SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Ciudad Madero

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

División de Estudios de Posgrado e Investigación

DESARROLLO DE UNA METAHEURISTICA PARALELA HIBRIDA POBLACIONAL PARA EL PROBLEMA DE DOBLADO DE PROTEINAS

TESIS

Que para obtener el grado de:
Maestro en Ciencias de la Computación

Presenta:
I.S.C. Anylu Melo Vega

Director:
Dr. Juan Javier González Barbosa

Codirector:
Dr. Ernesto Liñán García

Cd. Madero Tamaulipas **Noviembre 2017**

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Ciudad Madero

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

Cd. Madero, Tams., a 09 de Noviembre de 2017

OFICIO No.: U5.207/17
ÁREA: DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS.

C. ING. ANYLU MELO VEGA
No. DE CONTROL G06070435
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su Examen de Grado de Maestro en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"DESARROLLO DE UNA METAHEURISTICA PARALELA HIBRIDA POBLACIONAL PARA EL PROBLEMA DE DOBLADO DE PROTEÍNAS"

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE :	DR.	JUAN FRAUSTO SOLÍS
SECRETARIO:	DRA.	GUADALUPE CASTILLA VALDÉZ
VOCAL:	DRA.	LAURA CRUZ REYES
SUPLENTE:	DR.	JUAN JAVIER GONZÁLEZ BARBOSA
DIRECTOR DE TESIS :	DR.	JUAN JAVIER GONZÁLEZ BARBOSA
CO-DIRECTOR DE TESIS:	DR.	ERNESTO LIÑÁN GARCÍA

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con Usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

A T E N T A M E N T E
"POR MI PATRIA Y POR MI BIEN"®

DRA. ADRIANA ISABEL REYES DE LA TORRE
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

c.c.p.- Archivo
Minuta

AIRT ~~RAPR~~ 'mdcoa*

Ave. 1° de Mayo y Sor Juana I. de la Cruz Col. Los Mangos, C.P. 89440 Cd. Madero, Tam.
Tel. (833) 357 48 20. e-mail: itcm@itcm.edu.mx
www.itcm.edu.mx

Declaración de originalidad

Declaro que este documento de tesis es producto de mi trabajo original y que no infringe los derechos de terceros, tales como derechos de publicación, derechos de autor, patentes y similares.

Además, declaro que en las citas textuales que he incluido (las cuales aparecen entre comillas) y en los resúmenes que he realizado en publicaciones ajenas, indico explícitamente los datos de los autores y publicaciones.

Además, en caso de infracción de los derechos de terceros derivados de este documento de tesis, acepto la responsabilidad de la infracción y relevo de ésta a mi director y codirectores de tesis, así como al Tecnológico Nacional de México y a su Instituto Tecnológico de Ciudad Madero.

Ing. Anylu Melo Vega

Contenido

Capitulo 1. Introducción

Era el año de 1951 cuando el químico Linus Pauling realizó una predicción teórica, en la cual extendía los argumentos de minimización energética de sistemas cristalinos al caso de macromoléculas biológicas (Pauling, Corey, and Branson 1951). Para lo cual propuso la existencia de un estado termodinámicamente estable con estructura helicoidal para ciertas proteínas. Dos años más tarde en 1953 Watson y Crick llevan a cabo la confirmación experimental de lo propuesto por Pauling para una molécula de ADN (Watson and Crick 1953). Además, en 1958 John Kendrew, confirma lo anterior de una manera experimental para la mioglobina (Kendrew et al. 1958). La importancia de esta última investigación radica en el hecho de que la mioglobina es una proteína de un poco más de 150 aminoácidos, la cual pertenece al grupo de las hemoproteinas, es decir que están presentes en los músculos y cuya función principal es la de transportar y almacenar oxígeno; esta proteína tiene altas concentraciones en el músculo del corazón y los músculos esqueléticos, el gran mérito del profesor Kendrew y por el cual recibió el premio Nobel en 1955 fue relacionar la estructura de la proteína con la función de la misma (Ordway and Garry 2004) Un mérito mayor de este investigador es haber sido el primero en encontrar la estructura tridimensional de una proteína para su función natural; otra cuestión que fue impactante es que logró lo anterior sin utilizar los métodos de optimización existentes que en la actualidad se usan como punto de partida para determinar la estructura funcional también denominada Estructura Nativa (EN) (Levinthal 1968). El desarrollar métodos computacionales para encontrar la estructura nativa es un gran reto computacional y representa un problema denominado Problema de Doblado de Proteínas (PDP). PDP es un problema no resuelto hasta el momento, excepto para ciertas proteínas.

El proceso de doblado de proteínas se realiza de manera automática en la naturaleza y aún no se conoce con exactitud cómo es que se lleva a cabo. Pero lo que sí sabemos es que para que una proteína realice las funciones para las cuales fue diseñada, debe obtener la configuración de la EN, la cual se presenta en un estado de mínima energía de los componentes moleculares que cada proteína contiene.

Cuando una proteína en su estado funcional tiene una estructura tridimensional que es la nativa se dice que está naturalizada y como se indicó antes la energía del sistema de partículas que la componen (o energía de Gibbs) es la mínima; dicha estructura se puede perder por diversas causas (por ejemplo el incremento de la temperatura circundante) y cuando eso ocurre se dice que la proteína está desnaturalizada. Se puede entonces decir que el estado perfecto de las proteínas es el lograr la estructura de mínima energía de las proteínas naturalizadas. La cuestión es ¿Debería preocuparnos el que no alcance ese estado perfecto? La respuesta es: -Sí-, dado que si la configuración no es la adecuada, entonces la energía contenida en esa cadena de aminoácidos que conforman a la proteína será distinta y es justo en ese momento en el cual se presentan enfermedades provocadas por un mal

doblado de la proteína, por ejemplo, el caso de la encefalopatía espongiforme bovina (enfermedad de las vacas locas) o el Alzheimer.

Por otro lado, el problema computacional como el PDP es clasificado como NP-Complete (Crescenzi et al. 1998), lo cual en términos simples significa que no existe un algoritmo determinístico eficiente que resuelva PDP para todas las proteínas con una complejidad polinomial, es decir en tiempos razonablemente cortos para todas ellas. Sin embargo, al parecer dicho algoritmo sí existe, dado que en la naturaleza se resuelve de una manera muy rápida.

Esta tesis busca contribuir a la solución computacional del PDP para lo cual se proponen nuevos algoritmos híbridos que aplican métodos evolutivos, de recocido simulado y búsqueda dispersa, así como la paralelización de dichos algoritmos. Los algoritmos propuestos se han probado para un grupo de proteínas conocidas como péptidos con resultados satisfactorios.

1.1 Descripción del problema

El problema estudiado en este trabajo llamado Problema de Doblado de Proteínas (PDP), ha sido un reto científico desde los 1960's y aún en nuestros días sigue siendo un problema difícil de resolver (Dill and MacCallum 2012). PDP consiste en determinar la EN de una proteína a partir de una secuencia de aminoácidos dada. Para predecir ENs se pueden encontrar dos enfoques a seguir:

- Ab Initio
- Solución basada en plantillas o TB (del inglés, Template Based)

Ab Initio es un término en latín que significa "*desde el principio*", el cual se enfoca en encontrar ENs usando como información solo la secuencia de aminoácidos también conocida como estructura primaria de la proteína. Ab Initio hace la búsqueda guiándose principalmente por la secuencia de aminoácidos hasta encontrar la estructura terciaria o EN con el menor valor de energía potencial (Osguthorpe 2000).

La estrategia TB también inicia con la estructura primaria, pero en lugar de predecir la estructura terciaria, busca la estructura secundaria de la proteína. Al obtener la estructura secundaria cambia la estrategia de búsqueda a partir de fragmentos similares o estructuras parecidas a la molécula en un banco de proteínas conocido como Protein Data Bank (PDB). Este enfoque se basa en la habilidad de identificar plantillas que se adecuen a la alineación de secuencias de modelos estructurales ya conocidos (Khoury et al. 2014).

1.1.1 Función de Energía

En PDP para calcular el valor de energía que tiene una proteína se utiliza una función de energía. La función de energía utilizada en este trabajo es derivada de la investigación hecha por Eisenmenger (Eisenmenger et al. 2001) en la cual se utiliza el campo de fuerza ECEPP (Nemethy, Pottle, and Scheraga 1983) para calcular el potencial de energía interno de las proteínas. El campo ECEPP se calcula mediante la Ecuación 1 que es la sumatoria de la energía de Lennard Jones (E_{LJ}), la energía electroestática (E_{el}), la energía de enlaces de hidrogeno (E_{hb}) y la energía de torsión (E_{tor}).

$$E_{Tot(\Phi)} = E_{LJ} + E_{el} + E_{hb} + E_{tor} \quad (1)$$

Las primeras tres energías corresponden a la energía espacial y cada una es calculada como la sumatoria de las energías de la interacción entre pares de átomos. La cuarta energía, la de torsión, se obtiene de la sumatoria de los ángulos de torsión para cada enlace.

1.1.2 Formulación del problema de doblado de proteínas

El objetivo de PDP consiste en minimizar la energía del conjunto de moléculas de una proteína. La formulación del problema considera la función de energía que utiliza el campo de fuerza ECEPP que mide la energía interna total producida por la suma de las energías potenciales de los elementos de cada proteína; el campo de fuerza ECEPP tiene los elementos que se describen en la ecuación 1 (Eisenmenger et al. 2001),

La formulación de PDP se da a continuación:

1. Dada una secuencia de *n* aminoácidos $\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n$, la cual representa la estructura primaria de la proteína.

2. Dada una función de energía $f(\varphi_1, \varphi_2, \ldots, \varphi_m)$, la cual representa la energía libre.

La solución a PDP consiste en encontrar la estructura nativa en la cual la proteína tiene el valor de energía mínimo. Por lo tanto, el problema es minimizar el valor de energía dado por la siguiente ecuación (2):

$$min \quad E(\Phi) = \sum_{j>i}\left(\frac{A_{ij}}{r^{12}{}_{ij}} - \frac{B_{ij}}{r^{10}{}_{ij}}\right) + 332\sum_{j>i}\frac{q_i q_j}{\varepsilon r_{ij}} \quad + \sum_{j>i}\left(\frac{C_{ij}}{r^{12}{}_{ij}} - \frac{D_{ij}}{r^{6}{}_{ij}}\right) + \sum_n U_n(1 \pm cos(k_n \Phi_n)) \qquad (1)$$

Donde:

- $\Phi = (\varphi_1, \varphi_2, \ldots, \varphi_m)$, $\epsilon\ Ra^3$ (Espacio de Ramachandran), este vector contiene los ángulos que definen la estructura tridimensional de la proteína (Ramachandran, Ramakrishnan, and Sasisekharan 1963).
- r_{ij} es la distancia entre los átomos *i* y *j* en Å.
- $\frac{A_{ij}}{r^{12}{}_{ij}}\ y\ \frac{B_{ij}}{r^{10}{}_{ij}}$ son los potenciales de Lennard Jones para el par de átomos *i* y *j*.
- C_{ij}, y D_{ij} son los parámetros de los potenciales empíricos para la Energía de enlaces de hidrogeno.
- $q_i\ y\ q_j$ son las cargas parciales que actúan sobre los átomos *i* y *j*.
- ε es la constante dieléctrica que usualmente se fija en ε = 2 (Eisenmenger et al. 2001).
- 332 es un factor de conversión para poder expresar la energía en kcal/mol (Eisenmenger et al. 2001).
- U_n es la torsión energética alrededor el enlace *n*.
- k_n es la multiplicidad del ángulo de torsión φ_n.

En resumen, PDP consiste en encontrar la EN encontrando los valores de:

- $f^*(\varphi 1, \varphi_2, \ldots, \varphi_n)$ siendo el mínimo valor de energía y,
- $\Phi^* = (\varphi_1, \varphi_2, \ldots, \varphi_n)$ es la configuración de ángulos óptima que posee el mínimo valor de energía $f^*(\varphi 1, \varphi_2, \ldots, \varphi_n)$.

1.1.3 Instancia del Problema

Las instancias de PDP son comúnmente definidas en un archivo de texto el cual contiene la secuencia de aminoácidos para construir la molécula. A continuación, se muestra un ejemplo del archivo de texto para la molécula Met Enkephalin:

```
#Met-Enkephalin
Tyr Gly Gly Phe Met
```

El archivo de texto comienza con un símbolo "#" y puede (o no) contener el nombre de la molécula. Las siguientes líneas contendrán el listado de la secuencia de amino ácidos o residuos a experimentar. Los nombres de los residuos deberán estar separados por un espacio en blanco y estarán definidas en un código de tres letras ver **¡Error! No se encuentra el origen de la referencia.**.1.

Tabla 1.1 *Tipos de residuos definidos*

Neutral	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly
	His	Hise	Hyp	Ile	Leu	Lys	Met	Phe
	Pro	Ser	Thr	Trp	Tyr	Val		
Con carga	Arg^{+}	Asp^{-}	Glu^{-}	His^{+}				

1.1.4 Solución candidata

Una solución candidata está formada por un archivo de configuración de ángulos con números reales para los átomos de la molécula, cuyos valores pueden haber sido previamente determinados en otros estudios o, en caso de que no existan, son generados al azar. La Tabla 1.2, presenta un ejemplo de una solución candidata para la proteína conocida como Met Enkephalin, utilizando el siguiente esquema: Número de Residuo, Variable y Valor. Cada renglón en la tabla consiste en tres campos separados por ":".

Tabla 1.2 *Archivo de configuración de Met Enkephalin*

Residuo	**:Variable**	**:Valor**
1	: phi	:-86.24
1	: x1	: -172.59
1	: x2	: 78.71
1	: x6	: .165.88
2	: psi	: 156.18
2	: omg	: 180.00
2	: phi	: -154.53
3	: psi	: 83.64
3	: omg	: 180.00
3	: phi	: 83.66
4	: psi	: -73.86
4	: omg	: 180.00
4	: phi	: -137.04
4	: x1	: 58.79
4	: x2	: 94.60
5	: psi	: 19.33
5	: omg	: 180.00
5	: phi	: -163.63
5	: x1	: 52.76
5	: x2	: 175.28
5	: x3	: -179.83
5	: x4	: -58.57
5	: psi	: 160.45
5	: omg	: 180.00

- Primer campo de tipo INTEGER, se encuentra el número de residuo.

- Segundo campo de tipo STRING, se encuentra el nombre de la variable para cada ángulo diedro en específico.
- Tercer campo de tipo REAL, proporciona el valor de la variable; con este último valor se realiza el cálculo de la energía de Gibbs y se dice que se experimenta con él durante el proceso del algoritmo.

1.2 Complejidad del Problema

Existe un tema con respecto a todas las proteínas, desde las pequeñas, conocidas como péptidos, hasta las más grandes. Es conocido como la Paradoja de Levinthal y se puede formular con la siguiente pregunta: ¿Cómo una proteína puede tan rápidamente alcanzar su estructura tridimensional entre todas las posibles conformaciones y que tenga el mínimo valor de energía? (Levinthal 1968; Crescenzi et al. 1998). Para ejemplificar, asumamos una proteína con una secuencia de 200 aminoácidos y que solo pueda tomar tres configuraciones esto sería 3^{200}. Explorar todas esas posibilidades tomaría en una computadora de alto nivel un tiempo que podría ser más grande que la edad del universo. Sin embargo, la naturaleza realiza el proceso de doblado en solo algunos segundos.

En la literatura sobre la complejidad de PDP encontramos una reiteración más convincente de esa paradoja, la que nos muestra la reducción de PDP al problema del ciclo Hamiltoniano el cual es conocido como un problema NP-Completo (Crescenzi et al. 1998).

1.3 Objetivo General y Específicos

En esta tesis se busca contribuir a la solución computacional de PDP aportando nuevos algoritmos. Para el desarrollo de dichos algoritmos se comenzó analizando los publicados hasta el momento, que han sido eficientes y que presentan un área de oportunidad para mejorar los procesos de búsqueda. Con base en los resultados de la competencia CASP se analizaron las mejores estrategias algorítmicas publicadas con mejor puntaje. Se determinó que Recocido Simulado con estrategias evolutivas, y de búsqueda dispersa, acompañados de métodos de recalentamiento (conocidos como Quenching y MultiQuenching) representaban potencialmente un área de oportunidad para el caso de péptidos, debido a que estos permiten incrementar la variabilidad y la eficiencia de la búsqueda de la EN. Además, debido a los tiempos tan largos que se requieren durante las simulaciones de los algoritmos de PDP, se determinó que la paralelización de estos algoritmos es necesaria. Por último, se adoptó la estrategia de búsqueda Ab Initio debido a dos razones:

- Representa el mayor de los retos ya que busca resolver PDP. Tal como se planteó por Anfinsen(Anfinsen 1973), para determinar la EN de la proteína, solo se

requiere conocer la secuencia de aminoácidos que la conforman y buscar aquella estructura terciaria con mínima energía.
- Los métodos basados en plantillas no garantizan que la solución encontrada sea realmente la EN.

En base a lo anterior se plantean los objetivos de esta tesis.

1.3.1 Objetivo General

Desarrollar una metaheurística competitiva paralela hibrida poblacional basada en MultiQuenching, Algoritmos genéticos y Scatter Search para la configuración óptima en el problema de doblado de proteínas, aplicando el enfoque AB Initio.

1.3.2 Objetivos Específicos

- Paralelizar el algoritmo MultiQuenching tradicional.
- Desarrollar un algoritmo hibrido poblacional paralelo basado en el enfoque de enfriamiento rápido (MultiQuenching) y Algoritmos Genéticos, para resolver el problema de doblado de proteínas.
- Desarrollar una heurística híbrida poblacional paralela del método MultiQuenching con Scatter Search.
- Comparar los resultados obtenidos con métodos del estado del arte.

1.4 Justificación

El problema de doblado de proteínas representa un enorme desafío para la ciencia y es de interés interdisciplinario ya que se ven involucradas áreas como la biología, la bioinformática y la optimización combinatoria. Debido a que una proteína puede tomar un sin número de configuraciones hasta lograr alcanzar su EN, esto produce un problema difícil para obtener soluciones en un tiempo adecuado.

Por lo anterior, en la comunidad científica se ha buscado implementar estrategias híbridas, siendo Recocido Simulado la más exitosa hasta el momento. Por otro lado, los métodos de recalentamiento tipo Quenching y MultiQuenching, así como los evolutivos, comúnmente se utilizan para mejorar ese tipo de algoritmos. A su vez, los algoritmos para PDP comúnmente se desarrollan en su versión paralela, sin embargo, no se reportan algoritmos híbridos eficientes con esas estrategias en dicha versión en la literatura.

1.5 Alcances

Los algoritmos desarrollados en esta tesis se definieron para ser probados con un tipo de proteínas conocidas como péptidos. Este tipo de proteínas tienen la misma estructura general que el resto de las proteínas solamente que el número de aminoácidos es pequeño, de alrededor de 30 aminoácidos.

1.6 Limitaciones

- El modelo Matemático usado para el cálculo de los valores de energías de las instancias está basado en el campo de fuerza conocido como ECEPP/2. El cambio a otro campo de fuerza, sin embargo, no resta generalidad a los resultados encontrados. Los resultados reportados como energía no miden la energía realmente, sino que son valores virtuales.

- La implementación de los algoritmos presentados en esta tesis han sido realizados en el ambiente de programación para dinámica molecular conocido SMMP (Simple Molecular Mechanics for Protein). Los algoritmos desarrollados pertenecen al grupo de métodos de optimización aproximados, eso significa que la mejor solución obtenida por los algoritmos aquí desarrollados no necesariamente son el óptimo.

- Las métricas utilizadas para medir la calidad de las soluciones de los casos de prueba reportados en esta tesis son RMSD y TM-Score.

- La comparación de resultados se hizo solo contra los de las investigaciones realizadas con péptidos y que emplearon Ab Initio.

Capitulo 2. Marco Teórico

2.1 Proteínas

Para entender el proceso de PDP es necesario proporcionar una definición de las proteínas. Una proteína es una cadena polimérica de aminoácidos y, de acuerdo con la Real Academia Española es la "*Sustancia constitutiva de la materia viva, formada por una o varias cadenas de aminoácidos*". Las proteínas desempeñan un gran número de funciones y son las encargadas de la construcción de las estructuras de todos los seres vivos.

Las proteínas están formadas por aminoácidos unidos por enlaces peptídicos. Existen veinte estructuras diferentes de aminoácidos que conforman proteínas en los animales vivos. Aun con este número pequeño de aminoácidos, la naturaleza puede crear proteínas muy complejas. La secuencia en la cual los aminoácidos están colocados, dictamina la forma estructural que tomará la proteína. Dado que la función de las proteínas se basa en la capacidad de reconocer y unirse a moléculas específicas, que tengan la forma estructural correcta es crítico para que puedan realizar su trabajo adecuadamente.

Las proteínas cumplen una amplia gama de funciones, cada persona tiene varios cientos de miles de proteínas diferentes en su cuerpo. Las proteínas se pliegan en varias estructuras hasta lograr una variedad de formas tridimensionales, a continuación se describen las cuatro etapas estructurales por las cuales pasa cada proteína (Olivares-Quiroz, García, and Scherer 2004).

2.1.1 Estructuras de las proteínas

Existen diferentes aminoácidos pero todos comparten una misma estructura química. Cada aminoácido está compuesto por un átomo central de carbono, llamado *Cα*, mediante enlaces peptídicos liga a un grupo amino (NH_2), un grupo carboxilo (*COOH*), un átomo de hidrogeno (*H*) y un grupo residuo *R* el cual marca la diferencia para cada aminoácido.

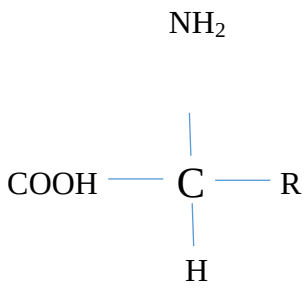


Figura 2.1 *Estructura de un aminoácido*

En la figura 2.1 se muestra la estructura general de un aminoácido. Cada proteína pasa por varios estados llamados estructuras primaria, secundaria y terciaria. Algunas proteínas pueden llegar a pasar a tener una estructura cuaternaria. A continuación, se muestran las definiciones de dichas estructuras:

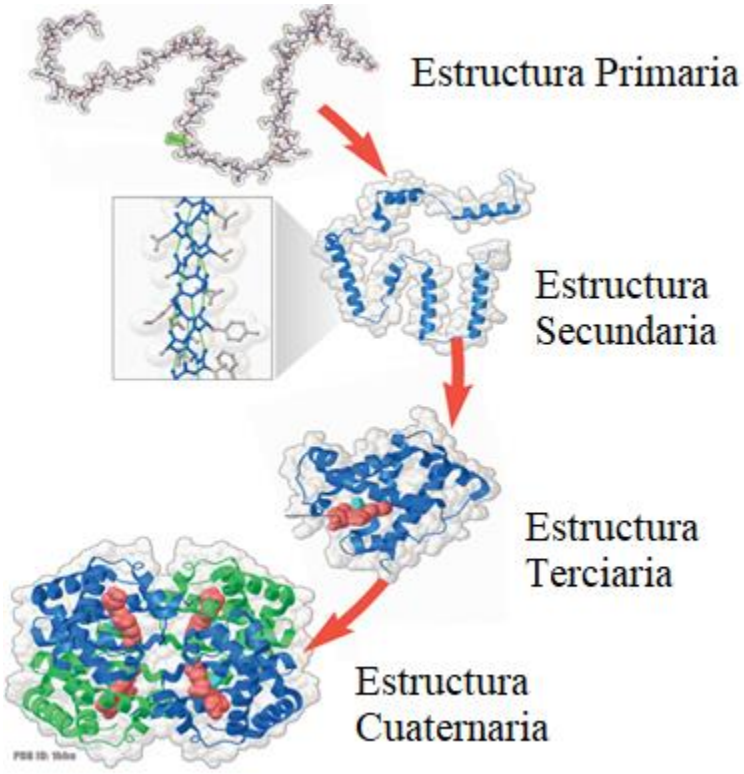


Figura 2.2 *Estructuras de la proteína 1HHO*

Estructura primaria: es la secuencia lineal de aminoácidos, la cual define como se doblará la proteína y que funciones tomará. Un cambio en la secuencia de aminoácidos cambiaría la estructura primaria, por ejemplo: si la proteína Hemoglobina sufriera un cambio en la secuencia que le da forma, podría provocar que las proteínas se reagruparan, resultando en una anemia.

Estructura secundaria: con una secuencia ya definida, la estructura primaria transita por una serie de estados en los cuales su configuración espacial se modifica, hasta alcanzar un estado termodinámicamente estable con una estructura tridimensional particular, la cual depende de la secuencia de aminoácidos inicial y es distinta para cada proteína. Esta estructura, define los sitios activos de la macromolécula y es la única relevante desde el punto de vista biológico. En la estructura secundaria se adopta un doblado en la cadena poli-peptídica cuando se presenta un enlace de hidrógeno entre los átomos que forman el

enlace peptídico, dando lugar a una conformación más estable, conformados por: las hélices α y láminas β (Anfinsen 1973).

Estructura terciaria: en esta estructura la proteína obtiene su forma tridimensional, aquí esta doblada por completo y puede realizar las funciones para las cuales fue diseñada desde la secuencia de aminoácidos.

Estructura cuaternaria: Todas las proteínas tienen estructuras primarias, secundarias y terciarias, las estructuras cuaternarias sólo surgen cuando una proteína se compone de dos o más cadenas de polipéptidos. El doblamiento de las proteínas también es impulsado y reforzado por la formación de muchos enlaces entre diferentes partes de la cadena. La formación de estos enlaces depende de la secuencia de aminoácidos (Olivares-Quiroz, García, and Scherer 2004).

2.1.2 Paradoja de Levinthal

El doblado de proteínas como fenómeno natural se ejecuta sumamente rápido, motivo por el cual resulta difícil analizar a detalle y poder encontrar su estructura exacta en ese mismo tiempo, ya que existe un número infinito de posibles soluciones para esa cadena de aminoácidos que guarda tan preciado tesoro. Es por eso que se realizan predicciones del fenómeno mediante simulaciones computacionales, pero esto incluye un nuevo reto ya que resulta imposible llevar a la práctica un algoritmo que explore todo ese conjunto de soluciones. Levinthal determinó que si una proteína se doblara explorando al azar todas las conformaciones posibles tardaría mucho más que la edad del universo encontrar su configuración nativa (Levinthal 1968). El hecho de que la naturaleza logra llegar a la estructura nativa en tiempos sumamente cortos, del orden de nanosegundos, y que por otra parte los algoritmos computacionales tarden tiempos exagerados como semanas o meses lleva a una paradoja conocida como la paradoja de Levinthal que se explica a continuación.

¿Cuánto tiempo le toma a una proteína llegar a su estructura nativa? Para ilustrarnos la paradoja de Levinthal, consideremos que cada enlace que conecta a los aminoácidos pueda tener varios estados (por ejemplo tres estados) y que la proteína tenga 101 aminoácidos; podrían existir $3^{100} = 5 \times 10^{47}$posibles configuraciones. Incluso si la proteína pudiese explorar todas las configuraciones a una taza de 10^{13} configuraciones por segundo o 3×10^{20} por año, tomaría 10^{27} años en probar todas. Levinthal concluye que búsquedas al azar no son efectivas para encontrar el estado nativo de las proteínas. Mas sin embargo, las proteínas se pliegan en cuestión de segundos. Esto es lo que se conoce como paradoja de Levinthal (Levinthal 1968).

2.1.3 Ángulos Diedros

Existen varios enfoques para definir las distintas configuraciones que una secuencia puede adoptar, el más común consiste en variar los ángulos entre dos planos formados por los átomos (ángulos diedros) mientras se mantienen constantes las distancias de los puentes formados por átomos adyacentes.

Los ángulos de la cadena principal son:

- φ para el ángulo que se forma entre el Cα y el grupo amino del mismo aminoácido.
- ψ corresponde al ángulo formado entre el Cα y el grupo carboxilo en un mismo aminoácido.
- ω es el ángulo de la unión de dos aminoácidos consecutivos, se forma entre el grupo amino del primero y el grupo carboxilo del segundo. Por convención normalmente este ángulo es constante al valor de 180° en las simulaciones computacionales.

Los ángulos de las cadenas laterales:

- χ son los ángulos que corresponden a las cadenas laterales.

2.1.4 Gráfica de Ramachandran

Ramachandran observó que los ángulos diedros φ y ψ se encuentran en valores de intervalos determinados las regiones donde se encuentran los ángulos de las proteínas dobladas en forma de hélice α y hoja β. Ramachandran realizo una serie de graficas donde se establecieron dichas regiones. Es posible comparar los valores de los ángulos en las cadenas de polipéptidos contra una gráfica de Ramachandran para saber si la estructura evaluada es válida o no (Ramachandran, Ramakrishnan, and Sasisekharan 1963).

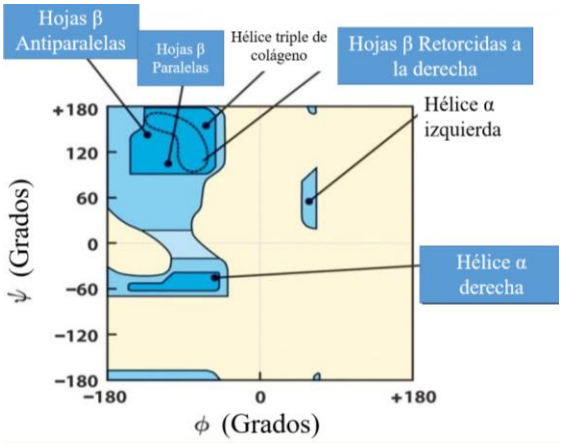


Figura 2.3 *Gráfica de Ramachandran.*

2.1.5 Hélices α y hojas β

Son estructuras que se forman al interior de la proteína, ambas son conocidas como estructuras secundarias y se caracterizan por la formación de puentes de hidrogeno entre el grupo amino NH_2 y el grupo carboxilo COOH.

Una hélice α en promedio contiene 3.6 aminoácidos por vuelta con un puente de hidrogeno en el grupo amino y carboxilo. En esta hélice los átomos de la proteína están empaquetados de forma adecuada para generar una configuración robusta y la interacción entre aminoácidos es de corto alcance.

Por su parte las hojas β requieren de interacciones de largo alcance entre los aminoácidos. La configuración de una hoja β agrupa entre 5 y 10 aminoácidos extendidos en un plano. La mayoría de las proteínas está constituida por combinaciones de estas dos estructuras secundarias, conectadas entre sí mediante cadenas de aminoácidos con forma irregular. Éstas se conocen como hebras de enlace y están constituidas, en promedio por no más de 8 aminoácidos (Olivares-Quiroz, García, and Scherer 2004).

En consecuencia de lo anterior debiera ser posible utilizar esta información para encontrar la estructura nativa de una proteína; ese aspecto es un campo abierto de investigación y está fuera del objetivo de la presente tesis.

2.2 Doblado de Proteínas

PDP es un problema de predicción del EN de la proteína a partir de la secuencia inicial de aminoácidos, este problema ha permanecido abierto en las últimas décadas. Debido a su dificultad combinatoria este problema constituye uno de los retos más importantes que se afronta en la actualidad (Dill and MacCallum 2012).

2.2.1 El Dogma Central

Según (Dill et al. 2008) el problema de doblado de proteínas trata de cómo la secuencia de aminoácidos de una proteína determina su estructura atómica tridimensional. La noción de un "problema" de doblado surgió por primera vez alrededor de 1960, con la aparición de las primeras estructuras de proteínas. Algún tipo de regularidad cristalina interna se esperaba previamente y hélices α habían sido anticipadas por Linus Pauling, pero las primeras estructuras de proteínas (las globinas) tuvieron hélices que fueron encontradas juntas en formas irregulares inesperadas. Desde entonces, el problema del plegamiento de proteínas ha llegado a ser considerado como tres problemas planteados en (Dill et al. 2008; Dill and MacCallum 2012)

1. El código de doblado: ¿cuál es el código físico por el cual una secuencia de aminoácidos dicta la estructura nativa de una proteína?
2. ¿Cómo es que las proteínas pueden doblarse tan rápido?
3. ¿Podremos idear un algoritmo para predecir las estructuras de las proteínas a partir de sus secuencias?(Dill and MacCallum 2012).

Esta tesis se enfoca principalmente al último de los puntos anteriores.

2.2.2 Campo de Fuerza

A partir de 1980 junto con la dinámica molecular y las simulaciones de Monte Carlo surgieron campos de fuerza para PDP a partir de campos de fuerza desarrollados en la química orgánica (Ponder and Case 2003). Siendo de gran importancia los campos de fuerza ECEPP (Momany et al. 1975; Nemethy, Pottle, and Scheraga 1983), CHARMM (Brooks, 1983). AMBER (Pearlman et al. 1995) y GROMACS (Berendsen, 1995).

Un campo de fuerza se refiere al conjunto de parámetros geométricos, cargas atómicas parciales, energía de las interacciones no enlazadas, energía de enlaces de hidrogeno y potenciales de torsión de un polipéptido o proteína (Momany et al. 1975).

El campo de fuerza que es más utilizado incorpora una función de energía potencial relativamente simple:

$$V(r) = \sum_{enlaces} k_b(b - b_0)^2 + \sum_{angulos} k_\theta(\theta - \theta_0)^2 + \sum_{torsiones} k_\varphi[cos(n\varphi + \delta) + 1] + \sum_{pares\ no\ enlazados} \left[\frac{q_i q_j}{r_{ij}} + \frac{A_{ij}}{r_{ij}^{12}} - \frac{C_{ij}}{r_{ij}^{6}}\right] \quad (2)$$

Las primeras tres sumatorias tratan sobre los enlaces (interacciones 1-2), los ángulos (interacciones 1-3) y torsiones (interacciones 1-4). El término de torsión puede incluir las llamadas torsiones “impropias”, donde los cuatro átomos que definen al ángulo no están conectados en enlaces covalentes. La suma final (sobre los pares de átomos i y j) excluye las interacciones 1-2 y 1-3 y a menudo utiliza parámetros separados para las interacciones 1-4 en comparación con los utilizados para átomos separados por más de tres enlaces covalentes. Dicha suma describe la electroestática que utiliza cargas parciales q_i en cada átomo que interactúa por la ley de Coulomb. La combinación de dispersión y fuerzas de repulsión de intercambio están representadas por el potencial 6-12 de Lennard-Jones al que a menudo se le denomina como el término " Van der Waals ". La *(2* se refiere a la función de energía potencial más simple que puede reproducir las características básicas de los escenarios de energía de la proteína a un nivel de detalle atómico. La combinación de una función de energía potencial y todos los parámetros que se encuentran en ella (k_b, b_0,

k_Θ, Θ_0, etc.) constituye al campo de fuerza. En la práctica, existe una conexión cercana entre los campos de fuerza y los algoritmos computacionales que los implementan (Ponder and Case 2003).

Campo de Fuerza ECEPP/2

El cálculo de la energía de un polipéptido o proteína requiere de un conjunto de parámetros que describan la geometría molecular y los potenciales interatómicos. Dicho conjunto de parámetros fue desarrollado y nombrado ECEPP (Empirical Conformational Energy Program for Peptides)(Momany et al. 1975) el cual es un programa informático que contiene:

- Datos sobre la geometría (longitudes de enlaces y ángulos de enlace) para los veinte aminoácidos de origen natural, así como para otros aminoácidos y grupos protectores terminales.
- Cargas atómicas parciales para cada átomo de los residuos y grupos terminales.
- Parámetros para las interacciones no enlazadas, electroestática, e interacciones de enlaces de hidrogeno y energías de torsión.

Estos parámetros fueron derivados de datos de cristales y de cálculos de orbitales moleculares de CNDO/2 (ON), como un conjunto interno consistente. ECEPP ha sido utilizado en cálculos computacionales sobre una variedad de polipéptidos.

En 1983 Nemethy actualiza algunos de los parámetros utilizados en ECEPP creando una nueva versión del programa ECEPP/2 (Nemethy, Pottle, and Scheraga 1983). ECEPP/2 se encuentra como un campo de fuerza que es frecuentemente utilizando por los trabajos realizados en polipéptidos y proteínas (Román Rangel 2006).

2.2.3 Calculo de la Función de Energía

El campo de fuerza ECEPP/2 tiene dos componentes de energía que define el potencial de energías de una proteína ($E_{potencial}$), una es la energía de interacciones enlazadas (E_{bonded}) y otra es el de las energías con interacciones no enlazadas ($E_{non\text{-}bonded}$); por lo tanto el potencial de energía es definido por (Nemethy, Pottle, and Scheraga 1983):

$$E_{potential} = E_{bonded} + E_{non\text{-}bonded} \quad (3)$$

Donde E_{bonded} está asociado con las fuerzas debidas a los enlaces covalentes, y $E_{non\text{-}bonded}$ está asociado con las fuerzas debidas a los enlaces no-covalentes

El término E_{bonded} está definido por:

$$E_{bonded} = E_{hydrogen\text{-}bond} + E_{Torsion} \quad (4)$$

El término $E_{non\text{-}bonded}$ está definido por:

$$E_{non\text{-}bonded} = E_{vdw} + E_{electrostatic} \quad (5)$$

La energía de las interacciones non-bonded es la suma de las energías de todos los átomos no enlazados que actúan entre ellos. Esto incluye la energía de interacción de Van der Waals y la energía de electroestática.

La energía Potencial $E_{potential}$ está definida por (Nemethy, Pottle, and Scheraga 1983):

$$E_{potential} = E_{hydrogen\text{-}bond} + E_{Torsion} + E_{vdw} + E_{electrostatic} \quad (6)$$

Un punto importante es realizar análisis sobres los diferentes tipos de proteínas y la contribución de los elementos anteriores y tipos de proteínas, de forma que se puedan mejorar los procesos de búsqueda. Este aspecto ha sido planteado originalmente en los años 1980´s y es una actividad pendiente en el campo de los métodos de optimización, el cual queda fuera del alcance de esta tesis.

2.2.3.1 Enlace de hidrogeno

El enlace de hidrógeno es la fuerza atrayente entre el hidrógeno unido a un átomo electronegativo de una molécula y a un átomo electronegativo de una molécula diferente. La energía asociada con los enlaces de hidrógeno $_{-bond}$ está basado en la siguiente formula (Eisenmenger and Hansmann 1997):

$$E_{hydrogen-bond} = \sum_{i<j} \left(\frac{A_{ij}}{r^{12}{}_{ij}} - \frac{B_{ij}}{r^{10}{}_{ij}} \right) \quad (7)$$

Donde A_{ij} y B_{ij} son los parámetros de atracción/repulsión entre dos cargas *i* y *j* respectivamente mientras que r_{ij} representan la distancia entre el átomo i y el átomo j.

2.2.3.2 Energía de Torsión

La energía de Torsión ($E_{Torsion}$) de una proteína es la suma de las energías de cada enlace. Se supone independencia de la rotación alrededor del enlace individual. La energía se calcula como (Eisenmenger and Hansmann 1997) :

$$E_{Torsion} = \sum_n U_n (1 \pm cos(k_n \varphi_n)) \quad (8)$$

Donde U_n es la barrera de energía de torsión alrededor del enlace *n* y k_n es la multiplicidad o la periodicidad del ángulo de torsión φ_n.

2.2.3.3 Interacción electroestática

La energía electroestática $E_{electrostatic}$ está basada en la Ley de Coulomb, la cual establece que la fuerza eléctrica entre dos partículas cargadas es directamente proporcional al producto de las cargas en las partículas, e inversamente proporcional al cuadrado de la distancia entre las dos partículas. La ecuación que determina la fuerza eléctrica es la clásica ley de Coulomb (Eisenmenger and Hansmann 1997):

$$E_{electrostatic} = \frac{kQ_iQ_j}{r^2{}_{ij}} \tag{9}$$

Donde *Qi* y *Qj* representan las cantidades de cambios (Coulomb) sobre la partícula uno y dos, respectivamente. r_{ij} representa la distancia entre *Qi* y *Qj*. *k* es la constante de proporcionalidad que depende de la constante dieléctrica ε que a su vez depende del medio en que se trate: (Eisenmenger and Hansmann 1997).

$$k = \frac{1}{4\pi\varepsilon} \tag{10}$$

2.2.3.4 Interacciones de van der Waals

El término de interacción de Van Der Waals, E_{vdw} es descrito normalmente por el potencial de Lennard-Jones. A continuación se define la interacción (Eisenmenger and Hansmann 1997):

$$E_{vdw} = \sum_{i<j}\left(\frac{C_{ij}}{r^{12}{}_{ij}} - \frac{D_{ij}}{r^{6}{}_{ij}}\right) \tag{11}$$

Donde r_{ij} es la distancia entre dos átomos no enlazados i y j; C_{ij} y D_{ij} son parámetros de atracción/repulsión entre dos cargas *i* y j respectivamente. Estos parámetros son calculados por el número de electrones por átomo, y la polarizabilidad atómica por átomo.

2.3 Heurísticas y Metaheurísticas

Dada la dificultad práctica para resolver de forma exacta un importante conjunto de problemas combinatorios, se comenzaron a desarrollar algoritmos que proporcionan soluciones de buena calidad, sin garantizar la obtención de la solución óptima, en un tiempo de cálculo razonable. Este tipo de algoritmos se denominan *Heurísticas*, del griego *Heuriskein* "para descubrir". Aunque en un primer momento no fueron bien vistas en los círculos académicos acusadas de escaso rigor matemático, su interés practico como herramienta útil que da soluciones a problemas reales, les fue abriendo poco a poco las

puertas, sobre todo a partir de los años 70's con la proliferación de resultados en el campo de la complejidad computacional (Díaz Velarde and González Fernández 1996).

Un método heurístico es un conjunto muy bien definido de pasos que rápidamente identifican una solución de alta calidad para un problema dado. Donde la solución es un conjunto de valores para las incógnitas del problema y la calidad está definida por una métrica de evaluación o criterio. Las soluciones se asumen que son factibles, conociendo todas las restricciones del problema. Su propósito es identificar soluciones del problema donde el tiempo es más importante que la calidad de la solución o el conocimiento de calidad. En (Silver et al. 1980) se propone la siguiente clasificación de métodos de resolución mediante heurísticas:

- *Métodos constructivos*, que se caracterizan por construir una solución definiendo diferentes partes de ella en pasos sucesivos.
- *Métodos de descomposición*, dividen el problema en varios más pequeños y la solución se obtiene a partir de la solución de cada uno de estos.
- *Métodos de reducción*, tratan de identificar alguna característica de la solución que permita simplificar el tratamiento del problema.
- *Métodos de manipulación del modelo*, obtienen una solución del problema original a partir de otra de otro problema simplificado (con menos restricciones, linealizando el problema, etc.).
- *Métodos de búsqueda por entornos*, en las que se parte de una solución inicial a la que se realizan modificaciones en sucesivas iteraciones para obtener una solución final. En cada iteración existe un conjunto de soluciones vecinas candidatas a ser nueva solución en el proceso. Este grupo contienen a las técnicas metaheurísticas.

Las técnicas metaheurísticas son procedimientos de búsqueda que tampoco garantizan la obtención del óptimo del problema considerado y que también se basan en la aplicación de reglas relativamente sencillas. Las técnicas metaheurísticas, a diferencia de las heurísticas, poseen mecanismos para evitar el estancamiento en óptimos locales con el objetivo de encontrar o aproximarse al óptimo global. La aplicación de las técnicas metaheurísticas es especialmente interesante en el caso de problemas de optimización combinatoria: problemas en los que las variables de decisión son enteras (o discretas, al menos) en las que, generalmente, el espacio de soluciones está formado por ordenaciones de valores de dichas variables. Sin embargo, las técnicas metaheurísticas se pueden aplicar también a problemas de otro tipo, como con variables continuas.

La lógica de las técnicas metaheurísticas es similar: el punto de partida es una solución o conjunto de soluciones, que típicamente no es óptima. A partir de ella se obtienen otras parecidas, de entre las cuales se elige una que satisface algún criterio, a partir de la cual comienza de nuevo el proceso. Este proceso se detiene cuando se cumple alguna condición establecida previamente (Silver et al. 1980).

A continuación se explican las características más importantes técnicas utilizadas en este trabajo: Simulated Annealing, Quenching, MultiQuenching, Algoritmos Genéticos y Búsqueda Dispersa.

2.3.1 Recocido Simulado

El algoritmo Recocido Simulado o en inglés Simulated Annealing (SA) fue propuesto por Kirkpatrik en 1983 (Kirkpatrick, Gelatt, and Vecchi 1983). SA imita el proceso de templado de metales hasta alcanzar el punto de fusión. Se sintoniza a una temperatura alta T_0, conforme avanza el proceso, esta temperatura se reduce gradualmente hasta alcanzar una temperatura final T_f muy próxima a cero. Para disminuir la temperatura SA utiliza diferentes funciones de enfriamiento siendo la más utilizada la función geométrica lineal $T_{k+1}= \alpha T_k$. Para PDP, el valor de α recomendado está en el rango $0.70 \leq \alpha < 1.0$. Los valores de T_0, T_f y α se establecen de forma experimental.

2.3.1.1 Ciclo de Metrópolis

SA tiene un método que produce soluciones vecinas, denominado Ciclo de Metrópolis o Metrópolis Cycle (MC) en inglés. MC genera soluciones basadas en la solución actual, utilizando la distribución de probabilidad de Boltzmann como parámetro de aceptación

Algoritmo 1. *Algoritmo Metrópolis*

```
1.  Begin
2.  Repeat
3.        Sj= Perturbación (Si)
4.      Diferencia = f(Sj) - f(Si)
5.      If Diferencia <= 0 Then
6.            Si = Sj
7.                 else
8.                   If (e^(Diferencia/T) > random[ 0,1 ] ) Then
9.                    Si = Sj
10.                        end if
11.                end if
12.       Until se alcance el equilibrio térmico
13. End
```

2.3.1.2 Algoritmo Recocido Simulado

A continuación se muestra el algoritmo de Simulated Annealing, al inicio del algoritmo crea la solución inicial S_i (línea 3). El valor de $T_{initial}$ que representa la temperatura inicial del proceso se inicializa (línea 4). El número de iteraciones del proceso

es representado por k y se inicializa en cero. Utilizando un ciclo externo se reduce el parámetro de temperatura. Dentro del ciclo se utiliza el algoritmo de Metrópolis y se incrementa el valor de k por uno, y la temperatura se reduce mediante una función de enfriamiento (línea 9).

Algoritmo 2. *Algoritmo Recocido Simulado*

```
1.  Begin
2.      Si= solución inicial
3.      T_initial = Valor de Temperatura Inicial
4.      k=0
5.      Repeat // Ciclo Externo
6.      Procedimiento Metrópolis ( )
7.          k = k + 1
8.          T = T - cool (T)
9.          Until Criterio de Stop
10. End
```

2.3.2 Quenching Annealing

El algoritmo de enfriamiento rápido o en inglés Quenching Annealing Algorithm (QA) (Frausto-Solis and Román 2007), es una metodología propuesta en 2007 para PDP. QA fue desarrollando utilizando dos fases de enfriamiento:

- Fase de Enfriamiento (Quenching Phase)
- Fase de Recocido (Annealing Phase)

La primera fase es una analogía del proceso de enfriamiento físico, el cual es similar al proceso de recocido simulado pero la temperatura decrece rápidamente hasta alcanzar el equilibrio cuasi-térmico. Para PDP, la energía cambia de forma caótica debido a las variaciones extremas que presenta. En las temperaturas muy altas se aplica la fase de enfriamiento (Quenching) y la temperatura decrece de forma exponencial. Una vez alcanzado el equilibrio cuasi-térmico por la función, el algoritmo ahora inicia la fase de recocido (Annealing) la cual decrecerá los valores de temperatura adaptando métodos de ajuste analítico.

2.3.3 MultiQuenching Annealing

MultiQuenching Annealing Algorithm (MQA) (Frausto-Solis, Soberon-Mainero, and Liñan-Garcia 2009) es una implementación eficiente de QA que también cuenta con dos Fases: Quenching (QP) y Annealing (AP) y subdivide estas fases en subfases.

La fase QP busca por soluciones en temperaturas extremas y muy altas justo cuando las variaciones de energía son altas y ocurren de manera caótica. Por lo tanto, QP es identificada como una fase de caos. La temperatura en QP decrece con una función exponencial modelada en las ecuaciones 13,14 y 15.

$$C(k+1) = \alpha\gamma_k C(k) \tag{13}$$

$$\gamma_{k=}(1-\tau_k) \tag{14}$$

$$\tau_k = \tau_{k-1}^2 \tag{15}$$

Cuando las variaciones de energía de una solución propuesta a otra son menos inestables QP pasa a la fase AP.

La fase AP busca soluciones óptimas en temperaturas altas y bajas donde las variaciones de energía alcanzan el equilibrio dinámico. En MQA, QP y AP están subdivididos en varias subfases al decrementar el parámetro de temperatura hasta alcanzar un estado cuasi-estable en la primera fase (QP) o el equilibrio dinámico en la segunda fase (AP). El equilibrio dinámico es conocido como estado de congelación, al ser alcanzado por MQA dicho estado el algoritmo termina su búsqueda.

MultiQuenching Annealing utiliza un método analítico para sintonizar los parámetros del algoritmo conocido como enfoque de Markov. Las temperaturas inicial T_0 y final T_f se determinan como en los problemas de satisfactibilidad, el número de iteraciones de cada subfase es sintonizado con un modelo analítico. Durante QP, las ecuaciones 13,14 y 15 son aplicadas con diferentes valores de α de acuerdo con la relación de soluciones aceptadas frente a las soluciones propuestas. Estos valores están en el rango de (0.7, 0.8).

$$C(k+1) = \alpha C(k) \tag{16}$$

Es importante aplicar la segunda fase (AP) después de finalizar QP, para obtener soluciones de gran calidad para PDP ya que aquí se encontrarán las soluciones más cercanas a la estructura nativa. Durante AP, la temperatura decrece gradualmente aplicando la ecuación 16, pero con diferentes valores de α que los utilizados en QP, estos valores están en el rango de (0.9, 0.98). En MQA, la fase actual es determinada por τ, el valor inicial para τ en cada subfase es muy cercano a uno (0.999). Debido a que en cada MC se actualiza el valor de τ utilizando la ecuación 15 eventualmente τ convergirá a cero, este valor es un objetivo para cambiar el valor de α de su valor inferior (0.7) hasta el superior (0.98). Por lo tanto una vez que α alcanza el mayor valor, el valor de τ será el más bajo.

2.3.3.1 Algoritmo MultiQuenching Annealing

En MQA (Frausto-Solis, Soberon-Mainero, and Liñan-Garcia 2009) se aplica una regla muy simple para las fases del algoritmo; si $\tau > 0$ se aplica la fase de enfriamiento QP,

pero si $\tau \approx 0$ se aplica la fase de recocido AP. A continuación se presenta el algoritmo MQA:

Algoritmo 3. *Algoritmo MultiQuenching Annealing*

```
1.  T_k = T_i = Analytical tunning ( ) //temperatura inicial
2.  T_f = Analytical tunning ( )  //temperatura final
3.  Initial Energy = function_energy()
4.  While (T_k > T_f) do
5.          Metropolis_cycle()
6.          Temperature_{k+1}=α (1- τ )temperature_k
7.          τ = τ^2
8.          If > 0 then
9.                  Apply_quenching phase()
10.                 Else
11.                         Apply_annealing_phase()
12.         End if
13. End do
```

2.3.4 Algoritmos Genéticos

Los algoritmos genéticos o GA (del inglés, Genetic Algorithms) son algoritmos evolutivos que simulan el proceso de los seres vivos (Goldberg 1989). De acuerdo a la teoría de la evolución, los individuos que no se adaptan al ambiente tienden a la extinción. Por otro lado, los individuos más adaptados sobreviven y se reproducen produciendo una mejor generación de individuos con genes heredados para su supervivencia (Singh Bhalla and Aggarwal 2013).

Los algoritmos genéticos son aplicados para resolver problemas de optimización cuya función objetivo es desconocida o siendo conocida es difícil de derivar. Las variables del problema son codificadas como genes de un individuo y la aptitud de este se establece mediante una función de aptitud o *fitness*. Cada individuo es una solución candidata con un grado de utilidad determinada por la función de *fitness* (Maulik and Bandyopadhyay 2000). Un algoritmo genético básico se presenta en el Algoritmo 4.

Algoritmo 4. *Genetic Algorithm*

```
1.  Begin
2.  t=0
3.  Generate (P(t)) /* Población inicial.
4.  Computar fitness P(t)
5.  t=t+1
6.  Do
7.          Select P(t) from P(t-1)
```

8. **Crossover** P(t)
9. **Mutate** P(t)
10. **While** Población no ha convergido
11. **End**
12. **End**

2.3.5 Algoritmo Scatter Search

Búsqueda Dispersa o en inglés Scatter Search (SS) es una estrategia determinística que ha sido aplicada satisfactoriamente en problemas de optimización y combinatorios. La primera descripción del método fue publicada en 1977 por Fred Glover donde establece los principios de SS (Glover Manuel Laguna and Martí 2000). SS opera sobre un conjunto de soluciones, llamado conjunto de referencia, combinando éstas para crear nuevas soluciones de modo que mejoren a las que las originaron. Por tal motivo se le considera como un método evolutivo, pero a diferencia de los algoritmos genéticos, los fundamentos de SS son elecciones sistemáticas y estratégicas sobre un conjunto pequeño de soluciones, habitualmente se utiliza un conjunto de 10 soluciones.

Una de las características más notables de SS es que se basa en integrar la combinación de soluciones con la búsqueda local. Aunque en diseños avanzados esta búsqueda local puede contener una estructura de memoria (Tabú Search), no es necesario que así sea, limitándose, en la mayoría de los casos a una búsqueda local convencional. (Martí and Laguna 2003). Scatter Search consta básicamente de cinco métodos:

- *Método de Generación de Diversificación:* Este método es generador de soluciones diversas. Este método genera un conjunto P de 100 soluciones diversas, del cual se extrae un subconjunto de 10 que se denomina Conjunto de Referencia (*RefSet).*
- *Método de Mejora:* es un método que mejora las soluciones mediante una búsqueda local. Este método podrá encontrar soluciones factibles a partir de soluciones no factibles y buscara mejorar el valor de la solución encontrada. Si el método no puede mejorar la solución inicial, se tomara como resultado la propia solución inicial.
- *Método de actualización del conjunto de referencia**:*** método que crea y actualiza el *RefSet.* A partir del conjunto de soluciones diversas P se extrae el *RefSet* de acuerdo a un criterio de soluciones con calidad y diversidad. Las soluciones son ordenadas de mejor a peor calidad.
- *Método de Generación de Subconjuntos:* Método que genera subconjuntos del *RefSet* a los que se aplicara un método de combinación. SS examina de forma exhaustiva las combinaciones del *RefSet*. Este método especifica la forma en la que se seleccionan los subconjuntos para después combinarlos mediante el método.
- *Método de combinación de soluciones:* este es un método que combina todas las soluciones del *RefSet*, para ello, considera los subconjuntos formados por el método de generación de subconjuntos y se les aplica el método de combinación. La

solución o soluciones obtenidas de la combinación son introducidas en el *RefSet* o almacenadas temporalmente en una lista hasta que se realizan todas las combinaciones para después analizar qué soluciones entraran al *RefSet*. En la figura 4 encontramos el esquema de Scatter Search donde se muestran los métodos descritos.

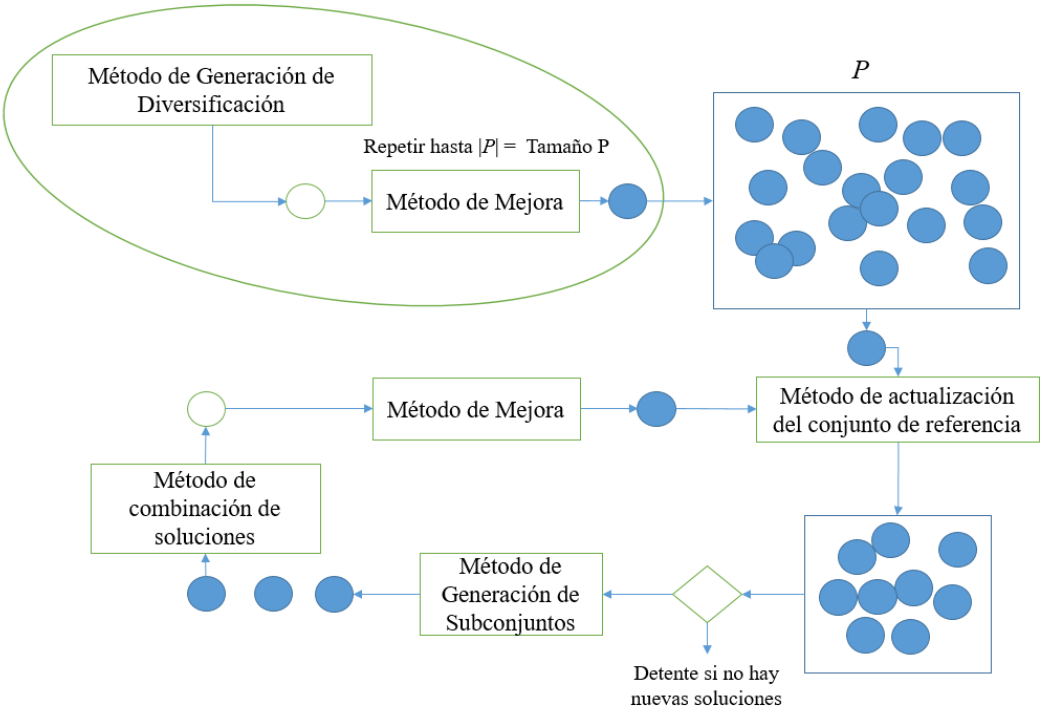


Figura 2.4 *Esquema del Método Scatter Search*

2.4 Cómputo paralelo

En la necesidad de más poder computacional, los desarrolladores de sistemas computacionales pensaron en una forma de unir sus máquinas existentes para que trabajaran en forma conjunta. Este pensamiento dio origen a maquinas paralelas y un nuevo tipo de programación (Hermanns 2002). El cómputo paralelo es un área de búsqueda y desarrollo que aún hoy sigue siendo un objetivo difícil de alcanzar. Se necesita de una programación cuidadosa y tener un buen conocimiento acerca de la arquitectura de la computadora para lograr un buen desempeño (Zomaya 2004).

2.4.1 Arquitectura del Cómputo Paralelo

El objetivo del cómputo paralelo es claro, resolver grandes problemas en un menor tiempo. Pero debido a una gran variedad de arquitecturas existentes, la programación paralela puede resultar difícil. Podemos identificar tres grandes tipos:

- *Arquitectura SM (Shared-Memory):* Estas máquinas paralelas están construidas sobre un conjunto de procesadores que tienen acceso a una memoria común.
- *Arquitectura de memoria distribuida***:** en estas máquinas paralelas cada procesador posee una memoria privada y se intercambia la información entre los procesadores mediante mensajes.
- *Arquitectura Hibrida DSM (Distribuited-Shared Memory):* son máquinas que emplean arquitecturas SM y de memoria distribuida. Es la conexión en red de múltiples maquinas con arquitectura SM, como cada procesador solo conoce su propia memoria, utiliza la red de comunicación para mover datos de una maquina a otra como en la arquitectura de memoria distribuida (Blaise Barney 2017).

Existen diferentes estándares de programación que toman ventajas de cada arquitectura. Uno de esos estándares conocido como OpenMP tiene pocos años en la industria y su objetivo es ser una base para el desarrollo de programas paralelos en máquinas con la arquitectura SM. A continuación describiremos los dos principales estándares utilizados en este trabajo.

2.4.2 OpenMP

Anteriormente cada proveedor de máquinas con arquitectura SM desarrollaba su propio “estándar” de directivas del compilador y librerías, esto permitía al programador usar las capacidades de la máquina. En un intento por estandarizar se creó ANSI X3H5 pero debido al poco soporte nunca fue formalmente adoptado por la comunidad.

En 1996-1997 renace el interés en una interfaz de programación SM estándar. OpenMP es el resultado de un gran acuerdo entre los proveedores de hardware y desarrolladores de compiladores y es considerado como un estándar de la industria. Es un conjunto de directivas de compilador, rutinas de librerías y variables de entorno que son usadas para especificar el paralelismo en programas de lenguajes como Fortran, C y C++. OpenMP consolida todo esto es una sola sintaxis y semántica creando una portabilidad de una sola fuente para el paralelismo SM. OpenMP aborda la incapacidad que tenían anteriores conjuntos de directivas de lidiar con el paralelismo de grano grueso. En el pasado, se creía que la programación SM se limitaba solamente al paralelismo de grano fino, debido al apoyo tan limitado que existía (Hermanns 2002).

- *Paralelismo de grano grueso:* el paralelismo en el programa es alcanzado a través de la descomposición del dominio objetivo en un conjunto de subdominios que se distribuye entre los procesadores de la máquina.

- *Paralelismo de grano fino:* el paralelismo en el programa se logra distribuyendo el trabajo de los ciclos del programa en diferentes procesadores, para que cada procesador compute parte de las iteraciones.

2.4.3 MPI

MPI es una biblioteca de paso de mensajes estándar (University of Tennessee 2015). MPI utiliza la arquitectura de memoria distribuida para el paso de mensajes, en donde los datos se trasladan del espacio de direcciones de un proceso a otro proceso a través de operaciones cooperativas en cada proceso. Las extensiones del modelo Message-Passing se proveen por medio de operaciones colectivas, operaciones de acceso a memoria remota, creación dinámica de procesos y entradas/salidas en paralelo.

MPI es una especificación, no una implementación. Esta especificación es para una interfaz de biblioteca. MPI no es un lenguaje y todas las operaciones de MPI se expresan como funciones, subrutinas o métodos, de acuerdo con los enlaces de idiomas apropiados para C y Fortran que ya forman parte del estándar MPI. Este estándar ha sido definido por la comunidad de proveedores, científicos y desarrolladores. El objetivo principal de MPI es establecer un estándar portable, eficiente y flexible para desarrollar programas Message-Passing.

El estándar MPI ha pasado por varias revisiones desde su creación en 1980'. Actualmente la versión más reciente es la MPI-3.x. Originalmente MPI fue diseñado para la arquitectura de memoria distribuida, muy popular en los 1980’s. Pero con la creación de arquitecturas hibridas de SM y memoria distribuida, MPI adaptó su librería para aceptar ambas arquitecturas. Actualmente acepta los tres tipos de arquitectura (Barney 2017).

Capitulo 3. Estado del Arte

Para resolver el problema de doblado de proteínas se han utilizado diferentes tipos de métodos heurísticos, entre ellos tenemos: algoritmos genéticos, recocido simulado, redes neuronales y búsqueda tabú entre otros. A continuación se citan los trabajos que se encuentran en el estado del arte de PDP.

3.1 CASP

Desde 1994 existe una competencia internacional llamada CASP (del inglés, Critical Assessment of protein Structure Predicction) cuyo objetivo es establecer el estado del arte actual con respecto a la predicción de estructuras de proteínas, identificando el progreso que se ha hecho y destacar hacia donde se deben enfocar los esfuerzos futuros (“Protein Structure Prediction Center” 2017).

CASP provee a los participantes de secuencias de aminoácidos para estructuras desconocidas, y estos regresan un modelo de la estructura para esas secuencias. Los modelos son comparados con el nuevo modelo que ha sido determinado experimentalmente. CASP finalmente publica los resultados de cada equipo. Hasta nuestros días, han pasado doce CASP, la competencia ocurre cada dos años. En el CASP XI casi se mostraron 60,000 modelos de 207 grupos de investigación para los 100 objetivos que propuso CASP (Moult et al. 2016).

Para la evaluación de los modelos estructurales CASP se divide en las siguientes categorías:

- Modelos basados en plantillas homólogas
- Modelos libres sin detección de plantillas homólogas

En la Tabla 3.1 se presenta un listado de los mejores servidores y sus características. El listado muestra a los mejores hasta el momento en la comunidad científica a nivel mundial para resolver PDP.

Tabla 3.1 *Mejores servidores públicos de PDP certificados por CASP*

Autor	Servidores de Predicción de Estructuras	Modelo	Método
Yang Zhang, 2008, (Bioinformatics and Zhang 2017)	**I-TASSER**	Alineación de hilado Perfil-Perfil (PPA) y modelo iterativo basado en plantilla	Monte Carlo – Recocido Simulado
David E. Kim, 2004, (Kim, Chivian, and Baker 2004)	**ROBETTA**	Ginzu (Modelo Homologo y Modelo Novo)	Multiple sequence alignment (MSA)
Johannes Söding, 2005, (Soding, Biegert, and Lupas 2005)	**HHpred**	Modelo Ab Initio con FASTA o formato PSI-BLAST	HMM (Hidden Markov Model)
Hongyi Zhou, 2007, (Zhou, Pandit, et al. 2007)	**METATASER**	SPARKS, SP^3, PROSPECTOR_3 métodos de hilado y el enfoque 3D-Jury.	Parallel Hyperbolic (Tipo Recocido) Monte Carlo
Zheng Wang, 2010, (Wang, Eickholt, and Cheng 2010)	**MULTICOM**	Modelo Basado en Plantilla	Hhsearch o Hidden Markov Model
Jesper Lundström, 2001, (Lundström et al. 2008)	**Pcons**	Modelo Homólogo	Redes Neuronales
Kevin Karplus, 2009, (Karplus 2009)	**SAM-T08**	Modelo Homólogo	Hidden Markov Model
Krzysztof Ginalski, 2003, (Ginalski et al. 2003)	**3D-Jury**	Modelo Ab Initio	CAFASP - Consensus groups
David Jones, 1998, (Jones 1998)	**THREADER**	Modelo de Hilado	Double dynamic Programming
Morten Källberg, 2012, (Källberg et al. 2012)	**RaptorX**	Modelo basado en plantilla	Conditional (Markov) Random Fields (CRFs)

3.2 Algoritmos híbridos para PDP con Recocido Simulado y Algoritmos Evolutivos

Actualmente los algoritmos que predicen la estructura tridimensional para proteínas pequeñas lo hacen con una buena precisión. En los años 80's los grupos de investigación comenzaron a utilizar el algoritmo Monte Carlo para el péptido de cinco aminoácidos llamado Met Enkephalin (Morales, Garduño-Juárez, and Romero 1991) el cual es un caso de prueba para los algoritmos, ya que tiene 10^{11} mínimos locales (Nayeem, Vila, and Scheraga 1991).

Para PDP tanto el algoritmo de Recocido Simulado como los Algoritmos Genéticos han mostrado tener un mejor desempeño que otros algoritmos (Zhou, Skolnick, et al. 2007). En la Tabla 3.2 se muestra una recopilación de los algoritmos híbridos que existen en la literatura. La Tabla 3.2 contiene en las columnas uno y dos, los autores y títulos de los trabajos analizados, en las columnas tres y cuarto tenemos los modelos y el método de solución correspondiente. Finalmente en la columna cinco se muestran las proteínas probadas y el número máximo de aminoácidos que fue estudiado.

Tabla 3.2 *Algoritmos Híbridos de tipo Recocido y Evolutivos para PDP.*

Autor	Titulo	Modelo	Enfoque de Hibridación	Proteínas	Max Aminoácidos
Flavia P. Agostini, 2006, (Agostini et al. 2006)	Generalized Simulated Annealing Applied to Protein Folding Studies	Grid Model	Simulated Annealing y Generalized Statistical Mechanics (Tipo Recocido)	18-alanine	Péptido pequeño
Luis B. Morales , 1991, (Morales, Garduño-Juárez, and Romero 1991)	Applications of Simulated Annealing to the Multiple-Minima Problem in Small Peptides	Ab Initio	Técnica Simulated Annealing que incluye Anti- correlaciones (Tipo Recocido)	Met- Enkephalin Leu- Enkephalin N-acetyl-N'-methyl-aspartic acid TRH(+) N-acetyl-N'-methyl-glycineamide, N-acetyl-N'-methyl-alanineamide	5
Lixin Zhan, 2006, (Zhan, Chen, and Liu 2006)	Conformational study of Met-enkephalin based on the ECEPP force fields.	Ab Initio	Basin Paving Method y Monte Carlo (Tipo Recocido)	Met-Enkephalin	5
Peng Fengbin, 2013, (Fengbin et al. 2013)	Protein Folding Study Based on Parallel Group Annealing Algorithms	Ab Initio	Simulated annealing y message passing interface (MPI) (Tipo Recocido)	Met- Enkephalin	5
Juan Frausto-Solis , 2007, (Frausto-Solis and Román 2007)	Analytically tuned simulated annealing applied to the protein folding problem	Ab Initio	Markov Chain y Simulated Annealing (Tipo Recocido)	Met Enkephalin C-peptide	13
Yoshitake Sakae, 2015, (Sakae et al. 2015)	Combination of genetic crossover and replica-exchange method for conformational search of protein systems	Ab Initio	Simulated Annealing y Genetic Algorithm (Tipo Recocido y Genético)	Trp-cage	20
Yuko Okamoto, 1999, (OKAMOTO 1999)	Tackling The Multiple-Minima Problem In Protein Folding By Monte Carlo Simulated Annealing And Generalized-Ensemble Algorithms	Ab Initio	Monte Carlo Simulated Annealing y Multicanonical Algorithms (Tipo Recocido)	C-peptide Fragment of BPTI (16-36)	20
Tomoyuki Hiroyasu, 2002, (Hiroyasu et al. 2002)	Energy Minimization of Protein Tertiary Structure by Parallel Simulated Annealing using Genetic Crossover	Ab Initio	Parallel Simulated Annealing with an operator of Genetic Algorithm (GA), (Tipo Recocido y Genético})	Met-Enkephalin C-peptide Parathyroid Hormone Fragment	34
Ge-Fei Hao, 2015, (Hao et al. 2015)	Multiple Simulated Annealing- Molecular Dynamics (MSA-MD) for Conformational Space Search of Peptide and Miniprotein	Ab Initio	Simulated Annealing-Molecular Dynamics (SA-MD) (Tipo Recocido)	ALPHA1 Trp-cage PolyAla 1UAO 1E0Q 1ERD 1GAB	40
Juan Frausto-Solis, 2015, (Frausto-Solis et al. 2015)	Golden Ratio Simulated Annealing for Protein Folding Problem	Ab Initio	Golden Ratio y Simulated annealing (Tipo Recocido)	Met-Enkephalin C-Peptide 1EOG 1ENH 1BDD	60
Juan Frausto-Solis , 2014, (Frausto-Solis et al. 2014)	ChaoticMultiquenching Annealing Applied to the Protein Folding Problem	Ab Initio	Simulated annealing, MultiQuenching annealing y Chaotic Local Search (Tipo Recocido)	Met- Enkephalin Proinsulin T0549 T0335 T0281	90
Juan Frausto-Solis, 2016, (Frausto-Solis et al. 2016)	Multiphase Simulated Annealing Based on Boltzmann and Bose-Einstein Distribution Applied to Protein Folding Problem	Ab Initio	Multiphase Simulated Annealing Algorithm using Boltzmann y Bose-Einstein distributions (MPSABBE) (Tipo Recocido)	Met-Enkephalin Proinsulin T0549 T0335 T0281.	90

Como se puede observar en la Tabla 3.2, los algoritmos con el enfoque de Recocido Simulado o Simulated Annealing han sido aplicados para tratar péptidos y proteínas. Algunos de ellos hibridados con enfoques de algoritmos evolutivos tales como los algoritmos genéticos y también con tecnologías como MPI o técnicas de paralelización para poder explotar el poder de cómputo con el que se cuenta actualmente. Considerando el buen desempeño que han tenido los algoritmos basados en recocido simulado en este trabajo se desarrolló un algoritmo híbrido evolutivo; con la idea de aprovechar la capacidad de exploración de los algoritmos poblacionales y evitar los altos costos computacionales que tienen los algoritmos genéticos se desarrolló un algoritmo basado en búsqueda dispersa que trabaja con poblaciones pequeñas que exploran en forma amplia el espacio de soluciones y se hibrida con un algoritmo multiquenching que ha mostrado un excelente desempeño para PDP con el fin de intensificar en la vecindad de buenas soluciones.

Capitulo 4. Metaheurística MultiQuenching paralela híbrida poblacional propuesta

En este capítulo se describen los algoritmos generados en el presente trabajo para el problema de doblado de proteínas. Se utilizaron algoritmos evolutivos para la construcción de soluciones iniciales y algoritmos estocásticos del tipo recocido simulado para la mejora de estas soluciones.

4.1 Parallel GMQA (Genetic-MultiQuenching Annealing)

Para la construcción de esta metaheurística es necesario explicar cómo cada estrategia fue aplicada al problema de doblado de proteínas. Comenzaremos por Genetic Algorithm, seguido de MultiQuenching Annealing Algorithm y por último se muestra la hibridación de los algoritmos y la aplicación del paralelismo en el nuevo algoritmo.

4.1.1 Algoritmo Genético para PDP

Un algoritmo Genético (Goldberg 1989) genera una población de soluciones aleatorias iniciales las cuales evoluciona mediante la aplicación de operaciones como la cruza, mutación, selección entre otras. Mediante un criterio llamado fitness se determina a los mejores individuos y se le aplica a toda la población los operadores que los lleva a mejorar en cada generación. El algoritmo termina cuando la población converge y devuelve la mejor solución. El algoritmo descrito se puede encontrar en algoritmo 4.

A continuación en el algoritmo 5 se calcula la menor energía para PDP mediante un algoritmo genético. Al igual que en el algoritmo de GA, basándose en la literatura se crea una población de *P* soluciones iniciales, decodificando cada solución como un conjunto de ángulos que corresponderán a la estructura de la proteína. La cantidad de ángulos está determinada por la instancia del problema, por ejemplo:

#Met-Enkephalin *Tyr Gly Gly Phe Met*

Dicha instancia genera el conjunto de ángulos de la estructura de la proteína Met Enkephalin. Como función de aptitud o *fitness* se evalúa cada solución de la población calculando el valor de energía (Energy value) de cada solución. A lo largo de *n* generaciones evolucionamos aplicando los operadores de Selección por torneo, Cruza SBX

y Mutación basada en la probabilidad *m*, para cada nueva generación. En cada generación se obtiene el mejor individuo el cual tiene el mejor fitness (el recíproco del valor de energía) y el conjunto de ángulos que lo conforman. El algoritmo termina cuando alcanza *n* generaciones. Una vez finalizado el algoritmo, este devuelve el mejor individuo y los valores de sus genes los que se decodifican en una configuración de ángulos.

Algoritmo 5. *Algoritmo Genético para PDP*

```
1.  //Chromosome code is created from peptide structure
2.  Generate (P(0))
3.  t:=0
4.  while not Termination_criterion(P(t)) do
5.  Evaluate(P(t) //Energy value as fitness function
6.  P'(t) = Selecction (P(t))
7.  P'(t) = Apply_Reproduction(P'(t))
8.  P(t+1) = Replace (P(t), P'(t))
9.  t=t+1
10. endwhile
11. BestSolution=FindBest(P(t) //Finding the best individual
12. return Best_Energy(P(t))  //Best energy
13. return BestSolution  //Configuration of the BestSolution
```

4.1.2 MultiQuenching Annealing para PDP

El algoritmo MultiQuenching para PDP (Frausto-Solis, Soberon-Mainero, and Liñan-Garcia 2009) se muestra en el algoritmo 6. Multiquenching ha sido aplicado a PDP con éxito.

Algoritmo 6. *Algoritmo MultiQuenching Annealing para PDP*

1. $T_k = T_i$ = Analytical tunning ( ) //temperatura inicial ; T_f = Analytical tunning ( ) //temperatura final
2. $T_{treshold}$ = Set_value ( ); α _value = valor_inicial_alpha, τ = valor_inicial_tao
3. S_i= Solución_inicial(*secuencia_aminoácidos*), E_i = Energia (S_i),S_{min}= *Si* , *Emin* = Energy (S_{min})
4. **While** ($T_k > T_f$) **do**
5. convergencia = false
6. **While** (convergencia = false) **do** //Ciclo_Metrópolis; $0< \tau_k <1$, $0.7<\alpha <1$
7. *Sk* = Perturbación (*Si*); E_k = Energia (S_k); *Delta_E = Ek –Ei*
8. **If** (*Delta_E* ≤ 0) **then** *Si = Sj*
9. **else if** ($e^{Delta\_E/Tk}$) > random [0,1] **then** *Si=Sj*
10. **If** (*Si* < *Smin*); guarda (Smin); guarda (*Emin* = Energia (*Smin*))
11. **If** ($T \leq T_{treshold}$) **then** $T_{k+1} = \alpha\ T_k$ //Applica_AP( );
12. **else If** ($\tau \leq \varepsilon$) **then** $\tau_k = \tau^2_{k-1}$; $T_{k+1} = \alpha\ (1- \tau_k)T_k$ **endif** //Applica_QP( )
13. TestConvergencia(convergencia)
14. **end_while**
15. **end_do**

El algoritmo toma la instancia del problema para crear una solución inicial aleatoria (línea 3). Crea un ciclo en el cual perturba la solución inicial y la somete a un criterio de aceptación estadístico conocido como criterio de Boltzman para aceptar o rechazar la solución perturbada (línea 8). El proceso se repite de manera iterativa, tal como se explicó en la sección 2.3.3.

4.1.3 GMQA para PDP

El algoritmo Genetic-MultiQuenching Annealing (GMQA) para PDP desarrollado en el trabajo reportado en esta tesis combina las estrategias de MultiQuenching Annealing (Frausto-Solis, Soberon-Mainero, and Liñan-Garcia 2009) y Genetic Algorithms (GA) (Goldberg 1989). En GMQA las estrategias trabajan en cooperación a lo largo del proceso del algoritmo hasta que este alcanza su criterio de convergencia. En el Algoritmo 7 se muestra el seudocódigo de GMQA.

El algoritmo toma la instancia del problema y codifica los ángulos en forma de genes de individuos creando una población inicial aleatoria (línea 4) como tradicionalmente se define en la literatura los GA (Goldberg 1989). Esta población inicial se somete al proceso evolutivo del GA y una vez terminado este se toma al mejor individuo de dicho proceso. El código genético del mejor individuo del GA es decodificado en ángulos, los que se toman como entrada del MQA para reiniciar el proceso de búsqueda, ahora empleando esta heurística. La solución producida por esta etapa es tomada como salida del proceso. Tanto MQA como GA son algoritmos diseñados para mejorar soluciones durante todo el proceso del algoritmo, cuando el algoritmo finaliza se habrá acercado a la solución óptima. Sin embargo, la solución encontrada podría estar aún lejos del óptimo global. Con GMQA se busca incorporar diversificación, para evitar quedar atrapados en óptimos locales. A continuación, se describe el nuevo algoritmo hibrido propuesto GMQA:

1. GMQA obtiene la secuencia de aminoácidos $a_1, a_2,\ldots, a_n$. de la instancia propuesta y transforma esta información en la codificación genética de los individuos de la población. Esta codificación representa los ángulos de la proteína.
2. Se crea una población de individuos o soluciones iniciales aleatorias.
3. Para evolucionar a la población se utilizan los operadores genéticos propuestos en la literatura de GA: selección, cruza y mutación. Esta evolución genera cien nuevas generaciones con el propósito de buscar en un espacio global de soluciones.
4. Al finalizar la última generación el algoritmo GMQA obtiene la solución resultante del proceso evolutivo para someterla a un nuevo proceso. Para este momento, la solución contará con un valor de energía menor a la inicial, pero aún lejos del óptimo.

5. GMQA somete la solución a un proceso estocástico llamado MQA que ha demostrado en el estado del arte acercarse al óptimo global. GMQA utiliza la probabilidad de Boltzman para diversificar, al permitir la aceptación de soluciones peores a la solución en curso. El principal propósito de esta etapa es realizar una búsqueda local con buenos resultados. También en este paso se logra escapar de mínimos locales.
6. Una vez que GMQA alcance la temperatura final del proceso el algoritmo devuelve la mejor solución.

Como podemos apreciar en el Algoritmo 7 GMQA sigue de forma general los enfoques de GA y MQA pero los combina en la búsqueda de soluciones de mejor calidad a las entregadas por cada enfoque por separado.

Algoritmo 7. *Algoritmo Genético MultiQuenching Annealing para PDP*

```
1.  BestEnergy = infinite
2.  GlobalBest = null
3.  Setting = ( currtem, alfa, tao, beta)
4.  Configs [][] = init_population ( amino_seq )
5.  For ( i=1 to 100 )
6.      Selection ( Configs )
7.      Crossover ( Config1, Config2 )
8.      Mutate ( Configs )
9.      iterBest = GetBest ( Configs )
10.     If  ( energy ( iterbest ) < BestEnergy ) then
11.             GlobalBest = iterBest
12.             BestEnergy = energy(iterBest)
13.     End if
14. End do
15. Angles = getAngles( GlobalBest )
16. While ( Temperature_k > criteria_stop ) do
17.     Metropolis_cycle(angles, currtem, alfa, tao, beta)
18.     Temperature_{k+1}= α (1- τ )temperature_k
19.     τ = τ^2
20.     If τ > 0 then
21.     Apply_quenching phase(angles, currtem, alfa ,tao, beta)
22.     Else
23.         Apply_annealing_phase(angles, currtem, alfa, tao, beta)
24.     End if
25. End do
26. Return OptimalConfigurationAngles( )
```

4.1.4 Algoritmo PG/MQA

El algoritmo Parallel Genetic-MultiQuenching Annealing (PG/MQA) es la versión en paralelo del algoritmo hibrido GMQA anteriormente desarrollado. PG/MQA optimiza la búsqueda en el espacio de soluciones al realizar procesos en paralelo utilizando el estándar de programación de memoria compartida OpenMP.

Al igual que en GMQA, el algoritmo PG/MQA toma la información de la secuencia de aminoácidos para crear los *m* individuos o soluciones iniciales que conformarán a la población inicial. Estas soluciones iniciales respetan las restricciones de Ramachandran lo que las hace factibles. En cada solución se calcula el valor de energía, el inverso de este valor será nuestro parámetro de *fitness* o aptitud para la población. Consideraremos al menor valor de energía como el mejor valor de *fitness*.

Algoritmo 8. *PG/MQA Parallel Genetic-MultiQuenching Annealing para PDP*

```
1.  BestEnergy = infinite
2.  GlobalBest = null
3.  Setting = ( currtem, alfa, tao, beta)
4.  Call OMP_set_num_threads(n);
5.  Configs [][] = init_population ( amino_seq )
6.  For ( i=1 to 100 )
7.        Selection ( Configs )
8.       Crossover ( Config1, Config2 )
9.       Mutate ( Configs )
10.      iterBest = GetBest ( Configs )
11.       If  ( energy ( iterbest ) < BestEnergy ) then
12.            GlobalBest = iterBest
13.      BestEnergy = energy(iterBest)
14.      End if
15. End for
16. TID = OMP_GET_THREAD_NUM()
17. If (TID = 0) then
18.       NTHREADS = OMP_GET_NUM_THREADS()
19. End if
20.      angles = getAngles( GlobalBest )
21.      While ( Temperature_k > criteria_stop ) do
22.          Metropolis_cycle(angles, currtem, alfa, tao, beta)
23.          Temperature_{k+1}= α (1- τ )temperature_k
24.          τ = τ^2
25.          If τ > 0 then
26.              Apply_quenching phase(angles, currtem, alfa ,tao, beta)
27.              Else
28.                    Apply_annealing_phase(angles, currtem, alfa, tao, beta)
29.          End if
```

30. ***End do***
31. END PARALLEL
32. ***Return*** *OptimalConfigurationAngles( )*

Una vez inicializada la población total, sometemos las soluciones a un proceso de evolución utilizando los operadores de selección, cruza y mutación hasta alcanzar el criterio de *n* generaciones. Una vez finalizado el proceso evolutivo PG/MQA selecciona la solución con mejor aptitud, la cual es la que posea el menor valor de energía, y se reporta su configuración genética. Esta solución es llamada *Global-Best.* Siguiendo con el algoritmo, *Global-Best* es sometida a procesos paralelos de MultiQuenching. Mediante el estándar OpenMP utilizamos un proceso como maestro de los demás procesos esclavos. Creamos *n* procesos MQA, la variabilidad de cada proceso está dada por la semilla aleatoria de cada proceso. Esto con la finalidad de poder realizar búsquedas locales paralelas en el espacio de búsqueda a partir de *Global-Best*.

Cada proceso reporta al proceso maestro el valor de energía al cual pudieron converger mediante el proceso estocástico MQA. El proceso maestro evalúa el valor de energía (Energy()) de *n* soluciones seleccionando el valor con menor energía, llamando a esta *Best-Solution* y reportando dicha configuración. A continuación, se muestra el algoritmo PG/MQA para PDP.

4.2 Algoritmo Scatter Search MQA Paralelo

En el desarrollo de este trabajo se creó un segundo algoritmo nuevo que incorpora otro enfoque evolutivo de la literatura llamado Scatter Search (Martí and Laguna 2003) y además como estándar de programación usa MPI. A continuación describiremos la construcción e incorporación de todas las heurísticas utilizadas en el algoritmo PSS/MQA (Parallel Scatter Search-MultiQuenching Annealing) para PDP.

4.2.1 PSS/MQA para PDP

Este algoritmo paralelo utiliza memoria distribuida con MPI. En el algoritmo 9 se muestra a detalle el funcionamiento de PSS/MQA, utilizando como base el diagrama de Scatter Search en la figura 5 podemos ver el nuevo algoritmo PSS/MQA que toma como base el proceso de SS pero incorpora el proceso de MultiQuenching y paralelismo en la búsqueda de soluciones.

Algoritmo 9. PSS/MQA Parallel Scatter Search - MultiQuenching Annealing para PDP

1. BestEnergy = infinite
2. GlobalBest = null

```
3.  mpi_init(rank,ierr) //Inicializar entorno paralelo
4.  call init_molecule(iabin,grpn,grpc) //Inicializar secuencia
5.  call setting(currtem,alfa,tao,beta) //Inicializar parámetros
6.  IF rank.eq.0) //Proceso maestro
7.              call initpop(ngene, POP) //Crear población inicial
8.              call ga(ngene, bestener,bestofev,POP,FITNESS) //Mejorar población inicial con un GA
9.              call genrs(FITNESS,POP, ngene, REFSET, nrefs) //Crear conjunto de referencia
10.             call gensubs(REFSET, ngene, COUPLES, nchilds) // Crear subconjuntos
11.             call gencross(ngene,CHILDS,COUPLES, nchilds) //Cruza generalizada
12.             DO i=1, nchilds //Unir conjunto de referencia e hijos'
13.                 DO j=1,ngene
14.                     REFSET(j,nrefs+i) = CHILDS(j,i)
15.                 ENDDO
16.             ENDDO
17.             call bestind(ngene,ibest,ener,fitrs, REFSET, nrefs+nchilds) //Obtener mejor solución
18.             BestEnergy=ener
19.               DO j=1,ngene
20.                   bestindg(j) = REFSET(j,ibest)
21.             ENDDO
22. ENDIF
23. notImproving = 0    //Contar ciclos sin mejora
24. WHILE(notImproving < 5)  //Ciclo de mejora scatter
25.             IF(myrank.eq.0) THEN
26.                 DO p=1, nrefs + nchilds   // Enviar a procesos esclavos una solución para mejora
27.                     call MPI_SEND(REFSET(1,p), ngene, MPI_DOUBLE_PRECISION, p,
                        2,MPI_COMM_WORLD, ierr)  //Enviando a proceso p solución candidata
28.                 ENDDO
29.                 improving=.false.
30.                 DO tp=1, nrefs + nchilds   //Recibir resultado de cada proceso esclavo
31.                     call MPI_RECV(ener, 1, MPI_DOUBLE_PRECISION, tp, 3,
                        MPI_COMM_WORLD, ierr) //Recibiendo ener de proceso tp energía
                        calculada
32.                     call MPI_RECV(bestofev, mxvr, MPI_DOUBLE_PRECISION, tp, 4,
                        MPI_COMM_WORLD, ierr) //Recibiendo solución de proceso tp
33.                     //Analizando solución tp para ver si es máximo global
34.                     //Es mejor que su predecesor?
35.                     IF(ener < fitrs(tp)) THEN //Reemplazar predecesor
36.                         DO j=1,ngene
37.                             REFSET(j,tp) = bestofev(tp)
38.                         ENDDO
39.                         fitrs(tp)=ener
40.                         //Es mejor que el máximo global?
41.                         IF (ener .lt. BestEnergy) THEN //Reemplazar máximo global
42.                          //Encender bandera indicadora de mejora
43.                          improving = .true.
44.                          BestEnergy = ener
45.                             DO j=1,ngene
46.                                 bestindg(j) = REFSET(j,tp)
47.                             ENDDO
48.                         ENDIF
49.                     ENDIF
50.                   ENDDO
51.                 //Si hubo mejora resetear contador de ciclos sin mejora
52.                //Si no hubo mejora incrementar contador de ciclos sin mejora
53.                IF (improving.eqv..false.) THEN
54.                     notImproving = notImproving + 1
```

```
55.                     ELSE
56.                         notImproving = 0
57.                 ENDIF
58.             ENDIF !myrank = 0
59.             //Procesos esclavos
60.             IF(myrank.ne.0) THEN
61.                 call MPI_RECV(bestofev, ngene, MPI_DOUBLE_PRECISION, 0,tag2,
                    MPI_COMM_WORLD, status, ierr) //Reciben solución canditata para mejorar
62.                 DO j=1,ngene //Procesar solución candidata con MultiQuenching
63.                     vlvr(idvr(j)) = bestofev(j)
64.                 ENDDO
65.             call mqpp(bestofev,currtem,fase,alfa,tao,bbeta,eol,smin2,finalq)
66.             //Proceso p envía resultado a proceso maestro
67.             call MPI_SEND(smin2, 1, MPI_DOUBLE_PRECISION, 0, 3,MPI_COMM_WORLD,
                ierr) //Energía
68.             call MPI_SEND(bestofev, ngene, MPI_DOUBLE_PRECISION, 0, 4,
                MPI_COMM_WORLD, ierr) //Configuración
69.             write(*,*) 'Proceso ', p, ' envió resultado'
70.             ENDIF
71. ENDDO
72. CALL WriteSolution(bestind,BestEnergy)
73. mpi_finalize(ierr)
```

PSS/MQA aprovecha la paralelización MPI en el método de mejora del algoritmo, ya que por la complejidad del problema que se aborda si se procesara de forma secuencial en ese punto, resultaría en un algoritmo de un tiempo muy grande. Con el paralelismo aprovechamos los recursos de múltiples procesadores y cada hijo creado es mejorado en un proceso MultiQuenching en paralelo. En PSS/MQA el proceso maestro del algoritmo decide si ha llegado a un estado donde no mejora la solución global, después de no encontrar cambios significativos el algoritmo llega a término.

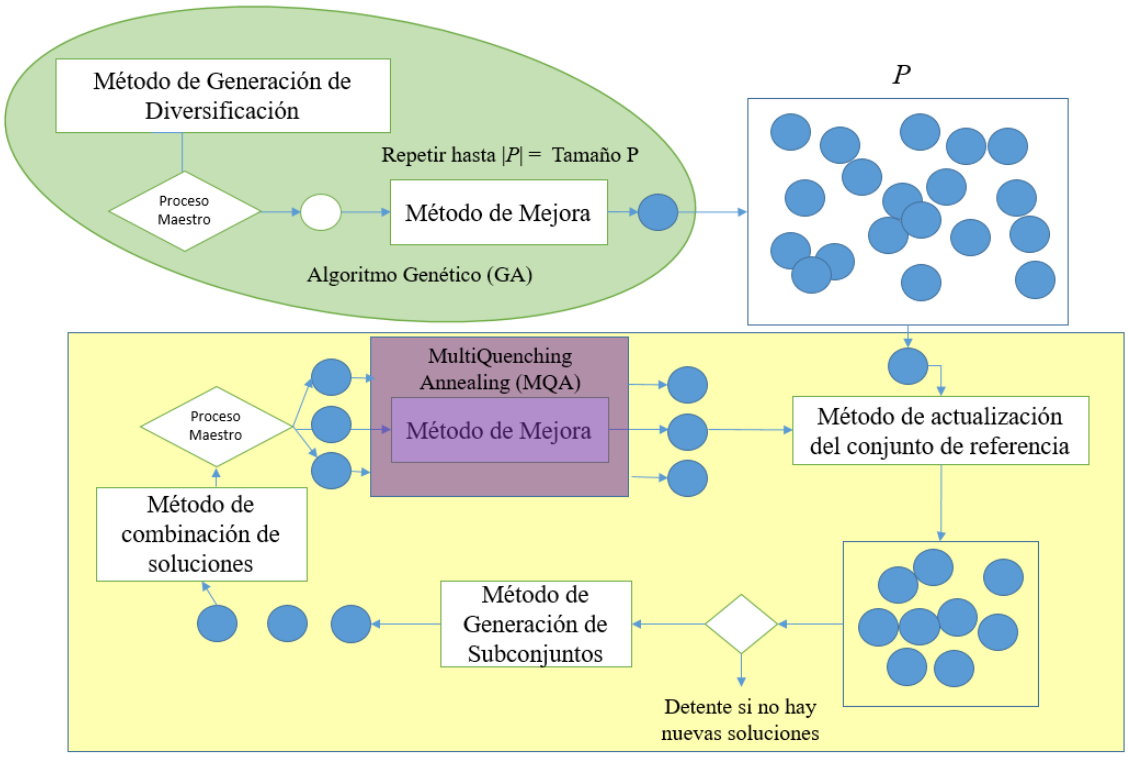


Figura 4.1 *Esquema de PSS/MQA para PDP*

Capitulo 5. Resultados Experimentales

5.1 SMMP

Para la implementación de este trabajo se utilizó el software libre Simple Molecular Mechanics for Proteins (SMMP) (Meinke et al. 2008), el cual es un paquete en lenguaje FORTRAN para la simulación de proteínas dentro de un modelo geométrico. La versión utilizada SMMPv.3.0 es una herramienta simple para investigadores y estudiantes.

Existen softwares que realizan simulaciones de proteínas entre los que están AMBER (Pearlman et al. 1995), CHARM (Brooks et al. 1983) y GROMACS (Berendsen, van der Spoel, and van Drunen 1995) los cuales cuentan con cientos de líneas de códigos. Para los dos primeros se necesita una licencia y el último se centra en simulaciones Molecular Dynamics.

SMMP es una alternativa ya muy bien establecida que se enfoca en métodos avanzados de Monte Carlo. SMMP fue introducido en 2001 (Eisenmenger et al. 2001) y fue mejorado en 2006 (Eisenmenger et al. 2006). EL objetivo del software es proveer un pequeño y fácil entorno de trabajo para implementar y probar nuevos algoritmos para PDP. Escrito en lenguaje Fortran y sin rutinas dependientes, puede ser compilado con cualquier compilador moderno de Fortran.

5.2 Métricas de desempeño en PDP

La similitud estructural de una proteína en PDP a menudo es medida por medio de la métrica Root Mean Squared Deviation (RMSD) o Global Distance Test Score (GDT_Score). Sin embargo por si solos estas medidas no ofrecen información acerca de cuan similar es la estructura probada con la estructura nativa (Xu and Zhang 2010). Existen otras métricas entre ellas:

- MaxSub score (Siew et al. 2000) ,
- MAMMOTH score (Ortiz, Strauss, and Olmea 2002)
- Dalí Z-Score (Holm and Sander 1995).

Con el propósito de desarrollar medidas sensitivas a la topología de las proteínas se desarrolló GDT-Score (Zemla et al. 1999), el cual cuenta el números de parejas de Cα con

una distancia < 1, 2,4 y 8Å después de superponerlas con el óptimo. MaxSub (Siew et al. 2000) , identifica el número máximo de subestructuras que tienen pares Cα < 3.5 Å. Estas dos métricas fueron ampliamente utilizadas por la comunidad CASP y sus experimentos para evaluar la precisión de modelado de las predicciones de estructura nativa. Sin embargo, los puntos de corte de distancia de ambas métricas son subjetivas y podrían necesitar ser sintonizadas manualmente para diferentes categorías de objetivos.

MAMMOTH (Ortiz, Strauss, and Olmea 2002) y Dali Z-Score (Holm and Sander 1995) también han sido explotados para evaluar la precisión de predicciones de estructuras nativas. Sin embargo, estas métricas descuidan la exactitud de la alineación del modelado de la estructura y tienen el inconveniente de que la similitud de las proteínas relacionadas depende de la longitud de estas.

En 2004 se desarrolló la métrica Template Modeling Score (TM-Score) (Zhang and Skolnick 2004) para abordar las deficiencias de las anteriores métricas. TM-Score cuenta todos los pares de residuos usando el peso Levitt-Gerstein (Levitt and Gerstein 1998) y no necesita puntos de corte discretos. Dado que la distancia corta en la matriz de Levitt-Gerstein es más fuerte que la larga distancia, TM-Score es más sensitivo a una topología global que a variaciones locales. Además, debido a que adopta una escala dependiente del tamaño de la proteína para normalizar las distancias de los residuos, la magnitud de TM-Score para los pares de proteínas aleatorios es independiente del tamaño de la proteína (Zhang and Skolnick 2004).

TM-Score mide la similitud entre dos proteínas o péptidos. En esta medición comúnmente se utiliza la estructura nativa para comparar con la estructura propuesta, los valores de TM-Score son un indicador de que tan buena es la solución. El rango de TM-Score esta en [0,1] en donde el 1 indica un alineación perfecta entre dos estructuras. Siguiendo estadísticas estrictas de estructuras en la PDB, las puntuaciones por debajo de 0.17 corresponden a proteínas aleatoriamente no relacionadas entre sí, mientras que con una puntuación superior a 0.5 se supone generalmente el mismo pliegue en SCOP/CATH (Zhang and Skolnick 2004).

5.2.1 Definición de TM-Score

El TM-Score evalúa la similitud topológica de dos estructuras de proteínas y se define como:

$$TM - score = \frac{1}{L}\left[\sum_{i=1}^{L_{ali}} \frac{1}{1+d_i^2/d_0^2}\right]_{mas} \quad (17)$$

Donde:

- L es la longitud de la proteína objetivo
- L_{ali} es el número de residuos equivalentes en dos proteínas.

- d_i es la distancia del i-ésimo par de residuos equivalentes entre las dos estructuras el cual depende de la matriz de superposición; "max" determina el procedimiento para identificar la matriz de superposición optima que maximiza la suma en la Ecuación 17. La escala $d_0 = \sqrt[3]{L - 15} - 1.8$ normaliza la métrica TM-Score para que la magnitud del promedio de TM-Score para pares aleatorios de la proteína sea dependiente del tamaño de la misma. TM-score se mantiene en el rango de (0,1] cuando el valor es alto indica una fuerte similitud

5.2.2 Estadística de TM-Score

La distribución de valores extremos (EVD, por sus siglas en inglés) se utiliza para moldear el valor más pequeño o más grande de entre un gran conjunto de valores aleatorios independientes y distribuidos de manera idéntica. Se ha demostrado que la comparación de puntajes tanto de la secuencia como de la estructura se ajustan a la EVD. La función general de la EVD se define:

$$y = f(x|\mu,\sigma) = \sigma^{-1} exp\left(\frac{\mu - x}{\sigma}\right) exp\left(-exp\left(\frac{\mu - x}{\sigma}\right)\right) \quad (18)$$

Donde μ es el parámetro de localización y σ es el parámetro de escala.

Zhang realizó un cálculo de la distribución de los valores TM-Score de 71583085 pares aleatorios de proteínas que fueron recolectadas de 6684 proteínas no homólogas de dominio simple de la librería PDB. Esta distribución coincide con la ecuación 18 con el mejor parámetro de ajuste μ=0.1512 y σ =0.0242 estimado por el método de Máxima Verisimilitud, el cual fue implementado por el módulo *Evfit* en MATLAB7. Con una tolerancia de error de 1.0e-6. Las proteínas fueron dividas en cuatro grupos de acuerdo al tamaño de la proteína: [80, 100], [101, 120], [121, 160], [161, 200]; en cada caso siguen la misma EVD.

Los datos demostraron la robustez de la EVD para la distribución de la métrica TM-Score usando pares formados al azar de proteínas con alineación estructural sin espacios o huecos. También, los datos obtenidos confirmaron una conclusión previa, en el sentido de que tanto la magnitud como la distribución de TM-Score para pares de proteínas formados al azar son independientes del tamaño de la proteína (Zhang and Skolnick 2004).

Es también interesante conocer la probabilidad que se tiene de obtener una puntuación de TM-Score igual o mayor a un cierto valor (*x*) entre pares aleatorios de proteínas, por ejemplo el *P-value* de un TM-Score. Este *P-value* se calcula obteniendo la integral definida entre *x* y *1* de la ecuación 18:

$$P - value(x) = \int_x^1 f(x|\mu,\sigma)dx = 1 - exp\left[-exp\left(\frac{\mu - x}{\sigma}\right)\right] \quad (19)$$

En la figura 5.1 se muestra la curva del *P-value* versus TM-Score con los valores de μ y σ antes mencionados. La probabilidad de encontrar un TM-Score ≤ 0.17 en pares estructurales aleatorios es cercano a 1. *P-value* se decremento rápidamente cuando el TM-Score es > 0.17. Se obtiene un valor significativamente < 1 cuando TM-Score > 0.3. Cuando TM-Score = 0.5, su valor correspondiente de *P-value* es 5.5 x 10^{-7}.

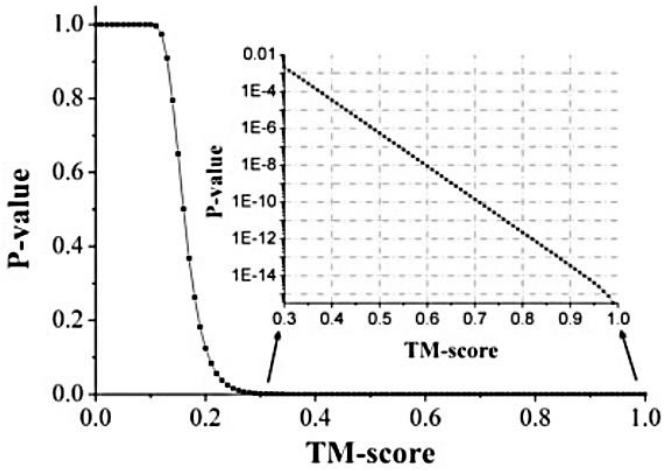


Figura 5.1 *P-value versus TM-Score*

Además, se ha demostrado que las métricas de RMSD, GDT_Score entre otras, son dependientes del tamaño de la proteína, mientras que la métrica TM-Score es totalmente independiente de la longitud de la proteína, lo cual permite expresar al valor de *P-value* como única función de TM-Score. En la figura 5.2 se muestra el promedio y la desviación del valor de TM-score con proteínas de diferentes tamaños. Los datos muestran de nuevo la independencia de los valores de TM-Score en pares de proteínas formados aleatoriamente. También presenta el número de pares de proteínas aleatorias que se necesitan para lograr o sobrepasar ciertos valores de TM-Score, estos valores se transforman a datos *P-value* mostrados en la figura 5.1.

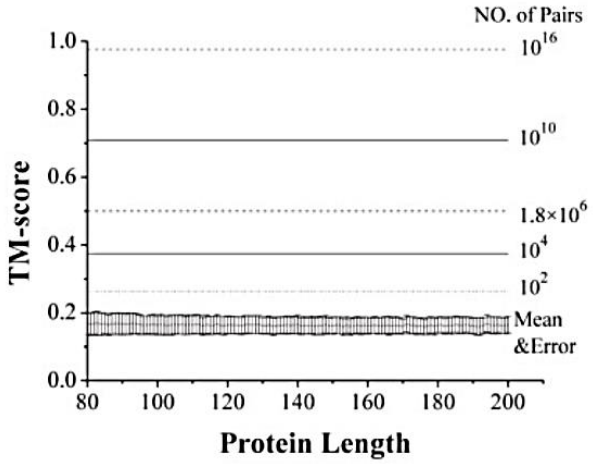


Figura 5.2 *Promedio y desviación de TM Sscore*

Como ejemplo, para valores de TM-Score=0.5, se necesitan al menos 1.8 millones de coincidencias estructurales aleatorias para que una estructura pueda alcanzar un TM-Score ≥ 0.5. Cuando un TM-Score = 0.72, el número de coincidencias estructurales se incrementa a 10 mil millones (Xu and Zhang 2010).

5.3 Experimentación

A continuación, se muestran los resultados experimentales para las metaheurísticas propuestas en el capítulo 4.

La experimentación fue realizada en el clúster de alto rendimiento Ehécatl del Instituto Tecnológico de Ciudad Madero con las siguientes características: Intel® Xeon® procesador de 2.30 GHz, Memoria de 64GB (4x16GB) ddr4-2133 y sistema operativo Linux CentOS.

El lenguaje utilizado para las metaheurísticas es Fortran 77 en el software de distribución libre SMMP. Los parámetros de energía están sometidos al uso del campo de fuerza ECEPP/2. Se fijaron los ángulos Ω a 180° y los ángulos Φ y Ψ respetan las restricciones de Ramachandran que se encuentran en el paquete SMMP. Cada experimento fue ejecutado 30 veces para cada instancia y los valores mostrados en este capítulo es el promedio de estas ejecuciones.

5.3.1 PG/MQA

Para el algoritmo PG/MQA se evaluaron las instancias de nombre Met-Enkephalin, Proinsulin y C-Peptide. En la Tabla 5.1 se describen los aminoácidos de las proteínas o instancias. La instancia Met-Enkephalin[5] es comúnmente utilizada por la comunidad científica para la evaluación de algoritmos computacionales nuevos. El mejor valor de energía conocido para esta instancia es de -10.72 kcal/mol (Zhan et al. 2006). Para la instancia C-Peptide[13] la mejor energía conocida es de -101.94 kcal/mol (Frausto-Solis et al. 2014). Por último, la instancia Proinsulin[31] el mejor valor de energía conocido es de -162.56 kcal/mol (Frausto-Solis et al. 2014).

Tabla 5.1 *Instancias de PDP*

Instancia	Aminoácidos
# **Met-Enkephalin**[5]	TYR GLY GLY PHE MET
# **C-Peptide**[13]	LYS GLU THR ALA ALA ALA LYS PHE GLU ARG GLN HIS MET
# **Proinsulin**[31]	GLU ALA GLU ASP LEU GLN VAL GLY GLN VAL GLU LEU GLY GLY GLY PRO GLY ALA GLY SER LEU GLN PRO LEU ALA LEU GLU GLY SER LEU GLN

En la Tabla 5.2 se muestran los resultados obtenidos por PG/MQA los cuales son valores de energía en kcal/mol

Tabla 5.2 *Resultados de energía de PG/MQA*

Ejecución	Met-Enkephalin	C-Peptide	Proinsulina
1	-5.3209	-60.3583	-110.944
2	-4.7716	-54.5043	-98.0230
3	-4.4657	-65.8224	-116.731
4	-6.4801	-72.9614	-94.8558
5	-4.1414	-66.5230	-94.3933
6	-4.1273	-74.1826	-92.8386
7	-8.3724	-71.2370	-104.750
8	-3.1271	-65.8349	-116.728
9	-4.8641	-59.5334	-94.4308
10	-6.8164	-54.4973	-90.8197
11	-4.1828	-71.8799	-70.9187
12	-6.0348	-67.6635	-89.6382
13	-7.3594	-67.6373	-108.463
14	-3.1300	-68.3828	-112.495
15	-4.3373	-64.2848	-96.2443
16	-5.5539	-62.4084	-99.7512
17	-3.1040	-59.3402	-103.375
18	-4.8808	-64.3937	-83.3116
19	-6.7116	-58.9684	-108.523
20	-4.7336	-62.9690	-87.2469
21	-4.6304	-66.1389	-120.303
22	-7.6291	-64.9313	-103.291
23	-5.5712	-64.6906	-91.8249
24	-4.3415	-61.4090	-108.707
25	-5.2887	-60.1844	-119.664
26	-4.1667	-44.3419	-105.522
27	-8.5506	-69.2032	-122.772
28	-6.9274	-67.1211	-127.172
29	-4.9781	-70.3984	-83.2345
30	-3.0431	-69.2039	-165.522

La Tabla 5.3 muestra los mejores y peores resultados obtenidos para las instancias. La mejor solución encontrada por PG/MQA para Met-Enkephalin[5] fue de -8.55 kcal/mol, con un tiempo de procesamiento de 1.55 minutos, la peor solución es de -3.04 kcal/mol con un tiempo de procesamiento de 0.39 minutos. Se muestra un resumen de las instancias Met-

Enkephalin[5], C-Peptide[13] y Proinsulin[31] con el resultado promedio y el mejor obtenido por PG/MQA.

Tabla 5.3 *Resumen de PG/MQA*

Instancias de Prueba				**Mejor Solución Kcal/mol**	**Solución Promedio Kcal/mol**	**Peor Solución Kcal/mol**
Proteina	#Amino acidos/ #variables	Tiempo	Valor de α			
Met-Enkephalin	5/19	0.5311	0.85	-8.5513	-5.2547	-3.0431
C-Peptide	13/68	1.3010	0.70	-74.1826	-64.3668	-60.1844
Proinsulin	31/132	17.6052	0.95	-165.5225	-104.0833	-70.9187

Para la medir la calidad de las soluciones obtenidas por PG/MQA en comparación con la estructura nativa que se encuentra en la Protein Data Bank (PDB) (H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov 2000), se utilizaron las métricas de desempeño RMSD y TM-Score. La métrica RMSD nos brinda la distancia promedio entre los átomos de la solución de PG/MQA y la estructura nativa. TM-Score además mide la similitud entre la solución y la estructura nativa.

Dada la importancia de la instancia Met-Enkephalin[5] para la determinación del desempeño de un algoritmo nuevo para PDP (Zhan, Chen, and Liu 2006; Mohamed, Hegazy, and Badr 2010; Nayeem, Vila, and Scheraga 1991; Eisenmenger and Hansmann 1997). En la tabla 5.4 se muestran los resultados obtenidos por las métricas para las mejores soluciones de PG/MQA para cada valor de α del algoritmo PG/MQA

Tabla 5.4 *RMSD y TM-Score de las soluciones de PG/MQA para la instancia Met-Enkephalin*[5]

Valor de α	**Energía de PG/MQA en Kcal/mol**	**TM-Score**	**RMSD**
0.70	-5.32	0.50109	0.65
0.75	-8.37	0.59189	0.06
0.80	-6.92	0.42097	1.03
0.85	-8.55	0.51690	0.20
0.90	-7.35	0.56351	0.33
0.95	-7.62	0.59416	0.05

Los valores significativos para la métrica RMSD son los valores que son cercanos a cero, lo cual indicaría que la solución propuesta tiene una estructura cercana a la estructura nativa. Por otro lado, TM-Score con valores > 0.5 indicarían que la solución es similar a la estructura nativa o la mejor conocida en la PDB. Nótese que para valores de α de PG/MQA cercanos a uno el valor de energía es mejor y los valores de RMSD y TM-Score corresponden con soluciones que tienen estructura bien alineada a la estructura nativa.

5.3.2 PSS/MQA

El segundo algoritmo desarrollado en este trabajo de tesis es PSS/MQA se evaluó con las mismas instancias que PG/MQA: Met-Enkephalin[5], C-Peptide[13] y Proinsulin[31]. A continuación se presentan los resultados obtenidos para estas instancias en la Tabla 5.5.

Tabla 5.5 Resultados de energía de PSS/MQA

Ejecución	Met-Enkephalin	C-Peptide	Proinsulina
1	-6.6892	-76.9449	-114.3527
2	-6.6249	-73.2178	-108.3987
3	-6.3105	-79.5841	-114.8372
4	-6.8205	-79.4848	-149.6874
5	-6.7124	-73.3667	-128.1635
6	-6.2269	-82.8816	-110.0329
7	-5.7269	-106.591	-122.3012
8	-6.8748	-74.8372	-144.2119
9	-6.8721	-74.3527	-130.6083
10	-6.2103	-78.3987	-124.9506
11	-6.0524	-105.239	-115.0934
12	-8.7526	-70.3456	-109.3405
13	-6.8853	-74.3944	-120.3434
14	-7.6291	-77.9448	-110.9234
15	-7.0158	-69.5084	-118.0261
16	-6.3102	-79.1809	-116.9834
17	-6.3104	-76.2047	-144.8563
18	-6.6910	-75.4982	-122.3958
19	-6.6916	-79.5338	-122.8613
20	-6.9483	-84.4971	-124.7528
21	-5.7437	-71.8864	-116.7202
22	-5.9762	-77.6648	-124.4331
23	-6.1635	-87.6334	-110.8141
24	-5.4519	-88.3810	-110.9155
25	-5.5171	-94.2810	-119.8231
26	-5.8011	-72.4080	-108.4674
27	-5.8004	-99.3406	-112.1346
28	-5.8018	-104.393	-126.2421
29	-5.8024	-78.9345	-109.7233
30	-5.4519	-76.5916	-113.3742

La Tabla 5.6 muestra los mejores y peores resultados obtenidos para las instancias Met-Enkephalin[5], C-Peptide[13] y Proinsulin[31]. La mejor solución encontrada por PSS/MQA

para Met-Enkephalin[5] fue de -8.75 kcal/mol, con un tiempo de procesamiento de 39.26 minutos, la peor solución es de -5.45 kcal/mol con un tiempo de procesamiento de 1.19 minutos.

Tabla 5.6 *Resumen de PSS/MQA*

	Instancias de Prueba			Mejor Solucion Kcal/mol	Solucion Promedio Kcal/mol	Peor Solucion Kcal/mol
Proteina	#Amino acidos/ #variables	Tiempo	Valor de α			
Met-Enkephalin	5/19	1.3010	0.95	-8.7526	-5.9748	-5.4519
C-Peptide	13/68	17.6052	0.95	-106.591	-81.4507	-69.5084
Proinsulin	31/132	0.5311	0.90	-149.6874	-120.1922	-108.398

El estudio realizado en la proteína Met-Enkephalin[5] con el algoritmo PSS/MQA se presenta en la Tabla 5.7. En dicho estudio se aplicaron las métricas RMSD y TM Score. Es importante señalar que PSS/MQA presentó un mejor valor de energía cuando el valor de α del algoritmo se encontraba en 0.95 y con respecto a la métrica TM Score el valor de similitud de la solución encontrada fue aún más exitoso que el encontrado por PG/MQA.

Tabla 5.7 *RMSD y TM-Score de las soluciones de PSS/MQA para la instancia Met-Enkephalin[5]*

Valor de α	Mejor de energía de PSS/MQA en Kcal/mol	TM-Score	RMSD
0.70	-6.82	0.37	0.38
0.75	-6.22	0.58	0.09
0.80	-5.72	0.49	0.23
0.85	-6.87	0.50	1.26
0.90	-5.80	0.49	0.23
0.95	-8.75	0.74	1.60

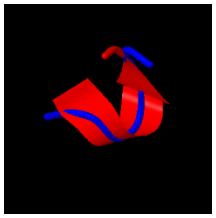
5.3.3 Comparación de PG/MQA con algoritmos del estado del arte

En este experimento se comparó el resultado de desempeño del algoritmo PG/MQA con los algoritmos MPSABBE (Frausto-Solis et al. 2016) y CMQA (Frausto-Solis et al. 2014)(). Los indicadores de desempeño considerados son TM-Score, RMSD y la energía promedio. La tabla 5.8 muestra en la primera columna las instancias evaluadas, mientras que en las columnas 2, 3 y 4 se incluyen los valores para los indicadores mencionados para el algoritmo PG/MQA, las columnas 5 6 y 7 los resultados de estos indicadores para el algoritmo MPSABBE y en las columnas 8, 9 y 10 los resultados correspondientes para el algoritmo CMQA.

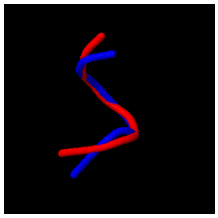
Tabla 5.8 Comparación de PG/MQA con algoritmos del estado del arte

Proteína	α	PG/MQA			MPSABBE			CMQA		
		TMScore	RMSD	Energía	TMScore	RMSD	Energía	TMScore	RMSD	Energía
Met Enkephalin	0.75	0.59	0.06	-8.37	---	0.45	-3.08	---	---	-3.28
	0.95	0.59	0.06	-7.62	---	0.36	-5.06	---	---	-5.07
Proinsulina	0.75	0.21	3.38	-122.89	---	3.13	-94.25	---	---	-93.99
	0.95	0.22	2.56	-165.52	---	3.12	-122.43	---	---	-120.76

Para los valores de la proteína Met-Enkephalin[5] se lograron valores de RSMD cercanos a 0 lo cual nos indica que está alineado con la EN y el valor de TM-Score es consistente con esa apreciación ya que alcanzó valores mayores a 0.5. Al estudiar la Met-Enkephalin[5] PG/MQA obtuvo menores valores de energía que los obtenidos por MPSABBE y CMQA; estas investigaciones no reportan TM-Score o información acerca de la alineación de los péptidos estudiados, solo en el caso de MPSABBE se reporta el RMSD pero PG/MQA obtiene valores menores. En la figura 5.3 se muestra la comparación entre la EN de la Met-Enkephalin[5] que se encuentra en la PDB y las estructuras generadas por PG/MQA para la misma proteína. Las estructuras se encuentran superpuestas, la EN es de color azul y la estructura de PG/MQA está indicada en color rojo.



α = 0.95
Energy = -6.3102
TM-Score = 0.64674



α = 0.70
Energy = -5.32
TM-Score= 0.50109

***Figura 5.3** Estructuras encontradas por PG/MQA para la Met-Enkephalin*

5.3.4 Comparación del desempeño de PG/MQA y PSS/MQA

En este experimento se compararon las soluciones resultantes de las metaheurísticas implementadas para las instancias estudiadas. En la Tabla 5.8 se muestra los promedios de las medidas TM-Score y RMSD para los algoritmos PG/MQA y PSS/MQA. Como se puede observar PSS/MQA muestra mejores resultados de energía que los obtenidos por PG/MQA. También los valores de desempeño son mejores para PSS/MQA pero el tiempo de ejecución es elevado con respecto al tiempo de PG/MQA.

Tabla 5.9 *Comparación de PG/MQA y PSS/MQA*

Instancia	TM-Score Promedio de PG/MQA	RMSD Promedio de PG/MQA	Tiempo Minutos	TM-Score Promedio de PSS/MQA	RMSD Promedio de PSS/MQA	Tiempo Minutos
Met-Enkephalin	0.53	0.35	0.53	0.53	0.63	39.26
C-Peptide	0.15	1.90	1.30	0.22	1.60	69.06
Proinsulin	0.21	3.25	17.60	0.27	3.20	72.60

El uso de la programación paralela plasmado en PSS/MQA ayudo a aumentar el espacio de búsqueda de las soluciones. Esto se puede notar en la calidad de las soluciones mostradas anteriormente, soluciones más consistentes y de menor valor de energía.

En el estudio realizado en específico para la instancia Met-Enkphalin[5] se puede apreciar que PSS/MQA genera soluciones de mejor calidad al tener valores de TM-Score >0.5. Para obtener este valor, de acuerdo a la estadística de esta métrica TM-Score, se necesitaría que al menos 1.8 millones de coincidencias estructurales aleatorias se alinearan para que pudiera alcanzar ese puntaje. Como se puede observar en la Tabla 11, PSS/MQA logra el menor valor de energía de ambos algoritmos con un TM-Score de 0.74. La estadística de TM-Score, nos dice que para valores = 0.72, el número de coincidencias necesarias para ese doblado incrementa a 10 billones.

Capitulo 6. Conclusiones y Trabajos futuros

6.1 Conclusiones

Se desarrollaron dos metaheurísticas paralelas hibridas poblacionales PSS/MQA la cual está basada en Scatter Search y MultiQuenching y PG/MQA basado en algoritmo Genético y Multiquenching. Los resultados experimentales muestran un desempeño competitivo de ambas metaheurísticas híbridas, en relación con los trabajos del estado del arte comparados.

El algoritmo PSS/MQA al explorar el espacio de búsqueda con programación paralela nos brindó la oportunidad de encontrar soluciones con una energía menor a las publicadas en el estado del arte, como en el caso de la instancia C-peptide[13]. Con el algoritmo PG/MQA se encontró una configuración con un valor menor de energía para la instancia Proinsulin[31]. Ambos algoritmos logran paralelizar e hibridar el algoritmo MultiQuenching y estrategias evolutivas con éxito, hasta nuestro conocimiento no se habían reportado híbridos eficientes con estas estrategias en la literatura. Siendo PSS/MQA y PG/MQA los primeros algoritmos híbridos paralelos MultiQuenching.

Además, se incorporó el uso de la métrica de desempeño TM-Score utilizada en CASP para medir el desempeño de los algoritmos ya que esta medida junto con RMSD nos brinda más información acerca de la conformación estructural de nuestras soluciones.

6.2 Publicaciones

Como resultado de esta investigación se ha publicado en foros científicos:

Año 2017:

> "Evolutionary Multi-Quenching Annealing for Protein Folding Problem", IJCOPY; 2017. Articulo aceptado para su publicación en International Journal of Combinatorial Optimization Problems and Informatics el cual pertenece a The Classification System of Mexican Journals of Scientific Research and Technology of the National Council of Science and Technology (CONACYT, *http://www.revistascytconacyt.mx*) and the Emerging Sources Citation Index of Clarivate Analytics (Thomson Reuters).

Año 2016:

1) “Algoritmo Multiquenching Poblacional Paralelo Para Doblado De Proteínas”. Ponencia y Memoria del V Congreso Nacional de la Sociedad Mexicana de Investigación de Operaciones. Ciudad Madero, Tamaulipas, 2016, Octubre, 26.

2) “Algoritmo Hibrido Multi-Quenching Poblacional Para el Problema de Doblado De Proteínas”, Ponencia en el 4° Encuentro de Jóvenes Investigadores de Tamaulipas organizado por COTACyT en Cd. Victoria, Tamaulipas, 2016, Noviembre, 1.

6.3 Trabajos Futuros

Algunos de los trabajos futuros con los que se puede continuar esta investigación son los siguientes:

- Algunas proteínas pueden seguir el proceso de doblado hasta alcanzar una cuarta estructura. Dicha estructura involucra el ensamblaje de estructuras nativas. Este es un problema que se puede estudiar a futuro.

- Diseñar y crear algoritmos heurísticos que aprovechen las hélices α.

- Diseñar métodos de computación inteligente hibridas e investigar otros métodos de paralelización.

- Investigar otros ambientes de programación y simulación de proteínas además de SMMP.

Capitulo 7. Referencias bibliográficas

Agostini, Flavia P., Diogo De O Soares-Pinto, Marcelo A. Moret, Carla Osthoff, and Pedro G. Pascutti. 2006. "Generalized Simulated Annealing Applied to Protein Folding Studies." *Journal of Computational Chemistry* 27 (11): 1142–55. doi:10.1002/jcc.20428.

Anfinsen, C B. 1973. "Principles That Govern the Folding of Protein Chains." *Science (New York, N.Y.)* 181 (4096): 223–30. http://www.ncbi.nlm.nih.gov/pubmed/4124164.

Barney, Blaise. 2017. "Message Passing Interface (MPI)." *Lawrence Livermore National Laboratory*. https://computing.llnl.gov/tutorials/mpi/#Routine_Arguments.

Berendsen, H. J C, D. van der Spoel, and R. van Drunen. 1995. "GROMACS: A Message-Passing Parallel Molecular Dynamics Implementation." *Computer Physics Communications* 91 (1–3): 43–56. doi:10.1016/0010-4655(95)00042-E.

Bioinformatics, Bmc, and Yang Zhang. 2017. "I-TASSER Server for Protein 3D Structure Prediction." Accessed March 17. doi:10.1186/1471-2105-9-40.

Blaise Barney, Lawrence Livermore National Laboratory. 2017. "Introduction to Parallel Computing." https://computing.llnl.gov/tutorials/parallel_comp/#HybridMemory.

Brooks, Bernard R., Robert E. Bruccoleri, Barry D. Olafson, David J. States, S. Swaminathan, and Martin Karplus. 1983. "CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations." *Journal of Computational Chemistry* 4 (2): 187–217. doi:10.1002/jcc.540040211.

Crescenzi, Pierluigi, Deborah Goldman, Christos Papadimitriou, Antonio Piccolboni, and Mihalis Yannakakis. 1998. "On the Complexity of Protein Folding." *Journal of Computational Biology* 5 (3): 423–65. doi:10.1089/cmb.1998.5.423.

Díaz Velarde, Adenso., and José Luis. González Fernández. 1996. *Optimización Heurística Y Redes Neuronales*. Paraninfo.

Dill, Ken A, and Justin L. MacCallum. 2012. "The Protein-Folding Problem, 50 Years On." *Science* 338 (6110): 1042–46. doi:10.1126/science.1219021.

Dill, Ken A, S Banu Ozkan, M Scott Shell, and Thomas R Weikl. 2008. "The Protein Folding Problem." *Annual Review of Biophysics* 37 (1): 289–316. doi:10.1146/annurev.biophys.37.092707.153558.

Eisenmenger, Frank, and Ulrich H. E. Hansmann. 1997. "Variation of the Energy

Landscape of a Small Peptide under a Change from the ECEPP/2 Force Field to ECEPP/3."

Eisenmenger, Frank, Ulrich H E Hansmann, Shura Hayryan, and Chin-Kun Hu. 2001. "[SMMP] A Modern Package for Simulation of Proteins LONG WRITE - UP." *Computer Physics Communications* 138: 192–212. www.elsevier.com/locate/cpc.

Eisenmenger, Frank, Ulrich H E Hansmann, Shura Hayryan, and Chin-Kun Hu. 2006. "An Enhanced Version of SMMP—open-Source Software Package for Simulation of Proteins." *Computer Physics Communications* 174 (174): 422–29. doi:10.1016/j.cpc.2005.10.013.

Fengbin, Peng, Zhang Huilin, Wei Yanjie, Feng Shengzhong, and Yin Zhixiang. 2013. "Protein Folding Study Based on Parallel Group Annealing Algorithms" 4 (5): 26–34.

Frausto-Solis, Juan, Ernesto Liñáan-García, Mishael Sanchez-Perez, and Juan Paulo Sanchez-Hernandez. 2014. "Chaotic Multiquenching Annealing Applied to the Protein Folding Problem." *The Scientific World Journal* 2014. doi:10.1155/2014/364352.

Frausto-Solis, Juan, Ernesto Liñán-García, Juan Paulo Sánchez-Hernández, J Javier González-Barbosa, Carlos González-Flores, and Guadalupe Castilla-Valdez. 2016. "Multiphase Simulated Annealing Based on Boltzmann and Bose-Einstein Distribution Applied to Protein Folding Problem." *Advances in Bioinformatics* 2016. Hindawi Publishing Corporation: 7357123. doi:10.1155/2016/7357123.

Frausto-Solis, Juan, and EF Román. 2007. "Analytically Tuned Simulated Annealing Applied to the Protein Folding Problem." *... Science–ICCS 2007*, 370–77. http://link.springer.com/chapter/10.1007/978-3-540-72586-2_53.

Frausto-Solis, Juan, Juan Paulo Sánchez-Hernández, Mishael Sánchez-Pérez, and Ernesto Liñán Liñan García. 2015. "Golden Ratio Simulated Annealing for Protein Folding Problem." *International Journal of Computational Methods* 12 (6): 1550037. doi:10.1142/S0219876215500371.

Frausto-Solis, Juan, Xavier Soberon-Mainero, and Ernesto Liñan-Garcia. 2009. "MultiQuenching Annealing Algorithm for Protein Folding Problem." *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5845 LNAI: 578–89. doi:10.1007/978-3-642-05258-3_51.

Ginalski, K., A. Elofsson, D. Fischer, and L. Rychlewski. 2003. "3D-Jury: A Simple Approach to Improve Protein Structure Predictions." *Bioinformatics* 19 (8). Oxford University Press: 1015–18. doi:10.1093/bioinformatics/btg124.

Glover Manuel Laguna, Fred, and Rafael Martí. 2000. "Fundamentals of Scatter Search and Path Relinking" 29 (3): 653–84.

Goldberg, D. E. 1989. "Genetic Algorithms in Search, Optimization, and Machine

Learning." *Reading, MA: Addison-Wesley.*

H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne. 2000. "RCSB Protein Data Bank - RCSB PDB." www.rcsb.org.

Hao, Ge-Fei, Wei-Fang Xu, Sheng-Gang Yang, and Guang-Fu Yang. 2015. "Multiple Simulated Annealing-Molecular Dynamics (MSA-MD) for Conformational Space Search of Peptide and Miniprotein." *Scientific Reports* 5 (October): 15568. doi:10.1038/srep15568.

Hermanns, Miguel. 2002. "Parallel Programming in Fortran 95 Using OpenMP." *School of Aeronautical Engineering*, no. April: 71. http://people.sc.fsu.edu/~jburkardt/pdf/hermanns.pdf.

Hiroyasu, Tomoyuki, Mitsunori Miki, Shinya Ogura, Keiko Aoi, Takeshi Yoshida, Yuko Okamoto, and Jack Dongarra. 2002. "Energy Minimization of Protein Tertiary Structure by Parallel Simulated Annealing Using Genetic Crossover." In *2002 Genetic and Evolutionary Computation Conference (GECCO 2002) Workshop Program*, 49–51.

Holm, Liisa, and Chris Sander. 1995. "Dali: A Network Tool for Protein Structure Comparison." *Trends in Biochemical Sciences* 20 (11): 478–80. doi:10.1016/S0968-0004(00)89105-7.

Jones, David. 1998. "THREADER: Protein Sequence Threading by Double Dynamic Programming." In , 285–311. doi:10.1016/S0167-7306(08)60470-6.

Källberg, Morten, Haipeng Wang, Sheng Wang, Jian Peng, Zhiyong Wang, Hui Lu, and Jinbo Xu. 2012. "Template-Based Protein Structure Modeling Using the RaptorX Web Server." *Nature Protocols* 7 (8): 1511–22. doi:10.1038/nprot.2012.085.

Karplus, K. 2009. "SAM-T08, HMM-Based Protein Structure Prediction." *Nucleic Acids Research* 37 (Web Server). Oxford University Press: W492–97. doi:10.1093/nar/gkp403.

Kendrew, J. C., G Bodo, H M Dintzis, R G Parrish, H Wyckoff, and D C Phillips. 1958. "A Three-Dimensional Model of the Myoglobin Molecule Obtained by X-Ray Analysis." *Nature* 181: 662.

Khoury, George A., James Smadbeck, Chris A. Kieslich, and Christodoulos A. Floudas. 2014. "Protein Folding and de Novo Protein Design for Biotechnological Applications." *Trends in Biotechnology* 32 (2). Elsevier Ltd: 99–109. doi:10.1016/j.tibtech.2013.10.008.

Kim, D. E., D. Chivian, and D. Baker. 2004. "Protein Structure Prediction and Analysis Using the Robetta Server." *Nucleic Acids Research* 32 (Web Server): W526–31. doi:10.1093/nar/gkh468.

Kirkpatrick, S, C D Gelatt, and M P Vecchi. 1983. “Optimization by Simulated Annealing.” *Science, New Series* 220 (4598): 671–80. http://links.jstor.org/sici?sici=0036-8075%2819830513%293%3A220%3A4598%3C671%3AOBSA%3E2.0.CO%3B2-8.

Levinthal, C. 1968. “Are There Pathways for Protein Folding ?” *Extrait Du Journal de Chimie Physique* 65 (1).

Levitt, Michael, and Mark Gerstein. 1998. “A Unified Statistical Framework for Sequence Comparison and Structure Comparison (Sequence Analysisstructure Analysisfold Familydatabase Statisticsprotein Evolution).” *Computational Biomolecular Science* 95: 5913–20. http://www.pnas.org/content/95/11/5913.full.pdf.

Lundström, Jesper, Leszek Rychlewski, Janusz Bujnicki, and Arne Elofsson. 2008. “Pcons: A Neural-Network-Based Consensus Predictor That Improves Fold Recognition.” *Protein Science* 10 (11): 2354–62. doi:10.1110/ps.08501.

Martí, Rafael, and Manuel Laguna. 2003. “Scatter Search : Diseño Básico Y Estrategias Avanzadas.” *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial* 7 (19): 123–30.

Maulik, Ujjwal, and Sanghamitra Bandyopadhyay. 2000. “Genetic Algorithm-Based Clustering Technique.” *Pattern Recognition* 33: 1455–65. doi:10.1016/S0031-3203(99)00137-5.

Meinke, Jan H., Sandipan Mohanty, Frank Eisenmenger, and Ulrich H.E. Hansmann. 2008. “SMMP v. 3.0—Simulating Proteins and Protein Interactions in Python and Fortran.” *Computer Physics Communications* 178 (6): 459–70. doi:10.1016/j.cpc.2007.11.004.

Mohamed, Adel Omar, Abdelfatah A Hegazy, and Amr Badr. 2010. “Solving Protein Folding Problem Using Hybrid Genetic Clonal Selection Algorithm.” *IJCSNS International Journal of Computer Science and Network Security* 10 (12).

Momany, F. A., R. F. McGuire, A. W. Burgess, and Harold A. Scheraga. 1975. “Energy Parameters in Polypeptides. VII. Geometric Parameters, Partial Atomic Charges, Nonbonded Interactions, Hydrogen Bond Interactions, and Intrinsic Torsional Potentials for the Naturally Occurring Amino Acids.” *The Journal of Physical Chemistry* 79 (22): 2361–81. doi:10.1021/j100589a006.

Morales, Luis B., Ramón Garduño-Juárez, and David Romero. 1991. “Applications of Simulated Annealing to the Multiple-Minima Problem in Small Peptides.” *Journal of Biomolecular Structure and Dynamics* 8 (4). Taylor & Francis Group : 721–35. doi:10.1080/07391102.1991.10507841.

Moult, John, Krzysztof Fidelis, Andriy Kryshtafovych, Torsten Schwede, and Anna Tramontano. 2016. “Critical Assessment of Methods of Protein Structure Prediction: Progress and New Directions in Round XI.” *Proteins: Structure, Function, and Bioinformatics* 84 (S1): 4–14. doi:10.1002/prot.25064.

Nayeem, Akbar, Jorge Vila, and Harold A. Scheraga. 1991. “A Comparative Study of the Simulated-Annealing and Monte Carlo-with-Minimization Approaches to the Minimum-Energy Structures of Polypeptides: [Met]-Enkephalin.” *Journal of Computational Chemistry*. John Wiley & Sons, Inc. doi:10.1002/jcc.540120509.

Nemethy, G, M S Pottle, and H a Scheraga. 1983. “Energy Parameters in Polypeptides. 9. Updating of Geometrical Parameters, Nonbonded Interactions and Hydrogen Bond Interactions for the Naturally Occuring Amino Acids.” *J. Phys. Chem.* 87 (11): 1883–87. doi:10.1021/j100234a011.

OKAMOTO, YUKO. 1999. “Tackling The Multiple-Minima Problem In Protein Folding By Monte Carlo Simulated Annealing And Generalized-Ensemble Algorithms.” *International Journal of Modern Physics C* 10 (8). World Scientific Publishing Company: 1571–82. doi:10.1142/S0129183199001352.

Olivares-Quiroz, Luis, Leopoldo García, and -Colín Scherer. 2004. “Plegamiento de Las Proteínas: Un Problema Interdisciplinario.” *Rev. Soc. Quím. Méx* 48: 95–105.

Ordway, G. A., and D. J Garry. 2004. “Myoglobin: An Essential Hemoprotein in Striated Muscle.” *Journal of Experimental Biology* 207 (20): 3441–46. doi:10.1242/jeb.01172.

Ortiz, Angel R, Charlie E M Strauss, and Osvaldo Olmea. 2002. “MAMMOTH (Matching Molecular Models Obtained from Theory): An Automated Method for Model Comparison.” *Protein Science : A Publication of the Protein Society* 11 (11). Wiley-Blackwell: 2606–21. doi:10.1110/ps.0215902.

Osguthorpe, D. 2000. “Ab Initio Protein Folding.” *Current Opinion in Structural Biology* 10 (2): 146–52. doi:10.1016/S0959-440X(00)00067-1.

Pauling, L., R. B. Corey, and H. R. Branson. 1951. “The Structure of Proteins: Two Hydrogen-Bonded Helical Configurations of the Polypeptide Chain.” *Proceedings of the National Academy of Sciences* 37 (4). National Acad Sciences: 205–11. doi:10.1073/pnas.37.4.205.

Pearlman, David A., David A. Case, James W. Caldwell, Wilson S. Ross, Thomas E. Cheatham, Steve DeBolt, David Ferguson, George Seibel, and Peter Kollman. 1995. “AMBER, a Package of Computer Programs for Applying Molecular Mechanics, Normal Mode Analysis, Molecular Dynamics and Free Energy Calculations to Simulate the Structural and Energetic Properties of Molecules.” *Computer Physics Communications* 91 (1–3). Elsevier: 1–41. doi:10.1016/0010-4655(95)00041-D.

Ponder, Jay W., and David A. Case. 2003. “Force Fields for Protein Simulations.” *Advances in Protein Chemistry*. doi:10.1016/S0065-3233(03)66002-X.

“Protein Structure Prediction Center.” 2017. Accessed March 18. http://www.predictioncenter.org/index.cgi.

Ramachandran, G N, C Ramakrishnan, and V Sasisekharan. 1963. “Stereochemistry of

Polypeptide Chain Configurations." *Journal of Molecular Biology* 7 (1): 95–99. doi:10.1016/S0022-2836(63)80023-6.

Román Rangel, Edgar Francisco. 2006. "Sintonización Del Esquema de Enfriamiento Del Algoritmo Recocido Simulado Para El Problema de Doblado de Proteínas Maestro En Ciencias de La Computación Asesores: Cuernavaca, Morelos. Noviembre 2006." *ITESM-Campus Cuernavaca*.

Sakae, Yoshitake, Tomoyuki Hiroyasu, Mitsunori Miki, Katsuya Ishii, and Yuko Okamoto. 2015. "Combination of Genetic Crossover and Replica-Exchange Method for Conformational Search of Protein Systems."

Siew, N, A Elofsson, L Rychlewski, and D Fischer. 2000. "MaxSub: An Automated Measure for the Assessment of Protein Structure Prediction Quality." *Bioinformatics (Oxford, England)* 16 (9): 776–85. http://www.ncbi.nlm.nih.gov/pubmed/11108700.

Silver, Edward A., R. Victor, V. Vidal, and Dominique de Werra. 1980. "A Tutorial on Heuristic Methods." *European Journal of Operational Research* 5 (3). North-Holland: 153–62. doi:10.1016/0377-2217(80)90084-3.

Singh Bhalla, Jasdeep, and Anmol Aggarwal. 2013. "Prediction of Protein Structure Using Parallel Genetic Algorithm." *International Journal of Computer Applications* 81: 7–11.

Soding, J., A. Biegert, and A. N. Lupas. 2005. "The HHpred Interactive Server for Protein Homology Detection and Structure Prediction." *Nucleic Acids Research* 33 (Web Server): W244–48. doi:10.1093/nar/gki408.

University of Tennessee. 2015. *MPI: A Message-Passing Interface Standard*.

Wang, Z., J. Eickholt, and J. Cheng. 2010. "MULTICOM: A Multi-Level Combination Approach to Protein Structure Prediction and Its Assessments in CASP8." *Bioinformatics* 26 (7). Oxford University Press: 882–88. doi:10.1093/bioinformatics/btq058.

Watson, J. D., and F. H. C. Crick. 1953. "THE STRUCTURE OF DNA." *Cold Spring Harbor Symposia on Quantitative Biology* 18 (0): 123–31. doi:10.1101/SQB.1953.018.01.020.

Xu, Jinrui, and Yang Zhang. 2010. "How Significant Is a Protein Structure Similarity with TM-Score = 0.5?" *Bioinformatics (Oxford, England)* 26 (7). Oxford University Press: 889–95. doi:10.1093/bioinformatics/btq066.

Zemla, A, C Venclovas, J Moult, and K Fidelis. 1999. "Processing and Analysis of CASP3 Protein Structure Predictions." *Proteins* Suppl 3: 22–29. http://www.ncbi.nlm.nih.gov/pubmed/10526349.

Zhan, Lixin, Jeff Z Y Chen, and Wing-Ki Liu. 2006. "Conformational Study of Met-

Enkephalin Based on the ECEPP Force Fields." *Biophysical Journal* 91 (7). The Biophysical Society: 2399–2404. doi:10.1529/biophysj.106.083899.

Zhan, Lixin, Jeff Z Y Chen, Wing-Ki Liu, J. Hughes, T.W. Smith, H.W. Kosterlitz, L.A. Fothergill, et al. 2006. "Conformational Study of Met-Enkephalin Based on the ECEPP Force Fields." *Biophysical Journal* 91 (7). Elsevier: 2399–2404. doi:10.1529/biophysj.106.083899.

Zhang, Yang, and Jeffrey Skolnick. 2004. "Scoring Function for Automated Assessment of Protein Structure Template Quality." *Proteins: Structure, Function and Genetics* 57 (4): 702–10. doi:10.1002/prot.20264.

Zhou, Hongyi, Shashi B. Pandit, Seung Yup Lee, Jose Borreguero, Huiling Chen, Liliana Wroblewska, and Jeffrey Skolnick. 2007. "Analysis of TASSER-Based CASP7 Protein Structure Prediction Results." *Proteins: Structure, Function, and Bioinformatics* 69 (S8): 90–97. doi:10.1002/prot.21649.

Zhou, Hongyi, Jeffrey Skolnick, J. Skolnick, V.S. Pande, M.B. Swindells, J.M. Thornton, J.J. Ward, K.M.S. Misura, and D. Baker. 2007. "Ab Initio Protein Structure Prediction Using Chunk-TASSER." *Biophysical Journal* 93 (5). The Royal Society: 1510–18. doi:10.1529/biophysj.107.109959.

Zomaya, A. Y. 2004. *Parallel Computing for Bioinformatics and Computational Biology*. *Chemistry & …*.