UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

SIMULACIONES MONTE CARLO DE UN MODELO DE PLEGAMIENTO DE PROTEÍNAS

T E S I S

QUE PARA OBTENER EL TÍTULO DE

F Í S I C O

P R E S E N T A

DIEGO GARRIDO RUIZ

DIRECTOR DE TESIS:
DR. DAVID PHILIP SANDERS

2011

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Datos del Jurado

1.Datos del alumno ..... 1. Datos del alumno
Apellido paterno ..... Garrido
Apellido materno ..... Ruiz
Nombre(s) ..... Diego
Teléfono 55 68 67 86
Universidad Nacional Autónoma de México .... Universidad Nacional Autónoma de México
Facultad de Ciencias ..... Facultad de Ciencias
Carrera ..... Física
Número de cuenta 405007460

2. Datos del tutor ..... 2. Datos del tutor
Grado ..... Dr
Nombre(s) ..... David Philip
Apellido paterno ..... Sanders

3. Datos del sinodal 1 ..... 3. Datos del sinodal 1
Grado ..... Dr
Nombre(s) ..... Ramón
Apellido paterno ..... Peralta
Apellido materno ..... y Fabi

4. Datos del sinodal 2 ..... 4. Datos del sinodal 2
Grado ..... Dr
Nombre(s) ..... Enrique
Apellido paterno ..... Hernández
Apellido materno ..... Lemus

5. Datos del sinodal 3 ..... 5. Datos del sinodal 3
Grado ..... Dr
Nombre(s) ..... Pablo
Apellido paterno ..... Padilla
Apellido materno ..... Longoria

6. Datos del sinodal 4 ..... 6. Datos del sinodal 4
Grado ..... Dr
Nombre(s) ..... Luis Antonio
Apellido paterno ..... Pérez
Apellido materno ..... López

7.Datos del trabajo escrito ..... 7. Datos del trabajo escrito
Título ..... Simulaciones Monte Carlo de un modelo de plegamiento de proteínas
Número de páginas ..... 116 p.
Año 2011

Agradecimientos

Al Dr. David Philip Sanders por guiarme a lo largo de este trabajo, por su interés y su disposición, por su paciencia y su apoyo, por todo lo que he aprendido de él en estos años y, principalmente, por su amistad.

A la UNAM por todas las oportunidades que me ha brindado.

Al Dr. Daniel Alejandro Fernandez Velasco y al Dr. Alejandro Sosa Peinado por todo lo que me han enseñado, por sus observaciones sobre este trabajo y por su pasión por las proteínas.

Al Dr. Enrique Hernández Lémus, al Dr. Pablo Padilla Longoria, al Dr. Ramón Peralta y Fabi y al Dr. Luis Antonio Pérez López por sus invaluables observaciones y sugerencias para este trabajo.

A mis colegas: Heiblum, Marduk, Asaf, Vicente, Carlos y Frias por su amistad, compañia y ayuda.

A mis amigos Vector y Dary.

A la Pulga por su apoyo y amistad.

A Time-Machine por la complicidad.

A mis abuelos.

A mi familia: Gabriela, Modesto, Pavis, Puck y Folie por su compañia y apoyo. Fiu Fu Fu.

La Nature est un temple où de vivants piliers
Laissent parfois sortir de confuses paroles ;
L'homme y passe à travers des fôrets de symboles
Qui l'observent avec des regards familiers.

Charles Baudelaire

Resumen

Se estudia el proceso del plegamiento de proteínas a través de simulaciones computacionales de un modelo simple, conocido como el *modelo HP*; además de un conjunto completo de movimientos reversibles, conocido como *pull moves*, y dos métodos tipo *Monte Carlo*, el algoritmo de *Metropolis* y el algoritmo *Wang–Landau*. Se implementa una simulación con la técnica de *recocido simulado* para ayudar a encontrar la configuración de mínima energía, dada una cadena en particular. Se desarrollaron simulaciones computacionales para reproducir la dinámica del plegamiento, en el caso del algoritmo de Metropolis, y para obtener una estimación de la densidad de estados del sistema, en el caso del algoritmo Wang–Landau. Se busca obtener un entendimiento *general* del proceso de plegamiento a través de las simulaciones que, debido a la simplicidad del modelo, no proporcionan resultados concretos y particulares, sino más bien proveen una idea general del proceso. Las simulaciones se implementaron tanto para una red cuadrada como para una red triangular. Esta última geometría de la red no ha sido estudiada tan a profundidad como la red cuadrada, por lo que se generalizaron los movimientos del caso cuadrado al caso triangular. A partir de ambos conjuntos de simulaciones, se extrajo información relevante al proceso de plegamiento, a partir de la cual se busca inferir características generales del mismo proceso. Se reconstruye la información termodinámica del sistema: el calor específico por monómero, la energía promedio en función de la temperatura y la distribución de energías en función de la temperatura, a partir de los resultados de las simulaciones.

Se contrastan las diferencias entre los resultados obtenidos utilizando la red cuadrada y aquéllos obtenidos utilizando la red triangular, así como las diferencias entre ambos algoritmos. Finalmente, se discute cuál de las dos redes presenta un comportamiento más realista.

Índice general

Capítulo 1

Introducción

En este capítulo haremos una breve introducción al plegamiento de proteínas. Primero, introduciremos términos y conceptos básicos de la parte bioquímica del proceso para poder definir en qué consiste el problema en sí. Luego daremos un breve panorama del tratamiento computacional del plegamiento de proteínas: qué se ha hecho y qué se está haciendo.

1.1. ¿Qué es una proteína?

Las proteínas son compuestos orgánicos poliméricos formados por aminoácidos de origen genético. Cada proteína es un polímero formado por combinaciones de hasta 20 diferentes tipos de aminoácidos. Una vez que estos forman parte de la cadena protéica, se les llama *residuos*. Las proteínas se clasifican como nanopartículas debido a su tamaño. Se caracterizan por su gran diversidad y versatilidad, y están presentes en cualquier bioorganismo. Existe una gran gama de funciones distintas que realizan distintas proteínas. Algunas funcionan como enzimas, catalizando reacciones bioquímicas esenciales para procesos metabólicos; otras realizan funciones de transporte, atrapando ciertos compuestos y acarreándolos hasta el lugar donde se necesitan; algunas otras realizan funciones estructurales, como en la piel o el cabello.

El número de aminoácidos en una proteína de tamaño promedio es de alrededor de 200–300; sin embargo, hay algunas que son mucho más pequeñas, del orden de 20 aminoácidos, y unas que son mucho más grandes, alrededor de 27,000 aminoácidos.

Las proteínas se forman mediante un proceso llamado *biosíntesis de proteínas*. El primer paso en este proceso es la *formación* de los aminoácidos. Ésta se realiza a través de procesos metabólicos a partir de fuentes de carbono como la glucosa. El siguiente paso se conoce como *transcripción*; en el núcleo de la célula donde se realiza la biosíntesis, se genera una molécula conocida como **ARN mensajero**, o **ARNm**, donde se encuentra la información genética obtenida del **ADN** para la formación de una proteína. Por último, está el proceso de *traducción*, que ocurre en el citoplasma de la célula, donde se encuentran los ribosomas. Estos traducen la información genética contenida en el **ARNm** en una secuencia específica de aminoácidos que se pliega para formar una proteína.

La estructura de una proteína en un solvente no es completamente rígida, sino que sufre variaciones debido a colisiones con otras moléculas o a fluctuaciones térmicas. También existen cambios conformacionales de la estructura de una proteína, generalmente inducidos al atrapar una molécula en alguna región de la proteína que reaccione ante el contacto. Un ejemplo de esto sería el cambio conformacional que sufre la hemoglobina al "atrapar" una molécula de oxígeno para transportarla a algún lugar del cuerpo.

Existen 4 diferentes niveles de estructura de una proteína:

- Estructura primaria: Se refiere a la secuencia de diferentes aminoácidos de la proteína. Los residuos están unidos por enlaces covalentes o enlaces peptídicos formados durante el proceso de traducción en la biosíntesis de la proteína. La secuencia de aminoácidos es única para cada proteína, determina su estructura y, por lo tanto, su función.
- Estructura secundaria: Se refiere a subestructuras locales regulares en la proteína, como las *hélices alfa* o las *hojas beta*.
- Estructura terciaria: Se refiere a la estructura tridimensional global de una proteína en específico.
- Estructura cuaternaria: Se refiere a la estructura formada por varias proteínas ligadas. No todas las proteínas tienen estructura cuaternaria. Las uniones entre proteínas son por lo general no covalentes.

Las proteínas también se dividen según su estructura terciaria:

- Proteínas globulares: Son solubles en agua; muchas de éstas son enzimas. La mayor parte de las proteínas son de este tipo.

- Proteínas fibrosas: Suelen ser aquéllas con funciones estructurales.
- Proteínas de membrana: Son aquéllas que se asocian a membranas celulares o de algún organelo.

Las proteínas que se consideran en este trabajo son proteínas globulares. Este hecho se toma en consideración y se discute al momento de elegir el modelo a través del cual estudiaremos a las proteínas.

En la literatura, existen distinciones entre proteína y péptido: proteína se usa para describir la biomolécula completa en una conformación estable, mientras que por péptido se entiende una cadena de corta longitud, entre 20 y 30 residuos, que no tiene una configuración estable. En el siguiente capítulo se retomará la discusión de las secuencias que exhiben un comportamiento similar al de una proteína y aquéllas que no, en el contexto del modelo que utilizamos.

Para más detalle, ver la referencia [1].

1.2. El plegamiento de proteínas

La función de una proteína está determinada por su estructura *nativa*, es decir, la estructura tridimensional final a la que se llega mediante el proceso de plegamiento. Para alcanzar estas estructuras funcionales, la proteína sufre cambios configuracionales por medio de interacciones no covalentes como el empacamiento hidrofóbico, la formación de puentes de hidrógeno, las interacciones iónicas, y las fuerzas de Van der Waals, entre otras. A este proceso se le conoce como *plegamiento*.

El proceso de plegamiento es el proceso físico a través del cual una proteína se transforma, desde una configuración cualquiera, en una estructura tridimensional funcional. Al conjunto de las estructuras "desplegadas" se le conoce como *estado desnaturalizado*. En el estado desnaturalizado existen una gran variedad de estructuras distintas. El proceso de plegamiento depende del solvente en el que se encuentre sumergida la proteína, de la salinidad, el pH, la temperatura, y la presencia de otras moléculas que asistan al plegamiento, entre otros factores.

El plegamiento es un proceso espontáneo que comienza durante o justo después de la sintésis de la cadena protéica. Existe el proceso "inverso", es decir, el desplegamiento de proteínas. Este desplegamiento puede inducirse alterando las condiciones del solvente en que se encuentre la proteína. Para alcanzar el estado nativo, algunas proteínas necesitan de moléculas chaperonas que asisten al proceso de plegamiento.

El interés por el proceso de plegamiento de proteínas surgió en la decada de 1960 con las primeras imágenes de resolución atómica de estructuras protéicas. De particular interés son las siguientes tres preguntas concretas [2]:

1. ¿Qué balance de fuerzas interatómicas interviene en el proceso de plegamiento para alcanzar el estado nativo de una secuencia?
2. ¿Cómo poder determinar la estructura nativa de una proteína a partir de una secuencia de aminoácidos?
3. ¿Qué estrategias toma la proteína para alcanzar su estado nativo tan rápidamente?

El proceso de plegamiento de proteínas es un proceso estocástico, debido a fluctuaciones térmicas; una secuencia sigue una trayectoria microscópica al plegarse, mientras que una réplica idéntica puede seguir alguna otra trayectoria.

1.2.1. ¿Por qué es importante el plegamiento de proteínas?

El proceso de plegamiento de proteínas es importante puesto que a través de él alcanzan su estructura funcional todas las proteínas necesarias para la vida. Como mencionamos anteriormente, las proteínas realizan una gran cantidad de funciones diversas, sin las cuales un organismo no podría subsistir. Entender cómo estas moléculas cambian de configuración hasta llegar a su estado funcional es uno de los grandes problemas de la biología molecular. Además, existen varias enfermedades que se deben a la formación de estructuras protéicas anómalas. Al plegarse a estas estructuras anómalas en vez de a las estructuras nativas usuales, las proteínas pueden perder su función o volverse tóxicas. Algunos ejemplos de este tipo de enfermedades son Alzheimer, Parkinson, diabetes mellitus tipo 2 e incluso cáncer [3, 4]. Si se pudiera elucidar exactamente el mecanismo del plegamiento, se podría, en teoría, diseñar tratamientos para estas y otras enfermedades.

1.2.2. Hipótesis de Anfinsen

A mediados del siglo XX, Christian B. Anfinsen, bioquímico estadounidense, propuso un postulado que vendría a conocerse como *Hipótesis de Anfinsen* o *Hipótesis termodinámica* [5], y le ameritaría el premio Nobel de química en 1972. Este postulado dice que, al menos para proteínas globulares, la secuencia de aminoácidos determina *completamente* el estado nativo de una proteína. Esto implica que para un

mismo conjunto de condiciones "ambientales" para el proceso de plegamiento, como temperatura, tipo de solvente, salinidad y demás condiciones, el estado nativo corresponde a un mínimo de energía libre que es *único*, *estable* y *cinéticamente accesible*. El que el estado nativo sea único implica que la secuencia no puede exhibir ninguna otra configuración con energía libre comparable. La estabilidad del estado nativo garantiza que pequeños cambios a las condiciones ambientales no provoquen cambios a la estructura de mínima energía. Además, este postulado sostiene que el estado nativo no depende de la ruta cinética tomada por la proteína.

La hipótesis de Anfinsen permitió estudiar el plegamiento *in vitro*, es decir, en tubos de ensayo en vez de dentro de una célula, gracias a que el estado nativo depende sólo de la secuencia de aminoácidos y las condiciones del solvente. El hecho de que una proteína pueda desnaturalizarse *in vitro* y que regrese a su estructura nativa cuando las condiciones ambientales sean favorables, abrió las puertas al estudio experimental del plegamiento en laboratorios, ya no en células. Encima de esto, el trabajo de Anfinsen deja claro que a pesar de que la evolución juega un papel importante para seleccionar las cadenas de aminoácidos que finalmente se pliegan en proteínas, son cuestiones fisico-químicas las que determinan el equilibrio y el mecanismo de plegamiento.

1.2.3. ¿Qué interacciones intervienen en el plegamiento?

Antes de 1980, se creía que el *código* de plegamiento, es decir, el conjunto de "reglas" que rigen el plegamiento, era la suma de pequeños factores que guiaban el proceso, como formación de puentes de hidrógeno, interacción hidrofóbica, interacciones de Van der Waals, enlaces iónicos y demás; se pensaba además que la estructura terciaria de una proteína era consecuencia de la estructura secundaria, la cual, a su vez, era consecuencia de la estructura primaria [6]. Sin embargo, gracias al desarrollo de modelos basados en mecánica estadística, ahora se considera que existe un elemento dominante que guía al proceso de plegamiento a la estructura nativa; este elemento es la interacción hidrofóbica [7]. Esta interacción se da entre residuos hidrofóbicos, que buscan reducir la superficie de contacto con algún medio acuoso en el cual se encuentran sumergidos. Existen varios hechos que fundamentan esta creencia. En particular, se sabe que las proteínas globulares exhiben núcleos hidrofóbicos [1], lo cual se debe a que los residuos de este tipo buscan tener el menor contacto con el solvente; por otro lado, se observa en general que las proteínas se desnaturalizan fácilmente en solventes no polares. Otra razón para creer que la interacción hidrofóbica domina el plegamiento es que algunas secuencias de aminoácidos han sido creadas a partir de re-

organizar los residuos de otra secuencia, manteniendo únicamente la polaridad de los residuos, es decir, se reemplazan los aminoácidos de la cadena por otros aminoácidos con la misma polaridad que el original, y a pesar de esto, estas secuencias se pliegan a las estructuras nativas esperadas [8]. El empacamiento hidrofóbico es favorable por dos razones: minimiza el área de contacto entre residuos hidrofóbicos y el solvente, y agrupa estos residuos hidrofóbicos de tal modo que se forman interacciones de Van der Waals entre ellos. Actualmente se cree que en el plegamiento intervienen interacciones locales y no locales, y que la estructura terciaria es tanto una consecuencia como una causa de la estructura secundaria y viceversa.

1.2.4. Paradoja de Levinthal

En 1968, Cyrus Levinthal, físico y biólogo molecular estadounidense, se dio cuenta de que, debido a que una cadena proteíca tiene un gran número de grados de libertad, la molécula tiene una cantidad enorme de conformaciones posibles [9]. Si una cadena de aminoácidos extendida quisiera llegar a su estructura nativa visitando secuencialmente todas las configuraciones que le son accesibles, tardaría muchísimo tiempo en terminar de plegarse, del orden de la edad de universo. Sin embargo, las proteínas pequeñas se pliegan en tiempos de milisegundos o microsegundos; algunas incluso se pliegan en tiempos de centenas de nanosegundos. Por lo tanto, las proteínas no hacen una búsqueda aleatoria indiscriminada en su espacio de conformaciones para llegar a la estructura nativa,sino su estrategia tiene que ser diferente.

De la observación de Levinthal, surgen varias interrogantes: ¿cómo es que distintas estructuras en el estado desnaturalizado se pliegan a la misma estructura nativa, no necesariamente siguiendo la misma ruta? ¿Cómo se eligen las configuraciones que se visitan?

Existen varios modelos de mecanismos de plegamientos. Levinthal mismo propuso que uno de los mecanismos posibles a través de los cuales una secuencia de aminoácidos alcanzaría su estructura nativa es formando interacciones locales que ayuden a encaminar a la proteína hacia su meta. Esto puede verse como que la proteína va armando "núcleos" locales de plegamiento alrededor de los cuales se desarrolla posteriormente la estructura global. Experimentalmente se han encontrado, para ciertas proteínas, estructuras intermediarias en el proceso de plegamiento que concuerdan con esta idea [10].

Otros mecanismos propuestos para el plegamiento incluyen modelos de *difusión–colisión* [11], donde pequeños dominios se forman y luego se difunden y colisionan

entre sí para formar estructuras más grandes; el mecanismo de *nucleación–condensación* [12], que propone que existe un conjunto de estructuras de transición que dan paso a la formación de la estructura nativa; un modelo que propone que las proteínas se van ensamblando gradualmente por subunidades estructurales llamadas *foldones* [13]; entre muchos otros.

1.2.5. Paisaje de energía

A pesar de las contribuciones de la hipótesis de Anfinsen y la paradoja de Levinthal para la teoría del plegamiento de proteínas, la mayor parte de los esfuerzos para abordar el problema eran de carácter experimental. A finales de la década de 1980, Joseph Bringelson y Peter Wolynes propusieron un modelo teórico a este problema: el paisaje de energía de una proteína, que es la función matemática F que describe la energía libre de una proteína en función de sus grados de libertad. Gráficamente, consiste en mapear todos los estados de la proteína según el nivel energético que les corresponda en un espacio bi- o tri-dimensional. A partir de esta idea, se propuso el *principio de mínima frustración* [14], que dice que las proteínas han sido "seleccionadas" por el proceso de evolución de tal modo que las secuencias alcanzan el estado nativo eficientemente, con pocos mínimos locales en el paisaje que actúen como "trampas". La *frustración* se refiere a los conflictos entre distintas contribuciones a la energía y da lugar a la rugosidad del paisaje de energía, es decir, las trampas. En otras palabras, el paisaje de energía exhibe mínimos locales que actúan como trampas cinéticas en el proceso de plegamiento.

1.2.6. Embudo de plegamiento

Como consecuencia del principio de mínima frustración, para las secuencias seleccionadas por la evolución, surge la idea de que las proteínas tengan un paisaje de energía en forma de embudo [15]. Esto básicamente quiere decir que no existen mínimos locales profundos, en comparación al mínimo global, de tal modo que el paisaje de energía, a pesar de ser rugoso, no está forrado de trampas cinéticas importantes, y que la *entropia configuracional*, es decir, el número de configuraciones por nivel de energía, es muy grande para energías altas y muy pequeño para energías bajas. En la literatura se le conoce como *folding funnel* o embudo de plegamiento. De esta manera, la secuencia puede alcanzar su estado nativo, que corresponde al fondo del embudo, siguiendo diferentes rutas a lo largo del paisaje de energía. A pesar de que se espera cierto grado de rugosidad en el "embudo", el estado nativo permanece cinéticamente

accesible, gracias a que las interacciones que contribuyen a la formación de la estructura nativa dominan sobre combinaciones de otras interacciones como nos indica el principio de mínima frustración. A través del embudo de plegamiento se pueden estudiar muchos aspectos del proceso, como su robustez o la evolución de las estructuras [16, 17, 18].

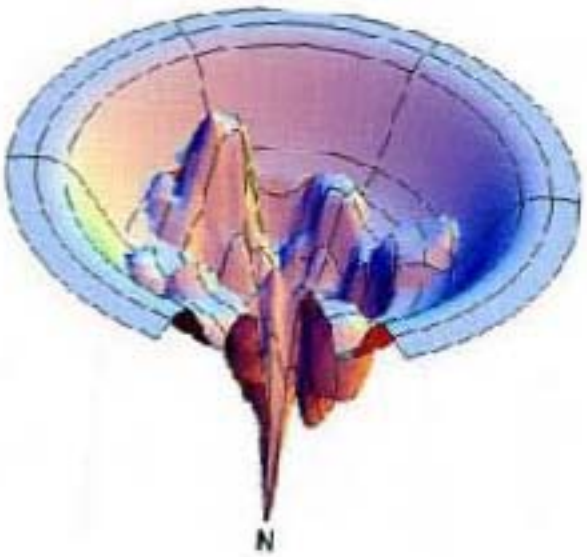


Figura 1.1: Ilustración del concepto del embudo de plegamiento. La energía disminuye con la profundidad en el eje vertical, mientras que en los otros ejes se grafican coordenadas de reacción abstractas. Conforme nos acercamos a la punta del embudo, la energía decrece. Observamos varios mínimos locales, con profundidades marcadamente menores a la profundidad del estado nativo, señalado por *N*. La figura fue tomada de la referencia [2].

La forma de embudo del paisaje de energía para proteínas, figura 1.1, ilustra la idea de diferentes rutas de plegamiento para una misma secuencia y las propiedades de la mecánica estadística del sistema. La transición a la estructura nativa es más que un simple cambio de configuración –es una transición de fase entre un estado desordenado y uno ordenado. Hoy en día se sabe que existen diferentes rutas de plegamiento para una misma secuencia bajo las mismas condiciones. Además se ha observado que diferentes rutas dominan bajo diferentes condiciones. Esto sostiene la idea del paisaje en forma de embudo, puesto que permite la existencia de diferentes "trayectorias" ya que el proceso depende sólo de alcanzar el estado de mínima energía.

Una de las predicciones más importantes del embudo de plegamiento es que algunas secuencias no pueden alcanzar su estado nativo debido a que éste es cinéticamente inaccesible, a pesar de ser termodinámicamente estable. Este hecho sugiere la existencia de un criterio de selección de secuencias que pueden formar estructuras protéicas.

La idea de que el paisaje de energía para las proteínas naturales tiene forma de embudo es consistente tanto con resultados de simulaciones computacionales como con estudios experimentales del proceso de plegamiento [2]. La topología de la proteína determina la forma del paisaje de energía, lo cual a su vez determina el mecanismo de plegamiento.

1.3. Modelación del proceso de plegamiento

En un principio, el estudio del plegamiento de proteínas se consideraba como un problema de bioquímica experimental, donde cada secuencia particular debía estudiarse como un sistema diferente. Actualmente, el proceso de plegamiento de proteínas se estudia a nivel teórico, experimental y computacional; cada uno de estas vertientes contribuye al entendimiento general del proceso, además de verificar postulados o resultados de otros trabajos. Como ejemplo, el desarrollo de la teoría del plegamiento ayudó a interpretar resultados experimentales, e incluso a diseñar nuevos experimentos. A su vez, los trabajos experimentales han dado forma a la teoría, además de verificar algunos de los postulados provenientes de ésta.

El estudio del plegamiento de proteínas ha mostrado que la conjunción de distintas ramas de la ciencia es una necesidad para abordar este tipo de problemas. Por esto, es abordado por científicos provenientes de distintos ámbitos, como médicos, biólogos, químicos y físicos, que unen sus esfuerzos trabajando en conjunto. Esto ha dado lugar al desarrollo de una gran gama de técnicas o métodos de estudio del problema, además de la incorporación de conceptos de cada una de las distintas disciplinas.

El presente trabajo consiste en un tratamiento computacional del problema, por lo que a continuación se ofrece un breve panorama de los esfuerzos realizados en esta dirección para el estudio del plegamiento de proteínas.

1.3.1. Modelos de grano-grueso

Debido a la complejidad del problema y sus altos costos computacionales, se han desarrollado modelos de *grano-grueso*, es decir, de menor resolución, que han per-

mitido estudiar aspectos termodinámicos y cinéticos del plegamiento a través de la mecánica estadística, con un menor costo computacional. Algunos modelos restringen la complejidad aún más trabajando con representaciones de grano-grueso en redes.

Los trabajos de grano-grueso han seguido dos grandes vertientes: el enfoque fenomenológico, a partir de los trabajos de Bryngelson y Wolynes [19], quienes postularon la existencia del paisaje de energía para una proteína, y el enfoque mecánico estadístico, a partir de los trabajos de Shakhnovich y colaboradores [20], que se basa en un análisis mecánico estadístico de modelos microscópicos que no asumen la existencia de un paisaje de energía ni preferencias configuracionales a priori, sino que deriva el paisaje de energía desde primeros principios.

Con modelos de grano-grueso, no se espera obtener resultados para proteínas específicas, sino resultados generales comunes al plegamiento de cualquier proteína [21]. Incluso estos modelos de grano-grueso pueden dar a entender más del proceso de lo que se cree, ya que el mecanismo de plegamiento parece depender más de la topología del estado nativo que de detalles atómicos [22].

1.3.2. Trabajos computacionales de plegamiento

A través de herramientas computacionales se puede estudiar el proceso de plegamiento y predecir estructuras de proteínas. Quizás la técnica más común para estudiar el plegamiento es la de *dinámica molecular*, que es un tipo de simulación computacional donde se dejan interactuar modelos de átomos o moléculas mediante un potencial de interacción. Sin embargo, resulta ser demasiado pesado computacionalmente correr una dinámica molecular para simular tiempos largos (a partir de microsegundos) y cadenas largas con resolución de todos los átomos. El costo computacional para dinámica molecular con detalle atómico y solvente explícito restringe los tiempos de simulación y las longitudes de las proteínas que se pueden estudiar. A causa de esto, se han desarrollado muchas técnicas y modelos diferentes que permiten estudiar el proceso de plegamiento con menor resolución pero con menor costo computacional. Es importante destacar que las simulaciones de este tipo de modelos simples no intentan encontrar particularidades del proceso para una proteína en específico, sino más bien poder decir algo acerca del proceso de plegamiento en general para cualquier proteína. Debido a que son las interacciones entre residuos las que guían el proceso de plegamiento, las simplificaciones sobre la estructura de la proteína se traducen en simplificaciones sobre la función de energía; esto se abordará con mayor detalle en el capítulo 2.

A pesar de las limitaciones que tiene el abordar este problema de manera computacional, han habido grandes avances en este aspecto, principalmente debido al desarrollo de supercomputadoras y del cómputo distribuido. Uno de los primeros grandes hitos fue una simulación en una supercomputadora, realizada en 1998 por Duan y Kollman en donde simularon el plegamiento de la cabeza de Villina, de 36 residuos, con un modelo de solvente explícito por un microsegundo [23]. La diferencia entre la estructura obtenida por la simulación y la estructura obtenida por resonancia nuclear magnética era de 4.5 Å. Otro esfuerzo de gran importancia es el desarrollo del proyecto *Folding@home* [24], una red de cómputo distribuido que corre como protector de pantalla en computadoras de voluntarios alrededor del mundo. Este proyecto es desarrollado por el grupo de Pande en la universidad de Stanford. En 2010 se logró por primera vez realizar simulaciones del orden de milisegundos; por un lado, por el grupo de Pande [25][1] y por otro, por el grupo privado de Shaw [26].

1.4. Plan de la tesis

En este trabajo se utilizará un modelo simple de proteínas en redes bidimensionales junto con un conjunto de movimientos y dos métodos Monte Carlo para estudiar el proceso de plegamiento. Con uno de los métodos utilizados, podemos estudiar la transición al estado de mínima energía a una temperatura fija, mientras que con el otro, podemos obtener un estimado del número de configuraciones por energía para una cadena en particular, independientemente de la temperatura. Mediante ambos métodos obtendremos información termodinámica del sistema e información del proceso en sí. Finalmente se discutirán las diferencias entre diferentes geometrías de las redes sobre las que se simula el proceso, para averiguar qué geometría da lugar a resultados más realistas.

En los siguientes capítulos, introduciremos en detalle el modelo, los movimientos y los métodos utilizados en las simulaciones que fueron desarrolladas. Una vez teniendo claro cómo funciona cada parte de la simulación, se explicará en qué consiste, es decir, de qué manera todos los componentes se juntan para dar lugar a una simulación de la cual podemos extraer resultados. Posteriormente, se presentarán los resultados de las simulaciones, se interpretarán y se discutirán para finalmente puntualizar las conclusiones del trabajo.

El texto se divide en dos partes; en la primera se explican las bases del estudio del

[1] El video de la simulación se encuentra en: `http://folding.typepad.com/news/2010/01/major-new-result-from-foldinghome-simulation-of-the-millisecond-timescale.html`

plegamiento de proteínas en el presente trabajo, mientras que en la segunda parte se presentan, analizan e interpretan los resultados de dicho estudio. En el primer capítulo se hace una breve introducción al plegamiento de proteínas. En el segundo capítulo se explica el modelo que se utiliza en este trabajo, así como el conjunto de movimientos que utilizamos para generar nuevas configuraciones del sistema. El tercer capítulo está dedicado a los métodos utilizados: el algoritmo de Metropolis y el algoritmo Wang–Landau; así como la técnica de recocido simulado. Además, se explica cómo todos estos elementos se juntan para dar lugar a las simulaciones. La segunda parte del texto comienza en el cuarto capítulo, donde se exponen los resultados obtenidos, los cuales se analizan, interpretan y contrastan. Finalmente, en el quinto capítulo se presentan las conclusiones de la tesis.

Capítulo 2

El modelo HP

En este capítulo describiremos el modelo que utilizamos para simular el proceso de plegamiento de proteínas, desde sus fundamentos teóricos hasta las distintas representaciones abstractas que se utilizaron para implementarlo. Es un modelo sencillo para simular el plegamiento de proteínas, conocido en la literatura como el *modelo HP*, sugerido por primera vez en 1985 por Ken A. Dill [27]. Se introdujo para estudiar de manera general y simplificada el proceso de plegamiento, en particular para obtener la estructura de mínima energía, la cual es relevante puesto que se supone que corresponde a la estructura funcional de la proteína, la estructura nativa. En este modelo de grano-grueso, se simplifica tanto la estructura de la proteína como las interacciones que intervienen en el proceso de plegamiento. Así, encontramos resultados que nos dan información general –el modelo no pretende dar resultados particulares acerca del proceso de plegamiento de una proteína, debido al nivel de simplificación, sino generalidades del proceso comunes a cualquier secuencia.

Posteriormente se discutirá cómo generar nuevas configuraciones de la proteína, las cuales serán de gran importancia para los algoritmos que utilizaremos, descritos en el siguiente capítulo. Para generar estas nuevas configuraciones, necesitamos definir los movimientos que la proteína puede realizar. Los movimientos consisten en desplazar monómeros según ciertas reglas que describiremos más adelante para obtener nuevas estructuras de la proteína y así explorar su espacio de configuraciones. Utilizamos un conjunto completo de movimientos locales conocidos como *pull moves*, propuestos en la referencia [28]. Este conjunto fue diseñado específicamente para alcanzar configuraciones compactas de la proteína, de las cuales se espera que tengan la menor energía.

2.1. Fundamentos del modelo HP

El modelo HP se basa en la idea de que, durante el proceso de plegamiento, la interacción de mayor magnitud y, por lo tanto, aquélla que domina la transición de una proteína hacia su estructura plegada, es la interacción *hidrofóbica*. Esto ha sido observado para proteínas globulares. La idea proviene de que, al verse inmersos en un medio acuoso, los residuos hidrofóbicos de la proteína buscan agruparse en un núcleo interno protegido por los residuos polares, o hidrofílicos, de manera que este núcleo hidrofóbico tenga la menor superficie de contacto con el medio. Este hecho depende a su vez del volumen que ocupan los demás segmentos del polímero, lo cual restringe la libertad conformacional. La configuración alcanzada en dicho caso será energéticamente más favorable que una configuración que tenga partes hidrofóbicas expuestas al solvente y partes polares aisladas del mismo.

Este hecho es tomado en consideración en tanto que en el modelo HP no tenemos una veintena de aminoácidos diferentes, sino sólamente 2: los hidrofóbicos y los polares. Esta simplificación se sigue de la restricción a un solo tipo de interacción. Puesto que solamente consideramos la interacción hidrofóbica, no nos interesa considerar con mayor detalle a los residuos que forman la secuencia; ningún otro atributo nos es relevante a parte de la polaridad de cada residuo, ya sea químico o físico. Esto no quiere decir que otras interacciones, como podrían ser las interacciones iónicas o de otro tipo, tengan un peso despreciable comparado con el de las interacciones hidrofóbicas, sino que sólo son de menor impacto.

El modelo HP simula el empacamiento hidrofóbico entre residuos al asignar un peso negativo, favorable energéticamente, a cada par de residuos hidrofóbicos vecinos en el espacio que no estén unidos por un enlace covalente, es decir, que no sean vecinos en secuencia. De esta manera, la configuración de mínima energía será aquélla que forme la mayor cantidad de enlaces hidrofóbicos. Suponemos que el estado de mínima energía corresponde al estado nativo de la proteína.

Cabe resaltar que estamos considerando implícitamente que la secuencia bajo estudio se encuentra sumergida en un solvente acuoso. Si no hubiera solvente, es decir, si nuestro sistema se encontrara al vacío, entonces no existiría ningún tipo de interacción debida a la polaridad de los residuos. Decimos que el solvente es implícito ya que no existe interacción entre nuestro sistema y éste. El solvente sólo interviene para permitir que haya interacciones entre residuos hidrofóbicos.

El modelo no toma en cuenta efectos estéricos, puesto que los residuos son modelados por partícula puntuales cuya única propiedad es la de ser hidrofóbico o polar.

2.2. Especificación del modelo HP

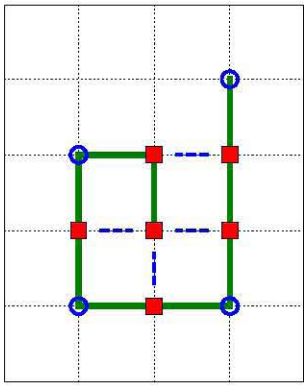


Figura 2.1: Ejemplo de la representación de una proteína en el modelo HP en una red cuadrada. Los monómeros cuadrados son hidrofóbicos y los circulares polares. Los enlaces hidrofóbicos están representados por líneas discontinuas que unen pares de residuos H.

En el modelo HP, una proteína es modelada como una secuencia de longitud l, acomodada en una red bi- o tri-dimensional, de dos tipos de residuos diferentes, *hidrofóbicos* (H) y *polares* (P), como se muestra en la figura 2.1. Los monómeros que conforman el polímero se sitúan en vértices de dicha red. Cada residuo es modelado por una partícula cuya única propiedad es la de ser H o P. En este modelo no nos interesa considerarlos con mayor detalle, puesto que no consideramos ningún otro tipo de interacción más que la interacción hidrofóbica.

Tomamos en cuenta que se forma un enlace hidrofóbico siempre que dos residuos tipo H se encuentren como vecinos en la red, sin que sean vecinos en secuencia, es decir, sin que haya un enlace covalente entre ellos. Un mismo residuo H puede formar varios enlaces con diferentes residuos H vecinos.

En la figura 2.1 podemos observar cómo los enlaces hidrofóbicos se dan entre residuos H, representados por cuadrados, que no están unidos entre sí por un enlace covalente. Se observa también que el monómero que se encuentra encerrado dentro de la estructura forma enlaces con 3 diferentes vecinos hidrofóbicos, que es el máximo posible en una red cuadrada.

En el modelo HP, la energía de una proteína acomodada en una red es el negativo del número de enlaces entre pares de residuos hidrofóbicos que son vecinos en la red, sin ser vecinos en secuencia. La energía es negativa debido a la formación de enlaces,

los cuales dan estabilidad a la estructura. El Hamiltoniano está dado entonces por [29]

$$\mathscr{H} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} P_i P_j \delta\left(r_{ij}, 1\right), \tag{2.1}$$

donde P_k corresponde a la polaridad del k-ésimo monómero, siendo $P_k = 0$ si el monómero es polar y $P_k = 1$ si es hidrofóbico, r_{ij} es la distancia que hay entre los monómeros i y j, medida en unidades de la red, y δ es la delta de Kronecker, que es igual a 1 si $r_{ij} = 1$ y 0 si $r_{ij} \neq 1$. Hay que señalar que la suma no considera aquellos monómeros que son vecinos en secuencia, puesto que estos no pueden formar enlaces hidrofóbicos entre ellos.

El estado nativo de una proteína, dada una secuencia de aminoácidos, está caracterizado por ser el estado donde el hamiltoniano es mínimo. Las configuraciones pertenecientes a un mismo estado se caracterizan por tener el mismo valor de la energía. En principio, esperamos que el número de configuraciones en el estado nativo sea mínimo respecto al numero de configuraciones de cualquier otro estado accesible al sistema, lo cual es natural, ya que esperamos que el estado nativo de la proteína sea aquel con menor degeneración de estados de todos los estados accesibles. Sin embargo, debido a la representación de grano-grueso que consideramos en este modelo, existe una mayor degeneración de estados por energía de lo que se espera en la realidad. Esto quiere decir que obtendremos un mayor número de configuraciones por energía que si consideráramos otras interacciones. Esto no significa un problema para el desarrollo de este trabajo, puesto que nosotros estamos interesados en estudiar generalidades del proceso de plegamiento, y no pretendemos estudiar con todo detalle una secuencia en particular.

2.3. Secuencias con comportamiento similar al de las proteínas

Para que una secuencia tenga un comportamiento similar al de las proteínas, es necesario que el mínimo de energía global sea pronunciado, de manera que el estado nativo de la secuencia sea estable, inclusive incrementando la temperatura. Existen discusiones acerca de si el estado nativo de una proteína es necesariamente el mínimo global de la energía o si podría ser un mínimo local lo suficientemente profundo como para soportar las posibles perturbaciones biológicas a las que estaría sujeto [30].

Las secuencias generadas aleatoriamente no tienen un mínimo de energía global

profundo, sino más bien su paisaje de energía está formado por muchos mínimos locales con profundidades comparables [31] [15].

La diversidad estructural en el estado nativo que esperamos de una secuencia con comportamiento similar al de las proteínas es mínima. Es claro que debido al nivel de grano-grueso del modelo que utilizamos no podemos esperar que el estado nativo corresponda a una única estructura, sin embargo, esperamos que haya una muy marcada diferencia entre el número de estructuras nativas y el número de estructuras no-nativas correspondientes a niveles superiores de energía.

Las secuencias que se utilizan en este trabajo son secuencias diseñadas para tener un comportamiento tipo proteína, es decir, exhibir un paisaje de energía en forma de embudo. Las secuencias generadas aleatoriamente no se comportan de manera similar al de una proteína, debido a que no fueron diseñadas para minimizar la frustración de la secuencia y por lo tanto, su paisaje de energía carece de un mínimo global claramente separado de los demás mínimos.

2.4. Representación de la proteína

En nuestra implementación computacional del modelo, hicimos uso de varias representaciones diferentes de la proteína, cada una con una parte de la información del polímero completo.

El primer paso para modelar una proteína es determinar la secuencia de residuos que queremos estudiar; esto corresponde a la estructura primaria de la proteína. Se genera una cadena de longitud l, que es la longitud de la proteína, de H's y P's. Entre cada par de monómeros vecinos en secuencia, siempre habrá un enlace covalente que los una, sin importar si se trata de dos tipos diferentes de residuo. En este modelo, los enlaces covalentes son irrompibles y son determinados por la secuencia inicial de la proteína. Para esto, guardamos la información de la polaridad de cada monómero en una lista ordenada de tamaño l, es decir, para cada monómero i de la proteína, corresponde la entrada P_i de la lista que nos da la polaridad de dicho monómero.

En la figura 2.2 se muestra un ejemplo de cómo se guarda la información de la polaridad de cada monómero.

Una vez que tenemos la "estructura primaria", introducimos la proteína en la red. Podemos introducirla de manera extendida o en alguna configuración arbitraria. De cualquier modo, la elección de los vértices en donde colocar la proteína es arbitraria. La información de la localización de cada monómero en la red se guarda también en

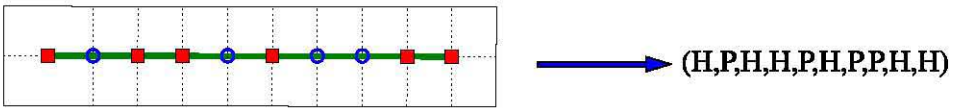


Figura 2.2: Representación de la polaridad de la proteína.

una lista ordenada, análogamente al caso de la polaridad de los residuos.

La manera en que se guarda la información de la posición de cada monómero en la red es ejemplificada en la figura 2.3. Cada monómero puede ocupar un vértice de la red en un tiempo dado con la condición de que la cadena de aminoácidos no puede ni romperse ni encimarse sobre ella misma. Por cada vértice de la red, definimos un par de coordenadas que lo identifiquen. De este modo, la posición de cada monómero está codificada en una lista ordenada de coordenadas de vértices ocupados.

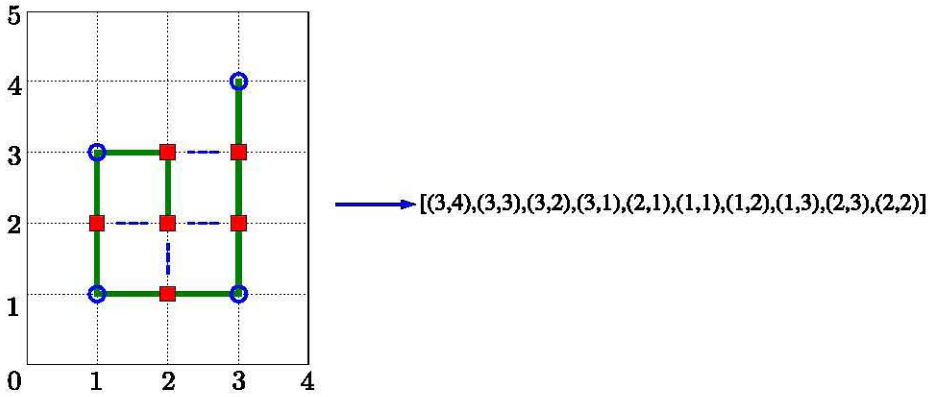


Figura 2.3: Representación de la configuración de la proteína.

Además de la polaridad y la localización de los monómeros de la proteína, necesitamos conocer la estructura global de ésta. Para esto, codificamos cada dirección en la red con un cierto valor; a partir del primer monómero, guardamos la dirección en la que queda el siguiente monómero y así sucesivamente. De esta forma, codificamos de manera unívoca la estructura global de la proteína en la red. La estructura de la proteína se codifica de la manera mostrada en la figura 2.4 para un caso particular. Hay que notar que la elección del primer monómero es completamente arbitraria, sin embargo, una vez elegido el primer monómero, hay que ser siempre consistentes con esto. En el ejemplo que presentamos, el primer monómero es el residuo polar, re-

presentado por un círculo azul. Cabe mencionarse también, que se puede codificar la estructura de manera relativa al primer monómero de modo que una estructura y sus rotaciones tengan un mismo código.

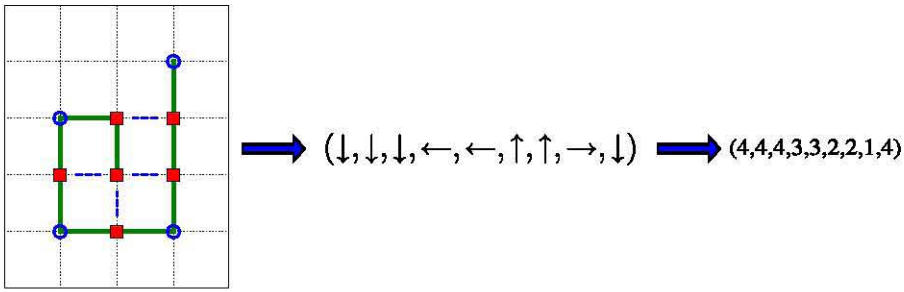


Figura 2.4: Representación de la estructura de la proteína. Elegimos un primer monómero arbitrariamente de entre cualquiera de ambos extremos, luego codificamos las direcciones de los siguientes monómeros respecto al anterior. Cada dirección tiene asociada un número.

Estas tres distintas representaciones incompletas de la proteína nos dan la representación completa al considerarlas en conjunto.

2.5. Redes

Como hemos mencionado anteriormente, se utiliza el modelo HP para modelar proteínas en redes bi- o tri-dimensionales. Por lo tanto, es necesario definir la red, es decir, el espacio donde vive la proteína.

Definimos una *red* como una colección ordenada de vértices idénticos conectados entre sí y acomodados en patrones regulares. En nuestro caso, modelamos el espacio donde vive la proteína con una red; esto implica que el espacio está discretizado, puesto que los monómeros de la proteína sólo pueden encontrarse sobre vértices de la red. Cada uno de estos vértices se conecta con sus vecinos inmediatos en la red. Dependiendo del número de vecinos y del acomodo de los vértices, tendremos diferentes *topologías*. Lo único que define a un vértice en particular en la red es su localización espacial y la característica de estar ocupado o desocupado.

En nuestra implementación del modelo, utilizamos redes periódicas con condicio-

nes de frontera periódicas, de tal modo que si la proteína cruza alguna frontera de la red, aparecerá por la frontera opuesta. En otras palabras, aquellos vértices que se encuentran en la frontera de la red están conectados con los vértices en la frontera opuesta. Es importante definir de esta manera la red, puesto que este hecho tiene implicaciones sobre la completez de los movimientos y, de manera más directa, sobre el espacio disponible para que la proteína se mueva conforme cambia de configuración. Los movimientos que cambian la configuración de la cadena, que explicaremos más adelante, mueven la cadena sobre la red en la que se simula el proceso. Si la simulación se realiza en una red finita, la estructura podría alcanzar las fronteras y querer traspasarlas, lo cual no sería posible y arruinaría la simulación. Otra manera de solucionar el problema de tener un espacio finito determinado por la red, es construir condiciones de frontera especiales tales que cuando la cadena toque alguna de las fronteras, la estructura se regrese automáticamente al centro de la red.

2.5.1. Diferentes geometrías de la red

En nuestro caso, implementamos el modelo tanto en una red cuadrada, como en la figura 2.1, como en una red triangular, como en la figura 2.5.

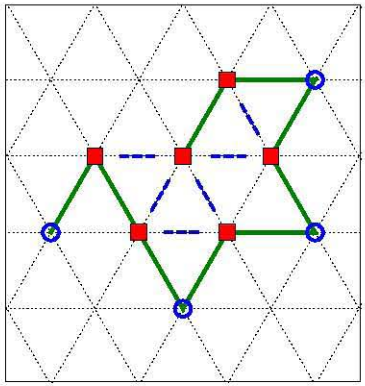


Figura 2.5: Ejemplo de la representación de una proteína en el modelo HP usando una red triangular. Los monómeros cuadrados son hidrofóbicos y los circulares polares. Los enlaces hidrofóbicos están representados por líneas discontinuas que unen pares de residuos.

Una diferencia importante entre utilizar una red triangular y una cuadrada es el número de vecinos por vértice. Para la red cuadrada, cada monómero tiene, a lo más, 3 sitios vecinos, los cuales pueden estar o no ocupados por otros monómeros capaces de

formar un enlace hidrofóbico. Decimos que tiene *a lo más* 3 sitios vecinos que pueden estar o no desocupados puesto que cada vértice de la red cuadrada está conectado a otros 4, sin embargo, al menos uno de estos necesariamente está ocupado por el monómero vecino en secuencia al que estamos considerando. En el caso en el que el monómero que consideramos no sea un extremo, tendrá dos monómeros vecinos en secuencia ocupando dos sitios vecinos en la red, por lo que este tipo de monómero tendrá a lo más 2 sitios vecinos que pueden estar o no desocupados. Otros residuos que no sean vecinos en la secuencia, pueden ocupar los sitios disponibles adyacentes en la red al monómero en cuestión. En el caso de una red triangular, el número de sitios vecinos por residuo, ya sea que estén vacíos u ocupados, es menor o igual a 5. Al tener un mayor número de vecinos, cada monómero hidrofóbico tiene la posibilidad de formar un mayor número de enlaces. Por esta razón, las configuraciones obtenidas usando la red triangular son, en general, más compactas, y por lo tanto de menor energía para un mismo valor de l.

Volviendo a la figura 2.5, nótese que usamos exactamente la misma cadena que aquélla usada en la figura 2.1, es decir, no sólo tienen la misma longitud, sino que la secuencia de monómeros es idéntica. Sin embargo, hay una clara diferencia en el número de enlaces hidrofóbicos formados. Ambas estructuras exhiben el número máximo de enlaces dada esa secuencia en particular.

Observemos que para los monómeros en la red triangular, existe la posibilidad de formar hasta 2 contactos hidrofóbicos más que en el caso cuadrado para cada uno de estos residuos. Por esta razón, esperamos que la energía de una secuencia en una red triangular sea similar al doble de la energía para la misma secuencia en una red cuadrada. Es fácil ver que no será siempre exactamente el doble puesto que la posibilidad de formar contactos no implica necesariamente la formación de los mismos; podría ser que algunos de los contactos que podrían formarse impidieran alcanzar la estructura de mínima energía.

2.6. Complejidad

Aún utilizando el modelo HP en dos dimensiones, el problema de encontrar la configuración de mínima energía es un problema *NP-completo* [32]. Un problema es *NP-completo* si:

- Forma parte del conjunto de problemas *NP* (Tiempo polinomial no-determinístico), es decir, cualquier solución al problema puede ser verificada en tiempo

polinomial;

- Forma parte del conjunto de problemas *NP-complejos*, es decir, cualquier problema *NP* puede ser transformado polinómicamente a éste problema.

Alternativamente, el conjunto de problemas *NP-completos* se define como la intersección entre los problemas *NP* y los problemas *NP-complejos*.

Esto se traduce en la dificultad de tratar de resolver este problema, ya que el tiempo de cómputo crece rápidamente al aumentar el tamaño del sistema. Por esta razón utilizamos algoritmos Monte Carlo, puesto que la aleatoreidad del algoritmo permite tener un tiempo promedio de corrida más rápido.

2.7. Pull moves

Para llevar a cabo las simulaciones del proceso de plegamiento, necesitamos generar secuencias de configuraciones de una misma cadena. Para esto, usamos un conjunto de movimientos, conocidos como *pull moves* [28], aprovechando que fueron diseñados para alcanzar configuraciones compactas del polímero en tiempos cortos. Otras propiedades de los pull moves por las cuales elegimos estos movimientos son la reversibilidad y la completez de los mismos. Decimos que el conjunto es *reversible* siempre que se pueda regresar a la configuración anterior en un solo movimiento, y decimos que el conjunto de movimientos es *completo* si entre cualesquiera dos configuraciones existe una sucesión finita de movimientos que nos lleven de una a otra. Nos interesan estas propiedades puesto que nos permiten asegurar que cualquier configuración del sistema es accesible desde cualquier otra. Otra característica de estos movimientos es que son *locales*, lo cual significa que en un movimiento, es decir, al trasladar monómeros a vértices vecinos vacíos en la red, estos vértices son adyacentes a aquellos en donde se encontraban inicialmente los monómeros. En otras palabras, los elementos desplazados no se desplazan grandes distancias y se intenta desplazar la menor cantidad de elementos por movimiento. Esto es importante puesto que los cambios en la energía, de haberlos, no pueden ser demasiado grandes, ya que no se desplaza un número grande de monómeros. Además, si cambiamos drásticamente la estructura de la proteína en cada movimiento, podríamos estar generando configuraciones de manera aleatoria sin encauzarlas hacia estructuras compactas.

La idea general de los *pull moves* en una red, ya sea cuadrada o triangular, es acomodar los residuos completando unidades básicas de la red, es decir, buscando que

queden en los vértices de celdas tal que se complete la mayor cantidad de cuadrados o triangulos, según sea el caso, y que así haya más contactos hidrofóbicos. Por celda entendemos un elemento básico de la red; en el caso de una red cuadrada, una celda sería la colección de vértices que forme el cuadrado más pequeño y el área dentro de este, mientras que en una red triangular, una celda sería lo análogo pero formando el triángulo equilátero más pequeño posible. Como las redes que usamos son regulares, cada celda es idéntica a cualquier otra. Al acomodar los residuos de la proteína de acuerdo con esta idea, la estructura obtenida será, en general, más compacta que aquéllas obtenidas siguiendo otra estrategia.

Existen dos tipos de movimientos que tenemos que considerar: el movimiento general de cualquier residuo para completar una celda, y el movimiento especial de los residuos en los extremos de la cadena. Estos últimos se incluyen para asegurar la reversibilidad de cualquier movimiento. Hay que subrayar que es necesario considerar estos movimientos en ambos sentidos, es decir, tanto jalando el monómero i hacia el monómero $i+1$ como jalándolo hacia el $i-1$. Nótese que esto no está claramente establecido en la referencia [28], pero es necesario para la reversibilidad de los movimientos. Un ejemplo claro de la necesidad de realizar los movimientos en ambos sentidos se puede observar si se realiza un jalón a uno de los extremos de la proteína extendida hacia cualquier dirección. En este caso, el único movimiento que regresaría a la proteína a su configuración extendida es jalar el otro extremo de la cadena en dirección opuesta al primer movimiento.

2.7.1. Pull moves en una red cuadrada

La idea básica es la de completar celdas cuadradas en la red con un monómero de la cadena en cada vértice. Definimos la configuración de una proteína al tiempo t como $P(t)$. Decimos que una estructura $P(t)$ exhibe un *loop* o *bucle* cuando un subconjunto de sus residuos están acomodados en la red ocupando cada uno de los vértices de una de sus celdas. Los movimientos de residuos no extremos buscan completar cuadrados, es decir, formar *bucles cuadrados* en la estructura de la proteína que ocupen una celda en la red. Cada celda en la red cuadrada está formada por 4 vértices en un acomodo cuadrado. Cualquier par de residuos siempre ocupa dos vértices de una celda en la red. Queremos ocupar los otros dos vértices con otros dos monómeros. Existen varios casos dependiendo de si el par de vértices restantes están ocupados o no.

- En caso de que hubiera otro monómero en otro vértice de esta misma celda, simplemente desplazamos diagonalmente al monómero que se encuentra dia-

gonalmente opuesto al vértice libre. Figura 2.6(b)

- En caso de que ambos vértices restantes estén desocupados, entonces podemos desplazar uno de los dos monómeros, que ya formaban parte de la celda, a alguno de los vértices vacíos, es decir, desplazar diagonalmente uno de los dos monómeros, mientras dejamos al otro monómero en su lugar. Figura 2.6(c).

Una vez realizado un movimiento, tenemos que asegurarnos de que la proteína quede conectada, para lo cual jalamos los monómeros restantes uno por uno hasta obtener una configuración conexa. De este hecho es de donde proviene el nombre de los movimientos.

Los movimientos de los residuos extremos son importantes para deshacer bucles. Esto es necesario puesto que hay ocasiones en las que se forman contactos que no permiten la formación de otros contactos, es decir, la estructura se queda atrapada en un estado metaestable del cual la única salida es mediante la desaparición de algunos contactos hidrofóbicos. Los movimientos extremos desplazan algún monómero extremo hacia cualquier vértice desocupado a dos espacios de distancia de su posición actual. Estos movimientos consisten en desplazar segmentos dos unidades de distancia en la red, puesto que deshacen bucles, y para deshacer un bucle cuadrado, necesitamos desplazar dos monómeros. Es claro que estos movimientos están restringidos por la estructura de la proteína, ya que no podemos permitir que la cadena se cruce a si misma.

Al realizar un pull move sobre la configuración $P(t)$ en el paso t, se genera otra configuración válida $P(t+1)$ en el tiempo siguiente. El conjunto M de todos los pull moves se forma de la unión de los movimientos en ambos sentidos, dado que ambos son necesarios para cumplir la condición de reversibilidad. La descripción se realizará en un solo sentido, puesto que en el otro sentido es completamente análogo –sólo hace falta considerar el monómero $i-1$ en vez del $i+1$, y así succesivamente. El conjunto de pull moves M es reversible, puesto que para todo elemento de M que lleve $P(t)$ a una configuración $P(t+1)$, existe al menos otro elemento de M que lleve $P(t') = P(t+1)$ a $P(t'+1) = P(t)$. Decimos que el conjunto de movimientos M es *completo* si entre cualesquiera dos configuraciones $P_1(t)$ y $P_2(t)$ existe una sucesión finita de elementos de M que nos lleve de una a otra. Los pull moves que describiremos a continuación son completos y por lo tanto reversibles; la justificación de esto se presenta más adelante.

Supongamos que queremos mover el monómero i para generar una nueva configuración de la cadena. Para que el movimiento sea posible, debe de haber un espacio

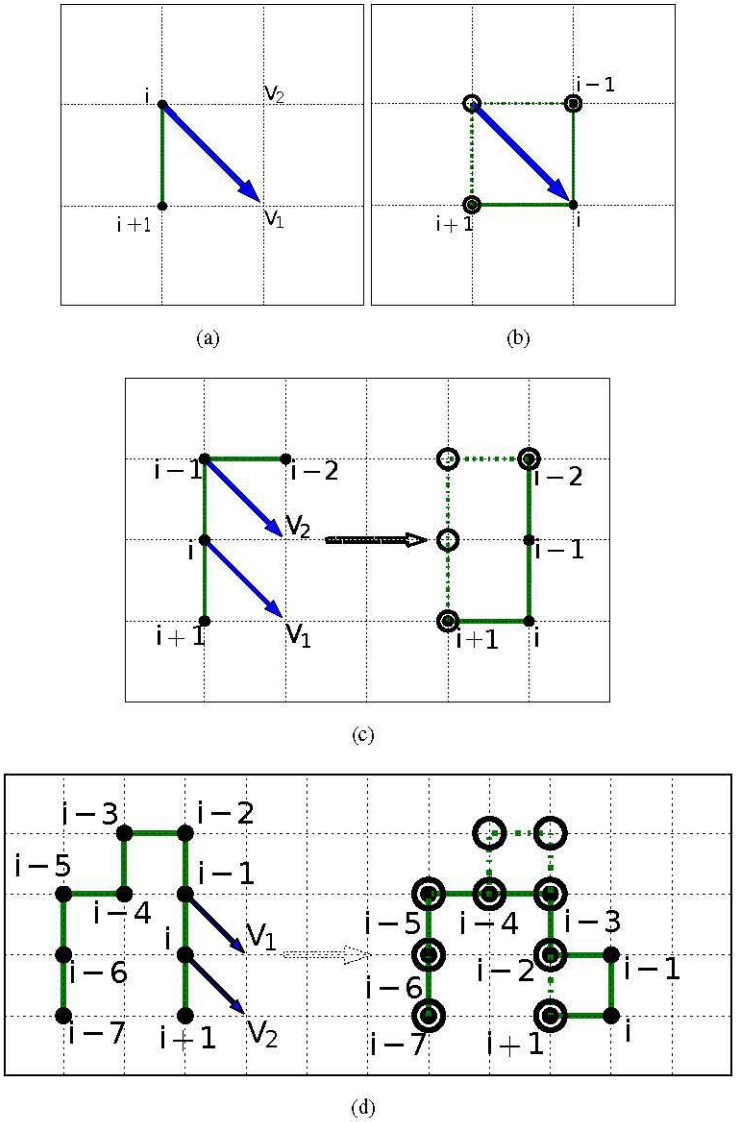


Figura 2.6: Descripción de los movimientos utilizados, los *pull moves*. Aquí los círculos vacíos representan espacios libres en la red que estaban ocupados por los monómeros en el tiempo anterior, mientras que los círculos negros unidos por líneas sólidas representan la nueva estructura de la proteína. Las flechas representan los movimientos que se realizan para comenzar el pull move.

vacío adyacente al monómero $i+1$ y diagonalmente adyacente al monómero i, que llamaremos V_1. Ambas posiciones de los monómeros i e $i+1$, llamémoslas $X_i(t)$ y $X_{i+1}(t)$, junto con el espacio vacío V_1 forman tres vértices de una celda cuadrada en la red; ver la figura 2.6(a). El vértice sobrante, que llamaremos V_2, debe estar libre, $V_2 \neq X_k(t), \forall k \in \{1,...,n\}$, si la proteína consta de n monómeros, o debe de contener al monómero $i-1$, $X_{i-1}(t) = V_2$, para que el movimiento ocurra.

Si la cuarta esquina, V_2, contiene al monómero $i-1$, $X_{i-1}(t) = V_2$, entonces el movimiento consiste solamente en mover al monómero i a la posición libre V_1, es decir $X_i(t+1) = V_1$. En este caso no es necesario realizar ningún movimiento adicional para mantener unida la cadena. Este caso está ejemplificado en la figura 2.6(b).

Por otro lado, si la cuarta esquina V_2 está libre, entonces necesitamos mover hacia este lugar el monómero $i-1$, $X_{i-1}(t+1) = V_2$, completando así el cuadrado. Si obtenemos una configuración donde la cadena no se haya roto, como en la figura 2.6(c), entonces el movimiento se termina. Pero si la cadena no es conexa, entonces tenemos que ir jalando todos los monómeros anteriores, uno por uno, hasta obtener una configuración válida de la cadena, la cual puede alcanzarse antes de llegar al último monómero. La manera de jalarlos es trasladar el monómero j a la posición donde se encontraba anteriormente el monómero $j+2$, para todas las $j \leq i-2$ hasta alcanzar una configuración válida, como en la figura 2.6(d). Esto es, $X_j(t+1) = X_{j+2}(t), \forall j \in \{i-2,...,1\}$ hasta obtener una configuración válida. Si se alcanza una configuración válida $P(t+1)$ antes de mover el monómero del extremo, entonces detenemos el movimiento, puesto que, como mencionamos anteriormente, nos interesa que los movimientos sean lo más locales posibles, es decir que modifiquen la posición del menor número de monómeros alrededor del monómero que queremos mover.

Para los movimientos de los extremos de la cadena, consideremos dos vértices libres adyacentes V_1 y V_2, tal que V_1 sea adyacente al monómero n. Podemos mover al monómero n al sitio V_2 y al monómero $n-1$ al sitio V_1, y luego jalar los demás monómeros de la cadena del mismo modo que con los otros movimientos, hasta obtener una configuración válida. Durante el desarrollo del presente trabajo, se encontró una restricción a los movimientos de los extremos tal como se plantean en la referencia [28], de modo que se conserve la reversibilidad. Esta restricción que imponemos consiste en no permitir el movimiento de un monómero extremo a sitios que sean directamente adyacentes a la cadena. Más adelante explicaremos esto con un ejemplo.

Consideremos el siguiente ejemplo del movimiento de un extremo. En la figura 2.7, vemos señalados con una $\times$ los espacios adyacentes al monómero n, que son los espacios que podríamos elegir como nuestro V_1. Los espacios adyacentes a estos espacios están señalados por el símbolo +; por cada elección de V_1, tenemos a lo más tres distintas elecciones para V_2. Para dos de los vértices posibles V_1, existen sólo 2 vértices posibles V_2, debido a la restricción que más adelante explicaremos. En este ejemplo, elegimos como sitio V_1 el espacio adyacente a la izquierda del monómero n y como sitio V_2 el espacio adyacente debajo del sitio V_1. Realizando el movimiento tal como fue descrito anteriormente, obtenemos una configuración válida $P(t+1)$ para la cadena sin necesidad de mover todos los monómeros hasta el final. Al mover el monómero $n-5$, el monómero $n-6$ permanece en su lugar, pero permite que la cadena ya sea conexa de nuevo.

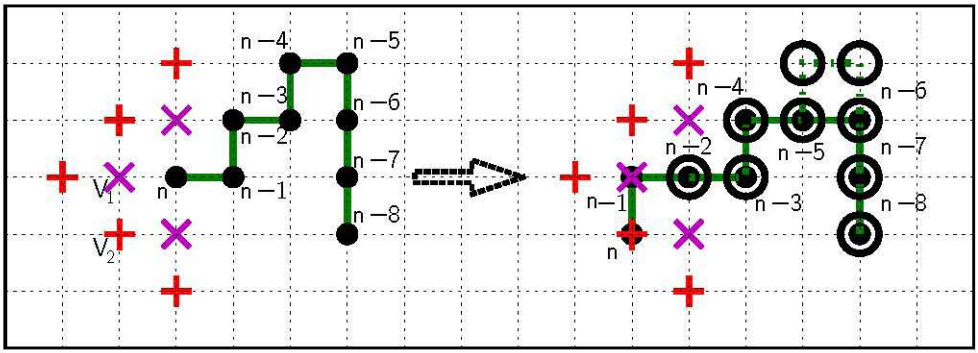


Figura 2.7: Ejemplo del movimiento de los extremos de la cadena.

Anteriormente mencionamos que los movimientos extremos no pueden ser tales que el vértice V_2 sea adyacente a algún monómero de la cadena, puesto que esto hace que ciertos movimientos no sean reversibles. Explicaremos esto mediante un ejemplo. Considérese la configuración $P(t)$ de la izquierda en la figura 2.8 en donde se indica, mediante una flecha roja, un movimiento del extremo en donde el vértice V_2 al cual se moverá el monómero 1 es adyacente a la cadena. Si realizamos este movimiento, obtenemos la configuración $P(t+1)$, representada en la parte media de la figura 2.8. Para regresar a la configuración $P(t+2) = P(t)$, necesitamos recrear el bucle cuadrado original de la configuración $P(t)$, moviendo al monómero i e $i-1$, y recorriendo el monómero j, con $j \leq i-2$, hasta obtener una configuración válida. Sin embargo, el monómero 1 no regresa a su posición original, puesto que no es necesario desplazarlo para obtener una configuración válida de la cadena. Esta configuración, representada en la parte derecha de la figura 2.8, no coincide con la configuración original $P(t)$ que

queríamos recuperar.

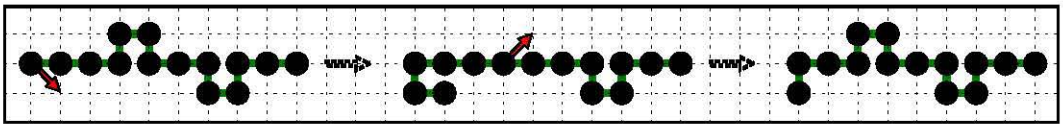


Figura 2.8: Ejemplo del tipo de movimiento de los extremos que rompe con la reversibilidad de los *pull moves*.

Mediante el ejemplo anterior se observa la necesidad de restringir los movimientos de los monómeros extremos, prohibiendo aquellos movimientos que lleven al monómero extremo a algún sitio adyacente a la cadena, para conservar el carácter reversible de los movimientos. En la referencia [28], no hay mención de dicha restricción en los movimientos de los extremos, por lo que fue necesario darse cuenta de este hecho para el desarrollo del presente trabajo.

2.7.2. Pull moves en una red triangular

Este tipo de movimientos puede ser generalizado también a otras geometrías, en particular a una red triangular. Hay que conservar en mente la idea básica, que es la de completar celdas en la red. En este caso, las celdas de la red están formadas por 3 vértices en acomodo de triángulo equilátero. Para completar una celda triangular donde dos monómeros viven en dos de sus vértices, sólo es necesario trasladar otro monómero al sitio vacío de la celda. Notemos que en el caso de la red cuadrada, movíamos dos monómeros a dos sitios disponibles de una misma celda, siempre que pudiéramos. Además, podemos darnos cuenta de que para deshacer un bucle en una celda triangular, sólo hace falta recorrer el monómero un lugar, a diferencia de los dos lugares que hacían falta en la red cuadrada. A continuación se presenta la generalización de los pull moves para una red triangular que se ideó para esta tesis siguiendo las ideas generales expuestas anteriormente. Estos movimientos fueron propuestos por el autor de la tesis, y luego se encontró que coincidían con la propuesta de la referencia [33].

En una red triangular, los movimientos se simplifican por las razónes expuestas anteriormente; no son necesarios movimientos que se desplacen más que una unidad de longitud en la red, lo cual hace que estos sean más fáciles de seguir. No necesitamos movimientos que desplacen más de un vértice a algún monómero, puesto que los bucles formados en una red triangular involucran únicamente a 3 monómeros, y estos

bucles pueden deshacerse desplazando algún monómero un solo espacio. La idea sigue siendo la misma: intentamos que los monómeros se acomoden en celdas básicas, ahora triangulares. De nuevo, explicaremos los movimientos en un solo sentido; ya que los movimientos en el sentido contrario son simétricos; sin embargo, recordemos que necesitamos de todos estos movimientos para que el conjunto sea reversible, y, por lo tanto, completo. Tal como en el caso cuadrado, existen dos tipos de movimientos a considerar.

Para entender el movimiento de cualquier monómero, excepto los extremos, supongamos que nos interesa desplazar el monómero i partiendo de una configuración $P(t)$. Como la idea principal de los movimientos en este caso es la de colocar los monómeros en celdas triangulares, y los monómeros i e $i+1$ ya ocupan dos vértices de una de estas celdas, en particular $X_i(t)$ y $X_{i+1}(t)$, este movimiento es posible siempre y cuando haya vértices vacíos adyacentes al monómero $i+1$ que también sean adyacentes al monómero i para la configuración $P(t)$. Dicho de otro modo, necesitamos que el vértice restante de esta celda triangular, denotémoslo como V, esté vacío para poder trasladar el monómero i a este vértice, $X_i(t+1) = V$, obteniendo una configuración diferente $P(t+1)$. Tal como en el caso de la red cuadrada, existen dos posibilidades al trasladar el monómero i al vértice libre de la celda en cuestión, dependiendo de $X_{i-1}(t)$, la localización del monómero $i-1$.

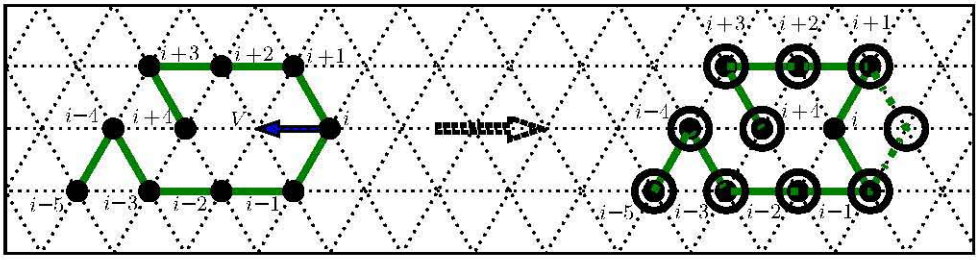


Figura 2.9: Ejemplo de un pull move en una red triangular en donde el vértice libre V es adyacente al monómero $i-1$.

Si el monómero $i-1$ es adyacente a V en la configuración $P(t)$, entonces el movimiento consiste solamente en desplazar el monómero i a V, es decir, $X_i(t+1) = V$. Al realizar este único desplazamiento, la configuración $P(t+1)$ ya es una configuración válida de la proteína, como se puede apreciar en la figura 2.9.

Por otro lado, si el monómero $i-1$ no es adyacente a V en la configuración $P(t)$, entonces el realizar el desplazamiento del monómero i a V, $X_i(t+1) = V$ no generará una configuración válida de la proteína, puesto que la cadena será disconexa.

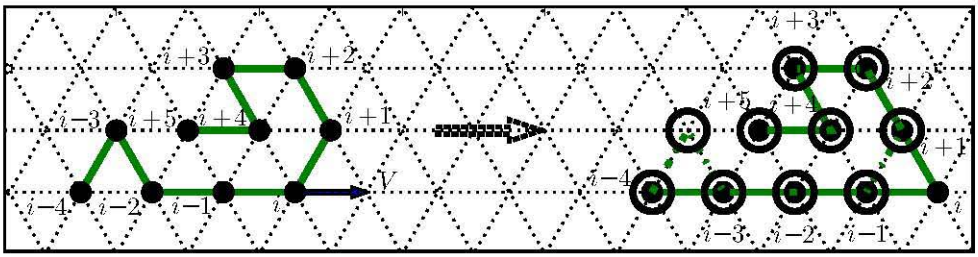


Figura 2.10: Ejemplo de un pull move en una red triangular en donde el vértice libre V no es adyacente al monómero $i-1$.

En este caso, necesitamos jalar el monómero j al lugar del monómero $j+1$ en la configuración $P(t)$, $X_j(t+1) = X_{j+1}(t)$ para todas las $j \leq i-1$, hasta obtener una configuración válida de la proteína $P(t+1)$. Podemos ver un ejemplo de este caso en la figura 2.10.

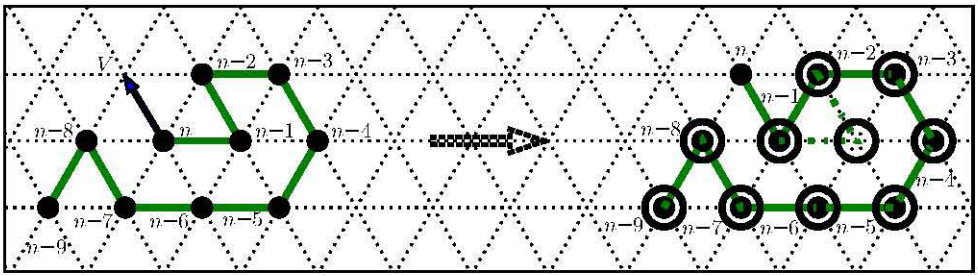


Figura 2.11: Ejemplo de un pull move en una red triangular para el mónomero extremo n.

Ahora, si queremos mover un monómero que se encuentra en el extremo de la cadena, digamos el monómero n, entonces podemos desplazarlo hacia cualquier sitio disponible sobre la red V, siempre que V sea adyacente al monómero n, pero que no sea adyacente al monómero $n-1$. La utilidad de los movimientos de los extremos es que ayudan a deshacer bucles, en este caso triangulares; sin embargo, si desplazáramos el monómero n a un vértice vacío V que fuera adyacente tanto a n como a $n-1$, entonces este movimiento consistiría en ese único desplazamiento y no modificaría la posición de ningún otro monómero en la red; por lo tanto, este movimiento no serviría para deshacer bucles. Una vez que se haya desplazado el monómero n a un vértice vacío V al tiempo t, necesitamos jalar el resto de la cadena de la misma manera de siempre, esto es, desplazamos el mónomero j al lugar del monómero $j+1$ en la configuración $P(t)$, $X_j(t+1) = X_{j+1}(t)$ para $j \leq i-1$, hasta obtener una configura-

ción válida de la proteína $P(t+1)$. En la figura 2.11, se muestra un ejemplo de este tipo de movimientos en donde se obtiene una configuración válida antes de alcanzar el otro extremo de la proteína.

2.8. Reversibilidad y completez de los movimientos

A continuación se muestra que el conjunto de los *pull moves* son reversibles y completos, lo cual se prueba en la referencia [28]. Esto se hace para el caso de una red cuadrada; para el caso de la red triangular, las demostraciones siguen la misma línea de pensamiento, con las sutiles diferencias que conlleva el cambio de geometría. También pueden revisarse los detalles de las pruebas para la red triangular en la referencia [33].

Reversibilidad Comencemos por mostrar la reversibilidad de los pull moves. Procederemos analizando diferentes casos. Los movimientos que desplazan uno o dos monómeros son claramente reversibles. En el caso en el que se mueve un sólo monómero siempre es posible mediante otro pull move regresarlo a su posición original, como en el caso de la figura 2.6(b). Si desplazamos dos monómeros, también es siempre posible regresar, puesto que existe siempre la posibilidad de jalar algún extremo de la cadena dos lugares en la dirección correcta para volver a la estructura original. Al desplazar tres o más monómeros de lugar en la red, la vuelta a la estructura original no es igual de fácil.

Consideremos, sin pérdida de generalidad, que desplazamos los vértices comprendidos entre el vértice i y el vértice j, ambos incluidos; supongamos además que $j < i$. En el caso en el que $j \neq 1$, el movimiento se detuvo antes de alcanzar el monómero extremo. Sabemos que si el movimiento se detuvo, entonces el monómero j fue el último en moverse; por lo tanto, sabemos también que existía un bucle cuadrado formado por los monómeros $j-1$, j, $j+1$ y $j+2$, ya que para que un movimiento se detenga antes de llegar al extremo de la cadena, es necesario que encuentre un bucle cuadrado. Para regresar a la configuración original es necesario solamente recrear el bucle cuadrado que contiene al monómero i, regresando los monómeros j y $j+1$ a sus posiciones originales y jalar los demás monómeros en dirección contraria. Este movimiento desplazará los monómeros siguientes hasta llegar al i, en donde se forma un bucle cuadrado a causa del movimiento de los monómeros i e $i-1$. Notemos que es imposible que el movimiento se detenga antes de llegar al monómero i, debido a que no existe, ni existía en la configuración original, ningún bucle cuadrado entre los monómeros j e i.

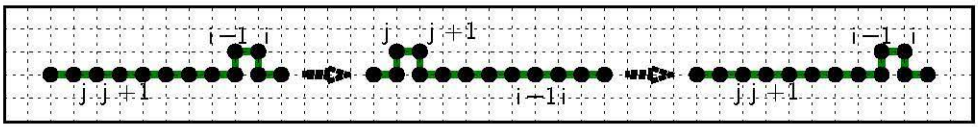


Figura 2.12: Ejemplo de la reversibilidad de los movimientos.

Podemos entender esto también de la siguiente manera. Al tiempo t, en la configuración $P(t)$, los vértices k y $k+3$ con $j < k < i-2$ no pueden ser adyacentes, puesto que de serlo, el movimiento se habría detenido antes de alcanzar al monómero j, ya que los monómeros k, $k+1$, $k+2$ y $k+3$ formarían un bucle cuadrado. Debido a que el monómero j fue el último en desplazarse, lo vértices $\big(x_{j-1}(t), y_{j-1}(t)\big)$, $\big(x_j(t), y_j(t)\big)$, $\big(x_{j+1}(t), y_{j+1}(t)\big)$, $\big(x_{j+2}(t), y_{j+2}(t)\big)$ forman los cuatro vértices de un bucle cuadrado. Podemos obtener la configuración original, $P(t-1)$, regresando los monómeros j y $j+1$ a sus posiciones originales y jalando el resto de la cadena en dirección contraria.

Supongamos ahora, buscando obtener una contradicción, que el último monómero en desplazarse en el movimiento de regreso a la configuración original $P(t-1)$ fue el monómero m, con $j < m < i-2$. Esto significaría que los vértices $m-2$ y $m+1$ eran adyacentes después del primer movimiento, es decir, en la configuración $P(t)$. Sin embargo, esto implica que los monómeros m y $m+3$ eran adyacentes en la configuración original $P(t-1)$, lo cual contradice el hecho de que el movimiento se haya detenido hasta alcanzar el monómero j. Es fácil entender la implicación de que los monómeros m y $m+3$ fueran adyacentes en la configuración $P(t-1)$ recordando que al jalar los monómeros lo que hacemos es desplazar al monómero q al lugar previamente ocupado por el monómero $q \pm 2$, según sea el sentido en el que hagamos el movimiento. Es posible mostrar que el movimiento tampoco puede detenerse en los monómeros $i-2$ e $i-1$ analizando con detalle estos casos.

Si el movimiento se detuviera en los monómeros $i-2$ o $i-1$, entonces la proteína no terminaría en una configuración válida, ya que quedaría desconectada: en la configuración original $P(t-1)$ existe un bucle cuadrado formado por los monómeros $i-2$, $i-1$, i e $i+1$, entonces, al regresar a esta configuración partiendo de la configuración $P(t)$, el monómero $i-2$ en la configuración $P(t+1) = P(t-1)$ ocupará el lugar del monómero i en la configuración $P(t)$, pero si el monómero $i-2$ es el último en moverse, tendríamos que ambos monómeros, $i-2$ e i, quedarían encimados en una misma posición, a menos de que existiera otro bucle cuadrado formado por los monómeros $i-4$ a $i-1$, lo cual ya vimos que contradice nuestras suposiciones

originales. Con el monómero $i-1$, podemos ver que al regresar a la posición original $P(t+1) = P(t-1)$, si este monómero es el último en desplazarse, entonces la cadena quedaría fragmentada, habría un hueco entre el monómero $i-1$ y el i.

Para el caso en el que $k = 1$, simplemente hay que considerar que el otro extremo de la proteína puede desplazarse a lo largo de cualquier trayectoria libre de longitud 2 adyacente a él.

Completez Para mostrar que el conjunto de los pull moves es completo, es decir, que cualesquiera dos configuraciones están conectadas por una sucesión finita de movimientos pertenecientes al conjunto de pull moves M, basta mostrar que podemos llevar una configuración cualquiera P_1 a una línea horizontal y de allí llevarla a otra configuración cualquiera P_2. La complicación en mostrar que siempre se puede llegar a una configuración extendida a partir de cualquier otra configuración surge de que los extremos de la cadena pueden encontrarse metidos dentro de la estructura misma. La idea para alcanzar una configuración lineal es liberar un extremo de la proteína, ya que ambos extremos pueden encontrarse rodeados por el resto de la cadena, y desplazarlo, jalando la cadena con él, hasta obtener una configuración lineal. Nos satisface sólo llevar a P a una línea, ya que gracias a que los movimientos son reversibles, de la proteína extendida podríamos ir a cualquier otra configuración P'.

Denotemos por $I(t)$ la línea vertical izquierda de la red sobre la cual yace el extremo izquierdo de la estructura de la proteína, $P(t)$, al tiempo t. Visto de otro modo, el enlace covalente en dirección vertical entre dos residuos de la proteína que se encuentre más hacia la izquierda definirá lo que llamamos $I(t)$. Análogamente, llamamos $D(t)$ a la línea vertical de la red que contenga el extremo derecho de la estructura $P(t)$ al tiempo t. Cabe mencionar que en $I(t)$ o en $D(t)$ no necesariamente están los monómeros extremos de la proteína, como es el caso de la figura 2.13; de hecho, los extremos de la proteína pueden encontrarse dentro de la región comprendida entre $I(t)$ y $D(t)$ o fuera de ella. Notemos que también es posible que $I(t) = D(t)$, es decir, que ambos extremos de la estructura compartan la coordenada x en la red, tal como en el caso mostrado en la figura 2.13. Si $P(t)$ no tiene extremos $I(t)$ ni $D(t)$, entonces la proteína ya está completamente extendida y orientada horizontalmente.

Sólo nos interesa fijarnos en uno de los dos monómeros extremos de la proteína, cualquiera que elijamos, puesto que podemos extender la estructura jalando cualquiera de los dos. Supongamos que uno de los vértices extremos de $P(t)$ yace en un extremo de la estructura o fuera de la región comprendida entre ambos extremos. Por ejemplo, supongamos que el primer vértice yace sobre $I(t)$ o a la izquierda de éste.

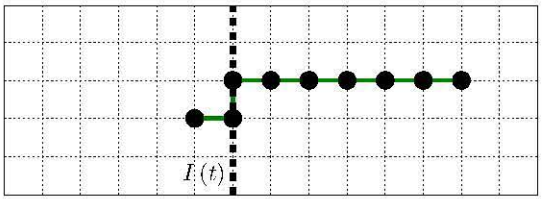


Figura 2.13: Ejemplo de una configuración válida en la que el monómero extremo no se encuentra sobre el extremo estructural $I(t) = D(t)$.

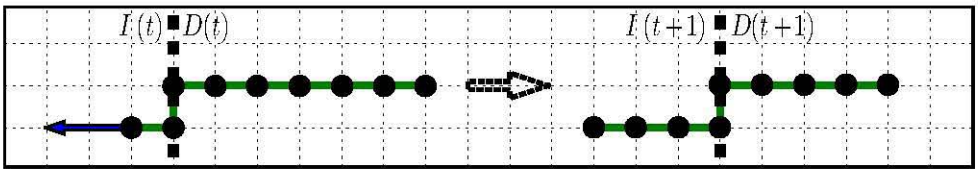


Figura 2.14: Estrategia para obtener una configuración extendida horizontal en el caso en el que el monómero extremo se encuentre fuera de la región comprendida entre $I(t)$ y $D(t)$.

Si el monómero 1 se encuentra a la izquierda de $I(t)$, entonces éste está conectado a $I(t)$ por una subsección horizontal de la proteína; como ejemplo, consideremos la figura 2.14. En ambos casos, los espacios en la red a la izquierda del monómero 1 están necesariamente desocupados. Utilizando un sucesión de pull moves que desplacen al monómero 1 dos espacios a la izquierda en la red, obtendremos eventualmente la configuración completamente extendida horizontalmente de la proteína, $P(t')$, para un tiempo $t' > t$. El caso en el que un monómero extremo de $P(t)$ se encuentre sobre $D(t)$ o a la derecha de éste es completamente análogo. Es importante recordar en este punto que en nuestra implementación, la red es periódica, por lo que no tenemos ningún problema relacionado a la completez de los movimientos debido al tamaño de la red.

Ahora, si el monómero extremo de $P(t)$ que elegimos, supongamos que es el primero, se encuentra en la región comprendida entre $I(t)$ y $D(t)$, tal como en la figura 2.15, la estrategia es la siguiente: Comenzando desde el monómero 1, seguimos la estructura de la proteína $P(t)$ hasta encontrar el primer enlace vertical que determine $I(t)$, $D(t)$ o ambos simultáneamente. Llamemos a los monómeros que comparten este enlace en $P(t)$ los monómeros i e $i+1$; si comenzaramos desde el último monómero n, llamaríamos a estos monómeros i e $i-1$. Podemos suponer, sin perdida de generalidad, que estos monómeros determinan $I(t)$. Sabemos que los espacios en la red a

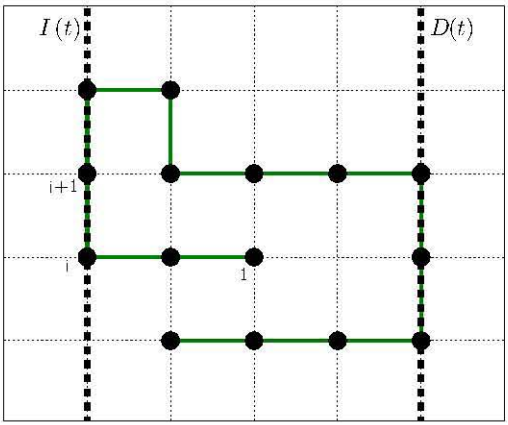


Figura 2.15: Caso en el que el monómero extremo se encuentra dentro de la región comprendida entre $I(t)$ y $D(t)$.

la izquierda del monómero i e $i+1$ están, necesariamente, vacíos. Entonces podemos aplicarle un pull move a $P(t)$ tal que desplacemos al monómero i a la izquierda del $i+1$. Este movimiento se continuaría para mantener unida la cadena, posiblemente, aunque no necesariamente, hasta el monómero 1. En la nueva configuración $P(t+1)$, el extremo $I(t+1)$, formado por los monómeros i e $i-1$, yace una unidad a la izquierda de $I(t)$. Repetimos el proceso moviendo el monómero $i-1$ a la izquierda del i en el siguiente paso, y así succesivamente hasta que el monómero 1 se encuentre en $I(t')$. De hecho, el proceso se realiza i veces para llegar al caso en el que el monómero extremo coincida con el extremo estructural de la proteína $P(t+i)$. Cuando esto suceda, simplemente realizamos el proceso descrito en el párrafo anterior hasta obtener una configuración extendida horizontal. Un ejemplo de este proceso se muestra en la figura 2.16 para la configuración exhibida en la figura 2.15. Notemos que la elección de llegar a configuraciones extendidas orientadas horizontalmente fue arbitraria, podríamos haber elegido trabajar con configuraciones extendidades verticalmente, para las cuales el procedimiento es completamente análogo.

Hemos mostrado que los pull moves forman un conjunto completo de movimientos reversibles. Nótese que hemos mostrado la reversibilidad y completez de los movimientos descritos para la red cuadrada; no hemos considerado explícitamente el caso de la red triangular en esta sección.

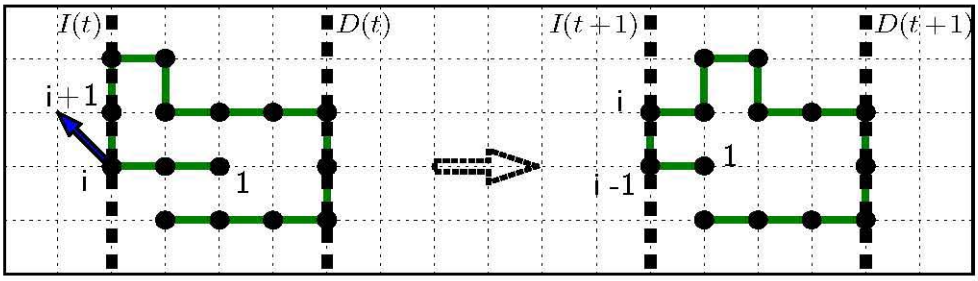


Figura 2.16: Ejemplo de movimiento a realizar repetidas veces para obtener una configuración extendida horizontal en el caso en el que el monómero extremo se encuentre dentro de la región comprendida entre *I(t)* y *D(t)*.

Capítulo 3

Métodos Monte Carlo

Una vez habiendo implementado el modelo y los movimientos, debemos definir el método a través del cual generaremos la evolución temporal del sistema y a partir del cual obtendremos resultados. Los métodos que utilizamos son ambos métodos *Monte Carlo*: el algoritmo de *Metropolis* [34] y el algoritmo *Wang–Landau* [35].

Al final del capítulo, se explica el funcionamiento de las simulaciones realizadas; incorporando el modelo, los movimientos y los métodos. La intención es dar una idea de cómo interactúan todos estos elementos para dar lugar a las simulaciones.

3.1. Métodos Monte Carlo

Los métodos Monte Carlo son probablemente los más usados para hacer simulaciones en física estadística. Estos métodos se basan en simular las fluctuaciones térmicas aleatorias de un sistema que cambia de estado a través de un cierto proceso estocástico. Para esto, se generan aleatoriamente largas secuencias de configuraciones del sistema, de tal modo que después de un cierto tiempo, las nuevas configuraciones se generan de acuerdo a una cierta distribución de equilibrio que describe adecuadamente al sistema. A partir de esta distribución, se pueden obtener otros resultados, pues conociendo el comportamiento microscópico del sistema podemos inferir el comportamiento macroscópico del mismo a través de la física estadística.

La idea básica de las simulaciones Monte Carlo, como ya se mencionó, es simular las fluctuaciones térmicas aleatorias de un sistema que transita entre sus diferentes mi-

croestados a lo largo de algún proceso. Las simulaciones que utilizan métodos Monte Carlo simulan el proceso directamente, es decir, creando un modelo del sistema bajo estudio en la computadora, haciéndolo visitar sus diferentes microestados de tal forma que la probabilidad de estar en un estado σ al tiempo t en la simulación coincida con la probabilidad de ocurrencia de dicho estado para el sistema real. Para lograr estas transiciones entre estados con las probabilidades adecuadas, necesitamos definir la *dinámica* de la simulación. Nos referimos por *dinámica* a las reglas de transición entre estados que definimos para que cada estado en la simulación aparezca con la probabilidad correspondiente en el sistema real. Existen varias maneras de obtener este resultado; la estrategia básica es la de elegir las probabilidades de transición entre estados $p_{\sigma\to\tau}$ de tal modo que la solución de equilibrio sea la distribución de Boltzmann. Utilizamos estas tasas de transición para elegir los estados visitados por nuestro sistema durante la simulación. De estos estados visitados, realizamos estimados de los observables que nos interesen. Un problema es que no conocemos la función de partición:

$$Z(\beta) = \sum_{\sigma} e^{-\beta \mathscr{H}(\sigma)}, \tag{3.1}$$

donde $\mathscr{H}(\sigma)$ es el hamiltoniano del sistema en el estado σ, y β es la temperatura inversa $\frac{1}{kT}$.
La ventaja de estos métodos es que podemos obtener un estimado bastante bueno de observables físicos sin necesidad de visitar cada microestado del sistema, sino sólo una fracción de éstos. Para más detalle de métodos Monte Carlo y sus aplicaciones se refiere a las referencias [36, 37].

En general, las simulaciones Monte Carlo se utilizan para calcular valores esperados de algún observable de interés, como podría ser la energía interna de un gas. Para un observable Q, su valor esperado se define como:

$$\langle Q \rangle = \frac{\sum_{\mu} Q_{\mu} e^{-\beta E_{\mu}}}{\sum_{\mu} e^{-\beta E_{\mu}}}, \tag{3.2}$$

donde Q_{μ} es el valor del observable Q en el estado μ con energía E_{μ}. La ecuación (3.2) no es otra cosa más que la suma del valor del observable Q sobre todos los estados μ del sistema, pesados con su respectiva probabilidad de Boltzmann. Sin embargo, debido a la gran cantidad de microestados que exhibe un sistema no trivial, este valor se puede calcular exactamente sólo para sistemas pequeños. Para sistemas grandes, este valor puede ser estimado realizando el promedio sobre un subconjunto de estados del sistema. Los métodos Monte Carlo escogen los estados de este subconjunto al

azar según una distribución de probabilidad p_μ que podemos escoger. Supongamos que escogemos un subconjunto de N microestados del sistema $\mu_1, ..., \mu_N$. La mejor aproximación que podemos tener del valor esperado del observable Q será [36]:

$$Q_N = \frac{\sum_{i=1}^{N} Q_{\mu_i} p_{\mu_i}^{-1} e^{-\beta E_{\mu_i}}}{\sum_{i=1}^{N} p_{\mu_i}^{-1} e^{-\beta E_{\mu_i}}}. \tag{3.3}$$

La cantidad Q_N tiende al valor $\langle Q \rangle$ en el límite cuando $N \to \infty$.

Queremos entonces una manera de escoger los N estados sobre los que sumamos de tal modo que Q_N sea un buen estimado de $\langle Q \rangle$. Para esto, basta con escoger adecuadamente la distribución de probabilidad p_μ. Podríamos pensar que una buena opción para p_μ sería visitar todos los estados con la misma probabilidad. Sin embargo, esto suele no ser una buena idea, ya que los estados que se visitan son significativamente pocos y las sumas de la ecuación (3.2) suelen tener contribuciones grandes de unos pocos estados y contribuciones casi nulas de una enorme cantidad de estados. Especialmente a bajas temperaturas, los sistemas suelen no disponer de suficiente energía térmica para encontrarse en estados más elevados en energía, por lo que estos sistemas permanecen en unos cuantos estados, o a veces en uno solo: el estado base. La probabilidad de elegir el estado base con una distribución de probabilidad plana, es decir, todos los estados son equiprobables, es prácticamente nula. Por el contrario, si supiéramos qué estados realizan las mayores contribuciones a las sumas en la ecuación (3.2), entonces podríamos elegir nuestro conjunto de N estados a visitar de acuerdo con esto y obtener una buena aproximación al valor de $\langle Q \rangle$. Ésta es la esencia de los métodos Monte Carlo, y la técnica para elegir aquellos estados que contribuyen mayormente a $\langle Q \rangle$ se llama *muestreo por importancia*. La forma más común de muestreo por importancia es aquélla en la que se visitan los estados del sistema de acuerdo a la distribución de probabilidades de Boltzmann, esto es, la probabilidad de visitar algún estado en particular es proporcional a su peso de Boltzmann. De esta manera, la ecuación (3.3) se vuelve:

$$Q_N = \frac{1}{N} \sum_{i=1}^{N} Q_{\mu_i}. \tag{3.4}$$

En este caso, los estados que se visiten con mayor frecuencia serán aquéllos que contribuyan de mayor manera al valor de Q_N y la frecuencia relativa con la que estos se elijan corresponderá a la cantidad de tiempo que el sistema real permanezca en dichos estados.

3.1.1. Cadenas de Markov

Necesitamos poder elegir los estados a visitar del sistema de tal forma que cada uno aparezca con su correspondiente probabilidad de Boltzmann. La manera más común de hacer esto es mediante *procesos de Markov*. Un proceso de Markov es un mecanismo que, dado un sistema en un estado σ, lo lleva a otro estado τ de manera aleatoria, como se explica a continuación. Este mecanismo no generará el mismo estado nuevo cada vez, partiendo de un mismo estado σ. La probabilidad de generar un estado τ en particular partiendo de un estado σ se conoce como la *probabilidad de transición*, $p_{\sigma\to\tau}$. Para que el mecanismo sea realmente un proceso de Markov, éste debe cumplir que las probabilidades de transición:

- no cambien con el tiempo
- dependan únicamente de las propiedades de los estados σ y τ, y no dependan de ningún estado previamente visitado.

Puesto de otro modo, podemos decir que la probabilidad de que un proceso de Markov genere un estado τ partiendo de un estado σ, es la misma siempre que parta de σ. Las probabilidades de transición deben satisfacer también que $\sum_{\tau} p_{\sigma\to\tau} = 1$. Además, puede existir una probabilidad no nula de que el sistema permanezca en su mismo estado mediante el proceso de Markov, es decir, $p_{\sigma\to\sigma} \geq 0$. Estos procesos de Markov se utilizan en simulaciones Monte Carlo repetidamente para generar lo que se conoce como una *cadena de Markov* de estados. Se elige esta estrategia ya que cuando se generan cadenas de Markov suficientemente largas partiendo de cualquier estado del sistema, éstas eventualmente generarán una sucesión de estados en donde cada uno aparecerá con su correspondiente probabilidad de acuerdo a la distribución de Boltzmann. Este proceso de alcanzar la distribución de Boltzmann es análogo al que realizan los sistemas reales al tender al equilibrio. Sin embargo, para alcanzar la distribución de Boltzmann, aún necesitamos imponer dos condiciones adicionales: la condición de *ergodicidad* y la condición de *balance detallado*.

3.1.2. Ergodicidad

La condición de *ergodicidad* se refiere a que debe ser posible que, mediante cadenas de Markov, el sistema alcance cualquier estado final partiendo de cualquier estado inicial en un tiempo finito. De hecho, es posible que algunas probabilidades de transición sean nulas para un par de estados en particular, pero cualquier estado del sistema

siempre debe ser accesible por al menos un "camino" en donde la probabilidad de transición a ese estado desde algún otro estado sea no-nula.

3.1.3. Balance detallado

La condición de *balance detallado* es la que garantiza que la distribución de equilibrio sea la distribución de Boltzmann. La condición de balance detallado es la siguiente:

$$P(\sigma)p_{\sigma\to\tau} = P(\tau)p_{\tau\to\sigma}, \tag{3.5}$$

donde $P(\sigma)$ es la probablidad de que el sistema se encuentre en el estado σ.
Esto nos dice que, en promedio, el sistema irá del estado σ al estado τ con la misma frecuencia con la que irá de τ a σ.

De este modo, podemos hacer que la distribución de probabilidad para los estados del sistema generados por procesos de Markov tienda a cualquier distribución de equilibrio eligiendo las probabilidades de transición que satisfagan la condición de balance detallado, ecuación (3.5). En particular, para generar la distribución de Boltzmann:

$$\frac{p_{\sigma\to\tau}}{p_{\tau\to\sigma}} = \frac{P^{\text{eq}}(\tau)}{P^{\text{eq}}(\sigma)} = \frac{P_\beta(\tau)}{P_\beta(\sigma)} = e^{-\beta(E_\tau - E_\sigma)} = e^{-\beta\Delta\mathscr{H}}, \tag{3.6}$$

donde $\Delta\mathscr{H} = E_\tau - E_\sigma$. Esta ecuación y el hecho de que $\sum_\tau p_{\sigma\to\tau} = 1$, son las constricciones en nuestra elección de probabilidades de transición. Al satisfacerlas, junto con la condición de ergodicidad, la distribución de equilibrio de los estados generados por los procesos de Markov será la distribución de Boltzmann.

3.2. Algoritmo de Metropolis

Existen varias formas de satisfacer las condiciones descritas en la sección anterior; en particular, la más conocida es el *algoritmo de Metropolis*, que fue introducido por Nicholas Metropolis y colegas en 1953 para simulaciones de gases de esferas duras [34]. La idea principal de este método es generar una secuencia larga de configuraciones tal que después de cierto tiempo, las nuevas configuraciones se generen siguiendo una distribución de probabilidad de equilibrio, que en este caso sería la distribución de Bolztmann.

Queremos obtener la distribución de Boltzmann como distribución de equilibrio.

$$P_\beta(\sigma) = \frac{1}{Z(\beta)} e^{-\beta\mathscr{H}(\sigma)}. \tag{3.7}$$

El problema es que no conocemos la función de partición, ecuación (3.1), pero sabemos que la probabilidad de estar en la configuración σ al tiempo $t+1$ está dada por la suma sobre todos los estados τ de la probabilidad de estar en uno de esos estados τ en el tiempo t por la probabilidad de hacer la transición entre τ y σ:

$$P_{t+1}(\sigma) = \sum_{\tau} P_t(\tau) p_{\tau \to \sigma}. \tag{3.8}$$

Podemos descomponer la probabilidad de transición de la siguiente manera:

$$p_{\sigma \to \tau} = s_{\sigma \to \tau} A_{\sigma \to \tau}, \tag{3.9}$$

en donde $s_{\sigma \to \tau}$ es la probabilidad de selección, que corresponde a la probabilidad, dado un estado inicial σ, de que la transición se considere hacia el estado final τ; y $A_{\sigma \to \tau}$ es la probabilidad de aceptación, es decir, la probabilidad de realizar la transición entre estos estados si ya seleccionamos el estado τ. Las condiciones que se deben cumplir para que la distribución de equilibrio sea la distribución que nosotros elegimos, son condiciones sobre las $A_{\sigma \to \tau}$, lo cual deja libre la elección de las $s_{\sigma \to \tau}$.

Elegimos entonces un conjunto de probabilidades de selección $s_{\sigma \to \tau}$ de alguna manera, y elegimos un conjunto particular de probabilidades de aceptación $A_{\sigma \to \tau}$ que satisfagan las condiciones de ergodicidad y balance detallado. De este modo, durante la simulación se generan nuevas configuraciones τ del sistema y aceptamos o rechazamos la transición entre estados aleatoriamente según las probabilidades de aceptación que elegimos. Si se acepta la transición, el estado del sistema se actualiza al nuevo estado τ. Si no, el sistema permanece en el estado inicial σ. El proceso se repite a lo largo de la simulación. La única restricción sobre la elección de las probabilidades de selección $s_{\sigma \to \tau}$, es que éstas deben satisfacer la condición de ergodicidad, es decir, cualquier estado del sistema debe ser accesible desde cualquier otro estado en un número finito de pasos. En el caso particular de la implementación computacional del plegamiento de proteínas presentada en éste trabajo, la elección de las probabilidades de selección se reduce a la elección de los *pull moves*. Generamos nuevas configuraciones a través de estos movimientos, descritos en el capítulo anterior. Por esta razón, es fundamental que el conjunto de movimientos que se elija sea completo. El que los *pull moves* sean locales tampoco es coincidencia; no queremos que haya brincos grandes en la energía en una transición entre estados. Esto significa que las transiciones se realizan entre estados del sistema en los que la energía es comparable.

En el caso más común donde se utiliza el algoritmo de Metropolis, el modelo de Ising, se suelen elegir las probabilidades de selección iguales entre sí, en el caso en el que la estrategia para obtener nuevas configuraciones del sistema sea mover un espín

a la vez. Supongamos que tenemos una red con N espines. Bajo la estrategia de mover un sólo espín a la vez, existen N diferentes espines que podrían moverse, y por lo tanto N posibles nuevas configuraciones del sistema. Por lo que la elección del valor de $s_{\sigma\to\tau}$ es clara:

$$s_{\sigma\to\tau} = \begin{cases} \frac{1}{N}, & \text{si } \tau \text{ es vecino} \\ 0, & \text{si no} \end{cases}. \tag{3.10}$$

Vemos que estando en el estado τ, tenemos también N posibles nuevas configuraciones que podemos generar. En particular, si elegimos regresar de τ al estado inicial σ, entonces $s_{\tau\to\sigma} = \frac{1}{N}$. Por lo tanto, en el caso común del modelo de Ising, donde suele explicarse el algoritmo de Metropolis, las probabilidades de transición se reducen a las probabilidades de aceptación, ya que:

$$\frac{p_{\sigma\to\tau}}{p_{\tau\to\sigma}} = \frac{s_{\sigma\to\tau}A_{\sigma\to\tau}}{s_{\tau\to\sigma}A_{\tau\to\sigma}} = \frac{A_{\sigma\to\tau}}{A_{\tau\to\sigma}}. \tag{3.11}$$

Sin embargo, en el caso de nuestra implementación del modelo HP con pull moves, no siempre existen el mismo número de nuevas configuraciones que se pueden generar desde dos estados diferentes σ y τ. Durante el desarollo del trabajo, este detalle pasó desapercibido por un tiempo, hasta que nos dimos cuenta de que era necesario incluirlo en nuestros métodos. Encontramos una mención al respecto de considerar este hecho en la referencia [38], sin embargo no se hace mucho énfasis ni se explica de donde viene realmente este término.

Escogemos la probabilidad de selección $s_{\sigma\to\tau}$ como el número de maneras de ir de una configuración σ a una nueva configuración τ sobre el número total de movimientos que llevan de una configuración σ a otra configuración cualquiera:

$$s_{\sigma\to\tau} = \frac{n_{\sigma\to\tau}}{N_\sigma}. \tag{3.12}$$

Es claro entonces que en nuestra implementación, los $s_{\sigma\to\tau}$ no pueden ser todos iguales, ni se cancelarán siempre en la ecuación (3.11). De hecho, la condición de balance detallado (3.5) se vuelve:

$$\frac{n_{\sigma\to\tau}}{N_\sigma}P^{\text{eq}}(\sigma)A_{\sigma\to\tau} = \frac{n_{\tau\to\sigma}}{N_\tau}P^{\text{eq}}(\tau)A_{\tau\to\sigma}, \tag{3.13}$$

donde $n_{\sigma\to\tau}$ es el número de movimientos que llevan al sistema de la configuración σ a la configuración τ, y N_σ es el número total de movimientos posibles partiendo de una configuración σ. Sin embargo, generalmente $N_\sigma \neq N_\tau$, es decir que el número de movimientos posibles partiendo desde un estado σ no coincide con el número de

movimientos posibles partiendo desde un estado τ. En cambio, $n_{\sigma\to\tau}$ y $n_{\tau\to\sigma}$ siempre coinciden debido a la reversibilidad de los *pull moves*. Más adelante se presenta un ejemplo de un caso simple en el cual no coinciden el número de movimientos posibles a partir de un estado σ y el número de movimientos posibles a partir de otro estado τ.

Una vez que escogemos nuestras probabilidades de selección correctamente, elegimos las probabilidades de aceptación para que cumplan con la ecuación de balance detallado (3.13). Entonces, para la implementación del método de Metropolis para el modelo HP usando pull moves, la ecuación (3.11) se vuelve más bien:

$$\frac{p_{\sigma\to\tau}}{p_{\tau\to\sigma}} = \frac{s_{\sigma\to\tau}A_{\sigma\to\tau}}{s_{\tau\to\sigma}A_{\tau\to\sigma}} = \frac{n_{\sigma\to\tau}N_\tau A_{\sigma\to\tau}}{n_{\tau\to\sigma}N_\sigma A_{\tau\to\sigma}} = e^{-\beta\Delta\mathscr{H}}, \tag{3.14}$$

con $\Delta\mathscr{H} := \mathscr{H}(\tau) - \mathscr{H}(\sigma)$. Por lo tanto,

$$\frac{A_{\sigma\to\tau}}{A_{\tau\to\sigma}} = \frac{n_{\tau\to\sigma}N_\sigma}{n_{\sigma\to\tau}N_\tau}e^{-\beta\Delta\mathscr{H}}, \tag{3.15}$$

Tomando en cuenta que en nuestra implementación no siempre hay el mismo número movimientos posibles partiendo de dos configuraciones distintas, la regla de Metropolis se vuelve:

$$p_{\sigma\to\tau} = \begin{cases} 1, & \text{si } \Delta\mathscr{H} \leq 0 \\ \frac{n_{\tau\to\sigma}N_\sigma}{n_{\sigma\to\tau}N_\tau}e^{-\beta\Delta\mathscr{H}}, & \text{si } \Delta\mathscr{H} > 0 \end{cases}. \tag{3.16}$$

La regla de Metropolis dice que si la nueva configuración τ tiene una energía menor a la configuración actual σ, entonces el sistema realiza automáticamente la transición entre estados. Sin embargo, si la nueva configuración τ tiene una mayor energía que la configuración σ, la transición se lleva a cabo con la probabilidad establecida por la regla de Metropolis.

Nótese que como estamos trabajando en el ensamble canónico, estamos obteniendo información del sistema a una temperatura T fija. Con una simulación a temperatura T, no podemos inferir mucho del comportamiento del sistema a otras temperaturas; si queremos estudiar el sistema a diferentes temperaturas, tenemos que hacer diferentes corridas, cada una a la temperatura deseada.

3.3. Recocido simulado

En conjunción con el método de Metropolis, se implementó un método metaheurístico conocido en inglés como *simulated annealing* y en español como *recocido simulado*. El nombre, en ambos casos, proviene de una técnica metalúrgica de la cual se

inspira éste método. Fue propuesto de manera independiente por Scott Kirkpatrick, C. Daniel Gelatt y Mario P. Vecchi en 1983 [39] y por Vlado Černý en 1985 [40].

En breve, lo que se implementó en este trabajo fue una simulación en donde, a lo largo de ésta, la temperatura del sistema disminuyera gradualmente desde un valor inicial T_0 hasta un valor final T_f en decrementos iguales ΔT en intervalos de tiempo regulares. Recordemos que en la implementación del método de Metropolis, nunca trabajamos directamente con la temperatura, sino más bien con β, la temperatura inversa. De este modo, y puesto que $T_0 > T_f$, al principio de la simulación, el sistema logra explorar con pocas restricciones su paisaje de energías debido a la energía térmica disponible de la que dispone el sistema para sobrepasar las barreras de energía libre que encuentre. Sin embargo, una simulación a temperatura alta, aunque permite muestrear una gran extensión del paisaje de energías, no permite que el sistema tienda hacia el estado de mínima energía. Mediante la disminución gradual de la temperatura, se busca entonces que el sistema explore libremente el paisaje de energía al principio de la simulación y que, conforme descienda la temperatura, el sistema vaya tendiendo hacia el mínimo global, para finalmente permanecer atrapado en éste al final de la simulación, cuando la temperatura sea mínima.

3.4. Ejemplo de no homogeneidad en la obtención de nuevas estructuras partiendo de diferentes configuraciones

A continuación presentamos un ejemplo simple de un caso en el que no coincide el número total de movimientos posibles partiendo de dos estructuras diferentes σ y τ. Es claro que, debido a la reversibilidad de cada *pull move*, $n_{\sigma\to\tau} = n_{\tau\to\sigma}$; si esto no se cumpliese, algún movimiento perteneciente al conjunto de los *pull moves* tendría que ser no-reversible en un solo movimiento. Consideremos la configuración inicial σ, extendida horizontalmente, como se muestra en la parte izquierda de la figura 3.1. Supongamos que elegimos el movimiento que mueve el penúltimo monómero debajo del último y completa el cuadrado. La configuración obtenida, llamémosla τ, se muestra en la parte izquierda de la figura 3.1. De un monómero extremo, cuyos movimientos no puedan ser interrumpidos por otros segmentos de la cadena, como es el caso en la configuración σ, se pueden realizar 7 movimientos. Esto es fácil de ver puesto que existen 3 primeros sitios a los cuales se puede mover el monómero vecino del extremo, mientras que el monómero extremo puede moverse a 3 diferentes

sitios adyacentes a uno de los primeros vecinos y a 2 diferentes sitios en los otros sitios adyacentes a los primeros vecinos. Esto se debe, como se mencionó al final de la sección 2.7.1 , a que no dejamos que el monómero extremo se "doble" sobre la estructura, ya que esto genera problemas de reversibilidad. Ahora, para la configuración extendida σ, cada monómero no extremo puede realizar 4 movimientos. Es decir, en total, se pueden realizar $N_\sigma = 46$ movimientos partiendo de la configuración σ. Ahora, contemos cuantos movimientos pueden realizarse a partir de la configuración τ. Empezando por el monómero extremo derecho, que forma parte de un *bucle cuadrado*, observamos que existen sólo 5 movimientos posibles para este extremo. El monómero que se encuentra justo antes tiene sólo dos movimientos posibles, mientras que el anterior a éste tiene un solo movimiento posible. El último monómero que forma parte del *bucle cuadrado*, tiene 3 movimientos posibles, aunque dos de ellos conducen a la misma estructura, con la única diferencia de que cada movimiento se realiza en sentido contrario en relación a la cadena. El primer monómero que ya no forma parte del *bucle cuadrado* puede realizar 3 movimientos, debido a que uno de los monómeros del bucle le estorba. Esto reduce el número total de movimientos posibles a $N_\tau = 37$, partiendo desde la configuración τ.

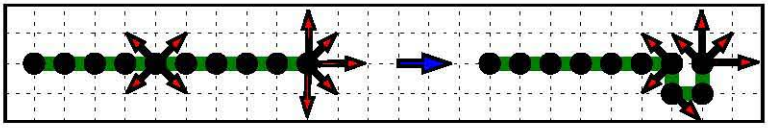


Figura 3.1: Ejemplo ilustrativo del cambio en el número total de movimientos realizables en diferentes configuraciones. La flecha azul representa un cambio de configuración, mientras que las rojas representan los posibles movimientos que se pueden generar a partir de mover los monómeros a los que pertenecen. Se presentan sólo los movimientos para un conjunto de monómeros por simplicidad.

Notemos que hay movimientos diferentes que implican mover un mismo monómero a un mismo sitio vacío. Por ejemplo, en la parte derecha de la figura 3.1, observamos que hay un monómero del cual se desprenden dos flechas rojas, y sin embargo indican 3 distintos movimientos. En este caso, la flecha roja que apunta hacia abajo en diagonal indica tanto el movimiento que consiste en mover ese monómero, llamémosle i, al sitio vacío que se encuentra a la izquierda del monómero $i+1$, como el movimiento que consiste en mover a i al sitio vacío que se encuentra debajo del monómero $i-1$, tomando en cuenta que los monómeros están ordenados de izquierda a derecha. Lo mismo ocurre para los movimientos del extremo de la configuración de la izquierda de la figura 3.1, en donde se representan 7 movimientos distintos mediante

5 flechas.

El que diferentes configuraciones tengan diferente número de movimientos posibles ocurre tanto en la red cuadrada como en la red triangular.

3.5. Enumeración exhaustiva

El método más básico que uno podría idear para calcular la densidad de estados de un sistema, es decir, el número de configuraciones por energía, es el de enumerar todas las configuraciones de dicho sistema y asociarles su correspondiente energía. Este método suele ser demasiado pesado computacionalmente, por lo que, en general, ningún problema se trata de manera tan ingenua. Sin embargo, para sistemas pequeños, en nuestro caso cadenas de pequeña longitud, la enumeración exhaustiva nos permite obtener *exactamente* la densidad de estados. Utilizamos este método para obtener la $g_{\text{exacta}}(E)$ para cadenas cortas y poder compararla con la $g_{W-L}(E)$ obtenida por medio del algoritmo Wang–Landau.

Para realizar la enumeración exhaustiva de una cadena de longitud l, simplemente partimos de una configuración inicial de la cadena, por ejemplo, una configuración horizontal completamente estirada. A partir de esta, calculamos todas las posibles configuraciones posibles a través de los *pull moves* a las que se puede llegar en un solo movimiento. Se guardan estas configuraciones en un conjunto. Luego, a todas las configuraciones obtenidas por un movimiento, se les busca todas las configuraciones que se pueden obtener a partir de ellas en otro movimiento. Así sucesivamente se añaden las configuraciones nuevas al conjunto, siempre y cuando éstas no formen ya parte de éste. Para cada configuración, calculamos su energía. Cuando no se encuentren más configuraciones nuevas del sistema que no formen parte ya del conjunto, habremos generado todas las configuraciones posibles y tendremos, además, la información de la energía para cada una de ellas. De este modo, conoceremos cuantas configuraciones por energía tiene el sistema, es decir, la densidad de estados $g(E)$. Este proceso se ejemplifica en la figura 3.2

3.6. Algoritmo Wang–Landau

El algoritmo Wang–Landau [35] fue propuesto en 2001 por F. Wang y D.P. Landau como una extensión al algoritmo de Metropolis. Fue diseñado para calcular en una sola simulación la densidad de estados de un sistema $g(E)$, que es el número de

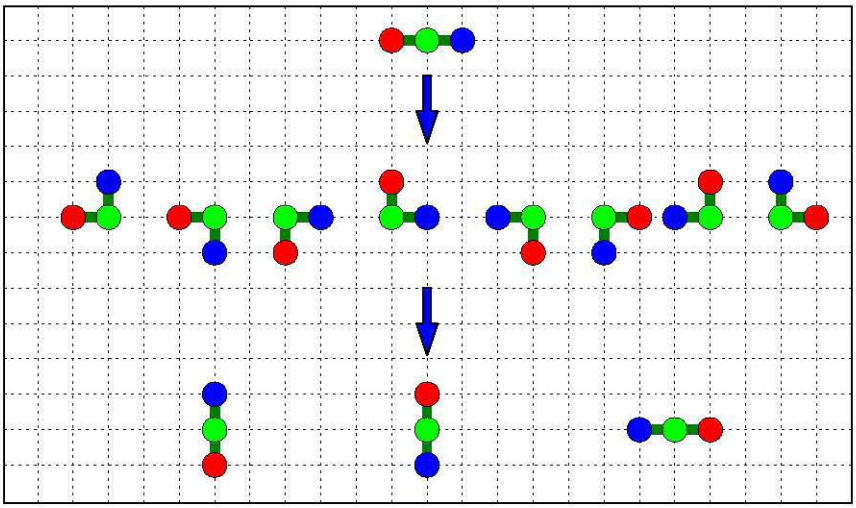


Figura 3.2: Ejemplo de enumeración exhaustiva para una cadena de longitud 3. Todas las estructuras pueden ser generadas en un solo movimiento, excepto por la cadena estirada horizontalmente en donde los monómeros están invertidos respecto a la estructura inicial.

microestados por energía que tiene un sistema. A diferencia del método de Metropolis, al utilizar el algoritmo Wang–Landau, sacamos información microcanónica. La densidad de estados no depende de la temperatura, sin embargo se puede extraer información termodinámica del sistema en función de la temperatura a partir de ésta. Este algoritmo está también muy relacionado al algoritmo de *muestreo entrópico* [41].

Recordando la expresión para la función de partición (3.1), los términos de la suma dependen solamente de la energía, por lo que, agrupando por configuraciones con una misma energía, obtenemos:

$$Z(\beta) = \sum_{E} \left[\sum_{\sigma:\mathscr{H}(\sigma)=E} e^{-\beta \mathscr{H}(\sigma)} \right]. \tag{3.17}$$

Los términos dentro de los corchetes tienen un mismo valor de $\mathscr{H}(\sigma)$, por lo tanto, la suma da como resultado $e^{-\beta E}$ multiplicado por el número de términos (estados) con esta energía E, $g(E)$. Esto es:

$$Z(\beta) = \sum_{E} g(E) e^{-\beta E}. \tag{3.18}$$

Esta última suma tiene mucho menos términos que (3.1), además no sólo podemos usarla para conocer la función de partición para cualquier valor de T, sino también

una gran variedad de cantidades termodinámicas, como se muestra más adelante en la sección 3.7.1. Conociendo la densidad de estados, podríamos evaluar la función de partición y obtener información del sistema.

La distribución de Boltzmann a temperatura fija permite al sistema visitar una reducida gama de valores distintos para la energía, aquéllos que sean más probables a ese valor fijo de la temperatura T. El algoritmo Wang–Landau intenta visitar el espectro completo de energías del sistema y evaluar directamente la $g(E)$. Para obtener la densidad de estados, modificamos la distribución de equilibrio para visitar de manera más uniforme todos los valores de la energía,

$$P^{\text{eq}}(E) = \sum_{\sigma:\mathscr{H}(\sigma)=E} P^{\text{eq}}(\sigma). \tag{3.19}$$

Es decir, queremos que la probabilidad de visitar un macroestado con energía E sea la misma para todo valor de E. En el caso de la ecuación anterior, la probabilidad de visitar algún estado con energía E está dada por la suma de las probabilidades de estar en un estado con este valor de la energía. Si suponemos que cada configuración tiene la misma probabilidad de ocurrir, entonces:

$$P^{\text{eq}}(E) = g(E)P^{\text{eq}}(\sigma_E), \tag{3.20}$$

para cada configuración σ_E con energía E. Por lo tanto, la probabilidad de equilibrio que deseamos está dada por la expresión:

$$P^{\text{eq}}(\sigma_E) = \frac{C}{g(E(\sigma_E))}, \tag{3.21}$$

donde C es una constante y $g(E(\sigma))$ es la densidad de estados con energía E correspondiente a la energía de la configuración σ. Elegimos de esta forma la probabilidad de equilibrio puesto que queremos que la P^{eq} sea una constante. De donde ahora tenemos,

$$\frac{P^{\text{eq}}(\tau)}{P^{\text{eq}}(\sigma)} = \frac{p_{\sigma\to\tau}}{p_{\tau\to\sigma}} = \frac{s_{\sigma\to\tau}A_{\sigma\to\tau}}{s_{\tau\to\sigma}A_{\tau\to\sigma}} = \frac{n_{\sigma\to\tau}N_\tau A_{\sigma\to\tau}}{n_{\tau\to\sigma}N_\sigma A_{\tau\to\sigma}} = \frac{g(E(\sigma))}{g(E(\tau))}. \tag{3.22}$$

De nuevo tenemos que considerar que no siempre hay el mismo número de movimientos posibles para dos configuraciones distintas. De este modo, definimos las probabilidades de aceptación de la siguiente manera:

$$\frac{A_{\sigma\to\tau}}{A_{\tau\to\sigma}} = \frac{n_{\tau\to\sigma}N_\sigma}{n_{\sigma\to\tau}N_\tau}\frac{g(E(\sigma))}{g(E(\tau))}. \tag{3.23}$$

Una solución a esto es la siguiente:

$$p_{\sigma\to\tau} = \begin{cases} 1, & \text{si } g(E(\tau)) \le g(E(\sigma)) \\ \frac{n_{\tau\to\sigma}N_\sigma}{n_{\sigma\to\tau}N_\tau}\frac{g(E(\sigma))}{g(E(\tau))}, & \text{si } g(E(\tau)) > g(E(\sigma)) \end{cases}. \tag{3.24}$$

Sin embargo, aún no conocemos $g(E)$; si la conociéramos, podríamos hacer una simulación con las probabilidades de transición entre estados tales que obtengamos la probabilidad de equilibrio deseada. Por esta razón, necesitamos estimar la densidad de estados.

3.6.1. Muestreo entrópico

Introduciremos la idea del algoritmo Wang–Landau en referencia a otro algoritmo conocido como *muestreo entrópico*, introducido por Lee en 1993 [41]. Se llama muestreo entrópico puesto que la entropía microcanónica está dada por $S(E) = k\log(g(E))$, y en implementaciones de este algoritmo se trabaja con la $S(E)$ en lugar de la $g(E)$, puesto que esta última puede tomar valores demasiado grandes para ser trabajados en una computadora. Las implementaciones del algoritmo Wang–Landau también trabajan con la entropía $S(E)$ por la misma razón, sin embargo, a continuación se explica todo en términos de $g(E)$ para aclarar la idea del método.

Empezamos con un estimado inicial de $g(E)$, llamémosle g_0; a partir de esto, generamos iterativamente los siguientes estimados de forma que converjan a la verdadera densidad de estados. Para esto, comenzamos la simulación utilizando el primer estimado g_0 para calcular las probabilidades de transición dadas por la regla (3.24) por un cierto número de iteraciones. Hasta éste punto, la probabilidad de equilibrio que imponemos es $P^{\text{eq}}(\sigma) = \frac{C}{g_0(\sigma)}$. Contamos el número de veces que el sistema visita cada valor de la energía durante esta parte de la simulación, a partir de lo cual generamos el histograma de visitas por energía $H(E)$. Sabemos que el valor de $H(E_i)$ para un valor E_i específico es proporcional a la probabilidad de que el sistema se encuentre en una configuración con esta energía E_i. Ahora, por construcción de (3.24), la probabilidad de visitar una energía E converge a:

$$P_0(E) = \sum_{\sigma:\mathscr{H}(\sigma)=E} \frac{C}{g_0(E(\sigma))} \propto \frac{g(E)}{g_0(E)}, \tag{3.25}$$

donde $g(E)$ es la densidad de estados real. Por lo tanto, podemos obtener una mejor aproximación a la densidad de estados real, si ahora consideramos:

$$g_1(E) \propto H(E)g_0(E). \tag{3.26}$$

Esta última no es necesariamente una igualdad puesto que el método deja libre la elección de una constante de normalización. Nótese, sin embargo, que al multiplicar la densidad de estados para cada valor de la energía por una misma constante, no se afectan en absoluto las propiedades termodinámicas del sistema.

De este modo generamos una secuencia de aproximaciones a la densidad de estados g_i que converjan a la verdadera densidad de estados y tal que:

$$g_{n+1}(E) \propto H_n(E) g_n(E). \tag{3.27}$$

El criterio para saber cuando cambiar a la siguiente aproximación para la densidad de estados es que el histograma $H(E)$ sea "plano". Existen distintas maneras de juzgar cuándo el histograma es lo suficientemente plano para pasar a la siguiente aproximación. En nuestro caso, el criterio fue determinar un número mínimo de visitas por cada valor de la energía previamente visitado a lo largo de la simulación. Si cada energía ha sido visitada al menos un número mínimo de veces $\nu_{\min}$, consideramos que el histograma es plano y calculamos la siguiente aproximación a $g(E)$ [42].

3.6.2. Calculo de $g(E)$ con Wang–Landau

La idea de Wang y Landau fue la de cambiar el estimado de $g(E)$ en cada paso del algoritmo. Esto surgió a causa de la lentitud en las implementaciones del algoritmo de muestro entrópico, el cual funciona en teoría, pero en la práctica es muy lento.

En el caso de Wang–Landau, cuando el sistema visita una configuración con energía E, se actualiza la densidad de estados para este valor de la energía:

$$\log(g(E)) \leftarrow \log(g(E)) + \log(f). \tag{3.28}$$

A f se le llama el *factor de modificación*, y determina la precisión del estimado de la densidad de estados. Mientras más pequeña sea la modificación, mayor precisión tendrá cada estimado de $g(E)$, pero también hará más tardada la simulación. Usualmente, se toma un valor inicial de $\log(f) = 1$ y se modifica después de un tiempo largo de la siguiente manera:

$$\log(f) \rightarrow \frac{\log(f)}{2}. \tag{3.29}$$

Este valor cambia sucesivamente, hasta alcanzar un valor final $\log(f)$. El cambio en el valor del factor de modificación es para refinar las primeras aproximaciones a la densidad de estados a lo largo de la simulación.

La inclusión del factor de modificación representa la presencia de una "fuerza" externa que empuja al sistema hacia valores de la energía aún no visitados. Si el sistema cae en una configuración con energía E que no había sido visitada, la correspondiente $g(E)$ tiene un valor pequeño, por lo que, de acuerdo a la regla (3.24), es más probable

que el sistema permanezca en esta energía, o incluso visite energías diferentes aún no visitadas, que el que regrese a estados con energías ya bien visitadas cuyos valores de $g(E)$ son mayores. De este modo, el algoritmo Wang–Landau permite que el sistema explore con mayor facilidad, y por lo tanto, con mayor rapidez, el espacio de energías. Esto también lo hace llegar al equilibrio de manera más rápida.

3.7. Análisis

Para estudiar más a fondo el proceso de plegamiento, calcularemos ciertas cantidades durante las simulaciones o a partir de los resultados que estas arrojen. A continuación se presenta brevemente qué cantidades se calcularán y de qué forma. El interés particular de realizar este análisis se motivará en el capítulo 4.

3.7.1. Cantidades termodinámicas

Puesto que el proceso de plegamiento depende directamente de la temperatura del sistema, nos interesa estudiar su termodinámica. Para esto, procedemos de dos maneras diferentes, según el algoritmo que usemos. En ambos casos, lo primero que hacemos es calcular la energía promedio $\langle E \rangle$.

Para el algoritmo de Metropolis, el cálculo de la energía promedio $\langle E \rangle$ es inmediato. Simplemente corremos una simulación a temperatura T fija y guardamos el valor de la energía a cada paso de la simulación para después promediarla, es decir,

$$\langle E \rangle = \frac{1}{N} \sum_{i=1}^{N} E_i \tag{3.30}$$

para una simulación de N pasos. De igual manera, nos interesa calcular $\langle E^2 \rangle$; para esto procedemos de la misma manera, guardando el valor de E^2 en cada paso de la simulación, de tal modo que

$$\langle E^2 \rangle = \frac{1}{N} \sum_{i=1}^{N} E_i^2. \tag{3.31}$$

Una vez teniendo estos valores, recordamos que la energía interna del sistema está dada por

$$U = \frac{1}{Z} \sum_{\sigma} E_\sigma e^{-\beta E_\sigma} = \langle E \rangle \tag{3.32}$$

y puede escribirse también en términos de la derivada de la función de partición respecto a la temperatura inversa, es decir,

$$U = \frac{1}{Z}\frac{\partial Z}{\partial \beta} = -\frac{\partial \log Z}{\partial \beta}. \tag{3.33}$$

De lo anterior, observamos que podemos operar con $\langle E \rangle$ y $\langle E^2 \rangle$ de la siguiente manera,

$$\langle (E - \langle E \rangle)^2 \rangle = \langle E^2 \rangle - \langle E \rangle^2 = \frac{1}{Z}\frac{\partial^2 Z}{\partial \beta^2} - \left[\frac{1}{Z}\frac{\partial Z}{\partial \beta}\right]^2 = -\frac{\partial^2 \log Z}{\partial \beta^2}, \tag{3.34}$$

para relacionar estas cantidades con otras cantidades termodinámicas como el calor específico,

$$\mathscr{C} = \frac{\partial U}{\partial T} = -k\beta^2 \frac{\partial U}{\partial \beta} = k\beta^2 \frac{\partial^2 \log Z}{\partial \beta^2}. \tag{3.35}$$

Resulta claro entonces, que podemos calcular el calor específico del sistema a temperatura fija T de los valores obtenidos para $\langle E \rangle$ y $\langle E^2 \rangle$ calculados durante la misma simulación,

$$\langle E^2 \rangle - \langle E \rangle^2 = \frac{\mathscr{C}}{k\beta^2}. \tag{3.36}$$

De aquí sigue que el calor específico por monómero está dado por la expresión

$$C = \frac{k\beta^2}{L}\left(\langle E^2 \rangle - \langle E \rangle^2\right), \tag{3.37}$$

donde L es la longitud del polímero.

En el caso del algoritmo Wang–Landau, de acuerdo con la ecuación (3.2), el valor esperado de la energía, o energía promedio, está dado por la expresión

$$\langle E \rangle = \frac{\sum_{\sigma} E_\sigma e^{-\beta E_\sigma}}{\sum_{\sigma} e^{-\beta E_\sigma}}. \tag{3.38}$$

Si agrupamos los términos de ambas sumas de la manera en que lo hicimos en la ecuación (3.17), obtenemos que

$$\langle E \rangle = \frac{\sum_{E} E g(E) e^{-\beta E}}{\sum_{E} g(E) e^{-\beta E}}. \tag{3.39}$$

De igual manera calculamos $\langle E^2 \rangle$

$$\langle E \rangle = \frac{\sum_{E} E^2 g(E) e^{-\beta E}}{\sum_{E} g(E) e^{-\beta E}}. \tag{3.40}$$

Así podemos calcular el calor específico por monómero a cualquier temperatura mediante la ecuación (3.37).

Otra cantidad que nos interesa calcular es la distribución de energías para una temperatura fija. El cálculo de esta cantidad es inmediato, puesto que la distribución está dada por la expresión

$$P_\beta(E) = \frac{1}{Z(\beta)} \sum_{\sigma:E(\sigma)=E} P_\beta(\sigma). \tag{3.41}$$

Así, agrupando como en la ecuación (3.17), obtenemos

$$P_\beta(E) = \frac{1}{Z(\beta)} g(E) e^{-\beta E}, \tag{3.42}$$

que corresponde a la distribución de energías para una temperatura dada.

3.8. Planteamiento de las simulaciones

Antes de comenzar la simulación, tenemos que plantear el problema. Con esto, nos referimos a crear la representación de la red y la proteína con las que se trabajará computacionalmente y preparar las simulaciones.

3.8.1. Red

Antes que nada, tenemos que definir el espacio en el cual habitará el sistema, es decir la red. Para crear la red, necesitamos especificar simplemente la longitud de la cadena que estudiaremos. En nuestra implementación computacional, existen dos representaciones distintas de la red: un arreglo de ceros en cada sitio vacío y $\pm n$, donde n es el número de monómero y su signo representa la polaridad de dicho monómero; y un arreglo donde se marca cada sitio en la red con un número entero, de manera que cada sitio tiene un número identificador único asociado según sus coordenadas. Se crean ambos arreglos al inicializar el objeto `Red`; como no hemos creado la proteína aún, el arreglo donde se registra el valor de la polaridad de cada monómero empieza siendo un arreglo de ceros.

Para cada sitio en la red, se genera una entrada en tres diferentes listas de tal modo que el número de la entrada en las listas corresponde al número entero asociado a ese sitio en particular. En estas listas guardamos la información acerca de sus sitios vecinos en la red, los vectores que apuntan hacia los distintos sitios vecinos, y las coordenadas del sitio en cuestión.

3.8.2. Proteína

Una vez teniendo listo el espacio en el cual puede moverse la proteína, creamos un objeto `Proteina` especificando en qué red queremos colocar al sistema, su longitud y la proporción de monómeros hidrofóbicos. De no especificar la proporción de monómeros hidrofóbicos en la cadena, la polaridad de cada monómero se asignará aleatoriamente. También es posible introducir una cadena en particular desde un archivo de texto que contenga una secuencia de caracteres que sean 'H' o 'P', indicando la polaridad de cada residuo.

Habiendo creado el objeto `Proteina`, lo introducimos en la red de la siguiente manera. Definimos arbitrariamente que la posición del primer monómero sea X_1; de esta manera, aquella entrada en la red de ceros que corresponda a la entrada con número entero X_1 en la otra representación de la red, contendrá la información de la polaridad del primer monómero, es decir ± 1. Del mismo modo, elegimos arbitrariamente una posición X_2 que sea vecina a X_1 para colocar la información de la polaridad del segundo monómero. Procedemos de la misma manera hasta colocar el último monómero en algún punto X_n en la red. De este modo, en esta representación de la red, tenemos la información de qué sitios están vacíos u ocupados, por quien son ocupados y qué polaridad tienen los monómeros que los ocupan. Además de esto, guardamos la información de la posición de cada monómero X_i en una lista `pos`; de este modo, no tenemos que buscar en toda la red por la posición del monómero i, sino que podemos acceder a esta información mediante la entrada i de la lista `pos`. En el presente trabajo, se eligió que la configuración inicial de los polímeros bajo estudio fuera la configuración completamente extendida. Esta elección es completamente arbitraria; si los movimientos que generan nuevas configuraciones de la proteína forman un conjunto completo, cualquier configuración puede ser alcanzada desde cualquier configuración inicial en un número finito de pasos.

3.9. Búsqueda de movimientos posibles

Teniendo la información de la proteína en la red y partiendo de una configuración inicial $P(t_0)$, se busca generar una lista de todos los *pull moves* posibles que se pueden realizar sobre $P(t_0)$. A esta lista le llamamos $M(t_0)$. A través de cualquier movimiento que forme parte de $M(t_0)$, generamos una nueva configuración de la cadena, $P(t_1)$. Por ejemplo, si la cadena está completamente extendida, existen dos movimientos en $M(t_0)$ que llevarán a una configuración $P(t_1) = P(t_0)$.

Para generar la lista de *pull moves* al tiempo t, $M(t)$, necesitamos encontrar aquellos movimientos que sean posibles, es decir, aquéllos para los cuales los sitios que serán ocupados al tiempo $t+1$ estén vacios al tiempo t. Si estos sitios, que buscan ser ocupados al tiempo $t+1$, están siendo ocupados por algunos monómeros al tiempo t, el movimiento no será tomado en consideración, debido a que dos residuos distintos no pueden ocupar el mismo sitio en la red al mismo tiempo. Buscamos los sitios en la red necesarios para realizar el movimiento del monómero i, donde $i \neq 0, l-1$. Supongamos que este monómero se encuentra en la posición $X_i(t)$, para que el movimiento –que llevará al monómero a la posición $X_i(t+1)$– sea considerado, el vértice de la red V_1, localizado en $X_{V_1}(t) = X_i(t+1)$, debe estar libre. De igual manera, el vértice que completaría el cuadrado, V_2, debe estar libre o contener al monómero $i-1$ o $i+1$, según sea el sentido del movimiento. Si el vértice V_2 contiene a cualquier otro monómero diferente de $i \pm 1$ (según sea el caso), el movimiento no podrá realizarse. La elección de los vértices que se verifican se hace conforme a lo establecido en el capítulo referente a los *pull moves*.

Como resultado de este proceso, obtenemos una lista de los movimientos posibles partiendo de una configuración particular $P(t)$.

3.10. Implementación del algoritmo de Metropolis

Una vez teniendo la lista de los movimientos posibles, se elige alguno de estos al azar. El movimiento se acepta o se rechaza de acuerdo con la regla de transición entre estados del algoritmo de Metropolis (3.16).

Para esto, necesitamos calcular cuántos movimientos es posible realizar desde la configuración actual $P(t)$, y cuántos movimientos es posible realizar desde la configuración hacia la cual se considera la transición $P(t+1)$. Puesto que sólo necesitamos saber el número total de movimientos en cada caso, simplemente creamos copias del objeto `Proteina` a las que acomodamos en las configuraciones correspondientes a $P(t)$ y $P(t+1)$ y buscamos todos los movimientos posibles.

3.10.1. Animaciones

Para este trabajo se implementó también una parte visual que consiste en animar esquemáticamente el proceso de plegamiento. Dichas animaciones tienen como fin el visualizar didácticamente el proceso de plegamiento que se simula. La visualización

del proceso, sin embargo, hace más lenta la simulación.

3.10.2. Implementación de la técnica de recocido simulado

Como se explicó anteriormente, la técnica de *simulated annealing*, o *recocido simulado*, consiste en disminuir gradualmente la temperatura del sistema. De esta forma, al principio de la simulación se tendrá una temperatura mayor, lo que permitirá al sistema sobrepasar con mayor facilidad las barreras de energía libre que separan los mínimos locales del paisaje de energía, mientras que al final de la simulación, a una menor temperatura, el sistema tenderá a quedarse atorado en un mínimo del paisaje de enegía, que esperamos, sea el mínimo global.

Para simular este proceso, sólo hay que dividir el número de iteraciones que durará la simulación en pequeños intervalos que correrán a distintas temperaturas. Es importante no hacer grandes cambios a la temperatura entre cada intervalo, porque queremos que la simulación sea cercana al equilibrio térmico, a pesar de estar cambiando la temperatura. Con esto nos referimos a que pretendemos que el proceso sea cuasiestático, es decir que los cambios en las temperaturas sean lo suficientemente pequeños y que el tiempo de relajación por temperatura sea lo suficientemente largo como para que el sistema nunca se aleje considerablemente del equilibrio.

3.11. Implementación del algoritmo Wang–Landau

Al principio de la simulación realizamos una estimación inicial del número total de valores de la energía N_E accesibles al sistema, y definimos un valor inicial y un valor final para el factor de modificación f, f_{inicial} y f_{final}. Creamos un arreglo de longitud N_E que será nuestro histograma $H(E)$, en donde registramos el número de visitas a cada estado correspondiente a una energía E. La entrada i del arreglo $H(E)$ corresponderá al macroestado $E = -i$ del sistema. En un principio, todas las entradas del histograma $H(E)$ serán -1, denotando que aún no se ha visitado ningún estado. El sistema explorará el espacio de configuraciones de acuerdo con las probabilidades de transición entre estados determinadas por (3.24). Cada vez que la proteína cambie de configuración, se calcula la energía de la nueva configuración alcanzada. De este modo, al alcanzar una configuración particular que exhiba una energía E_i, cambiamos la entrada i del histograma por $n+1$, donde n es el número de veces que esa energía había sido visitada antes durante la simulación. De visitar algúna energía que no haya sido alcanzada previamente, se cambia la entrada en $H(E)$ correspondiente a esta

energía de -1 a 1.

Paralelamente, creamos un arreglo $S(E)$ que corresponderá al valor de la entropía microcanónica para cada energía. $S(E)$ es un arreglo de longitud N_E, donde inicialmente cada entrada es 0. Por cada configuración alcanzada durante la simulación que exhiba una energía E_i, sumamos el valor actual de f a la entrada i del arreglo S.

Dejamos correr la simulación hasta que el histograma sea "plano". Si el número de visitas de cada entrada correspondiente a una energía previamente visitada alcanza un valor ν_{min}, entonces el histograma es considerado plano. No importa realmente si hay entradas cuyo número de visitas rebase el valor ν_{min} mientras cada una haya sido visitada, al menos, este número de veces. Cuando se cumpla esto, normalizamos el arreglo $S(E)$, reiniciamos el histograma $H(E)$ y disminuimos el valor del factor de modificación f. Para normalizar el arreglo $S(E)$, encontramos el valor mínimo S_{min} de las entradas de $S(E)$ y lo restamos para cada valor de E que haya sido previamente visitado, es decir, para aquellas entradas donde $H(E) \neq -1$. El histograma $H(E)$ se reinicia haciendo 0 todas las mismas entradas que hayan sido previamente visitadas. Por otro lado, refinamos la aproximación a $g(E)$ cambiando el valor de f según lo explicado en (3.29).

Se repite todo el proceso hasta que f alcance un valor final f_{final}, después de lo cual se tiene una aproximación a la densidad de estados $g(E)$. Si esta aproximación no es demasiado cercana a la densidad de estados real, puede refinarse la simulación extendiendo el número de iteraciones (cambiando el valor de f_{final} o disminuyendo el cambio en f) o aumentando el valor mínimo de visitas ν_{min} para considerar plano el histograma $H(E)$.

3.12. Comentarios sobre la elección de Python

Todas las simulaciones fueron escritas en el lenguaje de programación interpretado Python. Se eligió desarrollar todo el código en este lenguaje, aprovechando algunos paquetes especializados, debido a la facilidad que ofrece para trabajar con arreglos y a su versatilidad. Sin embargo, el programar en Python también tiene algunas desventajas. La mayor de todas es la lentitud de las simulaciones, debido en que es un lenguaje interpretado y no compila. Los tiempos de las simulaciones realizada para este trabajo se reducirían de haber sido programadas en un lenguaje de programación como C. Por otro lado, en el código se trabaja en muchos puntos con arreglos de datos, lo cual Python permite hacer fácilmente. En otros lenguajes que permitirían correr más rápido

las simulaciones, el trabajar con arreglos no es tan simple como lo es en Python.

En conclusión, la elección de Python como lenguaje de programación para desarrollar el código de las simulaciones de este trabajo se basó en la facilidad del lenguaje, sobre todo para operar sobre arreglos de datos, y en la versatilidad que ofrece.

Capítulo 4

Resultados

En este capítulo se muestran, analizan e interpretan los resultados obtenidos de las diversas simulaciones, utilizando el algoritmo de Metropolis, la técnica de recocido simulado y el algoritmo Wang–Landau, en las redes cuadrada y triangular. Se estudia principalmente una cadena de 64 residuos que fue diseñada para tener un comportamiento similar al de las proteínas. Antes de estudiar a profundidad la cadena de 64 residuos, utilizamos cadenas más cortas para formar una idea general del proceso y estudiar ciertas características de cada método. Al final del capítulo se discuten las diferencias entre los métodos y las geometrías de la red utilizadas. Se presentan argumentos fundamentados en los resultados de las simulaciones para discutir en cuál de las dos redes la cadena de 64 residuos presenta un comportamiento más realista.

4.1. Resultados con el algoritmo de Metropolis

Antes de correr las simulaciones finales –de las cuales sacaremos resultados y conclusiones–, es una buena idea medir el tiempo que tarda una simulación bajo diferentes valores de las características del sistema a tratar, en particular de la *longitud* y de la *hidrofobicidad* del polímero a simular. Medir el tiempo de la simulación en función de la longitud de la cadena nos permite determinar el tamaño del sistema que simularemos; si la cadena es larga, las simulaciones podrían tomar demasiado tiempo, mientras que si la cadena es corta, no observaremos un comportamiento particularmente interesante del sistema. Estudiamos también cómo afecta la hidrofobicidad, es decir el porcentaje de residuos hidrofóbicos en la cadena, al tiempo necesario para

llevar a cabo una simulación. Mediante los resultados de estos estudios preliminares, determinamos la viabilidad de la realización de simulaciones para valores específicos de estas características de la proteína modelo.

4.1.1. Tiempo de cómputo en función de la longitud

Para estudiar cómo crece el tiempo de cómputo en función de la longitud de la cadena, realizamos simulaciones para cadenas de distintas longitudes, midiendo el tiempo requerido para realizar cierto número de iteraciones. Debido al carácter estocástico del método de Metropolis, decidimos promediar el tiempo de simulación sobre 20 corridas independientes usando una misma longitud simulando 1000 iteraciones en cada caso. Queremos además que el porcentaje de residuos hidrofóbicos sea el mismo entre las distintas cadenas de una misma longitud, puesto que esperamos que este factor también modifique el tiempo de cómputo. Aún teniendo el mismo porcentaje de residuos hidrofóbicos en distintas cadenas de misma longitud, estos podrían acomodarse de diferentes maneras, resultando en variaciones considerables entre las simulaciones. Debido a esto, decidimos simular todas las longitudes con *todos* los residuos hidrofóbicos.

En la figura 4.1 podemos observar que el tiempo de cómputo no crece linealmente con la longitud de la cadena. Observamos también que el tiempo de cómputo es menor para la red triangular que para la cuadrada. Esto se debe al número de contactos hidrofóbicos que se forman en la red triangular, que es superior a la red cuadrada, tal como se discutió en el capitulo 2. Al formar más contactos hidrofóbicos, de acuerdo a la regla de transición de Metropolis (3.16), es menos probable cambiar de configuración por otra de mayor energía, por lo que el movimiento se rechaza. Es claro que rechazar movimientos es menos tardado computacionalmente que aceptarlos, puesto que implica menos operaciones sobre la cadena.

Queda claro que el tiempo de cómputo en función de la longitud no exhibe un crecimiento lineal, pero nos interesa determinar qué tipo de comportamiento es el que exhibe. Para esto, repetimos la gráfica 4.1 en escala logarítmica.

Mediante la gráfica 4.2, podemos decir que el crecimiento del tiempo de cómputo en función de la longitud de la cadena, ignorando los primeros datos, es lineal en esta escala. Por lo tanto, inferimos que el tiempo crece como una potencia de la longitud en ambos casos. Para investigar el valor de la potencia, se hizo un ajuste de mínimos cuadrados a los datos, de nuevo ignorando los primeros 4 valores en cada caso, y se obtiene como valor del exponente: 2.23 para la red cuadrada y 2.20 para la red

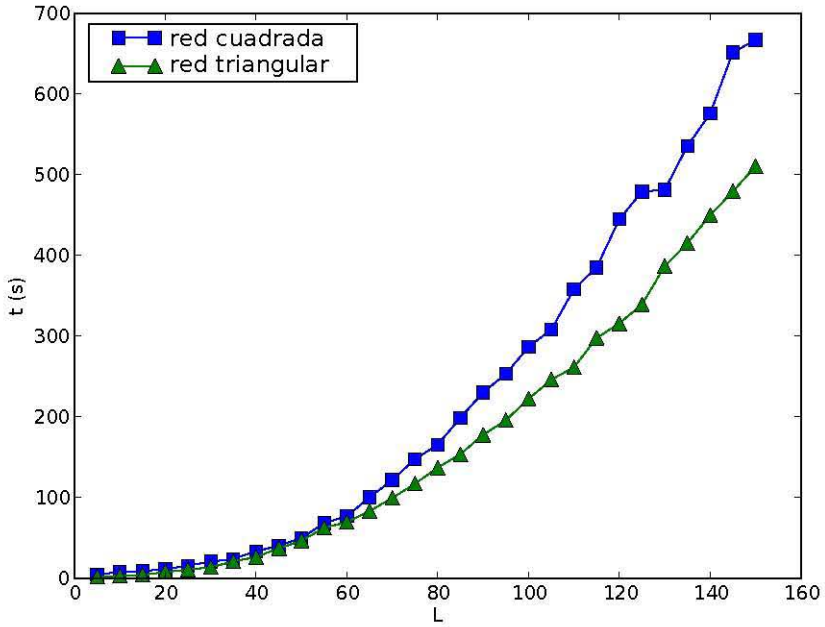


Figura 4.1: Tiempo promedio de simulación usando el algoritmo de Metropolis con 1000 iteraciones por réplica, 20 réplicas por longitud, con 100% de residuos hidrofóbicos.

triangular.

Ignoramos los primeros 4 valores de cada conjunto de datos para cada una de las redes, puesto que para longitudes pequeñas de la cadena, el tiempo de "preparación" de la simulación, es decir, el tiempo que lleva crear todos los objetos y acoplar las representaciones de la proteína a la red, entre otras cosas, es comparable con el tiempo de simulación. Sin embargo, este tiempo de "preparación" no crecerá de la misma manera con la longitud que el tiempo requerido para realizar la simulación en forma, es decir, los movimientos de la cadena de acuerdo al algoritmo de Metropolis.

De las figuras 4.3 y 4.4, observamos que los datos obtenidos del tiempo en función de la longitud crecen paralelamente a las potencias que calculamos mediante mínimos cuadrados, es decir, comparten el mismo valor de la pendiente en escala logarítmica. Esto indica que los datos crecen como una potencia cuyo valor coincide con esta pendiente.

Una vez que conocemos el exponente de la potencia de crecimiento de los datos,

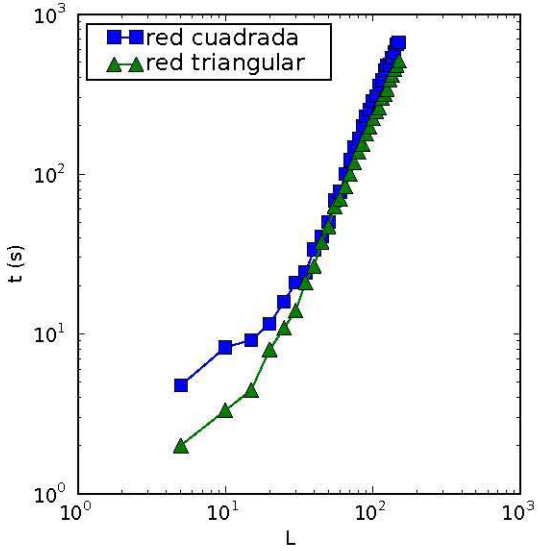


Figura 4.2: Crecimiento del tiempo de cómputo con el algoritmo de Metropolis en función de la longitud para ambas redes en escala logarítmica.

podemos volver a hacer un ajuste de mínimos cuadrados utilizando esta información para encontrar el polinomio que mejor ajuste nuestros datos. Conociendo este polinomio, podemos predecir cuánto tiempo tardará una simulación con todos sus residuos hidrofóbicos para cualquier valor de la longitud.

En la introducción del presente trabajo, se mencionó que el tamaño promedio –en términos del número de residuos– de una proteína es de 200 a 300 residuos. Si quisiéramos simular una secuencia de 200 residuos utilizando este algoritmo tal como ha sido implementado en este trabajo, para simular 10^6 iteraciones nos llevaría alrededor de 349 horas en la red cuadrada y 260 horas en la red triangular, como puede inferirse de la figura 4.5. Además, como veremos más adelante, estos 10^6 pasos podría no conducir a ningún resultado concreto, sobre todo si tomamos en cuenta que conforme crece la longitud de la secuencia a simular, el número de estados en energía accesibles crece, mas no así el estado nativo, en teoría. Recordemos que la estructura del estado nativo, para ser funcional, debe tener una única (o casi única) conformación. Estas

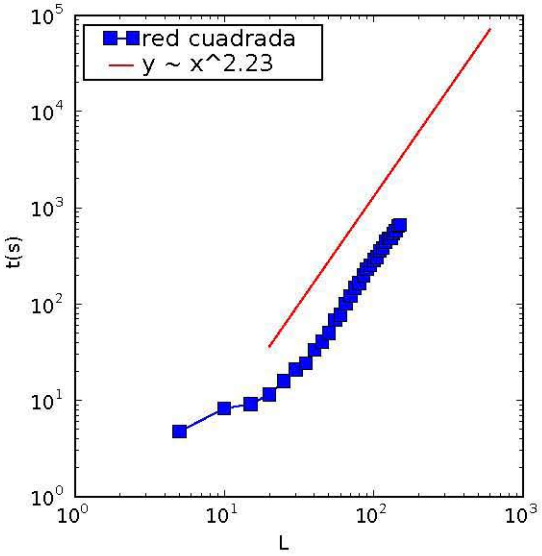


Figura 4.3: Análisis del crecimiento del tiempo de cómputo con el algoritmo de Metropolis en función de la longitud para la red cuadrada.

predicciones suponen que el tiempo de cómputo para una simulación con X veces el número de pasos más que otra, será X veces mayor. Esta suposición es acertada ya que durante la simulación, las operaciones que se repiten en cada iteración son básicamente las mismas, por lo que cada iteración nueva consiste en repetir lo que se hizo en la anterior; de modo que el tiempo que toma realizar 10^6 veces una misma operación equivale, aproximadamente, al tiempo que toma realizar 1000 iteraciones 1000 veces.

Del análisis concluimos que no es viable simular cadenas largas con este método, puesto que esto requiere de demasiado tiempo de cómputo. Más adelante retomaremos este análisis para argumentar el por qué se elige una cadena de longitud 64 para ser simulada mediante el algoritmo de Metropolis.

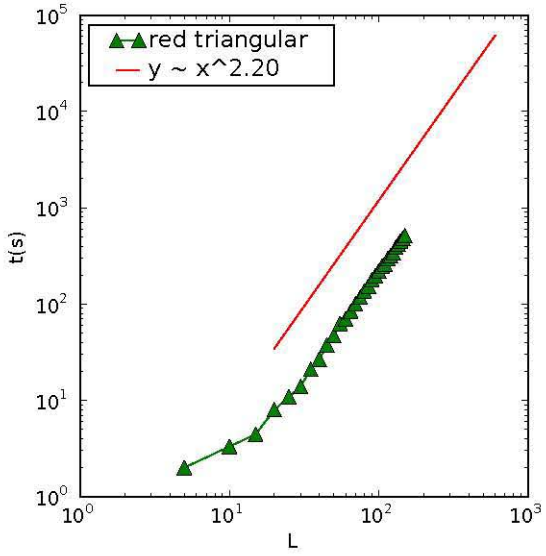


Figura 4.4: Análisis del crecimiento del tiempo de cómputo con el algoritmo de Metropolis en función de la longitud para la red triangular.

4.1.2. Tiempo de cómputo en función del porcentaje de residuos hidrofóbicos

Como mencionamos anteriormente, el porcentaje de residuos hidrofóbicos en una cadena modifica el tiempo de cómputo que requiere una simulación de Metropolis. Sabemos que mientras más contactos hidrofóbicos haya formados, será más difícil cambiar de configuración hacia aquellas que tengan una mayor energía. También es claro que el número de contactos hidrofóbicos que se formen depende del número de residuos hidrofóbicos de la cadena. Por lo que esperamos que conforme el porcentaje de residuos hidrofóbicos crezca, el tiempo de cómputo decrezca, debido a que, según la regla (3.16), una mayor cantidad de movimientos serán rechazados. Debido a esto, es importante determinar de qué manera influye la hidrofobicidad en el tiempo de cómputo.

Utilizamos una cadena de longitud 50 con distintos porcentajes de hidrofobici-

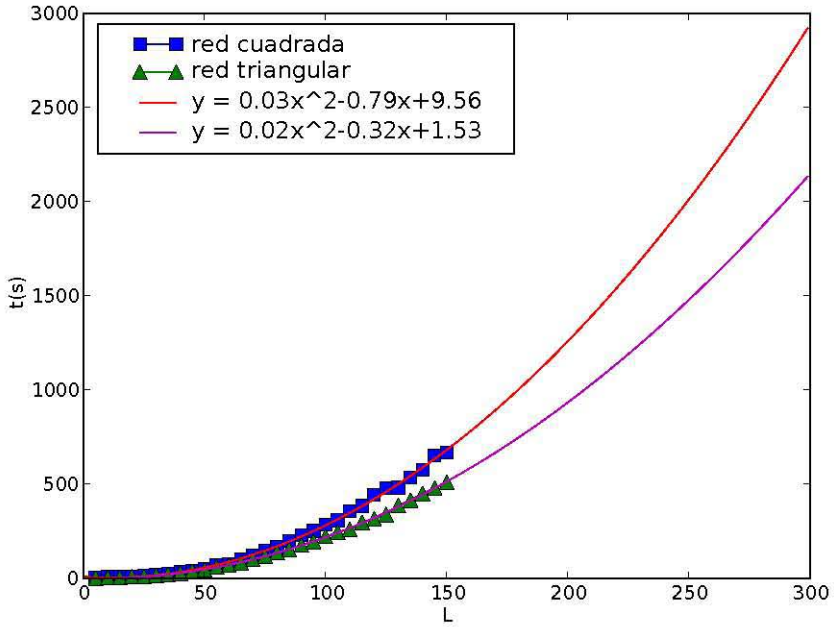


Figura 4.5: Extrapolación del crecimiento del tiempo necesario, para una simulación de Metropolis de 1000 pasos con una cadena completamente hidrofóbica, en función de la longitud de la secuencia.

dad. Realizamos 20 simulaciones por valor de porcentaje de residuos H (réplicas) y promediamos el tiempo.

De la figura 4.6 podemos confirmar que el tiempo de cómputo es menor mientras más residuos hidrofóbicos haya, lo cual esperabamos porque la cantidad de movimientos rechazados es mayor. Observamos un decalaje entre los tiempos de simulación en la red cuadrada y la red triangular. Esto puede explicarse debido a que los movimientos en la red triangular son más simples que en la red cuadrada. Por más simples, nos referimos a que, por lo general, se desplazan menos monómeros por movimiento.

Observamos también que la variación en el tiempo requerido no es tan grande para el rango de hidrofobicidad comprendido entre 60 % y 100 %. Este rango de valores del porcentaje de hidrofobicidad es, en el caso del presente trabajo, el que nos interesa, puesto que queremos estudiar el plegamiento de proteínas debido a la inter-

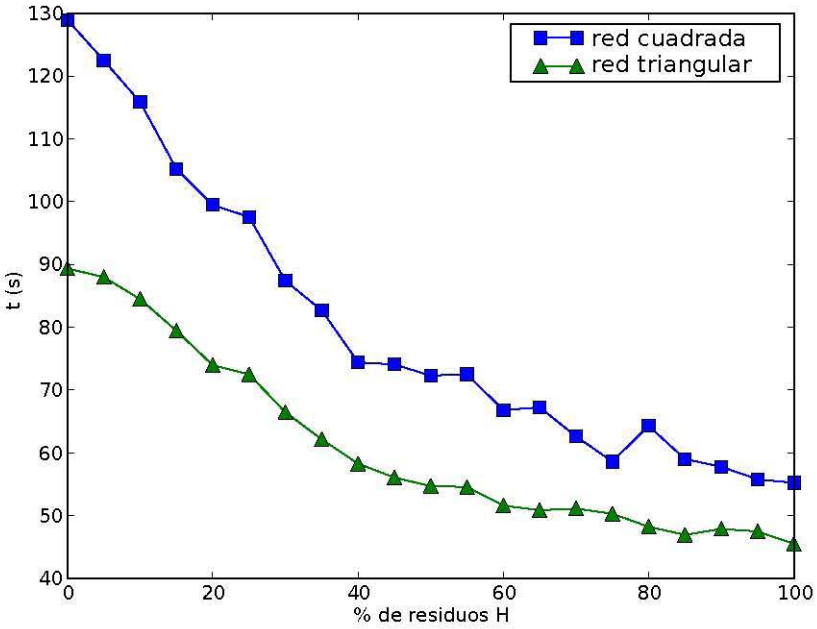


Figura 4.6: Tiempo promedio de simulación usando el algoritmo de Metropolis con 1000 iteraciones por réplica, 20 réplicas por valor de porcentaje de hidrofobicidad, con cadenas de longitud 50.

acción hidrofóbica. Por esta razón, requerimos que haya un porcentaje considerable de residuos hidrofóbicos para que el polímero que simularemos exhiba un proceso de plegamiento interesante.

4.1.3. Descripción cualitativa del comportamiento del modelo a distintas temperaturas

Sabemos que la dinámica de Metropolis depende de la temperatura, por lo que es buena idea hacer un breve estudio previo para elucidar el comportamiento del sistema a distintas temperaturas. Como queremos una descripción cualitativa del sistema para estudiar su comportamiento a grandes rasgos, utilizamos una cadena corta, de longitud 20 con 50 % de residuos hidrofóbicos: `HPHPPHHPHPPHPHHPPHPH`, que llamamos `sec20`, sacada de la referencia [43]. Sabemos de antemano que la energía

mínima de esta secuencia es $E = -9$, en una red cuadrada. Simularemos este polímero modelo por 1000 pasos a distintas temperaturas en la red cuadrada. No importa realmente en qué red corramos la simulación puesto que lo único que nos interesa es el comportamiento cualitativo a diferentes temperaturas, el cual es idéntico en ambas redes. Utilizamos la red cuadrada ya que es de la que sabemos la energía mínima de la secuencia.

En la figura 4.7, en donde se muestra la evolución temporal de la energía de las configuraciones visitadas, observamos los diferentes comportamientos de la secuencia simulada a diferentes temperaturas. Podemos ver que a temperaturas altas ($T = 10$), figura 4.7(a), el sistema se mueve indiscriminadamente por el paisaje de energía sin dirigirse hacia algún mínimo. El polímero simulado a esta temperatura se pliega parcialmente y se despliega continuamente a lo largo de la simulación. En $T = 1$, figura 4.7(b), el sistema exhibe un comportamiento similar, sin embargo en este caso, se despliega ($E = 0$) en menos ocasiones; el polímero pasa más tiempo en configuraciones con energía más baja que en el caso de $T = 10$, por lo que en $T = 1$ la energía promedio de la simulación es menor. Cabe mencionar que el estado $E = 0$ no consta solamente de estructuras completamente estiradas; decimos que las configuraciones con $E = 0$ están desplegadas por considerarlas como parte del estado desnaturalizado de la secuencia ya que no existe ningún contacto hidrofóbico. En el caso de $T = 0.5$, figura 4.7(c), observamos que la búsqueda en el paisaje de energía es menos indiscriminada que en los casos anteriores, sin embargo, observamos también que la energía térmica aún es tal que mantiene al sistema fluctuando entre estados de energía superiores a la energía mínima. Para $T = 0.3$, figura 4.7(d), observamos que la secuencia `sec20` alcanza su energía mínima ($E = -9$) después de ~ 800 iteraciones. Nótese que en repetidas ocasiones el sistema tiene que ir a estados más altos en energía que en los que se encuentra para posteriormente alcanzar estados de energía más bajos. Esto nos indica que probablemente el polímero tiene que deshacer ciertos contactos que ya tenía formados para poder formar un mayor número de contactos en otra configuración. Esta observación sugiere que hay contactos que favorecen la formación de la estructura nativa, contactos nativos, mientras que hay otros que estorban; estos últimos contactos son los que forman las trampas cinéticas de plegamiento, es decir, los mínimos locales en el paisaje de energía donde el sistema se "congela". Recordemos que estos mínimos locales están formados por la frustración de la secuencia, como se explicó en la sección 1.2.5. Disminuyendo la temperatura a $T = 0.2$, figura 4.7(e), observamos que el sistema se dirige más directamente al estado de mínima energía que en el caso anterior, pero con la diferencia de que, en este caso, el mínimo global no es alcanzado. Podríamos decir que el sistema se queda "congelado" en una

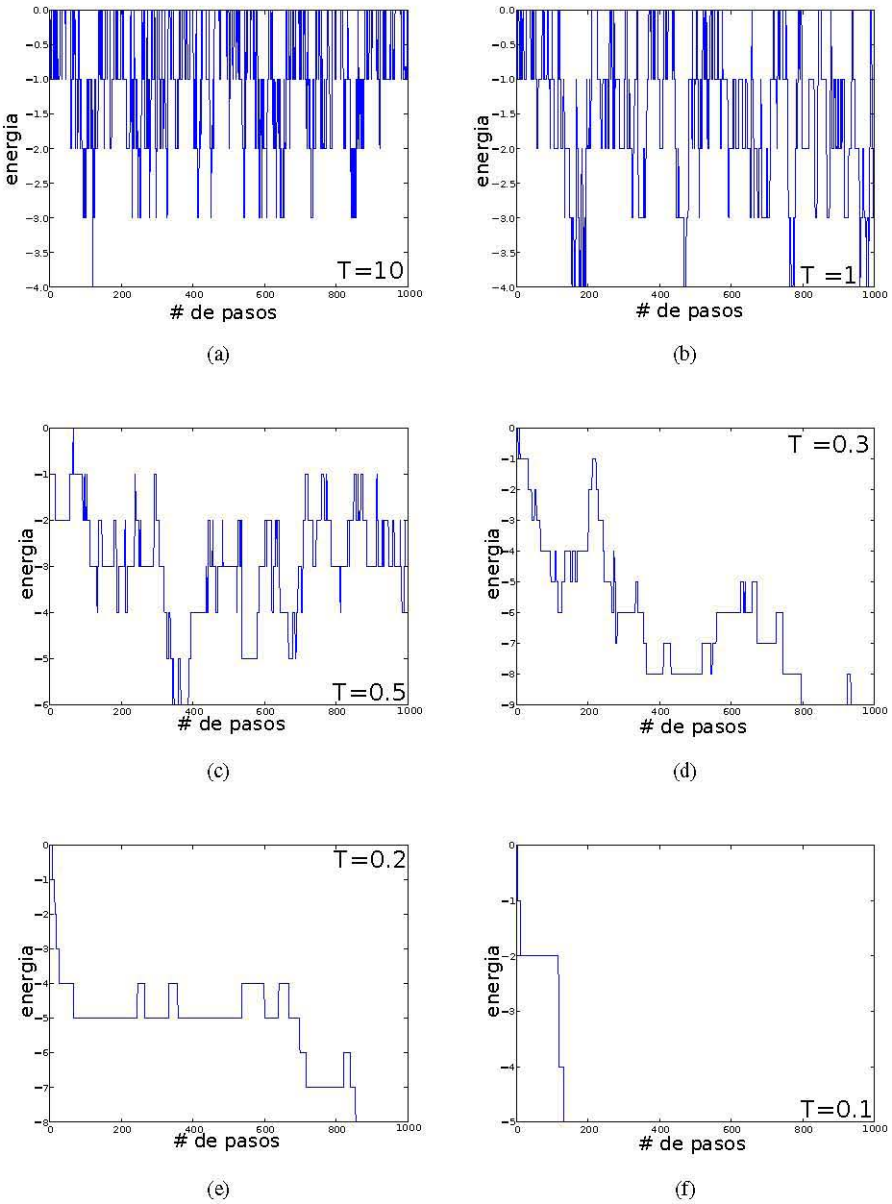


Figura 4.7: Evolución de la energía del sistema (`sec20`) a diferentes temperaturas con el algoritmo de Metropolis.

meseta desde la iteración ~ 80 hasta la ~ 700. Al disminuir aún más la temperatura, en $T = 0.1$, figura 4.7(f), lo que vemos es que el sistema se dirige directamente hacia un mínimo local del paisaje de energía en el cual se "congela", debido a que no hay suficiente energía térmica que le permita brincar las barreras de energía libre que lo rodean, permaneciendo "congelado" en $E = -5$ al final de la simulación. Cabe mencionar que de alargar la simulación, es posible que el sistema logre hacer la transición hacia menores energías.

A partir de esta descripción cualitativa del comportamiento del sistema a diferentes temperaturas podemos concluir que existen temperaturas de transición entre el comportamiento *indiscriminado* de la búsqueda en el paisaje de energía, el comportamiento que permite explorar el paisaje pero dirigiéndose hacia el mínimo global y el comportamiento de "congelación" del sistema.

En la figura 4.8 se muestran las configuraciones finales alcanzadas por cada simulación de la figura 4.7. Observamos que para las temperaturas altas, las estructuras finales alcanzadas están bastante extendidas. Para la simulación a $T = 10$, la estructura final tiene un solo contacto hidrofóbico, figura 4.8(a), mientras que para la simulación a $T = 1$, la estructura final tiene 3 contactos hidrofóbicos, figura 4.8(b). Para la siguiente simulación, a $T = 0.5$, observamos también 3 contactos hidrofóbicos, figura 4.8(c). Sin embargo, a pesar de que en estas últimas simulaciones, a $T = 1$ y a $T = 0.5$, ambas estructuras finales tienen la misma energía, notamos que hay un cambio significativo en términos estructurales. Para la simulación a $T = 0.5$, observamos un mayor grado de formación estructural, lo cual resulta en una estructura más compacta. Podríamos haber esperado esta diferencia a partir de la figura 4.7, ya que observamos que para la simulación a $T = 1$, figura 4.7(b), se visitan repetidamente estructuras con $E = 0$, mientras que para la simulación a $T = 0.5$, en donde sólo se regresa una vez al estado $E = 0$, se observa que al final, la dinámica fluctúa entre estructuras comprendidas entre $E = -1$ y $E = -4$, figura 4.7(c). En el caso de la simulación a $T = 1$, el sistema se pliega parcialmente y luego se despliega continuamente, lo cual no le permite formar demasiada estructura. Para la simulación a $T = 0.3$, observamos un ejemplo de configuración perteneciente al mínimo global de la energía $E = -9$, es decir, el estado nativo, figura 4.8(d). La siguiente simulación, a $T = 0.2$, alcanza una estructura final con energía $E = -8$, figura 4.8(d). Observamos que esta estructura es similar a la estructura nativa, al menos a primera vista. En la última simulación, a $T = 0.1$, se alcanzó una estructura con energía $E = -5$. Se observa en la figura 4.8(f), que esta estructura, a pesar de haber formado 5 contactos hidrofóbicos, aún no es muy compacta. Podemos explicar este hecho observando la figura 4.7(f), en donde vemos que una vez que se forma un contacto, éste jamás se deshace a lo largo de la simula-

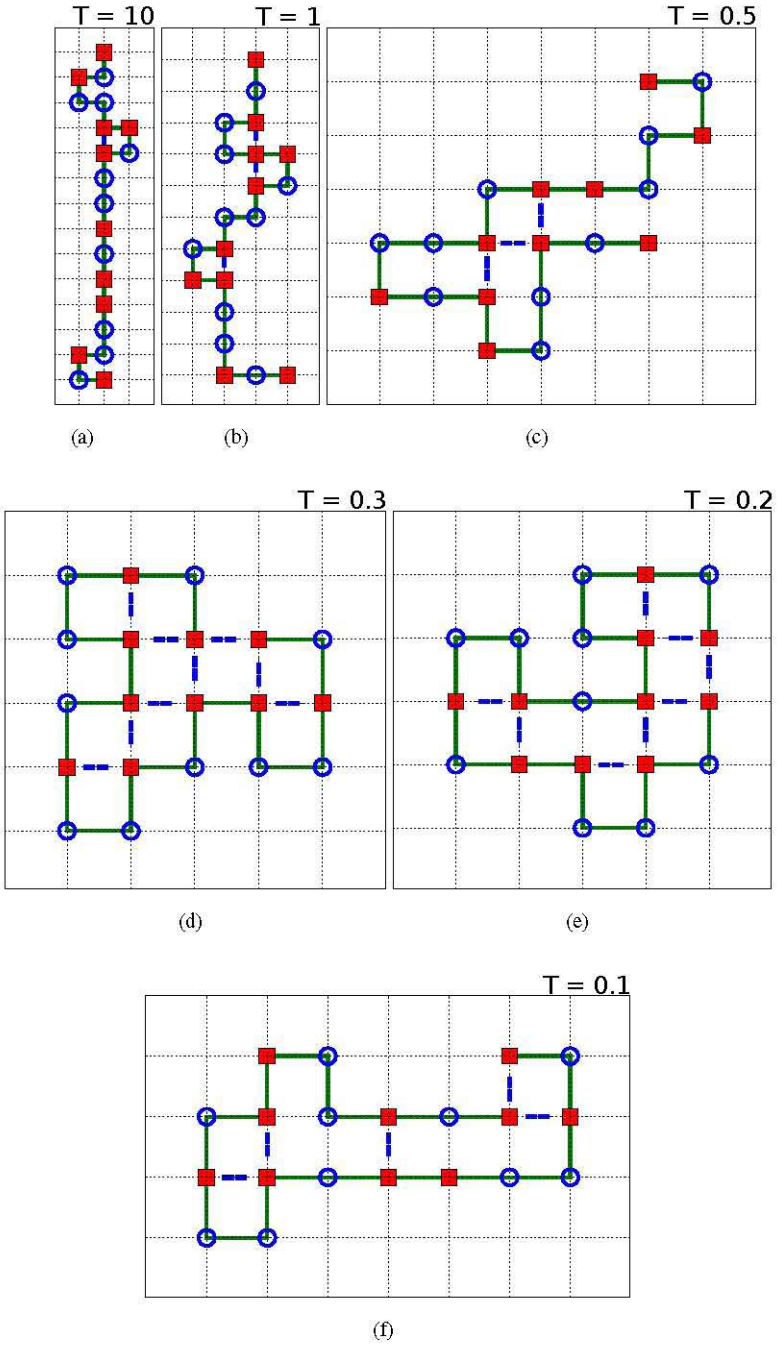


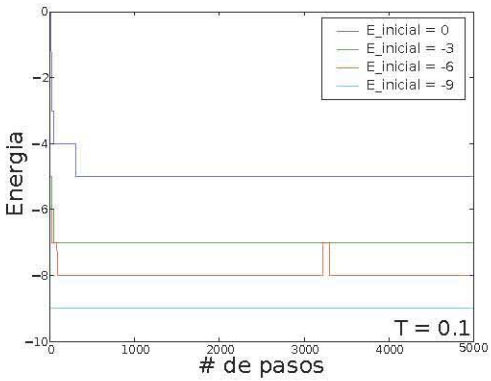
Figura 4.8: Estructuras finales de las simulaciones a diferentes temperaturas presentadas en la figura 4.7.

ción. Por esta razón, el sistema no logra deshacer contactos y explorar estructuras más intrincadas. Este es un claro ejemplo de lo que llamamos una estructura "congelada" en un mínimo local del paisaje de energía.

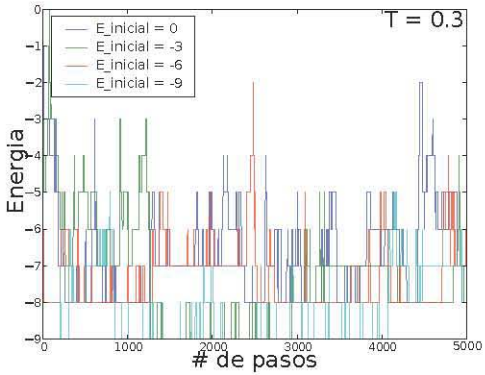
Otra observación que extraemos de analizar cualitativamente el comportamiento del polímero a diferentes temperaturas es la existencia de una cantidad de tiempo necesaria para alcanzar el equilibrio dinámico en función de la temperatura, que denotaremos $t_{\text{eq}}(T)$. Esperamos que $t_{\text{eq}}(T) \to \infty$ si $T \to 0$, y que tienda a 0 si $T \to \infty$.

Podemos estimar el tiempo de equilibrio a algún valor de la temperatura si corremos diferentes simulaciones independientes para una misma secuencia partiendo de diferentes configuraciones correspondientes a diferentes valores de E, y buscamos en cuánto tiempo alcanzan un equilibrio dinámico. De nuevo, utilizaremos la secuencia `sec20` para tres diferentes temperaturas, $T = 0.1$, 0.3 y 1. Para cada temperatura, corremos la dinámica del sistema utilizando el algoritmo de Metropolis para 4 réplicas de la cadena, cada una con una energía inicial diferente, $E = 0$, -3, -6 y -9.

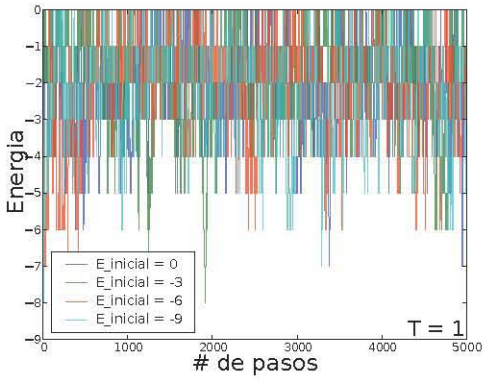
De la figura 4.9(a) observamos que para $T = 0.1$, en 5000 pasos, las diferentes corridas terminan claramente separadas entre sí. Nótese que la estructura que empezó en $E = -9$ nunca cambió de nivel de energía, mientras que las demás tendieron, en mayor o menor medida, hacia la energía mínima; por lo que podemos concluir que $t_{\text{eq}}(T = 0.1) \gg 5000$. Esto sugiere también que, como estudiaremos más adelante, la energía promedio en función de la temperatura, para simulaciones lo suficientemente largas, es $E_{\text{prom}}(T = 0.1) \sim -9$. Ahora, en la figura 4.9(b), podemos observar que, al final de los 5000 pasos a $T = 0.3$, las 4 réplicas del polímero tienen un comportamiento similar. La réplica cuya energía inicial era $E_{\text{inicial}} = -9$ termina con $E_{\text{final}} = -8$, la de $E_{\text{inicial}} = -6$ terminó con una $E_{\text{final}} = -6$, mientras que las otras dos réplicas, $E_{\text{inicial}} = -3$ y $E_{\text{inicial}} = 0$ terminaron ambas en $E_{\text{final}} = -7$. De esto, podemos concluir que $t_{\text{eq}}(T = 0.3) \sim 5000$. En una simulación un poco más larga, esperaríamos que las 4 réplicas tuvieran un comportamiento indistinguible a partir de ~ 5000 iteraciones, sin importar de qué estructura empezaron. De nuevo, estas observaciones sugieren que $E_{\text{prom}}(T = 0.3) \sim -7$. En la figura 4.9(c), a $T = 1$, observamos que prácticamente desde el comienzo de la simulación, el comportamiento de las cuatro réplicas es indistinguible. Lo cual sugiere que $t_{\text{eq}}(T = 1) \ll 5000$. En este caso, todas las réplicas terminaron en una energía final $E_{\text{final}} = -2$, por lo que podemos decir que $E_{\text{prom}}(T = 1) \sim -2$.



(a)



(b)



(c)

Figura 4.9: Diferentes corridas para diferentes temperaturas empezando de diferentes estados iniciales para la secuencia `sec20`.

4.1.4. Dinámica usando Metropolis para una cadena de 64 residuos

Comencemos por realizar simulaciones usando el algoritmo de Metropolis como primera aproximación. Elegimos una secuencia "benchmark", es decir bien estudiada, de 64 residuos: HHHHHHHHHHHHPHPHPPHHPPHHPPHPPHHPPHHPPHPPHHPPHHPPHPHPH HHHHHHHHHHHH, tomada directamente de la referencia [43]. Esta secuencia tiene 42 residuos hidrofóbicos y 22 residuos polares, es decir, tiene $\sim 65\,\%$ de residuos hidrofóbicos. En adelante, nos referimos a ella como `sec64`. Haciendo uso de la información exhibida en la figura 4.5, calculamos que una simulación de Metropolis de 10^6 de iteraciones para una cadena completamente hidrofóbica de longitud 64 llevará unas 28 horas en realizarse. Esta secuencia es lo suficientemente larga como para exhibir un comportamiento interesante y ser imposible de tratar mediante enumeración exhaustiva, pero no lo tanto como para hacer las simulaciones demasiado largas; de hecho, en [43], se menciona que esta cadena ha sido catalogada como difícil de plegar por varios otros grupos a los cuales se hace referencia; por difícil de plegar se entiende que ha sido difícil encontrar el estado de mínima energía $E = -42$ utilizando una variedad de métodos. Además, por ser una secuencia comúnmente estudiada, se conocen buenas aproximaciones a su densidad de estados en la red cuadrada, lo cual nos permite comparar ciertos resultados de nuestras simulaciones con las cantidades esperadas de esta secuencia. Otro punto de interés de esta secuencia es que sabemos que el estado de mínima energía $E = -42$ está escasamente poblado –existen escasas configuraciones del sistema que correspondan a esta energía– con respecto a cualquier otro estado de energía, al menos en el caso de la red cuadrada. De la referencia [43], sabemos que la densidad de estados para $E = -42$ es dos órdenes de magnitud menor que para $E = -40$. Esto nos indica que esta secuencia tiene un paisaje de energía similar al de una proteína real, es decir, en forma de embudo, tal como se mencionó en la sección 2.3. En la referencia [43] se muestran 2 configuraciones de energía $E = -42$ para esta secuencia.

Ambas configuraciones tienen los residuos extremos rodeados por la estructura. La dificultad de alcanzar el estado nativo proviene del hecho de tener esta característica estructural. No se encontró información alguna referente a esta secuencia en una red triangular. Sin embargo, conforme a la discusión encontrada en la sección 2.5.1, esperamos que el mínimo de energía en la red triangular sea cercano al doble del valor mínimo de energía en la red cuadrada, es decir ~ -80. Uno de los objetivos del presente trabajo es determinar si esta misma secuencia, `sec64`, mantiene un comportamiento parecido al de las proteínas reales en una red triangular.

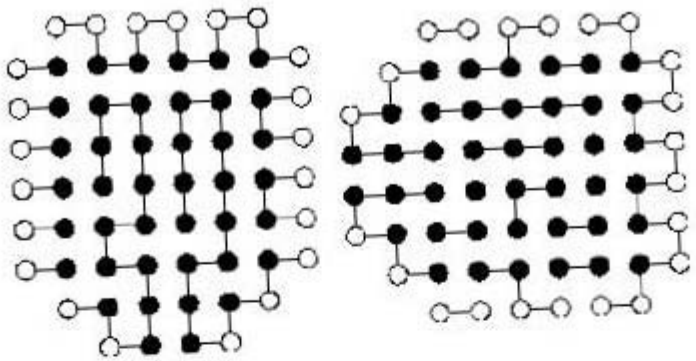


Figura 4.10: Configuraciones de energía $E = -42$ para la sec64 mostradas en la referencia [43]. Los círculos negros representan residuos hidrofóbicos y los blancos representan residuos polares.

En la figura 4.11 observamos una corrida de Metropolis a temperatura $T = 0.3$ para cada una de las redes. Se eligió simular al sistema a $T = 0.3$, debido a que este valor de la energía fue aquel que nos permitió alcanzar el valor mínimo de la energía en la sección anterior. En el caso de la red cuadrada, sabemos que el estado de mínima energía es $E = -42$, sin embargo, la menor energía alcanzada en esta simulación es $E = -36$. Observamos que después de las primeras 2×10^5 iteraciones, el sistema alcanza configuraciones cuyas energías fluctúan alrededor de $E \sim -35$ pero parece quedar "congelado" en este *mínimo local* de la energía sin poder hacer la transición hacia otros mínimos correspondientes a menores valores de la energía. Observamos a lo largo de la simulación que el sistema aumenta su energía deshaciendo contactos hidrofóbicos para intentar formar otros que le permitan alcanzar energías menores. El hecho de que el sistema se "congele" en un mínimo de energía local durante un tramo de la simulación no implica que la estructura permanezca idéntica. Para la red triangular, observamos que pasando las 6×10^5 iteraciones, la energía de las configuraciones obtenidas durante la simulación oscila alrededor $E \sim -57$, alcanzado el valor $E = -61$ al final de la simulación. Observamos que le lleva un mayor tiempo equilibrarse comparada con la simulación en la red cuadrada. A pesar de que no conocemos el valor de la energía mínima que puede alcanzarse con esta secuencia en una red triangular, esperamos que la energía mínima sea menor a $E = -61$, y más cercana al doble de la energía mínima en la red cuadrada, como se argumentó en la sección 2.5.1.

Si analizamos las estructuras correspondientes al mínimo de energía alcanzado

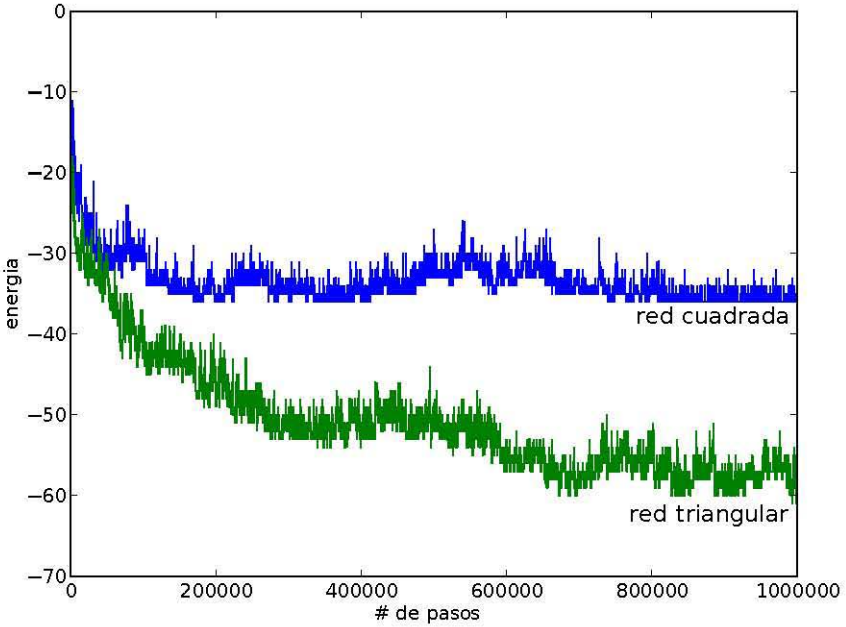
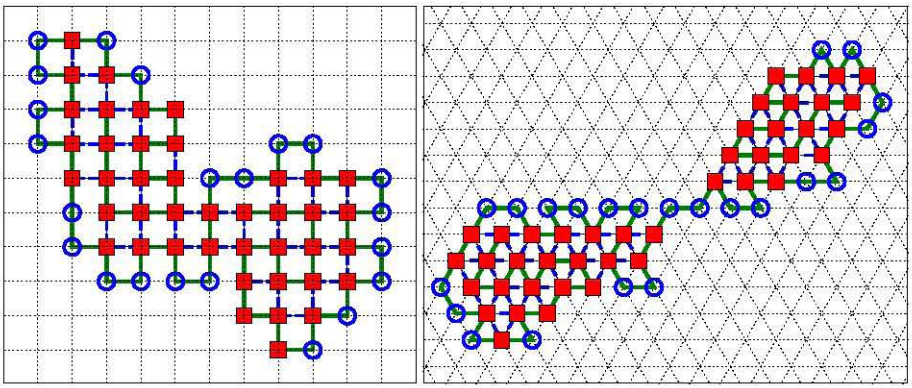


Figura 4.11: Evolución de la energía de las estructuras durante una simulación de Metropolis a $T = 0.3$ para la secuencia `sec64` en ambas redes.

durante las simulaciones de Metropolis, figura 4.12, observamos una característica similar en ambas estructuras. Podemos percatarnos de que en ambos casos –aunque es más claro en la red triangular– ambas estructuras constan de 2 cúmulos separados de residuos hidrofóbicos. Para alcanzar energías menores, estos cúmulos deberán juntarse y reacomodar sus residuos de modo que se maximicen los contactos hidrofóbicos. Esto implica deshacer cierta cantidad desconocida de contactos ya formados, lo cual pudiera no ocurrir debido a que no se dispone de la energía térmica necesaria o el tiempo suficiente. De las figuras 4.11 y 4.12, podemos suponer que el sistema termina la simulación atrapado en un mínimo local del cual no logra salir para después alcanzar el mínimo global del paisaje de energía.

Como podemos ver de los resultados obtenidos mediante simulaciones usando el algoritmo de Metropolis, este no nos permite una exploración adecuada del espacio de energías del sistema. Observamos que el sistema se "congela" fácilmente en estados metaestables correspondientes a mínimos locales en el paisaje de energía.



(a) (b)

Figura 4.12: Ejemplos de configuraciones de mínima energía de las simulaciones de la figura 4.11. La energía de las estructuras es $E = -36$ para la red cuadrada y $E = -61$ para la red triangular.

4.1.5. Recocido simulado

Una manera de explorar más uniformemente el paisaje de energía del sistema es utilizar la técnica de *recocido simulado* que se explicó en el capitulo 3. Al comenzar la simulación a temperaturas altas, se espera que el sistema pueda obtener la energía suficiente para sobrepasar aquellas barreras de energía libre, que separan a los mínimos locales del mínimo global, que no pudieron ser cruzadas en un principio mediante una simulación de Metropolis a temperatura fija. Conforme disminuye gradualmente la temperatura, se espera que el sistema abandone el caracter indiscriminado que caracteriza la exploración del paisaje de energía con el algoritmo de Metropolis a altas temperaturas, y tienda hacia un mínimo de la energía, idealmente el mínimo global, ya que al alcanzar bajas temperaturas, el sistema quedará atrapado.

En la figura 4.13, vemos la evolución de la energía del sistema a lo largo de una simulación de recocido simulado usando el algoritmo de Metropolis. Inicialmente la temperatura tiene un valor de $T_{\text{inicial}} = 1.01$; cada 1000 iteraciones, disminuimos el valor actual de la temperatura tal que $T_{\text{siguiente}} = T_{\text{actual}} - \Delta T$, con $\Delta T = 0.001$.

Lo primero que observamos es que, al final de la simulación, el valor de mínima energía alcanzado para cada una de las redes es menor que en el caso de las simulaciones de Metropolis sencillas. En el caso de la red cuadrada, se alcanzó un valor mínimo de $E = -40$ durante algunas iteraciones. Recordemos que el valor de mínima energía

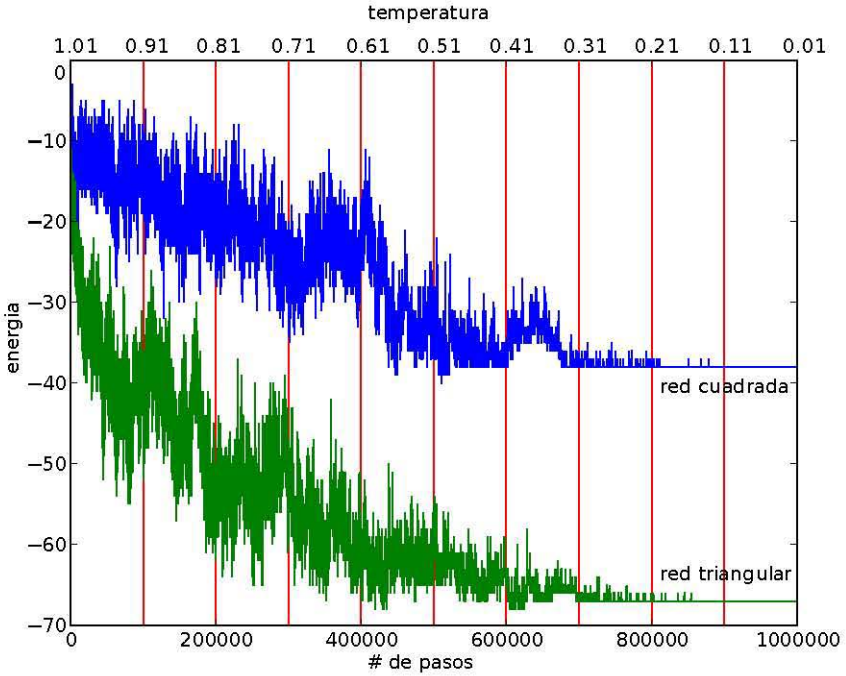


Figura 4.13: Evolución de la energía de las estructuras alcanzadas durante una simulación de recocido simulado usando el algoritmo de Metropolis con $T_{\text{inicial}} = 1.01$, $T_{\text{final}} = 0.01$ y $\Delta T = 0.001$ cada 1000 iteraciones por valor de T, para la secuencia `sec64` en ambas redes. El valor de la temperatura se indica en la parte superior de la figura.

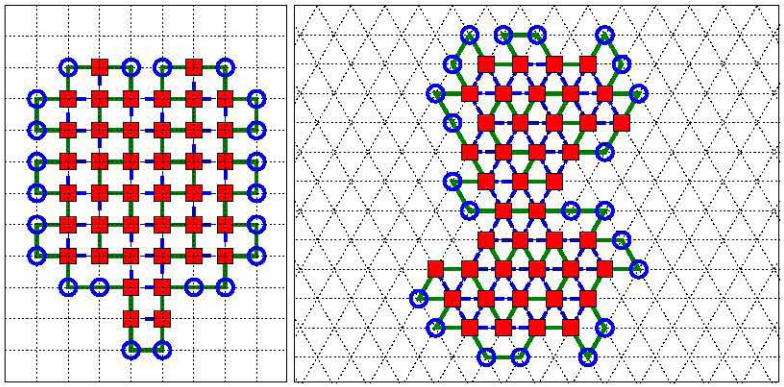
reportado para esta secuencia es $E = -42$, por lo que aún haciendo uso de la técnica de recocido simulado, no se logró llegar al mínimo global del paisaje de energía para la secuencia `sec64`. Este hecho sugiere que el mínimo global es un estado difícil de acceder y separado de los estados inmediatos a él por una gran barrera de energía libre. Si hubiésemos de imaginar esta sección del paisaje de energía, observaríamos una última gran rugosidad antes del punto mínimo del embudo. Sabemos además, de la referencia [43], que el estado correspondiente a $E = -42$ en la red cuadrada está escasamente poblado, dos ordenes de magnitud menor que el estado $E = -40$.

En el caso de la red triangular, la mínima energía alcanzada fue $E = -68$, que se alcanzó en repetidas ocasiones durante la simulación. Se obtuvieron energías menores a la mínima obtenida durante la simulación de Metropolis normal en los primeros

2×10^5 pasos.

En ambos casos, el sistema terminó la simulación "congelado" en un estado de mayor energía a la mínima alcanzada. Considerando el caso de la red cuadrada, observamos que la mínima energía alcanzada fue visitada en una sola ocasión a lo largo de la simulación. Esto parecería sugerir que dicho estado es difícil de acceder a partir de los estados cuyas energías sean cercanas, pero superiores, a la correspondiente a este. Dicho de otro modo, este estado puede ser alcanzado solamente a partir de un pequeño conjunto específico de estructuras con energía inmediatamente superior a aquélla exhibida por dicho estado. Encontrar la "ruta" correcta de plegamiento es, podríamos pensar, equivalente a buscar una aguja en un pajar. Sin embargo, observemos que la mínima energía alcanzada se visitó alrededor de la iteración 5×10^5. En el momento de la visita, el valor de la temperatura era $T \sim 0.5$, lo cual nos lleva a pensar que el sistema cambió de "ruta" de plegamiento debido a las fluctuaciones térmicas, y terminó "congelándose" en un estado de mayor energía sin antes encontrar su camino de vuelta al estado $E = -40$. Esto nos sugiere que la "ruta" de plegamiento, desde el estado completamente extendido hasta el estado nativo, se hace "estrecha", es decir que al ir de un estado de energía E_+ a uno de energía menor E_- hay una gran diferencia en la entropía configuracional de ambos estados. Estas observaciones apoyan la noción del paisaje de energía en forma de embudo. Considerando ahora el caso de la red triangular, y tomando en cuenta lo que discutimos anteriormente, podríamos aventurar dos hipótesis, tomando en cuenta que la mínima energía alcanzada durante la simulación fue visitada en repetidas ocasiones. La primera es que el estado correspondiente al valor de la energía $E = -68$ es aún muy elevado respecto al valor del mínimo global de energía, y por lo tanto, existe una gran diversidad estructural y la "ruta" de plegamiento no es tan estrecha aún, lo cual hace que no sea tan complicado acceder a dicho estado. La segunda teoría es que esta secuencia, `sec64` no tenga un comportamiento similar al de las proteínas reales en la red triangular, es decir, que su paisaje de energía no sea un embudo, básicamente. O lo que es equivalente, que la "ruta" de plegamiento no se "estreche" al disminuir la energía. Con las herramientas proporcionadas por los resultados de las simulaciones usando la técnica de recocido, no podemos ahondar más en estas hipótesis.

En la figura 4.14, se muestran ejemplos de configuraciones de la mínima energía alcanzada para ambas redes. Observamos que, para la red cuadrada, ya no es posible distinguir los 2 cúmulos de residuos hidrofóbicos que observábamos en la figura 4.12(a). Observamos que esta configuración tiene ambos extremos sumergidos dentro de la estructura, lo cual esperamos de la configuración de mínima energía. Para la red triangular, 4.14(b), distinguimos aún dos distintos dominios separados. En am-



(a) (b)

Figura 4.14: Ejemplos de configuraciones de mínima energía de las simulaciones de la figura 4.13. La energía de las estructuras es $E = -40$ para la red cuadrada y $E = -68$ para la red triangular.

bas configuraciones notamos que los residuos hidrofóbicos intentan formar un núcleo mientras que los residuos polares intentan acomodarse sobre la superficie de la estructura. Comparando la figura 4.14 con la figura 4.12, notamos que aquéllas con menor energía tienen sus residuos hidrofóbicos menos expuestos al solvente que aquéllas con mayor energía.

Estas simulaciones no proporcionan información termodinámica sobre el sistema debido a que cambiamos la temperatura durante la simulación, es decir, no se realiza en equilibrio térmico. Sin embargo, observamos que ayuda a sobrepasar barreras de energía libre sobre el paisaje de energías y así acercarse al mínimo global de manera más eficiente que con una simple simulación de Metropolis.

4.2. Wang–Landau

Como vimos en la sección anterior, el mínimo global de la secuencia `sec64` no pudo ser encontrado con simulaciones de Metropolis ni ayudándose de la técnica de recocido simulado. Una manera de explorar más eficientemente el paisaje de energía del sistema bajo estudio, y a la vez poder extraer información termodinámica del sistema, es utilizando el algoritmo Wang–Landau. Los detalles de este algoritmo se explican en la sección 3.6.

4.2.1. Validación de Wang–Landau

Antes de dedicar nuestros esfuerzos al algoritmo Wang–Landau para tratar a la secuencia que nos interesa, hacemos un paréntesis para comparar este algoritmo con el de Metropolis y darnos una idea de qué tanto mejora la manera de explorar el paisaje de energía.

Por motivos de brevedad, trabajaremos de nuevo con la secuencia `sec20`, además de que su densidad de estados exacta se conoce de la referencia [43]. Cabe mencionar que se implementó el código para enumerar exhaustivamente las configuraciones de cualquier secuencia, mediante el cual se intentó enumerar de dicha forma a la secuencia `sec20`. Sin embargo, resultó demasiado pesado computacionalmente y hubo que utilizar la densidad de estados de la referencia. El código para enumerar exhaustivamente fue utilizado para comparar los resultados del algoritmo Wang–Landau con las cantidades que se extraen de la densidad de estados exacta para otras cadenas de menor longitud a lo largo del desarrollo del trabajo, mas no se presentan aqui por motivos de brevedad.

Para hacer la validación, hacemos una corrida de Wang–Landau con $f_{\text{inicial}} = 1$, $f_{\text{final}} = 1 \times 10^{-5}$, con decrementos de $\frac{\log f}{4}$ cada que se cumple que el histograma sea plano, es decir, cada vez que cada energía previamente visitada se haya visitado al menos 100 veces en una misma iteración. Posteriormente, calculamos el calor específico y la energía promedio en función de la temperatura a partir del resultado de la simulación. Además, realizamos 10 conjuntos de corridas independientes con el algoritmo de Metropolis a diferentes temperaturas entre $T = 2.1$ y $T = 0.1$, con 1000 pasos iniciales para equilibrar y 10000 pasos siguientes para promediar la energía y su cuadrado, y calcular el valor del calor específico por cada temperatura.

En las figuras 4.15, 4.16 y 4.17 observamos respectivamente la entropía microcanónica, el calor específico por monómero y la energia promedio, extraídos de la densidad de estados exacta, aquéllos obtenidos mediante el algoritmo Wang–Landau, y los generados por 10 conjuntos independientes de corridas de Metropolis en un mismo intervalo de temperaturas. En las figuras se aprecia un comportamiento cualitativamente similar para ambos algoritmos, sin embargo, es claro que son los resultados obtenidos mediante el algoritmo Wang–Landau aquéllos que se asemejan más al valor exacto. En cambio, la aproximación mediante el algoritmo de Metropolis –aunque sea cualitativamente correcta– muestra desviaciones considerables del valor exacto a un mismo valor de la temperatura, en particular para temperaturas bajas.

Respecto a la forma de explorar el paisaje de energía, podemos observar en la

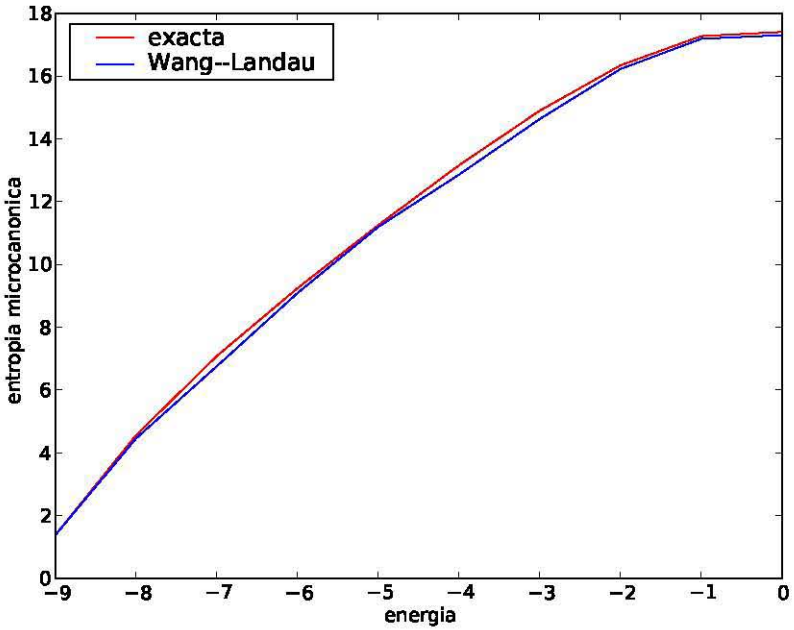


Figura 4.15: Comparación de la entropia microcanónica obtenida mediante el algoritmo Wang–Landau y la exacta, para la secuencia `sec20`.

figura 4.17 que en el caso de Metropolis, no siempre se alcanzó la configuración de mínima energía a temperaturas bajas; el último valor de la curva de Metropolis es superior a $E = -9$, lo cual nos indica que sólo en algunas de las 10 corridas se alcanzó la energía mínima. Contrastantemente, observamos que en la única simulación utilizando el algoritmo Wang–Landau se alcanzó la mínima energía.

En lo posterior, utilizaremos el algoritmo Wang–Landau para estudiar el proceso de plegamiento y otras características de la secuencia `sec64`, puesto que permite una exploración más uniforme y menos restringida del espacio de energía del sistema, además que de una única simulación podemos obtener toda la información termodinámica, y con mayor precisión que con el algoritmo de Metropolis.

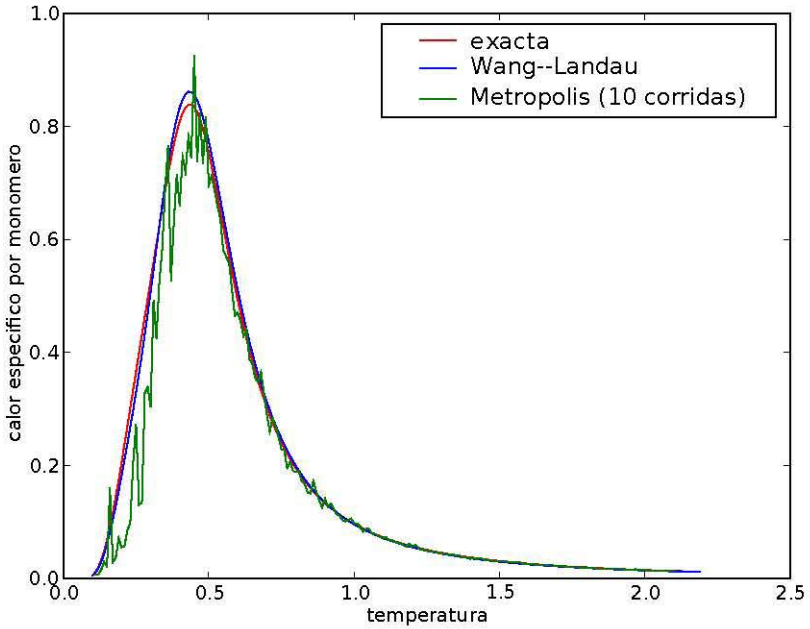


Figura 4.16: Calor específico por monómero en función de la temperatura para la secuencia `sec20`.

4.2.2. Resultados con Wang–Landau

Habiendo realizado el estudio preliminar del algoritmo Wang–Landau, podemos aplicarlo para el estudio de la secuencia que nos interesa, la secuencia `sec64`.

Para la simulación de Wang–Landau de la `sec64`, comenzamos con un valor inicial del factor de modificación $f_{\text{inicial}} = 1$. Cambiamos el valor actual del factor de modificación,

$$\log f_{n+1} = \frac{\log f_n}{4}, \tag{4.1}$$

cuando cada estado de energía previamente alcanzado haya sido visitado al menos 100 veces, es decir, cuando se cumpla la condición para checar si el histograma $H(E)$ ya es plano en esta iteración. Repetimos el proceso hasta alcanzar un valor de $f_{\text{final}} = 1 \times 10^{-5}$. Debido a la elección de estos valores para los parámetros de la simulación, realizamos 9 iteraciones por simulación, es decir, hacemos 8 refinamientos a la aproximación inicial de la densidad de estados del sistema. Podríamos ajustar

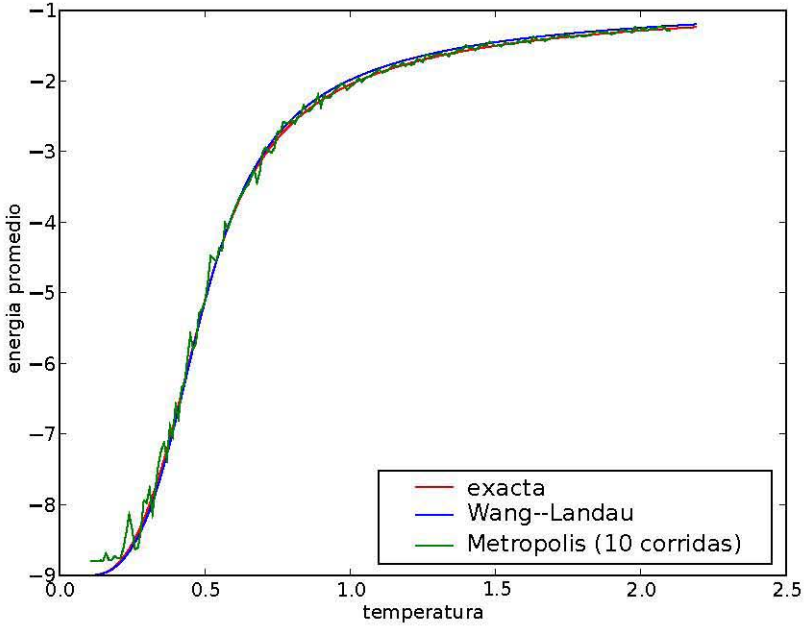


Figura 4.17: Energía promedio en función de la temperatura para la secuencia `sec20`.

los parámetros del sistema para correr más iteraciones, y por lo tanto, realizar más refinamientos a la aproximación inicial.

Cada simulación de Wang–Landau para la `sec64` tardó alrededor de una semana en correr, sin embargo, como se vio en la sección anterior, el correr una sola simulación nos da toda la información necesaria para reconstruir la termodinámica del sistema.

En la figura 4.18, observamos el resultado de la simulación realizada con el algoritmo Wang–Landau; sin embargo, esto no es aún la densidad de estados que estamos buscando, sino la entropía microcanónica, es decir, el logaritmo de la densidad de estados $S(E) = \log(g(E))$, como fue explicado en la sección 3.6. Observamos que en el caso de la red cuadrada, el sistema alcanzó el estado de mínima energía $E = -42$, mientras que, para la red triangular, la energía mínima alcanzada fue de $E = -75$. Sin conocimiento a priori del valor de la mínima energía posible para una secuencia, no podemos garantizar, por medio de los resultados de la simulación, que la mínima energía alcanzada en una simulación sea el mínimo global del paisaje de energía. Cabe

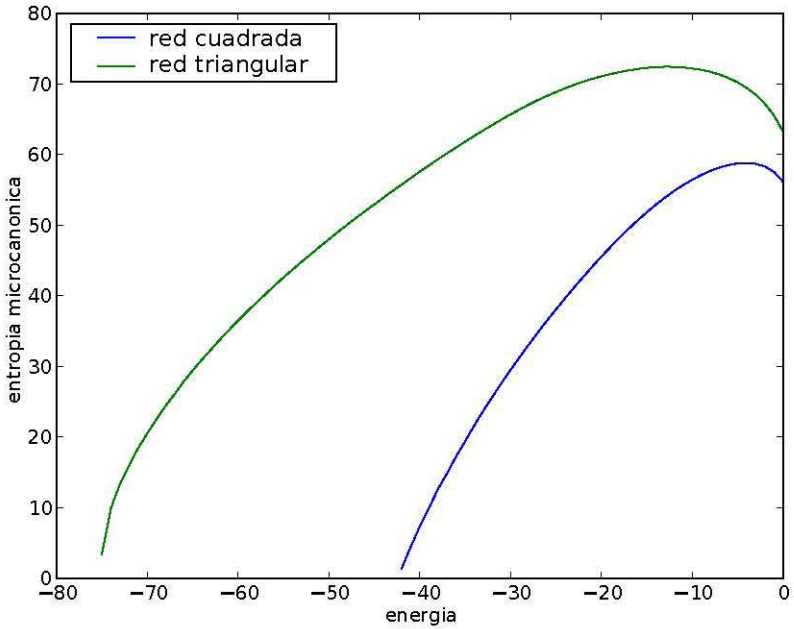


Figura 4.18: Perfil de entropía microcanónica obtenido mediante el algoritmo Wang–Landau para ambas redes.

mencionar que la entropía aquí presentada no es tal cual la que obtenemos directamente de la simulación Wang–Landau, sino que ha sido normalizada. Para normalizarla, restamos a cada entrada del arreglo S, donde guardamos la información de la entropía, el mínimo valor de la entropía y le sumamos el logaritmo del número de estructuras diferentes que encontramos en el estado de mínima energía. Puesto de otra forma, si j es la entrada tal que $S_j < S_i$ para todo valor $i \neq j$, normalizamos haciendo

$$S_i = S_i - S_j + \log(N_{\text{estructuras}}), \tag{4.2}$$

donde $N_{\text{estructuras}}$ corresponde al número de estructuras encontradas que exhiban la mínima energía.

Transformamos la entropía microcanónica obtenida de las simulaciones Wang–Landau para ambas redes, $S(E) = \log(g(E))$, en la densidad de estados, $e^{S(E)} = g(E)$.

En la referencia [43], se presenta la densidad de estados para esta misma secuencia, `sec64`, obtenida usando un algoritmo llamado *Equi-Energy Sampling*. Implemen-

taciones de este mismo algoritmo se comparan con implementaciones del algoritmo Wang–Landau, dando resultados concordantes, en la referencia [38]. Comparamos con la entropía microcanónica calculada de la densidad de estados encontrada en la primera referencia, puesto que en la segunda referencia no se da la lista de densidad de estados por energía. Dado que la densidad de estados reportada en [43] está normalizada, la transformamos escalándola a nuestros datos para compararla con la obtenida por nosotros mediante el algoritmo Wang–Landau.

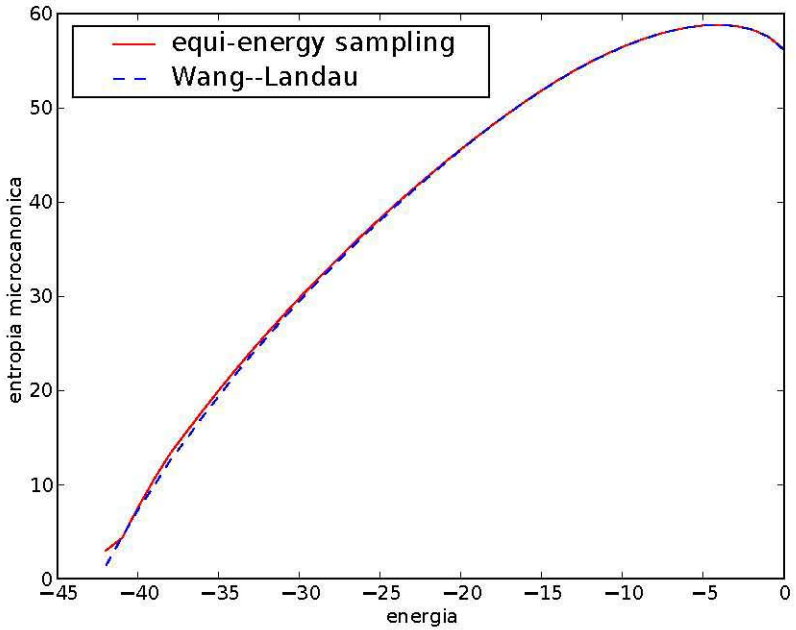


Figura 4.19: Comparación de la entropía microcanónica obtenida por el algoritmo de *equi-energy sampling* en la referencia [43] y el algoritmo Wang–Landau en nuestra implementación.

En la figura 4.19, observamos que nuestra implementación del algoritmo Wang–Landau ofrece resultados muy similares a aquéllos obtenidos por Kou, Oh y Wong [43], que a su vez, son igualmente similares a los obtenidos por Wüst y Landau [38]. También en esta figura apreciamos que existe una pequeña desviación para las menores energías alcanzadas entre el algoritmo Wang–Landau en nuestra implementación, y el algoritmo *equi-enery sampling*. Esta misma desviación, sólo que inversa, se observa al comparar estos últimos resultados y aquéllos reportados por Wüst y Landau

en [38]. Esta comparación nos indica que las simulaciones que implementamos reproducen satisfactoriamente la densidad de estados esperada para este sistema, lo cual indica a su vez que el sistema muestrea de manera adecuada su paisaje de energía.

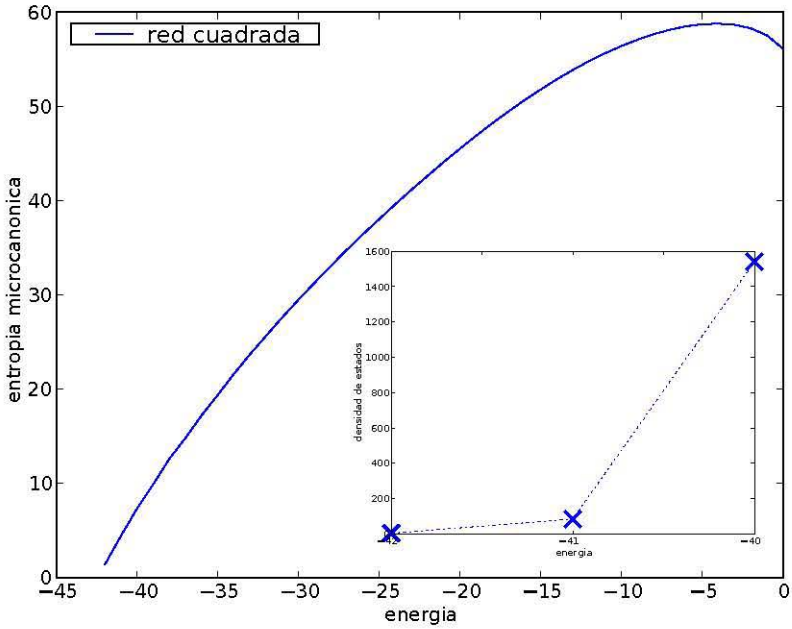


Figura 4.20: Entropía microcanónica para la `sec64` en la red cuadrada. El recuadro dentro de la figura muestra un acercamiento a la densidad de estados $g(E)$ para las menores energías.

En el recuadro de la figura 4.20, observamos una ampliación de la densidad de estados para las últimas energías; notamos que existe una separación de 3 órdenes de magnitud entre la densidad de estados en $E = -40$ y $E = -42$. De la referencia [43], esperábamos que la separación entre la densidad de estados para $E = -40$ y $E = -42$ fuera de 2 órdenes de magnitud. Sin embargo, notemos que en nuestra implementación, se encontraron 4 diferentes estructuras con energía $E = -42$. También hay que recordar que, en nuestra implementación, una estructura y la misma estructura rotada, reflejada o con el orden de monómeros invertido, no se consideran como la misma estructura. No es claro de la referencia [43], si ellos consideran de la misma manera que nosotros las rotaciones, reflexiones e inversión de una misma estructura. Se puede ver en la figura 4.19 que se encontraron un mayor número de estructuras con

energía $E = -42$ en la simulación de *Equi-Energy Sampling*, puesto que la curva correspondiente tiene un valor más alto de la entropia microcanonica, y por lo tanto, de la densidad de estados. Debido a que nosotros encontramos menos de 10 estructuras diferentes, la separación entre $E = -40$ y $E = -42$ es de 3 órdenes de magnitud en vez de 2.

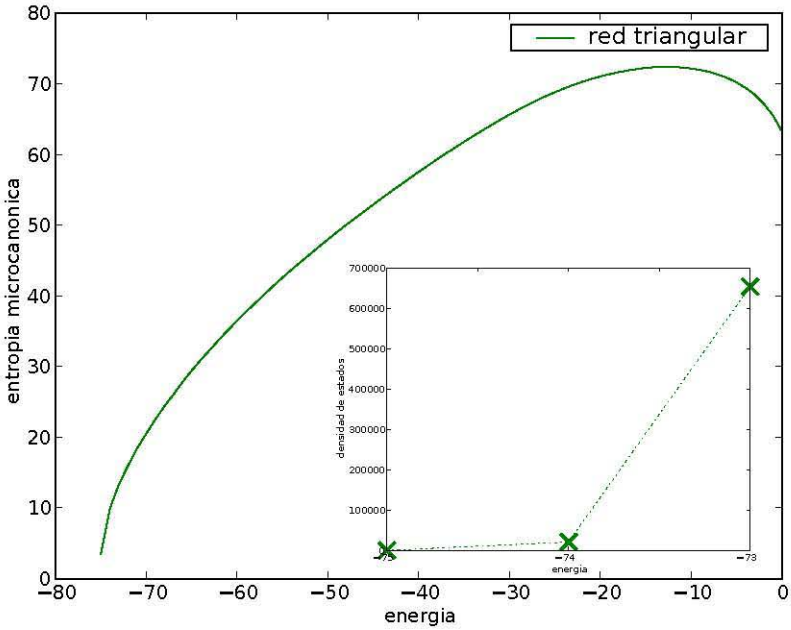


Figura 4.21: Entropía microcanónica para la `sec64` en la red triangular. El recuadro dentro de la figura muestra un acercamiento a la densidad de estados $g(E)$ para las menores energías. En el vemos que la densidad de estados crece muy rápido al dirigirse hacia energías mayores.

En el caso de la red triangular, como se muestra en la figura 4.21, notamos que la separación entre el estado de mínima energía alcanzado $E = -75$ y el estado $E = -73$ es de ~ 5 órdenes de magnitud. Se encontraron 31 diferentes estructuras correspondientes a este valor de la energía. Como se mencionó anteriormente, no hay manera de garantizar que la mínima energía alcanzada corresponda realmente al mínimo global del paisaje de energía. Inclusive, llevó algunos intentos alcanzar la energía $E = -75$ puesto que el sistema no llegaba a visitar dicho estado.

Como mencionamos existen diferentes configuraciones de la secuencia `sec64` que exhiben la energía mínima, pero trabajamos sólo con aquellas que encontramos mediante la simulación.

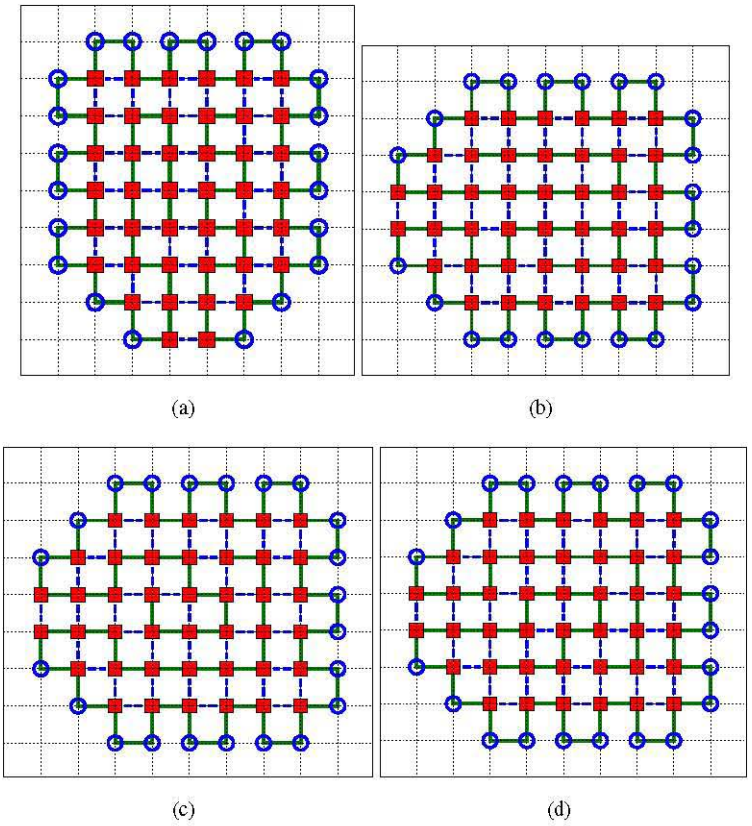


Figura 4.22: Ejemplos de configuraciones de mínima energía alcanzadas para la `sec64` en la red cuadrada.

En la figura 4.22 mostramos las 4 estructuras obtenidas correspondientes al mínimo valor de la energía para la secuencia `sec64`. Sabemos de la referencia [43], que este valor de la energía corresponde con el mínimo global del paisaje de energía para esta secuencia. En estas estructuras observamos un núcleo de residuos hidrofóbicos casi completamente aislado del solvente por los residuos polares. A primera vista, las estructuras parecen ser la misma, sin embargo, en una inspección cercana se distinguen sutiles cambios en el acomodo de la cadena. Notamos también que estas estructuras cumplen con que sus residuos extremos yacen sumergidos dentro de la estructura, tal como esperábamos de la referencia [43].

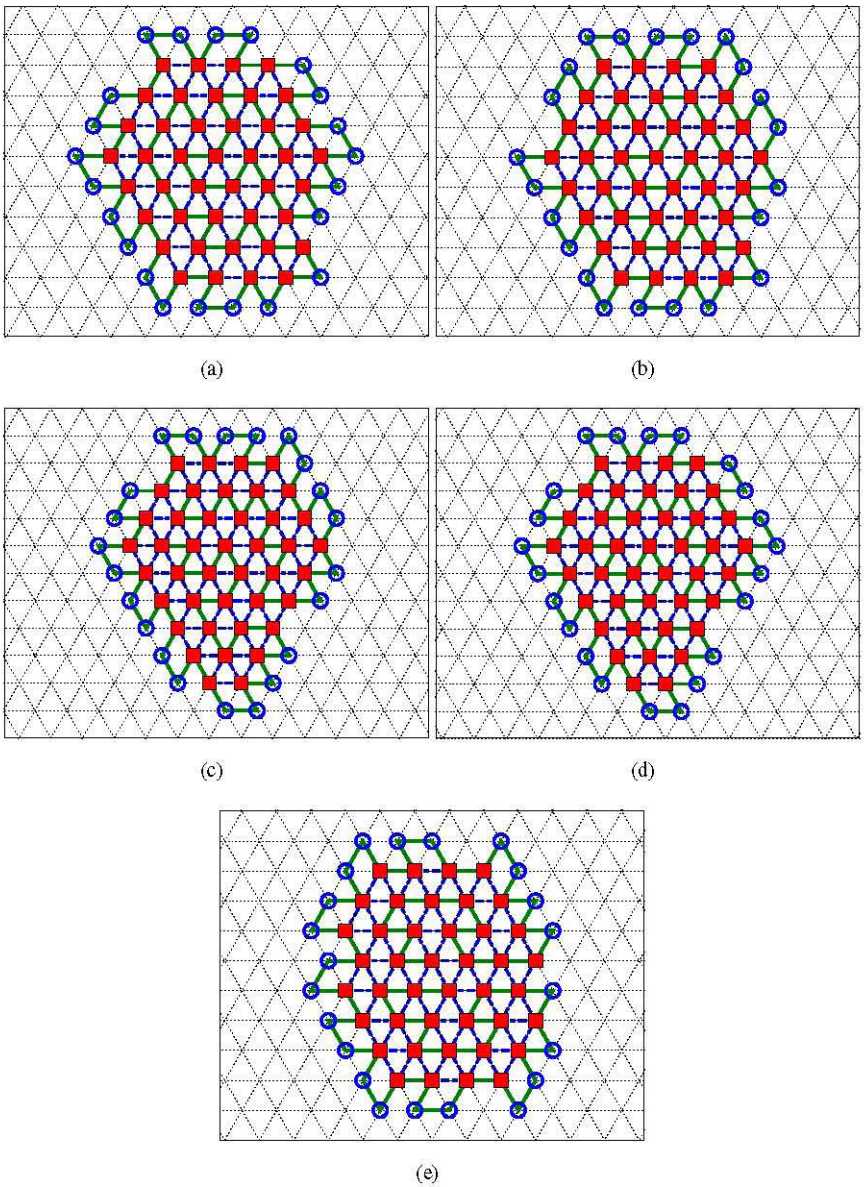


Figura 4.23: Ejemplos de configuraciones de mínima energía alcanzadas para la `sec64` en la red triangular.

En la figura 4.23 observamos 5 estructuras de las 31 que se encontraron con energía $E = -75$. Al igual que en la red cuadrada, observamos que estas estructuras presentan un núcleo hidrofóbico separado del solvente por los residuos polares. Observamos que para esta geometría, hay un menor número de residuos hidrofóbicos completamente aislados del solvente en el que se encuentra sumergido el polímero. Observamos que algunas de estas estructuras comparten el mismo núcleo hidrofóbico, con la única diferencia del acomodo de los residuos polares que lo rodean, figuras 4.23-(c) y -(d). Observamos también que el núcleo hidrofóbico puede tomar diferentes formas, figuras 4.23-(a) y -(c). Al igual que en el caso de la red cuadrada, ambos residuos extremos yacen sumergidos dentro de la estructura, lo cual sugiere la dificultad de alcanzar estas configuraciones.

4.3. Características de estructuras "plegadas"

A través de los resultados obtenidos de las simulaciones anteriores, podemos extraer información termodinámica del sistema para estudiar más a fondo su comportamiento y sus características. Además de esto, podemos también calcular propiedades de las estructuras alcanzadas durante las simulaciones que puedan proveernos información de interés sobre el proceso de plegamiento.

En la sección 3.7.1, discutimos la manera de extraer la información termodinámica de simulaciones de Metropolis y Wang–Landau. Sin embargo, en la sección 4.2.1, vimos que una sola simulación con el algoritmo Wang–Landau nos proporcionaba toda la información termodinámica del sistema y de manera más exacta que un conjunto de simulaciones con el algoritmo de Metropolis. Por esta razón los resultados que a continuación se presentan son sólo aquéllos calculados del resultado de una sola simulación con el algoritmo Wang–Landau.

4.3.1. Energía promedio

Calculamos la energía promedio para ambas simulaciones de acuerdo a lo establecido en la sección 3.7.1. De este modo, obtenemos el valor que tendrá en promedio la energía del sistema al encontrarse en algún valor de la temperatura.

En la figura 4.24, se muestra la energía promedio del sistema en función de la temperatura. Observamos que existe un punto de inflexión alrededor de $T = 0.5$ para la red cuadrada, mientras que para la red triangular observamos un punto de inflexión

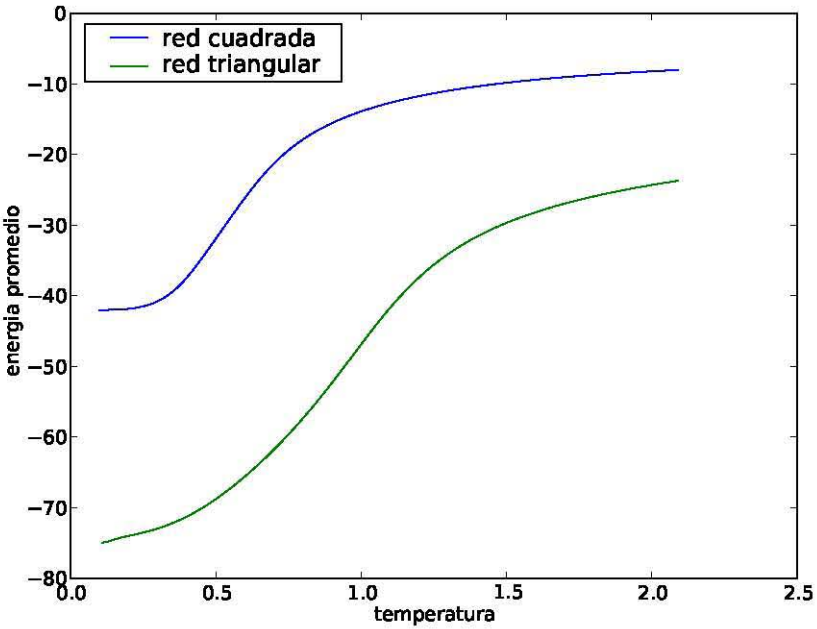


Figura 4.24: Energía promedio en función de la temperatura calculada del resultado de la simulación Wang–Landau para la secuencia `sec64` en la red cuadrada.

alrededor de $T = 1$. Físicamente este punto de inflexión indica que hay una transición en el comportamiento del sistema. Para temperaturas mayores a la temperatura de transición, el sistema tenderá a estar en un estado desordenado (no plegado), mientras que para temperaturas menores a esta temperatura de transición, el sistema tenderá a un estado ordenado (plegado).

Estas curvas de energía promedio en función de la temperatura nos permiten formar una idea de la dependencia térmica del proceso. Observamos que para un rango corto de temperaturas cercanas a las temperaturas que corresponden al punto de inflexión, hay un gran cambio en el valor de la energía promedio. Podemos decir que el comportamiento del sistema cambia drásticamente alrededor de estas temperaturas. Para temperaturas superiores a esta temperatura de transición, observamos que el sistema tiene una energía promedio alta en comparación con la del mínimo global, mientras que para temperaturas inferiores a la temperatura de transición, el sistema rápidamente tiende a energías promedio cada vez más cercanas al mínimo global.

4.3.2. Distribución de energías a una temperatura dada

Calculamos también la distribución de energías en función de la temperatura para el sistema, como se explicó en la sección 3.7.1. A través de esto, conocemos la distribución de energías para cualquier valor de la temperatura.

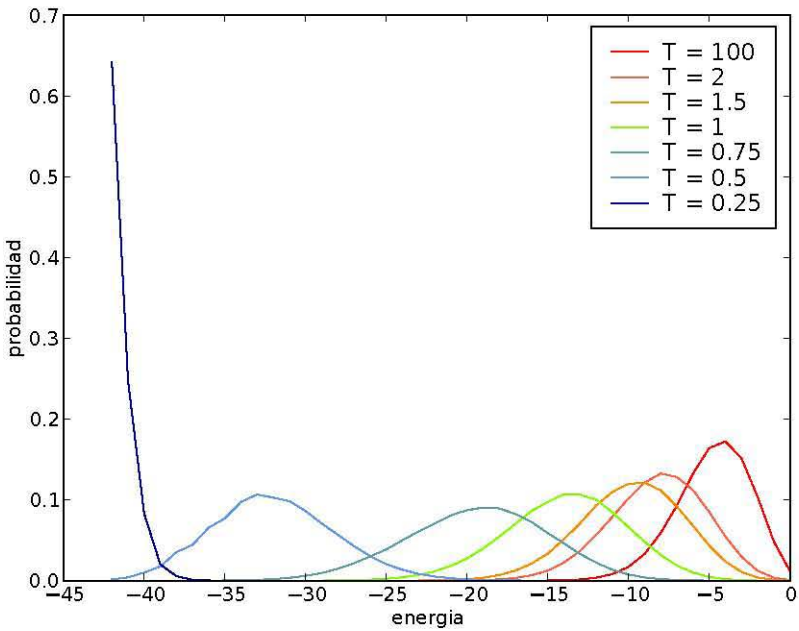


Figura 4.25: Distribución de energías en función de la temperatura para la secuencia `sec64` en la red cuadrada.

De las figuras 4.25 y 4.26, observamos cómo cambia la distribución de estados en energía del sistema al cambiar la temperatura. Observamos que a temperaturas altas, existe una mayor probabilidad de que el sistema se encuentre en una configuración correspondiente a una energía alta. Conforme disminuye la temperatura, la distribución se ensancha y el máximo se desplaza hacia energías menores. Sin embargo, existe una temperatura de transición, diferente en ambas redes, a partir de la cual, al disminuir la temperatura, la distribución se hace cada vez más angosta y el pico se mueve hacia las menores energías. Las temperaturas de transición se encuentran alrededor de $T = 0.5$ para la red cuadrada y alrededor de $T = 1$ para la triangular.

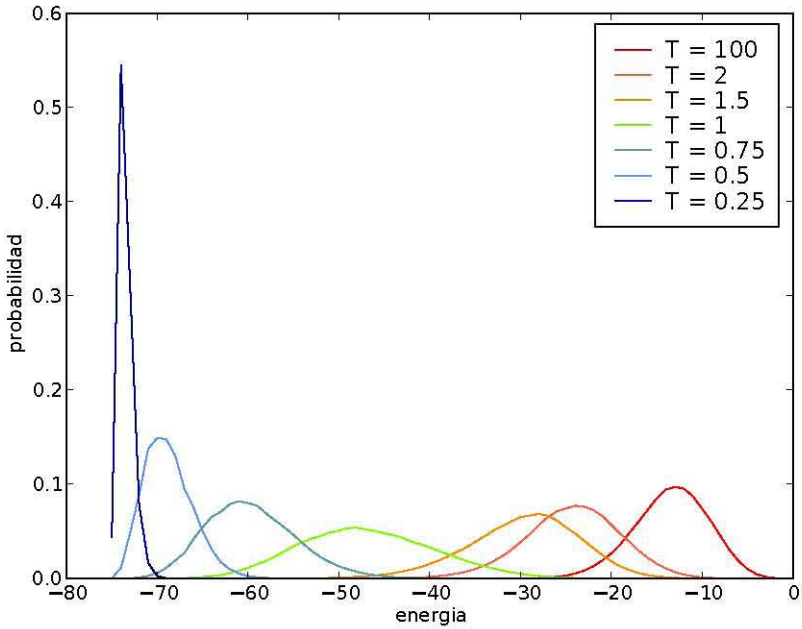


Figura 4.26: Distribución de energías en función de la temperatura para la secuencia `sec64` en la red triangular.

La información que observamos en las figuras 4.25 y 4.26, coincide con el análisis cualitativo del sistema realizado en la sección 4.1.3.

4.3.3. Calor específico

De igual manera a como se explicó en la sección 3.7.1, calculamos el calor específico del sistema a partir del resultado de una sola simulación con el algoritmo Wang–Landau para cada geometría de la red.

Observamos un pico en el calor específico por monómero en la temperaturas correspondientes al punto de inflexión de las curvas en la figura 4.24. Estas temperaturas representan una temperatura de transición en el comportamiento del sistema.

En las figuras 4.27 (b) y (c), y 4.28 (b) y (c), se muestran estructuras típicas para la secuencia `sec64` a temperaturas por debajo y por encima de la temperatura de transición. En ambos casos, tomamos inicialmente una estructura correspondiente a

la energía promedio del sistema para dos valores de la temperatura, uno por encima y otro por debajo de la temperatura de transición. Después de esto, se dejó evolucionar al sistema por medio de simulaciones de Metropolis de 1000 pasos, y se graficó la estructura obtenida al final de la simulación. Para la red cuadrada, las temperaturas escogidas para las simulaciones fueron $T = 0.2$, 4.27 (b), y $T = 0.8$, 4.27 (c), para las cuales las energías promedio son $\langle E \rangle \sim -41$ y $\langle E \rangle \sim -18$. Para la red triangular, las temperaturas escogidas para las simulaciones fueron $T = 0.5$, 4.28 (b), y $T = 1.5$, 4.28 (c), para las cuales las energías promedio son $\langle E \rangle \sim -68$ y $\langle E \rangle \sim -31$. En ambos casos observamos que para la temperaturas menores a la temperatura de transición, tenemos estructuras compactas, mientras que para las temperaturas mayores a la temperatura de transición, tenemos estructuras extendidas.

A través de las cantidades termodinámicas estudiadas podemos determinar una temperatura de transición para el sistema a partir de la cual, si disminuimos la temperatura, obtenemos configuraciones "plegadas", y si aumentamos la temperatura, obtenemos configuraciones "desplegadas".

4.4. Cuadrada vs. triangular: ¿qué red es más realista?

A través de los resultados expuestos en este capítulo y las discusiones que de ellos se desprenden, buscamos poder determinar cuál de las dos geometrías de la red sobre la cual se simula la proteína-modelo tiene un comportamiento más realista. Por comportamiento realista nos referimos a comportamiento similar al que se espera de una proteína. En la sección 2.3, se discutió brevemente qué cualidades se esperan de una secuencia modelo para decir que exhibe un comportamiento parecido al de las proteínas. Lo que determina si el comportamiento es parecido al de las proteínas es que haya una marcada separación entre el estado nativo y cualquier otro estado, principalmente los mínimos locales circundantes a éste. La secuencia `sec64` fue diseñada con este propósito para la red cuadrada; sin embargo, no es evidente que esta cualidad se conserve al transportarla a otras geometrías. Además de esto, buscamos que la estructura nativa sea única.

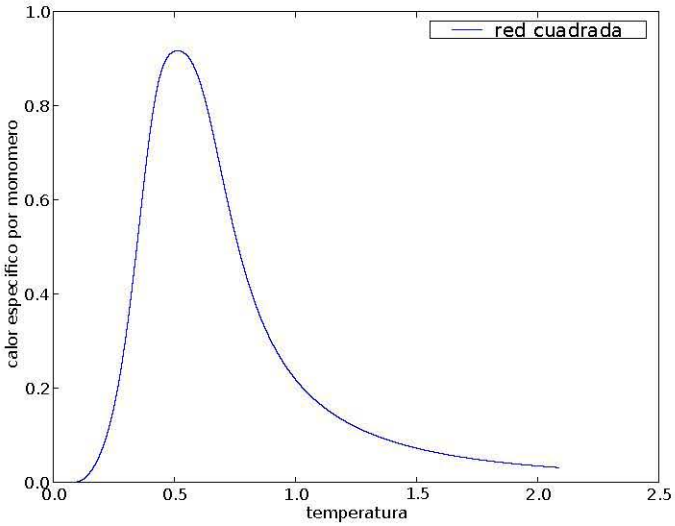
Volviendo a la figura 4.20, y fijándonos en el recuadro donde se muestran los valores obtenidos para la densidad de estados en la red cuadrada para la secuencia `sec64`, observamos que hay una diferencia de ~ 3 órdenes de magnitud con el estado $E = -40$ y de ~ 2 órdenes de magnitud con el estado $E = -41$ para el estado de

mínima energía $E = -42$. En la figura 4.21, observamos una separación de ~ 3 órdenes de magnitud con el estado contiguo $E = -74$ para el estado de mínima energía alcanzada $E = -75$. Estos resultados muestran que el mínimo global de la secuencia en la red cuadrada y el que suponemos que es el mínimo global de la misma secuencia en la red triangular se encuentran marcadamente separados de cualquier otro estado del sistema, en particular de los más cercanos. Notamos que existe más o menos la misma separación entre el mínimo encontrado en la red triangular y los estados inmediatos y entre el mínimo global de la red cuadrada y sus correspondientes estados contiguos. La magnitud de la separación entre un par de estados está directamente relacionado con la estabilidad del mismo. Sin embargo, también observamos que entre mayor sea esta separación, será más difícil alcanzar el estado de menor energía en una simulación.

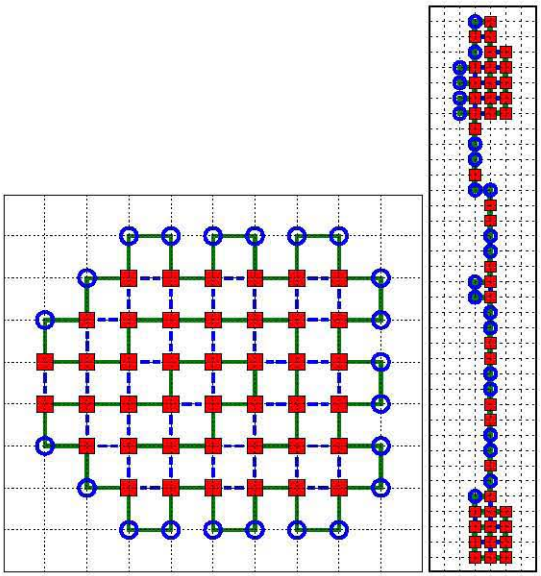
Para la red cuadrada, se encontraron 4 estructuras con la energía mínima $E = -42$, mientras que para la red triangular se encontraron 31 con la energía mínima $E = -75$. Es claro que, debido a que consideramos las rotaciones, reflexiones e inversiones de una estructura como estructuras diferentes entre sí, el estado nativo no constará de una única estructura. Además, debido al nivel de grano-grueso del modelo, esperamos que haya cierta degeneración de estados en el mínimo global. Sin embargo, analizando las estructuras presentandas en las figuras 4.22 y 4.23, nos damos cuenta de que hay una diferencia entre las estructuras encontradas para diferentes geometrías. Las estructuras de la red cuadrada tienen todas una forma similar, a pesar de que los detalles del núcleo varían en cada una. El acomodo de la capa hidrofílica de las estructuras en esta geometría es la misma en todos los casos. En contraste, las estructuras de la red triangular parecen poder adoptar diferentes formas globales, sin fijarse detenidamente en la estructura detallada, como se mencionó en la sección 4.2.2. También observamos que existen diferentes estructuras entre las cuales la única diferencia es la orientación de ciertos pedazos la capa hidrofílica de la estructura, a pesar de compartir un mismo acomodo de los residuos hidrofóbicos, como se ve en la figura 4.23(c) y (d). El modelo que utilizamos permite que haya esta flexibilidad a la hora de acomodar los residuos polares; sin embargo, ligeros cambios conformacionales podrían repercutir seriamente en la funcionalidad de una proteína real.

Esta secuencia, sec64, parece tener una separación similar entre el estado de mínima energía encontrado y aquél inmediatamente superior, en ambas redes; por otro lado, las estructuras obtenidas para la misma secuencia en la red cuadrada parecen ser más consistentes entre sí que las obtenidas en la red triangular. De acuerdo a los criterios establecidos para juzgar qué red exhibe un comportamiento más realista, podemos decir que ambas redes presentan un comportamiento realista, ya que en ambas

la secuencia exhibe un mínimo de energía estable y accesible, necesario para que se pliegue. Sin embargo, observamos una mayor degeneración de estructuras correspondientes a este mínimo para la red triangular, en donde los cambios no sólo son referentes a rotaciones, reflexiones o inversiones, sino a cambios en la forma global de la estructura y cambios en la orientación de los residuos polares que forman la superficie de contacto con el solvente. Debido a esto, podemos decir que este modelo simple presenta un comportamiento más realista en una red cuadrada puesto que conduce a estructuras más afines entre sí, que aquéllas obtenidas en la red triangular.

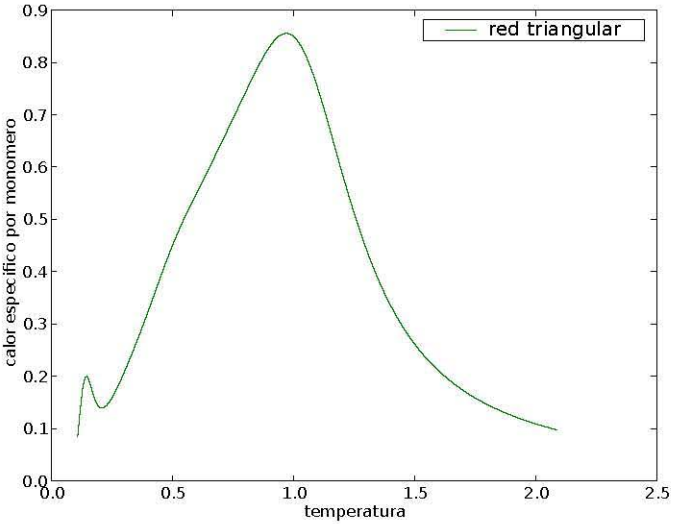


(a)

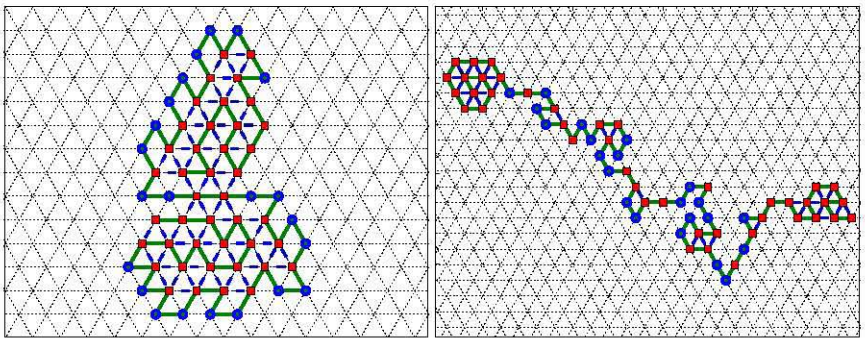


(b) (c)

Figura 4.27: Calor específico por monómero en función de la temperatura calculada del resultado de la simulación Wang–Landau para la secuencia `sec64` en la red cuadrada (a). Estructuras típicas encontradas para la secuencia `sec64` en la red cuadrada a temperaturas $T = 0.2$ (b) y $T = 0.8$ (c).



(a)



(b) (c)

Figura 4.28: Calor específico por monómero en función de la temperatura calculada del resultado de la simulación Wang–Landau para la secuencia `sec64` en la red triangular (a). Estructuras típicas encontradas para la secuencia `sec64` en la red cuadrada a temperaturas $T = 0.5$ (b) y $T = 1.5$ (c).

Capítulo 5

Conclusiones

A partir del modelo HP, se estudió el proceso de plegamiento de proteínas, mediante simulaciones con diferentes algoritmos Monte Carlo. Con el algoritmo de Metropolis, buscamos la configuración de mínima energía del sistema. Sin embargo, se observó que haciendo uso de este algoritmo, el sistema realiza un muestreo muy restringido de su paisaje de energía. Además, es muy susceptible a llevar al sistema a mínimos locales de los cuales no logra salir. Para ayudar al sistema a explorar de manera más libre su paisaje de energía, implementamos el algoritmo de Metropolis junto con la técnica de recocido simulado, que además ayuda al sistema a no verse atrapado en mínimos locales al ir disminuyendo gradualmente la temperatura. A pesar de esto, nunca se logró obtener la configuración de mínima energía en ninguna de las redes con estos métodos. Utilizamos el método Wang–Landau para obtener aproximaciones a la densidad de estados del sistema. Este método permite al sistema muestrear de manera más libre su espacio de energía, logrando visitar el intervalo completo de estados de energía.

A partir de los resultados obtenidos mediante el algoritmo Wang–Landau en ambas redes, calculamos el calor específico y la energía promedio de la secuencia en función de la temperatura. Se encontró una temperatura de transición, por encima de la cual, el sistema busca estar en configuraciones desplegadas, y por debajo de la cual, el sistema tiende a encontrarse en configuraciones plegadas.

Finalmente, comparando los resultados de las simulaciones en ambas redes, concluimos que, al menos para esta secuencia, la red cuadrada exhibe un comportamiento más adecuado a lo que se espera del comportamiento de una proteína, en el marco

teórico en el que se considera.

Trabajo futuro

Como trabajo a futuro en continuación a esta tesis, se establecen los siguientes puntos como aquéllos de mayor interés:

- Estudiar la diversidad estructural en las "mesetas" de energías similares que se observan en los resultados del algoritmo de Metropolis cuando el sistema queda "congelado". Buscar elementos comunes en estas estructuras y cuantificar qué tan similares o diferentes son las estructuras en un mismo mínimo local del paisaje de energía. A su vez, esto permitiría caracterizar con mayor detalle el paisaje de energía del sistema y estudiar la diversidad estructural en distintas "rutas" de plegamiento.
- Implementar el modelo en redes no regulares o regulares pero que permitan una mayor diversidad en el valor de los ángulos que puede formar la estructura de la cadena. Para estudiar esto, se sugiere primero modificar el modelo para restringir el movimiento indiscriminado de los residuos polares, tal como observamos para las estructuras finales en el caso de la red triangular.

Bibliografía

[1] D.L. Nelson and M.M. Cox. *Lehninger Principles of Biochemistry*. W. H. Freeman, fourth edition, 2004.

[2] K. A. Dill, S. B. Ozkan, M. S. Shell, and T. R. Weikl. The protein folding problem. *Annual Review of Biophysics*, 37:289–316, 2008.

[3] L.C. Walker and H. LeVine. Neurodegenerative disorders of protein conformation and assembly. *Molecular Neurobiology*, 21(1/2):83–95, 2000.

[4] M.D. Scott and J. Frydman. Aberrant protein folding as the molecular basis of cancer. *Methods in Molecular Biology*, 232:67–76, 2003.

[5] C.B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181(96):223–230, 1973.

[6] C.B. Anfinsen and H.A. Sheraga. Experimental and theoretical aspects of protein folding. *Advances in Protein Chemistry*, 29:205–300, 1975.

[7] K.A. Dill. Dominant forces in protein folding. *Biochemistry*, 29:7133–7155, 1990.

[8] M.H.J. Cordes, A.R. Davidson, and R.T. Sauer. Sequence space, folding and protein design. *Current Opinion on Structural Biology*, 6(1):3–10, 1996.

[9] C. Levinthal. Are there pathways for protein folding? *Journal de Chimie Physique et de Physico-Chimie Biologique*, 65(44-45), 1968.

[10] C.J. Bond et al. Characterization of residual structure in the thermally denatured state of barnase by simulation and experiment: description of the folding pathway. *Proceedings of the National Academy of Science USA*, 94:13409–13413, 1997.

[11] M. Karplus and D.L. Weaver. Diffusion-collision model for protein folding. *Biopolymers*, 18:1421–37, 1979.

[12] A.R. Fersht. Nucleation mechanisms in protein folding. *Current Opinion in Structural Biology*, 7:3–9, 1997.

[13] R.H. Callender, R.B. Dyer, R. Gilmanshin, and W.H. Woodruff. Fast events in protein folding: the time evolution of primary processes. *Annual Review of Physical Chemistry*, 49:173–202, 1998.

[14] J.D. Bryngelson, J.N. Onuchic, N.D. Socci, and P.G. Wolynes. Funnels, pathways, and the energy landscape of protein folding: A synthesis. *Proteins: Structure, Function, and Genetics*, 21(3):167–195, 1995.

[15] P.E. Leopold, M. Montal, and J.N. Onuchic. Protein folding funnels: A kinetic approach to the sequence-structure relationship. *Proceedings of the National Academy of Science USA*, 89(8721-8725), 1992.

[16] V.Sharma, V.R. Kaila, and A. Annila. Protein folding as an evolutionary process. *Physica A*, 388(6):851–862, 2008.

[17] E.D. Nelson and J.N. Onuchic. Proposed mechanism for stability of proteins of evolutionary mutations. *Proceedings of the National Academy of Science USA*, 95(18):10682–6, 1998.

[18] P.G. Wolynes. Symmetry and the energy landscapes of biomolecules. *Proceedings of the National Academy of Science USA*, 93(25):14249–55, 1996.

[19] P.G. Wolynes and J.D. Bryngelson. Spin glasses and the statistical mechanics of protein folding. *Proceedings of the National Academy of Science USA*, 84:7524–7528, 1987.

[20] E. Shakhnovich. Protein folding thermodynamics and dynamics: Where physics, chemistry, and biology meet. *Chemical Reviews*, 106(5):15591588, 2006.

[21] C. Clementi. Coarse-grained models of protein folding: toy models or predictive tools? *Current Opinion in Structural Biology*, 18:10–15, 2008.

[22] D. Baker. A surprising simplicity to protein folding. *Nature*, 405:39–42, 2000.

[23] Y. Duan and P.A. Kollman. Pathways to a protein folding intermediate observed in a 1-microsecond simulation in aqueous solution. *Science*, 282:740–744, 1998.

[24] `http://folding.stanford.edu/`.

[25] V.A. Voelz, G.R. Bowman, K. Beauchamp, and V.S. Pande. Molecular simulation of ab initio protein folding for a millisecond folder ntl9(139). *Journal of the American Chemical Society*, 132(5):1526–1528, 2010.

[26] D.E. Shaw et al. Atomic-level characterization of the structural dynamics of proteins. *Science*, 330(6002):341–346, 2010.

[27] K. A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24:1501–9, 1985.

[28] N. Lesh, M. Mitzenmacher, and S. Whitesides. A complete and effective move set for simplified protein folding. *RECOMB '03 Proceedings of the seventh annual international conference on Research in computational molecular biology*, pages 188–195, 2003.

[29] C. Micheletti, F. Seno, A. Maritan, and J.R. Banavar. Protein design in a lattice model of hydrophobic and polar amino acids. *Physical Review Letters*, 80(10):2237–2240, 1998.

[30] C. Levinthal. How to fold graciously. *Univ. of Illinois Bulletin*, 67(41):22–24, 1969.

[31] P. Wolynes. Energy landscapes and solved protein-folding problems. *Philosophical Transactions of the Royal Society A*, (363):453–467, 2005.

[32] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. *Journal of Computational Biology*, 5(3):423–465, 1998.

[33] H.J. Böckenhauer, A. Dayem Ullah, L. Kapsokalivas, and K. Steinhofel. A local move set for protein folding in triangular lattice models. *Lecture Notes in Computer Science*, 5251:369–381, 2008.

[34] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 6(21):1087–1092, 1953.

[35] F. Wang and D.P. Landau. Determining the density of states for classical statistical models: a random walk algorithm to produce a at histogram. *Physical Review E*, 64:056101, 2001.

[36] M.E.J. Newman and G.T. Barkema. *Monte Carlo Methods on Statistical Physics*. Oxford University Press, 1999.

[37] D.P. Sanders. Introducción a las transiciones de fase y a su simulación. Memorias de la XVII Escuela de Verano en Física de la UNAM (J. Recamier y M. Torres, eds.) UNAM, 2009.

[38] T. Wüst and D.P. Landau. The HP model of protein folding: A challenging testing ground for Wang–Landau sampling. *Computer Physics Communications*, (179):124–127, 2008.

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

[40] V. Cerny. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, 1985.

[41] J. Lee. New Monte Carlo algorithm: Entropic sampling. *Physical Review Letters*, 71(2):211–214, 1993.

[42] M. S. Shell, P. G. Debenedetti, and A. Z. Panagiotopoulos. An improved Monte-Carlo method for direct calculation of the density of states. *Journal of Chemical Physics*, 119(18):9406–9411, 2003.

[43] S.C. Kou, J. Oh, and W.H. Wong. A study of density of states and ground states in hydrophobic-hydrophilic protein folding models by equi-energy sampling. *Journal of Chemical Physics*, 124(24):244903, 2006.