidad de grafos en la base de datos	10000
| T |	tamaño promedio de grafos	15
L	cantidad de patrones basicos	10
| I |	tamaño promedio de patrones básico	7
N_v	cantidad de etiquetas de nodos	5
N_e	cantidad de etiquetas de ejes	5
P	probabilidad de existencia de ejes	20%
minsup	soporte minimo	10%

Tabla 6.3 Los valores por default utilizados en las evaluaciones de performance del algoritmo en el paper

Las observaciones de crecimiento de tiempo computacional fueron realizadas para las siguientes variables:

- cantidad de grafos en la base de datos
- tamaño promedio de grafos en la base
- cantidad de etiquetas de nodos
- soporte mínimo
- probabilidad de existencia de ejes entre nodos

En cada evaluación el resto de las variables tenían valores por defecto expuesto en la tabla 6.3.

El crecimiento de tiempo de cómputo es razonable para cada uno de estas variables.

Sin embargo, se puede observar que el tamaño promedio de patrones básicos (subgrafos isomorfos frecuentes resultantes) es 10 por defecto y la probabilidad de existencias de ejes es 20%.

Considerando que para nuestro problema estos valores no son suficientes y habiendo observado en reiteradas pruebas que el tiempo de cómputo y el porcentaje de uso de memoria crecen exponencialmente con relación a la cantidad de nodos y ejes en los subgrafos resultantes, se realizaron varias evaluaciones. Teniendo en cuenta que este algoritmo solamente trabaja con la memoria principal disponible en la máquina en la que se este ejecutando, es importante observar cuanta memoria consume en distintas situaciones presentadas.

El hecho de que el algoritmo trabaja solo con la memoria principal simplifica el análisis de la performance ya que el tiempo de ejecución no contiene tiempo de accesos al disco, y el porcentaje de almacenamiento de datos temporales puede ser observado con el uso de memoria principal. A

continuación mostramos un análisis detallado de performance del algoritmo contra diferentes variables del problema en estudio

6.5.1 Evaluaciones con cantidad de nodos en los subgrafos frecuentes

Si bien el trabajo de Inokuchi[37] muestra el tiempo de cómputo contra la probabilidad de existencia de ejes y el tamaño promedio de grafos en la base, el tipo de evaluación propuesto acá es diferente. El tiempo de cómputo muestra un crecimiento lineal con el aumento de tamaño promedio (de 10 a 20 nodos) de grafos en la base. Pero esta evaluación no tiene información sobre la cantidad de nodos en los subgrafos encontrados con 20% de existencia de ejes. Sin embargo, la observación de tiempo de proceso contra la cantidad de nodos en los subgrafos isomorfos es importante ya que el algoritmo va reduciendo el espacio de búsqueda según subgrafos frecuentes que va encontrando en cada nivel. Es decir, mientras menos cantidad de nodos en los subgrafos, el árbol de búsqueda se expande menos implicando menos tiempo de procesamiento.

6.5.1.1 Cantidad de nodos en los subgrafos Vs Tiempo computacional y consumo de memoria

Se construyeron varias series de grafos, cada una con subgrafos isomorfos frecuentes de distintos tamaños, en los cuales los nodos de los subgrafos están unidos solo por n-1 ejes. Es importante que los subgrafos tengan esta cantidad de ejes, ya que en esta evaluación solo se quiere observar cómo la cantidad de nodos en los subgrafos afectan la performance. N-1 ejes es la menor cantidad de ejes con los que se puede construir un grafo conexo de n nodos.

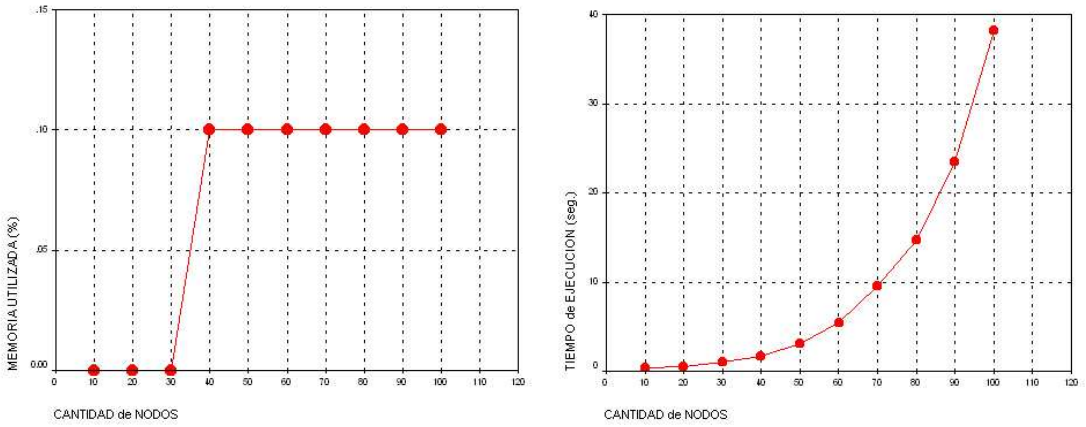


Fig. 6.16 La memoria utilizada en % por cantidad de nodos en los subgrafos encontrados .
El tiempo de ejecución por cantidad de nodos en los subgrafos encontrados [4]

[4] No se muestran gráficos de cantidad de ejes ya que la cantidad de ejes crece linealmente junto a la cantidad de nodos y los gráficos trazados son similares a los de cantidad de nodos.

Se pudo observar que el crecimiento de tiempo computacional es exponencial con la cantidad de nodos (Fig 6.17), pudiendo comprobar que la cantidad de nodos en los subgrafos isomorfos frecuentes tienen un impacto más fuerte en el tiempo de proceso que en el tamaño promedio de la cantidad total de nodos de grafos de la base. Es decir, aunque los grafos en la base tienen tamaños considerablemente grandes, entre 100 y 500 nodos, si los subgrafos isomorfos frecuentes entre ellos son pequeños, entre 10 y 15 nodos como el promedio de grafos utilizados en la evaluación del trabajo de Inokuchi[37], el tiempo de proceso no supera los 20 segundos. Al contrario, si la cantidad de nodos de los subgrafos es grande, no importa cuan pequeño sea el promedio de tamaño de grafos, el proceso toma mas tiempo.

El comportamiento observado en el crecimiento de la memoria utilizada parece algo inusual ya que crece de golpe cuando la cantidad de nodos es 40 y luego deja de crecer. Esto es debido a que la herramienta con la que se midió no puede expresar con precisión necesaria como para mostrar los cambios en el porcentaje de la memoria utilizada para cada caso.

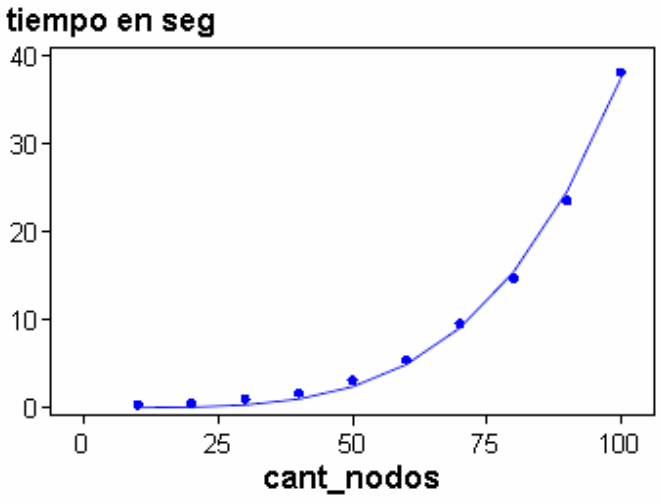


Fig. 6.17 Cantidad de nodos en el subgrafo (con n-1 enlaces) vs tiempo de ejecución.
Los puntos son los valores reales del experimento y la curva trazada es la función ($a \times cant\_nodos^{b}$) encontrada por la regresión

6.5.1.2 Cantidad de ejes en los subgrafos Vs Tiempo computacional y consumo de memoria

Para ver la influencia de cantidad de ejes en la perfomance del algoritmo, se construyeron varias series de datos que contienen distintos tamaños de subgrafos isomorfos frecuentes que son cliques.

6. Resultados

Un clique tiene $\left(n(n-1)/2 \right)$ [5] ejes entre *n* nodos. Estos subgrafos contienen muchos más ejes por nodo que aquellos de la evaluación anterior. La idea es, dada la misma cantidad de nodos en los subgrafos isomorfos frecuentes (y manteniendo las demás variables, salvo cantidad de ejes, con mismos valores), comparar el tiempo computacional y el porcentaje de memoria usada en los dos casos.

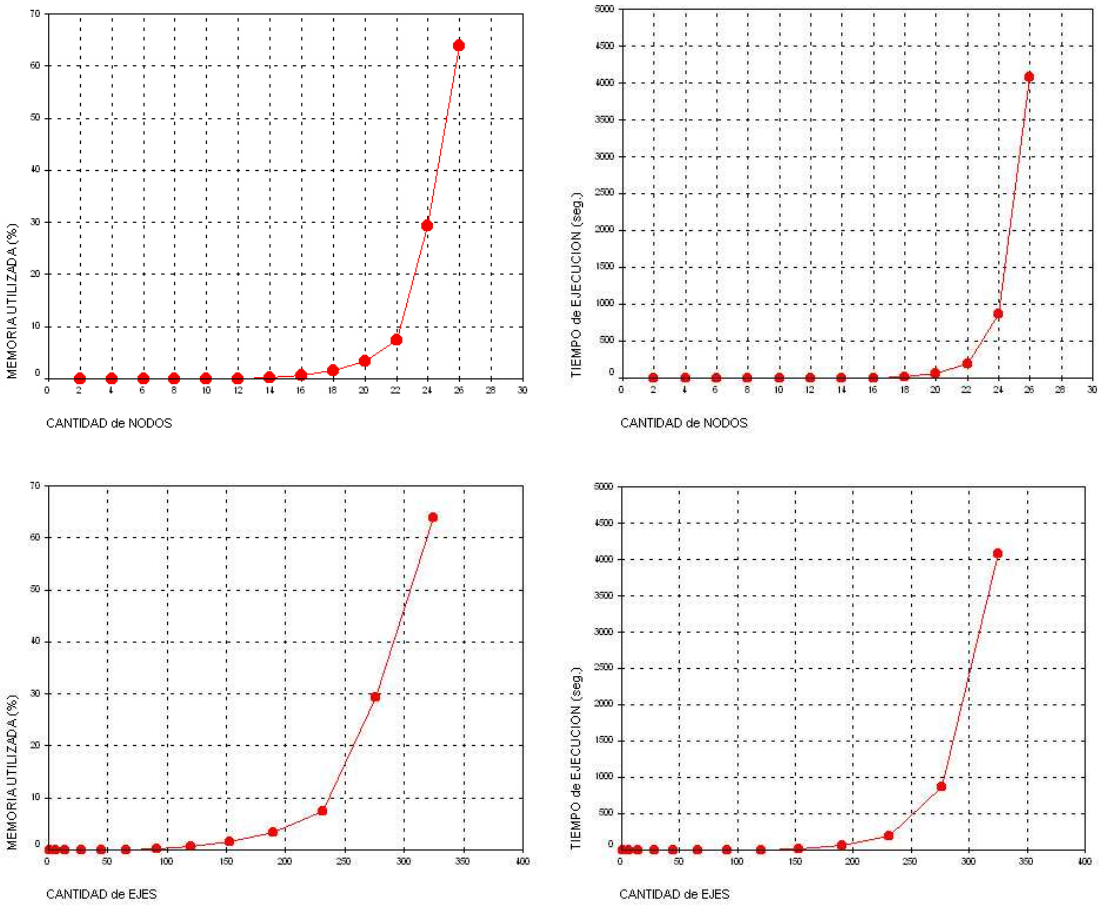


Fig. 6.18 La memoria utilizada en % por cantidad de nodos y cantidad de enlaces.
en los subgrafos encontrados . El tiempo de ejecución por cantidad de nodos y cantidad de enlaces en los subgrafos encontrados

Se puede observar en la figura 6.18 que la memoria utilizada llega a superar los 60% con tan solo 26 nodos. Es notorio el aumento de memoria utilizada de tener un valor menor a 1% cuando el subgrafo es una cadena a cuando es un clique. Este aumento se debe a los subgrafos de tamaños intermedios

[5] $\sum_{i=1}^{n-1} i = \left(n(n-1)/2 \right)$

que se van construyendo a lo largo de la ejecución. Si el único subgrafo isomorfo frecuente que contiene la base de grafos es una cadena, la cantidad de subgrafos candidatos unificables es pequeña, por lo que a lo largo de ejecuciones de niveles, la cantidad de los mismos van reduciéndose rápidamente. En cambio, si el subgrafo es un clique, si la cantidad de subgrafos candidatos en cada nivel(para el nivel n, es decir, los subgrafos candidatos de tamaño *n*, existen $\binom{n}{k}$ subgrafos candidatos), la memoria utilizada aumenta rápidamente ya que no se eliminan ninguno de los subgrafos de los niveles anteriores, y además cada subgrafo va aumentando en tamaño.

Se puede observar también, que el tiempo de ejecución se incrementa con más rapidez que al analizar grafos en cadenas. Este incremento es consecuencia de la cantidad de operaciones de unión que tiene que hacer en cada nivel. Además esta operación se hace más compleja en cada nivel por el tamaño de las submatrices con las que tiene que trabajar.

Esta observación demuestra que si bien la cantidad de nodos en los subgrafos isomorfos frecuentes hace crecer exponencialmente el tiempo de ejecución, la cantidad de ejes entre estos nodos tienen aún un efecto mayor.

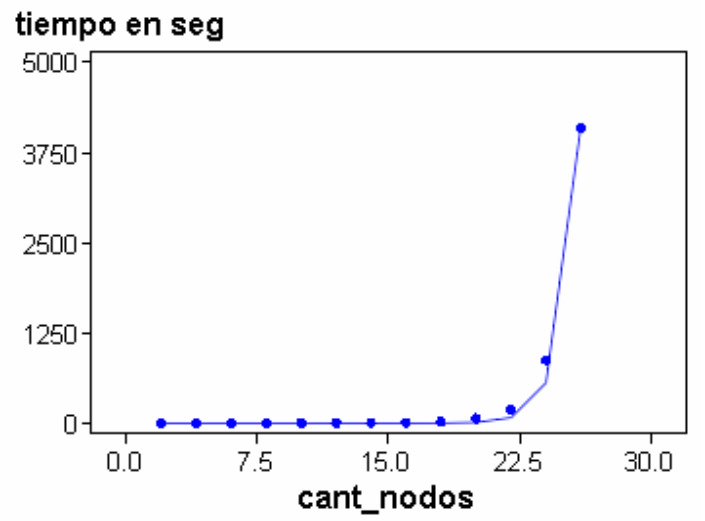


Fig. 6.19 Cantidad de nodos en el subgrafo (clique) vs tiempo de ejecución.
Los puntos son los valores reales del experimento y la curva trazada es
la función($e^{cant\_nodos - a}$ con a=17.6735) encontrada por la regresión

El tiempo de proceso crece rápidamente con la cantidad de nodos y de ejes en los subgrafos.

Este resultado es lo esperable para cualquier tipo de algoritmo que trata de resolver un problema considerado como NP-completo.

Esto nos muestra que la reducción de espacios de búsqueda en el árbol trae mejoras significativas solamente para aquellos problemas en donde los subgrafos no tienen más de cierta cantidad de nodos y que la cantidad de ejes entre ellos no sean considerablemente grandes.

6.5.1.3 Uso de memoria principal

Como se pudo observar en la evaluación de performance del algoritmo, este tiene una debilidad que si el tamaño de los subgrafos que pueden ser encontrados en la base de grafos es mayor a un cierto valor, el consumo de recursos crece en forma exponencial.

Uno de los factores que contribuye a este crecimiento de uso de recursos es el hecho de conservar todos los subgrafos isomorfos frecuentes encontrados hasta el último momento. Este hecho puede ser observado fácilmente en el archivo de salida del algoritmo:

size=1
VertexLabels = TRP,
code =
count = 20

size=2
VertexLabels = TRP,GLY,
code = C,
count = 20

size=3
VertexLabels = TRP,GLY,GAN,
code = C,0,C,
count = 20

size=4
VertexLabels = TRP,GLY,GAN,PRO,
code = C,0,C,C,0,0,
count = 20

size=5
VertexLabels = TRP,GLY,GAN,PRO,GAP,
code = C,0,C,C,0,0,0,C,0,0,
count = 20

size=6
VertexLabels = **TRP,GLY,GAN,PRO,GAP,GAP,**
code = C,0,C,C,0,0,0,C,0,0,0,0,0,C,
count = 20

size=4
VertexLabels = TRP,GLY,GAN,GAP,
code = C,0,C,0,C,0,
count = 20

size=5
VertexLabels = **TRP,GLY,GAN,GAP,GAP,**
code = C,0,C,0,C,0,0,0,0,C,
count = 20

size=3
VertexLabels = TRP,GLY,PRO,
code = C,C,0,
count = 20

size=4
VertexLabels = TRP,GLY,PRO,GAP,
code = C,C,0,0,C,0,
count = 20

size=5
VertexLabels **= TRP,GLY,PRO,GAP,GAP**,
code = C,C,0,0,C,0,0,0,0,C,
count = 20

Fig. 6.20 Subgrafos maximales en el archivo de salida

Se puede observar que la cantidad de submatrices que quedan guardadas en la memoria hasta el final de la ejecución es considerablemente mayor que la cantidad de matrices de subgrafos maximales frecuentes (los resaltados con negrita). El uso de memoria puede ser optimizado si se van eliminando las submatrices de niveles anteriores. Como pudimos observar en el ejemplo del algoritmo en la sección 4.2.3.3 esta modificación es factible. Una vez completado la fase de join de las matrices de tamaño k-1 y obtenido todas las matrices de tamaño k que representa subgrafos frecuentes de tamaño k, las matrices de tamaño k-1 dejan de ser consultadas. Las matrices de tamaño k+1 son construídas a partir del aquellas de tamaño k.

La tabla 6.4 muestra cómo en cada nivel la cantidad de subgrafos maximales es mucho menor a la cantidad de todos los subgrafos. Se puede observar que al ir eliminando en cada paso los subgrafos de dos niveles inferiores al nivel actual que no son maximales en cada paso se podría ahorrar alrededor de 70% de uso de memoria.

Cantidad de nodos	Cantidad de subgrafos	Cantidad de celdas para todos los subg	Cantidad de celdas acumuladas para todos los subg	Cantidad de subgrafos maximales	Cantidad de celdas para subg maximales	Cantidad de celdas acumuladas para subg maximales	Porcentaje
1	14	14	14	0	0	0	0.00%
2	68	204	218	2	6	6	2.75%
3	76	456	674	8	48	54	8.01%
4	85	850	1524	19	190	244	16.01%
5	75	1125	2649	25	375	619	23.37%
6	50	1050	3699	19	399	1018	27.52%
7	28	784	4483	12	336	1354	30.20%
8	21	756	5239	4	144	1498	28.59%
9	17	765	6004	3	135	1633	27.20%
10	11	605	6609	2	110	1743	26.37%
11	19	1254	7863	9	594	2337	29.72%
12	6	468	8331	0	0	2337	28.05%
13	4	364	8695	3	273	2610	30.02%
14	3	315	9010	1	105	2715	30.13%
15	3	360	9370	2	240	2955	31.54%
16	1	136	9506	0	0	2955	31.09%
17	1	153	9659	1	153	3108	32.18%
Total		9659			3108		

Tabla 6.4 Comparación de cantidad de subgrafos y cantidad de celdas utilizadas para almacenarlos. Se muestran las cantidades de todos los subgrafos que son encontrados en todos los niveles y las de los subgrafos maximales. En la columna Porcentaje se muestran los porcentajes de cantidad de celdas necesarias para guardar los subgrafos maximales en cada nivel comparado a aquella de todos los subgrafos.

7. Conclusión y futuros trabajos

Analizar las complejas estructuras de las proteínas no es una tarea sencilla y hoy en día la comunidad científica sigue dedicando un esfuerzo considerable a resolver este tema siendo unas de las áreas de desarrollo en biología computacional. En esta tesis se ha desarrollado con éxito una solución a este problema utilizando técnicas de grafos. Considerando a los aminoácidos como nodos y agregando ejes en función de la distancia entre nodos, la aplicación desarrollada permite estudiar gran cantidad de proteínas y encontrar detalles estructurales que se encuentran conservados entre ellas. El reducir el problema a la búsqueda de subgrafos nos permitió utilizar un algoritmo para la búsqueda de subgrafos isomorfos, AcGM, que fue evaluado exhaustivamente.

El algoritmo mostró ser efectivo para encontrar detalles estructurales. Se realizó una aplicación puntual del algoritmo para identificar una proteína en función de grupos predeterminados. Esta aplicación resultó ser exitosa ya que en la mayoría de los casos se pudo identificar claramente la proteína incógnita. Se puede decir entonces que se identificaron detalles estructurales que caracterizan a diferentes familias de proteínas. Este hecho muestra la utilidad del método desarrollado y la potencialidad que tiene para estudios futuros. Entre ellos se pretende a futuro realizar un estudio más exhaustivo extendiéndolo a toda la familia de proteínas kinasas. Asimismo se pretende desarrollar herramientas mediante minería de datos y trabajos estadísticos que permitan identificar un subconjunto de los subgrafos obtenidos a través de AcGM que maximice la capacidad de discriminación de pertenencia de una proteína. Es de esperar que aquellos subgrafos que permitan identificar claramente a una familia y no se encuentre en otras tendrá un rol importante en la función particular de esa enzima.

Se hizo un estudio de performance que muestra que hay espacios para mejorar el algoritmo. Pudimos mostrar que este algoritmo encuentra todos los subgrafos isomorfos en tiempo razonable si éstos cumplen algunas condiciones como por ejemplo que la cantidad de nodos en los subgrafos sean menores a cierto número. Asimismo reduce considerablemente el espacio de búsqueda en cada nivel de derivación lo cual ayuda en gran manera a acortar el tiempo de ejecución y el uso de recursos. Como se expuso en las secciones anteriores el método interactúa solamente con la memoria principal para almacenar datos de grafos y subgrafos lo cual termina siendo un límite para cuando los tamaños de los mismos crecen. El uso de disco rígido para el almacenamiento puede alargar el tiempo de ejecución pero permitiría la búsqueda de subgrafos más grandes en una base de grafos más grandes. En este sentido proponemos desarrollar en trabajos futuros una opción para poder interactuar con el disco rígido si es necesario.

El método desarrollado podría tener utilidad para los algoritmos que predicen estructuras de proteínas. Es de esperar que los detalles estructurales característicos que definen por ejemplo una función particular debería estar presentes en la estructura predicha de una proteína que se sabe tiene dicha función. También sería interesante extender este método para obtener una herramienta que pueda realizar un clustering de estructuras proteicas. De esta manera, sería posible dado un conjunto grande estructuras clasificarlas de acuerdo a los subgrafos isomorfos que se obtienen entre ellos. Por otro lado, es sabido que la flexibilidad de proteínas es indispensable para su funcionamiento.

7. Conclusión y futuros trabajos

En los últimos años la dinámica molecular, técnica que permite obtener "películas" del movimiento de las proteínas, se ha utilizado ampliamente para estudios de flexibilidad/función. Sin embargo es complicado obtener información relevante de estas "películas", dado la compleja red de interacciones que esta presenta en una proteína. El método presentado en esta tesis sería de gran utilidad para encontrar interacciones complejas conservadas a lo largo del tiempo, si cada foto de la película es un grafo y se buscan subgrafos conservados.

Finalmente es importante remarcar que la riqueza del método se puede extender fácilmente agregando propiedades que permitan construir diferentes grafos o grafos con distintos enlaces, como ser puentes hidrógenos, puentes salinos o interacciones hidrofóbicas.

REFERENCES

1. Lahoz-Beltrá R. Bioinformática Simulación, vida artificial e inteligencia artificial. Ediciones Díaz de Santos,S.A.Madrid, 2004, Introducción

2. Rouvray DH, Balaban AT. Chemical applications of graph theory. *Applications of Graph Theory*, Wilson RJ, Beineke LW (eds.). Academic Press: New York, 1979; 177–221. Copyright c 2004 John Wiley & Sons, Ltd. *Softw. Pract. Exper*. 2004; **34**:591–607 606 E. B. KRISSINEL AND K. HENRICK

3. Mitchell EM, Artymiuk PJ, Rice DW, Willett P. Use of techniques derived from graph theory to compare secondary structure motifs in proteins. *Journal of Molecular Biology* 1990; **212**:151–166.

4. Grindley HM, Artymiuk PJ, Rice DW,Willett P. Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm. *Journal of Molecular Biology* 1993; **229**:707–721.

5. Kuhl F, Crippen G, Friesen D. A combinatorial algorithm for calculating ligand binding. *Journal of Computational Chemistry* 1984; **5**:24–34.

6. Thorner DA, Willett P, Wright PM, Taylor R. Similarity searching in files of three-dimensional chemical structures: Representation and searching of molecular electrostatic potentials using field-graphs. *Journal of Computer-Aided Molecular Design* 1997; **11**:163–174.

7. Bruno IJ, Kemp NM, Artymiuk PJ,Willett P. Representation and searching of carbohydrate structures using graph-theoretic techniques. *Carbohydrate Research* 1997; **304**:61–67.

8. Gardiner EJ, Willett P, Artymiuk PJ. Graph-theoretic techniques for macromolecular docking. *Journal of Chemical Information and Computer Science* 2000; **40**:273–279.

9. McGregor J,Willett P. Use of a maximal common subgraph algorithm in the automatic identification of the ostensible bond changes occurring in chemical reactions. *Journal of Chemical Information and Computer Science* 1981; **21**:137–140.

10. Arita M. Graph modeling of metabolism. *Journal of the Japanese Society of Artificial Intelligence 2000; 15:703–710.*

11. Chen L, Robien W. Application of the maximal common substructure algorithm to automatic interpretation of 13C-NMR spectra. *Journal of Chemical Information and Computer Science* 1994; **34**:934–941.

12. Cone M, Venkataraghavan R, McLafferty F. Computer-aided interpretation of mass spectra. 20. Molecular structure comparison program for the identification of maximal common substructures. *Journal of the American Chemistry Society* 1977; **99**:7668–7671.

13. Gifford E, Johnson M, Smith D, Tsai C. Structure–reactivity maps as a tool for visualizing xenobiotic structure–reactivity relationships. *Network Science* 1996; **2**:1–33.

14. Raymond J, Willett P. Effectiveness of graph-based and fingerprint-based similarity measures for virtual screening of 2D chemical structure databases. *Journal of Computer-Aided Molecular Design* 2002; **16**:59–71.

15. Edgar SJ, Holliday JD, Willett P. Effectiveness of retrieval in similarity searches of chemical databases: A review of performance measures. *Journal of Molecular Graphics and Modelling* 2000; **18**:343–357.

16. Horaud R, Skordas T. Stereo correspondence through feature grouping and maximal cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1989; **11**:1168–1180.

17. Pelillo M, Siddiqi K, Zucker SW. Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1999; **21**:1105–1120.

18. Yang B, Snyder W, Bilbro G. Matching oversegmented 3D images to models using association graphs. *Image and Vision Computing* 1989; **7**(2):135–143.

19. Pla F, Merchant J. Matching feature points in image sequences through a region-based method. *Computer Vision Image Understanding* 1997; **66**:271–285.

20. Shearer K, Bunke H, Venkatesh S. Video indexing and similarity retrieval by largest common subgraph detection using decision trees. *Pattern Recognition* 2001; **34**:1075–1091.

21. Raymond J, Willett P. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design* 2002; **16**:521–533.

22. Bunke H. Recent advances in structural pattern recognition with applications to visual form analysis. *Visual Form 2001* (*Lecture Notes in Computer Science*, vol. 2059), Arcelli C, Cordella L, Sanniti di Baja G (eds.). Springer: Berlin, 2001; 11–23.

23. Levi G. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo* 1972; **9**:341–354.

24. Bron C, Kerbosch J. Algorithm 457—finding all cliques of an undirected graph. *Communications of the ACM* 1973; **16**:575–577.

25. McGregor J. Backtrack search algorithms and the maximal common subgraph problem. *Software—Practice and Experience* 1982; **12**:23–34.

26. Rohit Singh, Mitul Saha. Identifying structural motifs in proteins. *Accelrys Inc, San Diego Department of Mechanical Engineering, Stanford University*

27. J R Ullmann M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the theory of NP-completeness. Freeman 1979. An Algorithm for Subgraph Isomorphism *Journal of the ACM (JACM)*

28. Dehaspe, L., Toivonen, H. and King, R.D. 1998. Finding frequent substructures in chemical compounds. *In Proceedings of the Fourth International Conference onKnowledge Discovery and Data Mining (KDD-98),* pp.30-36.]

29. Cook, D.J., & Holder, L.B (1994) Substructure Discovery Using Minimun Description Length and Background Knowledge. *Journal of Artificial intelligence REsearch* Vol. 1 (pp. 231-255)

30. Inokuchi, I., Washio, T., & Motoda, H.(2000). An Aprior-based Algorithm for mining Frequent Subestructures form Graph Data. *Proc. Of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases.*

31. Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. *InProc. of the 20th VLDB Conference, pp.487-499*

32.Inokuchi, A.,Washio, T. and Motoda, H. 1999. Derivation of the topology structurefrom massive graph data. *Discovery Science: Proceedings of the Second International Conference, DS'99, pp.330-332.*

33.Fortin, S. 1996. The graph isomorphism problem. *Technical Report 96-20, University of Alberta, Edomonton, Alberta, Canada*

34. Tetsuhiro Miyahara, Takayoshi Shoudai , Tomoyuki Uchida, Tetsuji Kuboyama , Kenichi Takahashi and Hiroaki Ueda. 1999. Discovering New Knowledge from Graph Data Using Inductive Logic Programming. *Inductive Logic Programming pp. 222-233*

35. Ulrich Rückert, Stefan Kramer Frequent free tree discovery in graph data. 2004. *Symposium on Applied Computing: Proceedings of the 2004 ACM symposium on Applied computing pp.564-570*

36. A. Inokuchi, T. Washio, and H. Motoda, An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data *Proc. Fourth European Conf. Principles and* Practice of Knowledge Discovery in Databases (PKDD '00), *pp. 13-23, Sept. 2000.*

37. A. Inokuchi, T. Washio, K. Nishimura, and H. Motoda, A Fast Algorithm for Mining Frequent Connected Subgraphs *Technical Report RT0448, IBM Research, Tokyo Research Laboratory, 2002.*

38. Strogatz, S. H. Exploring Complex Networks *Nature 2001, 410, 268-76.*

39. Amitai, G.; Shemesh, A.; Sitbon, E.; Shklar, M.; Netanely, D.; Venger, I.; Pietrokovski, S. Network analysis of protein structures identifies functional residues *J Mol Biol 2004, 344, 1135-46*

40. Carl Branden; John Tooze. 1999, Introduction to Protein Structure 2nd ed. Pp. 1-30

41. Bruce Alberts; Denis Bray; Julian Lewis 1994, Biología Molecular de La Célula. Pp. 43-145

42. A G Turjanski, J P Vaqué and J S Gutkind . Oncogene (2007) 26, 3240–3253. doi:10.1038/sj.onc.1210415 MAP kinases and the control of nuclear events

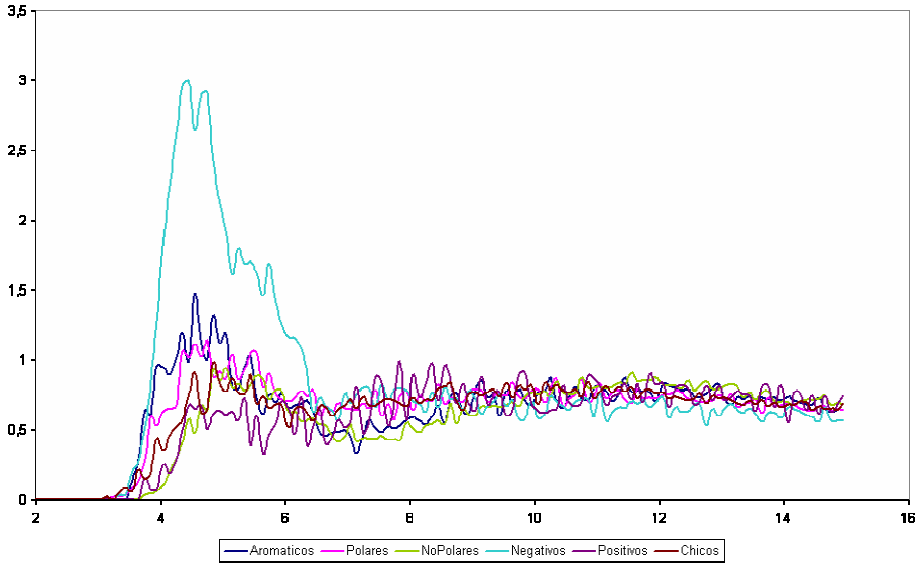
43. H. Matsuda; F. Taniguchi, A. Hashimoto, 1997. An Approach to Detection of Protein Structural Motifs using an Encoding Scheme of Backbone Conformation. *Pac Symp Biocomput 1997 280-91*

44. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *In Proc. of the ACM SIGMOD Conference on Management of Data, Washington, D.C., May 1993.*

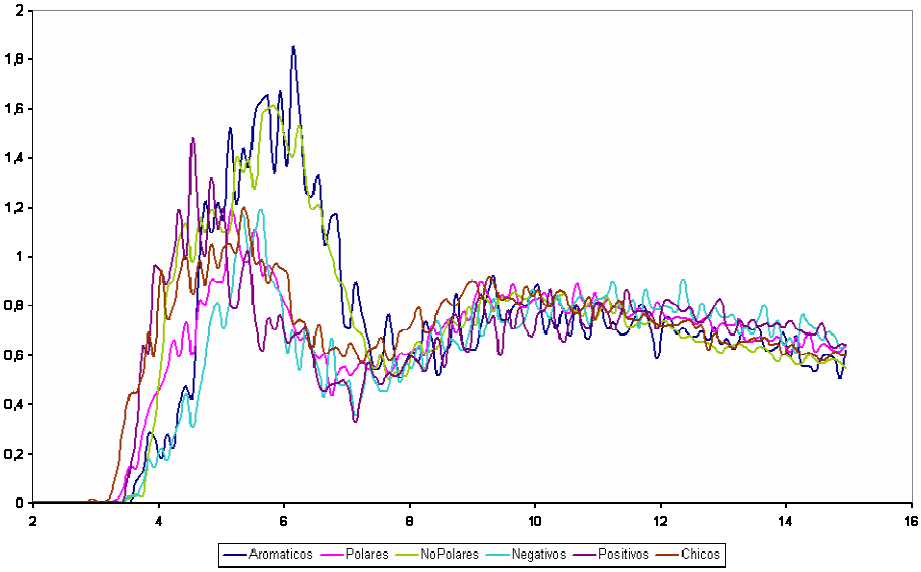
Apéndice

Gráficos de distancias entre distintos tipos de aminoácidos.

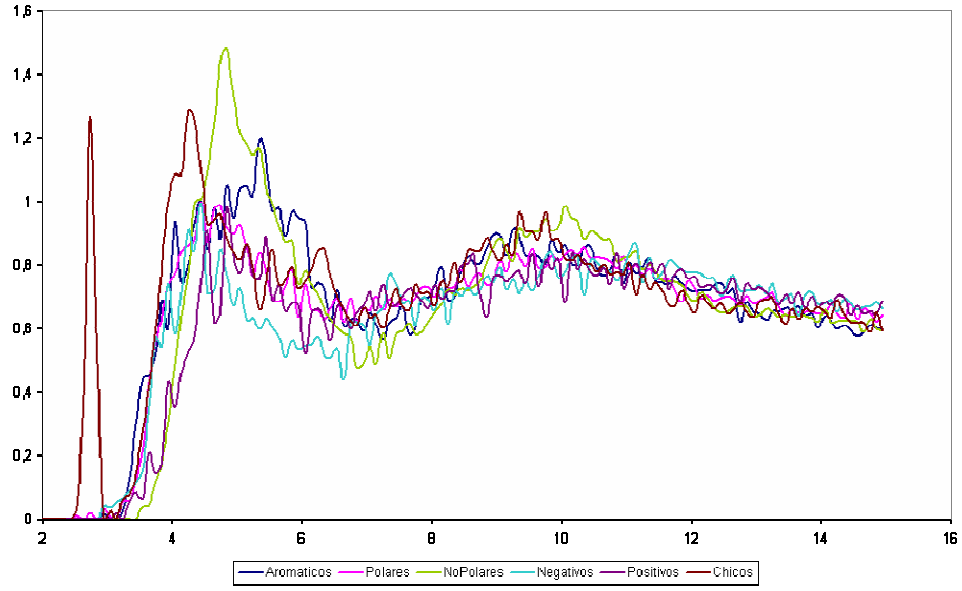
Gráficos de distancias de aminoácidos positivos con otros



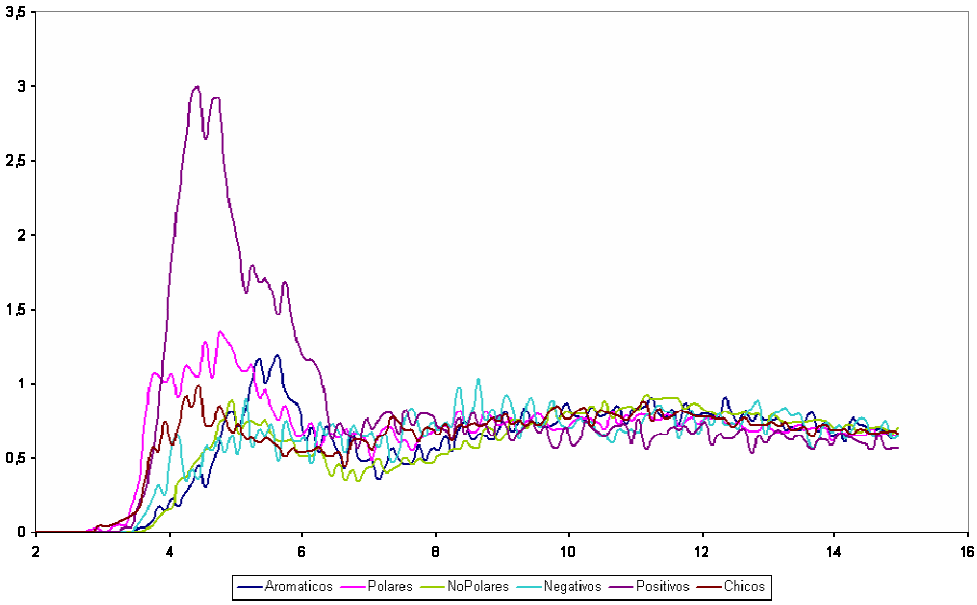
Gráficos de distancias de aminoácidos aromáticos con otros



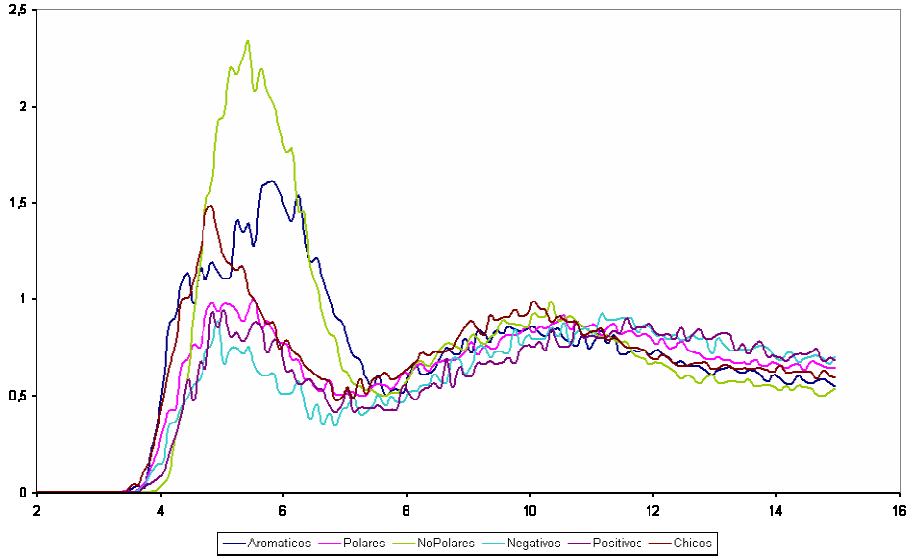
Gráficos de distancias de aminoácidos chicos con otros



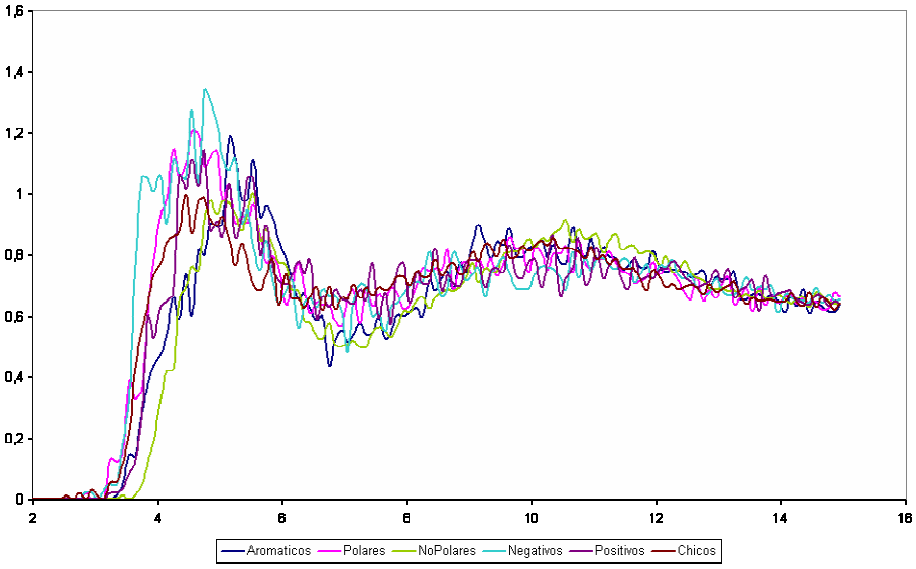
Gráficos de distancias de aminoácidos negativos con otros



Gráficos de distancias de aminoácidos no polares con otros



Gráficos de distancias de aminoácidos polares con otros



Resumen de los resultados del experimento de la sección 6

group/clase	subgrafo	cant_nodos en el subgrafo	cant_nodos en el subgrafo recuperado	% de nodos recuperados
g1/c2	52	11	9	0.818182
	36	11	9	0.818181818
	67	10	9	0.9
	43	10	9	0.9
	102	10	9	0.9
	86	10	9	0.9
	51	10	8	0.8
	55	11	9	0.818181818
	35	10	8	0.8
	39	11	9	0.818181818
	99	10	9	0.9
	83	10	9	0.9
Promedio		10.33333333	8.833333333	0.856060606
g1/c3	67	8	4	0.5
	64	9	4	0.444444444
	58	10	4	0.4
	39	12	5	0.416666667
	38	15	8	0.533333333
	32	16	8	0.5
	26	11	11	1
	24	19	8	0.421052632
	22	22	11	0.5
	17	12	12	1
	14	14	14	1
	12	18	18	1
	9	19	19	1
	8	20	20	1
	3	9	9	1
Promedio		14.26666667	10.33333333	0.714366472
g2	17	11	11	1
	13	8	7	0.875
	12	10	8	0.8
	9	8	8	1
	29	9	5	0.555555556
	28	9	6	0.666666667
	22	8	5	0.625
	21	8	5	0.625
Promedio		8.875	6.875	0.768402778
g3	3	15	15	1
	20	14	12	0.857142857
	19	9	6	0.666666667
	17	19	15	0.789473684
	16	22	18	0.818181818
	13	22	15	0.681818182
	10	14	12	0.857142857
	9	14	12	0.857142857
	8	14	13	0.928571429
	7	14	14	1
	6	15	13	0.866666667
	5	15	13	0.866666667
	4	15	14	0.933333333
Promedio		15.53846154	13.23076923	0.85560054