Tesis defendida por

David Omar Rodríguez Uribe

y aprobada por el siguiente Comité

Dr. Carlos Alberto Brizuela Rodríguez

Director del Comité

Dr. Israel Marck Martínez Pérez

Miembro del Comité

Dr. José Alberto Fernández Zepeda

Miembro del Comité

Dra. Clara Elizabeth Galindo Sánchez

Miembro del Comité

José Antonio García Macías

Coordinador del Programa de
Posgrado en Ciencias de la Computación

Dr. Jesús Favela Vara

Director de la Dirección
de Estudios de Posgrado

Febrero de 2014

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA, BAJA CALIFORNIA

Programa de Posgrado en Ciencias
en Ciencias de la Computación

Diseño de heurísticas para el problema de empaquetamiento de la cadena lateral en proteínas

Tesis
para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:
David Omar Rodríguez Uribe

Ensenada, Baja California, México
2014

Resumen de la tesis de David Omar Rodríguez Uribe, presentada como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación .

Diseño de heurísticas para el problema de empaquetamiento de la cadena lateral en proteínas

Resumen aprobado por:

Dr. Carlos Alberto Brizuela Rodríguez

Director de Tesis

Uno de los problemas más relevantes aún no resueltos en bioinformática estructural es el de predecir la estructura tridimensional de una proteína a partir de su secuencia de aminoácidos. La relevancia está en que la estructura de una proteína está directamente relacionada con sus funciones de bajo nivel. Actualmente, se conoce millones de secuencias de proteínas de las cuales sólo una fracción muy pequeña, en el orden de decenas de miles, tienen asociada una estructura. La manera de determinar la estructura es a través de cristalografía de rayos X o por resonancia magnética nuclear. Sin embargo, obtener la estructura de esta manera es costoso en tiempo y recursos, por lo que sería de gran ayuda contar con métodos computacionales que puedan predecir en forma precisa las estructuras.

El modelado homólogo es un enfoque de predicción que usa información de estructuras conocidas de proteínas que tienen un parecido en secuencia con la proteína de interés. Un desafío importante para la predicción exitosa con el modelado homólogo tiene que ver con el problema conocido como empaquetamiento de la cadena lateral, el cual es el tema central de este trabajo.

El empaquetamiento de la cadena lateral consiste en determinar las coordenadas espaciales de cada uno de los átomos de las cadenas laterales de una proteína, dada la secuencia de aminoácidos y las coordenadas de los átomos de su columna vertebral.

Los métodos para resolver este problema tienen tres componentes principales: una biblioteca de rotámeros, una función de energía y un algoritmo que busca la combinación de rotámeros que minimice la función de energía. Los métodos más precisos de la literatura proponen bibliotecas de rotámeros cada vez más complejas. Sin embargo, trabajos recientes demostraron que es posible obtener resultados superiores a los del estado del arte incluso con una biblioteca sencilla de rotámeros.

En este trabajo se propusieron tres algoritmos para abordar el problema de empaquetamiento. Los algoritmos usan una biblioteca sencilla de rotámeros y están basados en heurísticas del tipo: búsqueda tabú, recocido simulado y búsqueda local, todos navegan a través del espacio de soluciones minimizando una función de energía. Estos algoritmos se combinaron con un algoritmo voraz el cual se enfoca en reducir el número de colisiones atómicas. Con el objetivo de hacer eficiente el cómputo de la función de energía empleada en los métodos propuestos, se utilizaron estructuras de datos especiales. Por otro lado para reducir el tamaño del espacio de soluciones a ser explorado por estos algoritmos se implementó la estrategia conocida como Eliminación de Callejones sin Salida.

Se realizaron experimentos computacionales para evaluar el desempeño de los algoritmos propuestos sobre tres conjuntos de casos de prueba. En estos experimentos se obtuvo una precisión promedio de 82 % para la predicción del primer ángulo de torsión (χ_1) y 62 % para el segundo (χ_{1+2}). Dichos resultados no superaron el 85 % en χ_1 que logran los algoritmos

del estado del arte. Sin embargo, los algoritmos propuestos son efectivos para minimizar la función de energía empleada. Por otro lado la búsqueda voraz, guiada por el número de colisiones, logró reducir en un 70 % en promedio las colisiones de las estructuras predichas. Un resultado importante es que los términos de la función de energía implementados no consideran a las estructuras depositadas en la base de datos de proteínas como óptimos locales, por consiguiente no será posible que los métodos de optimización guiados por estos términos converjan a dichas estructuras. Este resultado motiva a que sea la función de energía el próximo punto a analizar con el objeto de superar el nivel de precisión logrado por los mejores métodos actuales.

Palabras Clave: **Estructura de proteínas, Problema de empaquetamiento de la cadena lateral en proteínas, Heurísticas, Bioinformática.**

Abstract of the thesis presented by David Omar Rodríguez Uribe, in partial fulfillment of the requirements of the Master in Sciences degree in Computer Science.

Design of heuristics for the protein side-chain packing problem

Abstract approved by:

Dr. Carlos Alberto Brizuela Rodríguez

Thesis Director

One of the most important open problems in structural bioinformatics is to predict the three-dimensional structure of a protein from its amino acid sequence. The significance lies in the fact that the protein's structure is directly related to its low-level functions. Currently, there are millions of known protein sequences, but only a small fraction of these, in the order of tens of thousands, have a known structure. The most reliable way of obtaining the three-dimensional structure of a protein is through experimental methods such as X-ray crystallography or nuclear magnetic resonance. However, obtaining the structure this way is costly in time and resources, so it would be helpful to have computational methods that can accurately predict the protein structure.

Homology modeling is a structure prediction approach that uses the information of known protein structures which have a similarity in sequence with the protein of interest. A major challenge for the successful prediction in homology modeling has to do with the problem known as side-chain packing, which is the focus of this work.

The side-chain packing problem consists in determining the spatial coordinates of each of the atoms in the side-chains of a protein, given the amino acid sequence and the coordinates of the atoms of its backbone.

The methods to solve this problem have three main components: a rotamer library, an energy function, and an algorithm that searches for the rotamer combination that minimizes the energy function. The most accurate methods in the literature suggest rotamer libraries increasingly complex. However, recent studies showed that it is possible to obtain better results than those of the state of the art using a simple rotamer library.

In this work three algorithms were proposed to address the side-chain packing problem. The algorithms use a simple rotamer library and they are based on heuristics of the following types: tabu search, simulated annealing and local search; all of them are guided through the solution space minimizing an energy function. These algorithms are combined with a greedy algorithm which focuses on reducing the number of atomic collisions. To calculate the energy function efficiently, the use of some special data structures is proposed. In addition, to reduce the solution space a *Dead-End Elimination* algorithm was implemented.

To evaluate the performance of the implemented algorithms they were tested using three data sets. In these experiments an average accuracy of 82 % was obtained for the first torsion angle (χ_1) and 62 % for the second (χ_{1+2}). This accuracy did not outperform the 85 % for χ_1 achieved by the state of the art algorithms. However, the proposed algorithms are effective minimizing the implemented energy function. Furthermore, the greedy algorithm was able to reduce by 70 % on average the number of collisions of the predicted structures. An important result is that the terms of the implemented energy function do not consider the structures in the protein data bank as local minima, thus it will not be possible for the optimization

methods guided by these terms converge to such structures. This result encourages a deeper analysis of the energy functions in order to improve the accuracy level achieved by the best current methods.

Keywords: **Protein structure, Protein side-chain packing problem, Heuristics, Bioinformatics.**

Dedicatoria

A mis padres y hermanos.

Agradecimientos

Agradezco a mi familia, por todo su amor y apoyo incondicional.

A mi director de tesis, el Dr. Carlos Alberto Brizuela Rodríguez, quien despertó mi interés por el área de la bioinformática. Gracias por su guía y enseñanzas.

A los miembros de mi comité de tesis, el Dr. Israel Marck Martínez Pérez, el Dr. José Alberto Fernández Zepeda y la Dra. Clara Elizabeth Galindo Sánchez, por su tiempo, observaciones y sugerencias durante el desarrollo de este trabajo.

A los investigadores y compañeros del posgrado del departamento de Ciencias de la Computación, por sus enseñanzas y experiencias durante esta estancia.

A mis amigos, en especial a Hugo, Shadai, Najash, Ricardo, Marina, Carlos y Juan, quienes me apoyaron en todo momento, y a la Maestra Claudia Quezada, gracias por animarme a entrar en el campo de la investigación.

A CICESE por permitirme estudiar este posgrado, y al personal administrativo por la excelente atención que siempre me brindaron.

Al CONACyT por su apoyo económico para realizar mis estudios.

Contenido

Página

Página

Página

Lista de figuras

Figura Página

Lista de tablas

Lista de algoritmos

Capítulo 1

Introducción

1.1. Antecedentes y motivación

La biocomputación es un área interdisciplinaria que utiliza conocimientos de biología, ciencias de la computación, bioquímica, estadística, entre otras, para responder preguntas en biología. Esta disciplina tiene un papel esencial tanto resolviendo problemas de genómica y proteómica, así como organizando y visualizando la información recolectada de un gran número de experimentos biológicos (Kanehisa y Bork, 2003).

Uno de los problemas importantes dentro de la biología molecular es la predicción de la estructura de una proteína, ya que la función que la misma realiza está relacionada con su estructura tridimensional, y al mismo tiempo, la estructura de una proteína depende de la secuencia de aminoácidos que la conforman (Corona de la Fuente, 2010). El proceso de obtener la secuencia de una proteína es más sencillo que el de determinar su estructura. Actualmente, el número de secuencias de proteínas es mayor a los 50 millones (UniProt, 2014); de las cuales, menos de 100 mil tienen una estructura asociada (PDB, 2014).

Actualmente, la manera más confiable de obtener la estructura tridimensional de una proteína es a través de métodos experimentales de laboratorio, como lo son el método de cristalografía por rayos X y el método de resonancia magnética nuclear (Gu y Bourne, 2009). Sin embargo, obtener la estructura de manera experimental es un proceso que consume tiempo y dinero, es por este motivo que surge la necesidad de diseñar métodos computacionales para predecir la estructura de proteínas a partir de su secuencia de aminoácidos.

Solucionar este problema es de gran importancia ya que tiene aplicación en áreas como el diseño de fármacos, el análisis de función de proteínas, y el diseño de proteínas con mejores o incluso nuevas propiedades.

Existen tres métodos *in-silico** de predición de estructura de proteínas, los cuales son: modelado homólogo (Rodriguez *et al.*, 1998), reconocimiento de plegado (Bryant y Altschul, 1995) y *ab initio* (Osguthorpe, 2000). Este trabajo se enfoca en un subproblema del modelado homólogo, denominado problema de empaquetamiento de la cadena lateral en proteínas (PSCPP por sus siglas en inglés). Éste consiste en predecir las coordenadas tridimensionales

**in-silico*: hecho por computadora.

de la proteína dada la secuencia de aminoácidos que la conforman y las coordenadas de la columna vertebral de la misma. Akutsu (1997) demostró que este problema es NP-Difícil, lo que implica que no se conoce ningún algoritmo de tiempo polinomial que lo pueda resolver, y la única alternativa exacta conocida requiere de la evaluación de un número exponencial de soluciones.

A la fecha existen varios métodos computacionales que abordan el PSCPP, algunos de los más destacados son:

- SCWRL4 (Krivov *et al.*, 2009) se basa en modelar las interacciones de la proteína como un grafo, el cual resuelve mediante algoritmos como el *Tree Decomposition* (Xu, 2005). Este algoritmo utiliza una biblioteca de rotámeros dependiente de la columna vertebral desarrollada por Shapovalov y Dunbrack (2011).

- OPUS-Rota (Lu y Dousis, 2008) implementa un nuevo término a la función de energía, el cual es un potencial estadístico para determinar la preferencia de orientación de las cadenas laterales. OPUS-Rota utiliza un algoritmo de recocido simulado por baño de calor de Monte Carlo (Newman y Barkema, 1999) para llevar a cabo la optimización, y una biblioteca de rotámeros dependiente de la columna vertebral propuesta por Dunbrack y Karplus (1993).

- CIS-RR (Cao *et al.*, 2011) se enfoca en reducir el número de colisiones entre átomos al momento de predecir la estructura de la proteína. Una colisión entre una pareja de átomos ocurre cuando la distancia entre ambos es menor al 60 % de la suma de sus radios de Van der Waals (Miao *et al.*, 2011). Este método también utiliza una biblioteca de rotámeros dependiente de la columna vertebral en su versión del 2002 (Dunbrack y Cohen, 1997; Dunbrack y Karplus, 1993).

- RASP (Miao *et al.*, 2011) de manera similar al CIS-RR, se enfoca en generar estructuras evitando las colisiones entre átomos; sin embargo, este algoritmo primero genera una estructura de buena calidad utilizando distintos algoritmos como por ejemplo de ramificación y terminación (Gordon y Mayo, 1999), búsqueda Monte Carlo (Liu, 2008) y de marcha atrás (Tarjan, 1971). Este algoritmo utiliza una biblioteca de rotámeros dependiente de la columna vertebral propuesta por Dunbrack y Cohen (1997).

Estos algoritmos obtienen una precisión de aproximadamente 85 % para $\chi_1(\%)$ y 75 % para $\chi_{1+2}(\%)$. Sin embargo, en trabajos recientes (Corona de la Fuente, 2010; Lezcano, 2012) se ha demostrado que usando una biblioteca sencilla de rotámeros independiente de la columna vertebral, es posible lograr una precisión de al menos 97.229 % para $\chi_1(\%)$. Este hecho motiva a usar esta biblioteca de rotámeros sencilla y enfocarse ya sea en el algoritmo de optimización y/o en la función de energía.

Con esta motivación a continuación se define una versión acotada del problema y se plantean los objetivos de investigación de este trabajo.

1.2. Definición del problema

El problema de empaquetamiento de la cadena lateral en proteínas consiste en predecir las coordenadas tridimensionales de todos los átomos que conforman la cadena lateral de cada uno de los aminoácidos de una proteína, conociendo la secuencia de aminoácidos que conforman la proteína y las coordenadas de los átomos de la columna vertebral de la misma.

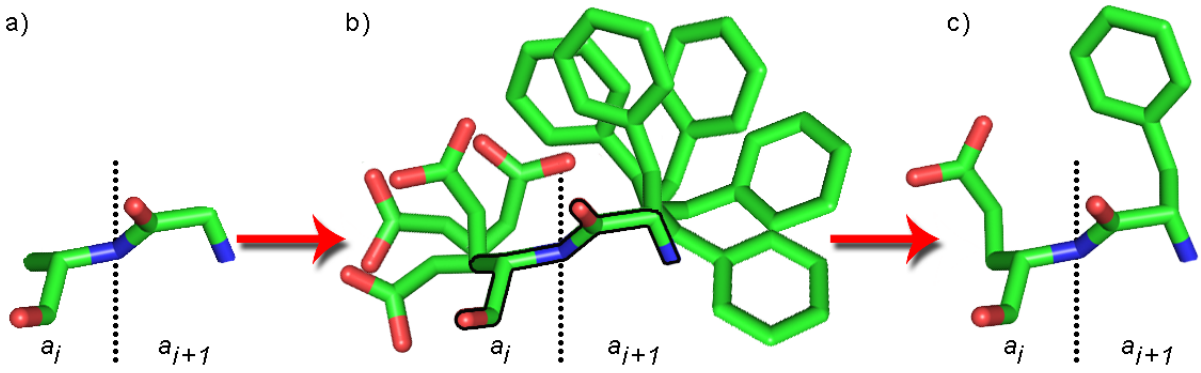

Figura 1: Ejemplo de empaquetamiento de la cadena lateral en proteínas
a) Columna vertebral de la proteína. **b)** Rotámeros para cada aminoácido. **c)** Estructura predicha.

En computación, este problema se modela como uno de optimización combinatoria, utilizando una biblioteca de rotámeros que define el espacio de búsqueda y una función de energía que se debe minimizar. La biblioteca de rotámeros es una colección estadística de distintas conformaciones que pueden tomar las cadenas laterales de los aminoácidos, esta biblioteca se genera utilizando información de estructuras conocidas de proteínas. La función de energía a minimizar es una aproximación de la manera en la que los átomos de la proteína interactúan entre si. Con base en esto, para solucionar este problema se desea encontrar la asignación

de rotámeros, para cada uno de los aminoácidos de la proteína, que genere la menor energía posible. En la Figura 1 se muestra un ejemplo que ilustra este problema. En **a)** se muestra la columna vertebral de la proteína, en este caso del aminoácido a_i y el a_{i+1}, la cual se recibe como dato de entrada. En **b)** se muestra la columna vertebral (delimitada por un contorno negro) junto con las conformaciones de la cadena lateral (rotámeros) de cada aminoácido. En **c)** se muestra la estructura predicha, la cual se conforma por la columna vertebral junto con una conformación de la cadena lateral para cada aminoácido. La energía de la estructura en **c)** debe ser la menor de todas las estructuras posibles formadas por todas las combinaciones de los rotámeros mostrados en **b)**.

1.3. Objetivos de la investigación

1.3.1. Objetivo general

Diseñar algoritmos heurísticos basados en vecindarios aplicados al problema del empaquetamiento de la cadena lateral en proteínas, y evaluar sus desempeños comparándolos con los algoritmos del estado del arte.

1.3.2. Objetivos específicos

- Proponer una manera optimizada de calcular la función de energía, para evitar cálculos de orden $O(n^2)$ en una proteína con n átomos.
- Analizar distintas heurísticas y el algoritmo *Dead-End Elimination* que ayuden a reducir el espacio de soluciones combinatorio.
- Analizar distintos términos de funciones de energía para utilizar aquellos que den una mejor precisión sin impactar de forma considerable en el tiempo de ejecución.
- Proponer un algoritmo híbrido resultado del análisis de los objetivos anteriores.
- Implementar el algoritmo propuesto y comparar sus resultados con los obtenidos por los algoritmos actuales para el PSCPP.

1.4. Metodología de solución propuesta

Para abordar este problema, se optó primero por encontrar una manera eficiente de calcular la energía de una proteína, evitando cálculos de orden $O(n^2)$ para una proteína con n átomos, ya que la evaluación de la energía es uno de los procesos que consume más tiempo computacional para el PSCPP. La energía de interacción entre una pareja de átomos depende de la distancia a la que ambos se encuentran; sin embargo, después de un cierto umbral de distancia la energía de interacción se vuelve despreciable. Por lo tanto, no es necesario calcular la energía de un átomo dado con respecto a cada uno de los átomos de la proteína, sino que sería más eficiente considerar únicamente los átomos que se encuentren dentro de cierto radio de interacción. Dentro de este radio es más probable que se encuentren átomos con los cuales la energía de interacción es relevante. Para lograr esto, se decidió generar para cada aminoácido i una lista que contenga el índice de los aminoácidos cercanos a él, y así sólo calcular la energía de interacción contra los átomos de los aminoácidos en dicha lista. Para generar estas listas de aminoácidos cercanos se creó un grid tridimensional, en donde a cada átomo se le asignó una celda de acuerdo a sus coordenadas tridimensionales. Posteriormente, a cada aminoácido se le asignó la misma celda que su átomo C^{β}, ya que este átomo es el que se localiza más al centro de la estructura de la mayoría de los aminoácidos. Una vez que cada átomo y aminoácido conoce la celda a la que pertenece, para cada aminoácido i se obtienen los aminoácidos que se encuentran dentro de su misma celda y en las celdas adyacentes, generando así la lista de aminoácidos cercanos a i. Utilizando este grid se logró que las listas de aminoácidos cercanos tuvieran un tamaño de $O(1)$.

Otra manera en la que se puede reducir el tiempo de ejecución del algoritmo consiste en implementar alguna heurística o método exacto que permita reducir el espacio de soluciones combinatorio. Para esto, se analizaron algunos de los algoritmos más recientes del estado del arte (SCWRL4 (Krivov *et al.*, 2009), OPUS-Rota (Lu y Dousis, 2008), CIS-RR (Cao *et al.*, 2011) y RASP (Miao *et al.*, 2011)). Con base en este análisis, se encontró que la mayoría de los algoritmos mencionados implementan una misma estrategia para disminuir el espacio de soluciones, la cual se denomina **eliminación de callejones sin salida** (Desmet *et al.*, 1992).

El propósito de los algoritmos de eliminación de callejones sin salida (DEE, por sus siglas en inglés) consiste en eliminar los rotámeros que no pueden formar parte de la conformación de

energía mínima global (GMEC) de la proteína. Este tipo de estrategias son una herramienta poderosa para la optimización combinatoria de la colocación de cadenas laterales en el diseño de proteínas y en el modelado homólogo (Looger y Hellinga, 2001).

Una vez que se logró reducir el espacio de búsqueda y optimizar el cálculo de la función de energía, el siguiente paso consistió en seleccionar una heurística para llevar a cabo el proceso de optimización. Como primer criterio para reducir las opciones de heurísticas se limitó la búsqueda a heurísticas para espacios de solución discretos, esto debido a que se utilizarán bibliotecas de rotámeros para generar las conformaciones de la cadena lateral.

La heurística que se seleccionó fue la búsqueda tabú, ya que este tipo de heurística ha tenido éxito en problemas combinatorios (Aarts y Lenstra, 1997). Existen más heurísticas que cumplen con los criterios de selección; sin embargo, actualmente existen algunos algoritmos en el estado del arte que implementan el método de recocido simulado, el cual ha sido un adversario común para la búsqueda tabú.

La búsqueda tabú fue la primer heurística implementada y fue a la que se le dedicó mayor tiempo de desarrollo en esta tesis; sin embargo, posteriormente se decidió implementar también heurísticas tipo recocido simulado y búsqueda local. El algoritmo de búsqueda local se implementó con el propósito de contar con un algoritmo sencillo y rápido, el cual garantice que la estructura predicha sea un óptimo local para la función de energía implementada.

Una vez que se concluyó con la implementación de las heurísticas, se procedió con la experimentación. Para esto se utilizaron tres conjuntos de pruebas con tamaños de 65, 373 y 723 proteínas, respectivamente.

1.5. Organización de la tesis

El presente trabajo está organizado de la siguiente manera:

En el Capítulo 2 se presenta una breve descripción de las proteínas y la bioinformática, brindando conceptos que son necesarios para la comprensión del problema tratado en este trabajo. También se incluye la definición del problema, así como la descripción de algunos de los algoritmos del estado del arte para el mismo.

En el Capítulo 3 se describen las estrategias que se utilizaron en los algoritmos implementados, junto con información referente a la función de energía y biblioteca de rotámeros utilizada.

En el Capítulo 4 se presentan los casos de prueba utilizados y los resultados obtenidos por los algoritmos implementados, junto con un análisis y una comparación con los métodos del estado del arte para el PSCPP.

En el Capítulo 5 se exponen las conclusiones a las que se llegó, así como también propuestas para la continuación de este trabajo de investigación.

En el Apéndice A.1 se describe de manera detallada el funcionamiento de la tabla hash implementada, la cual se menciona en la Sección 3.1.3.

En el Apéndice A.2 se incluye la lista de las proteínas que conforma cada uno de los casos de prueba utilizados.

En el Apéndice A.3, se presenta la calidad de las estructuras obtenidas por el algoritmo de recocido simulado utilizando distintos esquemas de hibridación.

En el Apéndice A.4, se presenta una comparación entre las estructuras de las proteínas antes y después de la optimización de la energía.

Capítulo 2

Marco teórico

2.1. Conceptos básicos relevantes

2.1.1. Proteínas

Una proteína es una macromolécula que está conformada por cadenas de aminoácidos (llamadas péptidos), las cuales están unidas por enlaces peptídicos, por lo que a las proteínas se les conoce también como cadenas polipeptídicas. Las proteínas están compuestas de una cadena de aminoácidos, existen 20 tipos de aminoácidos posibles, que componen dicha cadena.

Las proteínas son responsables de catalizar y regular reacciones bioquímicas, también forman la base para componentes celulares y tejidos como la piel, el cabello y los tendones. Las proteínas son producto de los procesos conocidos como transcripción y traducción, en donde la información en el ADN se transcribe a ARN, el cual se traduce después a una secuencia de aminoácidos, y finalmente esta cadena de aminoácidos se pliega produciendo una proteína. Este proceso se realiza en la célula en tiempos que van desde una fracción de segundos hasta unos pocos minutos (Chan y Dill, 1993).

2.1.2. Aminoácidos

Los 20 aminoácidos que componen las proteínas tienen una estructura parecida entre ellos, la cual consiste de un grupo amino, un grupo carboxilo y una cadena lateral. La cadena lateral es la que hace diferente a un aminoácido de otro y le otorga sus características químicas únicas. La cadena lateral de un aminoácido se encuentra unida a un átomo de carbono que se encuentra ubicado en el centro del aminoácido, es decir, entre el grupo amino y el grupo carboxilo. Este átomo de carbono central se denomina Carbono-α o simplemente C^{α}. En la Figura 2 se muestra un diagrama ilustrativo de la estructura de un aminoácido.

La cadena lateral de un aminoácido es una secuencia de átomos, los cuales se etiquetan según la distancia que tengan con respecto al C^{α}, usando las letras griegas: β, γ, δ, ϵ, η y ι, siendo el átomo etiquetado con β el más cercano al C^{α}; además, si dos átomos se encuentran a la misma distancia entonces se etiquetan con números arábigos (Corona de la Fuente, 2010).

Se denomina como la **columna vertebral** (BB, Backbone) de una proteína al conjunto

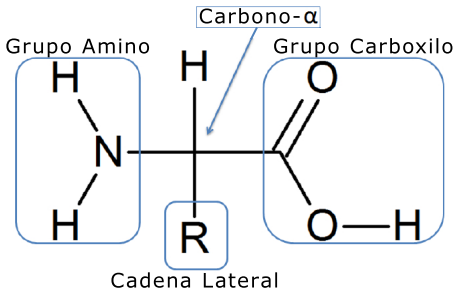

Figura 2: Estructura general de un aminoácido (Corona de la Fuente, 2010, p. 10).

de átomos pertenecientes al grupo amino (-NH_2)*, grupo carboxilo (-COOH)* y el carbono-α de todos los aminoácidos que constituyen dicha proteína.

2.1.3. Ángulos de torsión

Los ángulos de torsión de un aminoácido se agrupan en dos categorías, los ángulos de torsión de la columna vertebral y los ángulos de torsión de la cadena lateral.

Los ángulos de torsión de la columna vertebral se nombran como: ϕ, ψ y ω; mientras que los de la cadena lateral se nombran como: χ_1, χ_2, χ_3, χ_4 y χ_5. Si bien todos los aminoácidos cuentan con los tres ángulos de torsión de la columna vertebral, no todos tienen los cinco ángulos de torsión en la cadena lateral. Esto se debe a que el número de ángulos de torsión de la cadena lateral depende del aminoácido en cuestión. En la Figura 3 se muestra un fragmento de una proteína, sobre la cual se indican los ángulos de torsión de la columna vertebral, así como los átomos con los cuales se forman dichos ángulos.

Los valores de los ángulos de torsión se obtienen utilizando cuatro puntos en el espacio $(\mathbf{i}, \mathbf{j}, \mathbf{k}, \mathbf{l})$, en donde cada uno de estos puntos representa un átomo del aminoácido. Los puntos $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ definen al plano A, y los puntos $(\mathbf{j}, \mathbf{k}, \mathbf{l})$ definen al plano B. El ángulo de torsión φ_{ijkl} formado entre los planos A y B (Figura 4) se puede calcular como:

$$\varphi_{ijkl} = cos^{-1}\left((\hat{\mathbf{n}}_A) \cdot (\hat{\mathbf{n}}_B)\right) \tag{1}$$

En donde $\hat{\mathbf{n}}_A$ y $\hat{\mathbf{n}}_B$ son los vectores normales unitarios de los planos A y B. Para calcular dichos vectores unitarios es necesario formar dos vectores para cada plano, para esto utilizamos las coordenadas de los átomos que definen dicho plano. Por ejemplo, en el caso del plano

* N=nitrógeno, H=hidrógeno, C=carbono y O=oxígeno.

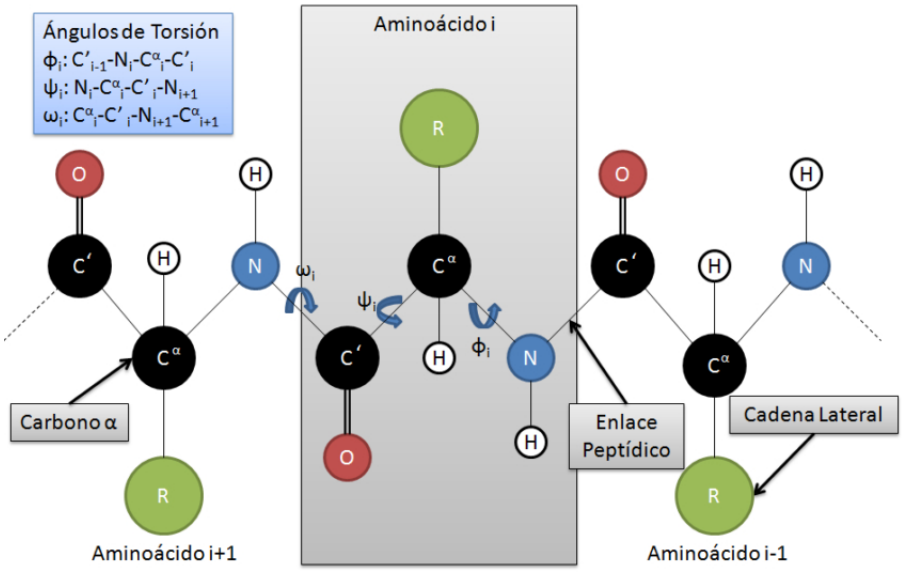

Figura 3: Ángulos de torsión de la columna vertebral (Corona de la Fuente, 2010, p. 16).

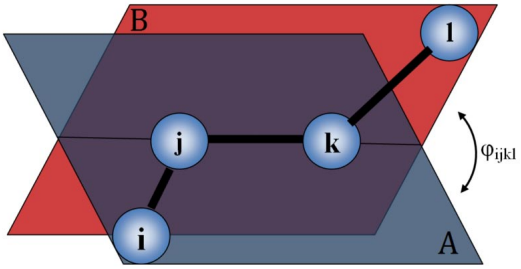

Figura 4: Los puntos $(\mathbf{i}, \mathbf{j}, \mathbf{k}, \mathbf{l})$ definen dos planos, el plano A está formado por los puntos $(\mathbf{i}, \mathbf{j}, \mathbf{k})$, y el plano B está formado por los puntos $(\mathbf{j}, \mathbf{k}, \mathbf{l})$. El ángulo de torsión φ_{ijkl} es el ángulo formado entre el plano A y el plano B (Corona de la Fuente, 2010, p. 16).

A, si cada átomo lo tratamos como un vector nulo podemos crear los vectores $\overrightarrow{v_0} = \overrightarrow{j} - \overrightarrow{i}$ y $\overrightarrow{v_1} = \overrightarrow{k} - \overrightarrow{j}$; con estos dos vectores obtenemos un vector normal al plano A ($\mathbf{n}_A$) simplemente calculando el producto cruz de ambos vectores. Una vez que se tiene un vector normal, si se divide cada uno los componentes de $\mathbf{n}_A$ entre $|\mathbf{n}_A|$, que representa el módulo del vector $\mathbf{n}_A$, se obtiene el vector normal unitario $\hat{\mathbf{n}}_A$. Se sigue el mismo procedimiento para calcular $\hat{\mathbf{n}}_B$, utilizando los vectores $\overrightarrow{v_1} = \overrightarrow{k} - \overrightarrow{j}$ y $\overrightarrow{v_2} = \overrightarrow{l} - \overrightarrow{k}$.

2.1.4. Rotámero

El término rotámero proviene de isómero rotacional. Los isómeros son compuestos químicos con la misma composición atómica pero diferente estructura, por lo que un rotámero se

define como una posible conformación de la cadena lateral de un aminoácido. Los rotámeros se obtienen utilizando proteínas con estructura conocida, es decir, proteínas que fueron sometidas a métodos de laboratorio para obtener su estructura tridimensional, como lo son el método de cristalografía por rayos X (Gu y Bourne, 2009) y el método de resonancia magnética nuclear (Gu y Bourne, 2009). La cristalografía por rayos X es la técnica más utilizada para determinar la estructura de proteínas y otras macromoléculas biológicas (Smyth, 2000). El proceso que se sigue durante la cristalografía por rayos X para la predicción de estructura en proteínas consiste en tomar una muestra purificada con altas concentraciones de una proteína y cristalizar dicha muestra, después el cristal se expone a rayos X. Con esto se producen ciertos patrones de difracción, los cuales se analizan para poder producir así un mapa de densidad de electrones sobre el cual se trata de encajar la estructura de la proteína de manera que adopte una conformación termodinámicamente favorable (Smyth, 2000).

Los rotámeros se almacenan en bibliotecas de rotámeros, las cuales son colecciones estadísticas de rotámeros. Las bibliotecas de rotámeros pueden ser dependientes o independientes de la columna vertebral (Dunbrack y Karplus, 1993).

Si la biblioteca es independiente de la columna vertebral, entonces los rotámeros se agrupan por tipo de aminoácido, para cada uno de los 18 aminoácidos que cuentan con una cadena lateral. Cada rotámero de la biblioteca incluye la frecuencia con la que aparece esa conformación de la cadena lateral en el conjunto de proteínas utilizado para generar la biblioteca de rotámeros; también incluye el valor de los ángulos de torsión asociados al rotámero, los cuales se encuentran en el intervalo $[-180°, 180°]$.

En el caso de las bibliotecas dependientes de la columna vertebral, además de contar con un conjunto de rotámeros para cada tipo de aminoácido, estos rotámeros también se agrupan de acuerdo a los ángulos de torsión de la columna vertebral ϕ y ψ del aminoácido.

2.1.5. Función de energía

En el PSCPP la función de energía es un modelo que aproxima la energía resultante de las interacciones entre los átomos de una proteína, es decir, su energía libre. Existen diferentes tipos de fuerzas de interacción entre los átomos de una proteína, por ejemplo, en CHARMM (Brooks *et al.*, 1983) la función de energía implementa los siguientes términos: potencial de enlaces, potencial del ángulo de enlace, potencial de ángulos de torsión, torsiones

impropias, potencial de Van der Waals, potencial electrostático y un potencial para puentes de hidrógeno. Las distintas fuerzas de interacción ejercidas por los átomos de una proteína se pueden clasificar en cinco categorías (Gordon *et al.*, 1999): (1) energía entre átomos no unidos por un enlace covalente; (2) interacciones polares que no provienen de un enlace; (3) energía de coordenadas internas; (4) energía de solvatación; (5) energía de entropía.

En las funciones de energía de los algoritmos del estado del arte para el PSCPP, el potencial de van der Waals (Winterton, 1970) es el término que más ampliamente se utiliza (Gordon *et al.*, 1999), generalmente como una variante de la expresión 12-6 de Lennard-Jones, la cual es un modelo matemático para representar la atracción y repulsión entre átomos (Lennard-Jones, 1925), o como una función por partes que asigna un valor de energía según el cociente entre la distancia entre dos átomos y la suma de sus radios de interacción.

Los términos utilizados en la función de energía pueden variar de un autor a otro; sin embargo, lo que se espera de una función de energía en el PSCPP es que proporcione una buena precisión sin que el costo computacional sea demasiado elevado, motivo por el cual es deseable que los términos que se implementen en la función de energía se puedan calcular entre parejas de átomos, ya que este tipo de términos requieren menos tiempo computacional para calcularse.

2.2. Métodos *in-silico* de predicción de estructura

Los tres enfoques principales *in-silico* de predicción de estructura de proteínas son: modelado homólogo (Rodriguez *et al.*, 1998), reconocimiento de plegado (Bryant y Altschul, 1995) y *ab initio* (Osguthorpe, 2000). El objetivo final de estos métodos es poder predecir la estructura de la proteína a partir de la secuencia de aminoácidos que la conforma, con una precisión comparable a la de los resultados obtenidos experimentalmente. Esto permitiría que se utilizaran con seguridad los modelos de proteínas generados rápidamente por métodos computacionales en campos donde actualmente solo los métodos experimentales se desempeñan adecuadamente, como lo son: diseño de fármacos basado en estructura, análisis de función de proteínas, comportamiento antígeno, y diseño de proteínas con mejor estabilidad o nuevas funciones (Bourne y Weissig, 2003). Además, en algunos casos el modelado de estructuras es la única manera de obtener información estructural si fallan las técnicas experimentales, ya que algunas proteínas son demasiado grandes para el análisis por resonancia

magnética nuclear, y no se pueden cristalizar para realizar una difracción por rayos X (Gu y Bourne, 2009).

A continuación se explican brevemente los métodos *ab initio* y reconocimiento de plegado, seguidos de una descripción más detallada del modelado homólogo, ya que uno de los pasos de dicho método es el problema abordado en este trabajo de investigación.

2.2.1. *Ab initio*

La predicción de estructura por *ab initio* busca predecir la estructura nativa de la proteína a partir de su secuencia de aminoácidos. Un método conocido como Rosetta (Rohl *et al.*, 2004) se basa en utilizar pequeños fragmentos de la proteína, minimizando la energía de su estructura local. Posteriormente se orientan dichos fragmentos con el fin de encontrar una manera en la que la energía total de la proteína sea mínima.

2.2.2. Reconocimiento de plegado

Reconocimiento de plegado se utiliza para modelar aquellas proteínas que tienen el mismo plegado que proteínas cuya estructura es conocida, pero que no tienen proteínas homólogas con estructura conocida. Reconocimiento de plegado funciona utilizando conocimiento estadístico de la relación entre la estructura depositada en el *Protein Data Bank* y la secuencia de la proteína que se desea modelar (secuencia objetivo) (Bourne y Weissig, 2003).

De manera general, el proceso de reconocimiento de plegado consiste en primero crear una base de datos de proteínas que serán usadas como plantillas estructurales. Después se debe diseñar una función objetivo para medir la aptitud entre las plantillas y la secuencia objetivo. Algunos de los términos que se consideran para la función objetivo son: potencial de mutación, potencial de aptitud de entorno, potencial por parejas (*e.g.* potencial de Van der Waals), compatibilidad de estructura secundaria y penalizaciones por huecos. Posteriormente, se procede a alinear la secuencia objetivo con las secuencias plantilla optimizando la función objetivo diseñada. Por último se selecciona el alineamiento que es estadísticamente más probable y se utilizan las coordenadas de la columna vertebral de la plantilla en la secuencia objetivo.

2.2.3. Modelado homólogo

De los tres enfoques principales para la predicción de la estructura tridimensional de una proteína, el modelado homólogo es el más fácil. El modelado homólogo se basa en dos observaciones:

1. La estructura de una proteína se determina de forma única por su secuencia de aminoácidos (Epstein *et al.*, 1963). Conocer la secuencia debería, por lo menos en teoría, ser suficiente para obtener la estructura; aunque existen excepciones, un gran número de proteínas cumplen con este enunciado.

2. En el transcurso de la evolución, la estructura de las proteínas es más estable, ya que los cambios estructurales ocurren con menos frecuencia que los cambios en su secuencia de aminoácidos asociada, de manera que secuencias similares se pliegan en estructuras prácticamente iguales. Esta relación fue identificada primero por Chothia y Lesk (1986), y después fue cuantificada por Sander y Schneider (1991). Gracias al crecimiento acelerado del Protein Data Bank (PDB), Rost (1999) pudo encontrar un límite preciso para esta regla, la cual se muestra en la Figura 5. En esta figura se observa que para porcentajes por arriba de 30 % de identidad de secuencia y aproximadamente 90 aminoácidos alineados se encuentra la zona segura para modelado homólogo, lo cual significa que está prácticamente garantizado que ambas secuencias se plieguen en la misma estructura.

Para llevar a cabo el modelado homólogo se requieren dos secuencias. La primera se denomina secuencia objetivo, la cual es la secuencia de la cual se desea conocer su estructura. La segunda se denomina la secuencia plantilla, que es la secuencia sobre la que se tiene información, la cual se utilizará para poder predecir la estructura de la secuencia objetivo. El modelado homólogo es un proceso que se puede resumir en siete pasos (Bourne y Weissig, 2003):

1. Reconocimiento de la plantilla y alineamiento inicial.
2. Corrección de alineamiento.
3. Generación de la columna vertebral.
4. Modelado de lazos.

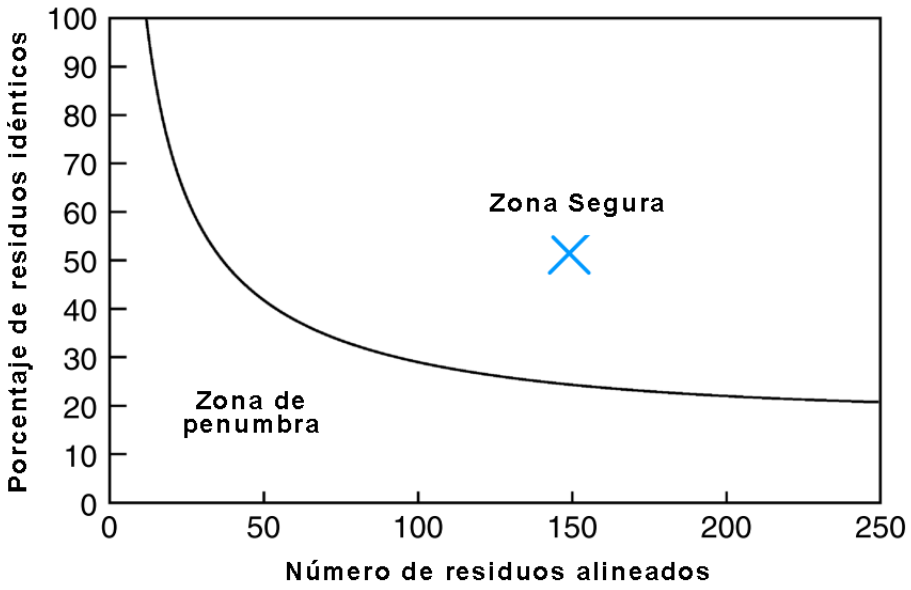

Figura 5: Dos secuencias están prácticamente garantizadas a plegarse en la misma estructura si su longitud y su porcentaje de identidad de secuencia cae dentro de la zona marcada como segura. Un ejemplo de dos secuencias con 150 aminoácidos, de los cuales el 50 % son idénticos, se muestra con una cruz (Rost, 1999, p. 508).

5. Modelado de la cadena lateral.
6. Optimización del modelado.
7. Validación del modelado.

A continuación se describe brevemente en qué consiste cada uno de estos pasos.

Reconocimiento de la plantilla y alineamiento inicial

Dentro de la zona segura para el modelado homólogo (Figura 5), el porcentaje de identidad entre la secuencia objetivo y la posible plantilla es lo suficientemente alto como para que sea posible encontrar la secuencia plantilla utilizando programas de alineamiento de secuencia como BLAST (Altschul *et al.*, 1990) o FASTA (Pearson, 1990).

Corrección de alineamiento

Este paso consiste en que una vez que se encontró una o más posibles plantillas utilizando los métodos mencionados en el punto anterior, recurrir a métodos más sofisticados para así conseguir un mejor alineamiento. Para esto, se puede utilizar lo que se conoce como alineamiento múltiple de secuencias. Existen varios programas que pueden llevar a cabo este proceso, por ejemplo CLUSTAL W (Thompson *et al.*, 1994).

Generación de la columna vertebral

Una vez que se terminó el alineamiento, se puede empezar con el proceso de modelado. Crear la columna vertebral es trivial para la mayor parte del modelo, ya que este proceso consiste simplemente en copiar las coordenadas de los aminoácidos de la plantilla que aparecen en el alineamiento con la secuencia objetivo. Si dos residuos alineados son distintos, entonces sólo se copian las coordenadas de los átomos de la columna vertebral (N, C^{α}, C y O). Si los residuos son iguales, entonces también se podrían copiar las coordenadas de los átomos de la cadena lateral, por lo menos las cadenas laterales más rigidas, ya que los rotámeros tienden a conservarse.

Modelado de lazos

En la mayoría de los casos, el alineamiento entre la secuencia objetivo y la plantilla contiene huecos. Ya sea huecos en la secuencia objetivo (deleciones) o en la secuencia plantilla (inserciones). En el primer caso, simplemente se omiten los residuos de la plantilla, creando un hueco se debe cerrar. En el segundo caso, uno toma la columna vertebral continua de la plantilla, la corta, e inserta los residuos faltantes. Ambos casos implican un cambio en la conformación de la columna vertebral.

Hay dos maneras en las que se resuelve el modelado de lazos:

1. Basado en conocimiento: este método consiste en buscar en el Protein Data Bank por lazos conocidos cuyos extremos coincidan con los residuos entre los cuales se tiene que insertar el lazo, y simplemente se copia la conformación del lazo.

2. Basado en energía: como en el método de predicción de estructura ab initio, se utiliza una función de energía para juzgar la calidad del lazo. Después se minimiza dicha función de energía, para llegar así a la mejor conformación para el lazo.

Modelado de la cadena lateral

Cuando se comparan las conformaciones de la cadena lateral de residuos que se conservan en proteínas de estructura similar, se encuentra que en ocasiones tienen ángulos de torsión χ_1 similares, por lo tanto, es posible simplemente copiar enteramente los residuos conservados, y de esta manera se puede lograr una mejor precisión que cuando simplemente se copia la

columna vertebral y se intenta predecir la conformación de la cadena lateral. En este trabajo se aborda este problema pero de manera más general, el cual se define en la Sección 2.3.

Es importante notar que las precisiones reportadas por la mayoría de los algoritmos que abordan este problema no se pueden alcanzar al momento de que se quieren aplicar en situaciones reales. Esto se debe a que estos métodos se evalúan tomando una estructura ya conocida, eliminando las cadenas laterales, y volviendo a predecirlas. Por lo tanto, estos algoritmos dependen de que la columna vertebral que reciben es la correcta, la cual no está disponible cuando se hace el modelado homólogo. Algunas veces, la columna vertebral de la secuencia plantilla difiere de manera significativa con la de la secuencia objetivo. Por este motivo, en algunos casos los rotámeros se predicen basándose en una columna vertebral incorrecta, por lo que la precisión de las predicciones tiende a ser menor en estas situaciones.

Optimización del modelado

Como se mencionó en el punto anterior, para obtener buena precisión en las cadenas laterales se requiere una columna vertebral correcta; sin embargo, ésta depende también de los rotámeros y su empaquetamiento. El enfoque más común que se le da a este problema es uno iterativo, el cual consiste en primero predecir los rotámeros, después realizar cambios en la columna vertebral, y volver a predecir los rotámeros para la nueva columna vertebral, repitiendo este proceso hasta que ninguno de los dos procesos puede hacer algún otro cambio a la estructura.

Validación del modelado

Cada modelado homólogo contiene errores. El número de errores depende principalmente de dos valores:

1. El porcentaje de identidad de secuencia entre la plantilla y el objetivo. Si es mayor al 90 %, entonces la precisión del modelo es comparable a las estructuras obtenidas por cristalografía (Chothia y Lesk, 1986). Entre el 50 % y 90 % de identidad de secuencia, el RMSD (Sección 2.3.1) puede llegar hasta 1.5Å. Si la identidad de secuencia es menor al 25 %, entonces el alineamiento resulta ser el principal cuello de botella para el modelado homólogo, lo cual lleva a errores importantes en las estructuras predichas.

2. El número de errores en la estructura de la plantilla.

2.3. Definición del problema

El problema de empaquetamiento de la cadena lateral en proteínas (PSCPP por sus siglas en inglés) es un problema del área de la bioinformática estructural, y es un paso muy importante en el proceso de modelado homólogo.

El PSCPP consiste en predecir las coordenadas tridimensionales de todos los átomos que conforman la cadena lateral de cada uno de los aminoácidos de una proteína, recibiendo como entrada la secuencia de aminoácidos que conforman la proteína y las coordenadas de los átomos de la columna vertebral de dicha proteína.

El PSCPP se puede reducir a un problema de optimización combinatoria. Para esto, se discretiza el espacio de búsqueda con el uso de rotámeros, y se utiliza como función objetivo una función de energía que modele el empaquetamiento de las cadenas laterales. Con estos elementos se puede definir el PSCPP como el problema de encontrar el conjunto de rotámeros $\mathbf{r} = (r_1, r_2, \cdots, r_n)$, donde r_i representa el rotámero seleccionado para el i-ésimo aminoácido en la secuencia, cuya función de energía resulta ser mínima, dadas las coordenadas de los átomos de la columna vertebral de la proteína, así como la secuencia de aminoácidos que la conforman. Akutsu (1997) demostró que este problema es NP-Difícil, lo que implica que no se conoce ningún algoritmo de tiempo polinomial que lo pueda resolver, y la única alternativa exacta conocida requiere de la evaluación de un número exponencial de soluciones.

En la Figura 1 de la Sección 1.2 se muestra un ejemplo del PSCPP.

2.3.1. Medidas de calidad

Para evaluar la calidad de los algoritmos que resuelven en forma aproximada el PSCPP se utilizan diferentes métricas. Por ejemplo, para comparar la calidad de los ángulos de torsión predichos se utiliza la precisión absoluta y la precisión condicional (Krivov *et al.*, 2009). Mientras que si se desea comparar la estructura completa de la proteína se puede utilizar la desviación media cuadrática (RMSD). A continuación se describe con mayor detalle la precisión absoluta y el RMSD, ya que estas métricas serán las que se utilizarán para comparar la calidad de las soluciones en el Capítulo 4.

Precisión Absoluta

Para comparar la precisión absoluta de los métodos que resuelven en forma aproximada el PSCPP se utilizan principalmente las métricas $\chi_1(\%)$ y $\chi_{1+2}(\%)$, $\chi_1(\%)$ representa el porcentaje de aminoácidos cuyo ángulo de torsión χ_1 fue predicho correctamente, mientras que $\chi_{1+2}(\%)$ es el porcentaje de aminoácidos cuyos ángulos de torsión χ_1 y χ_2 fueron ambos predichos correctamente.

En este trabajo se considera que un ángulo de torsión es correcto cuando el ángulo de torsión predicho se encuentra alejado a un máximo de 40° del ángulo de torsión de la estructura nativa de la proteína. Aunque no se ha reportado una justificación para este valor, es el margen de error utilizado por la mayoría de los algoritmos para el PSCPP (Krivov *et al.*, 2009; Lu y Dousis, 2008; Cao *et al.*, 2011; Miao *et al.*, 2011).

Existen algunos problemas al comparar ángulos de torsión, dichos problemas surgen debido a que estos ángulos se encuentran en el espacio continuo, y por lo tanto, errores de cálculo numérico pueden afectar las mediciones. Otro problema surge también por el uso de rotámeros, los cuales limitan los valores que pueden tomar los ángulos de torsión. Por estos motivos es que se utiliza un cierto margen de error en los ángulos predichos, para que estos sean considerados como correctos (Corona de la Fuente, 2010).

Desviación Media Cuadrática (RMSD)

RMSD se utiliza como una medida de diferencia entre estructuras, ya que indica qué tanta diferencia de distancia hay en promedio entre los átomos de dos estructuras. La RMSD se calcula de la siguiente manera:

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\delta_i^2}, \tag{2}$$

en donde δ_i es la distancia entre el par i de los N pares de átomos. Un par i se forma tomando el mismo átomo del mismo aminoácido para cada una de las dos estructuras, por ejemplo, si se toma el C^β del aminoácido número tres de cada estructura, entonces δ_i será la distancia entre ámbos átomos, la cual se calcula con base en las coordenadas que tiene cada átomo.

El costo computacional de calcular el valor de RMSD entre dos estructuras es de $O(N)$, en donde N es el número de átomos de la estructura. Note que las dos estructuras tienen el

mismo número de átomos.

2.4. Teorema de eliminación de callejones sin salida

Dentro del PSCPP, una proteína se modela como una estructura la cual está conformada por una secuencia de aminoácidos. Cada uno de estos se encuentra en un estado rotamérico específico, es decir, cada aminoácido i tiene un solo rotámero r_i asignado para describir la conformación de su cadena lateral. Esta cadena lateral se encuentra unida a un conjunto de átomos denominado columna vertebral de la proteína, los cuales se encuentran en una posición fija. Con base en esto, la energía total de una proteína se puede calcular como:

$$E_{total} = E_{BB} + \sum_{i} E(r_i) + \sum_{i} \sum_{j>i} E(r_i, r_j), \tag{3}$$

en donde E_{BB} es la energía de interacción entre los átomos de la columna vertebral, $E(r_i)$ es la energía de interacción entre los átomos de la cadena lateral del rotámero r_i con respecto a la columna vertebral, y $E(r_i, r_j)$ es la energía de interacción entre los rotámeros r_i y r_j de los aminoácidos i y j, respectivamente.

Con base en esta manera de calcular la energía entre parejas de rotámeros es que se propuso el teorema de eliminación de callejones sin salida (Dead-End Elimination Theorem, DEE, Desmet *et al.* (1992)), el cual tiene como propósito descartar rotámeros que generan una energía de interacción elevada con respecto al resto de los rotámeros de un cierto aminoácido, y por consiguiente, no son compatibles con la conformación general de energía mínima (GMEC) de la proteína. En una búsqueda combinatoria para encontrar la GMEC, las conformaciones de la cadena lateral que contengan alguno de estos rotámeros se podrían descartar de la búsqueda. Con base en esto, si para un par de rotámeros (i_r, i_t) de un mismo aminoácido i, se cumple que:

$$E(i_r) + \sum_{j} \min_{s} E(i_r, j_s) > E(i_t) + \sum_{j} \max_{s} E(i_t, j_s); \qquad i \neq j\,, \tag{4}$$

entonces el rotámero i_r no es compatible con la GMEC y se puede descartar de la búsqueda. Lo que la Ecuación 4 indica es que si la suma de interacciones mínimas de un rotámero i_r con respecto al resto de los aminoácidos j probando con cada uno de sus rotámeros s, es mayor

que las interacciones máximas de un rotámero i_t del mismo aminoácido, entonces se puede descartar el rotámero i_r. Dicho de otra manera, si la energía de interacción del rotámero i_r, en su mejor caso, es mayor que la energía de interacción de un rotámero i_t, en su peor caso, entonces se puede descartar el rotámero i_r.

En la Equación 4 se describe el primer criterio para DEE descubierto por Desmet *et al.* (1992); sin embargo, Goldstein (1994) propuso un criterio de eliminación más fuerte, conocido como el criterio de Goldstein:

$$E(i_r) - E(i_t) + \sum_j \min_s \left[E(i_r, j_s) - E(i_t, j_s)\right] > 0; \qquad i \neq j. \tag{5}$$

En palabras, la diferencia entre el criterio de Desmet y el criterio de Goldstein es que en el criterio de Desmet, un rotámero en particular i_r no puede contribuir a la GMEC si la menor energía posible de un confórmero* que contiene a i_r es mayor que la mayor energía posible de un confórmero que contiene a un rotámero i_t para el mismo aminoácido i; por otra parte el criterio de Goldstein indica que un rotámero i_r no puede formar parte de la GMEC si la energía total de la proteína siempre se puede reducir al reemplazar el rotámero i_r por el rotámero i_t manteniendo el resto de los aminoácidos sin cambiar de rotámero.

Los algoritmos de eliminación de callejones sin salida funcionan iterando sobre todos los aminoácidos y rotámeros de la proteína, descartando la mayor cantidad de rotámeros posible. El mejor caso posible sería que al terminar el algoritmo de DEE se eliminen todos los rotámeros a excepción de aquellos que conforman la GMEC, aunque por lo general se espera que por lo menos el tamaño del espacio de soluciones se reduzca lo suficiente como para que se resuelva por métodos heurísticos de enumeración combinatoria (Reingold *et al.*, 1977; Gordon y Mayo, 1999).

*Confórmero: cada una de las estructuras de la proteína que se obtienen al cambiar la conformación de la cadena lateral de un aminoácido.

2.5. Métodos del estado del arte para el PSCPP

2.5.1. SCWRL4

El método SCWRL4 (Krivov *et al.*, 2009) surgió como una nueva generación del método SCWRL, en el cual se implementaron varias mejoras para superar a su antecesor, el SCWRL3 (Canutescu *et al.*, 2003).

El algoritmo SCWRL4 es un método determinístico, y está compuesto de cinco módulos principales para predecir la estructura de la proteína. Dichos módulos son: (1) Entrada de los datos y construcción de las coordenadas de la cadena lateral; (2) cálculo de las energías; (3) construcción de un grafo basado en las interacciones entre los aminoácidos de la proteína; (4) optimización combinatoria mediante el uso de descomposición de aristas, eliminación de callejones sin salida y descomposición de árbol; (5) presentación de resultados.

El primer paso consiste en leer las coordenadas de la columna vertebral, que se encuentran en formato PDB, y utilizando estas coordenadas se calculan los ángulos de torsión ϕ y ψ para cada aminoácido. Después se lee la bibioteca de rotámeros para generar las coordenadas de los átomos de la cadena lateral para cada uno de los rotámeros y subrotámeros. Los subrotámeros se implementan en el SCWRL4 tomando un rotámero al cual se le suma o resta a uno de sus ángulos de torsión una desviación estándar con respecto al valor que tienen en la biblioteca de rotámeros.

El segundo paso consiste en el cálculo de la energía entre pares de átomos y la auto-energía de los rotámeros. El SCWRL4 usa tanto el modelo rígido de rotámeros (RRM) como el modelo flexible de rotámeros (RFM)(Mendes *et al.*, 1999). Si se permiten variaciones para cada ángulo de torsión al momento de generar los subrotámeros resulta un problema intratable para el SCWRL3, por lo que en SCWRL4 se implementó el RFM, en donde se emplean los subrotámeros para producir una aproximación de la energía libre utilizando la aproximación de superposición de Kirkwood (Kirkwood, 1935). En el RRM, que es el modelo con el que normalmente se representan los rotámeros, la energía total del sistema se expresa como:

$$E(\mathbf{r}) = \sum_{i=1}^{N} E_{self}(r_i) + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} E_{Pair}(r_i, r_j), \tag{6}$$

en donde $\mathbf{r} = (r_1, r_2, \cdots, r_n)$ es un vector que contiene el rotámero seleccionado r_i para cada

aminoácido i de la proteína. En la Ecuación 6 la auto-energía $E_{self}(r_i)$ del aminoácido i con el rotámero r_i seleccionado se calcula como:

$$E_{self}(r_i) = -k_i log \frac{p(r_i|\phi_i, \psi_i, A_i)}{p(r'_i|\phi_i, \psi_i, A_i)} + E_{frame}(r_i), \tag{7}$$

en donde el primer término expresa la energía relativa del rotámero r_i con respecto al rotámero de mayor frecuencia r'_i, dados los ángulos de torsión de la columna vertebral ϕ_i y ψ_i de un residuo i, el cual es un aminoácido del tipo A_i. E_{frame} expresa la interacción entre la cadena lateral y la columna vertebral o algún ligando presente. k_i es un factor de escalamiento, cuyo valor depende del tipo de aminoácido para el cual se esté calculando la energía. En la Ecuación 6 el término $E_{pair}(r_i, r_j)$ representa la energía de interacción entre dos pares de átomos, la cual incluye un término para calcular el potencial de van der Waals y uno para los puentes de hidrógeno. El potencial de van der Waals E_{vdW} para un par de átomos (i, j) se calcula como se muestra a continuación:

$$E_{vdW}(i,j) = \begin{cases} 10 & \text{si } \dfrac{d}{\sigma_{ij}} \leq 0.8254, \\ 57.273\left(1 - \dfrac{d}{\sigma_{ij}}\right) & \text{si } 0.8254 \leq \dfrac{d}{\sigma_{ij}} \leq 1, \\ E_{ij}\left(10 - 9\dfrac{d}{\sigma_{ij}}\right)^{\frac{57.273}{9E_{ij}}} - E_{ij} & \text{si } 1 < \dfrac{d}{\sigma_{ij}} < \dfrac{10}{9}, \\ \dfrac{E_{ij}}{4}\left(9\dfrac{d}{\sigma_{ij}} - 10\right)^2 - E_{ij} & \text{si } \dfrac{10}{9} \leq \dfrac{d}{\sigma_{ij}} \leq \dfrac{4}{3}, \\ 0 & \text{si } \dfrac{d}{\sigma_{ij}} \geq \dfrac{4}{3}. \end{cases} \tag{8}$$

En donde σ_{ij} es la suma de los radios de van der Waals de los átomos i y j, E_{ij} es $\sqrt{E_i E_j}$, donde los valores de E_i son los valores E_{min} del potencial del CHARMM param19 (Brooks *et al.*, 1983), mientras que d es la distancia en Å entre el átomo i y el átomo j.

El paso número tres consiste en la creación del grafo de interacciones. En este grafo cada vértice representa un aminoácido de la proteína, mientras que las aristas entre los vértices representan que al menos un rotámero de uno de los aminoácidos tiene un valor de energía de interacción diferente de cero con algún rotámero del otro residuo con el cual se conecta mediante dicha arista. Sin embargo, antes de crear el grafo primero se resuelven los enlaces

disulfuro* de la misma manera que en el SCWRL3 (Canutescu *et al.*, 2003), pero además en el SCWRL4 se añade un procedimiento para resolver ambigüedades. Una ambigüedad ocurre cuando más de un rotámero de una cisteína en particular forma un enlace disulfuro, o cuando un rotámero puede formar un enlace disulfuro con más de una cisteína.

El paso número cuatro consiste en la solución del grafo con descomposición de árbol (Xu, 2005); sin embargo, antes de que el proceso de optimización comience, el grafo de interacción pasa por un preprocesamiento, el cual consiste de una descomposición de aristas (Krivov *et al.*, 2009) y eliminación de callejones sin salida (DEE) utilizando el criterio de Goldstein (Goldstein, 1994). La descomposición de aristas elimina aristas con energías redundantes o que tienen valores de energía casi idénticos unas de otras bajo cierto umbral. Esto ayuda a simplificar el grafo y a eliminar casi todas las aristas con valores cercanos a cero. La DEE identifica y remueve rotámeros que no pueden formar parte de la conformación de energía mínima final de la proteína. Si para un cierto residuo queda sólo un rotámero después de realizar la DEE, entonces dicho rotámero formará parte de la solución final. Si este residuo tiene aristas adyacentes, entonces todas las energías de interacción con los vértices con los que comparte aristas se incorporan a su autoenergía y se eliminan dichas aristas. Esto hace que el residuo quede aislado, por lo que se puede remover del grafo, y su valor de autoenergía se suma al valor total de la energía mínima. Los pasos de la descomposición de aristas y la DEE se repiten hasta que nada más se puede remover. Después de estos pasos, el grafo resultante puede contener subgrafos o clusters sin ninguna arista entre ellos; a cada uno de estos grafos resultantes se les aplica descomposición de árbol (Krivov *et al.*, 2009). Si el árbol resultante se puede resolver por el método de programación dinámica propuesto (Krivov *et al.*, 2009) entonces se encuentra la energía mínima de cada cluster, si no, entonces se repite el proceso de descomposición de aristas pero ahora duplicando el valor del umbral utilizado.

El paso número cinco consiste únicamente en la presentación de los resultados, una vez que se encontró la energía mínima del modelo y su asignación correspondiente de rotámeros, entonces se procede a guardar la conformación de la estructura de la proteína predicha en un archivo con formato PDB.

Algo novedoso que se implementa en el SCWRL4 es la manera en la que detectan colisiones

*Enlace disulfuro: enlace covalente formado entre los grupos thiol(-SH, S=azufre H=hidrógeno) de dos cisteínas.

de manera eficiente, dado que este método utiliza muchos rotámeros y subrotámeros. Se implementó un algoritmo basado en kDOPs (k-Discrete Oriented Politopes, Klosowski *et al.* (1998)), los cuales se utilizan para envolver el área de interacción de los aminoácidos en politopos convexos, con los cuales se puede revisar si hay colisiones de manera eficiente.

La biblioteca de rotámeros utilizada es nueva, desarrollada por Shapovalov y Dunbrack (2011). Esta biblioteca de rotámeros utiliza estimaciones adaptativas de densidad de kernel (AKDEs) para calcular la probabilidad de los rotámeros. Esto lo hacen utilizando las AKDEs para generar las funciones de densidad de probabilidad de los rotámeros $p(\phi, \psi|r)$ del conjunto de datos de entrada $\{\phi_i, \psi_i, r_i\}$, y utilizando la regla de Bayes para invertir estas densidades y obtener la probabilidad de los rotámeros $p(r|\phi, \psi)$ (Shapovalov y Dunbrack, 2011).

2.5.2. OPUS-Rota

El método Opus-Rota (Lu y Dousis, 2008) es un algoritmo heurístico que aproxima el PSCPP, el cual agrega dos términos de energía importantes aparte de los usados más comúnmente, que son el término de frecuencia de rotámeros y el potencial de van der Waals. El primer término de energía que se agregó aparte de los antes mencionados es un potencial sensible a la orientación, denominado OPUS-PSP (Lu *et al.*, 2008); el segundo término implementado es un término de energía de solvatación explícita por parejas. Los términos de la función de energía utilizados por este método se discuten más adelante.

El método de muestreo que utiliza el OPUS-Rota es el de recocido simulado por baño de calor de Monte Carlo (Newman y Barkema, 1999). Dicho método es más eficiente que una búsqueda convencional de Monte Carlo cuando los estados son discretos y acotados, como es el caso cuando se utilizan rotámeros para predecir las conformaciones de la cadena lateral.

Durante el recocido simulado del OPUS-Rota, la probabilidad de aceptación para pasar de un estado actual o a uno nuevo n es:

$$p(o \to n) = Z^{-1} e^{(\Delta E_{tot}(o \to n)/k_B T)}, \tag{9}$$

en donde $Z = \sum_{n'} e^{(\Delta E_{tot}(o \to n')/k_B T)}$ para todos los posibles estados $\{n'\}$, $\Delta E_{rot}(o \to n)$ es la diferencia de energía al pasar de un estado o al estado n, k_B es la constante de Boltzmann, y T es la temperatura. En dado caso que $\Delta E_{tot}(o \to n)/k_B T > 7$, la probabilidad de cambio

es menor que 0.001, y por lo tanto el rotámero n se excluye sin evaluar el resto de la expresión.

La solución inicial para el recocido simulado comienza posicionando todas las cadenas laterales con el rotámero con la menor energía de interacción entre la cadena lateral y la columna vertebral. En esta etapa, al momento de calcular la energía, se excluye el término de solvatación; además, durante esta etapa también se eliminan de la búsqueda los rotámeros que tengan una energía de interacción entre la cadena lateral y la columna vertebral 7.5 unidades mayor que el estado con menor energía para ese mismo residuo.

Durante cada ciclo de la simulación, cada uno de los aminoácidos, exceptuando los de Glicina (Gly) y Alanina (Ala) (ya que estos aminoácidos no cuentan con rotámeros), se optimizan una vez por el baño de calor de Monte Carlo a una temperatura constante en un orden aleatorio. La temperatura decrece gradualmente de 2.5 a 0.05 durante un total de 97 ciclos, seguido de tres ciclos con temperatura cero. El término de solvatación es más lento de calcular que cualquier otro término de la función de energía, y dado que el área superficial accesible al solvente (SASA, por sus siglas en ingés) es menos significativa cuando la estructura sufre de varias colisiones atómicas, el término de solvatación no se considera durante los primeros 60 ciclos de la optimización.

La función de energía utilizada en el OPUS-Rota tiene en total cuatro términos, y se calcula de la siguiente manera:

$$E_{total} = E_{rot} + w_{vdW}E_{vdW} + w_{orient}E_{orient} + w_{solvation}E_{solvation}, \tag{10}$$

donde E_{rot} es el término de frecuencia de rotámero, E_{dW} es el potencial 6-12 de Lennard-Jones modificado, E_{orient} es el potencial de empaquetamiento sensible a la orientación (Lu *et al.*, 2008), y $E_{solvation}$ representa la energía de solvatación. En la Ecuación 10 se incluyen tres pesos, los cuales tienen los siguientes valores: $w_{vdW} = 1.0$, $w_{orient} = 0.15$ y $w_{solvation} = 0.1$, estos se optimizaron usando un grupo pequeño de 11 proteínas de alta resolución (1aac, 1bpi, 1isu, 1ptx, 1xnb, 256b, 2erl, 2hbg, 2ihl, 5rxn y 9rnt).

El término de frecuencia de rotámero E_{rot} toma la siguiente forma:

$$E_{rot} = -\sum_{m=1}^{n} \gamma log \frac{p(r_i|\phi_i, \psi_i, A_i)}{p(r'_i|\phi_i, \psi_i, A_i)}, \quad \text{con}$$

$$\gamma = \begin{cases} 3, & \text{si } A_m \in \{Phe, Tyr, Trp, His\} \\ 0, & \text{si } A_m \in \{Gly, Ala\} \\ 1, & \text{de otra manera.} \end{cases} \tag{11}$$

En donde para los n aminoácidos y el índice de aminoácido, $i \in [1, n]$. $p(r_i|\phi_i, \psi_i, A_i)$ representa la probabilidad del rotámero seleccionado r_i para el aminoácido i, cuyos ángulos de torsión de la columna vertebral son ϕ_i y ψ_i, y que además se trata de un aminoácido del tipo A_i; mientras que $p(r'_i|\phi_i, \psi_i, A_i)$ es la probabilidad del rotámero con mayor frecuencia (r'_i) con la misma configuración. γ es un factor de escalamiento, el cual permite que los aminoácidos aromáticos contribuyan más a la energía total de la proteína.

El potencial de van der Waals E_{vdW} para un par de átomos (i, j) se calcula de la siguiente manera:

$$E_{vdW}(i,j) = \begin{cases} \lambda(49.69 - 40.06 d^*_{ij}), & \text{si } d^*_{ij} \in [0, 1/1.33], \\ \lambda e_{ij} \left[(d^*_{ij})^{-12} - 2(d^*_{ij})^{-6}\right] & \text{si } d^*_{ij} \in (1/1.33, 1/1.12], \\ e_{ij} \left[(d^*_{ij})^{-12} - 2(d^*_{ij})^{-6}\right] & \text{si } d^*_{ij} \in (1/1.12, 2.5). \end{cases} \tag{12}$$

En donde $d^*_{ij} = d_{ij}/\sigma_{ij}$, d_{ij} es la distancia entre el átomo i y el átomo j, $\sigma_{ij} = \sigma_i + \sigma_j$ es la suma de los radios atómicos de los átomos i y j, $e_{ij} = \sqrt{e_i e_j}$ representa la energía potencial entre el átomo i y el átomo j, y λ es un factor de escalamiento con valor de 1.0 si tanto el átomo i como el átomo j son carbonos aromáticos, o toma un valor de 1.6 en cualquier otro caso. Mayor detalle sobre los parámetros del potencial de Lennard-Jones, así como de ciertas reglas al momento de sumar valores se describen en el artículo del OPUS-PSP (Lu *et al.*, 2008).

El potencial de empaquetamiento de la cadena lateral sensible a la orientación E_{orient}, representa a cada aminoácido como un subconjunto conectado de 19 bloques rígidos, los cuales se usan para caracterizar la preferencia de orientación del empaquetamiento. Para un par de tipos de bloque en contacto a y b, con una orientación de empaquetamiento Ω_{ab}, el

potencial de orientación toma la siguiente forma:

$$E_{orient} = \sum n(a,b)E(\Omega_{ab},a,b), \tag{13}$$

donde $n(a,b)$ es un término de peso por cada tamaño de bloque, definido como el número promedio de pares de átomos pesados en contacto, mientras que $E(\Omega_{ab},a,b)$ se define como:

$$E(\Omega_{ab},a,b) = -log\frac{p^{obs}(\Omega_{ab},a,b)}{p^{ref}(\Omega_{ab},a,b)}, \tag{14}$$

en donde p^{obs} es la probabilidad con respecto a todos los posibles estados de contacto observados para cualquier par de bloques, extraída de la base de datos no redundante de estructuras, la cual se generó utilizando el servidor PISCES (Wang y Dunbrack, 2003) con un total de 3211 proteínas; y p^{ref} es la probabilidad del estado de referencia. La sumatoria la Ecuación 13 incluye a todos los pares de bloques de residuos no consecutivos y en contacto, sin un choque atómico severo, el cual ocurre cuando un par de átomos tienen una distancia entre ellos menor a $0.7\sigma_{ij}$. Detalles de E_{orient} se pueden encontrar en el artículo del OPUS-PSP (Lu *et al.*, 2008).

La energía de solvatación $E_{solvation}$, sigue el método de Eisenberg y McLachlan (1986), y toma la siguiente forma:

$$E_{solvation} = \sum_i \Delta\sigma_i S_i, \tag{15}$$

en donde S_i es el área superficial accesible al solvente del átomo i, y $\Delta\sigma_i$ es el parámetro atómico de solvente obtenido de Sharp *et al.* (1991).

Dado que todos los términos de la función de energía del OPUS-Rota son de corto alcance y se calculan entre pares de átomos, se puede acelerar el cálculo de la energía implementando listas de vecinos. Esto con el objeto de catalogar cada uno de los pares de átomos cuya distancia esté dentro del intervalo de interacción.

El método OPUS-Rota presenta una buena precisión, comparable con los demás métodos del estado del arte. Sin embargo, los autores utilizaron un conjunto de prueba muy pequeño, el cual consiste de únicamente 65 proteínas, lo cual sesga los resultados que los autores muestran. Cabe mencionar que el OPUS-Rota no incluye en su función de energía término

Carbono, Nitrógeno, Oxígeno y Azufre

alguno referente a los enlaces disulfuro, como sí lo hacen otros métodos como el SCWRL4 (Krivov *et al.*, 2009), el CIS-RR (Cao *et al.*, 2011) y el RASP (Miao *et al.*, 2011).

2.5.3. CIS-RR

El CIS-RR (Cao *et al.*, 2011) es un método que para obtener una buena calidad en la solución, se enfoca en reducir el número de colisiones entre átomos al momento de predecir la estructura tridimensional de la proteína, esto lo hace tomando en cuenta que los choques atómicos pueden producir una gran cantidad de energía e inestabilidad en las estructuras. Este método realiza una búsqueda iterativa guiada por la detección de choques (CIS, por sus siglas en inglés) de los rotámeros de la cadena lateral, mientras optimiza la conformación de las cadenas laterales utilizando el método de gradiente conjugado, en particular el algoritmo de Fletcher-Reeves (Fletcher y Reeves, 1964). La implementación de este algoritmo es lo que dentro del método se denomina como relajación de rotámeros (RR).

La manera en la que trabaja el CIS-RR es la siguiente: primero, se genera una estructura inicial para la proteína, la cual se construye utilizando los rotámeros de mayor probabilidad de la biblioteca de rotámeros para cada aminoácido. Después, por cada residuo i de la proteína, se optimizan sus rotámeros utilizando la relajación de rotámeros (método descrito más adelante) y se determina si hay colisiones con algún otro aminoácido. En caso de que un rotámero r colisione con algún otro residuo, se fija temporalmente la posición del rotámero r y se explorarán los rotámeros de todos los residuos con los que colisione el rotámero r, cambiando el rotámero de los mismos en caso de que se pueda evitar dicha colisión o si se mejora el valor de energía. Después de que se exploraron todos los rotámeros del residuo i, entonces se actualizan las conformaciones de la cadena lateral del residuo i y los residuos con los que colisionó, utilizando los rotámeros que generaron la menor energía durante la búsqueda. Este proceso continúa iterando desde el aminoácido más expuesto al menos expuesto hasta que la energía total de la proteína llega a un óptimo local. El pseudocódigo del algoritmo CIS-RR se muestra en el Algoritmo 1.

La complejidad del algoritmo es $O(np^2)$, en donde n es el número de residuos y p es el número máximo de rotámeros por residuo.

La relajación de rotámeros (RR) se implementa utilizando un método de gradiente conjugado para optimizar los ángulos de torsión de la cadena lateral. El método sigue los pasos

Algoritmo 1: CIS-RR, Cao *et al.* (2011)

Entrada : Solución inicial S_0
Salida : Solución final S_{best}

```
Generar la estructura inicial utilizando los rotámeros de mayor probabilidad.
mejorEnergía = energía de la conformación inicial.
Por cada aminoácido i hacer
    Por cada rotámero r del aminoácido i hacer
        Utilizar el rotámero r en la estructura de la proteína.
        Si el rotámero r colisiona con algún otro aminoácido entonces
            Por cada aminoácido j con el que colisiona r hacer
                Por cada rotámero s del aminoácido j hacer
                    Utilizar el rotámero s en la estructura de la proteína.
                    Energía = energía de la proteína con los rotámeros r y s.
                    Si Energía < mejorEnergía entonces
                        mejorEnergía = Energía.
                        Guardar rotámeros r y s.
                    fin
                fin
            fin
        Sino
            Energía = energía de la proteína con el rotámero r.
            Si Energía < mejorEnergía entonces
                Guardar rotámero r.
            fin
        fin
    fin
    Actualizar la estructura con el rotámero r o con los rotámeros r y s.
fin
```

del algoritmo Fletcher-Reeves (Fletcher y Reeves, 1964) para producir un conjunto de direcciones para rotar la cadena lateral, y las mismas se calculan de las derivadas de la función objetivo con respecto a los ángulos de torsión. Información más detallada sobre la derivación de la función de energía puede ser encontrada en el material suplementario del CIS-RR (Cao *et al.*, 2011).

La función de energía que se utiliza en este método se adaptó de la utilizada en el SCWRL3 (Canutescu et al., 2003) y se obtiene de la siguiente manera:

$$E = E_{vdW} + k_{rot} E_{Rot}, \tag{16}$$

en donde E_{vdW} es el potencial de Van der Waals y para un par de átomos (i, j) se calcula de la siguiente manera:

$$E_{vdW}(i,j) = \begin{cases} k_{rep}(1 - d_{ij}/\sigma_{ij}) & \text{para } d_{ij} \leq \sigma_{ij}, \\ k_{att}\left[(d_{ij}/\sigma_{ij})^2 - 3.0(d_{ij}/\sigma_{ij}) + 2.0\right] & \text{para } \sigma_{ij} < d_{ij} < 2.0\sigma_{ij}, \\ 0.0 & \text{para } d_{ij} \geq 2.0\sigma_{ij}. \end{cases} \tag{17}$$

En donde d_{ij} es la distancia entre los átomos i y j, mientras que σ_{ij} es la suma de sus radios de Van der Waals, obtenidos del trabajo de Chothia (1976). El término de la función de energía E_{rot} mide la preferencia de los confórmeros de la cadena lateral, si para un aminoácido i se selecciona el rotámero r_i, su energía $E_{rot}(r_i)$ se calcula de la siguiente manera:

$$E_{rot}(r_i) = \gamma \left(-log\left(\frac{p(r_i|\phi_i, \psi_i, A_i)}{p(r_i'|\phi_i, \psi_i, A_i)}\right) + k_{tor} \sum_n \left(\frac{\chi_n(r_i) - \chi_{lib\_n}}{\sigma_{lib\_n}}\right)^2 \right), \tag{18}$$

en donde $p(r_i|\phi_i, \psi_i, A_i)$ es la probabilidad de ocurrencia del rotámero seleccionado r_i para el aminoácido i dados los ángulos de torsión de la columna vertebral ϕ_i y ψ_i, normalizado al rotámero con mayor probabilidad r_i' para el residuo i con tipo de aminoácido A_i, dados los mismos ángulos de torsión de la columna vertebral (Canutescu *et al.*, 2003). El segundo término de la ecuación 18 sirve para penalizar el desvío de los ángulos de torsión de la cadena lateral con respecto a los valores más cercanos de la biblioteca de rotámeros. El ángulo de torsión de la cadena lateral $\chi_n(r_i)$ es el enésimo ángulo de torsión del rotámero r_i, χ_{lib_n} y σ_{lib_n} son el ángulo de torsión de la biblioteca de rotámeros y su desviación

estándar, respectivamente. Dado que las cadenas laterales aromáticas están predispuestas a tener colisiones, su E_{rot} puede ser poco significativa comparada con su E_{vdW}, por lo tanto se incluye el factor de escalamiento γ:

$$\gamma = \begin{cases} 2, & \text{si } A_i \in \{\text{His, Phe, Tyr, Trp}\} \\ 0, & \text{si } A_i \in \{\text{Gly, Ala}\} \\ 1, & \text{de otra manera.} \end{cases} \tag{19}$$

Los parámetros utilizados en las ecuaciones anteriores tienen los siguientes valores: $k_{rot} = 0.35$, $k_{rep} = 5.882$, $k_{att} = 0.08$, $k_{tor} = 0.4$. Dichos valores se optimizaron utilizando 55 estructuras de alta resolución seleccionadas al azar del Protein Data Bank (PDB; Berman *et al.* (2000)).

Antes de realizar el cálculo de la función objetivo se evalúan los pares de Cisteína, los cuales generan un enlace de disulfuro, utilizando la expresión $E_{SS} = 0.1 \cdot S - 4.0$, en donde S es el puntaje del enlace de disulfuro, el cual se obtiene de la misma manera que en el SCWRL3 (Canutescu *et al.*, 2003). Si las cadenas laterales de los pares de cisteínas forman un enlace de disulfuro ($E_{SS} < 0$), entonces se agrega E_{SS} a la función objetivo.

La biblioteca de rotámeros utilizada en este método es la biblioteca de rotámeros dependiente de la columna vertebral en su versión de mayo del 2002 (Dunbrack y Cohen, 1997; Dunbrack y Karplus, 1993). En este método se ignoraron los rotámeros que tienen una probabilidad de ocurrencia menor al 1 %.

A pesar de que este método tiene una buena precisión, comparable con la de los demás algoritmos que aproximan el PSCPP, y de que predice estructuras con un menor número de colisiones que algoritmos como OPUS-Rota (Lu y Dousis, 2008) o SCWRL4 (Krivov *et al.*, 2009) tiene una desventaja, es muy lento cuando se le compara con métodos como el RASP (Miao *et al.*, 2011) o el SCWRL3 (Canutescu *et al.*, 2003), los cuales son órdenes de magnitud más rápidos.

2.5.4. RASP

RASP (Miao *et al.*, 2011) es un algoritmo que fue desarrollado en el mismo laboratorio que el CIS-RR (Cao *et al.*, 2011), y de manera similar también realiza una búsqueda guiada

por detección de choques para reducir el número de colisiones entre átomos. Sin embargo, la diferencia entre estos métodos radica en que RASP no empareja el proceso de eliminación de colisiones atómicas con el de empaquetamiento de la cadena lateral durante la búsqueda iterativa, sino que primero genera rápidamente una estructura de buena calidad considerando los elementos clave de los algoritmos de empaquetamiento de la cadena lateral basados en rotámeros, y después reduce el número de colisiones entre átomos sobre la estructura generada.

Para la generación de la estructura inicial el RASP implementa una combinación de los siguientes algoritmos: Eliminación de callejones sin salida (DEE, por sus siglas en inglés) (Desmet *et al.*, 1992), un algoritmo basado en teoría de grafos para el modelado comparativo de la estructura de proteínas (Samudrala y Moult, 1998), un algoritmo de ramificación y terminación, el cual es un algoritmo de optimización combinatoria para el diseño de proteínas (Gordon y Mayo, 1999), búsqueda Monte Carlo (Liu, 2008) y un algoritmo de marcha atrás (Tarjan, 1971). Una vez generada la estructura se procede a realizar la relajación de rotámeros siguiendo el mismo enfoque utilizado en el algoritmo CIS-RR, considerando como una colisión entre átomos cuando la distancia entre ambos es menor que el 60 % de la suma de sus radios de Van der Waals. Dichos radios fueron tomados del programa Rosetta (Rohl *et al.*, 2004).

El proceso de la búsqueda combinatoria se realiza de manera muy similar a la del SCWRL3 (Canutescu *et al.*, 2003), empezando con el DEE para reducir el espacio de búsqueda combinatorio, después se construye el grafo de interacciones y se divide en componentes biconectados, cada uno de estos componentes se resuelve utilizando el algoritmo de ramificación y terminación (Gordon y Mayo, 1999) para encontrar el mínimo de la energía de interacción entre los aminoácidos que pertenecen a cada componente biconectado. Una de las mejoras que se implementaron para la búsqueda combinatoria es una manera más eficiente de resolver el grafo una vez que se obtuvieron los componentes biconectados. Si el tamaño del espacio de búsqueda es mayor a 10^{15} se realiza un recocido simulado de Monte Carlo (Liu, 2008), en caso contrario se utiliza el algoritmo de ramificación y terminación para resolver el grafo.

El recocido simulado de Monte Carlo empieza con una estructura con todos los residuos, en donde para cada aminoácido se utiliza el rotámero con la menor auto-energía, es decir, el rotámero que tiene una menor energía para $E_{lib}(r_i)$ (la expresión para calcular $E_{lib}(r_i)$ se muestra más adelante). La probabilidad de un nuevo rotámero n de reemplazar a un rotámero

viejo o, se denota como $p\,(o \rightarrow n)$ y se calcula como:

$$p\,(o \rightarrow n) = e^{(-\Delta E_{rot}(o \rightarrow n)/T)}, \tag{20}$$

en donde $\Delta E_{rot}(o \rightarrow n)$ es la diferencia de energía que se obtiene al reemplazar el rotámero o por un nuevo rotámero n, mientras que T representa la temperatura.

Antes de realizar la búsqueda de Monte Carlo se eliminan los rotámeros que tienen una baja probabilidad de ser aceptados, más específicamente, cuando $p(o \rightarrow n) < e^{-10}$. Después se realizan 100 generaciones de la búsqueda de Monte Carlo con un decremento gradual de temperatura de 2 a 0.02 unidades, seguido de tres generaciones de búsqueda voraz.

Dada una selección de rotámeros $\mathbf{r}$=$\{r_1, r_2, r_3, \ldots, r_n\}$ la función de energía utilizada en el RASP está dada por:

$$E_{total}(\mathbf{r}) = \sum_{i}^{n} E_{lib}(r_i) + \sum_{i}^{n}\sum_{j} E_{bb-sc}(r_i, j) + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E_{sc-sc}(r_i, r_j), \tag{21}$$

en donde r_i es el rotámero seleccionado para el residuo i y n es el número total de aminoácidos de la proteína.

La Ecuación 21 se divide en dos partes. La primera parte es E_{lib}, un término asociado a la probabilidad del rotámero, y tiene la misma forma que el utilizado en el SCWRL4 (Krivov *et al.*, 2009), representando la probabilidad de ocurrencia relativa de un rotámero r_i con respecto al rotámero con más alta probabilidad r_i' del aminoácido i, dados los ángulos de torsión de la columna vertebral ϕ_i y ψ_i. Este término se calcula como se muestra a continuación:

$$E_{lib}(r_i) = -w_{aa} log \frac{p(r_i|\phi_i, \psi_i, A_i)}{p(r_i'|\phi_i, \psi_i, A_i)}, \tag{22}$$

en donde w_{aa} es un factor de escalamiento que es dependiente del tipo de aminoácido A_i.

La segunda parte de la función de energía es la que modela las interacciones entre átomos. Estas interacciones están separadas en dos sumatorias en la Ecuación 21; el término E_{bb-sc} se refiere a la interacción entre la columna vertebral y las cadenas laterales, mientras que E_{sc-sc} se refiere a la interacción entre las cadenas laterales. Estos términos están a su vez divididos

en tres partes, y se calculan como se muestra a continuación:

$$E_{bb-sc/sc-sc} = E_{vdW} + E_{SS} + E_{[O,H]}. \tag{23}$$

El término E_{vdW} representa el potencial de Van der Waals. La manera en que se obtiene este potencial en el RASP es una adaptación del método usado en el OPUS-PSP (Jain *et al.*, 2006; Lu *et al.*, 2008), la expresión para calcularlo está dada por:

$$E_{vdW}(i,j) = \begin{cases} 50e_{ij} & \text{si } d^*_{ij} < 0.465, \\ e_{ij}(80 - 64.5d^*_{ij}) & \text{si } 0.465 \leq d^*_{ij} < 0.75, \\ 1.63e_{ij}\left[\left(\frac{1}{d^*_{ij}}\right)^{12} - 2\left(\frac{1}{d^*_{ij}}\right)^{6}\right] & \text{si } 0.75 \leq d^*_{ij} < 0.8929, \\ 0.99e_{ij}\left[\left(\frac{1}{d^*_{ij}}\right)^{12} - 2\left(\frac{1}{d^*_{ij}}\right)^{6}\right] & \text{si } 0.8929 \leq d^*_{ij} < 2.3. \end{cases} \tag{24}$$

En donde e_{ij} representa la energía potencial entre el átomo i y el átomo j. Los valores de la energía potencial de cada átomo se obtuvieron del CHARMM param19 (Brooks *et al.*, 1983). $d^*_{ij} = d_{ij}/\sigma_{ij}$, en donde d_{ij} es la distancia entre el átomo i y el átomo j, mientras que σ_{ij} es la suma de los radios de Van der Waals de los átomos i y j (estos valores se encuentran en la tabla suplementaria S2 del RASP).

El término para los enlaces disulfuro E_{SS} es una versión simplificada del utilizado en el SCWRL3 (Miao *et al.*, 2011). La energía de los puentes de hidrógeno $E_{[O,H]}$ sólo considera los puentes de hidrógeno formados entre el grupo hidroxilo de un aminoácido y el grupo carboxilo del siguiente aminoácido, este término se calcula únicamente cuando la distancia entre un átomo de oxígeno del grupo carboxilo y el átomo de oxígeno del grupo hidroxilo es menor a 3.2Å (Miao *et al.*, 2011).

Los rotámeros utilizados provienen de una biblioteca de rotámeros dependiente de la columna vertebral (Dunbrack y Cohen, 1997). Estos rotámeros se leen de la biblioteca en orden, empezando con el rotámero de mayor probabilidad, hasta que la probabilidad acumulada de los rotámeros leídos es del 98 %. El algoritmo no toma en cuenta el resto de los rotámeros.

En cuanto a la calidad de las estructuras predichas por este algoritmo, se obtuvieron

buenos resultados en los casos de prueba utilizados por el autor, con un valor de 85 % de precisión en χ_1. Sin embargo, la principal característica de este algoritmo es su tiempo de ejecución, ya que el RASP es 14 veces más rápido que el OPUS-Rota (Lu y Dousis, 2008), 18 veces más rápido que el SCWRL4 (Krivov *et al.*, 2009) y alrededor de 40 veces más rápido que el CIS-RR (Cao *et al.*, 2011); además de que genera menos colisiones que los algoritmos antes mencionados.

La precisión promedio de estos métodos está en alrededor de 85 % para χ_1(%) y 75 % para χ_{1+2}. Sin embargo, se ha mostrado recientemente (Corona de la Fuente, 2010; Lezcano, 2012) que la precisión promedio que se puede lograr, usando una biblioteca sencilla de rotámeros, es de aproximadamente 97 % para χ_1(%), lo cual implica que aún queda un amplio margen para mejorar la precisión con respecto a los algoritmos del estado del arte.

En el siguiente capítulo se presentan los algoritmos propuestos para abordar este problema.

Capítulo 3

Algoritmos propuestos

En este trabajo se proponen tres algoritmos híbridos, los cuales combinan tres estrategias diferentes. La primera es una conocida como eliminación de callejones sin salida (DEE, Desmet *et al.* (1992)), este algoritmo sirve para descartar rotámeros que generan estructuras con un valor elevado de energía. Al eliminar estos rotámeros es posible reducir el espacio de búsqueda, lo cual ayuda a mejorar la calidad de la solución inicial y a reducir el tiempo de ejecución del algoritmo. La segunda consiste en una heurística para guiar el proceso de optimización, las heurísticas implementadas son: búsqueda tabú, recocido simulado y búsqueda local. La tercera estrategia es un algoritmo voraz, el cual se ejecuta una vez que el algoritmo de optimización llega a un factor α $(0 < \alpha < 1)$ del número máximo de iteraciones totales. Con esto, cada uno de los algoritmos híbridos está formado por una de las tres heurísticas, además del algoritmo voraz y el algoritmo DEE. A continuación se describen las heurísticas de optimización primero y el algoritmo voraz después.

3.1. Algoritmo de búsqueda tabú

El primero de los tres algoritmos implementados es una metaheurística tipo búsqueda tabú (Glover, 1989). Este tipo de metaheurística pertenece a la clase de búsqueda local, pero mejora su rendimiento con el uso de estructuras de memoria para evitar que se visiten soluciones que fueron exploradas recientemente.

La motivación para seleccionar esta metaheurística es que cuando se usan rotámeros en el PSCPP, el espacio de búsqueda es discreto; y aunque búsqueda tabú no es la única metaheurística para este tipo de espacios de búsqueda, ha demostrado buenos resultados para otros problemas combinatorios (Aarts y Lenstra, 1997). Por otro lado, actualmente existen algunos algoritmos en el estado del arte que ya implementan un competidor usual como lo es el recocido simulado, *e.g.* OPUS-Rota (Lu y Dousis, 2008) y RASP (Miao *et al.*, 2011), por lo que resulta natural preguntarse acerca del desempeño que una estrategia tipo búsqueda tabú puede lograr en este problema.

En el Algoritmo 2 se muestra una versión general del algoritmo de búsqueda tabú. Dicho algoritmo suele implementar tres tipos de memoria: a corto, mediano y largo plazo (Glover,

Algoritmo 2: Algoritmo de búsqueda tabú simplificado.

Entrada : Solución inicial S_0
Salida : Solución final S_{best}

```
1  S_best ← S_0
2  listaTabú ← null //se inicializa la lista tabú
3  Mientras ¬ condiciónDeParo( ) hacer   //mientras no se cumpla la condición de paro
4  |  listaDeSolucionesCandidato ← null.
5  |  Para S en el vecindario N(S_best) hacer
6  |  |  Si ¬ contieneElemento(S, listaTabú) entonces   //si la lista tabú no contiene a S
7  |  |  |  listaDeSolucionesCandidato ← listaDeSolucionesCandidato + S
8  |  |  fin
9  |  |  S_candidato ← localizarMejorCandidato(listaDeSolucionesCandidato)
10 |  |  Si fitness(S_candidato) > fitness(S_best) entonces   //si S_candidato es mejor que
   |  |  S_best
11 |  |  |  listaTabú ← listaTabú + características(S_candidato)
12 |  |  |  S_best ← S_candidato
13 |  |  |  Si size(listaTabú) > maxTamañoListaTabú entonces
14 |  |  |  |  expirarCaracterísticas(listaTabú)
15 |  |  |  fin
16 |  |  fin
17 |  fin
18 fin
19 Retornar S_best
```

1990). La memoria a corto plazo es la lista tabú, la cual almacena las soluciones visitadas recientemente para que no se visiten nuevamente hasta que su tiempo en dicha lista expire. La memoria a mediano plazo es un conjunto de reglas de intensificación, las cuales sesgan la búsqueda hacia áreas prometedoras en el espacio de búsqueda. La memoria a largo plazo es un conjunto de reglas de diversificación, las cuales guían la búsqueda hacia nuevas regiones, por ejemplo en caso de que la búsqueda se queda atorada en un callejón sin salida sub-óptimo.

En el algoritmo propuesto se utilizó la memoria a corto y mediano plazo. La memoria a corto plazo (lista tabú) se utilizó implementando una estructura de datos, la cual consiste de una lista en la cual se almacena el índice de los aminoácidos que se exploraron en iteraciones recientes. La memoria a mediano plazo se utilizó al guiar las soluciones hacia estructuras con pocas colisiones, este proceso se explica con mayor detalle en la Sección 3.1.6.

El Algoritmo 3 muestra el esquema implementado, y a continuación se describe de manera detallada en qué consisten los elementos principales del algoritmo. Todos los números generados de manera aleatoria para este algoritmo se toman de una distribución uniforme (función nextInt de la clase Random del lenguaje Java).

Algoritmo 3: Algoritmo propuesto tipo búsqueda tabú.

Entrada : Proteína en formato PDB, biblioteca de rotámeros, MAX_ITER
Salida : Una predicción de la estructura de la proteína

```
1  Inicialización de estructuras de datos y directorios.
2  Creación de un grid con los átomos de la proteína.
3  Cálculo de todas las interacciones de la proteína.
4  Aplicación del método DEE.
5  Creación de la estructura inicial.
6  Crear la lista tabú LT, LT = ∅
7  mejorEnergía = Eini // Energía de la estructura inicial.
8  Para n = 0 hasta el número máximo de iteraciones (MAX_ITER) hacer
9      Seleccionar aleatoriamente un aminoácido i que no se encuentre en LT.
10     Por cada rotámero r del aminoácido i hacer
11         Por cada aminoácido j en la lista de aminoácidos cercanos de i hacer
12             Si el aminoácido i con el rotámero r colisiona con el aminoácido j con su
               rotámero actual entonces
13                 Por cada rotámero s del aminoácido j hacer
14                     Energía= Energía de la proteína utilizando los rotámeros r y s.
15                     Si Energía < mejorEnergía entonces
16                         mejorEnergía = Energía
17                         Actualizar la estructura de la proteína utilizando los rotámeros r
                           y s para los aminoácidos i y j, respectivamente.
18                     fin
19                 fin
20             fin
21         fin
22     fin
23     Si el aminoácido i no colisionó con ningún otro aminoácido entonces
24         Por cada rotámero r del aminoácido i hacer
25             Energía = Energía de la proteína utilizando el rotámero r para i.
26             Si Energía < mejorEnergía entonces
27                 mejorEnergía = Energía
28                 Actualizar la estructura de la proteína utilizando el rotámeros r para el
                   aminoácido i.
29             fin
30         fin
31     fin
32     Agregar el aminoácido i a LT.
33     Si el número de generaciones es la mitad de MAX_ITER entonces
34         Ejecutar el algoritmo voraz para colisiones (Algoritmo 4).
35     fin
36 fin
37 Ejecutar el algoritmo voraz para reducir el número de colisiones (Algoritmo 4).
38 Generar archivos de resultado de la solución con la menor energía encontrada.
```

3.1.1. Inicialización de las estructuras de datos y directorios (línea 1 del Algoritmo 3)

En este paso se lee el archivo con formato .pdb de la proteína, con el cual se crea un objeto tipo estructura, el cual contiene toda la información de la proteína (su secuencia de aminoácidos y las coordenadas de sus átomos). Cabe mencionar que a pesar de que en este punto se cuenta con la información de las coordenadas de los átomos de la cadena lateral de todos los aminoácidos, esta información no se utiliza, dado que el proceso de optimización comienza una vez que se crea la solución inicial (Sección 3.1.5). En este punto se fuerza a seleccionar un rotámero de la biblioteca para cada aminoácido de la estructura de la proteína. Después de leer la estructura de la proteína también se leen los archivos que contienen la biblioteca de rotámeros y los archivos de parámetros para las funciones de energía utilizadas.

En este paso también se crean los directorios en donde se almacenarán los archivos generados por el algoritmo, los cuales incluyen al archivo de la estructura predicha, así como archivos con información extra sobre las características de la estructura predicha, como lo son los valores tanto iniciales como finales para cada término de la función de energía, así como los rotámeros utilizados en la estructura inicial y en la estructura predicha.

3.1.2. Creación de un grid con los átomos de la proteína (línea 2 del Algoritmo 3)

Algunos tipos de energía de interacción entre átomos dependen de la distancia que separa dichos átomos (*e.g.* Van der Waals y potencial electrostático). Sin embargo, cuando la distancia entre ambos átomos supera un cierto umbral, la energía de interacción disminuye tanto que se vuelve despreciable.

Normalmente, para calcular la energía de interacción entre un átomo i con respecto al resto de los átomos de la proteína, se tendría que tomar cada átomo $j \neq i$ y medir la distancia entre este y el átomo i para así calcular el valor de la energía de interacción. Sin embargo, este proceso es ineficiente ($O(n)$ por cada átomo), debido a que la energía de interacción del átomo i será relevante únicamente a aquellos átomos que se encuentren a una cierta distancia de este. Los átomos j que se encuentran lo suficientemente cerca de i como para generar un valor de interacción relevante, por lo general representan únicamente una fracción de tamaño constante del total de átomos de la proteína. Por consiguiente, la mayoría de las energías de

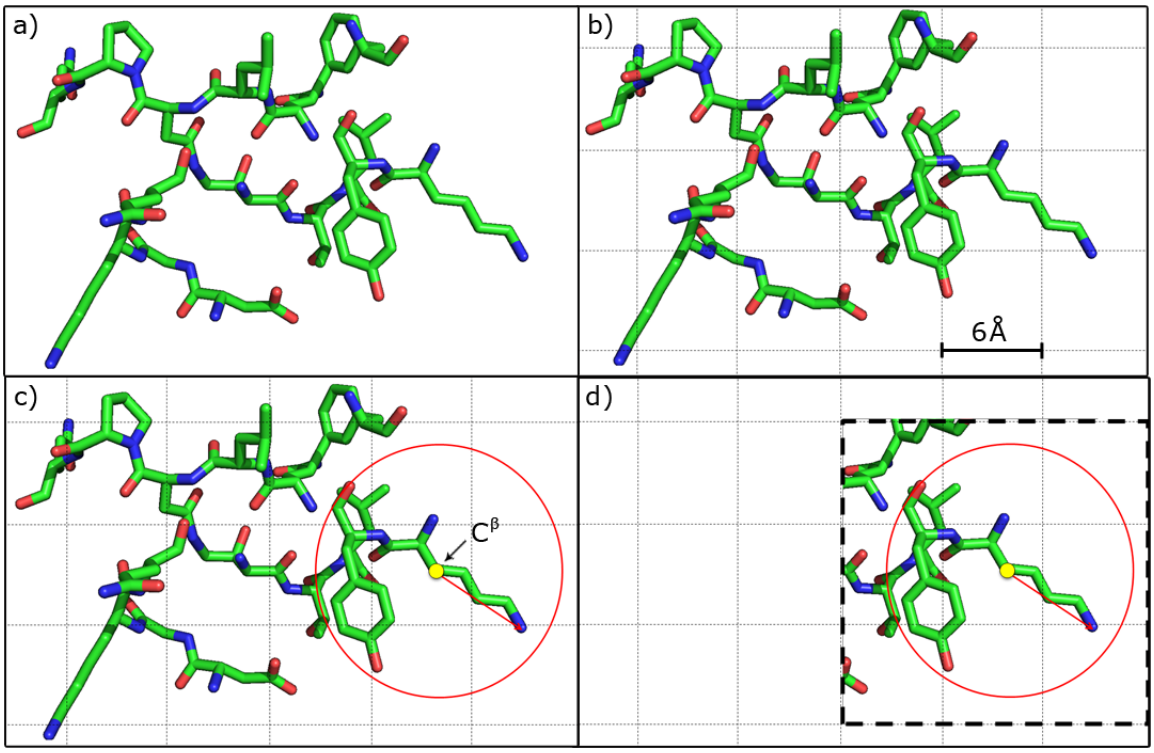

Figura 6: Creación del grid de átomos.
a) Estructura de la proteína. **b)** A cada átomo de la proteína se le asigna una celda del grid según sus coordenadas. **c)** Radio de interacción de un aminoácido, el cual tiene como centro el C^{β} de dicho aminoácido. **d)** Al calcular la energía, cada aminoácido considera únicamente los aminoácidos que se encuentren en su misma celda o celdas adyacentes.

interacción que involucren al átomo i calculadas de esta manera tendrán un valor de cero.

Para evitar cálculos innecesarios al momento de calcular la energía de interacción entre las distintas parejas de átomos, se construyó un grid tridimensional para almacenar todos los átomos de la proteína. En este grid a cada átomo se le asigna una celda según sus coordenadas (x, y, z).

Con esto se puede obtener rápidamente todos los átomos que se encuentren adyacentes a una celda en específico. Con esta estructura se obtiene la lista de los aminoácidos más cercanos a un aminoácido i; de esta manera se escogen los aminoácidos que tienen una mayor probabilidad de tener átomos que generen energía de interacción relevante, eliminando así muchos cálculos innecesarios. Sin embargo, es posible que se incluyan algunos aminoácidos que a pesar de que estén dentro de las celdas adyacentes en el grid, sus átomos estén lo suficientemente alejados como para no generar una energía de interacción relevante.

En la Figura 6 se muestra un ejemplo de cómo se utiliza el grid de átomos para obtener la

lista de aminoácidos cercanos a un aminoácido específico. En **a)** se muestra la estructura de la proteína. En **b)** se crea el grid asignándole a cada átomo una celda según sus coordenadas. El grid es tridimensional a pesar de que en la figura se aprecian únicamente dos dimensiones. El tamaño de cada celda es de 6Å de longitud por lado*. En **c)** se muestra el radio de interacción de un aminoácido seleccionado, el cual se denota por el círculo; dicho radio de interacción tiene como centro el átomo C^{β}. En **d)** se genera la lista de aminoácidos cercanos utilizando los áminoácidos que tengan por lo menos un átomo dentro de la celda que contiene al C^{β} o sus celdas adyacentes.

Esta lista de aminoácidos cercanos se calcula para cada aminoácido de la proteína. Cabe mencionar que esta lista genera una sóla vez para cada proteína, ya que la misma se almacena y se recupera cada vez que esta información se requiere. Es posible que dos átomos de un mismo aminoácido se asignen a celdas diferentes; sin embargo esto no genera problemas, ya que con que un sólo átomo se encuentre en las celdas adyacentes, se agrega el aminoácido completo a la lista.

El tiempo de creación del grid es $O(n)$, donde n es el número total de átomos en la proteína. La lista de aminoácidos cercanos creada se almacena durante el cálculo de las interacciones (Sección 3.1.3) sin agregarle costo asintótico adicional a dicho proceso.

3.1.3. Cálculo de todas las interacciones de la proteína (línea 3 del Algoritmo 3)

Sea $\mathbf{r} = (r_1, r_2, \cdots, r_n)$ el vector que contiene el rotámero seleccionado r_i para el aminoácido i de la proteína, entonces una forma sencilla de calcular la energía total de la proteína sería:

$$E(\mathbf{r}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} E(r_i, r_j), \tag{25}$$

$E(r_i, r_j)$ representa la energía de interacción entre el aminoácido i y el aminoácido j teniendo seleccionado el rotámero r_i y r_j, respectivamente. Se dice que un aminoácido i interacciona con un aminoácido j si por lo menos un átomo a_i del aminoácido i se encuentra dentro de

* Para definir el tamaño de las celdas del grid se usó como criterio el mayor radio de interacción de un átomo (en este caso el carbono con 1.3Å), este valor se multiplica por cuatro, y se le suma una pequeña cantidad de guarda para garantizar que se encuentren todos los átomos con los que puede generar un valor de interacción. Esto debido a que la mayoría de las funciones de energía asignan un valor relevante cuando la distancia entre ámbos átomos es menor al doble de la suma de sus radios.

un cierto umbral de distancia de un átomo a_j del aminoácido j. Esto se debe a que si la distancia entre la pareja de átomos se encuentra dentro del umbral de distancia, entonces se les asigna un valor de energía de interacción diferente de cero.

Durante el proceso de optimización, el cálculo de la energía total de la proteína es una de las operaciones que se realizan con mayor frecuencia, por lo que reduciendo el número de operaciones que se realizan cada vez que se calcula la energía total, se estará reduciendo el tiempo de ejecución del algoritmo.

La energía de interacción entre una pareja de aminoácidos $E(r_i, r_j)$ no cambia mientras ambos aminoácidos mantengan su respectivo estado rotamérico, es decir, la energía de interacción entre dos aminoácidos se mantiene igual siempre y cuando ninguno de los dos aminoácidos cambie la selección de su rotámero.

De lo anteriormente expuesto se puede observar que resultará beneficioso calcular una sola vez la energía de interacción entre una pareja de átomos en cierto estado rotamérico y almacenar en una estructura de datos el resultado obtenido. De esta manera, se puede cambiar la forma en la que se realiza el cálculo de la energía total de la proteína, de $O(n^2)$ cálculos de interacción, donde n es el número de átomos de la proteína, a simplemente una sumatoria de valores de energía según el rotámero seleccionado para cada aminoácido. Es decir, se puede reducir la complejidad del cálculo de energía entre una pareja de aminoácidos a $O(1)$ una vez que se han almacenado estos valores de energía para cada aminoácido con cada uno de sus rotámeros, por lo que el tiempo total para calcular la energía de la proteína es de $O(n)$, donde n es el número de aminoácidos totales de la proteína.

Los cálculos de interacciones entre aminoácidos en sus distintos estados rotaméricos no son requeridos únicamente por la heurística al momento de hacer la optimización, sino que también por el algoritmo DEE. Como consecuencia al generar esta estructura de datos que almacena todas las interacciones entre todas las parejas de aminoácidos con cada uno de sus distintos rotámeros también ayuda a reducir el tiempo del algoritmo DEE.

La estructura de datos que se eligió para almacenar todas las interacciones que ocurren entre las parejas de aminoácidos con cada uno de sus respectivos rotámeros es una tabla hash anidada de cuatro niveles, en donde se utilizan cuatro índices distintos para recuperar la energía de interacción entre una pareja de aminoácidos en un cierto estado rotamérico. Los índices son: i, que representa el primer aminoácido, el segundo índice r representa el

rotámero utilizado por el aminoácido i, el tercer índice j representa el segundo aminoácido, mientras que el cuarto índice s representa el rotámero utilizado por el aminoácido j. Cabe mencionar que si el aminoácido j no se encuentra dentro de la lista de aminoácidos cercanos del aminoácido i, la cual se generó utilizando el grid de átomos (Sección 3.1.2), entonces el valor de energía que se retorna es de cero. En el Apéndice A.1 se explica más detalladamente la manera en la que está estructurada la tabla hash de interacciones.

El tamaño de esta tabla es de $O(np^2)$, donde n es el número de aminoácidos de la proteína, mientras que p es el número máximo de rotámeros por aminoácido. Debido a que la lista de aminoácidos cercanos que utiliza el grid de átomos se crea antes de llevar a cabo el cálculo de interacciones en la proteína, se logra evitar un valor cuadrático en n, ya que gracias al grid el número de aminoácidos cercanos se mantiene en un valor constante (alrededor de 30 aminoácidos en promedio).

3.1.4. DEE - Eliminación de callejones sin salida (línea 4 del Algoritmo 3)

Como se mencionó en la Sección 3, el propósito del algoritmo DEE es el de eliminar rotámeros que generen valores altos de energía, y por lo tanto no puedan formar parte de la conformación de energía mínima global (GMEC). Además de descartar estos rotámeros también se eliminaron los rotámeros cuya probabilidad asignada en la biblioteca de rotámeros es menor a 0.01. El algoritmo implementado de DEE utiliza el criterio de Goldstein (Goldstein, 1994), el cual se explica a detalle en la Sección 2.4.

3.1.5. Creación de la solución inicial (línea 5 del Algoritmo 3)

Una vez que se ha reducido el número total de rotámeros con el algoritmo de DEE, se procede a seleccionar los rotámeros que conformarán la estructura de la solución inicial. El proceso de crear la solución inicial para cada aminoácido i consiste en seleccionar al rotámero con la mayor probabilidad de la lista de rotámeros que se eliminaron mediante el algoritmo de DEE. En el caso de que dos o más rotámeros compartan el valor máximo de probabilidad, entonces se utiliza el primer rotámero que se encontró con dicho valor de probabilidad, respetando el orden de la biblioteca de rotámeros.

3.1.6. Ciclo principal del algoritmo (líneas 8-36 del Algoritmo 3)

El ciclo principal del algoritmo (líneas 8-36 del Algoritmo 3) consiste en llevar a cabo la optimización combinatoria mediante un algoritmo de búsqueda tabú. Durante cada una de las iteraciones del algoritmo se llevan a cabo las siguientes operaciones:

1. Se selecciona aleatoriamente un aminoácido i de la proteína, dicho aminoácido i tiene que ser uno que no se encuentre en la lista tabú LT de longitud LT_MAX. El proceso que se lleva a cabo en los siguientes pasos consiste en explorar los rotámeros del aminoácido i, por lo tanto, en este paso no se seleccionan aminoácidos de tipo Alanina o Glicina, ya que ninguno de estos aminoácidos cuenta con los átomos suficientes en su cadena lateral para formar algún ángulo de torsión, y por lo tanto, no cuentan con ningún rotámero.

2. Se prueba cada uno de los rotámeros r del aminoácido i en busca de colisiones con los aminoácidos cercanos, de los cuales se tiene una lista (obtenida gracias a la estructura de grid), generando los siguientes posibles escenarios:

   - En caso de que algún rotámero r' de i genere una colisión con otro aminoácido j, entonces se fija el rotámero r' y se empieza a iterar con los rotámeros s del aminoácido j. Una vez que se han explorado todos los rotámeros del aminoácido j, se selecciona de entre éstos el rotámero s^* el cual genere la menor energía para la conformación de la proteína en su estado actual, es decir, usando tanto el rotámero r' para el aminoácido i como el rotámero s^* para el aminoácido j, por lo que ahora s^* es el nuevo rotámero de j. Este proceso se repite por cada aminoácido j que colisione con algún rotámero del aminoácido i. Al terminar de explorar todos los rotámeros r' de i, se selecciona el rotámero r^* que haya generado la menor energía para ser la conformación que se le asigne al aminoácido i.
   - En caso de que ningún rotámero de i colisione con algún otro aminoácido, entonces únicamente se selecciona el rotámero r^* de i que genere la menor energía de interacción con el resto de la proteína.

   El peor caso en tiempo de cómputo se da cuando el aminoácido i colisione con todos sus aminoácidos cercanos; sin embargo, es poco frecuente que un solo rotámero r_i colisione

con más de 2 aminoácidos. Por lo tanto, este ciclo se ejecuta $O(p)$ veces por cada aminoácido i.

3. Se agrega el aminoácido i a la lista tabú para evitar que sus rotámeros se exploren nuevamente en un periodo corto de tiempo. Si la lista ya contiene LT_MAX elementos, entonces se elimina el índice que lleve más tiempo dentro de LT. En la Sección 4.2.1 se especifica el tamaño máximo que se le asignó a la lista tabú.

4. Posteriormente, si se llegó a la mitad ($\alpha = 1/2$) del número máximo de iteraciones (MAX_ITER), entonces se ejecuta el algoritmo voraz presentado en la Sección 3.1.7.

Cabe mencionar que cuando se exploran los rotámeros de un aminoácido, esto se hace únicamente con los rotámeros que no fueron eliminados por el algoritmo DEE (Sección 2.4).

La estrategia utilizada está inspirada en el algoritmo CIS-RR (Cao *et al.*, 2011), el cual realiza el mismo proceso de fijar el rotámero del aminoácido i para explorar los rotámeros del aminoácido j en caso de una colisión. Sin embargo, en el algoritmo CIS-RR, el proceso se realiza iterando desde el aminoácido más expuesto al menos expuesto, hasta que la energía llega a un óptimo local. En la Sección 2.5.3 se explica con mayor detalle el funcionamiento del algoritmo CIS-RR.

3.1.7. Algoritmo voraz para reducir el número de colisiones (línea 34 del Algoritmo 3)

El reportar un número bajo de colisiones no es una métrica estándar al comparar los resultados de algoritmos para el PSCPP, sin embargo, esto es una característica deseable, ya que un número bajo de colisiones da más seguridad de que la estructura predicha es físicamente posible. En proteínas reales las colisiones no ocurren, por ejemplo, ninguna proteína en los conjuntos de pruebas utilizados en los experimentos (Sección 4) tiene colisión en el archivo .pdb que contiene su estructura.

Se implementó un algoritmo que usa una estrategia voraz con el propósito de reducir el número de colisiones de una estructura dada. La estrategia del algoritmo consiste en recorrer la proteína desde el primer hasta el último aminoácido, y para cada aminoácido i, escoger el rotámero r_i^c, que genera el menor número de colisiones entre el aminoácido i y el resto de la proteína; el pseudocódigo se muestra en el Algoritmo 4. El tiempo de ejecución de dicho

algoritmo es $O(np)$, donde n es el número total de aminoácidos de la proteína, mientras que p es el número máximo de rotámeros por aminoácido.

Algoritmo 4: Algoritmo voraz para reducir el número de colisiones.

Entrada : Estructura de la solución actual **r**
Salida : Una solución $\mathbf{r}'$ con un número menor o igual de colisiones.

```
1  mejoresColisiones = Número total de colisiones de la solución r.
2  Por cada aminoácido i de la proteína hacer     //i = 1 hasta n
3  |  r_i^c = r_i.
4  |  Por cada rotámero r'_i del aminoácido i que no fue eliminado por DEE hacer
5  |  |  r' = (r_1, r_2, ..., r'_i, ..., r_n).
6  |  |  colisiones = Número total de colisiones de la solución r'.
7  |  |  Si colisiones < mejoresColisiones entonces
8  |  |  |  mejoresColisiones = colisiones.
9  |  |  |  r_i^c = r'_i.
10 |  |  fin
11 |  fin
12 |  r' = (r_1, r_2, ..., r_i^c, ..., r_n).
13 fin
14 Retornar r' como la solución optimizada en cuanto a colisiones.
```

3.1.8. Generar archivos de resultado de la solución con la menor energía encontrada (línea 38 del Algoritmo 3)

El algoritmo de búsqueda tabú explora los aminoácidos de la proteína de manera aleatoria, motivo por el cual el resultado puede variar entre distintas ejecuciones. Para facilitar múltiples ejecuciones del algoritmo se le agregó al programa una variable para seleccionar el número de ejecuciones que se desean realizar para todas las proteínas, por lo que los archivos de resultados que se mencionarán a continuación se generan en carpetas diferentes, una por cada ejecución distinta del algoritmo.

Una vez que el algoritmo termina la ejecución del ciclo principal, se obtiene una estructura de datos, la cual contiene las coordenadas finales de cada átomo de la proteína en la posición predicha. Con esta información se procede a generar un archivo con formato .pdb. Éste es el archivo de resultado principal, ya que contiene toda la información referente a la estructura de la proteína predicha. Además de la estructura de la proteína se generan otros archivos, los cuales proporcionan información adicional sobre la optimización y las estructuras predichas. Uno de los archivos que se generan contiene la precisión de cada proteína con respecto al archivo .pdb que se recibió como entrada, para esta precisión se considera de igual manera un

margen de error de 40°. Otro archivo generado contiene la energía tanto de la solución inicial como de la final para cada proteína, así como los índices de los rotámeros que se utilizaron en ambos casos.

Además de estos archivos mencionados, también se generan otros tres, los cuales contienen información recopilada de todas las ejecuciones que se hayan realizado. El primer archivo contiene la mejor precisión que se logró obtener para cada proteína, indicando en cuál ejecución se generó dicha estructura. El segundo archivo contiene la energía tanto inicial como final de todas las corridas realizadas, ordenadas por número de ejecución y agrupadas por proteína. El tercer archivo contiene los tiempos de ejecución; por cada proteína se muestra el tiempo invertido en cada uno de los tres procesos que consumen el mayor tiempo, los cuales son: la creación de la estructura de datos que contiene todas las interacciones de la proteína (Sección 3.1.3), eliminación de rotámeros por DEE (Sección 3.1.4) y el algoritmo de optimización utilizado.

3.2. Algoritmo de recocido simulado

El segundo algoritmo de optimización implementado es una metaheurística tipo recocido simulado (Kirkpatrick *et al.*, 1983). En este tipo de algoritmos la probabilidad de seleccionar una solución de peor calidad que la actual depende de una variable denominada temperatura. Esta variable se decrementa iteración tras iteración hasta llegar a un mínimo valor cercano a cero.

En el Algoritmo 5 se muestra una versión general del algoritmo de recocido simulado. El algoritmo implementado sigue la estrategia mostrada, únicamente agregando los detalles específicos del problema abordado en este trabajo.

De igual manera que con el algoritmo de búsqueda tabú, el algoritmo de recocido simulado se hibridó utilizando los mismos algoritmos, tanto el de eliminación de callejones sin salida como el algoritmo voraz para reducir el número de colisiones.

En el Algoritmo 6 se muestra el pseudocódigo implementado, y dado que para el algoritmo de recocido simulado se utilizan la mayoría de los puntos que fueron ya explicados en el algoritmo de búsqueda tabú (Sección 3.1) a continuación se explican con mayor detalle únicamente los puntos restantes. Los números generados de manera aleatoria para este algoritmo se toman de una distribución uniforme (función nextInt de la clase Random del

Algoritmo 5: Algoritmo de recocido simulado simplificado.

Entrada : Solución inicial S_0
Salida : Solución final S_{best}

```
1  S_best ← S_0
2  E_best ← energía(S_best)
3  k ← 0
4  Mientras k < k_max hacer   //mientras la temperatura no llegue a T_min
5  |  T ← k/k_max
6  |  S_candidato ← vecino(S_best)
7  |  E_candidato ← energía(S_candidato)
8  |  Si E_candidato < E_best entonces   //si S_candidato es mejor que S_best
9  |  |  S_best ← S_candidato
10 |  |  E_best ← E_candidato
11 |  Sino
12 |  |  Si P(E_candidato, E_best, T) > Random( ) entonces   //se acepta la solución
   |  |  candidato con probabilidad basada en T
13 |  |  |  S_best ← S_candidato
14 |  |  |  E_best ← E_candidato
15 |  |  fin
16 |  fin
17 |  k ← k + 1
18 fin
19 Retornar S_best
```

lenguaje Java).

3.2.1. Seleccionar una solución r′ en el vecindario de r (línea 13 del Algoritmo 6)

Para seleccionar un vecino $\mathbf{r}'$ en el vecindario de $\mathbf{r} = (r_1, r_2, \cdots, r_n)$ se implementaron cinco métodos distintos para la generación del vecindario, los cuales se explican a continuación:

Método aleatorio simple (Algoritmo 7)

El método más simple e intuitivo para generar una solución $\mathbf{r}'$ en el vecindario de $\mathbf{r}$ consiste en escoger un rotámero $r_i' \neq r_i$ de manera aleatoria de la lista de rotámeros que el algoritmo DEE no eliminó. La estrategia de este método no consiste en llegar a buenas estructuras desde el principio de la optimización, si no en aprovechar las iteraciones en donde la temperatura es cercana a T_{max} para explorar más ampliamente el espacio de soluciones. Esto debido a que a temperaturas cercanas a T_{max} es más probable elegir soluciones de menor calidad.

Algoritmo 6: Algoritmo propuesto tipo recocido simulado.

Entrada : Proteína en formato PDB, biblioteca de rotámeros, MAX_ITER
Salida : Una predicción de la estructura de la proteína

```
Inicialización de estructuras de datos y directorios.
Creación de un grid con los átomos de la proteína.
Cálculo de todas las interacciones de la proteína.
Aplicación del método DEE.
Creación de la solución inicial r.
mejorEnergía = Eini // Energía de la estructura inicial.
T_max = MAX_ITER
T = T_max
N = Número total de aminoácidos en la proteína
Para n = 0 hasta el número máximo de iteraciones (MAX_ITER) hacer
    Para m = 0 hasta N hacer
        Seleccionar un aminoácido i de manera aleatoria.
        Seleccionar una solución r' en el vecindario de r cambiando el rotámero del
        aminoácido i.
        Energía = Energía de la solución r'.
        ΔE = Energía − mejorEnergía
        Si Energía < mejorEnergía entonces
            Sustituir la solución r por su vecino r'.
            mejorEnergía = Energía
        Sino
            Sustituir la solución r por su vecino r' con probabilidad e^(−ΔE/T).
        fin
    fin
    Si el número de generaciones es la mitad de MAX_ITER entonces
        Ejecutar el algoritmo voraz para colisiones (Algoritmo 4).
    fin
    T = T_max − n − 1 // Decrementa la temperatura.
fin
Ejecutar algoritmo de Búsqueda Local (Algoritmo 13).
Ejecutar el algoritmo voraz para reducir el número de colisiones (Algoritmo 4).
Generar archivos de resultado de la solución con la menor energía encontrada.
```

Algoritmo 7: Método aleatorio simple.

Entrada : Índice i del aminoácido seleccionado para buscar el nuevo vecino de $\mathbf{r}$
Salida : Solución $\mathbf{r}'$ en el vecindario de $\mathbf{r}$

$\mathbf{r} = (r_1, r_2, \cdots, r_i, \cdots, r_n)$.
Repetir
  Seleccionar de manera aleatoria un rotámero r_i' de la lista de rotámeros no eliminados por el algoritmo DEE para el aminoácido i.
hasta que $r_i' \neq r_i$
$\mathbf{r}' = (r_1, r_2, \cdots, r_i', \cdots, r_n)$.
Retornar $\mathbf{r}'$ como la solución vecina de $\mathbf{r}$.

Método aleatorio simple + verificación de colisión (Algoritmo 8)

Al seleccionar un rotámero r_i' de manera aleatoria para el aminoácido i, se pueden generar colisiones con algún otro aminoácido j. Esto debido a que no se toma en cuenta el nivel de energía que genera r_i' al momento de seleccionarlo aleatoriamente; por lo que un aumento del valor de la energía, producto del cambio de r_i por r_i', podría indicar una colisión atómica. Por lo tanto, para este tipo de vecindario se modificó el método simple agregando una verificación de colisiones.

Algoritmo 8: Método aleatorio simple + verificación de colisión.

Entrada : Índice i del aminoácido seleccionado para buscar el nuevo vecino de $\mathbf{r}$
Salida : Solución $\mathbf{r}'$ en el vecindario de $\mathbf{r}$

$\mathbf{r} = (r_1, r_2, \cdots, r_i, \cdots, r_n)$.
Repetir
  Seleccionar de manera aleatoria un rotámero r_i' de la lista de rotámeros no eliminados por el algoritmo DEE para el aminoácido i.
hasta que $r_i' \neq r_i$
Si el rotámero r_i' genera por lo menos una colisión **entonces**
  Se selecciona aleatoriamente un aminoácido j entre los aminoácidos con los que colisiona r_i'.
  Repetir
    Seleccionar de manera aleatoria un rotámero r_j' de la lista de rotámeros no eliminados por el algoritmo DEE para el aminoácido j.
  hasta que $r_j' \neq r_j$
  $\mathbf{r}' = (r_1, r_2, \cdots, r_i', \cdots, r_j', \cdots, r_n)$.
Sino
  $\mathbf{r}' = (r_1, r_2, \cdots, r_i', \cdots, r_n)$.
fin
Retornar $\mathbf{r}'$ como la solución vecina de $\mathbf{r}$.

Si el rotámero r_i' genera una colisión con uno o más aminoácidos j, de igual manera que

en el método simple, se selecciona para j un rotámero $r'_j \neq r_j$ de manera aleatoria de la lista de rotámeros que el algoritmo DEE no eliminó. Por lo tanto, el vecino $\mathbf{r}'$ de $\mathbf{r}$ consistiría en reemplazar r_i y r_j por r'_i y r'_j, respectivamente. En caso de que r'_i colisione con más de un aminoácido, se seleccionará el aminoácido j de manera aleatoria entre los aminoácidos con los que r'_i colisiona. Cabe mencionar que es posible que con los nuevos rotámeros r'_i y r'_j, siga existiendo una colisión.

En este método, de igual manera que con el anterior, se explora el espacio de soluciones durante las primeras iteraciones al seleccionar un rotámero al azar; sin embargo, como ya se mencionó, el hacer esto puede generar colisiones fácilmente. Lo que este método quiere lograr al seleccionar un nuevo rotámero para el aminoácido j es reducir la energía de i y evitar la colisión, mejorando la calidad de la selección del rotámero r'_i para el aminoácido i.

Método aleatorio simple + mejor colisión (Algoritmo 9)

Este método de encontrar una solución vecina de $\mathbf{r}$ es una variante del método anterior. La diferencia consiste en que en vez de escoger un rotámero r'_j de manera aleatoria, se enfoca en seleccionar el rotámero r^*_j que mejor interactúe con el rotámero r'_i seleccionado (en términos de energía); con lo cual se pueden obtener soluciones $\mathbf{r}'$ con un mejor valor de energía.

La modificación del método anterior consiste en que una vez que se tiene seleccionado un aminoácido j para el caso en el que el rotámero r'_i genera colisiones, la manera de elegir un nuevo rotámero para el aminoácido j es iterando entre todos los rotámeros que el algoritmo DEE no eliminó y seleccionando el rotámero r^*_j que genere la menor energía. Tomando en cuenta que el rotámero r'_i seleccionado ya forma parte de la estructura de la proteína.

Método simple: mejor vecino (Algoritmo 10)

De igual manera que con el método anterior, éste consiste en una modificación del método simple, cambiando su habilidad de exploración por la capacidad de obtener soluciones con menores niveles de energía.

La diferencia de este método con respecto al método simple consiste en que el rotámero r'_i no se selecciona de manera aleatoria, sino que se itera entre todos los rotámeros que el algoritmo DEE no eliminó para el aminoácido i y se selecciona el rotámero $r^*_i \neq r_i$ que genere la estructura con menor energía.

Algoritmo 9: Método aleatorio simple + mejor colisión.

Entrada : Índice i del aminoácido seleccionado para buscar el nuevo vecino de $\mathbf{r}$
Salida : Solución $\mathbf{r}'$ en el vecindario de $\mathbf{r}$

```
1  r = (r_1, r_2, ··· , r_i, ··· , r_n).
2  Repetir
3  |  Seleccionar de manera aleatoria un rotámero r'_i de la lista de rotámeros no
   |  eliminados por el algoritmo DEE para el aminoácido i.
4  hasta que r'_i ≠ r_i
5  r' = (r_1, r_2, ··· , r'_i, ··· , r_n).
6  Si el rotámero r'_i ocasiona por lo menos una colisión entonces
7  |  Se selecciona aleatoriamente un aminoácido j entre los aminoácidos con los que
   |  colisiona r'_i.
8  |  mejorEnergía = Energía total de la solución r'.
9  |  r*_j = r_j.
10 |  Por cada rotámero r'_j de j que no fue eliminado por DEE hacer
11 |  |  r' = (r_1, r_2, ··· , r'_i, ··· , r'_j, ··· , r_n).
12 |  |  energía = Energía total de la solución r'.
13 |  |  Si energía < mejorEnergía entonces
14 |  |  |  r*_j = r'_j.
15 |  |  |  mejorEnergía = energía.
16 |  |  fin
17 |  fin
18 |  r' = (r_1, r_2, ··· , r'_i, ··· , r*_j, ··· , r_n).
19 fin
20 Retornar r' como la solución vecina de r.
```

Método simple: mejor vecino + mejor colisión (Algoritmo 11)

Este método es una combinación de los métodos "simple: mejor vecino" y "aleatorio simple + mejor colisión", escogiendo tanto para el aminoácido i como para el aminoácido j el rotámero que proporcione la menor energía. Como se quiere que el vecino generado por este método sea diferente de $\mathbf{r}$, el mejor rotámero seleccionado r_i^* debe ser diferente de r_i, mientras que para r_j^* sí es válido el caso en el que $r_j^* = r_j$.

3.2.2. Aceptación de un vecino $\mathbf{r}'$ de acuerdo a la temperatura T del sistema (línea 20 del Algoritmo 6)

Como es característico de los algoritmos de recocido simulado, en dado caso de que la energía de una solución $\mathbf{r}'$ en el vecindario de $\mathbf{r}$ sea peor que la energía de $\mathbf{r}$, aún así es posible cambiar a dicha "peor solución" evaluando una expresión dependiente de la temperatura actual del sistema. Esta expresión permite que durante las primeras generaciones sea fácil explorar distintas soluciones, aunque su energía sea peor que la energía de la solución actual.

Algoritmo 10: Método simple: mejor vecino

Entrada : Índice i del aminoácido seleccionado para buscar el nuevo vecino de $\mathbf{r}$
Salida : Solución $\mathbf{r}'$ en el vecindario de $\mathbf{r}$

```
1  r = (r_1, r_2, ..., r_i, ..., r_n).
2  primerRotámero = true.
3  Por cada rotámero r'_i de i que no fue eliminado por DEE hacer
4  |  r' = (r_1, r_2, ..., r'_i, ..., r_n).
5  |  Si primerRotámero == true entonces
6  |  |  mejorEnergía = Energía total de la solución r'.
7  |  |  r*_i = r'_i.
8  |  |  primerRotámero = false.
9  |  Sino
10 |  |  energía = Energía total de la solución r'.
11 |  |  Si energía < mejorEnergía entonces
12 |  |  |  mejorEnregía = energía.
13 |  |  |  r*_i = r'_i.
14 |  |  fin
15 |  fin
16 fin
17 r' = (r_1, r_2, ..., r*_i, ..., r_n).
18 Retornar r' como la solución vecina de r.
```

Sin embargo, a medida que avanzan las iteraciones, se va haciendo cada vez más difícil para soluciones "malas" el ser aceptadas, lo cual obliga a que durante las últimas generaciones del algoritmo se seleccionen únicamente los vecinos con mejor valor de energía.

La condición que se utilizó en el algoritmo de recocido simulado para aceptar una solución de acuerdo a la temperatura actual del sistema se muestra en el Algoritmo 12.

En la línea 26 del recocido simulado (Algoritmo 6) se lleva a cabo el decremento de la temperatura. La temperatura es igual al número máximo de iteraciones (MAX_ITER) menos el número de iteraciones que han transcurrido (n) menos uno (debido a que n comienza en cero). Por ejemplo, si se van a ejecutar un total de 10 iteraciones, y durante la quinta iteración se quiere evaluar la condición de aceptación antes mencionada, la temperatura con la que se evaluaría sería de $T = 5$.

3.2.3. Ejecutar algoritmo de búsqueda local (línea 28 del Algoritmo 6)

Este algoritmo se utilizó como un algoritmo de optimización por si mismo, combinando los métodos de búsqueda de vecindario "simple: mejor vecino" y "simple: mejor vecino + mejor colisión". El mismo se explica con mayor detalle en la siguiente sección.

Algoritmo 11: Método simple: mejor vecino + mejor colisión.

Entrada : Índice i del aminoácido seleccionado para buscar el nuevo vecino de $\mathbf{r}$
Salida : Solución $\mathbf{r}'$ en el vecindario de $\mathbf{r}$

$\mathbf{r} = (r_1, r_2, \cdots, r_i, \cdots, r_n)$.
primerRotámero = *true*.
Por cada rotámero r'_i de i que no fue eliminado por DEE **hacer**
    $\mathbf{r}' = (r_1, r_2, \cdots, r'_i, \cdots, r_n)$.
    Si *primerRotámero* == *true* **entonces**
        mejorEnergía = Energía total de la solución $\mathbf{r}'$.
        $r^*_i = r'_i$.
        primerRotámero = *false*.
    Sino
        energía = Energía total de la solución $\mathbf{r}'$.
        Si *energía* < *mejorEnergía* **entonces**
            mejorEnregía = *energía*.
            $r^*_i = r'_i$.
        fin
    fin
    $\mathbf{r}' = (r_1, r_2, \cdots, r^*_i, \cdots, r_n)$.
    Si el rotámero r^*_i ocasiona por lo menos una colisión **entonces**
        Se selecciona aleatoriamente un aminoácido j entre los aminoácidos con los que colisiona r'_i.
        mejorEnergía = Energía total de la solución $\mathbf{r}'$.
        $r^*_j = r_j$.
        Por cada rotámero r'_j de j que no fue eliminado por DEE **hacer**
            $\mathbf{r}' = (r_1, r_2, \cdots, r^*_i, \cdots, r'_j, \cdots, r_n)$.
            energía = Energía total de la solución $\mathbf{r}'$.
            Si *energía* < *mejorEnergía* **entonces**
                $r^*_j = r'_j$.
                mejorEnergía = *energía*.
            fin
        fin
        $\mathbf{r}' = (r_1, r_2, \cdots, r^*_i, \cdots, r^*_j, \cdots, r_n)$.
    fin
fin
Retornar $\mathbf{r}'$ como la solución vecina de $\mathbf{r}$.

Algoritmo 12: Condición de aceptación para un vecino $\mathbf{r}'$ de acuerdo a la temperatura T del sistema.

Entrada : Soluciones $\mathbf{r}$ y $\mathbf{r}'$, y la temperatura actual del sistema T.
Salida : Un valor booleano, *verdadero* en caso de que se acepte la solución $\mathbf{r}'$.

```
1 Energía = Energía de la solución r.
2 energíaNueva = Energía de la solución r'.
3 ΔE = energíaNueva−Energía.
4 R =Número flotante aleatorio entre 0 y 1.
5 Si e^{−ΔE/T} > R entonces
6 |  Devolver verdadero.
7 Sino
8 |  Devolver falso.
9 fin
```

3.3. Algoritmo de búsqueda local

El tercer algoritmo de optimización es un algoritmo de búsqueda local, el cual surgió primero como una rutina para el algoritmo de recocido simulado (Sección 3.2). El objetivo de este algoritmo es visitar los mismos tipos de vecindario que en el algoritmo de recocido simulado, pero sin considerar la temperatura. Únicamente se toman en cuenta los niveles de energía, para así garantizar que al final de la ejecución del recocido simulado, la solución final sea un óptimo local para los tipos de vecindario que se están visitando.

A pesar de que este algoritmo se utilizó principalmente como una rutina para el recocido simulado, y dado que su condición de parada garantiza un óptimo local, se decidió comprobar el desempeño del mismo como algoritmo de optimización por su propia cuenta y comparar sus resultados con los obtenidos por los algoritmos de búsqueda tabú y recocido simulado.

La estrategia de este algoritmo consiste en recorrer de manera secuencial la proteína, es decir, desde el primer aminoácido hasta el último; y para cada aminoácido buscar un vecino con algunos de los métodos explicados en la Sección 3.2.1, repitiendo este proceso hasta que para ningún aminoácido i se encuentre un vecino que sea mejor. Para ésto se seleccionaron dos tipos de vecindario, cada uno con su respectivo método de navegación. Los métodos de navegación seleccionados son los que dieron los mejores resultados en pruebas preliminares: "simple: mejor vecino" y "simple: mejor vecino + mejor colisión". A continuación se explican los dos tipos de vecindario utilizados y la manera en la que se combinan.

Sea $\mathbf{r}_i = (r_1, r_2, \cdots, r_n)$ una solución, donde el índice i representa el aminoácido que ha sido seleccionado como pivote para la búsqueda de una solución vecina $\mathbf{r}'_i$, la cual consiste en una posible modificación del rotámero del aminoácido i. Una vez que se seleccionó una

solución vecina $\mathbf{r}'_i$ se realiza un aumento del índice del pivote; con esto se obtiene la solución $\mathbf{r}'_{i+1}$, dado que la proteína se recorre de manera secuencial como ya se mencionó.

El primer tipo de vecindario es el que se explora con el método simple: mejor vecino, en donde el vecindario $N(\mathbf{r}_i)$ de la solución $\mathbf{r}_i$ consiste de todos los rotámeros del aminoácido i que el algoritmo de DEE (Sección 2.4) no eliminó.

El segundo tipo de vecindario es el que se explora con el método simple - mejor vecino + mejor colisión, en donde el vecindario $N(\mathbf{r}_i)$ de la solución $\mathbf{r}_i$ es más complejo, ya que consiste tanto de los rótameros r_i del aminoácido i que DEE no eliminó, como de parejas de rotámeros $r_i - r_j$. Estas parejas de rotámeros pertenecen al vecindario $N(\mathbf{r}_i)$ cuando un rotámero r_i colisiona con un aminoácido j. Cada colisión de un rotámero r_i con un aminoácido j genera una cantidad de parejas igual al número de rotámeros de j que el algoritmo DEE no eliminó, ya que se crea una pareja $r_i - r_j$ por cada uno de los rotámeros de j. El método simple: mejor vecino + mejor colisión se encarga de seleccionar un rotámero o pareja de rotámeros del vecindario $N(\mathbf{r}_i)$.

El tamaño de un vecindario $N(\mathbf{r}_i)$ es $O(p)$. En el caso del método simple: mejor vecino, la construcción de dicho vecindario toma tiempo $O(p)$, mientras que para el método simple: mejor vecino + mejor colisión, toma tiempo $O(p^2)$.

La manera en la que se combinan estos dos tipos de vecindario y estrategias de navegación es bastante sencilla. Primero se recorre secuencialmente la proteína utilizando el método simple: mejor vecino hasta que se pueda hacer una iteración completa de la proteína sin encontrar ningún vecino que mejore la energía de la solución. Luego se cambia al método simple: mejor vecino + mejor colisión para iterar sobre la proteína. Si al terminar un recorrido de la proteína se encontró por lo menos un vecino que mejoró la energía de la solución, entonces se vuelve a repetir el proceso hasta que se llega a una solución $\mathbf{r}^*$, con la cual se puede recorrer la proteína primero por el método simple: mejor vecino y luego por el método simple: mejor vecino + mejor colisión sin encontrar ningún vecino que mejore dicha solución.

En el Algoritmo 13 se muestra el pseudocódigo implementado, y de igual manera que con el algoritmo de recocido simulado, una gran parte de sus procedimientos ya se explican en el algoritmo de búsqueda tabú (Sección 3.1) y en el algoritmo de recocido simulado (Sección 3.2.1).

Algoritmo 13: Algoritmo de búsqueda local.

Entrada : Proteína en formato PDB, biblioteca de rotámeros
Salida : Una predicción de la estructura de la proteína

```
1  Inicialización de estructuras de datos y directorios.
2  Creación de un grid con los átomos de la proteína.
3  Cálculo de todas las interacciones de la proteína.
4  Aplicación del método DEE.
5  Creación de la solución inicial r.
6  mejorEnergía = Energía total de la solución r.
7  Repetir
8  |  flag = false.
9  |  Por cada aminoácido i de la proteína hacer
10 |  |  r' = Vecino de la solución actual utilizando el método "simple: mejor vecino".
11 |  |  energía = Energía total de la solución r'.
12 |  |  Si energía < mejorEnergía entonces
13 |  |  |  mejorEnergía = energía.
14 |  |  |  flag = true.
15 |  |  fin
16 |  fin
17 |  Si flag == false entonces
18 |  |  Por cada aminoácido i de la proteína hacer
19 |  |  |  r' = Vecino de la solución actual utilizando el método "simple: mejor vecino
   |  |  |  + mejor colisión".
20 |  |  |  energía = Energía total de la solución r'.
21 |  |  |  Si energía < mejorEnergía entonces
22 |  |  |  |  mejorEnergía = energía.
23 |  |  |  |  flag = true.
24 |  |  |  fin
25 |  |  fin
26 |  fin
27 hasta que flag == false
28 Ejecutar el algoritmo voraz para reducir el número de colisiones (Algoritmo 4).
29 Generar archivos de resultado de la solución con la menor energía encontrada.
```

3.4. Diagrama de flujo de los algoritmos propuestos

En la Figura 7 se muestra un diagrama que describe la manera en la que se ejecuta el algoritmo y cómo fluye la información (estructura del PDB, biblioteca de rotámeros, energías de interacción, entre otras) entre los módulos del algoritmo. En el recuadro de línea de puntos se indican los pasos que forman parte del esquema de hibridación propuesto.

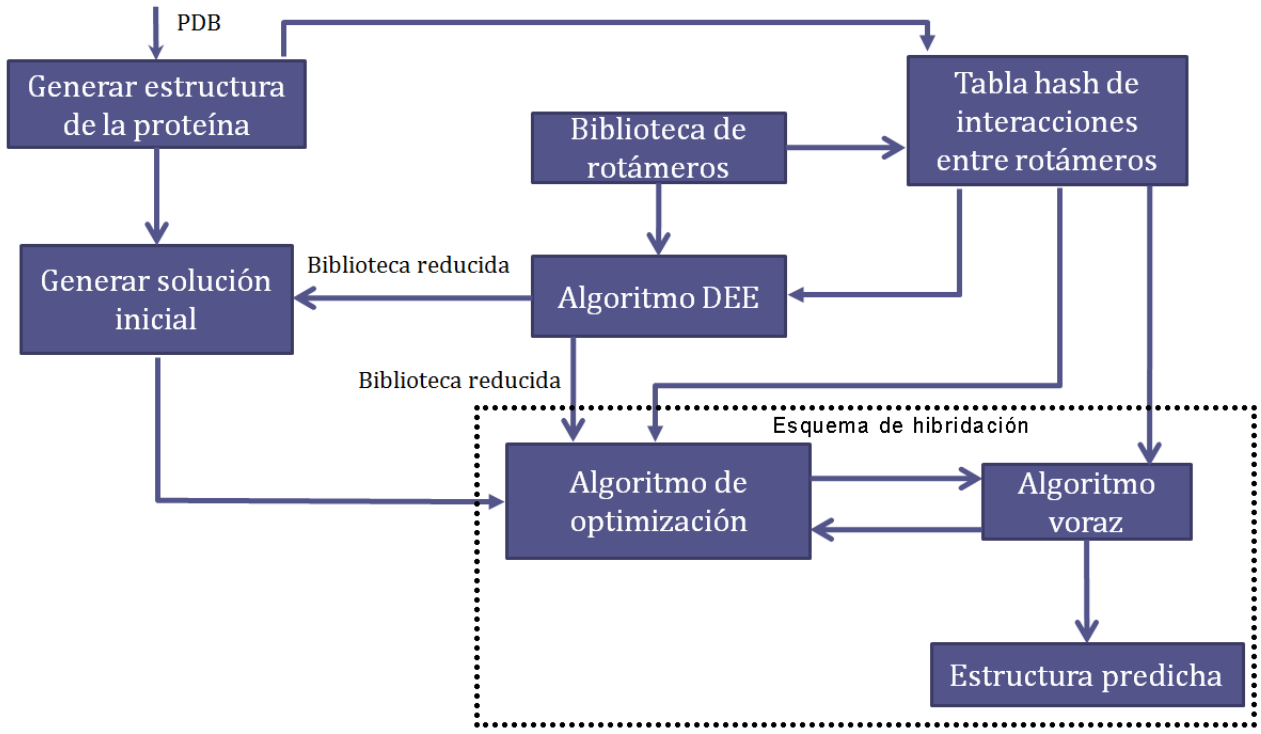

Figura 7: Diagrama de flujo de los algoritmos propuestos
Diagrama de flujo del algoritmo implementado, en donde el recuadro "Algoritmo de optimización" se refiere a una de las heurísticas implementadas: búsqueda tabú, recocido simulado y búsqueda local. El recuadro de línea punteada delimita los pasos que pertenecen al esquema de hibridación propuesto.

3.5. Parámetros del algoritmo

3.5.1. Función de energía

La función de energía utilizada en los algoritmos propuestos toma la siguiente forma:

$$E(\mathbf{r}) = \sum_{i}^{N-1} \sum_{j=i+1}^{N} E_{EV}(r_i, r_j) + \sum_{i} E_{RSCE}(r_i), \tag{26}$$

la cual cuenta con dos términos de energía diferentes, el primer término se denomina *Excluded Volume* (EV), EV es un potencial que crea una gran repulsión entre los átomos cuando se están acercando a una distancia próxima a su radio de van der Waals; el segundo término es un potencial de energía de Ramachandran de la cadena lateral (RSCE), el cual asigna un valor de energía según la probabilidad de encontrar los ángulos de torsión de cada aminoácido. Los parámetros de ambos términos fueron obtenidos del algoritmo MESHI (Kalisman *et al.*, 2005).

El término EV se calcula entre una pareja de átomos y se obtiene como se muestra a continuación:

$$E_{EV}(a_i, a_j) = \begin{cases} k\delta^4 & \text{si } a_i \text{ o } a_j \text{ pertenecen a la columna vertebral,} \\ \delta^2 & \text{si } a_i \text{ y } a_j \text{ pertenecen a una cadena lateral,} \\ 0 & \text{si } d > \delta. \end{cases} \tag{27}$$

Se tiene que $\delta = d - \sigma$, en donde d es la distancia entre el átomo a_i y el átomo a_j, σ es un parámetro que se asigna de acuerdo al tipo de átomo de a_i y a_j, así como al aminoácido al que pertenezcan. La energía del término EV se diferencia con respecto al potencial de Van der Waals en que EV no considera ningún potencial de atracción entre átomos, considera únicamente la repulsión entre los mismos.

El término de RSCE se obtuvo utilizando interpolación de splines (Subbotin, 1967) para suavizar los valores de una base de datos que contiene el análisis estadístico de los ángulos de torsión de la columna vertebral y cadena lateral observados en muchas estructuras de proteínas. Para calcular el valor de energía de RSCE de un aminoácido se requieren los ángulos de torsión de la columna vertebral ϕ y ψ, así como todos los ángulos de torsión de la cadena lateral de dicho aminoácido. El valor de energía que se obtiene intenta aproximar las siguientes probabilidades de obtener dichos ángulos de torsión:

$$E_{RSCE}(r_i) = \begin{cases} p(\phi_i, \psi_i) & \text{Si } i_\chi = 0, \\ p(\phi_i, \psi_i, \chi_1^i) & \text{Si } i_\chi = 1, \\ p(\phi_i, \psi_i, \chi_1^i) * p(\chi_2^i | \chi_1^i) & \text{Si } i_\chi = 2, \\ p(\phi_i, \psi_i, \chi_1^i) * p(\chi_2^i | \chi_1^i) * p(\chi_3^i | \chi_1^i) & \text{Si } i_\chi = 3, \\ p(\phi_i, \psi_i, \chi_1^i) * p(\chi_2^i | \chi_1^i) * p(\chi_3^i | \chi_1^i) * p(\chi_4^i | \chi_1^i) & \text{Si } i_\chi >= 4. \end{cases} \tag{28}$$

En donde i_χ representa el número de ángulos de torsión de la cadena lateral que tiene el aminoácido i. $p(\phi_i, \psi_i, \chi_1^i)$ representa la probabilidad conjunta de que el aminoácido i tenga los ángulos de torsión de la columna vertebral ϕ_i y ψ_i, así como el ángulo de torsión de la cadena lateral χ_1^i. $p(\chi_n^i|\chi_1^i)$ representa la probabilidad condicional de que el aminoácido i tenga de valor χ_n^i como el n-ésimo ángulo de torsión de la cadena lateral, teniendo en cuenta que tuvo χ_1^i para el primer ángulo de torsión.

3.5.2. Biblioteca de rotámeros

La biblioteca de rotámeros utilizada es una biblioteca independiente de la columna vertebral generada por Dunbrack y Cohen (1997) en su versión de Mayo del 2002. Dicha biblioteca se generó utilizando análisis estadístico Bayesiano de las conformaciones de la cadena lateral en proteínas del *Protein Data Bank* (PDB).

Capítulo 4

Experimentos y resultados

En este capítulo se presentan los experimentos, resultados y análisis realizados para comparar la calidad de las estructuras predichas por los algoritmos propuestos (Capítulo 3) con las obtenidas por algunos de los algoritmos del estado del arte (Sección 2.5).

4.1. Conjuntos de prueba

Para evaluar el desempeño de los algoritmos de optimización implementados se utilizaron tres conjuntos de prueba: uno de 65 proteínas, uno de 373 proteínas y uno de 723 proteínas.

El primer conjunto de 65 proteínas ha sido utilizado por varios métodos en la literatura (Peterson *et al.*, 2004; Jain *et al.*, 2006; Lu y Dousis, 2008) y se considera como un conjunto de proteínas de alta resolución (mejor a 1.2Å), una longitud de secuencia de entre 150 a 300 aminoácidos, y una identidad entre secuencias menor al 20 %.

El segundo conjunto de 373 protínas es un subconjunto del utilizado en el SCWRL4 (Krivov *et al.*, 2009). Las proteínas de este conjunto cuentan con una resolución mejor o igual a 1.8Å, una longitud de secuencia entre 50 a 450 aminoácidos, y un porcentaje de identidad entre secuencias no mayor al 30 %.

El tercer conjunto de 723 proteínas es un subconjunto del propuesto por Corona de la Fuente (2010), el cual consiste de proteínas con las siguientes características: resolución mejor o igual a 2Å, una sola cadena, longitud de secuencia de 40 a 400 aminoácidos, identidad máxima entre secuencias del 25 % y método experimental de obtención de estructura a través de cristalografía por rayos X.

En el caso del segundo y tercer conjunto de pruebas, únicamente se seleccionó un subconjunto de las proteínas; esto se debe a que el primer conjunto de pruebas ya considera las proteínas que fueron omitidas.

En el Apéndice A.2 se incluye la lista de identificadores del PDB de las proteínas de cada conjunto de pruebas.

4.2. Configuración de los algoritmos

Cada uno de los algoritmos de optimización implementados tiene su propio criterio de parada, además de algunas variables características según el tipo de heurística utilizada. A continuación se describen las configuraciones utilizadas para cada uno de ellos.

Los experimentos se realizaron en una computadora Mac Pro Server con procesador Intel Xenon de seis núcleos a 2.66GHz, sistema operativo OS X 10.8.3 y 64Gb de memoria RAM a 1333MHz.

4.2.1. Búsqueda tabú

Para el algoritmo de optimización tipo búsqueda tabú se usó como criterio de parada el número de iteraciones. Durante cada generación, el algoritmo de búsqueda tabú cambia los rotámeros para múltiples aminoácidos, no sólo para el aminoácido i que se seleccionó de manera aleatoria durante dicha iteración, sino que también puede cambiar el rotámero de cada aminoácido j que haya generado alguna colisión con el aminoácido i. Con base en esto, y con ayuda de experimentos preliminares, se encontró que el algoritmo de búsqueda tabú no requiere de un gran número de iteraciones para llegar a un óptimo local. Por este motivo, el número de generaciones utilizadas para este algoritmo fue igual a cinco veces el número de aminoácidos de la proteína.

Además del criterio de parada este algoritmo requiere un valor para la variable que representa el tamaño de la lista tabú a utilizar (LT_MAX). El tamaño para la lista tabú utilizada es igual al número de aminoácidos de la proteína dividido entre cuatro (redondeado al valor entero más cercano). De igual manera que con el número de iteraciones, se llegó a este valor durante experimentos preliminares a base de prueba y error.

El algoritmo de búsqueda tabú selecciona un aminoácido a explorar de manera aleatoria en cada iteración, toda vez que el mismo no se encuentre en la lista tabú. Por este motivo la estructura predicha para una misma proteína puede variar entre diferentes ejecuciones del algoritmo. Para poder determinar de manera más confiable la precisión promedio del algoritmo, se realizaron 30 ejecuciones para cada proteína.

4.2.2. Recocido simulado

De igual manera que con el algoritmo de búsqueda tabú, el algoritmo de recocido simulado utiliza el número de iteraciones como condición de parada. Sin embargo, el algoritmo de recocido simulado modifica un máximo de dos aminoácidos por vez, y dado que su propósito es el de disminuir lentamente la temperatura del sistema para obtener mejores resultados, se fijó que el número de iteraciones sea igual a 10 veces el número de aminoácidos en la proteína. Se denomina iteración al proceso de seleccionar un aminoácido y buscar si cuenta con un rotámero con mejor valor de energía.

El algoritmo selecciona un aminoácido a explorar de manera aleatoria en cada generación, por este motivo la estructura predicha para una misma proteína puede variar entre diferentes ejecuciones del algoritmo. Para poder determinar de manera más adecuada la precisión del algoritmo se realizaron 30 ejecuciones para cada proteína.

En la Sección 3.2.1 se describieron cinco implementaciones diferentes de la generación de un vecindario para el algoritmo de recocido simulado. De estos vecindarios el que dio mejores resultados en pruebas preliminares fue el de "mejor simple + mejor colisión", por lo tanto, los experimentos se realizaron utilizando este tipo de vecindario exclusivamente.

La temperatura inicial que se utilizó es de $T = 10$, con un decremento de una unidad por cada vez que se ejecute un ciclo de n iteraciones, con esto se llega a una temperatura final $T = 1$. n es el número total de aminoácidos de la proteína, por lo que un ciclo de n iteraciones es similar a recorrer la proteína de manera aleatoria una vez.

4.2.3. Búsqueda local

Como se mencionó en la Sección 3.3, el criterio de parada del algoritmo de búsqueda local se da cuando una solución $\mathbf{r}$ no encuentra solución alguna $\mathbf{r}'$ en su vecindario cuya energía sea mejor (*i.e.*, $\mathbf{r}$ es un óptimo local). Debido a esto, el algoritmo de búsqueda local no cuenta con ninguna variable para indicar el número máximo de iteraciones que se realizarán.

El algoritmo de búsqueda local no involucra ningún elemento aleatorio al momento de seleccionar el aminoácido i a explorar, ya que se recorre la proteína de manera secuencial desde el primer aminoácido hasta el último. Por este motivo, basta con ejecutar el algoritmo de búsqueda local una sola vez por cada proteína durante los experimentos.

4.3. Resultados

A continuación se muestran los resultados obtenidos de los experimentos realizados.

4.3.1. Desempeño del algoritmo voraz para reducir el número de colisiones

El algoritmo voraz (Algoritmo 4) se basa únicamente en el número de colisiones para cambiar la estructura de la proteína, por lo que con tal de evitar una colisión se podría aumentar el nivel de energía de la estructura. Se utilizó el conjunto de pruebas de 65 proteínas para analizar los resultados que proporciona este algoritmo y así poder determinar si los mismos son favorables o no.

Del conjunto de pruebas de 65 proteínas se detectó que son 19 (1a8q, 1arb, 1cc7, 1d4t, 1dhn, 1edg, 1hcl, 1ic6, 1ixh, 1koe, 1mml, 1ql0, 1qlw, 1qnj, 1qq4, 1rcf, 1vjs, 2baa, 7rsa) las que después de ejecutar un algoritmo de optimización de energía, se termina con una estructura que contiene colisiones. Por lo tanto, se analizarán únicamente estas 19 estructuras ya que si la estructura no contiene colisión alguna, el algoritmo voraz no hará ninguna modificación a dicha estructura.

Para cada una de las 19 proteínas antes mencionadas se ejecutaron cada uno de los tres algoritmos de optimización (búsqueda tabú, recocido simulado y búsqueda local), y una vez teniendo la estructura predicha por el algoritmo, se ejecutó el algoritmo voraz para colisiones. Posteriormente, se comparó la presición para χ_1 y χ_{1+2}, el nivel de energía y el número de colisiones de la estructura antes y después de ejecutar el algoritmo voraz. Los resultados obtenidos se muestran en las tablas 1, 2 y 3. Cada una de estas tablas está organizada de la siguiente manera. En la columna PDB se muestra el identificador de cada una de las proteínas. Las siguientes cuatro columnas llevan el encabezado del algoritmo de búsqueda tabú, en las cuales se muestra la precisión de las estructuras para $\chi_1(\%)$ y $\chi_{1+2}(\%)$, así como el número de colisiones de cada proteína y su energía. Las siguientes cuatro columnas contienen los mismos campos; sin embargo, pertenecen a las estructuras resultantes de ejecutar el algoritmo voraz a las estructuras generadas por el algoritmo de búsqueda tabú. Por último, las tres columnas restantes son una comparación porcentual de la precisión y energía de las estructuras antes y después del algoritmo voraz para colisiones. En estas columnas de comparación entre estructuras, si el porcentaje es positivo significa que la precisión o la

energía fue mayor en la estructura obtenida al aplicar el algoritmo voraz después del de búsqueda tabú, en caso de que sea negativo, entonces los valores de la estructura generada por la búsqueda tabú son mayores.

Antes de ejecutar el algoritmo voraz para este conjunto de 19 proteínas, fueron dos las proteínas (1arb y 1vjs) que tuvieron en su estructura más de 10 colisiones, las otras 17 proteínas tuvieron un promedio de 2.5 colisiones.

Una vez que se ejecutó el algoritmo voraz para reducir el número de colisiones, fueron ocho las proteínas (1cc7, 1d4t, 1hcl, 1ic6, 1qlw, 1qnj, 2baa y 7rsa) para las cuales se logró eliminar por completo las colisiones. Sin embargo, se encontraron seis proteínas (1a8q, 1dhn, 1edg, 1koe, 1mml y 1rfc) para las cuales el algoritmo voraz no pudo eliminar las colisiones.

En las tablas 1, 2 y 3, se puede observar que para las proteínas que tienen el mayor número de colisiones (1arb, 1cc7, 1qq4 y 1vjs) se logró eliminar la mayoría de las mismas, mejorando incluso la precisión de $\chi_1(\%)$ para dichas proteínas. Por otra parte para algunas proteínas con pocas colisiones (1d4t, 1hcl, 1ql0, 1qnj y 7rsa) la precisión de $\chi_1(\%)$ disminuyó: aunque también hubo tres proteínas con pocas colisiones (1ic6, 1ixh y 2baa) para las cuales se redujo el número de colisiones y la precisión de $\chi_1(\%)$ incrementó.

De los tres algoritmos de optimización, el que obtuvo el menor número de colisiones para este conjunto de proteínas fue el de recocido simulado, con un total de 71; mientras que el de búsqueda tabú fue el peor con un total de 77. A pesar de la diferencia en el número de colisiones entre los tres algoritmos, al final los tres métodos de optimización lograron reducir el número total de colisiones; con un total de 20 para recocido simulado y búsqueda local, y 21 para la búsqueda tabú.

Tabla 1: Desempeño del algoritmo voraz con búsqueda tabú.

	Búsqueda Tabú				**Búsqueda Tabú + Colisiones**				**Comparación**		
PDB	$\chi_1(\%)$	$\chi_{1+2}(\%)$	No.Cols	Energía	$\chi_1(\%)$	$\chi_{1+2}(\%)$	No.Cols	Energía	$\Delta\chi_1(\%)$	$\Delta\chi_{1+2}(\%)$	ΔEnergía
1a8q	84.89 %	64.57 %	1	8987	84.89 %	64.57 %	1	8987	0.00 %	0.00 %	0.00 %
1arb	83.66 %	68.50 %	14	25167	84.16 %	67.72 %	5	38856	0.50 %	-0.78 %	54.39 %
1cc7	78.79 %	52.17 %	7	1078	80.30 %	52.17 %	0	1315	1.51 %	0.00 %	21.93 %
1d4t	85.39 %	64.06 %	1	2986	84.27 %	62.50 %	0	2987	-1.12 %	-1.56 %	0.01 %
1dhn	81.90 %	62.65 %	3	1331	81.90 %	62.65 %	3	1331	0.00 %	0.00 %	0.00 %
1edg	85.11 %	62.20 %	1	4719	85.11 %	62.20 %	1	4719	0.00 %	0.00 %	0.00 %
1hcl	74.52 %	44.83 %	2	4499	74.13 %	44.83 %	0	4760	-0.39 %	0.00 %	5.82 %
1ic6	82.16 %	58.78 %	1	4395	82.16 %	58.02 %	0	4480	0.00 %	-0.76 %	1.95 %
1ixh	82.94 %	65.79 %	4	5182	82.94 %	65.79 %	1	5285	0.00 %	0.00 %	1.99 %
1koe	81.94 %	67.33 %	2	4533	81.94 %	67.33 %	2	4533	0.00 %	0.00 %	0.00 %
1mml	83.71 %	58.66 %	1	10255	83.71 %	58.66 %	1	10255	0.00 %	0.00 %	0.00 %
1ql0	87.44 %	68.49 %	5	3189	86.93 %	68.49 %	4	3190	-0.51 %	0.00 %	0.05 %
1qlw	84.23 %	67.69 %	5	3806	84.62 %	68.21 %	0	3862	0.39 %	0.52 %	1.49 %
1qnj	86.32 %	64.71 %	2	5036	85.26 %	64.71 %	0	7459	-1.06 %	0.00 %	48.11 %
1qq4	81.82 %	62.03 %	7	4115	82.52 %	63.29 %	1	4196	0.70 %	1.26 %	1.97 %
1rcf	87.32 %	70.00 %	1	1813	87.32 %	70.00 %	1	1813	0.00 %	0.00 %	0.00 %
1vjs	78.01 %	54.55 %	14	8201	78.77 %	54.87 %	1	8795	0.76 %	0.32 %	7.24 %
2baa	75.28 %	56.15 %	5	4021	75.28 %	56.15 %	0	4255	0.00 %	0.00 %	5.81 %
7rsa	74.31 %	56.72 %	1	1273	73.39 %	56.72 %	0	1282	-0.92 %	0.00 %	0.69 %
		Total:	**77**			**Total:**	**21**	**Promedio:**	**-0.01 %**	**-0.05 %**	**7.97 %**

Comparación de la calidad de las estructuras antes y después de aplicar el algoritmo voraz para colisiones en estructuras optimizadas por el algoritmo de búsqueda tabú.

Tabla 2: Desempeño del algoritmo voraz con recocido simulado.

	Recocido Simulado				**Recocido Simulado + Colisiones**				**Comparación**		
PDB	$\chi_1(\%)$	$\chi_{1+2}(\%)$	No.Cols	Energía	$\chi_1(\%)$	$\chi_{1+2}(\%)$	No.Cols	Energía	$\Delta\chi_1(\%)$	$\Delta\chi_{1+2}(\%)$	ΔEnergía
1a8q	84.89 %	63.43 %	1	8994	84.89 %	63.43 %	1	8994	0.00 %	0.00 %	0.00 %
1arb	82.18 %	68.50 %	14	25209	82.67 %	67.72 %	5	38897	0.49 %	-0.78 %	54.30 %
1cc7	83.33 %	56.52 %	7	1087	84.85 %	56.52 %	0	1323	1.52 %	0.00 %	21.76 %
1d4t	85.39 %	62.50 %	1	3005	84.27 %	60.94 %	0	3005	-1.12 %	-1.56 %	0.00 %
1dhn	78.10 %	60.24 %	3	1344	78.10 %	60.24 %	3	1344	0.00 %	0.00 %	0.00 %
1edg	83.89 %	59.06 %	0	4729	83.89 %	59.06 %	0	4729	0.00 %	0.00 %	0.00 %
1hcl	72.20 %	45.32 %	2	4515	71.81 %	45.32 %	0	4777	-0.39 %	0.00 %	5.80 %
1ic6	80.28 %	57.25 %	1	4402	80.28 %	56.49 %	0	4487	0.00 %	-0.76 %	1.95 %
1ixh	80.56 %	64.74 %	4	5248	80.56 %	64.74 %	1	5340	0.00 %	0.00 %	1.77 %
1koe	80.56 %	65.35 %	2	4227	80.56 %	65.35 %	2	4227	0.00 %	0.00 %	0.00 %
1mml	81.00 %	54.19 %	1	10262	81.00 %	54.19 %	1	10262	0.00 %	0.00 %	0.00 %
1ql0	85.93 %	65.75 %	5	3217	85.43 %	65.75 %	4	3217	-0.50 %	0.00 %	0.00 %
1qlw	82.69 %	64.62 %	5	3852	83.08 %	65.13 %	0	3852	0.39 %	0.51 %	0.00 %
1qnj	83.68 %	60.50 %	1	5052	83.16 %	60.50 %	0	5054	-0.52 %	0.00 %	0.04 %
1qq4	77.62 %	60.76 %	7	4161	78.32 %	62.03 %	1	4240	0.70 %	1.27 %	1.89 %
1rcf	83.80 %	68.18 %	1	1822	83.80 %	68.18 %	1	1822	0.00 %	0.00 %	0.00 %
1vjs	77.49 %	54.55 %	11	8207	78.01 %	54.55 %	1	8803	0.52 %	0.00 %	7.25 %
2baa	76.97 %	61.54 %	5	4032	76.97 %	61.54 %	0	4265	0.00 %	0.00 %	5.77 %
7rsa	72.48 %	59.70 %	0	1275	72.48 %	59.70 %	0	1284	0.00 %	0.00 %	0.69 %
		Total:	**71**			**Total:**	**20**	**Promedio:**	**0.06 %**	**-0.07 %**	**5.33 %**

Comparación de la calidad de las estructuras antes y después de aplicar el algoritmo voraz para colisiones en estructuras optimizadas por el algoritmo de recocido simulado.

Tabla 3: Desempeño del algoritmo voraz con búsqueda local.

	Búsqueda Local				**Búsqueda Local + Colisiones**				**Comparación**		
PDB	$\chi_1(\%)$	$\chi_{1+2}(\%)$	No.Cols	Energía	$\chi_1(\%)$	$\chi_{1+2}(\%)$	No.Cols	Energía	$\Delta\chi_1(\%)$	$\Delta\chi_{1+2}(\%)$	ΔEnergía
1a8q	84.00 %	64.57 %	1	8982	84.00 %	64.57 %	1	8982	0.00 %	0.00 %	0.00 %
1arb	83.66 %	67.72 %	14	25158	84.16 %	66.93 %	5	38846	0.50 %	-0.79 %	54.41 %
1cc7	80.30 %	52.17 %	7	1080	81.82 %	52.17 %	0	1317	1.52 %	0.00 %	21.89 %
1d4t	87.64 %	65.62 %	1	2979	86.52 %	64.06 %	0	2980	-1.12 %	-1.56 %	0.01 %
1dhn	80.95 %	62.65 %	3	1317	80.95 %	62.65 %	3	1317	0.00 %	0.00 %	0.00 %
1edg	84.19 %	60.63 %	0	4708	84.19 %	60.63 %	0	4708	0.00 %	0.00 %	0.00 %
1hcl	75.29 %	47.78 %	2	4489	74.90 %	47.78 %	0	4751	-0.39 %	0.00 %	5.83 %
1ic6	81.69 %	59.54 %	1	4375	81.69 %	58.78 %	0	4460	0.00 %	-0.76 %	1.96 %
1ixh	82.54 %	64.21 %	4	5177	82.54 %	64.21 %	1	5280	0.00 %	0.00 %	1.97 %
1koe	81.25 %	66.34 %	2	4219	81.25 %	66.34 %	2	4219	0.00 %	0.00 %	0.00 %
1mml	83.26 %	58.66 %	1	10253	83.26 %	58.66 %	1	10253	0.00 %	0.00 %	0.00 %
1ql0	87.94 %	68.49 %	5	3182	87.44 %	68.49 %	4	3184	-0.50 %	0.00 %	0.05 %
1qlw	84.23 %	67.69 %	5	3787	84.62 %	68.21 %	0	3851	0.39 %	0.52 %	1.68 %
1qnj	86.32 %	63.87 %	2	5035	85.26 %	63.87 %	0	7458	-1.06 %	0.00 %	48.12 %
1qq4	81.12 %	60.76 %	7	4114	81.82 %	62.03 %	1	4195	0.70 %	1.27 %	1.97 %
1rcf	86.62 %	69.09 %	1	1810	86.62 %	69.09 %	1	1810	0.00 %	0.00 %	0.00 %
1vjs	77.24 %	52.92 %	14	4114	78.01 %	53.25 %	1	4195	0.77 %	0.33 %	1.97 %
2baa	74.72 %	56.15 %	5	8194	74.72 %	56.15 %	0	8789	0.00 %	0.00 %	7.26 %
7rsa	75.23 %	58.21 %	1	4011	74.31 %	58.21 %	0	4244	-0.92 %	0.00 %	5.82 %
		Total:	**76**			**Total:**	**20**	**Promedio:**	**-0.01 %**	**-0.05 %**	**8.05 %**

Comparación de la calidad de las estructuras antes y después de aplicar el algoritmo voraz para colisiones en estructuras optimizadas por el algoritmo de búsqueda local.

Tabla 4: Análisis de los resultados del algoritmo voraz.

Algoritmo	$\Delta\chi_1(\%)$	$\Delta\chi_{1+2}(\%)$	ΔEnergía(%)	ΔNo.Cols(%)
Búsqueda Tabú	-0.01 %	-0.05 %	7.97 %	-72.73 %
Recocido Simulado	0.06 %	-0.07 %	5.33 %	-71.83 %
Búsqueda Local	-0.01 %	-0.05 %	8.05 %	-73.68 %
Promedio:	**0.01 %**	**-0.06 %**	**7.12 %**	**-72.75 %**

Valores promedio recopilados de las tablas 1, 2 y 3, los cuales se obtuvieron al ejecutar el algoritmo voraz para colisiones después de cada uno de los tres algoritmos de optimización. En esta tabla se muestra el aumento o decremento porcentual para la precisión, la energía y el número de colisiones para cada algoritmo.

En la Tabla 4 se muestran los valores promedio para los tres algoritmos. Se puede observar que en promedio sobre los tres algoritmos probados, el número de colisiones decrementó en un 72.75 %, mientras que la precisión aumentó un 0.01 % para χ_1 y decrementó un 0.06 % para χ_{1+2}. Sin embargo, los valores de energía aumentaron en un 7.12 %. Con base en estos resultados se decidió que para los tres algoritmos se ejecutará el algoritmo voraz para colisiones después de realizar el proceso de optimización de energía, siendo esta última estructura que se obtenga por el algoritmo voraz la que se considerará como la estructura predicha. Los aspectos que se consideraron para tomar esta decisión consisten principalmente en que el algoritmo voraz logra su objetivo de reducir el número de colisiones, pero aún más importante, sin afectar de manera considerable la precisión de la estructura. Debido a que la energía de la solución con menor número de colisiones aumenta, esta estructura no sería seleccionada como estructura final por el algoritmo de optimización a pesar de que dicha estructura es considerada como una mejor solución. Esto se debe a que el algoritmo se guía por la energía y no por el número de colisiones.

4.3.2. Resultados de los algoritmos implementados

Una vez que se ejecutaron los tres algoritmos de optimización (algoritmos 3, 6 y 13), se procedió a análizar los resultados generados, para así realizar una comparación entre los tres. La comparación se hizo tomando en cuenta cuatro aspectos: precisión absoluta, número de colisiones, RMSD y tiempo de ejecución.

En las tablas 5, 6 y 7, se muestra la calidad promedio de las estructuras predichas por cada uno de los tres algoritmos implementados para los conjuntos de pruebas de 65, 373 y

Tabla 5: Calidad promedio de las estructuras predichas para el conjunto de pruebas de 65 proteínas.

Algoritmo	$\chi_1(\%)$	$\chi_{1+2}(\%)$	No. Colisiones(Prom)	RMSD(Å)	T_{total}(min)
Búsqueda Tabú	82.58	62.79	20(0.3076)	0.8054	77.80
Recocido Simulado	82.54	62.61	20(0.3076)	0.8080	133.09
Búsqueda Local	82.13	62.19	20(0.3076)	0.8154	8.36

Los resultados se obtuvieron promediando la calidad de las estructuras predichas para todas las proteínas del conjunto de pruebas de tamaño 65. El valor entre paréntesis en la columna de colisiones representa el número de colisiones promedio por proteína, mientras que T_{total} representa el tiempo de ejecución total del algoritmo (las 30 ejecuciones de las 65 proteínas para el caso de búsqueda tabú y recocido simulado).

723 proteínas, respectivamente.

Para el primer conjunto de pruebas (65 proteínas, Tabla 5), el algoritmo que obtuvo la mejor precisión fue el de búsqueda tabú con 82.58 % para $\chi_1(\%)$ y 62.79 % para $\chi_{1+2}(\%)$. Sin embargo, los tres algoritmos presentaron precisiones similares, ya que el algoritmo de búsqueda tabú fue superior al de recocido simulado únicamente por 0.04 % en $\chi_1(\%)$ y 0.18 % para $\chi_{1+2}(\%)$, mientras que al de búsqueda local lo superó por 0.45 % en $\chi_1(\%)$ y 0.60 % en $\chi_{1+2}(\%)$.

En términos de colisiones los tres algoritmos obtuvieron el mismo valor, con un total de 20. Para el RMSD, el algoritmo que presentó el mejor resultado fue de igual manera el de búsqueda tabú con 0.8054Å, aunque superó a los algoritmos de recocido simulado y búsqueda local por un margen pequeño, 0.0026Å y 0.01Å, respectivamente. Como referencia de la magnitud del RMSD se puede decir que en la literatura un buen resultado en este criterio se considera un valor menor a 2Å (Shindyalov y Bourne, 1998).

Para el segundo conjunto de pruebas (373 proteínas, Tabla 6), nuevamente el algoritmo de búsqueda tabú fue el mejor de los tres métodos, con una precisión de 81.06 % para $\chi_1(\%)$ y 61.89 % para $\chi_{1+2}(\%)$; superando al de recocido simulado por 0.13 % en $\chi_1(\%)$ y 0.21 % en $\chi_{1+2}(\%)$, y al de búsqueda local por 0.52 % en $\chi_1(\%)$ y 0.68 % en $\chi_{1+2}(\%)$. De los tres conjuntos de pruebas, éste fue en el que se obtuvo la precisión más baja.

En cuanto a colisiones, tanto búsqueda tabú como recocido simulado fueron los mejores, con un total de 77 colisiones; mientras que el algoritmo de búsqueda local obtuvo un total de 80 colisiones. La diferencia de colisiones entre búsqueda local y recocido simulado fue en

Tabla 6: Calidad de las estructuras predichas para el conjunto de pruebas de 373 proteínas.

Algoritmo	$\chi_1(\%)$	$\chi_{1+2}(\%)$	No. Colisiones(Prom)	RMSD(Å)	T_{total}(min)
Búsqueda Tabú	81.06	61.89	77(0.2064)	0.8708	468.16
Recocido Simulado	80.93	61.68	77(0.2064)	0.8732	904.97
Búsqueda Local	80.54	61.21	80(0.2144)	0.8811	47.25

Los resultados se obtuvieron promediando la calidad de las estructuras predichas para todas las proteínas del conjunto de pruebas de tamaño 373. El valor entre paréntesis en la columna de colisiones representa el número de colisiones promedio por proteína, mientras que T_{total} representa el tiempo de ejecución total del algoritmo (las 30 ejecuciones de las 65 proteínas para el caso de búsqueda tabú y recocido simulado).

Tabla 7: Calidad de las estructuras predichas para el conjunto de pruebas de 723 proteínas.

Algoritmo	$\chi_1(\%)$	$\chi_{1+2}(\%)$	No. Colisiones(Prom)	RMSD(Å)	T_{total}(min)
Búsqueda Tabú	81.71	62.35	180(0.2489)	0.8483	956.68
Recocido Simulado	81.57	62.12	173(0.2392)	0.8502	1724.30
Búsqueda Local	81.26	61.74	174(0.2406)	0.8572	98.32

Los resultados se obtuvieron promediando la calidad de las estructuras predichas para todas las proteínas del conjunto de pruebas de tamaño 723. El valor entre paréntesis en la columna de colisiones representa el número de colisiones promedio por proteína, mientras que T_{total} representa el tiempo de ejecución total del algoritmo (las 30 ejecuciones de las 65 proteínas para el caso de búsqueda tabú y recocido simulado).

3 proteínas (1snt, 2dyu y 2i49), con una colisión extra en cada una de ellas; sin embargo, con respecto al de búsqueda tabú fueron 6 proteínas en las que hubo diferencias (1snt, 1tuo, 2ahf, 2e7a, 2i49 y 2o6x), ya que en algunas proteínas el algoritmo de búsqueda local obtuvo menos colisiones, pero en otras obtuvo más. Para el RMSD, el algoritmo de búsqueda tabú fue el mejor con 0.8708Å en promedio, superando a recocido simulado por 0.0024Å y al de búsqueda local por 0.0103Å.

Para el tercer conjunto de pruebas (723 proteínas, Tabla 7), de igual manera, el algoritmo de búsqueda tabú fue el que consiguió la mejor precisión, reportando un valor de 81.71 % para $\chi_1(\%)$ y 62.65 % para $\chi_{1+2}(\%)$; superando a recocido simulado por 0.14 % en $\chi_1(\%)$ y por 0.23 en $\chi_{1+2}(\%)$.

El algoritmo que obtuvo el menor número de colisiones fue el de recocido simulado con 173, superando al de búsqueda local que obtuvo 174; mientras que el de búsqueda tabú fue

Tabla 8: Calidad promedio de las estructuras predichas para los tres conjuntos de pruebas.

Algoritmo	$\chi_1(\%)$	$\chi_{1+2}(\%)$	Colisiones promedio	RMSD(Å)
Búsqueda Tabú	81.78	62.34	0.2543	0.8415
Recocido Simulado	81.68	62.13	0.2510	0.8438
Búsqueda Local	81.31	61.71	0.2542	0.8512
Desviación estándar	**0.2475**	**0.3207**	**0.0018**	**0.0050**

Los resultados se obtuvieron promediando la calidad de las estructuras predichas por los tres algoritmos propuestos en cada uno de los conjuntos de pruebas (65, 373 y 723 proteínas).

el peor en cuando a colisiones se refiere, con un total de 180. La diferencia de colisiones entre búsqueda tabú con respecto a las de búsqueda local y recocido simulado fue en 6 y 7 proteínas, respectivamente (1f2j, 1ru4, 1vr8, 2e3h, 2nfx y 2vbu, con 1r5y como proteína adicional en recocido simulado). La diferencia de igual manera fue de una sola colisión para todas las estructuras a excepción de 2vbu, con la cual hubo una diferencia de cinco colisiones. Para el RMSD, el algoritmo de búsqueda tabú fue el que se desempeñó mejor con 0.8483Å, superando al de recocido simulado por 0.0019Å y a la búsqueda local por 0.0089Å.

En la Tabla 8 se muestra la precisión promedio de los algoritmos implementados, la cual se obtuvo al promediar los resultados de los tres conjuntos de pruebas (tablas 5, 6 y 7), así como la desviación estándar entre los tres métodos para cada una de las medidas de calidad utilizadas. En esta tabla se puede observar que el algoritmo de búsqueda tabú fue el que mejor se desempeñó, ya que superó al recocido simulado y a la búsqueda local en todos los criterios a excepción de las colisiones, en donde el recocido simulado obtuvo los mejores resultados.

El tiempo de ejecución de los tres algoritmos varía mucho de uno a otro. Esto se debe principalmente a que cada uno de los algoritmos tiene un criterio de parada diferente, además de que el algoritmo de búsqueda local se ejecuta una sola vez, y no 30 veces como es el caso de búsqueda tabú y de recocido simulado.

En la Tabla 9 se muestra una comparación del tiempo de ejecución de cada algoritmo, detallado por los procesos que requieren la mayor cantidad de tiempo. Los tiempos se muestran de una manera en la que es más justo realizar comparaciones, esto es, promediando el tiempo que toma cada ejecución de los algoritmos, tipo búsqueda tabú y el recocido simulado.

El tiempo de optimización mostrado es lo que le toma al algoritmo en generar la estructura

Tabla 9: Comparación del tiempo de ejecución promedio de los algoritmos de optimización.

Algoritmo	Interacciones(ms)	DEE(ms)	Optimización(ms)
Búsqueda Tabú	6443	618	2348
Recocido Simulado	6501	631	4459
Búsqueda Local	6476	628	955

Los tiempos se muestran para los tres procesos principales de los algoritmos implementados, los cuales son: el cálculo de todas las interacciones, la ejecución del algoritmo de DEE y por último el algoritmo de optimización. Los resultados que se muestran se obtuvieron promediando los resultados de los tres conjuntos de pruebas. Los tiempos se muestran en milisegundos.

predicha empezando con la estructura inicial. Este proceso de optimización es el único que se repite 30 veces para el caso de los algoritmos de búsqueda tabú y recocido simulado, es decir, primero se realizan los cálculos de interacción y el algoritmo DEE una sola vez, seguido de 30 repeticiones del proceso de optimización. Es por este motivo que los tiempos de ejecución promedio de la Tabla 9 no son el resultado de promediar los tiempos totales de las tablas 5, 6 y 7, y dividir el resultado entre 30. Los tiempos promedio para procesos como el cálculo de interacciones y el algoritmo de DEE son similares entre los tres algoritmos implementados, con una diferencia de unos pocos milisegundos, lo cual es de esperarse ya que estos procesos son los mismos para los tres tipos de algoritmo de optimización.

En el proceso que sí hubo una diferencia de tiempo considerable es en la optimización. El algoritmo de búsqueda local fue el que presentó la rutina de optimización más rápida entre los tres algoritmos implementados, siendo aproximadamente 2.5 veces más rápida que la rutina de optimización del algoritmo de búsqueda tabú, y aproximadamente 4.6 veces más rápida que la de recocido simulado. La optimización que realiza el algoritmo de búsqueda tabú es casi el doble de rápida que la del recocido simulado. Esta diferencia de tiempo se debe principalmente a que el algoritmo de recocido simulado realiza el doble de iteraciones que el de búsqueda tabú.

4.3.3. Comparación de las variantes con y sin hibridación

Para comparar qué tan efectivo resultó el esquema de hibridación propuesto (Sección 3.4), se tomó el conjunto de pruebas de 65 proteínas y se realizaron pruebas del algoritmo de búsqueda tabú, pero esta vez sin ejecutar el algoritmo voraz para reducir el número de

colisiones, tanto al llegar a la mitad del número de iteraciones como al finalizar la ejecución de la búsqueda tabú (TS). La TS se comparó con tres maneras diferentes de realizar la hibridación de la búsqueda tabú con el algoritmo voraz. TSH-1 representa el algoritmo de búsqueda tabú ejecutando el algoritmo voraz cuando se llega a la mitad del número máximo de iteraciones. En TSH-2 se ejecuta el algoritmo voraz sólo al terminar la búsqueda tabú, mientras que en TSH-3 la búsqueda voraz se ejecuta tanto al llegar a la mitad de las iteraciones como al final de la búsqueda tabú. El algoritmo DEE se ejecutó en todos los casos ya que se utiliza como un pre-procesamiento para el algoritmo de optimización.

En la Tabla 10 se muestran los resultados obtenidos. Como se puede observar, tanto la precisión como el RMSD se mantuvieron en valores semejantes para los cuatro casos. Sin embargo, TS, TSH-1 y TSH-2, tuvieron valores más cercanos entre ellos, mientras que TSH-3 se desempeñó ligeramente mejor.

Podemos agrupar los distintos esquemas en dos grupos, grupo 1: TS y TSH-1; grupo 2: TSH-2 y TSH-3. Esto debido a que los algoritmos incluidos en cada grupo se comportaron de manera similar, esta distinción de grupos debe su desempeño principalmente al hecho de si ejecutó el algoritmo voraz al terminar la búsqueda tabú o no.

En cuanto al número de colisiones y la energía, sí hubo una diferencia significativa, en donde el grupo 2 tuvo aproximadamente un 74 % menos de colisiones, pero un 8 % más de energía, estos resultados son congruentes con los presentados en la Sección 4.3.1. En cuanto a la diferencia en costo computacional, en la Sección 3.1.7 se mencionó que el algoritmo voraz para colisiones es de tiempo $O(np)$, el cual se vio reflejado con una diferencia de tiempo promedio de 171.8 milisegundos.

El propósito por el que se implementó un esquema de hibridación es el de ayudar al algoritmo de optimización a salir de un posible óptimo local, cambiando la función objetivo después de algunas iteraciones, para así llegar a una estructura intermedia que no hubiera sido visitada manteniendo la misma función objetivo. Recordando que el proceso de hibridación consiste en realizar primero la optimización basada en la función de energía, seguida de una optimización basada en colisiones, y después volviendo a optimizar usando la función de energía. Por último se le aplica nuevamente el algoritmo voraz para colisiones a la estructura resultante.

En el Apéndice A.3, se presenta una comparación de la calidad obtenida por el algoritmo

Tabla 10: Comparación entre búsqueda tabú híbrida contra no híbrida.

Algoritmo	$\chi_1(\%)$	$\chi_{1+2}(\%)$	No. Colisiones	RMSD(Å)	$T_{optProm}$(ms)	E_{prom}
TS	82.26	62.63	78	0.8109	1477.4	3186.70
TSH-1	82.25	62.45	75	0.8112	1703.5	3184.15
TSH-2	82.28	62.58	20	0.8095	1686.5	3462.58
TSH-3	82.58	62.79	20	0.8054	1821.2	3463.14

TS representa el algoritmo de búsqueda tabú sin el esquema de hibridación. En cuanto a la hibridación se hizo la comparación con los tres esquemas posibles. TSH-1 representa el algoritmo de búsqueda tabú ejecutando el algoritmo voraz para reducir el número colisiones cuando se llega a la mitad del número máximo de iteraciones, en TSH-2 se ejecuta el algoritmo voraz al terminar la búsqueda tabú, mientras que en TSH-3 se ejecuta tanto al llegar a la mitad de las iteraciones como al final de la búsqueda tabú. $T_{optProm}$ representa el tiempo promedio que tardó cada método al ejecutar el algoritmo de optimización, es decir, sin contar el tiempo del algoritmo DEE o algún otro proceso. E_{prom} es la energía promedio de las estructuras predichas.

de recocido simulado al ser hibridado de diferentes maneras con el algoritmo de búsqueda local y el algoritmo voraz. Los resultados muestran el aporte de cada componente de la hibridación.

4.4. Comparación de la calidad con los métodos del estado del arte

En esta sección se discute una comparación de la precisión obtenida por los algoritmos implementados en este trabajo con algunos de los algoritmos del estado del arte.

En la Sección 2.5 se mencionaron algunos de los principales algoritmos del estado del arte para el PSCPP, por lo tanto, en esta sección se lleva a cabo una comparación entre dichos algoritmos y los implementados en este trabajo.

Cada uno de los métodos del estado del arte reporta la precisión que logró obtener utilizando sus propios conjuntos de pruebas. En la Tabla 11 se muestran los valores de precisión absoluta para $\chi_1(\%)$ y $\chi_{1+2}(\%)$ que obtuvo cada uno de los métodos, junto con el tamaño del conjunto de pruebas utilizado por cada uno de los algoritmos.

El algoritmo que reportó la mayor precisión para $\chi_1(\%)$ fue el OPUS-Rota, con 89.0 %. Sin embargo, fue el algoritmo que utilizó el conjunto de pruebas más pequeño, con únicamente 65 proteínas. El método RASP fue el que utilizó el conjunto de pruebas más extenso, con 2412 proteínas en total, y obteniendo una precisión para $\chi_1(\%)$ de 86.0 %.

Actualmente no existe conjunto de pruebas estándar para evaluar el desempeño de los

Tabla 11: Precisión reportada por los métodos SCWRL4, OPUS-Rota, CIS-RR y RASP para el PSCPP.

Algoritmo	$\chi_1(\%)$	$\chi_{1+2}(\%)$	Tamaño del conjunto de prueba
SCWRL4 (Krivov *et al.*, 2009)	86.1	74.8	379
OPUS-Rota (Lu y Dousis, 2008)	89.0	79.1	65
CIS-RR (Cao *et al.*, 2011)	85.7	75.9	180
RASP (Miao *et al.*, 2011)	86.0	75.9	2412

Precisión absoluta de cada uno de los algoritmos mencionados en la Sección 2.5. La precisión absoluta que se muestra es la que cada método reporta en su respectivo artículo, además de que se incluye el tamaño del conjunto de proteínas que cada método utilizó.

algoritmos para el PSCPP. Es por esto que cada algoritmo presenta su propio conjunto de pruebas. Sin embargo, los autores del RASP (Miao *et al.*, 2011) utilizaron el conjunto de pruebas propuesto por el SCWRL4 (un total de 379 proteínas) y probaron el desempeño de algunos de los algoritmos más populares en el estado del arte para el PSCPP. Los resultados obtenidos en esta comparación se muestran en la Tabla 12.

Tabla 12: Comparación de la precisión de los algoritmos en el conjunto de pruebas del SCWRL4, según Miao *et al.* (2011).

Algoritmo	$\chi_1(\%)$	$\chi_{1+2}(\%)$	Colisiones(Prom)	RMSD(Å)
SCWRL4	85.03	75.44	411(1.08)	1.46
OPUS-Rota	85.03	71.74	623(1.64)	1.43
CIS-RR	84.88	74.88	59(0.15)	1.47
RASP	85.10	74.71	47(0.12)	1.47

Comparación de la calidad de las soluciones de algunos de los algoritmos del estado del arte en el conjunto de pruebas del SCWRL4 (379 proteínas). El valor entre paréntesis en la columna de colisiones representa el número de colisiones promedio por proteína. Resultados obtenidos por Miao *et al.* (2011).

Comparando las precisiones obtenidas en este conjunto de pruebas, el algoritmo que obtuvo la mejor precisión en $\chi_1(\%)$ fue el RASP, mientras que para $\chi_{1+2}(\%)$ el SCWRL4 fue el método que obtuvo la precisión más alta. En cuanto a colisiones, tanto el SCWRL4 como el OPUS-Rota fueron los que peor se desempeñaron, generando más de 6 veces el número de colisiones del CIS-RR y más de 8 veces el número de colisiones del RASP. Sin embargo, el número promedio de colisiones por proteína sigue siendo bajo. Cabe mencionar que la

estrategia de los métodos CIS-RR y RASP se enfoca en reducir el número de colisiones para generar una estructura de buena calidad.

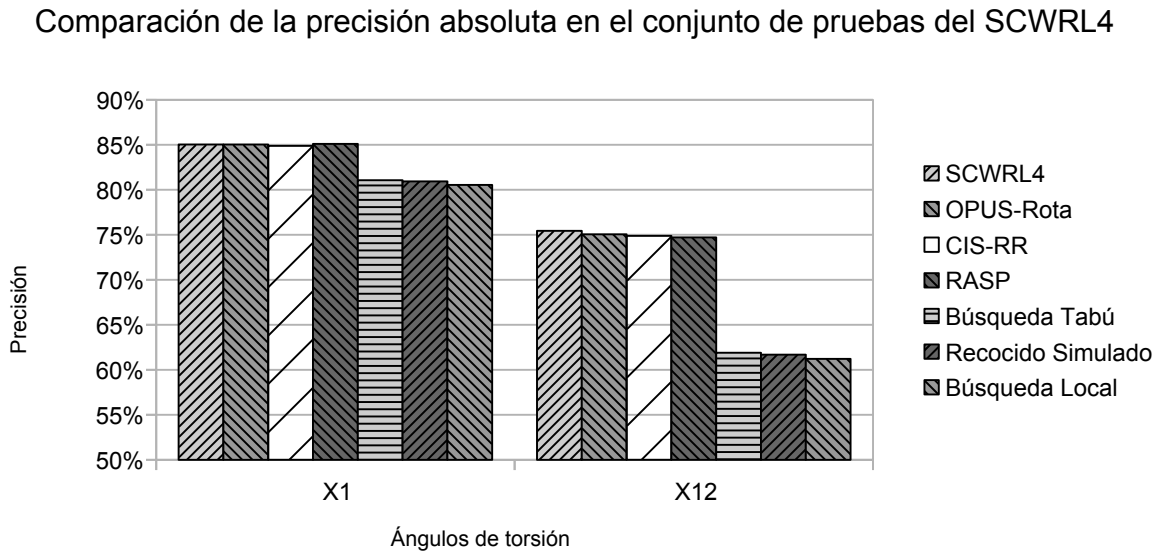

Figura 8: Comparación de la precisión entre los algoritmos del estado del arte y los implementados en este trabajo para el conjunto de pruebas del SCWRL4.

En la Figura 8 se muestra una gráfica comparativa de la precisión absoluta obtenida por los métodos del estado del arte y los implementados en este trabajo. El conjunto de pruebas que se utilizó para los algoritmos de búsqueda tabú, recocido simulado y búsqueda local, no es el mismo que para el resto de los métodos, sino que es el segundo conjunto de pruebas de los mencionados en la Sección 4.1, el cual no toma en cuenta seis de las 379 proteínas del conjunto de pruebas del SCWRL4. El motivo por el cual se excluyen estas seis proteínas (1byi, 1hcl, 1koe, 1mml, 1nar y 1vjs) es porque las mismas ya se encuentran en el conjunto de 65 proteínas.

La precisión de los algoritmos implementados en este trabajo resultó ser inferior a la de los métodos del estado del arte; sin embargo, obtuvo buenos resultados en términos de colisiones, superando por un amplio margen al SCWRL4 y al OPUS-Rota, además de obtener un número de colisiones cercano a las que obtuvieron el CIS-RR y el RASP. En la Figura 9 se muestra una gráfica comparativa del número de colisiones obtenidas por cada algoritmo.

En cuanto a RMSD se refiere, se puede observar de las tablas 6 y 12 que los tres métodos implementados en este trabajo mejoraron el RMSD de los algoritmos del estado del arte;

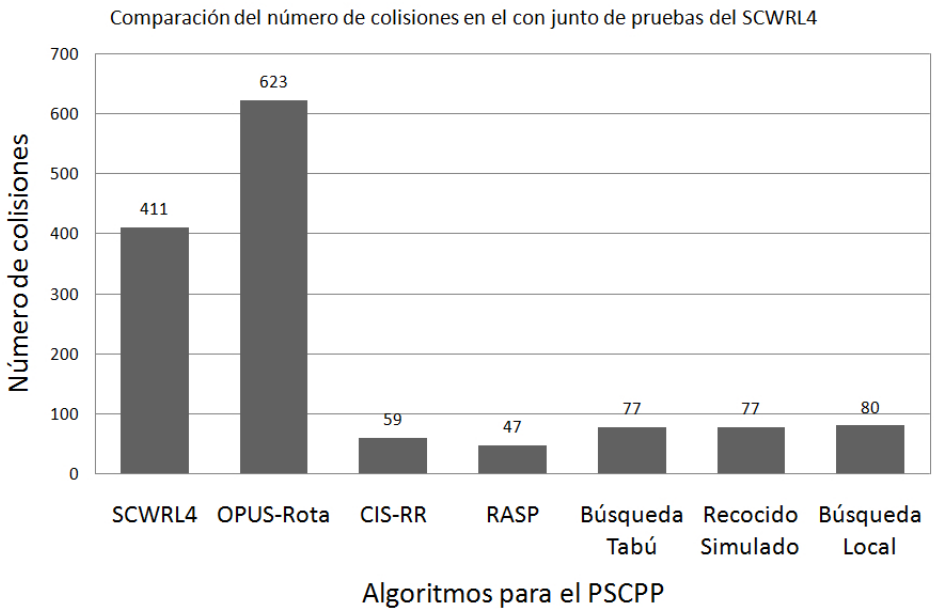

Figura 9: Comparación del número de colisiones entre los algoritmos del estado del arte y los implementados en este trabajo para el conjunto de pruebas de 373 proteínas.

sin embargo, debido a que hay una diferencia de aproximadamente 0.6Å, es probable que los autores del RASP hayan calculado el RMSD únicamente para los átomos de las cadenas laterales sin considerar los átomos de la columna vertebral, aunque en el artículo no se hace mención alguna al respecto.

4.5. Discusión

En esta sección se discutirán sobre las principales diferencias entre los métodos del estado del arte para el PSCPP y los algoritmos implementados en este trabajo, y qué factores se considera que afectaron el desempeño de los algoritmos implementados, impidiéndoles alcanzar la precisión de algoritmos como el SCWRL4 y RASP. También se incluye un análisis de las funciones de energía y los valores que se obtienen al calcular la energía de una estructura proveniente de un archivo con formato pdb.

4.5.1. Diferencias entre los algoritmos propuestos y los algoritmos del estado del arte

Antes de decidir implementar la función de energía mencionada en la Sección 3.5.1, se implementaron las funciones de energía de van der Waals utilizadas por los algoritmos men-

cionados en la Sección 2.5. El motivo por el cual se implementó únicamente el potencial de van der Waals, y no el resto de los términos de las funciones de energía, es que dicho potencial es el mayor contribuyente en calidad de las estructuras predichas (Lu y Dousis, 2008). Sin embargo, a pesar de que las funciones de energía que se implementaron eran las mismas, no se pudieron utilizar los mismos parámetros que en los algoritmos originales, los cuales se obtuvieron de las versiones más recientes del CHARMM (Brooks *et al.*, 1983). Esto debido a que el acceso a CHARMM tiene un costo, incluso para utilizarlo con fines académicos. Por este motivo, para los valores de e_{ij} de las funciones de energía se utilizaron los parámetros obtenidos de AMBER (Weiner *et al.*, 1984).

La primer función de energía que se implementó fue la del algoritmo CIS-RR (Sección 2.5.3). Al optimizar la función de energía para el potencial de Van der Waals del algoritmo CIS-RR, se encontró que la misma no se desempeñó de la manera esperada; ya que a pesar de que la energía de la estructura disminuía, la precisión absoluta, que debería aumentar, también disminuía. De igual manera se probó con las funciones de energía del resto de los algoritmos (SCWRL4, OPUS-Rota y RASP); obteniéndose resultados similares.

En la Tabla 13 se muestra el desempeño que tuvieron las funciones de energía para el potencial de Van der Waals de cada uno de los algoritmos mencionados en la Sección 2.5. Estos resultados se obtuvieron de ejecutar el algoritmo de búsqueda local como se menciona en la Sección 3.3, incluyendo el algoritmo de DEE, pero sin ejecutar el algoritmo voraz para colisiones. Se excluyó el algoritmo voraz para colisiones debido a que se desea comprobar qué tan efectivas son las funciones de energía para modelar el empaquetamiento de la cadena lateral; y como se mencionó en la Sección 4.3.1, el algoritmo voraz para reducir las colisiones puede incrementar los valores de energía de las estructuras. La estructura inicial se generó de la misma manera que para los algoritmos propuestos en el Capítulo 3: una vez ejecutado el algoritmo DEE, se selecciona para cada aminoácido el rotámero con la mayor probabilidad entre los rotámeros que no fueron eliminados.

Como se puede observar en los resultados de la Tabla 13, para las cuatro funciones objetivo se dio el caso de que al optimizar la estructura reduciendo la energía, sucede que a pesar de que la misma disminuye, la precisión, contrario a lo esperado, también disminuye. Esto se puede deber a una dependencia entre los parámetros utilizados y la función de energía, así como también una posible dependencia entre la función de energía y la biblioteca de rotámeros.

Tabla 13: Desempeño de las funciones de energía de los algoritmos del estado del arte.

	Estructura inicial			Estructura predicha		
Función de energía	$\chi_1(\%)$	$\chi_{1+2}(\%)$	Energía	$\chi_1(\%)$	$\chi_{1+2}(\%)$	Energía
SCWRL4	67.24	50.74	368.98	54.41	33.21	-11.89
OPUS-Rota	67.63	50.10	572.80	53.66	29.70	-141.42
CIS-RR	71.38	54.01	-3.50	69.45	44.43	-65.96
RASP	68.72	51.41	20.66	55.45	31.15	-125.67

Resultados promedio obtenidos al realizar la predicción de cadenas laterales para el conjunto de 65 proteínas utilizando el algoritmo de búsqueda local y las funciones de energía de Van der Waals de los algoritmos del estado del arte.

En cuanto a la función de energía que se implementó para los algoritmos propuestos, un factor que probablemente ayudaría a mejorar su desempeño sería considerar más términos en la misma; por ejemplo, un potencial para los puentes de hidrógeno y el potencial electrostático. El algoritmo MESHI (Kalisman *et al.*, 2005), del cual se obtuvieron los parámetros y las funciones de energía implementadas, sí cuenta con un potencial electrostático el cual se implementó; sin embargo, en pruebas preliminares no demostró mejorar la calidad de las soluciones predichas. Esto se puede deber a que como se mencionó en la Sección 3.5.1, el término Excluded Volume es un potencial repulsivo únicamente, el cual puede hacer despreciable la fuerza de atracción que el potencial electrostático podría generar. Por este motivo, podría ser posible obtener mejores resultados si se combina un potencial electrostático con un potencial de van der Waals, ya que de esta manera ambos términos considerarían fuerzas tanto de atracción como de repulsión entre átomos.

En la Sección 4.4 se realizó una comparación entre la calidad de las estructuras predichas por los algoritmos implementados en este trabajo, con la de algunos de los algoritmos del estado del arte. Cabe mencionar que aunque la precisión para $\chi_1(\%)$ es inferior, ésta es aproximadamente 3 % menor, mientras que para $\chi_{1+2}(\%)$ es alrededor de 13 % menor. Con esto se observa que la diferencia de precisión entre $\chi_1(\%)$ y $\chi_{1+2}(\%)$, para los métodos del estado del arte, es de 10 %, mientras que para los algoritmos implementados es de 20 %.

Con el objetivo de ver el incremento de precisión que introducen los algoritmos con respecto a las estructuras iniciales se presentan datos adicionales en el Apéndice A.4. En este se observa también la capacidad de los algoritmos para optimizar la función de energía.

4.5.2. Evaluación de algunos términos de la función de energía en estructuras del *Protein Data Bank*(PDB)

Para el PSCPP, el caso ideal sería tener una función de energía que modele las interacciones entre átomos de tal manera que la estructura de una proteína con la menor energía sea la que tiene los rotámeros que mejor se adaptan a la estructura de la proteína original. Sin embargo, esto resulta difícil cuando las funciones de energía son únicamente una aproximación de la interacción entre átomos. Por este motivo se decidió comprobar si la estructura de la proteína del archivo con formato .pdb es un óptimo local para alguna de las funciones de energía implementadas.

En el PSCPP se utiliza una biblioteca de rotámeros para seleccionar las conformaciones de la cadena lateral para los aminoácidos, por este motivo no es posible crear estructuras idénticas para las proteínas del PDB. Considerando lo anterior, se creó un tipo de estructura denominada estructura *Best*, la cual se construye seleccionando para cada aminoácido el rotámero que mejor se adapta a los ángulos de torsión de la proteína en el archivo pdb, y que por lo tanto, es la estructura a la que se desearía llegar al utilizar algún algoritmo para el PSCPP (En la Tabla 14 se observa una precisión de casi 100 % para χ_1(%) en la estructura *Best*).

El proceso que se planteó para comprobar si alguna de las funciones de energía considera a la estructura del archivo .pdb como un óptimo local consiste en tomar la estructura *Best* y ejecutar un algoritmo de optimización, con lo cual se espera que la estructura final sea la misma estructura *Best*, para así considerar a *Best* como un óptimo local para dicha función de energía. Este procedimiento se propuso originalmente en el trabajo de Corona de la Fuente (2010) y fue mejorado en el de Lezcano (2012).

En la Tabla 14 se muestran los resultados obtenidos al ejecutar el algoritmo de búsqueda local en la estructura *Best* con cada una de las funciones de energía de Van der Waals de los algoritmos SCWRL4, OPUS-Rota, CIS-RR y RASP, así como también la función de energía implementada en este trabajo. Se seleccionó el algoritmo de búsqueda local ya que este no incluye ningún factor aleatorio al seleccionar qué aminoácido será modificado, ya que se quieren comparar las funciones de energía bajo las mismas condiciones sin dejar que el factor aleatorio influya en los resultados.

Se puede observar que ninguna de las funciones de energía analizadas (correspondientes al

Tabla 14: Comparación de la energía de la estructura *Best* antes y después de la optimización.

	Estructura *Best*			Estructura optimizada		
Función de energía	$\chi_1(\%)$	$\chi_{1+2}(\%)$	Energía	$\chi_1(\%)$	$\chi_{1+2}(\%)$	Energía
EV+R	99.55	93.04	6717.3	90.12	76.22	3685.8
SCWRL4	99.55	93.04	54.54	68.78	53.65	-13.35
OPUS-Rota	99.55	93.04	-45.96	63.58	45.58	-141.89
CIS-RR	99.55	93.04	-49.58	80.62	66.35	-62.42
RASP	99.55	93.04	-69.04	65.59	47.45	-119.59

Resultados promedio obtenidos al realizar la predicción de cadenas laterales para el conjunto de 65 proteínas utilizando el algoritmo de búsqueda local y las funciones de energía de Van der Waals de los algoritmos del estado del arte sobre la estructura *Best*. EV+R es la función de energía implementada en este trabajo, mencionada en la Sección 3.5.1.

término de van der Waals) considera la estructura *Best* como un óptimo local, ya que en todos los casos se encontró otra estructura con un menor valor de energía. Algo que es importante mencionar, es el hecho de que no solamente se encontraron estructuras con menor valor de energía, si no que estas estructuras tienen una precisión que va desde 9 % hasta más de 30 % de decremento para $\chi_1(\%)$; con lo cual disminuye nuestra confianza en la habilidad de las funciones de energía implementadas para modelar de manera correcta el empaquetamiento de la cadena lateral.

4.5.3. Mejor estructura posible después del algoritmo DEE

En la Sección 4.5.2 se presentó la calidad de las mejores estructuras que se pueden formar utilizando todos los rotámeros de la biblioteca (estructura *Best*), en esta sección se analizan las mejores estructuras que se pueden formar utilizando únicamente los rotámeros que no fueron eliminados por el algoritmo DEE, las cuales se denominarán como estructuras $Best_{DEE}$. Este análisis se hace con el propósito de comparar qué tanta diferencia hay entre la calidad de las mejores estructuras que se pueden formar después de la ejecución del algoritmo DEE, y las estructuras predichas. Esto debido a que como ya se mencionó, los términos de energía implementados no modelan de manera adecuada el empaquetamiento de la cadena lateral, y debido a que el algoritmo DEE se basa en la energía para descartar rotámeros, existe la posibilidad de que se hayan descartado los rotámeros que seleccionó la estructura *Best* para obtener una precisión del 99.5 % para $\chi_1(\%)$, o rotámeros con los cuales se pueda obtener

una precisión similar.

Para comprobar lo antes mencionado se creó la estructura $Best_{DEE}$ utilizando los rotámeros que no fueron eliminados, esto se hizo para las proteínas de los tres casos de prueba. Una vez creada la estructura $Best_{DEE}$ se ejecutó el algoritmo de búsqueda local en dicha estructura, esto con el propósito de verificar si la estructura $Best_{DEE}$ es un óptimo local para la función de energía implementada o no.

Tabla 15: Comparación de calidad de la estructura *Best* después del algoritmo DEE.

	Estructura $Best_{DEE}$				Estructura $Best_{DEE}$ + BL			
Caso de pruebas	$\chi_1(\%)$	$\chi_{1+2}(\%)$	Col.	Energía	$\chi_1(\%)$	$\chi_{1+2}(\%)$	Col.	Energía
65 proteínas	91.02	74.69	73	4038.82	82.56	62.78	76	3176.83
373 proteínas	90.22	74.77	257	3892.88	81.29	62.23	289	2982.50
723 proteínas	91.10	75.83	574	4267.10	82.32	63.38	618	3074.80

Resultados promedio obtenidos al realizar la predicción de cadenas laterales para los tres casos de pruebas, utilizando el algoritmo de búsqueda local (BL) sobre la estructura $Best_{DEE}$.

En la Tabla 15 se muestran los resultados obtenidos. La precisión promedio de la estructura $Best_{DEE}$ para los tres casos de prueba es de aproximadamente 91 % para $\chi_1(\%)$ y 75 % para $\chi_{1+2}(\%)$. La precisión de las estructuras $Best_{DEE}$ es aproximadamente un 8.5 % menor que la de las estructuras *Best* para $\chi_1(\%)$, sin embargo, su precisión es aproximadamente 6 % mayor que la de los algoritmos del estado del arte.

Después de la ejecución del algoritmo de búsqueda local sobre la estructura $Best_{DEE}$ se puede observar que la precisión disminuyó aproximadamente un 9 % para $\chi_1(\%)$ y un 12 % para χ_{1+2}. Sin embargo, la energía disminuyó alrededor de 25 % en promedio para los tres casos de prueba, lo cual reitera el hecho de que el minimizar el valor de los términos de la función de energía implementados, no está guiando la búsqueda hacia estructuras con una mejor precisión.

Capítulo 5

Conclusiones

En este capítulo se presentan las conclusiones a las que se llegó en este trabajo de tesis, así como algunas perspectivas de investigación sobre el problema abordado.

5.1. Sumario

El problema del empaquetamiento de la cadena lateral en proteínas (PSCPP) se modela como un problema de optimización combinatoria, mediante el uso de una función de energía para aproximar las interacciones entre los átomos de la proteína, una biblioteca de rotámeros para discretizar las conformaciones de la cadena lateral de los aminoácidos, y un método de búsqueda para encontrar el mínimo de la función de energía propuesta.

El cálculo de la energía de la proteína puede llegar a ser $O(n^2)$ si se hacen comparaciones entre todas las parejas de átomos de la proteína. Para reducir el número de operaciones se implementó una rejilla de átomos, la cual permite obtener de manera eficiente, para cada aminoácido i, una lista de sus aminoácidos cercanos. Las listas de aminoácidos cercanos generadas permiten que el tiempo para calcular la energía de la proteína se reduzca a $O(n)$. También se implementó una tabla hash para almacenar los valores de energía entre dos aminoácidos en un estado rotamérico específico, de esta manera se puede recuperar la energía en tiempo $O(1)$ sin necesidad de recalcularla cada vez que sea requerida.

Para abordar el PSCPP, se implementaron tres heurísticas: búsqueda tabú, recocido simulado y búsqueda local, así como un algoritmo de eliminación de callejones sin salida (Desmet *et al.*, 1992) como pre-procesamiento. Este último elimina rotámeros que no pueden pertenecer a la conformación de energía mínima de la proteína, reduciendo así el espacio de soluciones.

Se propuso un esquema de hibridación para las heurísticas implementadas, el cual consiste en cambiar la función objetivo de las heurísticas una vez que se llega a la mitad del número total de iteraciones; para esto se ejecuta un algoritmo voraz para reducir el número total de colisiones y posteriormente regresar al algoritmo de optimización para que este concluya con el proceso de empaquetamiento de la cadena lateral.

Para comprobar la calidad de los algoritmos implementados se utilizaron tres casos de

prueba, de 65, 373 y 723 proteínas, respectivamente. Los resultados obtenidos se compararon con algunos de los algoritmos del estado del arte.

A continuación se enuncian las conclusiones a las que se llegó con base en los experimentos realizados en este trabajo de investigación.

5.2. Conclusiones

El uso de estructuras de datos como la rejilla de átomos y la tabla hash de interacciones lograron disminuir el tiempo de ejecución de los algoritmos implementados.

De los tres algoritmos implementados, el de búsqueda tabú fue el que dio mejores resultados para los tres conjuntos de prueba. Sin embargo, la desviación estándar de la precisión de los tres algoritmos es de 0.25 para $\chi_1(\%)$ y de 0.32 para $\chi_{1+2}(\%)$, lo cual indica que los tres algoritmos implementados tuvieron un desempeño similar.

Se comprobó que el algoritmo de eliminación de callejones sin salida (Desmet *et al.*, 1992) da buenos resultados eliminando rotámeros que no pueden pertenecer a la conformación de energía mínima de la proteína, reduciendo así el espacio de soluciones y mejorando la calidad de la estructura inicial.

Los algoritmos implementados alcanzaron aproximadamente un 82 % de precisión promedio en $\chi_1(\%)$ y 62 % para $\chi_{1+2}(\%)$, para los tres casos de pruebas utilizados. Esta precisión no logró superar la de los algoritmos del estado del arte.

El esquema de hibridación propuesto no demostró una mejora significativa en cuanto a precisión se refiere; sin embargo, logró obtener estructuras con un número menor de colisiones. Aunque el número de colisiones no es una medida de calidad estándar para las estructuras predichas en el PSCPP, un valor bajo en este criterio es una característica deseable. El esquema de hibridación logró generar estructuras con un 72 % menos de colisiones que el esquema sin hibridación, lo cual se ve reflejado en aproximadamente 200ms de tiempo computacional adicional por proteína. Con base en esto se puede concluir que el desempeño del esquema de hibridación fue satisfactorio.

Con base en el análisis de componentes de las funciones de energía implementadas (Sección 4.5.2), se encontró que las estructuras de proteínas del PDB no son consideradas como un óptimo local para las funciones de energía utilizadas. Este resultado implica que ninguna heurística que optimice sólo los componentes analizados podrá llegar a la estructura del

PDB, ya que las heurísticas terminan su ejecución en un óptimo local con la hipótesis de que éste será una configuración de energía mínima en la proteína.

5.3. Propuestas de trabajo futuro

5.3.1. Función de energía

Un factor importante en la calidad de las soluciones que es necesario estudiar a profundidad es la función de energía utilizada, motivo por el cual, antes de intentar proponer alguna nueva heurística para el PSCPP, sería interesante realizar un estudio de distintas funciones de energía y bibliotecas de rotámeros, en búsqueda de alguna combinación que logre considerar las estructuras del *Protein Data Bank* (PDB) como un óptimo local. De esta manera podría ser posible que el algoritmo de optimización llegue a dicha estructura.

Con la reducción del tiempo para el cálculo de la energía, de $O(n^2)$ a $O(n)$, sería interesante implementar una heurística basada en población y comparar su desempeño.

5.3.2. Calidad de las soluciones

Los algoritmos del estado del arte promedian alrededor de 75 % de precisión para $\chi_{1+2}(\%)$, este valor representa una diferencia de 10 % con respecto a la precisión que obtienen en $\chi_1(\%)$. Sin embargo, para los algoritmos implementados en este trabajo la precisión para $\chi_{1+2}(\%)$ disminuye 20 %, por lo que sería interesante investigar qué es lo que causa este comportamiento.

5.3.3. Algoritmo implementado

El código implementado en este trabajo de tesis cuenta con tres heurísticas y cinco funciones de energía, por lo que al diseñar el código; éste se separó en módulos, los cuales se adaptaron para que simplemente al cambiar una línea de código se pueda seleccionar con cuál heurística se va a realizar la optimización. Con la misma facilidad se puede cambiar tanto la función de energía como la biblioteca de rotámeros.

Este código se puede adaptar para implementar interfaces, las cuales permitirían estandarizar y facilitar la implementación de nuevas heurísticas y funciones de energía, permitiendo así la creación de una plataforma con la cual se podría comparar el desempeño de distintas

combinaciones de heurísticas, funciones de energía y bibliotecas de rotámeros, de manera sencilla.

De igual manera se podría crear una interfaz que permita facilitar la implementación de nuevos esquemas de hibridación, permitiendo ejecutar alguna rutina cuando se llegue a cierto número de iteraciones; o incluso al terminarse de ejecutar la heurística principal se podría ejecutar una heurística diferente, para ver si ésta puede mejorar la solución que se tenga hasta el momento.

En el algoritmo voraz, para reducir el número de colisiones, se recorre la proteína de manera secuencial. Sería interesante ver qué sucede al cambiar el orden en el que se recorren los aminoácidos de la proteína.

Referencias blibiográficas

Aarts, E. y Lenstra, J. K. (1997). *Local Search in Combinatorial Optimization*. Princeton University Press. p. 512.

Akutsu, T. (1997). NP-hardness results for protein side-chain packing. *Genome informatics*, **8**: 180–186.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., y Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, **215**(3): 403–410.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., y Bourne, P. E. (2000). The Protein Data Bank. *Nucleic acids research*, **28**(1): 235–242.

Bourne, P. E. y Weissig, H. (2003). *Structural Bioinformatics*. Methods of Biochemical Analysis. Wiley. p. 681.

Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S., y Karplus, M. (1983). CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry*, **4**(2): 187–217.

Bryant, S. H. y Altschul, S. F. (1995). Statistics of sequence-structure threading. *Current Opinion in Structural Biology*, **5**(2): 236–244.

Canutescu, A. A., Shelenkov, A. A., y Dunbrack, R. L. (2003). A graph-theory algorithm for rapid protein side-chain prediction. *Protein science*, **12**(9): 2001–2014.

Cao, Y., Song, L., Miao, Z., Hu, Y., Tian, L., y Jiang, T. (2011). Improved side-chain modeling by coupling clash-detection guided iterative search with rotamer relaxation. *Bioinformatics*, **27**(6): 785–790.

Chan, H. S. y Dill, K. A. (1993). The protein folding problem. *Phisics Today*, **37**: 24–32.

Chothia, C. (1976). The nature of the accessible and buried surfaces in proteins. *Journal of Molecular Biology*, **105**(1): 1–12.

Chothia, C. y Lesk, A. M. (1986). The relation between the divergence of sequence and structure in proteins. *The EMBO journal*, **5**(4): 823–826.

Corona de la Fuente, R. I. (2010). *Análisis Comparativo de dos Heurísticas para el Problema de Empaquetamiento de la Cadena Lateral en Proteínas*. Tesis de maestría, CICESE.

Desmet, J., De Maeyer, M., Hazes, B., y Lasters, I. (1992). The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*, **356**: 539–542.

Dunbrack, R. L. y Cohen, F. E. (1997). Bayesian statistical analysis of protein side-chain rotamer preferences. *Protein science : a publication of the Protein Society*, **6**(8): 1661–1681.

Dunbrack, R. L. y Karplus, M. (1993). Backbone-dependent rotamer library for proteins. Application to side-chain prediction. *Journal of molecular biology*, **230**(2): 543–574.

Eisenberg, D. y McLachlan, A. D. (1986). Solvation energy in protein folding and binding. *Nature*, **319**(6050): 199–203.

Epstein, C. J., Goldberger, R. F., y Anfinsen, C. B. (1963). The Genetic Control of Tertiary Protein Structure: Studies With Model Systems. *Cold Spring Harbor Symposia on Quantitative Biology*, **28**: 439–449.

Fletcher, R. y Reeves, C. M. (1964). Function minimization by conjugate gradients. *Comput J.*, **7**: 149–154.

Glover, F. (1989). Tabu Search–Part I. *INFORMS Journal on Computing*, **1**(3): 190–206.

Glover, F. (1990). Tabu search: A tutorial. *Interfaces*, **20**: 74–94.

Goldstein, R. F. (1994). Efficient rotamer elimination applied to protein side-chains and related spin glasses. *Biophisics*, **66**: 1335–1340.

Gordon, D. B. y Mayo, S. L. (1999). Branch-and-terminate: a combinatorial optimization algorithm for protein design. *Structure*, **7**: 1089–1098.

Gordon, D. B., Marshall, S. A., y Mayo, S. L. (1999). Energy functions for protein design. *Current opinion in structural biology*, **9**(4): 509–513.

Gu, J. y Bourne, P. E. (2009). *Structural Bioinformatics*. Wiley. p. 1035.

Jain, T., Cerutti, D. S., y McCammon, J. A. (2006). Configurational-bias sampling technique for predicting side-chain conformations in proteins. *Protein science : a publication of the Protein Society*, **15**(9): 2029–2039.

Kalisman, N., Levi, A., Maximova, T., Reshef, D., Zafriri-Lynn, S., Gleyzer, Y., y Keasar, C. (2005). MESHI: a new library of Java classes for molecular modeling. *Bioinformatics*, **21**(20): 3931–3932.

Kanehisa, M. y Bork, P. (2003). Bioinformatics in the post-sequence era. *Nature genetics*, **33 Suppl**: 305–310.

Kirkpatrick, S., Gelatt, C. D., y Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, **220**(4598): 671–680.

Kirkwood, J. G. (1935). Statistical Mechanics of Fluid Mixtures. *The Journal of Chemical Physics*, **3**(5): 300–313.

Klosowski, J., Held, M., Mitchell, J., Sowizral, H., y Zikan, K. (1998). Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, **4**(1): 21–36.

Krivov, G. G., Shapovalov, M. V., y Dunbrack, R. L. (2009). Improved prediction of protein side-chain conformations with SCWRL4. *Proteins*, **77**(4): 778–795.

Lennard-Jones, J. E. (1925). On The Forces Between Atoms And Ions. *Proc. R. Soc. Lond.*, **109**(752): 584–597.

Lezcano, C. G. (2012). *Problema de empaquetamiento de la cadena lateral de proteínas: Análisis de la calidad de soluciones de una biblioteca de rotámeros simple*. Tesis de maestría, Universidad Nacional de Asunción.

Liu, J. (2008). *Monte Carlo Strategies in Scientific Computing*. Springer. p. 343.

Looger, L. L. y Hellinga, H. W. (2001). Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: implications for protein design and structural genomics. *Journal of molecular biology*, **307**(1): 429–445.

Lu, M. y Dousis, A. D. (2008). OPUS-Rota : A fast and accurate method for side-chain modeling. *Protein Science*, **17**: 1576–1585.

Lu, M., Dousis, A. D., y Ma, J. (2008). OPUS-PSP: an orientation-dependent statistical all-atom potential derived from side-chain packing. *Journal of molecular biology*, **376**(1): 288–301.

Mendes, J., Baptista, A. M., Carrondo, M. A., y Soares, C. M. (1999). Improved modeling of side-chains in proteins with rotamer-based methods: a flexible rotamer model. *Proteins*, **37**(4): 530–543.

Miao, Z., Cao, Y., y Jiang, T. (2011). RASP: rapid modeling of protein side chain conformations. *Bioinformatics*, **27**(22): 3117–3122.

Newman, M. E. J. y Barkema, G. T. (1999). *Monte Carlo methods in statistical physics*. Universities Press. p. 475.

Osguthorpe, D. J. (2000). Ab initio protein folding. *Current Opinion in Structural Biology*, **10**(2): 146–152.

PDB (2014). RCSB Protein Data Bank, http://www.rcsb.org/pdb/home/home.do.

Pearson, W. R. (1990). Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods in Enzymology*, **183**: 63–98.

Peterson, R. W., Dutton, P. L., y Wand, A. J. (2004). Improved side-chain prediction accuracy using an ab initio potential energy function and a very large rotamer library. *Protein science : a publication of the Protein Society*, **13**(3): 735–751.

Reingold, E. M., urg Nievergelt, J., y Deo, N. (1977). *Combinatorial Algorithms: Theory and Practice*. Prentice Hall. p. 433.

Rodriguez, R., Chinea, G., Lopez, N., Pons, T., y Vriend, G. (1998). Homology modeling, model and software evaluation: three related resources. *Bioinformatics*, **14**(6): 523–528.

Rohl, C. A., Strauss, C. E. M., Misura, K. M. S., y Baker, D. (2004). Protein structure prediction using Rosetta. *Methods in enzymology*, **383**: 66–93.

Rost, B. (1999). Twilight zone of protein sequence alignments. *Protein Engineering Design and Selection*, **12**(2): 85–94.

Samudrala, R. y Moult, J. (1998). A graph-theoretic algorithm for competitive modeling of protein structure. *Molecular Biology*, **279**: 287–302.

Sander, C. y Schneider, R. (1991). Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins*, **9**(1): 56–68.

Shapovalov, M. V. y Dunbrack, R. L. (2011). A smoothed backbone-dependent rotamer library for proteins derived from adaptive kernel density estimates and regressions. *Structure*, **19**(6): 844–858.

Sharp, K. A., Nicholls, A., Friedman, R., y Honig, B. (1991). Extracting hydrophobic free energies from experimental data: relationship to protein folding and theoretical models. *Biochemistry*, **30**(40): 9686–9697.

Shindyalov, I. N. y Bourne, P. E. (1998). Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering Design and Selection*, **11**(9): 739–747.

Smyth, M. S. (2000). x Ray crystallography. *Molecular Pathology*, **53**(1): 8–14.

Subbotin, Y. N. (1967). Piecewise-polynomial (spline) interpolation. *Mathematical Notes of the Academy of Sciences of the USSR*, **1**(1): 41–45.

Tarjan, R. (1971). Depth-first search and linear graph algorithms. En: *12th Annual Symposium on Switching and Automata theory*, pp. 114–121.

Thompson, J. D., Higgins, D. G., y Gibson, T. J. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, **22**(22): 4673–4680.

UniProt (2014). UniProt, http://www.ebi.ac.uk/uniprot/TrEMBLstats.

Wang, G. y Dunbrack, R. L. (2003). PISCES: a protein sequence culling server. *Bioinformatics*, **19**(12): 1589–1591.

Weiner, S. J., Kollman, P. A., Case, D. A., Singh, U. C., Ghio, C., Alagona, G., Profeta, S., y Weiner, P. (1984). A new force field for molecular mechanical simulation of nucleic acids and proteins. *Journal of the American Chemical Society*, **106**(3): 765–784.

Winterton, R. H. S. (1970). Van der Waals forces. *Contemporary Physics*, **11**(6): 559–574.

Xu, J. (2005). *Rapid protein side-chain packing via tree decomposition*. Springer. pp. 423–439.

A.1. Tabla hash de interacciones

Como se mencionó en la Sección 3.1.3, para los algoritmos implementados se utiliza una tabla hash de cuatro niveles para almacenar todas las interacciones entre los rotámeros de la proteína.

La manera en la que se recupera un valor de interacción almacenado es utilizando cuatro índices i, r, j, y s, donde i y j representan la pareja de aminoácidos de la cual se quiere saber su energía de interacción, mientras que r y s representan sus respectivos rotámeros.

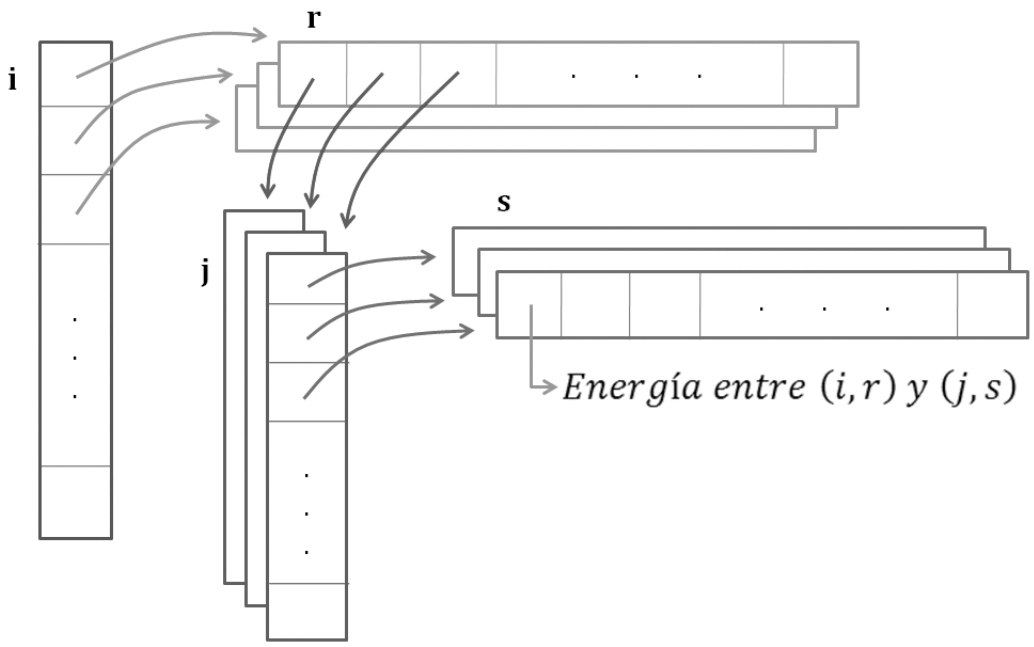

Figura 10: Diagrama de la estructura de la tabla hash de cuatro niveles utilizada.

En la Figura 10 se muestra un diagrama de cómo se utiliza cada uno de los cuatro índices para navegar dentro de la tabla hash, de manera a recuperar el valor de interacción entre dos aminoácidos en un cierto estado rotamérico. A continuación se describe paso a paso la navegación por dicho diagrama.

El primer paso consiste en utilizar el índice i para seleccionar en el primer nivel de la tabla hash, el primer aminoácido. En este paso se puede seleccionar cualquier aminoácido de la proteína, por lo tanto, este nivel es de tamaño $O(n)$. Dentro de cada una de las posiciones

de la tabla hash se almacena la dirección en memoria de una tabla hash diferente. Esto aplica para los niveles del 1 al 3.

El segundo paso consiste en utilizar el índice r para seleccionar en el segundo nivel el rotámero del aminoácido i. Dado que existe una posición en la tabla hash para cada rotámero del aminoácido i, el tamaño de este nivel es $O(p)$.

El tercer paso consiste en utilizar el índice j para seleccionar en el tercer nivel uno de los aminoácidos cercanos al aminoácido i. En dado caso que el aminoácido j seleccionado no se encuentre dentro de la lista de aminoácidos cercanos, entonces la búsqueda de la energía de interacción se detiene y se retorna cero como resultado. Como solamente se tiene un espacio reservado en la tabla hash cuando el aminoácido j es un aminoácido cercano a i, entonces el tamaño de este nivel es $O(1)$.

El cuarto paso consiste en utilizar el índice s para seleccionar en el cuarto nivel el rotámero del aminoácido j. De igual manera que con el segundo nivel, existe una posición en esta tabla hash por cada rotámero del aminoácido j, por lo que el tamaño de este nivel es $O(p)$. Dentro de este último nivel es donde se encuentra almacenado un valor de tipo flotante, el cual representa la energía de interacción entre los aminoácidos i y j.

Al multiplicar el tamaño de cada uno de los niveles obtenemos que el tamaño total de la tabla hash es $O(np^2)$. Como se utilizaron tablas hash, y el número de llaves para cada tabla es muy pequeño, se puede realizar la operación de búsqueda en tiempo $O(1)$.

A pesar de que teóricamente el tamaño de la tabla hash es $O(np^2)$, en la práctica la estructura de datos es más pequeña. Por ejemplo, para la proteína 153l de 185 aminoácidos, el número de interacciones almacenadas en la tabla hash es de 3854; sin embargo, si evaluamos np^2 considerando $n = 185$ y $p = 17.5$ (número promedio de rotámeros por aminoácido en la biblioteca de rotámeros utilizada) tenemos que el número total de interacciones es de 56656. Esta diferencia en el número de interacciones se debe a los siguientes factores:

- El número de Alaninas y Glicinas de la proteína, ya que estos aminoácidos no tienen cadena lateral, y por lo tanto, tampoco tienen rotámeros definidos en la biblioteca.

- Si un rotámero tiene una frecuencia de aparición menor al 1 % entonces es descartado, por lo que no se almacenaría ningún valor de energía relacionado a dichos rotámeros.

- Sea i un aminoácido con rotámero r. La energía de interacción entre i y un aminoácido

j con rotámero s se almacena en la posición de la tabla hash derivada del menor índice entre los dos aminoácidos; es decir, si $i < j$ entonces la energía de interacción se almacena en la posición (i, r, j, s), pero si $j < i$ entonces se almacena en (j, s, i, r). Esta regla se implementó con el propósito de no almacenar dos veces el mismo valor de energía en las posiciones mencionadas de la estructura de datos.

A.2. Conjuntos de pruebas utilizados

En la Sección 4.1 se describieron las características de los conjuntos de pruebas utilizados. A continuación se incluye la lista de los identificadores de las proteínas del PDB que conforman cada conjunto de pruebas.

Tabla 16: Conjunto de pruebas de 65 proteínas.

153l	1a7s	1a8q	1agy	1ako	1amm	1arb	1b9o	1bd8	1bj7	1byi	1c5e	1c9o	1cbn	1cc7	1cem
1cex	1chd	1cku	1ctj	1cz9	1czb	1czp	1d4t	1dhn	1eca	1edg	1gci	1hcl	1ic6	1ifc	1igd
1ixh	1koe	1mla	1mml	1nar	1nls	1noa	1npk	1plc	1qj4	1ql0	1qlw	1qnj	1qq4	1qtn	1qtw
1qu9	1rcf	1thv	1vfy	1vjs	1whi	2baa	2cpl	2end	2hvm	2pth	2rn2	3lzt	5p21	5pti	7rsa
1thx															

Identificadores de las proteínas del PDB del conjunto de pruebas de 65 proteínas.

Tabla 17: Conjunto de pruebas de 373 proteínas.

1aho	1atz	1b2p	1b8z	1b9w	1bgf	1byi	1c02	1c48	1cei	1dpt	1dq0	1dv7	1dys	1e5m	1e6f
1edq	1elk	1erz	1es5	1es9	1f41	1f60	1f94	1fcq	1fo9	1fpo	1fvk	1g61	1g8a	1g8q	1gmu
1go3	1gpp	1gqn	1gs9	1gso	1gvp	1gxn	1h03	1h4a	1h4y	1hcl	1hz6	1hz9	1i4j	1igq	1ijq
1ijy	1ilk	1iu8	1j23	1j2a	1j7g	1jb3	1jcd	1jks	1jy2	1k33	1kmt	1kmz	1kn3	1koe	1kpt
1ku8	1kva	1kve	1kyf	1kzq	1l3k	1lbv	1lm5	1ltu	1m5t	1m6j	1md6	1mf7	1mg7	1mix	1mkk
1mml	1mol	1n1j	1n93	1nar	1nm8	1nwa	1nxm	1oai	1oaq	1ogm	1ok7	1p1x	1p6z	1p9h	1pcf
1pdo	1pe9	1pgv	1pm4	1pxz	1qah	1qkd	1r12	1r29	1r77	1r8n	1rfy	1rgx	1rl0	1rwz	1ryl
1rz2	1s7i	1s7k	1sau	1sh8	1smx	1snt	1sqe	1srv	1suu	1swh	1t1j	1tje	1tks	1tp6	1tua
1tuo	1ty0	1tzv	1u07	1u2h	1u5x	1ueb	1uek	1ujn	1ukf	1uln	1ulr	1unp	1usm	1uxz	1uz3
1v05	1v0s	1v6t	1v7q	1v8e	1v8h	1v8i	1vdk	1ve2	1vgt	1vh5	1vjs	1vpi	1w5r	1w7b	1wba
1wd7	1wka	1wko	1wlg	1wlz	1wm3	1wmh	1wqj	1wtj	1wu9	1wvh	1wyc	1wz3	1x1e	1x2i	1x6i
1xdz	1xfk	1xs0	1xxo	1y2t	1y7y	1yac	1yhh	1yn3	1yo3	1ypf	1yt4	1ytl	1yu5	1yw5	1yxy
1yzm	1z0c	1z0p	1zkr	1zo2	1zrs	1zuh	1zv1	1zva	1zvt	1zxt	2a35	2a6w	2a8f	2ahf	2ahn
2b0a	2b0j	2b2f	2bay	2bk8	2bpd	2bvp	2cg7	2cgh	2chc	2ci3	2ciu	2cov	2cwc	2cwk	2cwl
2cwr	2cyg	2d4p	2d68	2d8e	2dqw	2dyu	2e01	2e10	2e3z	2e64	2e7a	2e8f	2e8g	2e9y	2ebb
2ebe	2ecr	2egj	2ehg	2epi	2etx	2ex0	2f23	2f5g	2f6l	2fbn	2fbq	2fd5	2fhz	2fjz	2fl4
2flu	2frg	2fvh	2fw7	2g2u	2g30	2g40	2g69	2g7i	2g7o	2gas	2gdq	2gec	2ggv	2giy	2gkg
2gkv	2gom	2gqv	2gxg	2h14	2h2r	2h2z	2h7w	2h7z	2h8e	2h8o	2hc8	2hlr	2hly	2hoq	2hpl
2hww	2hy5	2hzf	2i3f	2i49	2i5d	2i6v	2ibl	2ic6	2ic7	2iia	2ijk	2ip2	2ipr	2iru	2ium
2ixm	2iy9	2iz6	2j5y	2j6b	2j71	2j8b	2j9w	2jcp	2nml	2nnu	2npt	2nrr	2nv0	2o0q	2o2k
2o37	2o6s	2o6x	2oeb	2ohw	2oix	2ol7	2osa	2otu	2p1g	2p38	2p4h	2p52	2p5d	2p5k	2p65
2p84	2pbp	2pbq	2pef	2pet	2pge	2pkf	2pmr	2pnd	2pst	2ptv	2pv2	2q8o	2qiy	2qpw	2qr3
2qt4	2r6u	2r77	2r99	2rcz	2rfa	2rik	2rjd	2rk5	2vc8	2yxf	2yyv	2yz1	2z14	2z1e	2z37
2zfy	3bb7	3bn6	3c4s	6xia											

Identificadores de las proteínas del PDB del conjunto de pruebas de 373 proteínas.

Tabla 18: Conjunto de pruebas de 723 proteínas.

1a2j	1a3c	1a53	1a68	1a6m	1aba	1ako	1amf	1amx	1arb	1b2v	1b68	1bam	1bdo	1bea	1bgf
1bj7	1bkf	1bkr	1bm8	1brt	1bud	1bxe	1by2	1byi	1byr	1c3j	1c3p	1c75	1c7k	1cc8	1cei
1ceo	1chd	1cnv	1cpn	1cpq	1cqy	1cuj	1cv8	1cxq	1czs	1d2n	1d3h	1d4o	1dcs	1df7	1dg6
1dg9	1dhn	1dk8	1dlw	1dqg	1dqy	1dun	1dus	1dxj	1dyp	1e58	1edg	1ej0	1elj	1eok	1ep0
1es9	1eur	1euw	1evf	1ew4	1eye	1ezm	1ezw	1f1e	1f2j	1f7l	1faz	1fcq	1fcy	1fk5	1fna
1fo8	1frw	1fua	1fx2	1fye	1g2r	1g5t	1g66	1ga8	1gbs	1gci	1gew	1gmx	1gny	1gp6	1gpp
1gpr	1gq8	1gqv	1gr3	1gsi	1gsj	1gui	1gv9	1gvd	1gvg	1gw1	1gwm	1gxn	1gxq	1gxu	1gyn
1gz8	1h0a	1h0s	1h6l	1h70	1h7l	1hbk	1hcv	1hd2	1hdo	1hh8	1hq0	1huf	1huw	1hv6	1hxi
1hzt	1i0v	1i27	1i2t	1i40	1i6p	1i8o	1i9s	1iab	1icx	1ifc	1ijb	1ilk	1im5	1iom	1iqq
1iqz	1iul	1iuz	1ix4	1ixh	1izc	1j0p	1j27	1j5x	1j8q	1j8u	1j98	1j9b	1jb3	1jbe	1jeo
1jer	1jg1	1jhg	1jjf	1jk7	1jl6	1jm1	1jpc	1jyh	1k12	1k1b	1k30	1k51	1k5c	1k77	1k7c
1k8u	1kfr	1kgd	1kll	1klx	1km4	1knb	1knm	1koe	1kp6	1kr7	1kt6	1kux	1kwf	1kyp	1l3p
1l6p	1l9l	1ljo	1lke	1lki	1ll2	1lmi	1ln4	1lo7	1lr0	1lri	1lrk	1lst	1lu4	1lv7	1lyv
1lzj	1lzk	1m1q	1m2j	1mc2	1mdc	1me3	1mfm	1mg4	1mhn	1mjc	1msc	1msk	1mug	1mun	1muw
1mvl	1n1f	1n3l	1n55	1n8n	1n8u	1n93	1nar	1nb9	1nc5	1nep	1nf9	1nfp	1nkd	1nls	1nnh
1nnx	1nog	1nox	1npk	1npu	1nrg	1nsj	1nte	1nwa	1nwz	1nzj	1o08	1o1x	1o1y	1o1z	1o22
1o3u	1o4r	1o4v	1o4w	1o4y	1o50	1o9g	1oaa	1oap	1od6	1odm	1oej	1oew	1oho	1ojq	1ojr
1ok0	1oot	1opd	1os6	1ozn	1p3c	1p4c	1p5f	1p5x	1p5z	1p90	1p99	1pa7	1pb7	1pbj	1pdo
1php	1pjx	1pmh	1poc	1pqe	1puc	1pwa	1pz4	1pzt	1q0n	1q0r	1q35	1q5z	1q8d	1q92	1qcx
1qdd	1qj4	1qlm	1qqf	1qre	1qtw	1qus	1qwg	1qwy	1qzm	1r0u	1r26	1r29	1r2q	1r3d	1r5y
1r69	1r8m	1rec	1rh9	1ri6	1ris	1rlh	1rlj	1rqw	1rtq	1ru4	1rw1	1rwj	1rwr	1s2w	1s7z
1s9u	1sbp	1sbx	1sdi	1sfp	1sgw	1shu	1sk4	1snc	1sq9	1sra	1srv	1sur	1suu	1svi	1svk
1swx	1sx7	1syy	1t07	1t2i	1t3y	1t46	1t8k	1tbf	1tca	1thf	1tib	1tif	1tjy	1tp6	1tqg
1tr9	1tt8	1tu9	1tuh	1tuk	1tuw	1txj	1tzv	1u14	1u36	1uai	1ual	1ubi	1ucs	1ufy	1ui0
1uku	1ukz	1umg	1umh	1uoz	1upq	1uq5	1uuy	1uv4	1uwf	1v05	1v0a	1v0l	1vhh	1vjo	1vk1
1vk4	1vkk	1vl1	1vlc	1vls	1vmb	1vmg	1vp8	1vpr	1vqb	1vr8	1vyi	1vyr	1w0n	1w3l	1w3u
1w53	1w66	1wba	1wc2	1wd5	1wde	1whi	1whz	1wka	1wlu	1wm3	1wp5	1wri	1wwi	1wxi	1wxj
1x06	1x82	1x8q	1x91	1xbi	1xdn	1xeo	1xgk	1xki	1xkn	1xmk	1xmt	1xqo	1xte	1xtp	1xvo
1y0k	1y8a	1y93	1ye8	1yfq	1yht	1ymk	1yoy	1ypc	1yu5	1ywf	1yzm	1z4r	1z67	1z6m	1z6n
1z70	1zce	1zgk	1zk4	1zma	1znd	1zrn	1zuu	1zv9	1zzk	1zzm	2a14	2a15	2a4v	2a6z	2a7b
2a84	2abk	2abs	2ah5	2ai2	2am9	2amh	2aq8	2asf	2azw	2b06	2b0a	2b3m	2b4w	2b61	2b65
2b69	2b8m	2bbh	2bkf	2brf	2bwq	2c0h	2c3f	2c60	2c71	2c9q	2ccb	2ccq	2ccv	2ccw	2cfe
2chh	2cic	2cj7	2cnq	2ctc	2cu9	2cul	2cw4	2cwy	2cxh	2cxv	2cyg	2d28	2d2j	2d4p	2d81
2dfb	2djh	2dri	2duy	2e3b	2e3h	2ebn	2ecq	2end	2erf	2erl	2esk	2et1	2eut	2ew0	2ewr
2f71	2f9f	2fbh	2fdn	2fdr	2fe5	2ffc	2fg1	2fi1	2fk8	2fou	2fsq	2fsr	2fsu	2fuf	2fup
2fuz	2fvv	2fvy	2fzp	2g2c	2g3a	2g40	2g62	2g64	2g7o	2g9f	2gdm	2gfo	2ggc	2gkp	2gm6
2gnp	2gpi	2gs5	2gui	2gux	2gvk	2h5c	2hcf	2hew	2hje	2hk6	2hkv	2hlj	2hly	2hng	2hts
2huh	2hvf	2hx0	2hxm	2i1b	2i5u	2i6c	2i9c	2ia7	2iay	2ica	2icg	2ie7	2igd	2ihk	2im9
2imq	2isb	2iuw	2ivy	2iw1	2iyv	2izk	2j6b	2j8k	2jdc	2jek	2lis	2mcm	2mhr	2nlr	2nml
2nn8	2nqw	2nr7	2nrk	2nsz	2nwv	2nx2	2nxf	2o0m	2o1a	2o2x	2o8p	2ob5	2oc5	2od5	2oeb
2oh3	2ons	2oo3	2oqz	2oso	2ot9	2ou6	2ov0	2p3k	2p4o	2p5k	2pb1	2pii	2pmr	2pnw	2ppq
2ppx	2pq7	2pst	2ptd	2pth	2pvb	2pxr	2q3m	2q3p	2q3t	2q3w	2q4m	2q4n	2qed	2qia	2qsb
2qsw	2qwc	2qyq	2r2z	2r31	2r4q	2r9f	2rbk	2rfr	2rh3	2rhw	2riu	2rn2	2sak	2uyq	2v3g
2v3k	2v7f	2vb1	2vbu	2vmh	2vpa	2vry	2yvt	2yxf	2z3v	2z94	2z98	2zfi	3b7c	3bcj	3bfp
3bi7	3boe	3bzt	3ckm	3dm8	3dni	3due	3e99	3eby	3ejv	3eln	3en8	3il8	3nul	3vub	4rhn
5csm	6fiv	8abp													

Identificadores de las proteínas del PDB del conjunto de pruebas de 723 proteínas.

A.3. Hibridación en recocido simulado

El algoritmo de recocido simulado (Algoritmo 6) que se utilizó en los experimentos presentados en el Capítulo 4, se ejecuta junto con el algoritmo voraz para reducir el número de colisiones (Algoritmo 4) y el de búsqueda local (Algoritmo 13). En la Tabla 19 se muestran los resultados obtenidos al ejecutar el algoritmo de recocido simulado combinándolo con los algoritmos antes mencionados. Los experimentos se realzaron utilizando el conjunto de pruebas de 65 proteínas.

Tabla 19: Comparación de la calidad de las .

Algoritmo	$\chi_1(\%)$	$\chi_{1+2}(\%)$	RMSD(Å)	Colisiones
RS	80.06	60.08	0.8592	71
RS + BL	82.20	62.28	0.8150	77
RS + AV	80.62	60.71	0.8479	26
RS + BL + AV	82.54	62.61	0.8080	20

Diferentes esquemas de hibridación del algoritmo de recocido simulado (RS), con el algoritmos de búsqueda local (BL) y el algoritmo voraz (AV).

Se puede observar que el algoritmo de recocido simulado logra resultados inferiores cuando no se combina con el de búsqueda local. Sin embargo, en los casos en los que se combina recocido simulado con búsqueda local, se logra una mejor precisión que la que obtiene cada uno de ellos por separado. Con base en esto, no se aconseja ejecutar el algoritmo de recocido simulado por su cuenta, ya que no sólo la precisión que obtiene es inferior, sino que requiere más tiempo de cómputo para ejecutarse. Sin embargo, si se combina con búsqueda local y el algoritmo voraz, se obtienen mejores resultados.

A.4. Comparación de la estructura inicial y la estructura predicha

En la Sección 4.3.2 se presentó la calidad de las estructuras predichas por los algoritmos implementados. A continuación se muestra una comparación entre las estructuras predichas y las estructuras iniciales, las cuales se crearon seleccionando para cada aminoácido el rotámero con la mayor probabilidad de entre los rotámeros que no fueron eliminados por el algoritmo DEE.

En las tablas 20, 21 y 22 se presentan los resultados obtenidos. La precisión promedio de la estructura inicial es de 73.89 % para $\chi_1(\%)$ y de 55.35 % para $\chi_{1+2}(\%)$.

Tabla 20: Comparación de la estructura inicial y la estructura predicha en el conjunto de 65 proteínas.

	Estructura inicial					Estructura predicha				
Algoritmo	$\chi_1(\%)$	$\chi_{1+2}(\%)$	RMSD	C_{prom}	Energía	$\chi_1(\%)$	$\chi_{1+2}(\%)$	RMSD	C_{prom}	Energía
BT	74.29	55.15	0.9614	5.8769	5397.73	82.58	62.79	0.8054	0.3076	3497.02
RS	74.29	55.15	0.9614	5.8769	5397.73	82.54	62.61	0.8080	0.3076	3490.38
BL	74.29	55.15	0.9614	5.8769	5397.73	82.13	62.19	0.8154	0.3076	3460.09

Comparación de la calidad entre la estructura inicial y la estructura predicha para los algoritmos de búsqueda tabú (BT), recocido simulado (RS) y búsqueda local (BL), en el conjunto de 65 proteínas. C_{prom} representa el número de colisiones promedio por proteína.

Tabla 21: Comparación de la estructura inicial y la estructura predicha en el conjunto de 373 proteínas.

	Estructura inicial					Estructura predicha				
Algoritmo	$\chi_1(\%)$	$\chi_{1+2}(\%)$	RMSD	C_{prom}	Energía	$\chi_1(\%)$	$\chi_{1+2}(\%)$	RMSD	C_{prom}	Energía
BT	73.07	55.06	1.0234	4.7625	5940.28	81.06	61.89	0.8708	0.2064	3103.08
RS	73.07	55.06	1.0234	4.7625	5940.28	80.93	61.68	0.8732	0.2064	3084.45
BL	73.07	55.06	1.0234	4.7625	5940.28	80.54	61.21	0.8811	0.2144	3088.95

Comparación de la calidad entre la estructura inicial y la estructura predicha para los algoritmos de búsqueda tabú (BT), recocido simulado (RS) y búsqueda local (BL), en el conjunto de 373 proteínas. C_{prom} representa el número de colisiones promedio por proteína.

En el conjunto de pruebas de 65 proteínas, la estructura predicha presentó un aumento promedio de 8.12 % para $\chi_1(\%)$ y de 7.38 % para $\chi_{1+2}(\%)$ con respecto a la estructura inicial. En cuanto a la energía, esta disminuyó un promedio de 35.48 %.

En el conjunto de pruebas de 373 proteínas, la calidad de la estructura predicha mejoró 7.77 % y 6.53 %; para $\chi_1(\%)$ y $\chi_{1+2}(\%)$, respectivamente. En cuanto a la energía, esta disminuyó un promedio de 47.94 %.

Tabla 22: Comparación de la estructura inicial y la estructura predicha en el conjunto de 723 proteínas.

	Estructura inicial					Estructura predicha				
Algoritmo	$\chi_1(\%)$	$\chi_{1+2}(\%)$	RMSD	C_{prom}	Energía	$\chi_1(\%)$	$\chi_{1+2}(\%)$	RMSD	C_{prom}	Energía
BT	74.29	55.82	0.9933	5.2851	5978.46	81.71	62.35	0.8483	0.2489	3319.02
RS	74.29	55.82	0.9933	5.2851	5978.46	81.57	62.12	0.8502	0.2392	3301.29
BL	74.29	55.82	0.9933	5.2851	5978.46	81.26	61.74	0.8572	0.2406	3302.48

Comparación de la calidad entre la estructura inicial y la estructura predicha para los algoritmos de búsqueda tabú (BT), recocido simulado (RS) y búsqueda local (BL), en el conjunto de 723 proteínas. C_{prom} representa el número de colisiones promedio por proteína.

En el conjunto de pruebas de 723 proteínas, se observó un incremento promedio en la precisión de la estructura predicha de 7.22 % para $\chi_1(\%)$ y de 6.25 % para $\chi_{1+2}(\%)$. En cuanto a la energía, esta disminuyó un promedio de 44.67 %.

Se puede observar que en los tres conjuntos de pruebas, el algoritmo de búsqueda tabú fue el que obtuvo la mejor precisión; sin embargo, también fue el que obtuvo la mayor energía. Con lo cual se observa nuevamente que el minimizar la función de energía implementada no implica que se obtendrán estructuras con una mejor precisión.