UNIVERSIDAD NACIONAL DE COLOMBIA

Búsqueda de conformaciones estables de interacción proteína-proteína utilizando métodos de inteligencia computacional

Juan Guillermo Carvajal Patiño

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial

Bogotá D.C., Colombia

2013

Búsqueda de conformaciones estables de interacción proteína-proteína utilizando métodos de inteligencia computacional

Juan Guillermo Carvajal Patiño

Tesis de investigación presentada como requisito parcial para optar al título de:

Magister en Ingeniería de Sistemas y Computación

Director:

Luis Fernando Niño V., Ph.D.

Línea de Investigación:

Bioinformática

Grupo de Investigación:

Laboratorio de Investigación en Sistemas Inteligentes LISI

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial

Bogotá, D.C., Colombia

2013

A mis padres y hermanos.

“Convierte en peldaños las piedras con que tropiezas, porque lo importante no es dónde nos encontramos sino en qué dirección vamos”

Agradecimientos

Ante todo, agradezco a Dios. Al profesor Luis Fernando Niño por su apoyo, comprensión y conceptos que me orientaron durante el desarrollo de esta tesis. A mis padres y hermanos que a pesar de las adversidades siempre me han alentado a alcanzar las metas propuestas y a enfrentar los desafíos venideros. A Liliana Olarte por su apoyo y acompañamiento, y a David Becerra por introducirme en esta línea de investigación y colaborarme en todo lo que pudo. Finalmente, agradezco al resto de mi familia y a mis amigos por estar ahí.

Resumen

El desarrollo de esta tesis se enfoca en el problema de la interacción proteína-proteína, el cual es un problema biológico vigente. El problema consiste en encontrar la estructura tridimensional de un complejo proteína-proteína, dadas las coordenadas atómicas de las proteínas que lo conforman. En este trabajo se presenta un modelo para la predicción de la interacción proteína-proteína considerando las proteínas como cuerpos rígidos (etapa 1) y otro modelo para la inclusión de flexibilidad en el ligando (etapa 2). En la etapa 1 se utiliza un algoritmo genético y en la etapa 2 un algoritmo de búsqueda local. Se realizaron experimentos sobre un conjunto de pruebas de cuatro complejos proteína-proteína (2SNI, ICGI, 1CLV y 1PPE) obteniendo resultados que pueden constituirse en un punto de partida para futuros desarrollos e investigaciones. Con el objetivo de acelerar el proceso de cómputo se propone una implementación paralela, tanto para la interacción proteína-proteína de cuerpo rígido como para la etapa de inclusión de flexibilidad.

Palabras clave: proteínas, interacción proteína-proteína, algoritmos de búsqueda, complejos proteicos, inteligencia computacional, computación paralela.

Abstract

The development of this thesis focuses on the current biological problem of protein-protein interaction. The problem is finding the three-dimensional structure of a protein-protein complex given the atomic coordinates of the proteins composing it. In this work a model for the prediction of protein-protein interaction considering proteins as rigid bodies (stage 1) and a model for the inclusion of ligand flexibility are proposed (stage 2). In stage 1 a genetic algorithm is used; in stage 2, a local search algorithm is run. Experiments on a group of tests of four protein-protein complexes (2SNI, ICGI, 1CLV and 1PPE) were conducted, achieving results which may become a starting point for future development and research. Aiming to accelerate the computing process, a parallel implementation for rigid body protein-protein interaction and for the ligand flexibility inclusion stage is proposed.

Keywords: proteins, protein-protein interaction, search algorithms, protein complex, computational intelligence, parallel computing.

Contenido

Lista de figuras

Lista de tablas

Lista de abreviaturas

Abreviatura	Término
ADN	Ácido desoxirribonucleico
ARN	Ácido ribonucleico
PDB	Protein Data Bank
RMN	Resonancia Magnética Nuclear
AG	Algoritmo Genético
SISD	Single Instruction Single Data
SIMD	Single Instruction Multiple Data
MISD	Multiple Instruction Single Data
MIMD	Multiple Instructions Multiple Data
MM	Monoprocessor Machines
PM	Parallel Machines
LC	Local Clusters
DC	Distributed Clusters
SVM	Support Vector Machines
CAPRI	Critical Assessment of PRedicted Interactions
CASP	Critical Assessment of Techniques for Protein Structure Prediction
JVM	Java Virtual Machine
RMSD	Root Mean Square Deviation
KVM	Kernel-based Virtual Machine
VMD	Visual Molecular Dynamics
HTC	High-Throughput Computing

Introducción

Los avances científicos y tecnológicos en áreas como la genética y la biología han dado como resultado la obtención acelerada de un gran volumen de datos biológicos de diverso tipo. La necesidad de analizar y gestionar esos datos biológicos de forma eficiente, ha permitido que la bioinformática evolucione como disciplina científica en las últimas décadas [1].

La bioinformática y la relación de la ingeniería de sistemas o de las ciencias de la computación con problemas biológicos va más allá de la obtención, organización, procesamiento y análisis de los datos generados a partir de la experimentación en dichos problemas; también tiene que ver con el entendimiento de los procesos que se llevan a cabo biológicamente y su abstracción para el diseño de modelos y algoritmos orientados a su comprensión.

Existen problemas biológicos que aún no tienen una solución definitiva. En la era de la genómica, gracias a las diferentes técnicas de secuenciación del ácido desoxirribonucleico (ADN), se ha conocido el genoma de diferentes organismos, tal como el genoma humano. Teniendo como base este conocimiento, en la era post-genómica, la investigación se ha enfocado a la identificación y caracterización de las proteínas que son codificadas a partir de los genomas, siendo de particular interés el estudio de la estructura y de la función de las mismas [2].

De acuerdo con lo anterior, se presentan dos problemas biológicos fundamentales: uno es el relacionado con el plegamiento de las proteínas y el otro es el de la interacción entre las mismas. El conocimiento y el estudio de la estructura tridimensional de las proteínas y de los complejos proteicos formados por la interacción entre éstas, es de gran importancia teniendo en cuenta que miles de proteínas presentes en el cuerpo humano y su interacción realizan una gran cantidad de funciones estructurales, cinéticas, catalíticas y de señalización [3].

Aunque existen métodos experimentales que buscan dilucidar la estructura y la función de las proteínas, estos pueden ser costosos en términos de tiempo y recursos. Como alternativa, desde la bioinformática, se han desarrollado modelos que permiten analizar y dar hipótesis sobre estos procesos biológicos, teniendo puntos de partida para los métodos experimentales.

La tasa de determinación de la estructura de las proteínas ha aumentado rápidamente en años recientes. Hoy en día existen más de 70.000 entradas de estructuras de proteínas en el *Protein Data Bank (PDB);* no obstante, la determinación de la estructura de complejos proteicos continúa siendo un problema difícil, reflejándose esto en un menor porcentaje de las estructuras de dichos complejos en el PDB. Por lo tanto, los algoritmos enfocados a predecir las interacciones proteína-proteína han ganado gran relevancia en los últimos tiempos [4].

En un trabajo interdisciplinario, desde las ciencias de la computación, con la aplicación de métodos de inteligencia computacional como las técnicas de aprendizaje computacional y los algoritmos de búsqueda, se pueden establecer modelos y enfoques para solucionar diferentes problemas biológicos.

El problema de predecir cómo van a interactuar dos proteínas, que constituye el tema de esta tesis, consiste en calcular la estructura tridimensional del complejo proteico que se forma a partir de las coordenadas atómicas de las dos proteínas que lo conforman [5]. El objetivo general de la presente investigación es proponer y desarrollar un modelo para la predicción de la interacción proteína-proteína utilizando métodos de inteligencia computacional.

Para tal fin, el problema se resuelve en dos etapas: en la primera las proteínas se consideran como cuerpos rígidos, y en la segunda se realiza un refinamiento de los complejos encontrados en la primera etapa, incluyendo cierto grado de flexibilidad en el ligando. Las dos proteínas que forman un complejo proteico se conocen como ligando y receptor. El ligando, generalmente es la proteína de menor tamaño, y es la que se une o interactúa con el receptor.

Para lograr el objetivo general, tanto en la primera como en la segunda etapa, se define el sistema de representación computacional de las proteínas, el método de inteligencia

computacional a ser utilizado en la búsqueda de soluciones y la función de puntuación que guía el método de inteligencia computacional durante el proceso de búsqueda.

A continuación se presenta la estructura de este documento:

En el capítulo uno se describen los fundamentos biológicos y computacionales básicos que permiten comprender el desarrollo de la investigación, así como la definición del problema. En lo relacionado con los fundamentos biológicos se presentan los conceptos de los aminoácidos, la estructura primaria, secundaria, terciaria y cuaternaria de las proteínas, y también se describen las fuerzas que estabilizan la estructura tridimensional de las mismas. En lo referente a los conceptos computacionales, se presentan los fundamentos de algoritmos de búsqueda y de computación paralela y distribuida.

En el capítulo dos se presenta la revisión de algunos de los métodos computacionales que han sido propuestos en la literatura para solucionar el problema de la interacción proteína-proteína. Se describen diferentes métodos de búsqueda y funciones de puntuación que han sido utilizadas para evaluar los complejos proteicos encontrados, también métodos para predecir potenciales sitios de unión en las proteínas y, finalmente, la forma en la que se evalúan y comparan las soluciones obtenidas.

En el capítulo tres se presenta el modelo propuesto para la predicción de la interacción proteína-proteína utilizando métodos de inteligencia computacional. Específicamente se describen los sistemas de representación de las proteínas, utilizados en la primera y en la segunda etapa; el algoritmo genético (AG) utilizado como método de búsqueda en la etapa en la que las proteínas se consideran como cuerpos rígidos, y el algoritmo de búsqueda local empleado en la etapa de inclusión de flexibilidad. Para el cálculo de la función de puntuación se presenta el paquete de software Tinker, que permite evaluar la energía de las proteínas.

En el capítulo cuatro se describe la implementación paralela de los algoritmos genético y de búsqueda local, utilizados en la primera y en la segunda etapa del modelo propuesto. En lo relacionado con el algoritmo genético, se exponen los conceptos de Java utilizados para paralelizarlo, y en lo que respecta al algoritmo de búsqueda local, se presenta el sistema de computación distribuida Condor y su aplicación al modelo propuesto.

En el capítulo cinco se presentan los resultados obtenidos con la aplicación del modelo propuesto, considerando las proteínas como cuerpos rígidos y también incluyendo flexibilidad en el ligando. Así mismo, se comparan los resultados obtenidos con el modelo propuesto con los resultantes de aplicar métodos existentes. Se concluye este capítulo con el análisis de los resultados.

Finalmente, en el capítulo seis se presentan las conclusiones, recomendaciones y trabajo futuro resultantes del desarrollo de esta tesis.

1. Fundamentos y definición del problema

En este capítulo se presentan los conceptos básicos utilizados en el desarrollo de esta tesis. En la primera parte se describen los fundamentos biológicos necesarios para entender en qué consiste el problema de la interacción proteína-proteína, y en la segunda, se presentan los conceptos computacionales necesarios para comprender el modelo propuesto y su implementación. Finalmente, en la última sección de este capítulo se presenta la definición del problema.

1.1 Fundamentos biológicos

1.1.1 Aminoácidos y sus propiedades

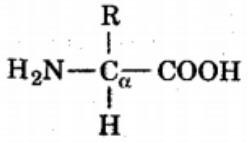
Las proteínas junto a los ácidos nucleicos y a los polisacáridos pertenecen a una de las clases de macromoléculas que contienen los seres vivos [6][7]. Las proteínas y los ácidos nucleicos son moléculas de particular importancia, ya que son moléculas informativas, es decir, que en su estructura química almacenan las instrucciones de lo que ocurre en la célula [8].

La mayor parte de la información genética (instrucciones almacenadas en la estructura del ADN) contiene las instrucciones necesarias para el ensamblaje de todas las proteínas de la célula [8]. La interrelación entre el ADN, el Ácido ribonucleico (ARN) y las proteínas constituye el “dogma central de la biología molecular” que establece que la información genética es transmitida del ADN al ARN mensajero (mARN) a través de la transcripción, para luego, mediante la traducción producir las proteínas. Es decir, existe una relación entre la secuencia de ADN de un gen y la secuencia de aminoácidos de una proteína codificada por un gen [7].

Las proteínas, los ácidos nucleicos y los polisacáridos son polímeros (del griego polys que significa mucho y meros que significa parte). Un polímero está compuesto por unidades más pequeñas llamadas monómeros [8]. En las proteínas, los monómeros se llaman aminoácidos y en los ácidos nucleicos se llaman nucleótidos. Las proteínas y los ácidos nucleicos son polímeros heterogéneos, ya que todos los monómeros que los conforman son diferentes. En las proteínas existen 20 aminoácidos estándar y en los ácidos nucleicos 4 nucleótidos diferentes [8].

Los 20 aminoácidos estándar de las proteínas comparten la misma estructura general, la cual se ilustra en la Figura 1-1. Todos los aminoácidos, excepto la prolina, son α-aminoácidos, ya que tienen un grupo amino ($-NH_2$) y un grupo carboxilo ($-COOH$) unidos al mismo átomo de carbono, llamado alfa (α) [6][8]. Cada α-aminoácido se diferencia de los demás en la estructura de su grupo R, también llamadas cadenas laterales.

Figura 1-1. Estructura general de un aminoácido.



Estructura general de los α -aminoácidos. Existen 20 grupos R diferentes en los aminoácidos estándar (imagen tomada de [6]).

De acuerdo con la estructura general de los α aminoácidos, al presentarse 20 opciones para cada aminoácido en una cadena polipeptídica existe un gran número de proteínas diferentes [6]. Se utilizan convenciones para identificar los carbonos dentro de un aminoácido; los carbonos adicionales de un grupo R se designan comúnmente como β , γ , δ , ε, etc., empezando a partir del carbono α [9].

Las cadenas laterales de los aminoácidos poseen propiedades físico-químicas que determinan cómo un aminoácido participa en la interacción con otros, así como cuál es su papel durante el plegamiento de la proteína y su función [7]; esto se debe a que las proteínas se pliegan adoptando su conformación nativa, en gran medida como respuesta

a separar sus cadenas laterales hidrofóbicas del contacto con el agua y a la asociación con el agua de sus cadenas laterales hidrofílicas[6][9].

Los 20 aminoácidos estándar se clasifican según la polaridad de sus cadenas laterales. De acuerdo con las características del grupo R, los aminoácidos se clasifican en tres grupos: aminoácidos con grupos R no polares, aminoácidos con grupos R polares sin carga, y aminoácidos con grupos R polares con carga [6][9]. A los aminoácidos estándar se les han asignado abreviaturas de tres letras y símbolos de una sola letra que se utilizan para indicar de manera abreviada la composición y secuencia de aminoácidos en las proteínas [9].

Los residuos hidrofóbicos no interactúan favorablemente con el agua, por lo tanto esos residuos tienden a evitar el contacto con esta y a menudo interactúan con otros residuos hidrofóbicos o con otros grupos no polares. El efecto hidrofóbico es uno de los factores estabilizantes de las estructuras plegadas de las proteínas [7].

Los residuos polares pueden interactuar con otros residuos polares, con el agua, y con grupos polares en ligandos a través de puentes de hidrógeno e interacciones electrostáticas [7].

Los residuos polares sin carga son más solubles en agua o hidrofílicos que los aminoácidos no polares, ya que tienen grupos funcionales que forman puentes de hidrógeno con el agua.

Los aminoácidos más hidrofílicos son los que están cargados ya sea positiva o negativamente. Los aminoácidos cargados positivamente (básicos) son aquellos en los que los grupos R tienen una carga neta positiva significativa a pH 7,0. Los aminoácidos cargados negativamente (ácidos) son aquellos en los que los grupos R tienen una carga neta negativa a pH 7,0 [9].

1.1.2 Estructura de las proteínas

Las proteínas desempeñan una variedad de funciones bioquímicas esenciales. Las propiedades funcionales de las proteínas son determinadas por su estructura tridimensional. En general, todas las funciones de las proteínas involucran un reconocimiento específico y la unión con otras moléculas (ligandos pequeños o

macromoléculas). La capacidad de las proteínas de reconocer y unirse a ligandos específicos es la propiedad más importante asociada con su función [7].

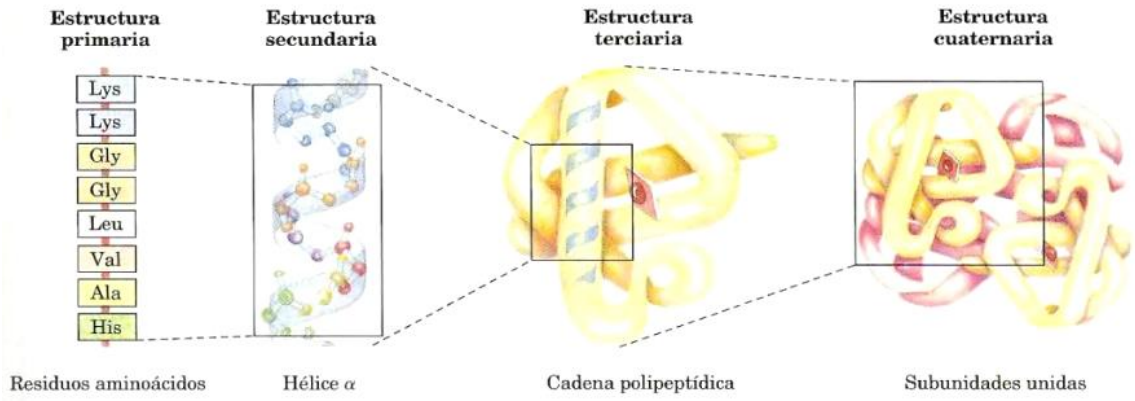
Las proteínas se pueden clasificar de acuerdo con su función, su composición y su forma [8]. En la clasificación de las proteínas según su función, se pueden encontrar las proteínas catalíticas, que son enzimas que incrementan o regulan ciertas reacciones químicas dentro de la célula, y las proteínas estructurales que carecen de una verdadera función dinámica (no participan en reacciones químicas), entre otras [8].

Existen dos categorías basadas en la composición natural de las proteínas. Las proteínas conjugadas, las cuales tienen un componente polipeptídico asociado con un componente no peptídico (orgánico o inorgánico) llamado grupo prostético, y las proteínas no conjugadas, las cuales sólo tienen material polipeptídico [8].

De acuerdo con su forma y solubilidad, las proteínas se clasifican en dos categorías: las proteínas fibrosas, que tienen formas moleculares muy alargadas, y las proteínas globulares, que tienen una estructura mucho más compacta qué las proteínas fibrosas [8].

Independiente de que las proteínas sean fibrosas o globulares, las proteínas se pueden ver desde cuatro niveles estructurales [8] los cuales se presentan en la Figura 1-2. La estructura primaria hace referencia a la secuencia de aminoácidos de una proteína. La estructura secundaria se refiere a disposiciones particularmente estables de los aminoácidos que dan lugar a patrones estructurales repetitivos. Los elementos de la estructura secundaria de una proteína pueden unirse y formar la estructura terciaria de las proteínas, la cual puede contener una o varias unidades estructurales llamadas dominios. Un dominio es una estructura compacta y estable de una parte de la proteína que se pliega independientemente. Mientras que las proteínas más pequeñas pueden presentar un solo dominio, las proteínas de mayor tamaño presentan varios de ellos. Muchas proteínas contienen más de una cadena polipeptídica. La asociación de dos o más cadenas polipeptídicas se denominan agregados moleculares o complejos proteicos y hacen relación a la estructura cuaternaria de las proteínas [7][8][9][10].

Figura 1-2. Estructura de las proteínas.

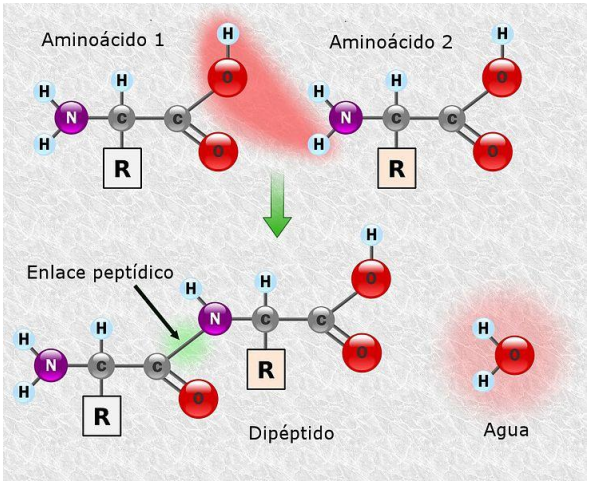


Estructura primaria, secundaria, terciaria y cuaternaria de las proteínas (imagen tomada de [9]).

Estructura primaria

La estructura tridimensional de una proteína se basa en su secuencia de aminoácidos. En la formación de las proteínas un aminoácido se une a otro a través de un enlace peptídico. El enlace peptídico es un enlace entre el grupo amino de un aminoácido y el grupo carboxilo de otro aminoácido. En la formación del enlace peptídico se elimina una molécula de agua, quedando los residuos de los aminoácidos que lo componen. La formación de un enlace peptídico se presenta en la Figura 1-3.

Figura 1-3. Enlace peptídico.



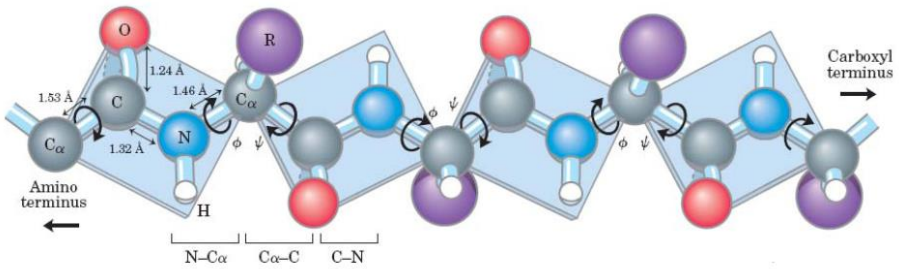
Formación de un enlace peptídico entre dos aminoácidos (imagen tomada de [11])

Los compuestos formados por dos, tres, unos pocos (3 – 10) o muchos aminoácidos se conocen como dipéptidos, tripéptidos, oligopéptidos y polipéptidos, respectivamente [6]. El término proteína se utiliza para moléculas relativamente grandes (50 o más aminoácidos), mientras que el término péptido se utiliza para moléculas pequeñas (menos de 50 aminoácidos). Cada proteína posee una secuencia de aminoácidos característica, la cual provee información acerca de esta [7][9].

El esqueleto de las proteínas es la secuencia formada por los enlaces peptídicos que conectan los aminoácidos de una proteína sin tener en cuenta el grupo R, es decir, es la secuencia de enlaces entre el nitrógeno del grupo amino N(H) , el carbono α (R), y el carbono del grupo carbonilo C(=0) [7].

Los enlaces covalentes imponen límites a las posibles conformaciones de una proteína. En el enlace peptídico los carbono α de aminoácidos adyacentes se encuentran separados por tres enlaces covalentes, ordenados así: $C_\alpha - C - N - C_\alpha$. El enlace $C - N$ no puede rotar libremente debido a su carácter parcial de doble enlace. La rotación es posible alrededor de los enlaces $N - C_\alpha$ (φ phi) y $C_\alpha - C$ (ψ psi). El esqueleto de las proteínas puede visualizarse como una serie de planos rígidos en la que los planos consecutivos comparten un punto común de giro alrededor del C_α como se presenta en la Figura 1-4. No todos los valores para φ y ψ son permitidos debido a posibles restricciones estéricas entre los grupos no vecinos del esqueleto de las proteínas, reduciendo el número de posibles estructuras regulares en las proteínas [9][10].

Figura 1-4. Esqueleto de las proteínas.



Tres enlaces separan los carbonos α consecutivos en una cadena polipeptídica. Los enlaces $N - C_\alpha$ (φ phi) y $C_\alpha - C$ (ψ psi) pueden rotar. El enlace $C - N$ no puede rotar libremente. (imagen tomada de [9]).

En principio φ y ψ pueden adoptar cualquier valor entre -180° y 180°, pero, como se mencionó anteriormente, muchos de esos valores no son factibles a causa de las restricciones estéricas entre átomos del esqueleto polipeptídico y de las cadenas laterales de los aminoácidos [9]. Los valores permitidos para φ y ψ se pueden visualizar gráficamente en el gráfico de Ramachandran [7][9][10]. El gráfico de Ramachandran, que se presenta en la Figura 1-5, representa a ψ frente a φ. Las conformaciones consideradas posibles son las que tienen poca o ninguna interferencia estérica. Las áreas sombreadas en azul oscuro reflejan conformaciones sin solapamiento estérico y por lo tanto permitidas; el azul de intensidad media indica conformaciones permitidas en los límites extremos de contactos atómicos desfavorables; y el azul claro indica conformaciones que están permitidas si se presenta cierta flexibilidad en los ángulos de enlace [9].

Figura 1-5. Gráfico de Ramachandran.

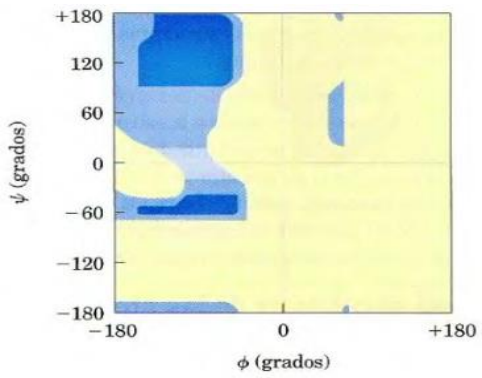


Gráfico de Ramachandran (imagen tomada de [9]).

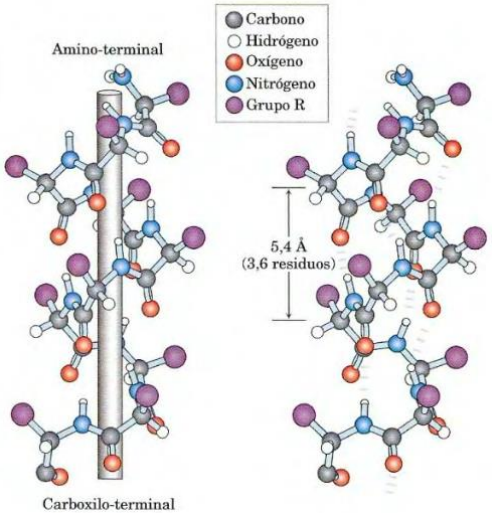
Estructura secundaria

La estructura secundaria de una proteína se define como la conformación local de su esqueleto [6]. Todos los tipos comunes de estructuras secundarias regulares que se encuentran en las proteínas caen dentro de las regiones permitidas del gráfico de Ramachandran [6]. Las conformaciones de estructura secundaria más destacables son las hélices α y las hojas β [9].

Las hélices son los elementos más sobresalientes de la estructura secundaria de las proteínas. Si una cadena polipeptídica experimenta un giro de la misma magnitud alrededor de cada uno de sus átomos de C_α , adopta una conformación helicoidal. Como

una alternativa para especificar los ángulos φ y ψ, una hélice puede caracterizarse por el número, $\boldsymbol{n}$, de unidades peptídicas por vuelta de la hélice y por la distancia de la vuelta complete de esta, $\boldsymbol{p}$, que es la distancia que indica el ascenso de la hélice a lo largo de su eje. Las hélices pueden girar a la izquierda o a la derecha [6]. La hélice α, presentada en la Figura 1-6, es la conformación más sencilla que puede asumir una cadena polipeptídica debido a que presenta un modelo de enlaces de hidrógeno favorable. En esta estructura el esqueleto polipeptídico se encuentra compactamente enrollado alrededor del eje longitudinal imaginario de la molécula y los grupos R de los residuos aminoácidos sobresalen del esqueleto helicoidal [9].

Figura 1-6. Hélice α.



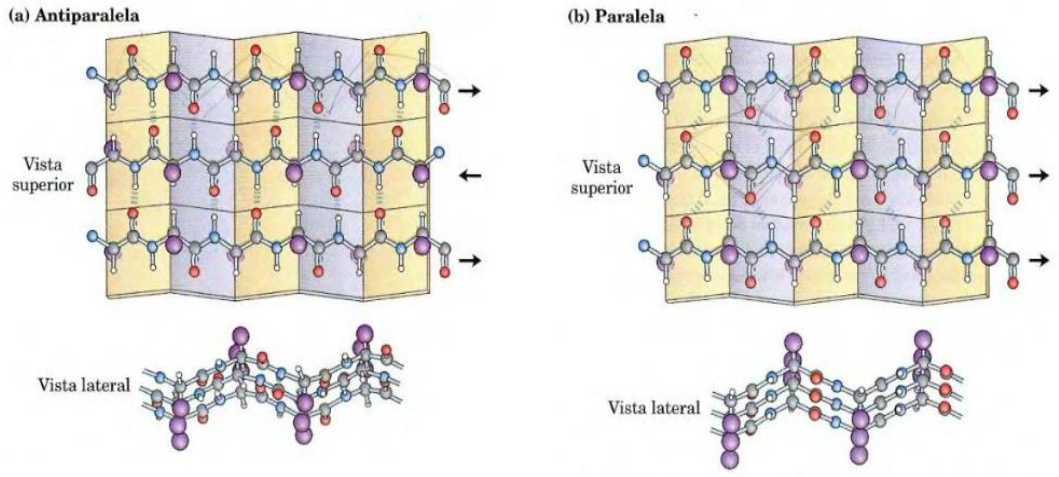
Hélice α (imagen tomada de [9]).

Las láminas u hojas β son un segundo tipo de estructura secundaria en la que el esqueleto de la cadena polipeptídica está extendido en zigzag. Las cadenas polipeptídicas en zigzag pueden disponerse de manera adyacente formando una estructura que se asemeja a la de una serie de pliegues. Los aminoácidos que forman una lámina β están normalmente cercanos en la cadena polipeptídica pero también pueden estar muy distantes uno de otro dentro de la secuencia lineal del polipéptido; incluso pueden pertenecer a segmentos de diferentes cadenas polipeptídicas. Los grupos R de aminoácidos adyacentes sobresalen de la estructura en zigzag en direcciones opuestas, dando lugar a un patrón alternante [9]. En las láminas β los enlaces de

hidrógeno se establecen entre cadenas polipeptídicas vecinas en lugar de en el interior de la misma cadena como ocurre en las hélices α [6].

Las cadenas polipeptídicas adyacentes de una hoja β, las cuales se presentan en la Figura 1-7, pueden ser paralelas (con la misma orientación amino-carboxilo en el polipéptido) o anti paralelas (con orientaciones opuestas) [9].

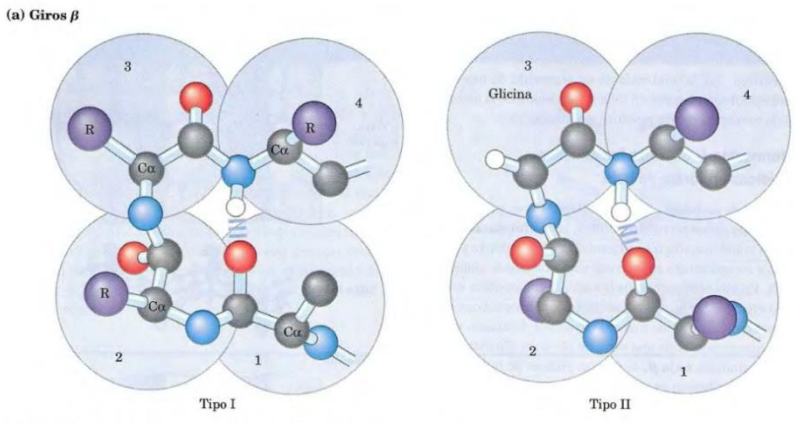
Figura 1-7. Hoja β.



Hoja β (imagen tomada de [9]).

Las estructuras secundarias regulares (hélice α y hoja β) representan algo menos de la mitad de las proteínas globulares estándar. Los segmentos polipéptidos restantes de la proteína se dice que poseen una conformación de giro o bucle [6]. Las proteínas globulares están constituidas, en su mayor parte, por tramos aproximadamente rectos de estructura secundaria, unidos por hebras de polipéptidos que cambian de dirección abruptamente. Estos giros de inversión o torsión β (designados de este modo porque, con frecuencia, conectan hebras sucesivas de hojas β antiparalelas) aparecen casi siempre en la superficie de las proteínas; en realidad, definen parcialmente estas superficies. La mayor parte de giros de inversión incluyen cuatro residuos de aminoácidos sucesivos dispuestos, más o menos, en dos tipos: el Tipo I y el Tipo II, los cuales se presentan en la Figura 1-8, que se diferencian por un cambio de dirección de 180° de la unidad peptídica que liga los residuos 2 y 3 [6][9].

Figura 1-8. Giros β.



Giros β (imagen tomada de [9]).

Estructura terciaria

La estructura tridimensional de una proteína hace referencia a la disposición tridimensional de todos los átomos de esta. La estructura terciaria es la estructura tridimensional completa de una cadena polipeptídica. Mientras que la estructura secundaria se refiere al ordenamiento espacial de los residuos aminoácidos adyacentes en la estructura primaria, la estructura terciaria incluye aspectos de largo alcance en la secuencia de aminoácidos. Aminoácidos que se encuentran alejados en la secuencia polipeptídica y que se encuentran en tipos de estructura secundaria diferentes pueden interactuar dentro de la estructura totalmente plegada de la proteína.

Muchos de los enlaces de una larga cadena polipeptídica permiten la libre rotación de los átomos que unen, lo cual confiere al esqueleto de la proteína una gran flexibilidad. Por lo tanto, cualquier molécula proteica puede, en principio, adoptar un número grande de formas diferentes (conformaciones). Sin embargo, la mayoría de las cadenas polipeptídicas se pliegan adoptando una sola conformación particular. Siempre que las cadenas laterales apropiadas se encuentren en posiciones cruciales de la cadena, se ejercerán importantes fuerzas que hacen que una conformación particular sea especialmente estable [10].

La mayoría de las proteínas pueden plegarse espontáneamente en su forma correcta. Uno de los factores más importantes que condicionan el plegamiento de una proteína es

la distribución de sus cadenas laterales polares y no polares. Las cadenas hidrofóbicas, tienden a agruparse en el interior de la molécula, lo cual les permite evitar el contacto con el entorno acuoso. Por el contrario, las cadenas laterales polares tienden a disponerse cerca de la parte externa de la molécula proteica, donde pueden interactuar con el agua y con otras moléculas polares [10]. La localización y características químicas de los diferentes átomos de la superficie hacen que cada proteína sea única y permiten a la molécula fijarse a otras superficies macromoleculares y a ciertas moléculas pequeñas [10].

Se denomina conformación a la disposición espacial de los átomos de una proteína. Las posibles conformaciones de una proteína incluyen cualquier estado estructural que pueda lograrse sin romper enlaces covalentes. De entre las numerosas conformaciones teóricamente posibles para una proteína que tiene cientos de enlaces sencillos, pocas predominan en condiciones biológicas. Las conformaciones existentes en unas condiciones determinadas son generalmente las más estables termodinámicamente y las que poseen la menor energía libre de Gibbs. Las proteínas que se encuentran en cualquiera de sus conformaciones funcionales y plegadas se denominan conformaciones nativas. En el contexto de la estructura de las proteínas, el termino estabilidad se puede definir como la tendencia a mantener la conformación nativa. Entre las interacciones químicas que estabilizan la conformación nativa se incluyen los enlaces disulfuro y las interacciones débiles (no covalentes): enlaces de hidrógeno, interacciones iónicas e interacciones hidrofóbicas. El papel de estas interacciones es importante para comprender de qué modo son capaces de plegarse las cadenas polipeptídicas adoptando estructuras secundarias y terciarias específicas y de combinarse con otros polipéptidos para formar estructuras cuaternarias [10].

Los enlaces covalentes individuales que contribuyen a las conformaciones nativas de las proteínas son mucho más fuertes que las interacciones débiles individuales. Sin embargo, al ser tan numerosas, las interacciones débiles son las que predominan como fuerza estabilizadora de la estructura de las proteínas. En general, la conformación proteica de menor energía libre (es decir, la más estable) es aquella que posee mayor número de interacciones débiles [10].

Estructura cuaternaria

Muchas proteínas presentan múltiples subunidades polipeptídicas. La asociación de cadenas polipeptídicas puede servir para diversas funciones. Gran número de proteínas que están compuestas por subunidades tienen funciones reguladoras; la unión de pequeñas moléculas puede alterar la interacción entre subunidades produciendo grandes cambios en la actividad de la proteína en respuesta a pequeños cambios en la concentración de sustrato o moléculas reguladoras. En otros casos, subunidades diferentes pueden llevar a cabo funciones distintas aunque relacionadas, como la catálisis y la regulación [9].

Una proteína con más de una subunidad es un multímero. Las proteínas multiméricas pueden tener de dos a cientos de subunidades. Un multímero con sólo unas pocas subunidades se denomina a menudo oligómero**.** Si un multímero está constituido por varias subunidades diferentes, la estructura global de la proteína puede ser asimétrica y bastante complicada. Sin embargo, la mayoría de los multímeros tienen subunidades idénticas o grupos repetidos de subunidades no idénticas a menudo dispuestos simétricamente. La unidad de repetición estructural en este tipo de proteína multimérica, sea una sola subunidad o un grupo de subunidades, se denomina protómero [9].

1.1.3 Estabilidad de la estructura de las proteínas

La mayoría de proteínas se pliegan de forma estable, de tal manera que su secuencia de aminoácidos pueda tener muchas más interacciones favorables en la propia cadena que cualquier otra [10].

Una cadena macromolecular se mantiene unida por medio de enlaces covalentes, que son suficientemente fuertes como para preservar la secuencia de subunidades durante largos períodos de tiempo. A pesar de que la secuencia de subunidades determina la información contenida en una macromolécula, la utilización de esta información depende en gran medida de otros enlaces mucho más débiles, los enlaces no covalentes. Estos enlaces débiles se forman entre diferentes regiones de la misma macromolécula y también entre diferentes macromoléculas. Por ello, estos enlaces determinan en gran medida tanto la estructura tridimensional de las cadenas macromoleculares como la manera en que estas estructuras interaccionan entre sí [7][10].

Los enlaces no covalentes que se presentan en las moléculas biológicas suelen clasificarse en tres tipos: enlaces iónicos, enlaces de hidrógeno y atracciones de van der Waals. Otra fuerza débil se genera por la estructura tridimensional del agua, la cual tiende a unir los grupos hidrofóbicos con el fin de minimizar su efecto destructor de la red de moléculas de agua unidas por enlaces de hidrógeno. Esta expulsión de la solución acuosa genera lo que algunas veces se considera como un cuarto tipo de enlace débil no covalente, enlace hidrofóbico [10].

Fuerzas de van der Waals

A una distancia corta dos átomos cualesquiera muestran una débil interacción de enlace debido a sus cargas eléctricas fluctuantes. Esta fuerza recibe el nombre de atracción de van der Waals. Sin embargo, dos átomos se repelen muy energéticamente si se los acerca demasiado. Esta repulsión de van der Waals desempeña un papel importante limitando las posibles conformaciones de una molécula [10].

Cada tipo de átomo tiene un radio, conocido como radio de van der Waals, en el que las fuerzas de van der Waals están equilibradas. A pesar de que estas atracciones de van der Waals son individualmente muy débiles, pueden ser importantes cuando dos superficies macromoleculares se adaptan una a otra [10].

Enlaces de Hidrógeno

Un átomo de hidrógeno es compartido por dos átomos (ambos electronegativos, como el O y el N) formándose un enlace de hidrógeno. Las moléculas que pueden formar enlaces de hidrógeno con otras, también pueden alternativamente formar enlaces de hidrógeno con moléculas de agua. Debido a esta competencia con las moléculas de agua los enlaces de hidrógeno formados entre dos moléculas disueltas en agua son relativamente débiles [10].

Interacciones hidrofóbicas

El agua une los grupos hidrofóbicos con objeto de minimizar los efectos destructores de estos grupos sobre la red de enlaces de hidrógeno del agua [10]. Las interacciones hidrofóbicas juegan un papel especial en la estabilización de la conformación de las proteínas; el interior de una proteína es generalmente un núcleo densamente empaquetado de cadenas laterales hidrofóbicas de aminoácidos [9].

Enlaces iónicos

Las interacciones iónicas se producen entre grupos totalmente cargados (enlace iónico) o entre grupos parcialmente cargados [10]. Los enlaces iónicos son muy importantes en los sistemas biológicos. Una enzima que se una a un substrato cargado positivamente, a menudo presentará en el lugar apropiado un residuo de aminoácido cargado negativamente [10].

1.2 Fundamentos computacionales

Muchos de los problemas científicos actuales, requieren de la optimización o maximización de una función objetivo. Para esto, existen métodos de inteligencia computacional como los algoritmos de búsqueda que permiten explorar el espacio de soluciones y obtener una respuesta a dichos problemas. En el marco de este trabajo, los algoritmos de búsqueda constituyen la base para el modelo propuesto.

1.2.1 Algoritmos de búsqueda

En general, los algoritmos de búsqueda tienen como propósito encontrar una solución a un problema teniendo en cuenta las particularidades de este [12].

Un algoritmo de búsqueda toma como entrada un problema y devuelve una solución. El primer paso en un algoritmo de búsqueda es establecer el objetivo que se debe satisfacer, una vez hecho esto, se definen el conjunto de acciones a seguir con las cuales se cumplirá el objetivo. El proceso de búsqueda consiste en seleccionar las acciones a ejecutar basándose en los resultados obtenidos en cada paso [12].

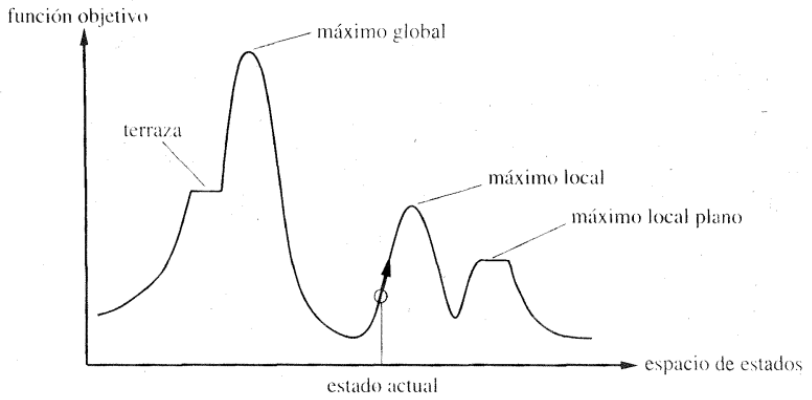
Un problema de búsqueda se puede definir formalmente por cuatro componentes [12]:

- El estado inicial en el que se comienza
- Conjunto de acciones disponibles en el estado actual. La formulación más común utiliza una función sucesor, la cual devuelve el conjunto de acciones y de estados alcanzables ejecutando las acciones disponibles. El estado inicial y la función sucesor definen el espacio de estados del problema. Un camino en el espacio de estados es una secuencia de estados conectados por una secuencia de acciones.

- La prueba del objetivo, la cual determina si un estado es un estado objetivo. Algunas veces existe un conjunto de posibles estados objetivo, y la prueba del objetivo simplemente comprueba si el estado es uno de ellos.
- Una función costo del camino, que asigna un costo numérico a cada trayectoria recorrida en el espacio de búsqueda, en la que se evalúan las posibles soluciones.

Una solución a un problema es una trayectoria desde el estado inicial a un estado objetivo. La calidad de la solución se mide por la función costo del camino, y una solución óptima tiene el menor costo entre todas las soluciones. En la Figura 1-9 se presenta el paisaje de estados, el cual es importante tener en cuenta en los algoritmos de búsqueda enfocados a resolver problemas de optimización. El estado actual corresponde a determinada solución encontrada en el espacio de búsqueda y, dependiendo del valor de la solución establecida, se determina si el estado corresponde a un mínimo local o global en el caso de problemas de minimización, o si corresponde a un máximo local o global en el caso de problemas de maximización [12].

Figura 1-9. Paisaje del espacio de estados en problemas de optimización.



En los problemas de minimización el objetivo es encontrar el mínimo global y en los problemas de maximización el máximo global (imagen tomada de [12])

1.2.2 Computación paralela y computación distribuida

En la última década la capacidad de procesamiento de los computadores personales se ha incrementado vertiginosamente, al mismo tiempo que el costo de adquisición de los mismos ha disminuido en el mercado, lo cual constituye un soporte importante en la función investigadora, mejorando los tiempos de ejecución y respuesta de algoritmos que demandan grandes recursos informáticos [13].

El concepto de paralelismo hace referencia a tener varias unidades de trabajo y la colaboración entre estas en la realización de determinada tarea. Este aspecto comprende desde computadores con múltiples procesadores hasta sistemas distribuidos geográficamente [13].

Existen varias formas de hacer sistemas paralelos, algunas arquitecturas proveen mejor rendimiento que otras, y esto depende del tipo de aplicación sobre la cual se está aplicando [13].

Los sistemas paralelos están clasificados en cuatro clases [13]:

- *Single Instruction Single Data* (SISD): Corresponde a los sistemas de un solo procesador que ejecutan una sola instrucción a la vez sobre datos almacenados en memoria.
- *Single Instruction Multiple Data* (SIMD): Es la clase de sistemas compuestos por un vector de procesadores o gran número de pequeñas unidades de cómputo que permiten la aplicación de una instrucción en varios datos al mismo tiempo. Los sistemas SIMD son utilizados para conseguir paralelismo a nivel de datos.
- *Multiple Instruction Single Data* (MISD): Es la arquitectura computacional en donde múltiples procesadores ejecutan instrucciones sobre los mismos datos.
- *Multiple Instructions Multiple Data* (MIMD): Son los sistemas capaces de realizar múltiples instrucciones en diferentes datos al mismo tiempo.

Otros criterios para tener en cuenta en la clasificación de las arquitecturas paralelas son el uso de la memoria y el alcance geográfico de los sistemas que componen el arreglo de procesadores [13].

En cuanto al uso de la memoria, esta puede ser compartida por los procesadores, es decir, todos los procesadores acceden a la misma memoria, o distribuida entre ellos, con lo cual cada procesador tiene su propia memoria y el acceso a ésta es exclusiva de cada procesador [13].

De acuerdo con la ubicación geográfica los sistemas de procesamiento paralelo se clasifican en [13]:

- *Monoprocessor Machines (MM):* son computadores con un solo procesador. Representan principalmente los computadores SISD e incluye sistemas SIMD que contienen un vector de procesadores.
- *Parallel Machines (PM):* son computadores que contienen varias unidades de procesamiento. Incluyen las arquitecturas SIMD y MIMD y potenciales combinaciones de estas.
- *Local Clusters (LC):* Es una colección de computadores independientes reunidos en el mismo espacio físico y conectados entre sí por medio de una red local. Aunque son intrínsecamente orientadas a la arquitectura MIMD, los sistemas de tipo SIMD pueden ser utilizados al nivel de los nodos.
- *Distributed Clusters (DC):* Es una colección de LC distribuidos alrededor del mundo y conectados entre sí a través de Internet. Principalmente siguen el modelo MIMD, pero puede incluir partes SIMD.

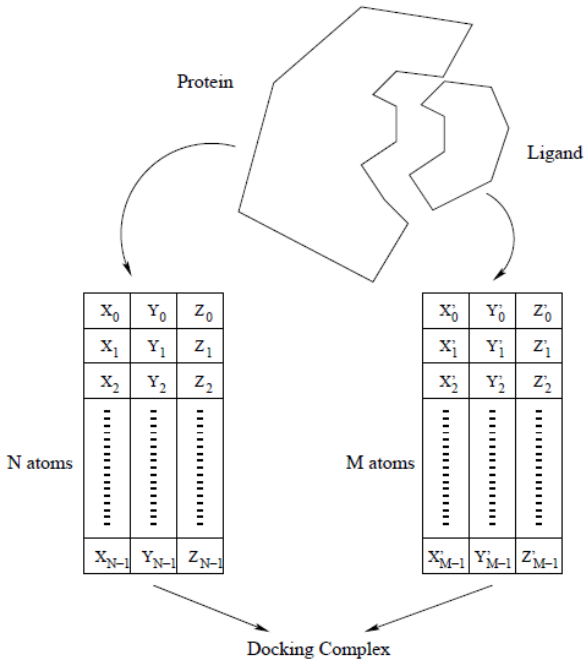
1.3 El problema de la interacción proteína-proteína

Gran parte de los procesos biológicos y bioquímicos dependen de las proteínas y de las interacciones entre estas. Existen métodos experimentales como la espectroscopía de masa o la técnica del doble híbrido en levadura con los cuales se investiga y se obtiene información acerca de la interacción proteína-proteína a una escala genómica. Además de la información obtenida por las técnicas experimentales existentes, es de interés conocer cómo las proteínas interactúan a nivel atómico, determinando la estructura tridimensional del complejo proteico de interés [14].

El problema de la interacción proteína-proteína también se conoce con el nombre de *docking* proteína-proteína. El término *docking* es utilizado para hacer referencia a los esquemas computacionales que intentan encontrar la mejor conformación entre dos

moléculas: el ligando y el receptor [2]. El problema de la interacción proteína-proteína se puede definir de la siguiente manera: Dadas las coordenadas atómicas de dos moléculas, predecir la mejor asociación entre éstas [2]. La Figura 1-10 presenta un ejemplo de interacción molecular.

Figura 1-10. Interacción molecular



A partir de las coordenadas atómicas de dos proteínas, el problema de la interacción proteína-proteína consiste en encontrar las coordenadas atómicas del complejo proteico que se forma (imagen tomada de [15]).

De forma general la entrada al problema de la interacción proteína-proteína la constituyen las estructuras individuales de las proteínas que forman el complejo de interés [14][2]; adicionalmente, también se puede tener información acerca de los sitios de unión de las mismas. Se debe tener en cuenta que esta información está relacionada con potenciales sitios de unión y que pueden existir otros sitios de unión adicionales a los que se indiquen como entrada del problema [2]. Si bien el conocimiento de los potenciales sitios de unión facilita el problema, no se garantiza que los sitios de unión indicados sean los reales. Se supone que el sitio de unión que se indica en el momento de modelar la interacción proteína-proteína, participa en la conformación del complejo proteico que se forma [2].

Las estructuras de las proteínas que forman los complejos proteína-proteína, generalmente son obtenidas a través de métodos como la Resonancia Magnética Nuclear (NMR) o la cristalografía de rayos X. Con el método de cristalografía de rayos X se obtiene una única estructura de la proteína en estudio, mientras que con la NMR se obtienen múltiples copias de la proteína en diferentes conformaciones. Para la ejecución de métodos orientados a la predicción de la interacción proteína-proteína se recomienda utilizar las estructuras obtenidas a través de la cristalografía de rayos X con resolución menor a 2.5 Å, ya que con las estructuras obtenidas por medio de la NMR se debe seleccionar la estructura más representativa teniendo en cuenta las restricciones experimentales utilizadas en el proceso de determinación de la estructura proteica. Usualmente, las proteínas de entrada se obtienen en formato PDB el cual contiene las coordenadas cartesianas (x, y, z) de las posiciones de los átomos [14].

El *docking* proteína-proteína puede ser subdividido en dos categorías: *docking* no predictivo y *docking* predictivo, de acuerdo con el origen de las proteínas que van a interactuar. En el *docking* no predictivo, las estructuras del ligando y del receptor se obtienen a partir de la estructura del complejo que forman, la cual es conocida. El objetivo del *docking* no predictivo es reconstruir el complejo conocido. En el *docking* predictivo las estructuras del ligando y del receptor son determinadas individualmente y el objetivo es realizar la predicción de la estructura del complejo que forman. En términos de dificultad el *docking* predictivo es más difícil que el *docking* no predictivo, puesto que se deben modelar los movimientos de las cadenas laterales y del esqueleto de las proteínas durante el proceso de unión [14][2].

Para modelar el problema de *docking* molecular se deben tener en cuenta 3 elementos fundamentales: la representación del sistema, el espacio de búsqueda de conformaciones y la función de costo de las posibles soluciones [2]. Teniendo en cuenta lo anterior para obtener una posible solución al problema de la interacción proteína-proteína, desde el punto de vista computacional, se deben plantear algoritmos de búsqueda eficientes que exploren el espacio de soluciones de manera efectiva y funciones de costo que permitan clasificar y dar conceptos entre dos soluciones obtenidas [2].

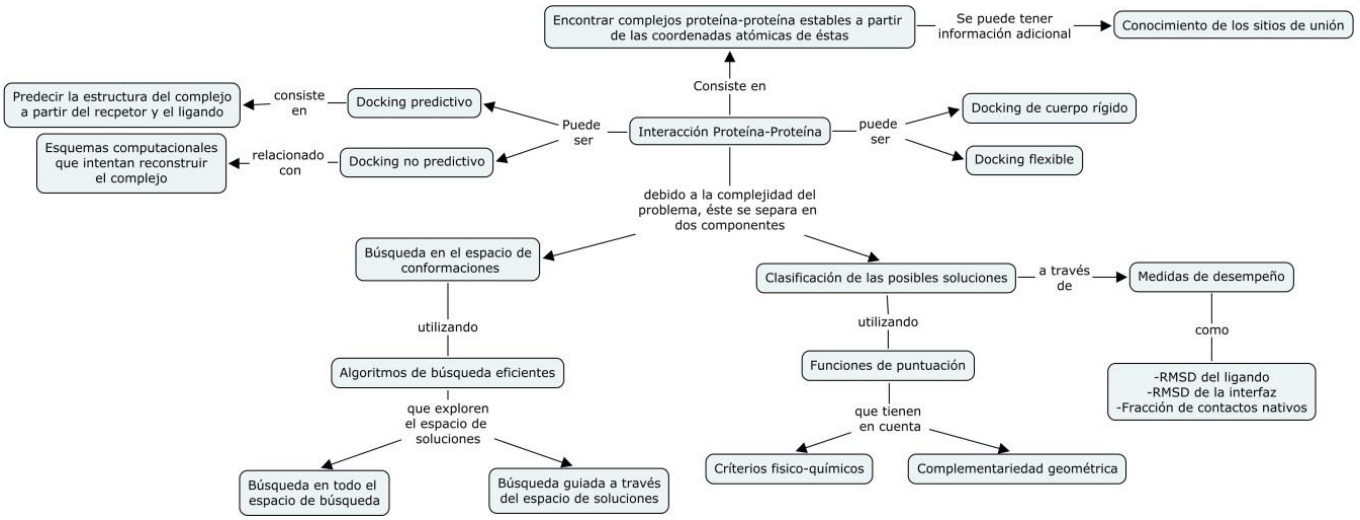
Debido al tamaño de las moléculas, explorar todo el espacio de búsqueda de posibles conformaciones que puede adquirir un complejo proteico formado por dos proteínas puede ser no factible. Los grados de libertad del problema están dados por el nivel de flexibilidad considerada en las proteínas. Teniendo en cuenta la flexibilidad en el ligando, los grados de libertad del problema son el número de sus ángulos de torsión, y considerando las proteínas como cuerpos rígidos, los grados de libertad son seis: tres traslacionales y tres rotacionales.

Frente a lo anterior, para realizar una exploración del espacio de conformaciones de complejos proteína-proteína, el problema de la interacción proteína-proteína se divide en dos etapas para su solución: En la primera etapa las proteínas se consideran como cuerpos rígidos, mientras que en la segunda se realiza un refinamiento de las estructuras obtenidas en la primera etapa, incluyendo cierto grado de flexibilidad en el ligando.

Teniendo en cuenta los costos computacionales en términos de tiempo de ejecución y consumo de memoria, necesarios para predecir la interacción proteína-proteína, se han realizado implementaciones paralelas y distribuidas de los algoritmos de búsqueda empleados para hacer frente a la mencionada situación.

El esquema general del problema de la predicción de la interacción proteína-proteína se presenta en la Figura 1-11.

Figura 1-11. Esquema general del problema de la interacción proteína-proteína.



2. Antecedentes

En este capítulo se presentan métodos existentes para la predicción de la interacción proteína-proteína. Se parte de la revisión de algoritmos de búsqueda que han sido utilizados, tanto para *docking* de cuerpo rígido como para *docking* flexible. Se muestran enfoques que han sido propuestos para la inclusión de flexibilidad en el ligando, algunas de las funciones de evaluación utilizadas, algoritmos para la predicción de sitios de unión y, finalmente, las medidas de rendimiento de los métodos de *docking* computacional desarrollados.

2.1 Métodos de búsqueda

En la década de los 70 Wodak y Janin propusieron el primer algoritmo para predecir la interacción de las estructuras en 3D de la tripsina pancreática y su inhibidor natural; desde entonces, el *docking* proteína-proteína ha llegado a ser una disciplina computacional que reúne los conocimientos y técnicas de ciencias como la física, la química, la biología, las matemáticas y la computación, con el objetivo de modelar el comportamiento de macromoléculas tales como las proteínas [16].

La predicción de la interacción entre proteínas parte de la representación de estas computacionalmente. Para la representación de la superficie de las proteínas se tienen modelos matemáticos, como descriptores de la forma geométrica que contienen las descripciones físico-químicas de la proteína, y modelos de representación a partir de mallas o cuadrículas [17].

Las proteínas también se pueden representar geométricamente; la geometría molecular es la disposición tridimensional de los átomos que constituyen una molécula. La posición de cada átomo se determina por la naturaleza de los enlaces químicos con los que se

conecta con sus átomos vecinos. La geometría molecular puede describirse por las posiciones de los átomos en el espacio, por la longitud del enlace que forman dos átomos, por el ángulo que forman los enlaces de tres átomos conectados, y por los ángulos de torsión que forman tres enlaces consecutivos [18].

Dada la posibilidad de que los ángulos de rotación cambien, se puede tener más de una configuración (relación geométrica entre un conjunto determinado de átomos) de los complejos. En el sentido clásico, los enfoques para predecir la interacción entre dos proteínas se dividen en algoritmos de cuerpo rígido y en algoritmos de *docking* flexible [2].

Gran número de métodos comienzan con el docking de cuerpo rígido, el cual genera una lista inicial de conformaciones encontradas con complementariedad de la superficie favorable y, posiblemente, con buenas propiedades electrostáticas. El método usado con mayor frecuencia es el de la correlación de la Transformada Rápida de Fourier (FFT) introducido en 1992. Debido a la eficiencia numérica de este algoritmo se realizan los cálculos de acoplamiento en una primera instancia, mientras que el uso de otros algoritmos de optimización en el filtrado de las soluciones se hace posible teniendo en cuenta las capacidades de cómputo [19].

Los algoritmos de correlación de Fourier trabajan con representaciones moleculares basadas en mallas. A cada nodo de la malla se le asigna un valor numérico correspondiente a la superficie de la proteína, al núcleo de la proteína o al espacio vacío. La complementariedad de la superficie se determina calculando la correlación de Fourier de dos mallas [5]. El enfoque de la Transformada Rápida de Fourier ha sido usado en programas como ZDOCK [20], ClusPro [21], FRODOCK [22], PIPER [23], HexServer [24] y F3DOCK [25], entre otros.

Otro método usado es el Hashing Geométrico, en donde los puntos críticos de la superficie de una molécula son usados para definir un sistema de referencia local, en el cual, las posiciones de los puntos cercanos a los puntos críticos son usados como índices en una tabla hash que almacena el marco de coordenadas [2].

La optimización por enjambre de partículas es un método en el cual se inicializa el conjunto de soluciones con N partículas. Cada partícula es un punto en el espacio de búsqueda, lo cual indica una posible solución. En este método se tiene en cuenta la posición de cada partícula, la velocidad de cada partícula con la cual se va a mover a través del espacio de soluciones, la mejor solución individual y la mejor solución grupal, las cuales van a guiar la solución del problema. En [26] los autores solucionan el problema del *docking* molecular utilizando este enfoque; cada partícula hace referencia a un complejo en el cual se tiene en cuenta la flexibilidad del ligando. Las soluciones son evaluadas utilizando una función de puntuación que tiene en cuenta tanto la energía intramolecular del ligando como la del receptor, y la energía del complejo.

El recocido simulado es un método probabilístico de optimización. El algoritmo comienza a partir de un estado específico o aleatorio, a una temperatura inicial y con un esquema de enfriamiento. En cada paso de la simulación el ligando explora el espacio de conformaciones realizando pequeños cambios en cada uno de los grados de libertad del problema. En el caso de la interacción proteína-proteína cada conformación es evaluada utilizando una función de energía. Si la energía del nuevo complejo es menor que la energía del anterior el cambio conformacional se acepta, si no, el nuevo estado se escoge teniendo en cuenta el valor de la temperatura y el delta de energía [27].

Varios enfoques han utilizado los algoritmos genéticos para resolver el problema de *docking* molecular o de interacción proteína-proteína. Un algoritmo genético es un método de búsqueda basado en cambios sucesivos sobre poblaciones de individuos, en el cual cada individuo corresponde a una solución de un problema particular. Los cromosomas consisten en las posiciones relativas de las moléculas, teniendo la opción de incluir flexibilidad en el ligando [17]. La función de puntuación es la energía libre entre el ligando y el receptor. Los cambios en cada individuo se realizan a través de los operadores genéticos. Los operadores genéticos comúnmente utilizados son: selección, cruce y mutación. El operador de selección se utiliza para escoger los individuos que se cruzarán para generar nuevas soluciones. El operador de cruce simula la reproducción sexual y genera nuevas soluciones que comparten características de los individuos seleccionados, y el operador de mutación, es un operador que modifica el cromosoma de un solo individuo [27].

Los algoritmos de planeación de rutas son algoritmos originalmente aplicados en la robótica, y tienen como objetivo encontrar un camino libre de colisiones para los robots con varios grados de libertad. Los algoritmos de planeación de rutas tienen varias etapas: en la primera etapa o etapa de aprendizaje se generan las soluciones en el espacio de búsqueda y se almacenan como un grafo dirigido, en el cual las aristas se crean de acuerdo con una probabilidad de transición que tiene en cuenta la energía de cada solución. En la etapa de consulta se encuentra el camino óptimo entre dos soluciones recorriendo el conjunto de aristas que hay entre estas. Las moléculas pueden ser modeladas como estructuras articuladas en un espacio tridimensional, las cuales se mueven bajo la influencia del potencial de energía. Las diferentes rutas en el contexto de la biología molecular estructural hacen referencia a la discretización de las regiones energéticamente factibles del espacio de conformaciones [28].

En varios métodos de *docking* hay un tratamiento implícito de flexibilidad al permitir cierto grado de interpenetración de la superficie de las proteínas que se encuentran en interacción [29].

La manera más común de incluir flexibilidad en las proteínas es permitiendo movimientos en todo el esqueleto de las mismas. Para muchas proteínas, cambios pequeños en el esqueleto pueden alterar la energía de la proteína dramáticamente y durante el proceso de minimización de energía o inclusión de flexibilidad, se pueden obtener resultados energéticamente mejores o se puede inducir a errores. Se debe tener en cuenta que al incluir flexibilidad en todo el esqueleto de las proteínas se aumentan los grados de libertad del problema, siendo más difícil encontrar el mínimo global [30].

Además de considerar flexibilidad en el esqueleto de las proteínas, también existen modelos que permiten incluir flexibilidad en las cadenas laterales. Sin embargo, es necesario identificar las regiones flexibles de una proteína. Muchos métodos describen la flexibilidad de las cadenas laterales basados en una librería de rotámeros [29][31], la cual define conformaciones de las cadenas laterales energéticamente factibles.

2.2 Funciones de puntuación para el problema de la interacción proteína-proteína

Respecto a la evaluación de las soluciones, la función de puntuación más sencilla es la complementariedad geométrica [2][17]. La complementariedad geométrica juega un papel importante en la determinación de la estructura de un complejo. Las estructuras tridimensionales de la mayoría de complejos proteicos presentan una alta complementariedad entre las interfaces del ligando y del receptor. Esta función se utiliza como filtro para la generación de un primer conjunto de soluciones, en las cuales se pueden presentar falsos positivos, con complejos proteicos con complementariedad geométrica mejor que una solución energéticamente más favorable. Otras funciones tienen en cuenta la superposición intermolecular e intramolecular de las estructuras y el área de contacto entre las proteínas.

Las funciones de energía existentes para las proteínas se pueden clasificar de la siguiente manera: funciones basadas en campos de fuerza empíricos y funciones de energía basadas en el conocimiento [32].

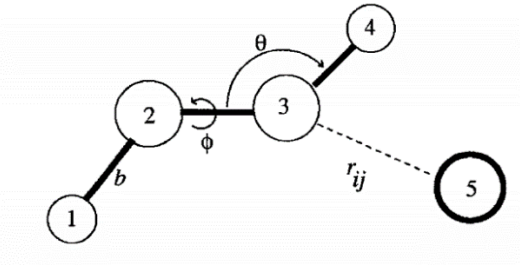
En las funciones basadas en campos de fuerza empíricos, la energía potencial de una proteína describe las interacciones entre los átomos de la misma [32]. La energía potencial es calculada en función de las posiciones de los átomos de la proteína e incluye términos para las contribuciones de las interacciones internas y externas. Las interacciones internas incluyen los enlaces entre dos átomos, los ángulos formados por dos enlaces y los ángulos de torsión. Las contribuciones externas incluyen las interacciones de van der Waals [32]. La función de energía potencial comúnmente utilizada se presenta en la Ecuación (2-1) [33].

$$V(r) = \sum_{bonds} k_b(b - b_0)^2 + \sum_{angles} k_\theta(\theta - \theta_0)^2 + \sum_{torsions} k_\phi\,[cos(n\phi + \delta) + 1] + \sum_{nonbond\ pairs} \left[\frac{q_i q_j}{r_{ij}} + \frac{A_{ij}}{r_{ij}^{12}} - \frac{C_{ij}}{r_{ij}^{6}}\right] \quad (2\text{-}2)$$

Los tres primeros términos de la función de energía potencial corresponden a la energía de los enlaces formados entre dos átomos (interacción entre 1 y 2 presentada en la Figura 2-1), de los ángulos que forman dos enlaces (interacción entre 1 y 3 en la Figura

2-1) y de los ángulos de torsión (interacción entre 1 y 4 en la Figura 2-1) respectivamente. El término restante describe la energía electrostática a partir de las cargas parciales de cada átomo. Este término a menudo se llama el término de van der Waals [33].

Figura 2-1. Interacciones atómicas



(imagen tomada de [33])

Las funciones de energía basadas en el conocimiento se pueden clasificar en dos grupos: el primer grupo se basa en el análisis estadístico de las bases de datos de estructura de proteínas, en el cual la potencial interacción entre dos residuos se estima a partir de su frecuencia relativa en las bases de datos; y el segundo grupo, el cual se basa en la optimización, en el que los parámetros de la función de energía son optimizadas por algún criterio [32].

2.3 Predicción de sitios de unión

Una proteína se une a otra a través de los sitios de unión. Con el objetivo de entender el proceso mediante el cual las proteínas interactúan, es importante conocer las características de los sitios de unión. Computacionalmente los sitios de unión se identifican a partir de la estructura tridimensional de las proteínas o de sus secuencias. Los métodos que se han utilizado para la predicción de sitios de unión se basan en la identificación de características generales que son compartidas por varios sitos de unión, con las cuales se identificarán potenciales nuevos sitios. La diferencia entre los métodos radica en el conjunto de datos de entrenamiento del algoritmo de clasificación, en los descriptores y en el algoritmo seleccionado [14].

Los algoritmos que se han utilizado principalmente han sido algoritmos de clasificación como las máquinas de soporte vectorial (*Support Vector Machines (SVM)*) o las redes bayesianas [34][35].

La metodología general para predecir los sitios de unión en una proteína es la siguiente: seleccionar un conjunto de datos de entrenamiento y definir los residuos que son de superficie o de interfaz basados en información como la conservación de la secuencia de la proteína, información sobre la estructura secundaria, información sobre el área accesible al solvente, información relacionada con las vecindades de cada residuo utilizando ventanas deslizantes, entre otras. [34][36]. Una vez se tiene el vector de características, con la información sobre residuos de interfaz o de superficie, se entrena el algoritmo de clasificación seleccionado y, finalmente, se realiza la clasificación del residuo objetivo.

2.4 Evaluación de los complejos proteicos obtenidos

Como se dijo anteriormente, los resultados de las diferentes simulaciones de la interacción entre proteínas deben ser puntos de partida para nuevas experimentaciones. Para esto los métodos utilizados para las predicciones se deben validar extensamente. Para la evaluación de los métodos existentes, CAPRI (*Critical Assessment of PRedicted Interactions*) provee los complejos objetivo con los cuales se evalúa la capacidad de los diferentes métodos para predecir las interacciones [37]. Las predicciones de la interacción proteína-proteína de los diferentes métodos son comparados con la estructura del complejo proteico determinada por medio de la Cristalografía de Rayos X.

Debido a la dificultad de encontrar complejos apropiados para la evaluación de los métodos, CAPRI a diferencia de CASP (*Critical Assessment of Techniques for Protein Structure Prediction*) empieza cuando un investigador suministra un complejo objetivo adecuado.

Las medidas de desempeño para las predicciones son:

• Fracción de contactos nativos: número de contactos residuo-residuo en la predicción dividido sobre el número de contactos en el complejo objetivo [19][38].

• Fracción de contactos no nativos: número de contactos residuo-residuo no nativos, es decir, aquellos contactos que no se encuentran en el complejo nativo, dividido por el número total de contactos en dicho complejo [19][38].
• RMSD del ligando: RMSD (Root Mean Square Deviation) del esqueleto del ligando en el complejo predicho vs. el ligando del complejo objetivo [19][37][38].
• RMSD del sitio de unión: RMSD calculado para los residuos del ligando que están en contacto con el receptor [2][37][38].
• Choques atómicos: existe un umbral para los choques atómicos permitidos para las predicciones de determinado complejo. Se considera que hay un choque atómico cuando hay contacto entre dos átomos diferentes al hidrógeno separados por 3 Armstrong [38].

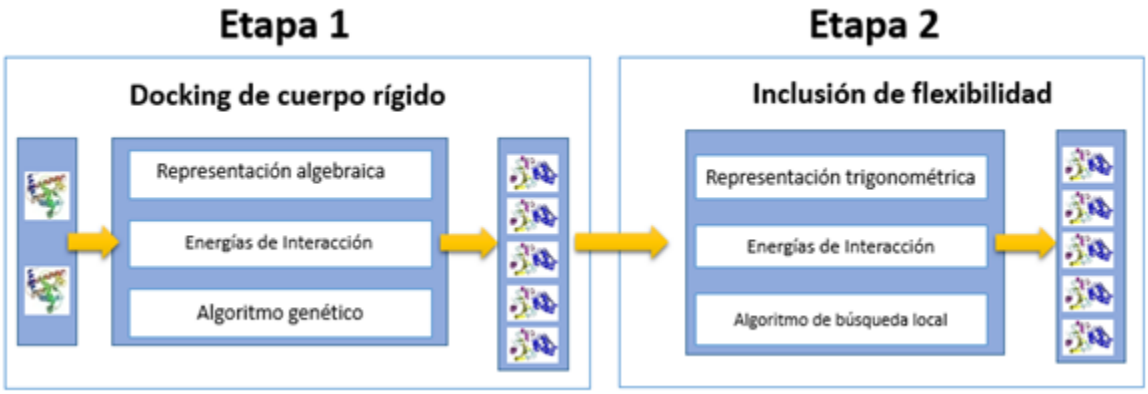
El problema de la interacción proteína-proteína cuenta con un conjunto de datos que puede ser utilizado para comparar los diferentes métodos que solucionan el problema. Este conjunto de datos puede ser descargado de la siguiente página web: http://zlab.umassmed.edu/benchmark/ [39].

3. Modelo propuesto para la predicción de la interacción proteína-proteína

En este capítulo se presentan el modelo y la metodología propuesta para la búsqueda de conformaciones estables de interacción proteína-proteína. Los métodos de inteligencia computacional que se utilizan para el desarrollo del modelo propuesto son algoritmos genéticos y algoritmos de búsqueda local.

En la Figura 3-1 se ilustra el esquema general del modelo propuesto. Este se compone de dos etapas: la primera etapa recibe como entrada dos proteínas que se supone que interactúan. Las proteínas son consideradas como cuerpos rígidos y se tiene como objetivo explorar el espacio de búsqueda obteniendo una solución o un conjunto de soluciones. En la segunda etapa las entradas son los complejos proteicos obtenidos de la primera etapa, a los cuales se les realiza un refinamiento, que consiste en la inclusión de cierto grado de flexibilidad en el ligando.

Figura 3-1. Esquema general del modelo propuesto para realizar la búsqueda de conformaciones estables de interacción proteína-proteína.



A continuación se presentan en detalle las dos etapas del modelo propuesto.

3.1 Interacción proteína-proteína de cuerpo rígido

Para predecir cómo interactúan dos proteínas, el primer enfoque desarrollado considera las proteínas como cuerpos rígidos. La principal razón para considerar las proteínas como cuerpos rígidos es el tamaño del espacio de búsqueda (espacio de posibles conformaciones o complejos que se pueden formar), el cual puede ser muy grande dependiendo del tamaño de las proteínas, y aún más, si se considera flexibilidad en las estructuras. En esta primera etapa, el objetivo es realizar una exploración primaria del espacio de conformaciones, cubriendo la mayor parte de este.

El desarrollo de esta etapa contempla los tres componentes que tienen los métodos existentes de *docking* proteína-proteína: representación computacional de las proteínas, algoritmo de búsqueda de soluciones y la selección de una función de puntuación.

El algoritmo de búsqueda utilizado es un algoritmo genético (AG). Los AG son algoritmos inspirados en la evolución biológica. Comienzan con un conjunto de estados generados aleatoriamente, llamados población. Cada estado o individuo está representado a través de su información genética codificada en su cromosoma. Un cromosoma, en términos de los AG, codifica una solución a un problema. Los cromosomas computacionalmente pueden ser vistos, por ejemplo, como una cadena sobre un alfabeto (por ejemplo, 1 y 0) [12][40].

Para aplicar un algoritmo genético se requieren cinco componentes [40]:

- Una representación de las soluciones potenciales del problema.
- Una forma de crear una población inicial de posibles soluciones (usualmente un proceso aleatorio).
- Una función de evaluación, que permita comparar las soluciones en términos de su aptitud.
- Operadores genéticos que alteren la información genética de los hijos que se producirán para las siguientes generaciones, y

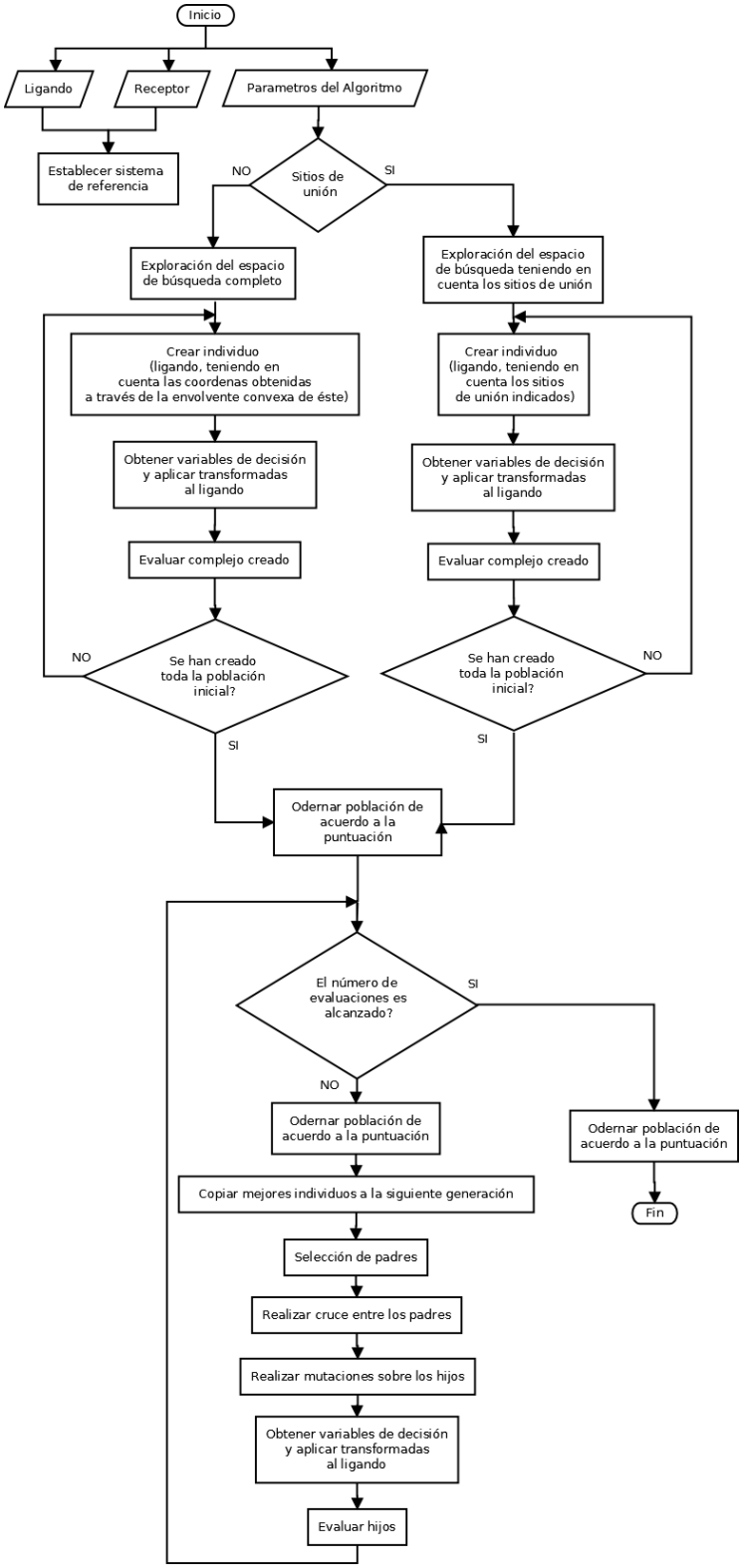
- Valores de los diferentes parámetros que utiliza el algoritmo genético (tamaño de la población, probabilidad de cruce, probabilidad de mutación, número máximo de generaciones, entre otros).

Para implementar el algoritmo genético propuesto se usó el paquete de algoritmos de optimización jMetal [41][42], el cual es un software de código abierto y puede ser descargado de la dirección web http://sourceforge.net/projects/jmetal/. jMetal está escrito en Java y no tiene dependencias de otros programas o paquetes externos.

La arquitectura de jMetal permite implementar un conjunto de algoritmos de optimización, los cuales solucionan problemas a través de la generación de un conjunto de soluciones y un conjunto de operadores. En términos de algoritmos evolutivos, los conjuntos de soluciones son las poblaciones y las soluciones son los individuos. Para el problema de la interacción proteína-proteína, las poblaciones son un conjunto de complejos proteína-proteína y los individuos son complejos individuales.

El algoritmo genético utilizado en esta etapa se presenta en el diagrama de flujo que se muestra en la Figura 3.2.

Figura 3-2. Diagrama de flujo de la primera etapa: *docking* de cuerpo rígido.

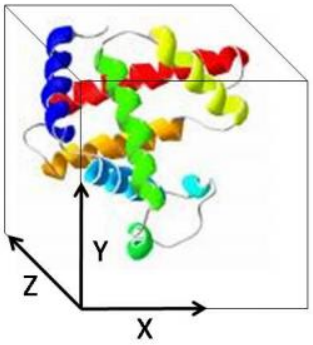


3.1.1 Sistema de representación de las proteínas

Para el desarrollo de este trabajo se requiere que las proteínas estén en formato PDB, en el cual se provee la información de la estructura tridimensional de las proteínas. Para la ejecución del algoritmo, primero se hace un pre-procesamiento de los archivos de las proteínas que se reciban como entrada, teniendo en cuenta únicamente los átomos de los aminoácidos estándar.

Al considerar las proteínas como cuerpos rígidos, el sistema de representación computacional de las proteínas es una representación algebraica, la cual representa la estructura de las proteínas a través de las coordenadas cartesianas. Es decir, representa las posiciones (x, y, z) de cada uno de los átomos de la proteína en el espacio tridimensional [43]. La representación algebraica se muestra en la Figura 3-3.

Figura 3-3. Representación algebraica de las proteínas.



Representación algebraica de la estructura de las proteínas (imagen tomada de [43]).

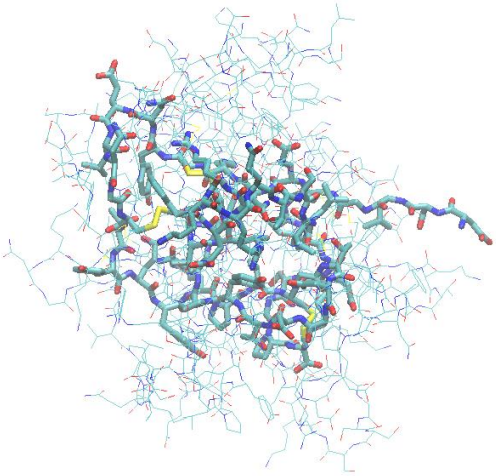
La representación algebraica de la estructura de las proteínas fue seleccionada tomando en consideración que en esta etapa no se van a tener en cuenta cambios conformacionales en la estructura de las proteínas.

Una vez se leen las dos proteínas, la información de los átomos y sus coordenadas es almacenada en una estructura de datos computacional. Con la información obtenida de las proteínas, se define cuál es el receptor y cuál es el ligando con base en el número de átomos que tengan. La proteína con mayor número de átomos se considera el receptor y la otra proteína el ligando.

Establecer cuál de las dos proteínas es el receptor es de gran importancia, ya que de esta manera se define el sistema de referencia para la ejecución del algoritmo. El origen del sistema de referencia es establecido como el centroide o centro geométrico del receptor. El centroide se define como el punto medio de las posiciones de los átomos en el espacio.

Como se ilustra en la Figura 3-4, el centroide del ligando se traslada al centroide del receptor. Durante la ejecución del algoritmo se cambian la posición y la orientación del ligando con respecto al receptor

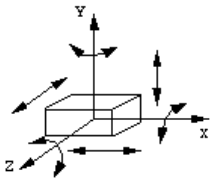
Figura 3-4. Receptor y ligando en el mismo sistema de referencia.



3.1.2 Algoritmo de búsqueda

Considerando las proteínas como cuerpos rígidos, estas no presentan cambios conformacionales durante el proceso de búsqueda, es decir, su estructura tridimensional permanece igual. Teniendo en cuenta lo anterior, el problema tiene seis variables de decisión o grados de libertad, correspondientes a los grados de libertad de un cuerpo rígido, tres relacionados con las traslaciones (posición) y tres relacionados con las rotaciones (orientación). Los grados de libertad de un cuerpo rígido se presentan en la Figura 3-5.

Figura 3-5. Grados de libertad de un cuerpo rígido.



Tres grados de libertad para las traslaciones sobre los ejes X, Y y Z y tres para las rotaciones sobre estos ejes (imagen tomada de [44]).

Representación de las soluciones

Las soluciones se codifican con base en los grados de libertad del problema. El cromosoma que se utiliza es de seis genes y la codificación es real. Utilizando la codificación real, el cromosoma es un vector de números reales. La representación del cromosoma se presenta en la Figura 3-6.

Figura 3-6. Cromosoma de los individuos del algoritmo genético propuesto.

1	2	3	4	5	6
x	y	z	α	ω	β

Gen	Codificación
1	Traslación en el eje x
2	Traslación en el eje y
3	Traslación en el eje z
4	Ángulo de rotación con respecto al eje x
5	Ángulo de rotación con respecto al eje y
6	Ángulo de rotación con respecto al eje z

Los genes 1, 2 y 3 hacen referencia a la posición del ligando con respecto al receptor, y los genes 4, 5 y 6 hacen referencia a la orientación del ligando con respecto al receptor.

Como se mencionó anteriormente, en cada iteración del algoritmo genético, el ligando va a cambiar su posición y orientación con respecto al receptor. Los valores que puede tomar cada gen son de -10 a 10 Angstrom para las traslaciones (genes 1, 2 y 3) y de -180 a 180 grados para las rotaciones (genes 4, 5 y 6).

Los movimientos del ligando alrededor del receptor se ejecutan computacionalmente a través de las transformaciones de los cuerpos rígidos, las cuales se explican a continuación.

Transformaciones de las proteínas

Una transformación de un cuerpo rígido es una función $h: A \rightarrow W$, que asocia a cada punto de A uno en W, con la restricción de que la distancia entre cualquier par de puntos de A debe ser preservada [45].

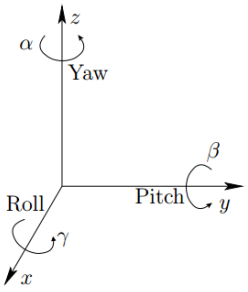
La traslación de un cuerpo rígido se realiza a través de tres parámetros, los cuales son los valores a trasladar en $X, Y\ y\ Z$. La función de traslación se presenta en la Ecuación 3-1.

$$(\mathrm{x}, \mathrm{y}, \mathrm{z}) = (\mathrm{x} + \mathrm{x_t}, \mathrm{y} + \mathrm{y_t}, \mathrm{z} + \mathrm{z_t}) \quad (3\text{-}1)$$

Con la función de traslación cada punto $(\mathrm{x_i}, \mathrm{y_i}, \mathrm{z_i})$ es reemplazado por $(\mathrm{x_i} + \mathrm{x_t}, \mathrm{y_i} + y_t, z_i + z_t)$.

Un cuerpo rígido puede ser rotado sobre tres ejes ortogonales como se ilustra en la Figura 3-7.

Figura 3-7. Rotación 3D.



Cualquier rotación tridimensional puede ser descrita como una secuencia de rotaciones sobre los ejes X, Y, Z (imagen tomada de [45]).

La rotación sobre el eje X está dada por la matriz de rotación presentada en la Ecuación 3-2 [45]:

$$R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{pmatrix} \quad (3\text{-}2)$$

La rotación sobre el eje Y está dada por la matriz de rotación presentada en la Ecuación 3-3 [45]:

$$R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \quad (3\text{-}3)$$

La rotación sobre el eje Z está dada por la matriz de rotación presentada en la Ecuación 3.4 [45]:

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3\text{-}4)$$

Las rotaciones sobre los ejes X, Y y Z pueden ser utilizadas para colocar un cuerpo rígido en cualquier orientación. Se pueden realizar diferentes movimientos sobre los ejes en una sola matriz de rotación multiplicando las diferentes matrices de rotación. Se debe tener en cuenta el orden en que se multiplican las matrices, ya que el cambio de orden implica un cambio en el resultado obtenido. De igual manera se pueden realizar rotaciones y luego traslaciones para ubicar un cuerpo rígido en cualquier posición y orientación. La matriz de transformación para una rotación $R(\alpha, \beta, \gamma)$ seguida de una traslación dada por x_t, y_t, z_t se presenta en la Ecuación 3-5 [45].

$$T = \begin{pmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma & x_t \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & y_t \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma & z_t \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3\text{-}5)$$

Los movimientos que puede realizar el ligando sobre el receptor están definidos por las posibles combinaciones de la multiplicación de las matrices de rotación y traslación. Por lo tanto, en cada iteración el ligando puede obtener una nueva posición y orientación con respecto al receptor.

Una vez definidos el marco de referencia de las proteínas y la forma en la que el ligando se va a mover alrededor del receptor, se comienza con la ejecución del algoritmo genético.

Generación de la población inicial

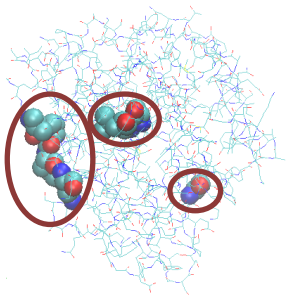
Para la generación de la población inicial se tienen dos enfoques que se basan en si se tiene el conocimiento de los sitios activos o sitios de unión del receptor o no. Para predecir los sitios de unión de las proteínas se utilizó el algoritmo BSpred [46]**,** el cual se puede descargar o utilizar a través de un servicio web en la siguiente dirección: http://zhanglab.ccmb.med.umich.edu/BSpred/.

Para utilizar BSpred se debe tener la secuencia de aminoácidos de las proteínas, las cuales se obtienen a partir del archivo PDB de estas.

El algoritmo BSpred da como resultado los potenciales residuos de interfaz de la proteína objetivo. Los residuos de interfaz del receptor son los residuos que van a interactuar con el ligando. Específicamente, BSpred indica el número del residuo, el tipo del residuo y un puntaje de confianza, el cual, entre más alto es mejor.

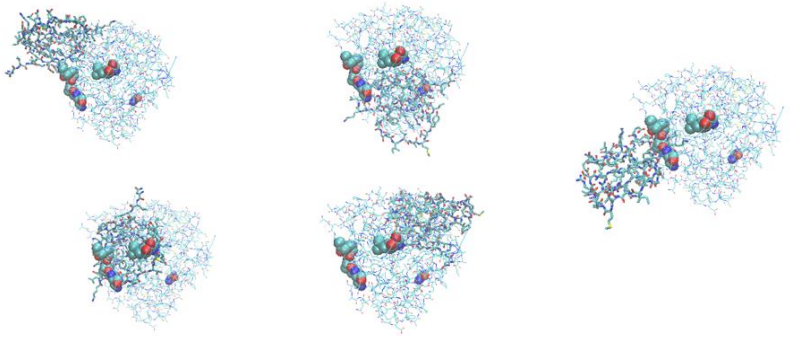
Con la información de los sitios de unión de las proteínas se busca que los individuos de la primera generación estén ubicados cerca de un potencial sitio de unión y limitar la búsqueda en las zonas del espacio de conformaciones donde se encuentran dichos sitios. En la Figura 3-8 se observa una proteína y sus potenciales sitios de unión y en la Figura 3-9 se presenta un ejemplo de generación inicial creada según la información de los sitios de unión.

Figura 3-8. Predicción de los potenciales sitios de unión.



Predicción de los sitios de unión para la proteína 1UBN realizada con BSpred.

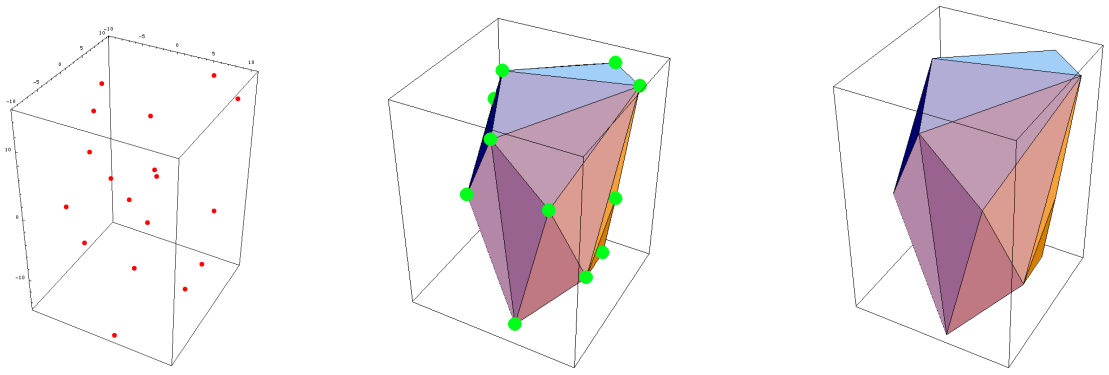
Figura 3-9. Población inicial generada con información de potenciales sitios de unión



Generación inicial para el complejo 2SNI teniendo en cuenta los sitios de unión.

Si no se tiene la información de los sitios de unión, la población inicial se genera teniendo en cuenta la envolvente convexa de los átomos del receptor. La envolvente convexa para un conjunto de puntos se puede ver como un subconjunto de puntos que en su interior contengan a todo el conjunto. En la Figura 3-10 se presenta un ejemplo de envolvente convexa. Para calcular la envolvente convexa de las proteínas se utiliza el paquete de software llamado QuickHull3D [47], disponible en la dirección web http://www.cs.ubc.ca/~lloyd/java/quickhull3d.html.

Figura 3-10. Envolvente convexa de un conjunto de puntos en el espacio.

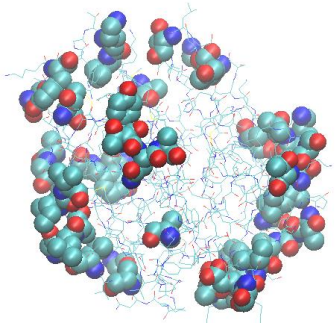


Conjunto de puntos Envolvente convexa con vértices resaltados Envolvente convexa

(Imágenes tomadas de [48])

Teniendo en cuenta que la representación algebraica de la estructura de las proteínas consiste en la posición (x, y, z) de cada átomo de éstas, se puede calcular la envolvente convexa para el conjunto de átomos de las proteínas. La Figura 3-11 presenta un ejemplo de esto.

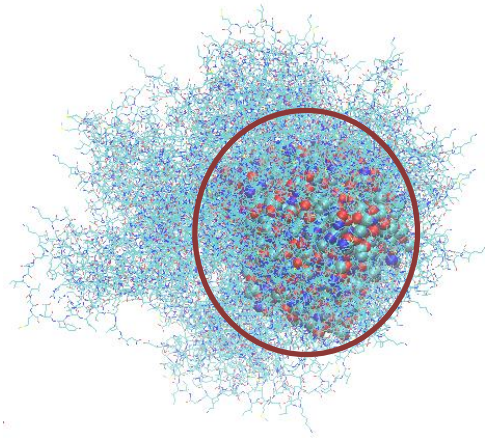
Figura 3-11. Átomos de una proteína que pertenecen a su envolvente convexa.



Vértices de la envolvente convexa para la proteína 1UBN.

De la misma manera que con la información de los sitios de unión, conociendo los átomos que pertenecen a la envolvente convexa se genera la población inicial. El objetivo de generar la población inicial con la información de la envolvente convexa es cubrir la mayor parte del espacio de búsqueda. En la Figura 3-12 se presenta un ejemplo de generación de población inicial teniendo en cuenta la información de la envolvente convexa; en este ejemplo se ve como alrededor de todo el receptor se ubicó el ligando para comenzar con la exploración del espacio de búsqueda.

Figura 3-12. Población inicial generada a partir de la envolvente convexa del receptor.



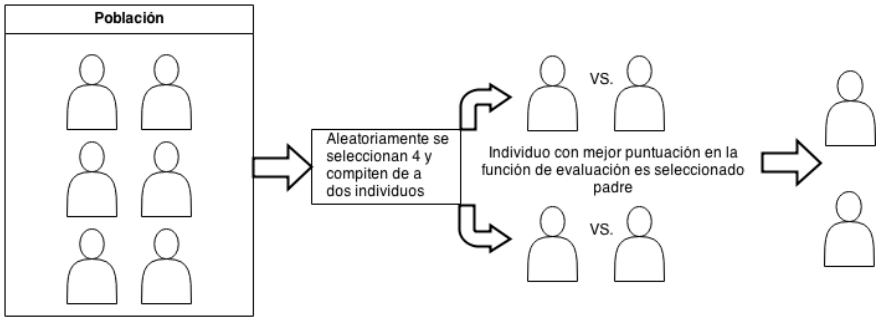
Generación inicial para el complejo 2SNI teniendo en cuenta los vértices de la envolvente convexa.

Operadores Genéticos

Los operadores genéticos utilizados son: selección, cruce y mutación. El operador de selección se utiliza para seleccionar los padres que serán cruzados en cada generación; los individuos con mejor valor en la función de puntuación tienen mayor probabilidad de ser seleccionados. El operador de cruce simula la reproducción sexual, combina dos cromosomas para generar nuevos individuos que comparten características de los padres. El operador de mutación es un operador que modifica el cromosoma de un solo individuo.

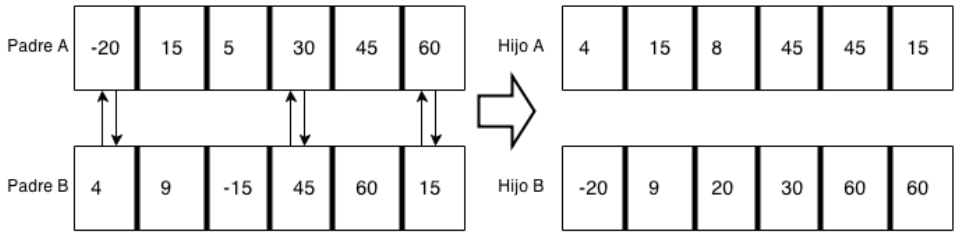
El operador de selección utilizado es el operador de selección por torneo. Con este operador, los padres son seleccionados de la siguiente manera: se toman dos individuos de la población por cada padre que se vaya a seleccionar y se ponen a competir entre sí; el individuo con mejor función de evaluación es el que queda seleccionado como padre en cada uno de los casos. Lo anterior se puede ver en la Figura 3.13.

Figura 3-13. Ejemplo del operador de selección.



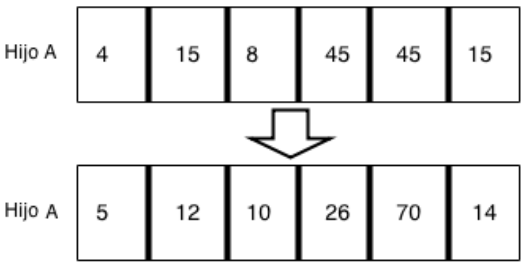
Una vez seleccionados los padres se ejecuta el operador de cruce. Para el operador de cruce se tiene en cuenta el cromosoma definido para el problema. De acuerdo con la probabilidad de cruce se asignan nuevos valores a cada gen o se intercambian los valores entre los padres o finalmente permanecen igual que los padres. Lo anterior se puede ver en la Figura 3-14.

Figura 3-14. Ejemplo del operador de cruce.



El operador de mutación se ejecuta sobre los hijos o individuos resultantes del operador de cruce. Dependiendo de la probabilidad de cruce se cambian los valores del cromosoma de cada hijo. Lo anterior se puede ver en la Figura 3-15.

Figura 3-15. Ejemplo del operador de mutación.



En cada generación una proporción de la población ordenada de mayor a menor de acuerdo con el valor de la función de evaluación es copiada a la siguiente generación.

Criterio de parada

El criterio de parada para el algoritmo genético utilizado es el número máximo de evaluaciones.

3.1.3 Función de evaluación

El problema de la interacción proteína-proteína puede ser visto como un problema de optimización en donde el objetivo es minimizar la energía libre de unión durante la formación del complejo proteico. Teniendo en cuenta esto, durante le ejecución del algoritmo se calculó la energía potencial del complejo en formación. El cálculo de esta energía se realiza en función de las coordenadas atómicas de la estructura de las

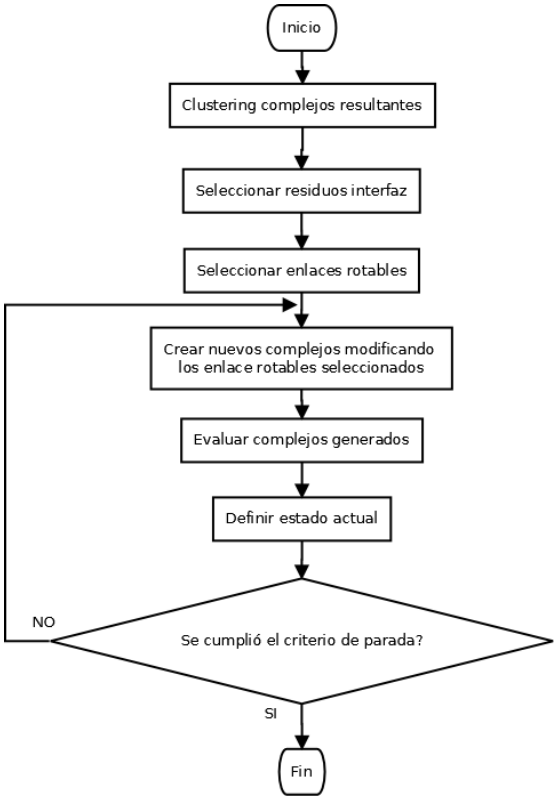
proteínas. Para el cálculo de la energía potencial, el campo de fuerza utilizado en la ejecución del algoritmo propuesto es Amber, la cual contiene la información relacionada con los diferentes tipos de átomos que componen una proteína.

Para calcular la energía libre de unión se utilizó el paquete de diseño molecular TINKER [49]. Para realizar el cálculo de la energía, la información de la proteína que se tiene en formato PDB debe ser convertida al formato XYZ.

3.2 Inclusión de flexibilidad

La entrada a la etapa de la inclusión de flexibilidad la constituyen los individuos de la última generación del algoritmo genético. El algoritmo para esta etapa se presenta en la Figura 3-16.

Figura 3-16. Diagrama de flujo de la etapa de inclusión de flexibilidad



El primer paso en la etapa de inclusión de flexibilidad consiste en seleccionar los ligandos con los cuales se va a trabajar. Para realizar esta selección, las estructuras resultantes o soluciones obtenidas en la primera etapa se agrupan de acuerdo con el valor del RMSD, realizando la selección de las estructuras según el valor de energía del complejo que forman.

Los grados de libertad del problema que se presenta en esta etapa dependen del tamaño de las proteínas y del número de enlaces rotables de los residuos que se consideran de interfaz. Los residuos de interfaz, son los residuos del ligando que interactúan con el receptor. Los residuos de interfaz del ligando se seleccionan de acuerdo con la distancia entre los residuos del ligando con los residuos del receptor. Si un átomo de un residuo del ligando se encuentra a menos de 10 Å de distancia de un átomo del receptor, se considera que el residuo al que pertenece el átomo pertenece a la interfaz del complejo proteína-proteína formado.

La selección de los enlaces rotables se realiza de acuerdo con el grado de flexibilidad que se quiera incluir en el ligando. Si únicamente se va a incluir flexibilidad en el esqueleto de la proteína, se tienen en cuenta los enlaces φ y ψ de los residuos presentes en la interfaz. Si se va a incluir flexibilidad en las cadenas laterales del ligando, la selección de los enlaces rotables (χ_i) depende del tipo de residuo. El número de ángulos χ que puede tener cada residuo se presenta en la Tabla 3-1 [50].

Tabla 3-1. Número de ángulos χ por tipo de residuo

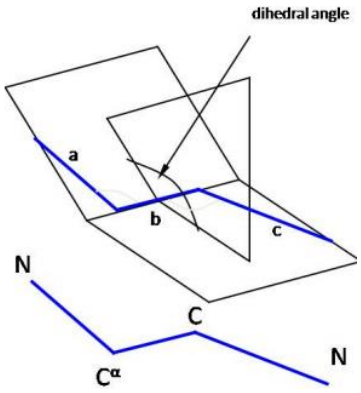
Residuo	**Ángulos χ**
GLI, ALA, PRO	Cadena principal
SER, CYS, THR, VAL	χ_1
ILE, LEU, ASP, ASN, HIS, PHE, TYR, TRP	χ_1, χ_2
MET, GLU, GLN	χ_1, χ_2, χ_3
LYS, ARG	$\chi_1, \chi_2, \chi_3, \chi_4$

3.2.1 Representación de las proteínas

La representación de las proteínas en esta etapa es la representación trigonométrica, en la que se tienen en cuenta los ángulos de torsión de las mismas. La representación trigonométrica se basa en que las distancias entre los átomos y los ángulos que forman los enlaces atómicos pueden determinar la estructura de una proteína [43].

El objetivo de utilizar la representación trigonométrica de las proteínas es calcular los cambios conformacionales en el esqueleto de las proteínas y en las cadenas laterales de estas. En la figura 3-17 se muestra la representación trigonométrica de las proteínas.

Figura 3-17. Representación trigonométrica de las proteínas



(imagen tomada de [43])

Computacionalmente la representación trigonométrica se trabaja a través de las coordenadas internas de las proteínas, las cuales tienen en su sintaxis la siguiente información:

$$Etiqueta, átomo\ 1, tamaño\ enlace, átomo\ 2, ángulo\ del\ enlace, átomo\ 3, ángulo\ dihedral$$

Las soluciones se codifican con base en los grados de libertad del problema. En términos de algoritmos evolutivos, el cromosoma que se utiliza depende del número de residuos de interfaz del ligando y del número de ángulos de torsión de estos. Utilizando la codificación real el cromosoma es un vector de números reales. La representación de las soluciones se presenta en la figura 3-18.

Figura 3-18. Codificación de las soluciones para el algoritmo de búsqueda local.

Φ	ψ	χ_1	χ_2	χ_3	χ_4	Φ	ψ	χ_1	…
Valor para el residuo 1	Valor para el residuo 1	Valor para el residuo 1	Valor para el residuo 1	Valor para el residuo 1	Valor para el residuo 1	Valor para el residuo 2	Valor para el residuo 2	Valor para el residuo 2	…

Teniendo el vector con el valor de los ángulos rotables para cada residuo, estos valores son actualizados en la matriz de coordenadas internas de las proteínas. Con las coordenadas internas actualizadas se recalculan las coordenadas atómicas de las proteínas, al hacer la conversión de la representación trigonométrica a la representación algebraica de las mismas.

La matriz de coordenadas internas se obtiene utilizando el programa *xyzint* de TINKER [49] y la conversión de la representación trigonométrica a la representación algebraica de las proteínas se realiza con el programa *intxyz* de TINKER [49].

3.2.2 Algoritmo de búsqueda local

El algoritmo de búsqueda usado en esta etapa es un algoritmo de búsqueda local. El objetivo es realizar transformaciones a los enlaces rotables de la proteína a la que se le está incluyendo flexibilidad, obteniendo de esa manera nuevas soluciones.

El algoritmo utilizado se basa en el algoritmo *de Metropolis y* consiste en crear un grafo dirigido cuyos nodos son conformaciones de un complejo y las aristas entre dos nodos representan la probabilidad de pasar de una conformación a otra [28].

En la fase de construcción del grafo se generan conformaciones aleatorias a partir del complejo objetivo. Las transformaciones a los enlaces rotables se realizan a través del operador de mutación y los valores que pueden adoptar los enlaces rotables dependen del tipo de residuo. Para determinar los valores que se puede rotar un enlace, se tiene en cuenta la librería de rotámeros que se presenta en la Tabla 3-2. La librería de rotámeros indica los valores que pueden tomar los ángulos de torsión de cada aminoácido.

Tabla 3-2. Librería de rotámeros utilizada en el modelo propuesto

Residuo	χ_1	χ_2	χ_3	χ_4
ARG	[-177°,62°]	[-167°,180°]	[-65°,180°]	[-175°,180°]
LYS	[-177°,62°]	[-68°,180°]	[-68°,180°]	[-65°,180°]
MET	[-177°,62°]	[-65°,180°]	[-75°,180°]	N/A
GLU	[-177°,70°]	[-80°,180°]	[-60°,60°]	N/A
GLN	[-177°,70°]	[-75°,180°]	[-100°,100°]	N/A
ASP	[-177°,62°]	[-60°,65°]	N/A	N/A
ASN	[-177°,62°]	[-80°,120°]	N/A	N/A
ILE	[-177°,62°]	[-60°,170°]	N/A	N/A
LEU	[-177°,62°]	[65°,175°]	N/A	N/A
HIS	[-177°,62°]	[-165°,165°]	N/A	N/A
TRP	[-177°,62°]	[-105°,95°]	N/A	N/A
TYR	[-177°,62°]	[-85°, 90°]	N/A	N/A
PHE	[-177°,62°]	[-85°,90°]	N/A	N/A
PRO	[-30°,30°]	N/A	N/A	N/A
THR	[-177°, 62°]	N/A	N/A	N/A
VAL	[-60°, 175°]	N/A	N/A	N/A
SER	[-177°,62°]	N/A	N/A	N/A
CYS	[-177°, 62°]	N/A	N/A	N/A

La elaboración de la Tabla 3-2 fue realizada en el marco de investigaciones previas del Grupo de Investigación LISI [50]. Dicha tabla se basa en las librerías de rotámeros reportadas en [51] y [52].

Operador

El operador utilizado en el algoritmo de búsqueda local es el de mutación, el cual como se dijo anteriormente, es un operador que modifica el cromosoma de un solo individuo.

Rondas de mejora

Las rondas de mejora hacen referencia a la fase de construcción del grafo del algoritmo de búsqueda local. Por cada individuo de la población y en cada iteración, las rondas de mejora corresponden al número de nodos del grafo. Cada solución o nodo del grafo es

creado a partir de la aplicación del operador de mutación sobre el individuo con el que se está trabajando.

Cuando se generan las nuevas conformaciones del complejo, a través de la función de evaluación se determina cuál es el nuevo estado (la nueva conformación). Si las conformaciones que se generaron son mejores que la actual, se va a realizar un cambio de estado por la conformación que menor energía tenga; en caso de que ninguna de las conformaciones generadas sea mejor, se generan nuevamente conformaciones aleatorias y se evalúa la probabilidad de cambiar de un estado a otro.

Criterio de parada

El criterio de parada definido para esta etapa corresponde al número máximo de evaluaciones.

3.2.3 Función de evaluación

La función de evaluación en esta etapa mide la probabilidad de transición de un estado a otro y se presenta a continuación [28]:

$$A = \begin{cases} \exp(-\Delta E/k_B T) & si\ \ \varepsilon'/\varepsilon\ < 1 \\ 1 & en\ otro\ caso \end{cases}$$

$$\varepsilon'/\varepsilon\ = \exp(-\Delta E/k_B T)$$

$$\Delta E = E(q') - E(q)$$

Si el cambio en la conformación disminuye la energía, la probabilidad de pasar de un estado a otro siempre es aceptada, en otro caso, es aceptada con una probabilidad $\exp(-\Delta E/k_B T)$. Las diferencias de energía se calculan con TINKER.

4. Implementación paralela del modelo propuesto

El objetivo de utilizar la computación paralela en este trabajo es disminuir el tiempo de ejecución del modelo propuesto, teniendo en cuenta el costo computacional del problema que se está abordando. Específicamente se implementó un sistema paralelo MIMD, el cual para la primera etapa del modelo propuesto se ejecuta en un sistema PM (*Parallel Machines*) y para la segunda etapa en un sistema LC (*Local Clusters*).

La implementación del sistema paralelo PM (arquitectura sobre computadores con múltiples procesadores) de la primera etapa del modelo propuesto, tuvo por objetivo realizar una prueba de concepto sobre la forma de paralelizar los algoritmos de búsqueda utilizados.

Una vez realizada la implementación paralela del algoritmo genético utilizado en la primera etapa, se realizó la implementación paralela del algoritmo de búsqueda local sobre una arquitectura distribuida, es decir, ejecutándola sobre un *cluster* computacional y no sobre una sola máquina con múltiples procesadores.

A continuación se detallan el diseño y las herramientas utilizadas en la implementación propuesta.

4.1 Implementación paralela del algoritmo genético

La implementación paralela del AG aprovecha los múltiples procesadores con los que cuentan las estaciones de trabajo hoy en día. Para el caso del AG utilizado se implementó un algoritmo paralelo MIMD, el cual se basa en la ejecución de múltiples instrucciones sobre múltiples datos.

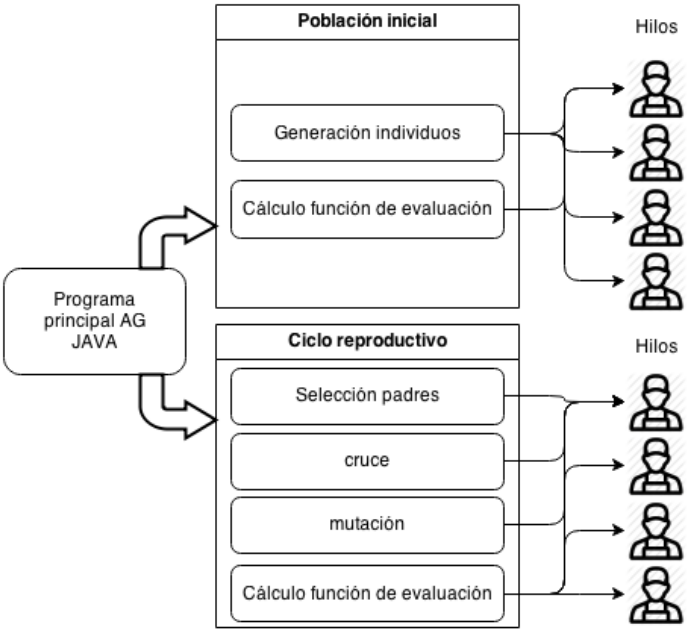
Como el lenguaje de desarrollo utilizado para este trabajo fue Java, se utilizó la librería Java Concurrent, la cual facilita la escritura de programas concurrentes. La concurrencia

hace relación a la capacidad de ejecutar varios programas o varias partes de un programa al mismo tiempo, mejorando el rendimiento del mismo [53].

A nivel de sistema operativo, un proceso se ejecuta a la vez e independiente de otros procesos, sin acceder directamente a datos compartidos con otros procesos. Un hilo se ejecuta dentro de un proceso y puede acceder a una memoria compartida por otros hilos que se encuentren dentro del mismo proceso. Las aplicaciones Java se ejecutan en un solo proceso y pueden manejar múltiples hilos para lograr un procesamiento paralelo [53].

En la Figura 4-1 se presenta el diseño del AG paralelo, en el cual el programa principal puede ser visto como un maestro que envía trabajos, y los hilos son vistos como trabajadores, los cuales ejecutan los trabajos enviados por el maestro.

Figura 4-1. Partes del AG paralelizadas



En Java, para el desarrollo de programas concurrentes o paralelos, se tiene la opción de crear un conjunto de hilos (*pool* de hilos) fijo, los cuales se encargan de ejecutar las funciones establecidas o paralelizadas. En esta implementación del AG se utilizaron los objetos Java *Future* y *Callable*, los cuales permiten el envío de parámetros a los hilos y estos devuelvan los resultados al programa principal.

El AG utilizado se paraleliza en dos partes fundamentales: la primera corresponde a la generación de la población inicial, en donde cada hilo de ejecución crea un nuevo individuo y calcula su función de evaluación, y la segunda corresponde al ciclo reproductivo del AG, en donde cada hilo realiza la selección de padres, el cruce, la mutación y el cálculo de la función de evaluación para los respectivos individuos. Esto indica, que en la práctica, dependiendo del número de procesadores disponibles (n), el tiempo de ejecución del algoritmo deber ser aproximadamente n veces más rápido que la versión secuencial.

4.2 Implementación distribuida de la etapa de inclusión de flexibilidad

Un *cluster* es un tipo de sistema de procesamiento en paralelo o distribuido, formado por un conjunto de computadores autónomos interconectados que trabajan en forma cooperativa como si formaran un único recurso computacional [54].

El software Condor es un sistema de administración de cargas de trabajo intensivo que crea un entorno de computación de alto rendimiento (HTC - High-Throughput Computing). Se caracteriza por utilizar los recursos de cómputo de los nodos pertenecientes a una red de computadores. Es usado para distribuir y ejecutar tareas que demandan altos recursos computacionales en una o más máquinas pertenecientes a un *cluster* de computadores, y también puede realizar tareas secuenciales o en paralelo [55].

Para entender el funcionamiento del sistema Condor es necesario explicar algunos conceptos relacionados con este software [54]:

- Condor pool: es un número arbitrario de máquinas, que pueden tener diversas arquitecturas y sistemas operativos que se conectan a una red. Es decir, es una colección de recursos y de peticiones de recursos.
- Administrador de Condor: es el usuario que configura la red y verifica el funcionamiento del *Condor pool* y de los procesos que están ejecutándose en todas las máquinas o computadores del *pool*.

- Job: Un trabajo en Condor es una tarea que es enviada por un usuario al *cluster* de computadores para que sea ejecutada.

En el sistema Condor las máquinas del pool se caracterizan por tener varios roles o tipos de funcionamiento. El comportamiento de una máquina en el *pool* estará de acuerdo con los roles que se le definan; estas pueden tener varios roles simultáneamente [54]. Los roles que se presentan en un *Condor pool* son los siguientes:

Central Manager: es la máquina encargada de gestionar los recursos del *Condor pool* con el fin de asignar las tareas que se envían y, de este modo, ejecutarlas en máquinas destinadas a esa función. Esta máquina se encarga de recolectar la información de disponibilidad de las máquinas donde se ejecutan los trabajos; del mismo modo, se encarga de negociar los recursos que hay en el *pool* y las peticiones de recursos que se generan por parte de las máquinas que forman parte del mismo [54].

Worker Node - Execute Machine: es una máquina capaz de ejecutar tareas enviadas al *pool*.

Universo Condor: un universo en Condor define un ambiente de ejecución. Entre ellos se encuentran: *Standard, Vanilla y* Java*,* entre otros. El universo bajo el que un trabajo es ejecutado se especifica en el archivo de envío. Si el universo no es especificado, por defecto se ejecuta en el universo *Standard,* el cual está en un ambiente para programas desarrollados en el lenguaje de programación *C.*

Un trabajo en *Condor* se caracteriza por tener varios elementos que lo diferencian de otros, ellos son [55]:

- El ID es el identificador del trabajo y está formado por dos números:
    - El número antes del punto representa el *cluster*. Un *cluster* es el conjunto de trabajos creado en un envío.
    - El número después del punto representa el trabajo dentro del *cluster*.
- El usuario que envió el (los) trabajo(s).
- La fecha del envío.
- El tiempo de ejecución.
- El estado actual del trabajo. Algunos valores posibles son:

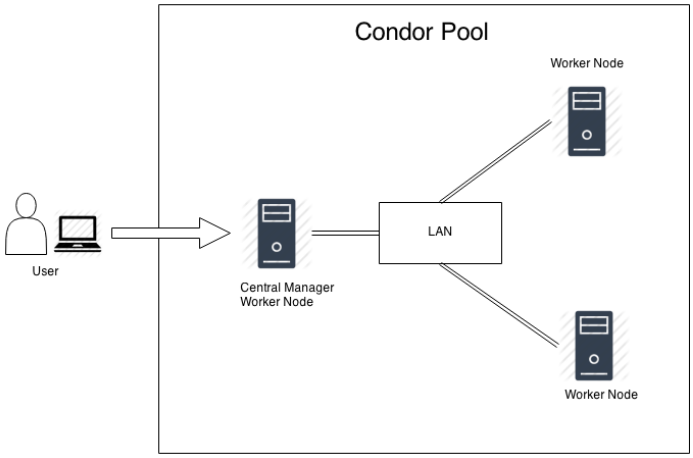
- **I:** No se está ejecutando porque aún no se le ha asignado a alguna máquina (IDLE).
- **R:** Ejecutándose actualmente (RUNNING).
- **H:** El trabajo no se está ejecutando por requerimiento del propietario (HOLD).
- La prioridad del trabajo que ha especificado el usuario.
- El tamaño de la imagen del trabajo en megabytes.
- Nombre del ejecutable.

Los pasos para ejecutar un trabajo en Condor son los siguientes [54]:

1. Un usuario envía el trabajo a Condor especificando los parámetros de ejecución de este en un archivo de configuración. En este archivo se indican las entradas, el archivo de salida, el archivo donde se imprimen los posibles errores de ejecución, el archivo de registro de eventos, los requisitos de sistema operativo, el universo Condor que se va a utilizar y la ruta del directorio del programa a ejecutar.
2. El trabajo es añadido a la cola de trabajos de Condor.
3. El *Central Manager* busca una máquina que cumpla con los requisitos (memoria, sistema operativo, arquitectura del sistema, etc.) especificados por el usuario para ejecutar el trabajo.
4. El *Central Manager* asigna el trabajo a un *worker node* y este empieza a ejecutarlo. Si el trabajo tiene inconvenientes al ejecutarse, se graban en el archivo de registro de cada trabajo los errores de ejecución.
5. Al finalizar la ejecución el *worker node* envía una señal de tarea finalizada al *Central Manager.*
6. El *Central Manager* elimina el trabajo de la cola.
7. El usuario consulta los resultados obtenidos de la ejecución del trabajo.

Dadas las ventajas computacionales que ofrecen Condor y la computación distribuida, para este trabajo de investigación se utilizó una arquitectura conformada por tres máquinas de un *cluster* computacional. Todas estas tienen la capacidad de ejecutar tareas y una de ellas también es la encargada de ser Central Manager. En la Figura 4-2 se observa el despliegue realizado.

Figura 4-2. Arquitectura del *cluster* computacional.



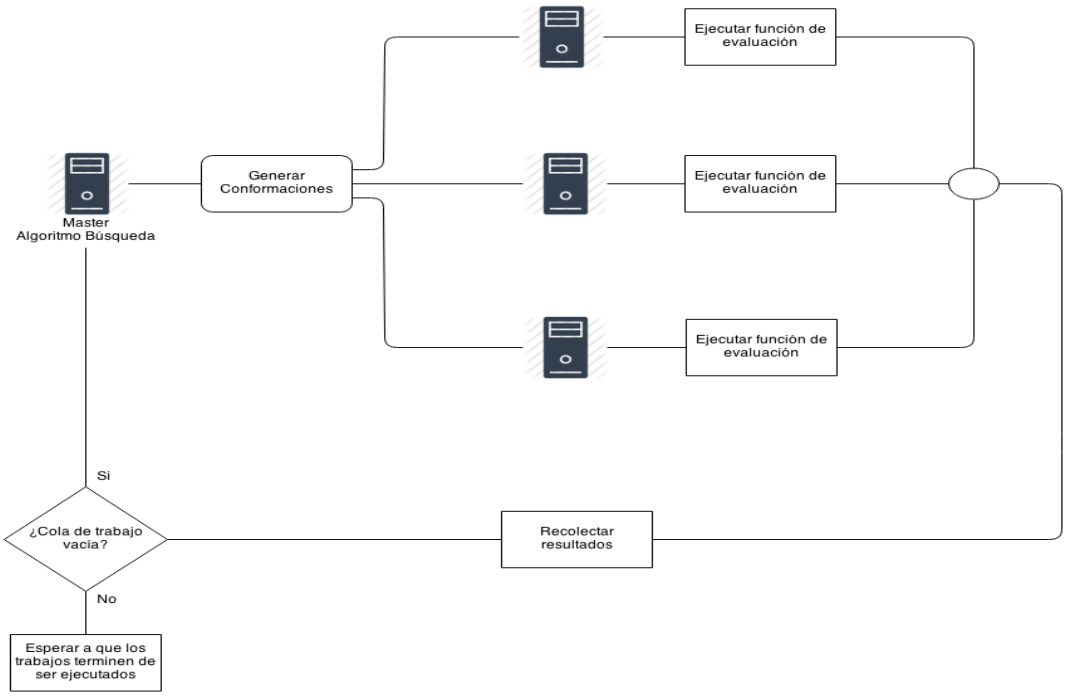
Dado que el modelo propuesto fue implementado en Java, se utilizó el universo de Condor Java, el cual ejecuta programas escritos en Java en cualquier máquina del *pool* que tenga instalada la Máquina Virtual de Java (JVM – Java Virtual Machine).

Para acceder a los servicios del sistema Condor desde Java se utilizó la librería Condor Java API [56], la cual permite enviar, supervisar y controlar trabajos de Condor desde un programa Java. Para el envío de trabajos utilizando Condor Java API, se tiene la opción de crear un nuevo archivo de configuración o utilizar uno existente.

La distribución de la función de evaluación se presenta en la Figura 4-3.

En la etapa de generación de conformaciones se obtienen los valores de las variables de decisión y en la etapa de ejecución de la función de evaluación, se actualizan las coordenadas internas y se realiza el cálculo de la función de puntuación.

Figura 4-3. Esquema de distribución utilizado para el algoritmo de búsqueda local.



5. Experimentación y resultados

En este capítulo se presentan la experimentación y los resultados obtenidos utilizando el modelo propuesto en las dos etapas: considerando las proteínas como cuerpos rígidos y la de inclusión de flexibilidad.

Los complejos seleccionados para la experimentación pertenecen a un conjunto de datos de referencia utilizado para evaluar y comparar los métodos computacionales desarrollados para el *docking* proteína-proteína [39]. Los datos utilizados se presentan en la Tabla 5-1.

Tabla 5-1. Conjunto de datos utilizado en la experimentación

Complejo	Tipo	PDB ID 1	Prot. 1	Res.	PDB ID 2	Prot. 2	Res.	Método Prot. 1 y 2
2SNI	Enzyme/Inhibitor	1UBN_A	Subtilisin	2.4	2CI2_I	Chymotrypsin inhibitor 2	2	Difracción de rayos X
1CGI	Enzyme/Inhibitor	2CGA_B	Bovine chymotrypsinogen	1.8	1HPT_	PSTI	2.3	Difracción de rayos X
1CLV	Enzyme/Inhibitor	1JAE_A	α-amylase	1.65	1QFD_A(1)	α-amylase inhibitor		Difracción de rayos X / NMR
1PPE	Enzyme/Inhibitor	1BTP_	Bovine trypsin	2.20	1LU0_A	CMTI-1 squash inhibitor	1.03	Difracción de rayos X

Los archivos en formato PDB de las proteínas con las que se realizó la experimentación se obtuvieron del conjunto de datos presentado en [39]. Para cada complejo el conjunto de datos proporciona la información de la estructura tridimensional del ligando y del receptor que forman el complejo nativo, y la información de estructura del ligando y del receptor para el caso en que el complejo aún no se ha formado.

El lenguaje de desarrollo para la implementación del modelo propuesto fue Java. Para la etapa en la que las proteínas se consideran cuerpos rígidos, los experimentos fueron ejecutados en una estación de trabajo con un procesador Core (TM) i7-2600 CPU 3.40 GHz y con memoria RAM de 8 GB. El sistema operativo en el cual se ejecutaron los experimentos fue Scientific Linux 6.4. Para la etapa de inclusión de flexibilidad, los experimentos se corrieron en un *cluster* computacional conformado por tres máquinas virtualizadas con el hipervisor KVM (*Kernel-based Virtual Machine*) con sistema operativo Scientific Linux 6.4.

Los parámetros utilizados en el algoritmo genético de la primera etapa, donde las proteínas se tratan como cuerpo rígidos, se presentan en la Tabla 5-2:

Tabla 5-2. Parámetros utilizados en el AG propuesto para la etapa 1.

Parámetro	Valor
Tamaño de la población con información de sitios de unión	Depende del número de residuos que son potenciales sitios de unión
Tamaño de la población sin información de sitios de unión	Depende del tamaño de la proteína
Número máximo de evaluaciones de la función de aptitud con información de sitios de unión	20000
Número máximo de evaluaciones de la función de aptitud sin información de sitios de unión	300000
Probabilidad de cruce	0,8
Probabilidad de mutación	0.03
Proporción de la población copiada a la siguiente generación	20%

En la experimentación el tamaño de la población para el algoritmo genético dependía de si se tenía en cuenta la información de los sitios de unión del receptor, o si no. Específicamente, si se contaba con la información de los sitios de unión y el número de sitios de unión es menor a 50 residuos, el tamaño de la población es de 50 individuos.

Por otro lado, si el número de sitios de unión era mayor a 50, el número de individuos se tomó como el número de sitios de unión. Para el caso en el que no se cuenta con la información de los sitios de unión, el número de individuos depende del número de átomos que hacen parte del conjunto de vértices de la envolvente convexa formada para el receptor. La regla previamente descrita se definió teniendo en cuenta que el número de potenciales sitios de unión establecidos por BSpred para los casos de estudio (1UBN = 8 y 2CGA = 7) es menor a 10 residuos y que al no contar con el conocimiento de todos los residuos de la superficie de una proteína, la envolvente convexa puede considerarse una representación de los puntos en el espacio donde se encuentran los residuos de la superficie de esta.

En la etapa de inclusión de flexibilidad los parámetros utilizados fueron: máximo 200 evaluaciones, probabilidad de mutación de 1/número de grados de libertad y 5 rondas de mejoramiento por individuo.

En cada experimento se tuvieron en cuenta las medidas de rendimiento que se relacionan a continuación para evaluar los resultados obtenidos:

- Fracción de contactos nativos (f_{nat}).
- Fracción de contactos no nativos (f_{no-nat}).
- RMSD: valor obtenido al comparar la estructura del complejo predicho con la estructura del complejo nativo. El valor del RMSD se calculó con el programa *superpose* perteneciente a *TINKER.*
- Energía libre de unión:

  $$Energía_{\ unión} = Energía_{complejo} - Energía_{ligando} - Energía_{receptor}$$

  Los cálculos de la energía se realizaron con el programa *analyze* perteneciente a *TINKER*.

5.1 Resultados obtenidos con los complejos proteína-proteína de prueba

En esta sección se presentan los resultados obtenidos con el modelo propuesto para cada uno de los complejos indicados en la Tabla 5-1.

5.1.1 2SNI

El complejo 2SNI está conformado por las proteínas con código PDB 1UBN y 2CI2. Este complejo es un ejemplo de la interacción enzima/inhibidor. El receptor (1UBN) está compuesto por 275 residuos y el ligando (2CI2) por 64.

Para este complejo se ejecutaron experimentos teniendo en cuenta la información de los potenciales sitios de unión y también sin tener en cuenta dicha información. Para el caso de prueba en el que se conocían los sitios de unión, el tamaño de la población inicial fue de 50 individuos. La predicción de los sitios de unión se realizó a través del servicio web de BSpred que obtuvo los residuos del receptor indicados en la Tabla 5-3. Para el caso en el que no se cuenta con la información de los sitios de unión, se estableció el tamaño de la población en 77 individuos. Los resultados obtenidos se presentan en la Tabla 5-4.

Tabla 5-3. Potenciales sitios de unión de la proteína 1UBN

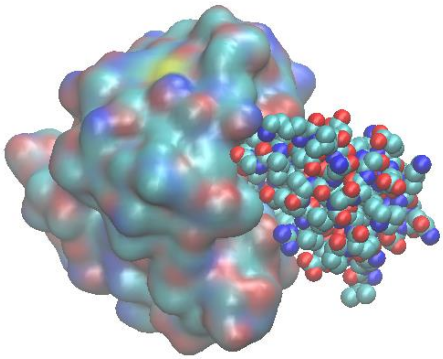
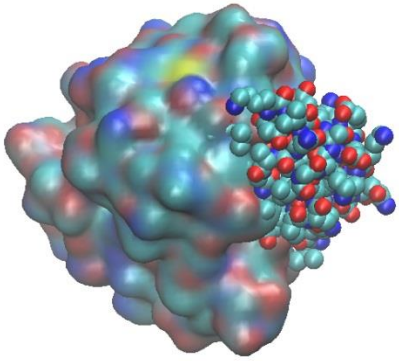
Potenciales sitios de unión	
4 - VAL	227 – SER
9 – SER	165 – SER
230 - ASN	231 – THR
5 - PRO	3 - SER

Tabla 5-4. Resultados para el complejo 2SNI.

2SNI	f_{nat}	f_{no-nat}	***RMSD***	$\boldsymbol{Energía}$ $(\boldsymbol{kcal/mol})$
Sin sitios de unión	0.068	2.041	3.75	3744.005
Sitios unión	0	0.287	23.36	-88.175
Complejo obtenido con inclusión de flexibilidad	0	0.808	14.85	2782.8254

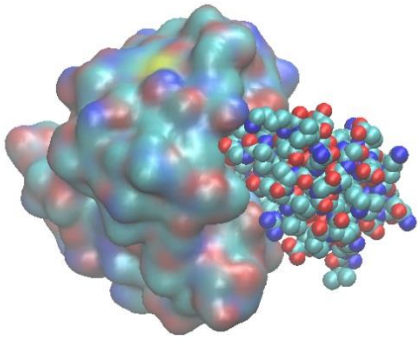
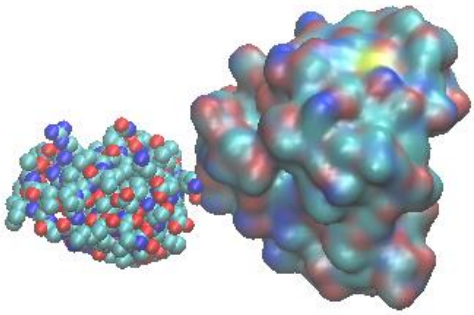
En la Figura 5-1 se presentan el complejo nativo y el complejo obtenido, correspondiente a la ejecución del algoritmo sin tener en cuenta los sitios de unión.

Figura 5-1. Complejo 2SNI nativo comparado con el complejo obtenido, sin tener en cuenta la información relacionada con los sitios de unión.

Complejo 2SNI nativo	Complejo 2SNI obtenido
	

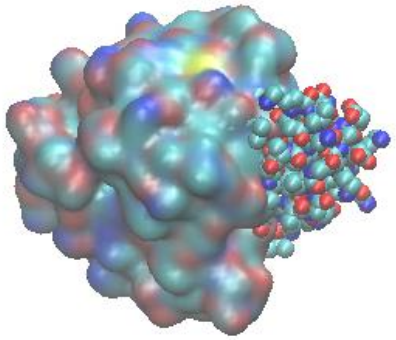
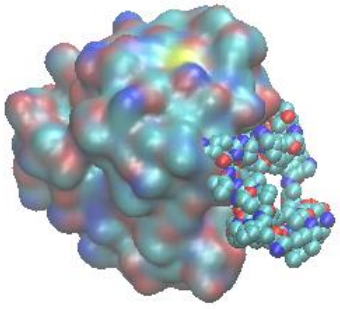
Por otro lado, en la Figura 5-2 se presentan el complejo nativo y el complejo obtenido después de la ejecución del algoritmo teniendo en cuenta los sitios de unión.

Figura 5-2. Complejo 2SNI nativo comparado con el complejo obtenido, teniendo en cuenta la información de sitios de unión.

Complejo 2SNI nativo	Complejo 2SNI obtenido
	

En la Figura 5-3 se muestra el complejo obtenido en la primera etapa sin considerar los sitios de unión y el complejo resultante después de la inclusión de flexibilidad.

Figura 5-3. Complejo 2SNI obtenido en la etapa 1 comparado con el resultado obtenido después de la inclusión de flexibilidad.

Complejo 2SNI obtenido etapa 1	**Complejo 2SNI con inclusión de flexibilidad**
	

Las imágenes de los complejos se obtuvieron con el programa de visualización molecular Visual Molecular Dynamics (VMD) [57].

5.1.2 1CGI

El complejo 1CGI está compuesto por las proteínas con código PDB 2CGA y 1HPT. Este complejo es un ejemplo de la interacción enzima/inhibidor. El receptor (2CGA) está compuesto por 245 residuos y el ligando (1HPT) por 56 residuos.

Para este complejo se realizaron experimentos teniendo en cuenta la información de los potenciales sitios de unión y sin tener en cuenta esta información. Para el caso en el que se conocían los sitios de unión, el tamaño de la población inicial fue de 50 individuos. Los residuos identificados por BSpred como potenciales sitios de unión del receptor se indican en la Tabla 5-5. Para el caso en el que no se contaba con la información de los sitios de unión, se estableció el tamaño de la población en 74 individuos. Los resultados obtenidos se resumen en la Tabla 5-6.

Tabla 5-5. Potenciales sitios de unión de la proteína 2CGA.

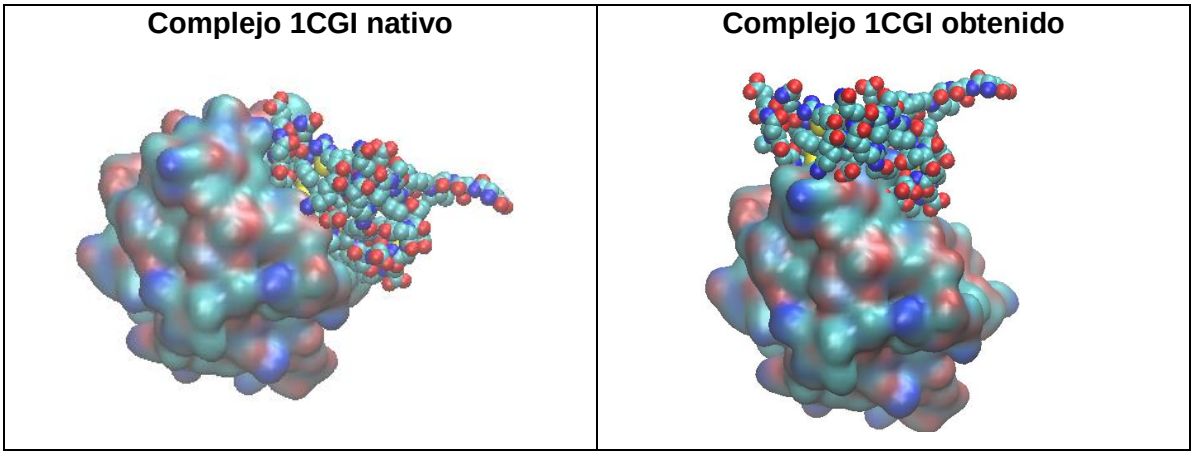
Potenciales sitios de unión	
5 - ALA	167 - SER
152 - TYR	151 - LYS
149 - ASN	3 - VAL
4 - PRO	

Tabla 5-6. Resultados para el complejo 1CGI.

1CGI	f_{nat}	f_{no-nat}	***RMSD***	$\boldsymbol{Energía\ (kcal/mol)}$
Sin sitios de unión	0	1.523	6.33	-2625.064
Sitios unión	0	0.558	9.73	-2624.926
Complejo obtenido con inclusión de flexibilidad	0,011	1,116	9.39	-2132.7455

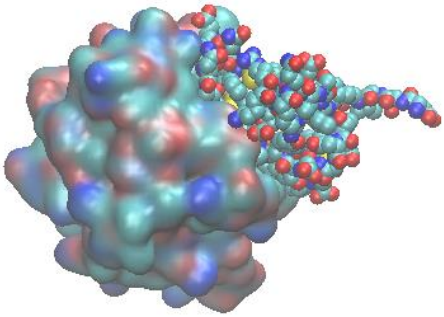
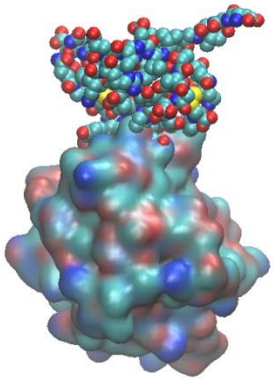
En la Figura 5-4 se muestran el complejo nativo y el complejo obtenido, correspondiente a la ejecución del algoritmo sin tener en cuenta los sitios de unión.

Figura 5-4. Complejo 1CGI nativo comparado con el complejo obtenido, sin tener en cuenta la información relacionada con los sitios de unión.



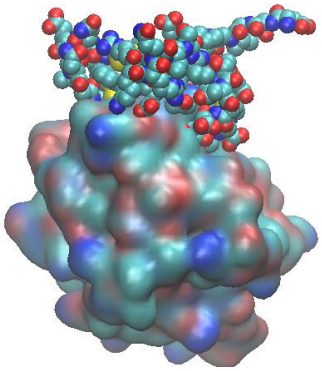
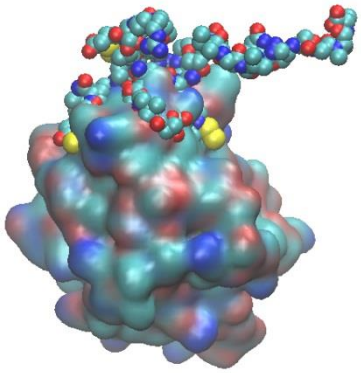
Así mismo, en la Figura 5-5 se muestran el complejo nativo y el complejo obtenido después de la ejecución del algoritmo teniendo en cuenta los sitios de unión.

Figura 5-5. Complejo 1CGI nativo comparado con el complejo obtenido, teniendo en cuenta la información de sitios de unión.

Complejo 1CGI nativo	**Complejo 1CGI obtenido**
	

En la Figura 5-6 se muestra el complejo obtenido en la primera etapa, sin considerar los sitios de unión, y el complejo obtenido después de la inclusión de flexibilidad.

Figura 5-6. Complejo 1CGI obtenido en la etapa 1 comparado con el resultado obtenido después de la inclusión de flexibilidad.

Complejo 1CGI obtenido etapa 1	**Complejo 1CGI con inclusión de flexibilidad**
	

5.1.3 1CLV

El complejo 1CLV está conformado de las proteínas con código PDB 1JAE y 1QFD. Este complejo es un ejemplo de la interacción enzima/inhibidor. El receptor (1JAE) está compuesto por 470 residuos y el ligando (1QFD) por 32 residuos.

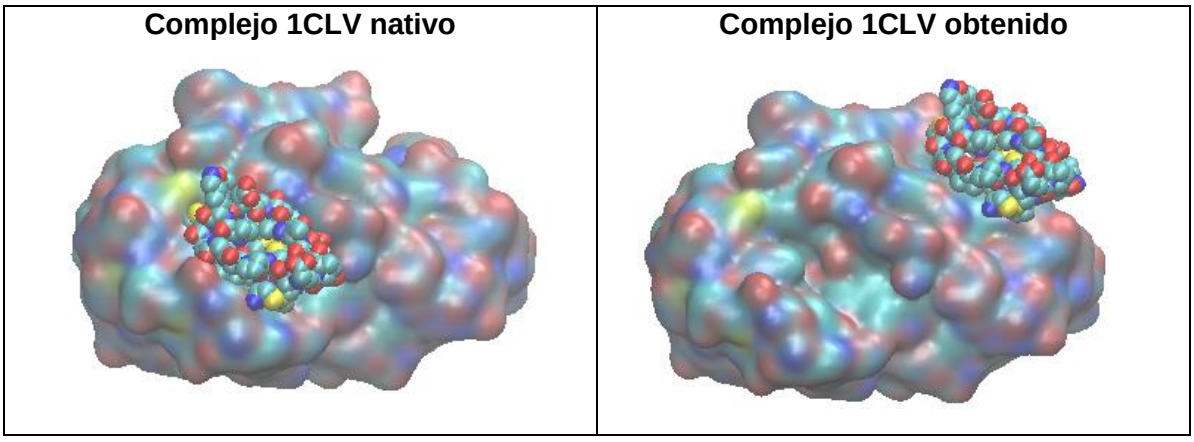
Para este complejo se ejecutaron experimentos sin tener en cuenta la información de los potenciales sitios de unión. El tamaño de la población para la ejecución de la primera etapa fue de 89 individuos. Los resultados obtenidos se resumen en la Tabla 5-7.

Tabla 5-7. Resultados para el complejo 1CLV.

1CLV	f_{nat}	f_{no-nat}	***RMSD***	$\boldsymbol{Energía\ (kcal/mol)}$
Sin sitios de unión	0	0.460	8.26	-96.890
Complejo obtenido con inclusión de flexibilidad	0	0,578	8.75	9283.5603

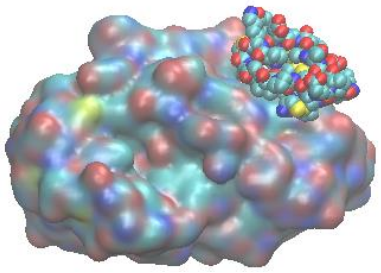
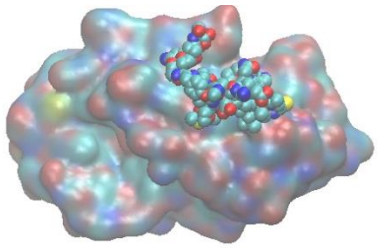
En la Figura 5-7 se presenta la comparación visual entre el complejo nativo y el complejo obtenido, resultante de la experimentación realizada.

Figura 5-7. Complejo 1CLV nativo comparado con el complejo obtenido, sin tener en cuenta la información relacionada con los sitios de unión.



En la Figura 5-8 se muestra el complejo obtenido en la primera etapa, sin considerar los sitios de unión, y el complejo obtenido después de la inclusión de flexibilidad.

Figura 5-8. Complejo 1CLV obtenido en la etapa 1 comparado con el resultado obtenido después de la inclusión de flexibilidad.

Complejo 1CLV obtenido etapa 1	**Complejo 1CLV con inclusión de flexibilidad**
	

5.1.4 1PPE

El complejo 1PPE está conformado por las proteínas con código PDB 1BTP y 1LUO. Este complejo es un ejemplo de la interacción enzima/inhibidor. El receptor (1BTP) está compuesto por 223 residuos y el ligando (1LUO) por 29 residuos.

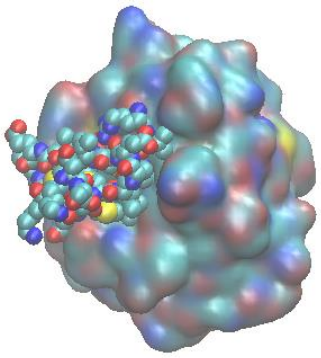
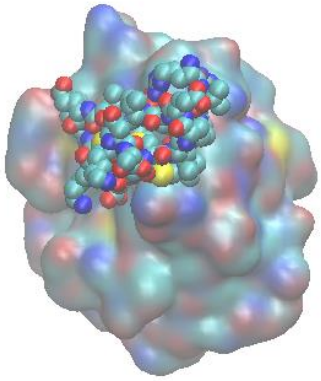
Para este complejo se ejecutaron experimentos sin tener en cuenta la información de los potenciales sitios de unión. El tamaño de la población para la ejecución de la primera etapa fue de 78 individuos. Los resultados obtenidos se resumen en la Tabla 5-8.

Tabla 5-8. Resultados para el complejo 1PPE.

1PPE	f_{nat}	f_{no-nat}	***RMSD***	$Energía\ (kcal/mol)$
Sin sitios de unión	0	0,428	6.55	18.939
Complejo obtenido con inclusión de flexibilidad	0.015	0.492	7.75	616.6848

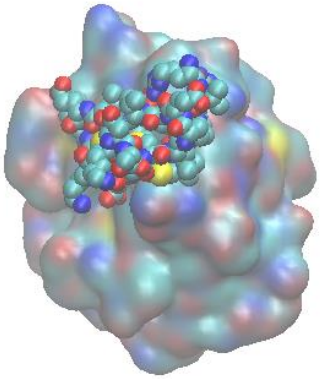
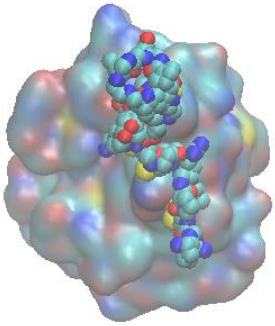
En la Figura 5-9 se muestran el complejo nativo y el complejo obtenido.

Figura 5-9. Complejo 1PPE nativo comparado con el complejo obtenido, sin tener en cuenta la información relacionada con los sitios de unión.

Complejo 1PPE nativo	**Complejo 1PPE obtenido**
	

En la Figura 5-10 se muestra el complejo obtenido en la primera etapa, sin considerar los sitios de unión, y el complejo obtenido después de la inclusión de flexibilidad.

Figura 5-10. Complejo 1PPE obtenido en la etapa 1 comparado con el resultado obtenido después de la inclusión de flexibilidad.

Complejo 1PPE obtenido etapa 1	**Complejo 1PPE con inclusión de flexibilidad**
	

5.1.5 Comparación con métodos existentes para el *docking* proteína-proteína

En esta sección se presenta la comparación de los resultados obtenidos con el modelo propuesto con los obtenidos por otros métodos existentes.

En la Tabla 5-9 se presenta la comparación de los resultados obtenidos con el modelo propuesto para el complejo 2SNI y los obtenidos con dos métodos de *docking* proteína-proteína existentes. Particularmente, se compararon los resultados con los obtenidos por *Hex Server* (http://hexserver.loria.fr) y *ZDOCK Server* (http://zdock.umassmed.edu/).

En las Tablas 5-10, 5-11 y 5-12 se presenta la comparación de los resultados obtenidos para los complejos 1CGI, 1CLV y 1PPE, respectivamente.

Tabla 5-9. Comparación de los resultados por el método propuesto con otros existentes para el complejo 2SNI.

Método/Servidor	f_{nat}	f_{no-nat}	$RMSD$	$Energía\ (kcal/mol)$
Complejo nativo				235511.2825
Complejo obtenido con el modelo Propuesto	0.068	2.041	3.75	3744.005
Hex Server	0.876	0.328	0.343	5620.0834
ZDOCK	0.082	1	0.551	10388.9653

Tabla 5-10. Comparación de los resultados por el método propuesto con otros existentes para el complejo 1CGI.

Método/Servidor	f_{nat}	f_{no-nat}	$RMSD$	$Energía\ (kcal/mol)$
Complejo nativo				-55.2101
Complejo obtenido con el modelo Propuesto	0	1.523	6.33	-2625.064
Hex Server	0.965	0.186	0.072	144.6163
ZDOCK	0.244	0.965	4.711	-141636.4172

Tabla 5-11. Comparación de los resultados por el método propuesto con otros existentes para el complejo 1CLV.

Método/Servidor	f_{nat}	f_{no-nat}	$RMSD$	$Energía\ (kcal/mol)$
Complejo nativo				-141.5618
Complejo obtenido con el modelo Propuesto	0	0.460	8.26	-96.890
Hex Server	0,815	0,355	0.338	40233.5302
ZDOCK	0	0,973	1.094	2.1441E9

Tabla 5-12. Comparación de los resultados por el método propuesto con otros existentes para el complejo 1PPE.

Método/Servidor	f_{nat}	f_{no-nat}	$RMSD$	$Energía\ (kcal/mol)$
Complejo nativo				-141.2467
Complejo obtenido con el modelo Propuesto	0	0,428	6.55	18.939
Hex Server	0,714	0,269	0.291	1780.3156
ZDOCK	0	1,031	0.461	816.7578

5.1.6 Análisis de los resultados obtenidos

Al analizar los resultados obtenidos, desde un punto de vista general, se puede decir que estos no son lo suficientemente satisfactorios de acuerdo con lo esperado, teniendo en cuenta que los valores de RMSD son mayores a 2 Å. Los mejores resultados se obtuvieron para el complejo 2SNI obteniendo un RMSD de 3.75 Å, al comparar la estructura del complejo obtenido con la del complejo nativo. Por otro lado, los valores de RMSD para los complejos 1CGI, 1CLV y 1PPE fueron de 6.33 Å, 8.26 Å y 6.55 Å, respectivamente. Al contrastar los valores de RMSD obtenidos con el tamaño de los complejos de prueba, se observa que los mejores resultados se obtuvieron para las proteínas de menor tamaño. En cuanto a las soluciones obtenidas con las proteínas de menor tamaño, estas pueden deberse a que el espacio de búsqueda es más pequeño. Por otro lado, la diferencia de RSMD al comparar las estructuras resultantes con los complejos nativos podría ser explicable en el hecho de que la función de puntuación

utilizada favorece las soluciones con un menor valor de energía (complejos 2SNI, 1CGI) o que el algoritmo convergió a un mínimo local (complejos 1CLV, 1PPE).

Específicamente para el complejo 2SNI el algoritmo de búsqueda encontró una conformación estructuralmente similar a la del complejo nativo, teniendo en cuenta el valor del RMSD. Sin embargo, se observa que el ligando tiene un alto grado de traslapo con el receptor, lo que pudo influir en el valor del índice de fracción de contactos no nativos, el cual fue de 2.041.

Para los complejos 2SNI y 1CGI, con respecto a la ejecución del algoritmo, teniendo en cuenta la información de potenciales sitios de unión, durante la exploración los ligandos se salen de las regiones establecidas por los sitios de unión. Esto se puede ver reflejado en los resultados obtenidos para el complejo 2SNI, ya que el valor del RMSD es de 23.36 Å y la fracción de contactos nativos es 0. Estos resultados pueden estar relacionados con el enfoque propuesto para la generación de la población inicial y también a que el algoritmo genético está orientando a encontrar soluciones con menor valor de energía (-88.175 kcal/mol). Lo anterior también se puede ver reflejado en los residuos de interfaz del complejo obtenido, los cuales se presentan en la Tabla 5-13. Comparando estos residuos con los indicados por BSpred, se observa que ninguno de los residuos que se indicaron como potenciales sitios de unión hace parte de la interfaz del complejo obtenido. De igual manera, se resalta que los resultados obtenidos para el complejo 1CGI teniendo en cuenta los sitios de unión, conducen a las mismas conclusiones.

Tabla 5-13. Residuos de interfaz del complejo 2SNI obtenido teniendo en cuenta la información de los potenciales sitios de unión.

Número residuo	Residuo
15	ALA
19	GLN
237	LYS
238	HIS
253	THR
275	GLN

Para el complejo 1CGI los resultados obtenidos indican que la solución que se obtuvo está acorde con el algoritmo genético utilizado, ya que este está orientado a la minimización de la función de energía. En relación con lo anterior, se observa que el

complejo obtenido tiene un valor de energía mucho menor que la del complejo nativo, siendo -2625.064 kcal/mol y -55.210 kcal/mol, respectivamente.

Específicamente para el complejo 1CLV, teniendo en cuenta que para el complejo obtenido el valor del RMSD fue de 8.26Å, la fracción de contactos nativos fue 0 y la fracción de contactos no nativos de 0.460, se observa que la estructura de este complejo es lejana en comparación con la reportada para el complejo nativo.

En general, se puede observar que en el espacio de soluciones pequeños cambios en la posición del ligando pueden ocasionar grandes cambios de energía en el complejo formado, convirtiendo el complejo obtenido en una mala solución. Específicamente para el complejo 1PPE, la diferencia de energía es grande siendo la de la solución obtenida 18.939 kcal/mol y la del complejo nativo -141.2467 kcal/mol.

En la etapa de inclusión de flexibilidad los resultados para el complejo 2SNI presentaron minimización de la energía en el complejo obtenido. Para los complejos 1CGI, 1CLV y 1PPE el valor de la energía del complejo obtenido fue mayor. En cuanto al valor del RMSD los resultados indican que para los casos de estudio la estructura del complejo obtenido se volvió más lejana en comparación con la estructura del complejo nativo. Se considera que si se puede contar con la información sobre las regiones rígidas y flexibles de las proteínas, se puede usar ese conocimiento para reducir los grados de libertad del problema (ángulos rotables) y para evitar la violación de restricciones de tipo biológico en cada cambio conformacional.

De los métodos con los que se comparó el modelo propuesto (*Hex Server* y *ZDOCK*) se puede resaltar que en el proceso de búsqueda de soluciones se incluye información relacionada con las características de la superficie de las proteínas, y la función de puntuación incluye información relacionada con la complementariedad geométrica de las proteínas que conforman el complejo, los valores de la energía electrostática y de desolvatación.

Al analizar los resultados presentados en las Tablas 5-9, 5-10, 5-11 y 5-12 se establece la necesidad de integrar nuevos términos a la función de puntuación utilizada en el modelo propuesto para mejorar la precisión del mismo. En el modelo propuesto la función de puntuación corresponde a una función de energía y el algoritmo de búsqueda corresponde a un algoritmo genético que está diseñado para minimizar la función

objetivo. Dado esto, en varios casos el complejo con menor energía no corresponde a una estructura cercana a la del complejo nativo.

6. Conclusiones y trabajo futuro

En la presente tesis se trató un problema de gran importancia e interés en la biología computacional, la interacción proteína-proteína. El modelo propuesto intenta solucionar este problema utilizando métodos de inteligencia computacional para la simulación de procesos biológicos, los cuales están dirigidos a la exploración del espacio de soluciones de las conformaciones de complejos proteicos, teniendo en cuenta sus valores de energía. El método de solución planteado consta de dos etapas: la primera tiene como objetivo identificar las interacciones proteína-proteína considerando las proteínas como cuerpos rígidos, y la segunda incluye la flexibilidad en el ligando. El desarrollo de este trabajo, además de resolver un problema científico, provee una implementación distribuida del método propuesto.

Con respecto a la etapa en la que las proteínas se consideran cuerpos rígidos se puede concluir lo siguiente:

- Tomando en consideración que la información con la que se cuenta al iniciar la ejecución del algoritmo corresponde a las coordenadas atómicas de las proteínas, el sistema de representación algebraica es adecuado porque facilita la aplicación de las transformaciones que permiten explorar el espacio de búsqueda y realizar la evaluación de la función de puntuación, entre otros aspectos.

- Al aplicar el método propuesto, se observó que no siempre los complejos con una menor energía de unión representan soluciones cercanas estructuralmente a las reportadas. Por lo tanto, la función de puntuación debería incluir términos que correspondan a otros aspectos relacionados con la estructura del complejo.

Con respecto a la etapa de inclusión de flexibilidad se puede concluir lo siguiente:

- La representación trigonométrica de las proteínas es el sistema de representación apropiado para esta etapa, ya que permite trabajar con los grados de libertad correspondientes al problema de la interacción proteína-proteína flexible.

- Al considerar flexibilidad en el ligando, si se tuviera información acerca de las regiones flexibles y rígidas de la proteína, se podría facilitar el proceso de búsqueda disminuyendo el número de grados de libertad.

Con base en los resultados obtenidos, se debería mejorar la función de puntuación incluyendo información relacionada con la estructura tridimensional de las proteínas, como por ejemplo características de su superficie, y otra información que ayude a relacionar la estructura formada y su energía.

De acuerdo con los resultados obtenidos se sugieren futuras exploraciones para el ajuste del modelo propuesto, que permitan mejorar la calidad de las soluciones obtenidas. Dentro de estas exploraciones, algunas actividades proyectadas como trabajo futuro son:

- Realizar experimentación adicional con otros valores de los parámetros utilizados en los algoritmos y con más datos de referencia aplicando la metodología propuesta, con el fin de efectuar una validación más completa del modelo utilizado.

- Explorar la aplicación de otras funciones de puntuación, además de la inclusión de nuevos términos a la función utilizada, en el modelo propuesto con el propósito de tratar de obtener mejores configuraciones de los complejos.

A. Anexo: Instalación del software Condor

A continuación se describen los pasos para la instalación del sistema Condor en un computador con el sistema operativo Scientific Linux 6. También se presenta la configuración de un sistema de archivos compartido por red (NFS: Network File Shared) que servirá como carpeta de acceso a los archivos de configuración y ejecución de trabajos para las máquinas que conforman el *cluster*.

Para crear un directorio de archivos compartidos por red se tienen dos funciones principales: la primera es el servidor, que es la máquina encargada de crear el espacio compartido, y la segunda es el cliente, que son máquinas que acceden a ese directorio de acuerdo con los permisos dados por el servidor [58].

A continuación se presentan los pasos, con sus respectivos comandos de consola, para crear un directorio compartido:

1. Instalar el paquete de software NFS en el servidor, es decir, las librerías y demás dependencias que se necesitan para la creación de un NFS.

```
[root@server ~]# yum install nfs* -y
```

2. Iniciar el servicio NFS en el servidor

```
[root@server ~]# /etc/init.d/nfs start
```

   Al arrancar el servicio correctamente, la consola muestra los siguientes mensajes:

```
Starting NFS services: [ OK ]
Starting NFS mountd: [ OK ]
```

```
Stopping RPC idmapd: [ OK ]
Starting RPC idmapd: [ OK ]
Starting NFS daemon: [ OK ]
```

3. Configurar el servicio NFS para que inicie al arrancar el servidor.

```
[root@server ~]# chkconfig nfs on
```

4. Instalar NFS en el cliente

```
[root@client]# yum install nfs* -y
```

5. Al iniciar el servicio NFS en el cliente ejecutando el comando

```
[root@client]# /etc/init.d/nfs start
```

Se muestran en la consola los siguientes mensajes

```
Starting NFS services: [ OK ]
Starting NFS quotas: [ OK ]
Starting NFS mountd: [ OK ]
Stopping RPC idmapd: [ OK ]
Starting RPC idmapd: [ OK ]
Starting NFS daemon: [ OK ]
```

6. Configurar el servicio NFS para que inicie al arrancar el cliente.

```
[root@client]# chkconfig nfs on
```

7. Crear el directorio compartido en el servidor y asignar permisos de lectura y escritura.

```
[root@server ~]# mkdir /home/example
[root@server ~]# chmod 755 /home/example/
```

8. Exportar el directorio compartido en el servidor para que pueda ser accedido por un cliente. El comando que se ejecuta abre el archivo exports en un editor de texto, al cual se le adiciona la línea de configuración que aparece más abajo.

```
[root@server ~]# vi /etc/exports
```

```
/home/example 192.168.1.0/24(rw,sync,no_root_squash,no_all_squash)
```

El significado de los parámetros en esta línea es el siguiente:

- 192.168.1.0/24 - Es la dirección IP de la máquina o el segmento de red que puede acceder al directorio compartido.
- rw – Permite que el directorio compartido sea de lectura y escritura.
- sync – Permite sincronizar el directorio compartido cada vez que se cree un subdirectorio o archivo en él.
- no_root_squash – Habilita privilegios de superusuario (root). Los usuarios pueden leer, escribir y eliminar archivos del directorio compartido.
- no_all_squash - Habilita la autorización de los usuarios

9. Reiniciar el servicio NFS en el servidor ejecutando el comando que despliega los mensajes que aparecen después de él:

```
[root@server ~]# /etc/init.d/nfs restart
Shutting down NFS daemon: [ OK ]
Shutting down NFS mountd: [ OK ]
Shutting down NFS services: [ OK ]
Starting NFS services: [ OK ]
Starting NFS mountd: [ OK ]
```

```
Stopping RPC idmapd: [ OK ]

Starting RPC idmapd: [ OK ]

Starting NFS daemon: [ OK ]
```

10. Configurar reglas en el firewall para que no se bloquee el servicio NFS ni los puertos que son usados para la transmisión de datos. Esto se logra con las siguientes acciones:
    a. Abrir el archivo /etc/sysconfig/nfs y suprimir los comentarios (eliminar el carácter '**#**') de las líneas que se relacionan a continuación:

```
RQUOTAD_PORT=875

LOCKD_TCPPORT=32803

LOCKD_UDPPORT=32769

MOUNTD_PORT=892

STATD_PORT=662

STATD_OUTGOING_PORT=2020
```

    b. Reiniciar el servicio NFS, ejecutando el siguiente comando.

```
[root@server ~]# /etc/init.d/nfs restart
```

11. Adicionar reglas de excepción en el firewall (iptables) para los puertos usados en NFS. El comando que se ejecuta abre el archivo iptables en un editor de texto, al cual se le adicionan las líneas de configuración que aparecen más abajo.

```
[root@server ~]# vi /etc/sysconfig/iptables
```

```
-A INPUT -m state --state NEW -m udp -p udp --dport 2049 -j ACCEPT

-A INPUT -m state --state NEW -m tcp -p tcp --dport 2049 -j ACCEPT
```

```
-A INPUT -m state --state NEW -m udp -p udp --dport 111 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 111 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 32769 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 32803 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 892 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 892 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 875 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 875 -j ACCEPT
-A INPUT -m state --state NEW -m udp -p udp --dport 662 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 662 -j ACCEPT
```

12. Reiniciar el servicio de iptables, ejecutando el siguiente comando.

```
[root@server ~]# service iptables restart
```

13. Montar el directorio compartido en el cliente, es decir, crear un punto de montaje para cargar los directorios compartidos por el servidor, lo cual significa crear o utilizar una carpeta en la cual se tendrá el directorio compartido. Esto se lleva a cabo con los siguientes comandos:

```
[root@client]# mkdir -p /nfs/shared
```

```
[root@client]# mount -t nfs 192.168.1.2:/home/example/ /nfs/shared/
```

El significado de los parámetros en esta línea es el siguiente:

192.168.1.2:/home/example/ es la ubicación del directorio compartido, en este caso 192.168.1.2 es la dirección IP del servidor.

14. Probar el directorio compartido creando una carpeta desde el cliente, en este caso en '/nfs/shared', utilizando el siguiente comando:

```
[root@client shared]# mkdir test
```

15. El comando que se ejecuta abre el archivo fstab en un editor de texto, al cual se le adiciona la línea de configuración que aparece más abajo, la cual permite realizar el montaje del directorio compartido de forma automática en el cliente, en caso de reiniciar el sistema.

```
[root@server ~]# vi /etc/fstab
```

```
192.168.1.2:/home/example /nfs/shared nfs rw,sync,hard,intr 0 0
```

Instalación del software Condor

1. Después de tener configurado el NFS se procede a instalar el software Condor en todos los nodos del cluster; para ello se deben realizar los siguientes pasos:

    a. Descargar el paquete de software Condor. Esto se puede realizar a través de un repositorio YUM o descargando el archivo RPM de la página web http://research.cs.wisc.edu/htcondor/downloads/ . Para mayor practicidad se realizará con YUM. Para ello se debe agregar la información del repositorio de Condor a YUM; esto se logra ejecutando los siguientes comandos:

```
cd /etc/yum.repos.d
wget http://www.cs.wisc.edu/condor/yum/repo.d/condor-stable-rhel6.repo
```

2. Instalación del paquete de software Condor, con el siguiente comando.

```
yum install condor
```

3. En el servidor NFS exportar el directorio /nfs/condor/condor-etc , con el propósito de compartir los archivos de configuración del cluster, realizando las siguientes acciones:

a. Descargar el archivo de configuración del cluster de Condor, este se puede encontrar en la siguiente dirección: https://twiki.opensciencegrid.org/twiki/pub/Tier3/CondorRPMInstall/condor_config.cluster

b. Editar en el archivo descargado condor_config.cluster las líneas que se muestran a continuación.

```
UID_DOMAIN = yourdomain.org
CONDOR_HOST = gc1-ce.yourdomain.org
```

El significado de estas líneas es el siguiente:

- UID_DOMAIN: Nombre del dominio en que se encuentra el central manager.
- CONDOR_HOST: Nombre de la máquina central manager.

4. Crear un archivo de configuración para el central manager, uno para los workers nodes y otro para los interactive nodes.

```
vi /nfs/condor/condor-etc/headnode.config
vi /nfs/condor/condor-etc/worker.config
vi /nfs/condor/condor-etc/interactive.config
```

Estos archivos contienen los procesos que se ejecutarán dependiendo del rol que se le asigne a la máquina, como se indica a continuación:

- Central manager:

```
DAEMON_LIST = MASTER, COLLECTOR, NEGOTIATOR
```

- Worker node:

```
DAEMON_LIST = MASTER, STARTD
```

- Interactive node:

```
DAEMON_LIST = MASTER, SCHEDD
```

5. Para cada nodo del cluster crear un vínculo que apunte a los archivos de configuración del sistema Condor, dependiendo de su rol. Para ello se deben ejecutar los comandos que se relacionan a continuación:

```
ln -s condor_config.headnode condor_config.master
ln -s condor_config.worker condor_config.worker1
ln -s condor_config.worker condor_config.worker2
```

Cada nodo del cluster debe tener un archivo condor_config.<hostname>, donde hostname es el nombre de la máquina en la red.

6. Modificar el archivo de configuración del sistema Condor de cada máquina, este se encuentra en el directorio /etc/condor/condor_config , con el fin de acceder a los archivos de configuración almacenados en el directorio compartido. En este archivo se debe modificar la línea LOCAL_CONFIG_FILE por:

```
LOCAL_CONFIG_FILE       = /nfs/condor/condor-etc/condor_config.cluster
```

Cabe anotar que este fue el archivo que se descargó y se modificó previamente en el numeral 3.

7. Crear una contraseña para el sistema Condor, ejecutando el siguiente comando.

```
condor_store_cred -c add
```

Esta contraseña debe ser la misma para todos los nodos del cluster.

8. Habilitar el inicio automático del sistema Condor al arrancar una máquina del cluster, para lo cual se utiliza el siguiente comando. Se recomienda realizar esta acción en todos los nodos del cluster.

```
chkconfig --level 235 condor on
```

9. Iniciar el sistema Condor, para lo cual se ejecuta el siguiente comando. Hacer esto para todos los nodos del cluster.

```
service condor start
```

10. Adicionar reglas de excepción en el firewall para el intervalo de puertos que maneja el sistema Condor. Introducir las siguientes líneas en el archivo /etc/sysconfig/iptables

```
-A RH-Firewall-1-INPUT  -s <network_address> -m state --state ESTABLISHED,NEW -p tcp -m tcp --dport 9000:10000 -j ACCEPT

-A RH-Firewall-1-INPUT  -s <network_address> -m state --state ESTABLISHED,NEW -p udp -m udp --dport 9000:10000 -j ACCEPT
```

11. Finalmente, probar la instalación del software Condor, ejecutando el siguiente comando en uno de los nodos del cluster. Este comando muestra los nodos pertenecientes al cluster y a su estado en el sistema.

```
condor_status
```

Bibliografía

[1] C. Gibas and P. Jambeck, *Developing Bioinformatics Computer Skills*, 1st ed. O'Reilly Media, 2001.

[2] I. Halperin, B. Ma, H. Wolfson, and R. Nussinov, "Principles of docking: An overview of search algorithms and a guide to scoring functions.," *Proteins*, vol. 47, no. 4, pp. 409–443, 2002.

[3] R. K. MURRAY, D. K. GRANNER, P. A. MAYES, and V. W. RODWELL, *Bioquímica de Harper*, 14th ed. El Manual Moderno, p. 878.

[4] H. A. Gabb, R. M. Jackson, and M. J. Sternberg, "Modelling protein docking using shape complementarity, electrostatics and biochemical information.," *J. Mol. Biol.*, vol. 272, no. 1, pp. 106–120, 1997.

[5] M. J. Sternberg, H. A. Gabb, and R. M. Jackson, "Predictive docking of protein-protein and protein-DNA complexes.," *Curr. Opin. Struct. Biol.*, vol. 8, no. 2, pp. 250–256, 1998.

[6] D. Voet and J. G. Voet, *Bioquímica*. Ed. Médica Panamericana, 2006, p. 1756.

[7] Y. Xu, D. Xu, and J. Liang, *Computational Methods for Protein Structure Prediction and Modeling: Volume 2: Structure Prediction*. Springer, 2010.

[8] R. C. Bohinski, *Bioquímica*, 5th ed. Pearson Educación, 1991.

[9] D. L. Nelson, A. L. Lehninger, and M. M. Cox, **PRINCIPIOS DE BIOQU{Í}MICA, 4/ED.* Ediciones Omega, S.A., 2005.

[10] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watson, *Biología Molecular de la Célula*, 3rd ed. Omega, 1999, p. 1450.

[11] A. Porto, "Formación de un enlace peptídico entre dos aminoácidos," 2012. [Online]. Available: http://commons.wikimedia.org/wiki/File:EnlacePeptidico.jpg. [Accessed: 15-Dec-2013].

[12] S. J. A. RUSSELL and P. A. NORVIG, *Inteligencia artificial: Un enfoque moderno*. Pearson Educaci{ó}n, 2004.

[13] J. M. Bahi, S. Contassot-Vivier, and R. Couturier, *Parallel Iterative Algorithms: From Sequential to Grid Computing*. Taylor & Francis, 2007.

[14] R. Nussinov and G. Schreiber, *Computational Protein-Protein Interactions*. Taylor & Francis, 2010.

[15] J.-C. Boisson, L. Jourdan, E.-G. Talbi, and D. Horvath, “Parallel multi-objective algorithms for the molecular docking problem,” *2008 IEEE Symp. Comput. Intell. Bioinforma. Comput. Biol.*, 2008.

[16] D. W. Ritchie, “Recent Progress and Future Directions in Protein-Protein Docking,” *Curr. Protein Pept. Sci.*, vol. 9, no. 1, pp. 1–15, 2008.

[17] G. R. Smith and M. J. Sternberg, “Prediction of protein-protein interactions by docking methods.,” *Curr. Opin. Struct. Biol.*, vol. 12, no. 1, pp. 28–35, 2002.

[18] G. Cui, T. Bibi, and Z. Wu, “MTMM -- A MATLAB Toolbox for Macromolecular Modeling.” p. 24, 2002.

[19] S. Vajda and C. J. Camacho, “Protein-protein docking: is the glass half-full or half-empty?,” *Trends Biotechnol.*, vol. 22, no. 3, pp. 110–116, 2004.

[20] R. Chen, L. Li, and Z. Weng, “ZDOCK: an initial-stage protein-docking algorithm.,” *Proteins*, vol. 52, no. 1, pp. 80–87, 2003.

[21] S. R. Comeau, D. W. Gatchell, S. Vajda, and C. J. Camacho, “ClusPro: a fully automated algorithm for protein-protein docking.,” *Nucleic Acids Res*, vol. 32, no. Web Server issue, pp. 96–99, 2004.

[22] J. I. Garzon, J. R. Lop�z-Blanco, C. Pons, J. Kovacs, R. Abagyan, J. Fernandez-Recio, and P. Chacon, “FRODOCK: a new approach for fast rotational protein�protein docking,” *Bioinformatics*, vol. 25, no. 19, pp. 2544–2551, 2009.

[23] D. Kozakov, R. Brenke, S. R. Comeau, and S. Vajda, “PIPER: an FFT-based protein docking program with pairwise potentials.,” *Proteins*, vol. 65, no. 2, pp. 392–406, 2006.

[24] G. Macindoe, L. Mavridis, V. Venkatraman, M.-D. Devignes, and D. W. Ritchie, “HexServer: an FFT-based protein docking server powered by

graphics processors," *Nucleic Acids Res.*, vol. 38, no. Web Server issue, pp. W445–W449, 2010.

[25] C. Bajaj, R. A. Chowdhury, and V. Siddavanahalli, "F3Dock: A Fast, Flexible and Fourier Based Approach to Protein-Protein Docking," *IEEEACM Trans. Comput. Biol. Bioinforma.*, pp. 1–20, 2009.

[26] Y. Liu, W. Li, Y. Wang, and M. Lv, "An Efficient Approach for Flexible Docking Base on Particle Swarm Optimization.," in *BMEI*, 2009, pp. 1–7.

[27] E. Atilgan and J. Hu, "Improving Protein Docking Using Sustainable Genetic Algorithms," *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.* , vol. 3, pp. 248–255, 2011.

[28] M. S. Apaydin, D. L. Brutlag, C. Guestrin, D. Hsu, and J.-C. Latombe, "Stochastic Conformational Roadmaps for Computing Ensemble Properties of Molecular Motion," in *Algorithmic Foundations of Robotics V*, 2004, pp. 131–147.

[29] A. M. J. J. Bonvin, "Flexible protein-protein docking.," *Curr. Opin. Struct. Biol.*, vol. 16, no. 2, pp. 194–200, 2006.

[30] C. Wang, P. Bradley, and D. Baker, "Protein-protein docking with backbone flexibility.," *J. Mol. Biol.*, vol. 373, no. 2, pp. 503–519, 2007.

[31] C. B-Rao, J. Subramanian, and S. D. Sharma, "Managing protein flexibility in docking and its applications.," *Drug Discov. Today*, vol. 14, no. 7–8, pp. 394–400, 2009.

[32] Y. Xu, D. Xu, and J. Liang, *Computational Methods for Protein Structure Prediction and Modeling: Volume 1: Basic Characterization*, no. v. 1. Springer, 2010.

[33] J. W. Ponder and D. A. Case, "Force fields for protein simulations," *Adv. Protein Chem.*, vol. 66, pp. 27–85, 2003.

[34] J. R. Bradford and D. R. Westhead, "Improved prediction of protein–protein binding sites using a support vector machines approach," *Bioinformatics*, vol. 21, no. 8, pp. 1487–1494, 2005.

[35] J. R. Bradford, C. J. Needham, A. J. Bulpitt, and D. R. Westhead, "Insights into Protein–Protein Interfaces using a Bayesian Network Prediction Method," *J. Mol. Biol.*, vol. 362, no. 2, pp. 365–386, 2006.

[36] B. Liu, X. Wang, L. Lin, B. Tang, Q. Dong, and X. Wang, “Prediction of protein binding sites in protein structures using hidden Markov support vector machine,” *BMC Bioinformatics*, vol. 10, no. 1, p. 381+, 2009.

[37] J. Janin, K. Henrick, J. Moult, L. T. Eyck, M. J. E. Sternberg, S. Vajda, I. Vakser, and S. J. Wodak, “CAPRI: A Critical Assessment of PRedicted Interactions,” *Proteins*, vol. 52, no. 1, pp. 2–9, 2003.

[38] R. Méndez, R. Leplae, L. De Maria, and S. J. Wodak, “Assessment of blind predictions of protein-protein interactions: current status of docking methods.,” *Proteins*, vol. 52, no. 1, pp. 51–67, 2003.

[39] H. Hwang, T. Vreven, J. Janin, and Z. Weng, “Protein–protein docking benchmark version 4.0,” *Proteins*, vol. 78, no. 15, pp. 3111–3114, 2010.

[40] C. A. Coello Coello, “Introducción a la Computación Evolutiva.” México, D.F., p. 310, 2008.

[41] J. J. Durillo and A. J. Nebro, “jMetal: A Java framework for multi-objective optimization,” *Adv. Eng. Softw.*, vol. 42, pp. 760–771, 2011.

[42] J. J. Durillo, A. J. Nebro, and E. Alba, “The jMetal Framework for Multi-Objective Optimization: Design and Architecture,” in *CEC 2010*, 2010, pp. 4138–4325.

[43] Z. Wu, “Structural Representation.” http://www.math.iastate.edu/wu/Math597HW0000/index.htm, p. 10.

[44] Y. Zhang, S. Finger, and S. Behrens, “Introduction to Mechanisms. Basic Kinematics of Constrained Rigid Bodies.” [Online]. Available: http://www.cs.cmu.edu/~rapidproto/mechanisms/chpt4.html. [Accessed: 16-Dec-2013].

[45] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[46] S. Mukherjee and Y. Zhang, “Protein-protein complex structure predictions by multimeric threading and template recombination.,” *Structure*, vol. 19, no. 7, pp. 955–966, 2011.

[47] J. E. Lloyd, “QuickHull3D: A Robust 3D Convex Hull Algorithm in Java.” .

[48] “convexHull.” [Online]. Available: http://xlr8r.info/mPower/pages/convexHull.html. [Accessed: 16-Dec-2013].

[49] J. W. Ponder, “TINKER - Software Tools for Molecular Design.” Department of Chemistry Washington University in Saint Louis, Saint Louis.

[50] D. C. Becerra Romero, “A Multi-objective Ab-initio Model for Protein Folding Prediction at an Atomic Conformation Level,” NATIONAL UNIVERSITY OF COLOMBIA, 2010.

[51] R. L. Dunbrack and F. E. Cohen, “Bayesian statistical analysis of protein side-chain rotamer preferences.,” *Protein Sci.*, vol. 6, no. 8, pp. 1661–1681, 1997.

[52] S. C. Lovell, J. M. Word, J. S. Richardson, and D. C. Richardson, “The penultimate rotamer library.,” *Proteins*, vol. 40, no. 3, pp. 389–408, 2000.

[53] L. Vogel, “Java concurrency (multi-threading) - tutorial ,” 2008. [Online]. Available: http://www.vogella.com/articles/JavaConcurrency/article.html. [Accessed: 15-Dec-2013].

[54] P. P. M. María Victoria Bueno Delgado, “Estudio, configuración y prueba de un entorno de computación Condor,” Universidad Politécnica de Cartagena, 2002.

[55] U. of W. Condor Team, “Condor Version 7.6.4 Manual.” University of Wisconsin–Madison, p. 1014, 2011.

[56] H. Nakada, “Java Interface for Condor.” 2008.

[57] W. Humphrey, A. Dalke, and K. Schulten, “VMD: visual molecular dynamics.,” *J. Mol. Graph.*, vol. 14, no. 1, pp. 33–38, 27–28, 1996.

[58] Christopher Smith, “Setting Up an NFS Server,” 2006. [Online]. Available: http://nfs.sourceforge.net/nfs-howto/ar01s03.html. [Accessed: 11-Sep-2013].