//

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

UN ALGORITMO GENETICO HIBRIDO PARA RESOLVER EL PROBLEMA DE PLEGADO DE PROTEINAS

T E S I S

QUE PARA OBTENER EL TITULO DE :

LICENCIADO EN CIENCIAS DE LA COMPUTACION

P R E S E N T A :

LUIS GERMAN PEREZ HERNANDEZ

DIRECTORES: DR. LUIS B. MORALES MENDOZA
DR. RAMON GARDUÑO JUAREZ

2002

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

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

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

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

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

M. EN C. ELENA DE OTEYZA DE OTEYZA
Jefa de la División de Estudios Profesionales de la
Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo escrito:

"[illegible] diseño para resolver el problema de plegado de proteínas"

realizado por [illegible]

con número de cuenta [illegible], quién cubrió los créditos de la carrera de [illegible]

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis [illegible]
Propietario

Propietario [illegible]

Propietario [illegible]

Suplente [illegible]

Suplente [illegible]

Consejo Departamental [illegible]

FACULTAD DE CIENCIAS
CONSEJO DEPARTAMENTAL
DE
MATEMÁTICAS

Un Algoritmo Genético Híbrido para resolver el problema de Plegado de Proteínas

Luis Germán Peréz Hernández

8 de marzo de 2002

Agradecimientos

Agradezco ha mis directores de tesis, los doctores Luis B. Morales Mendoza y Ramón Garduño Juárez quienes me ofrecieron la oportunidad de trabajar con ellos. Asimismo a los sinodales M.I. María de Luz Gasca Soto, M.C. José de Jesús Galaviz Casas y al Mat. Adrián Girard Islas que de forma profesional cedieron parte de su tiempo y aceptaron revisar esta tesis.

De manera muy especial les agradezco a mis padres y a mis estimados amigos Fernando, Alejandro, Shariar e Israel el apoyo que me dieron para el logro de esta meta.

El presente trabajo ha sido posible a la generosa donación de tiempo de supercómputo por la Dirección General de Cómputo Académico de la UNAM. Se agradecen los apoyos económicos otorgados para esta tesis por el Proyecto IN109999 financiado por la Dirección General de Asuntos del Personal Académico de la UNAM, así como el otorgado al inicio de este trabajo por el Proyecto CONACYT 25245-E del Dr. Ramón Garduño Juárez del Centro de Ciencias Físicas de la UNAM.

Índice General

Resumen

Este trabajo da a conocer un algoritmo para solucionar de manera eficiente el problema denominado plegamiento de proteínas, esto consiste en dar a conocer la estructura tridimensional más probable que adoptará una proteína teniendo como única información su secuencia de aminoácidos. Para tratar de solucionar el problema del plegamiento de proteínas se diseñó un procedimiento híbrido hecho de un Algoritmo Genético acoplado con un optimizador local numérico, que es una extensión de los Algoritmos Genéticos tradicionales, que mejoran por cada generación a los individuos para así tener mejores resultados en menor tiempo. Las proteínas de prueba fueron Met-encefalina y Leu-encefalina.

Capítulo 1

Introducción

El plegamiento de proteínas es uno de los principales eventos estudiados en el campo de la Bioquímica. Matemáticamente, este evento es mejor conocido como el problema de los múltiples mínimos, o como el problema del plegado de las proteínas. Este problema esta caracterizado por la compleja hipersuperficie de energía potencial que presentan las moléculas flexibles, como las proteínas, sobre la cual es casi imposible encontrar el mínimo global por medio de una búsqueda sistemática. Termodinámicamente, este problema se puede explicar en términos de los grados de libertad que posee una molécula, es decir, si una molécula no lineal esta compuesta de n átomos, esta tendrá $3n-6$ grados de libertad. Para el caso de una proteína promedio hecha de 100 residuos de aminoácido, estos grados de libertad ascienden a (100 residuos * $\approx$20 átomos por residuo)*3-6 = 5994 grados de libertad. Un sistema de ecuaciones con tal numero de variables independientes es imposible de resolver hoy en día. Si para esta proteína no hubiera restricciones en su conformación y sólo su estructura primaria fuera dada, el número de confórmeros para una proteína promedio ($\tilde{}$100 residuos) sería de aproximadamente (5 ángulos de torsión por 5 valores posibles por ángulo de torsión)^{100} = 25^{100} confórmeros. Esto significa que aun en el mejor de los casos 25^{100} conformaciones tendrían que ser evaluadas para encontrar el mínimo global. Desde el punto de vista computacional, no existe actualmente un programa polinomial que pueda resolver este problema en un tiempo razonable. Por lo tanto, tenemos que recurrir a métodos de muestreo conformacional. Las técnicas de muestreo conformacional pueden ser agrupadas en métodos determinístes y métodos estocásticos. Los primeros incluyen cualquier método por el cual la generación y evaluación de la conformación de una molécula dada están de-

terminadas por los anteriores moléculas. Los métodos estocásticos incluyen a los ampliamente usados Métodos de Montecarlo, donde la conformación se genera al tomar aleatoriamente los valores de cualquier parámetro que define a la conformación. Por lo tanto, la hipersuperficie conformacional de una molécula se somete a un muestreo aleatorio, lo que nos permite calcular la probabilidad de encontrar todos los estados de más baja energía potencial sin explorar exhaustivamente todo el espacio. Entre estos métodos que proveen la posibilidad de encontrar una solución óptima están los algoritmos evolutivos, los cuales copian las manifestaciones de la evolución natural y que se pueden programar en computadoras; esto es llamado computación evolutiva. Por lo tanto, la computación evolutiva es esencialmente heurística, es decir, tiene un componente azaroso, y nunca podrá garantizar que se obtendrá el mínimo global, pero si puede garantizar que se obtendrá una solución cercana a la solución óptima en tiempo de cálculo razonable.

Este problema tiene una gran repercusión en el entendimiento de la vida, ya que el plegamiento de las proteínas es una de las bases para entenderla. Puesto que las proteínas conforman a las células y cada uno de los diversos tipos de proteínas que contienen a éstas desarrollan una labor específica.

En una célula las moléculas de proteínas se pliegan para llevar a cabo sus funciones biológicas [24]. Este plegado se lleva a cabo en tiempos que van desde milisegundos hasta minutos para llegar a una estructura compacta biológicamente activa. La ruta que sigue una proteína para plegarse está íntimamente ligada a la secuencia de sus aminoácidos constituyentes. Para la mayoría de las proteínas globulares la información codificada en la secuencia de sus aminoácidos es suficiente para determinar la estructura secundaria de su estado nativo. En general, todas las proteínas nativas se pliegan a estructuras compactas específicas a pesar de la gran cantidad de posibles conformaciones. Como las proteínas están formadas de miles de átomos que interactúan entre sí y su estructura biológica depende en gran medida de sus interacciones con el disolvente que les rodea, actualmente no es posible el calcular su función de energía libre por primeros principios; por lo tanto, es necesario adaptar seudopotenciales obtenidos a partir de estructuras conocidas en los bancos de datos. Aún con estos potenciales mínimos, encontrar la estructura con una energía más baja es una tarea muy compleja ya que el espacio de búsqueda está determinado por el número de conformaciones posibles, el cual es de orden de 2^N, donde N es el número de variables independientes de la función del potencial conformacional. Matemáticamente, este problema pertenece a la clase de problemas NP-completos [9]. Obtener

por medio de simulaciones en computadora la conformación activa de una molécula de proteína a partir de su secuencia de aminoácidos es difícil por dos motivos: primero, la contribución de la energía libre en la estabilización de la estructura plegada es muy poco estudiada, y segundo, el espacio de sus posibles conformaciones es muy grande y complejo. Mientras que analizar el primero requiere de modelos detallados de la estructura de la proteína, el segundo motivo requiere de un modelo simplificado y de diseñar buenos métodos de búsqueda.

A diferencia de los algoritmos de búsqueda exhaustiva, como Montecarlo, la idea básica de una búsqueda heurística es que, más que tratar de explorar todos los pasos posibles de búsqueda, se trata de enfocarse a zonas que parecen acercarse a la solución deseada. Para problemas muy complejos los heurísticos se acercan lo más posible a la solución, y aunque parezca que se toman mucho tiempo y que los pasos para llegar son complicados, éstos al menos nos dan una muy buena aproximación a la solución buscada. Diseñar una técnica heurística que sea rápida y eficiente es siempre un desafío, ya que generalmente depende del tipo de la función de evaluación, el número de variables y la complejidad del problema.

Para buscar esta solución del problema del plegado de proteínas y lograr simular sus estructuras tridimensionales, se han usado diversos métodos. Pero como este problema en particular presenta una gran cantidad de elementos combinatorios, se requiere de técnicas muy elaboradas para su solución. En el pasado se han empleado varias técnicas heurísticas con buenos resultados, como lo son la búsqueda Tabú, el Recocido Simulado y las Redes Neuronales, entre otros. Sin embargo aunque se ha reducido mucho el tiempo en que se obtiene un buen resultado y se dan buenas aproximaciones, todavía se puede mejorar esto utilizando otras técnicas alternativas.

En este trabajo reportamos el uso de un procedimiento híbrido de un algoritmo genético acoplado con el optimizador local SUMLS [13] para dar otra posible solución a este problema. En donde el objetivo principal, aparte de mostrar una técnica más en solucionar este problema, es la de tratar de mejorar el rendimiento en tiempo de cómputo y la precisión en los resultados obtenidos. Para esto se realizaron pruebas con los neuropéptidos Met-encéfalina y Leu-encéfalina para los cuales se conocen sus estructuras biológicamente activas por otros trabajos, para que así se pueda medir la eficiencia y la exactitud de este método. Posteriormente se realizarán experimentos con otras proteínas menos estudiadas.

Actualmente con los métodos de ingeniería genética se pueden producir

proteínas con una secuencia dada de aminoácidos. Si se conociera como se puede plegar esta proteína, se podrían predecir sus propiedades químicas y biológicas. Simplificando la labor de interpretación de la información de la proteína. Una de sus principales aplicaciones sería en el genoma humano, ya que permitiría entender el mecanismo de enfermedades infecciosas y hereditarias, con esto se podrían diseñar drogas con propiedades específicas terapéuticas y cosechando polímeros biológicos con propiedades de materiales específicos.

Capítulo 2

Antecedentes del problema

En este capítulo se explicará más profundamente el plegado de proteínas y se planteará el problema en términos matemáticos para describir la metodología para resolverlo.

También se dará a conocer la terminología que se usará durante este trabajo. Así como algunos conceptos bioquímicos que son fundamentales para entender el funcionamiento de este sistema y los objetivos perseguidos.

2.1 Marco Teórico

En la naturaleza, las proteínas se pliegan para llevar a cabo sus funciones biológicas y éstas necesitan de milisegundos a minutos para plegarse a una estructura biológicamente compacta activa [14]. Para la mayoría de las proteínas globulares la información codificada en la secuencia de sus aminoácidos es suficiente para determinar la conformación de su estado nativo. Las proteínas naturales se pliegan a estructuras compactas específicas a pesar de la gran cantidad de sus posibles conformaciones. Como las proteínas contienen miles de átomos que interactúan entre sí, así como con otros miles de moléculas de agua, no es posible calcular su función de energía libre por principios básicos; por lo tanto, es necesario adaptar seudopotenciales obtenidos a partir de estructuras conocidas en los bancos de datos. Aún con estos potenciales mínimos, encontrar la estructura con la energía más baja es una tarea costosa ya que el espacio de búsqueda está determinado por el número de conformaciones posibles, el cual es del orden de 2^N, donde N es el número de variables independientes de la función del potencial conformacional. Este

problema pertenece a la clase de NP-Completos.

Termodinámicamente hablando, el plegado de una proteína puede visualizarse como un embudo energético [14]. Definimos un embudo de plegado como una colección de estructuras colapsadas geométricamente semejantes, donde una de éstas es termodinámicamente más estable con respecto al resto de ellas. Si una secuencia de aminoácidos tiene un embudo de plegado que lleva a una conformación estable única, se dice que ésta es plegable. Por el contrario, una secuencia no es plegable si tiene embudos múltiples que no llevan a un mínimo energético único.

Los métodos para la predicción de la estructura secundaria de proteínas han avanzado en las últimas décadas. Hoy en día, con técnicas de hilvanado se puede predecir la estructura secundaria de α-hélices y de β-plegadas. Sin embargo, las estructuras de vuelta de horquilla, siguen siendo evasivas [14]. Se pretende determinar la estructura terciaria de pequeños péptidos que contengan vueltas de horquilla, empleando el acoplamiento de un modelo detallado de la molécula, de un seudopotencial para evaluar su energía conformacional, y de un método heurístico para obtener una muestra del espacio conformacional de manera rápida.

La búsqueda conformacional de moléculas de proteínas, en una primera aproximación, puede ser vista como el problema de encontrar una estructura molecular tridimensional (3D) que corresponda al mínimo local más bajo de una función matemática apropiada que describa la energía potencial del sistema. Las simulaciones por computadora permiten obtener un conjunto de conformaciones de baja energía con significado biológico, esto es, considerando que el estado nativo termodinámico de estas moléculas puede corresponder al mínimo más bajo de energía o mínimo global. Una posible manera de tener acceso a este mínimo es por medio de una búsqueda cuidadosa del paisaje de energía conformacional que eventualmente nos acercará a un mínimo global. En principio, podría ser suficiente encontrar un conjunto crudo de estructuras alrededor del mínimo global que por medio de una optimización local puedan llegar al buscado mínimo de energía. Dado que la mayoría de las funciones de energía potencial que describen el espacio conformacional de péptidos, tienen un gran número de mínimos locales y posiblemente único mínimo global, encontrarlo es una forma certera y rápida es de interés general.

Una molécula de proteína es un polímero compuesto de una secuencia específica de 20 aminoácidos naturales, los ladrillos de construcción o monómeros, conectados a través de enlaces péptidos. En la naturaleza la

mayoría de las proteínas se pliegan espontáneamente a sus conformaciones nativas. Bajo condiciones biológicas, algunos movimientos internos de estas moléculas ocurren lapsos más cortos que otros. Experimentalmente, el promedio de la distancia de enlace covalente y el ángulo covalente son relativamente constantes, llevando a la suposición de que los cambios conformacionales observados a través de los ángulos diedros podrían determinar completamente la forma general de la molécula de proteína como una función de sus coordenadas internas (distancia de enlace, ángulos de enlace y ángulos diedros). Bajo la suposición de longitudes de enlace constantes y ángulos de enlace constantes, el problema reduce drásticamente su tamaño en el espacio conformacional, que por sí mismo es extremadamente complicado por muchos factores externos que también dan origen a una gran cantidad de mínimos.

Varios campos de fuerza diferentes han sido diseñados como la suma de un conjunto de contribuciones a la energía potencial. Entre los más usados están: ECEPP [30], MM2 [20], ECEPP/2 [30], CHARMM [3], DISCOVER [15], AMBER [4], MM3 [20] y ECEPP/3 [30]. En este trabajo se utiliza el campo de fuerza ECEPP/3 [30], que supone que todas las longitudes de los enlaces covalentes y los ángulos de enlace se encuentran fijos en sus valores de equilibrio. Que la energía conformacional correspondiente surge de una suma de las siguientes energías: la electrostática, de no-enlace de puentes de hidrógeno y contribuciones de energía torsional, además de varios potenciales empíricos para el cerrado del ciclo de un puente bisulfuro y una energía conformacional fija para el anillo pirolidino en ambos residuos prolil e hidroxiprolil.

Aunque el tamaño del problema puede reducirse cuando la función de energía se escribe en términos de los ángulos torsionales, es sabido que cuando una función se somete a técnicas de optimización simple, generalmente se llegará a mínimos locales. Para superar este efecto muchos autores han diseñado interesantes métodos estocásticos y deterministas que imponen restricciones y predisponen la búsqueda hacia la región donde puede encontrarse el mínimo global. Entre los métodos estocásticos empleados se encuentran el Recocido Simulado (RS) [25], Método de Umbrales (MU) [22], Montecarlo con Minimización (MCLM) [27], Ensamble Multicanónico (EM) [8] y los Algoritmos Genéticos tradicionales (AG) [10]. Entre los métodos deterministas se encuentran el Método de la Ecuación de Difusión (MED) [17], la Técnica del Campo Medio (TCM) y la Programación Dinámica (PD). La mayoría de estos métodos presentan una baja eficiencia y están limitados por la di-

mensión del problema. Muy recientemente un método llamado αBB [1] ha reclamado tener la garantía teórica de obtener la solución global mínima de funciones analíticas diferenciables.

Esta tesis forma parte de un proyecto que experimenta con un método heurístico que pretende reducir el campo de la búsqueda conformacional y el tiempo necesario para encontrar una estructura 3D cerca o en el mínimo global de péptidos. Reportes previos han lidiado con el uso de Recocido Simulado (RS) y Método de Umbrales (MU), para encontrar un mínimo global de un péptido bien estudiado, la Met-encéfalina, que a través de los años ha servido como modelo de validación. Con RS y MU se ha encontrado excelentes estructuras de baja energía, pero estos métodos carecen de eficiencia. En este trabajo, se sugiere un acercamiento usando Algoritmos Genéticos (AG), un método estocástico de optimización desarrollado para tratar grandes tareas de optimización combinatoria. Los Algoritmos Genéticos aplicados en otros problemas de semejante complejidad han probado tener una buena eficiencia en los resultados que se obtienen en un tiempo relativamente corto. Se ha diseñado un procedimiento de optimización local para encontrar la estructura más baja de energía mínima de la Met-encéfalina usando un optimizador local llamado SUMS con respecto a la función empírica de energía ECEPP/3.

Una breve descripción del campo de fuerza ECEPP/3 es:

$$U = U_{elec} + U_{nonb} + U_{hb} + U_{tor} + U_{loop} + U_{S-S},$$

donde

$$U_{elec} = \sum_{\text{ß}} \sum_{i \neq j} 332.0 q_{\text{ß}} q_j / Dr_{ij}$$

$$U_{nonb} = \sum_{\text{ß}} \sum_{i \neq j} FA / r_{\text{ß}}^{12} - C / r_{ij}^{6}$$

$$U_{hb} = \sum_{\text{ß}} \sum_{i \neq j} A'_{hx} / r_{hj}^{12} - B_{hx} / r_{hx}^{10}$$

$$U_{tor} = \sum_{k} (U_0/2.0)(1 \pm \cos n_k \theta_k)$$

$$U_{loop} = \sum_{1} B_1 \sum_{i=1}^{i=3} (r_{i1} - r_{i0})^2$$

$$U_{S-S} = \sum_{S} A_s (r_{4s} - r_{40})^2$$

Todas las constantes son ajustadas apropiadamente a datos experimentales.

2.2 Estructura de las Proteínas

En los organismos vivos, las proteínas están implicadas en una mayor cantidad y variedad de eventos bioquímicos que cualquier otro tipo de moléculas, De ahí la importancia de su estudio.

Los aminoácidos, son las unidades básicas de las proteínas. Los aminoácidos se unen por medio de los enlaces peptídicos para formar cadenas peptídicas. A cada una de estas unidades se le conoce como residuo. Un residuo consiste en un grupo amino, un grupo carboxilo, un átomo de hidrógeno y un grupo distintivo "R"[1] enlazados al átomo de carbono que se llama carbono-α. Una proteína o cadena peptídica está formada por una cadena principal y una cadena lateral. El grupo distintivo R de cada aminoácido forma la cadena lateral, Figura 2.2

Existen veinte tipos de cadenas laterales de aminoácidos, estas varían en tamaño, forma, carga, capacidad de puentes de hidrógeno y reactividad química. Estos tipos son el alfabeto fundamental de las proteínas. A los ángulos internos de la cadena se les conoce como ángulos χ, como se observan en la sección sombreada de las proteínas mostradas en la Figura 2.1.

Se llamará *conformación* a la organización de los átomos que componen de manera tridimensional la estructura de la proteína. Con esto debemos hacer notar que la secuencia de *monómeros*[2] que forman una proteína no cambia, sino que la forma de la proteína esta determinada en el espacio conformacional por la rotación de los ángulos de enlace.

[1] Un grupo "R" se refiere a una cadena lateral o conformación de los átomos que no forman parte de la cadena principal.

[2] El término monómero se utilizará para referirse a un sólo residuo o aminoácido.

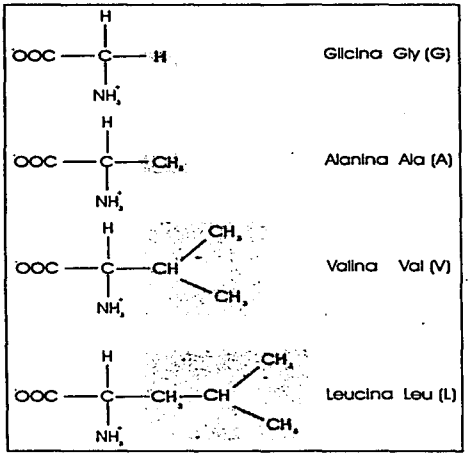

Figura 2.1: Ejemplos de estructuras de Aminoácidos.

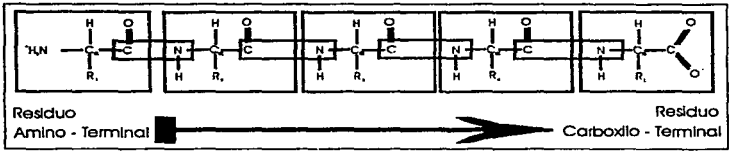

Figura 2.2: Polipéptido formado por 5 aminoácidos. La cadena comienza en el extremo-amino.

2.3 Espacio Conformacional de las Proteínas

El término *Conformación*, en general, se emplea para determinar los arreglos espaciales de una molécula por rotaciones en uniones de enlace simple. En el caso de las proteínas, la conformación describe la organización espacial del conjunto de *péptidos* que la constituye.

La conFiguración se refiere a un arreglo específico, característico y estable de átomos o grupos de átomos. La conversión de una conFiguración puede existir en un número infinito de conformaciones. Por ejemplo, si consideramos una proteína como una secuencia de péptidos específicos, su conFiguración se referirá al orden que tiene esa secuencia, mientras que su conformación indicará la posición en el espacio de cada uno de los péptidos que forman esa secuencia.

Una *cadena peptídica* tienen una dirección porque sus elementos constructores tienen extremos diferentes. Por convención se toma al extremo amino como el comienzo de la cadena, Figura 2.2

En las proteínas el grupo *alfa carboxilo* se une al grupo *alfa amino* por medio de un *enlace peptídico*, la unión de dos aminoácidos se lleva a acabo a través de la pérdida de una molécula de agua.

Una característica de las proteínas es que poseen estructuras tridimensionales bien definidas. Una cadena estirada u organizada al azar no tiene actividad biológica.

El grupo *peptídico* es una *unidad planar rígida*, que quiere decir que el hidrógeno del grupo - NH - esta casi siempre en posición *"trans"* (ángulo diedro de 180 grados) respecto" al oxigeno del *grupo carboxilo* (CO); existe muy poca libertad de rotación alrededor del enlace peptídico. A esta rotación se le llama ángulo ω.

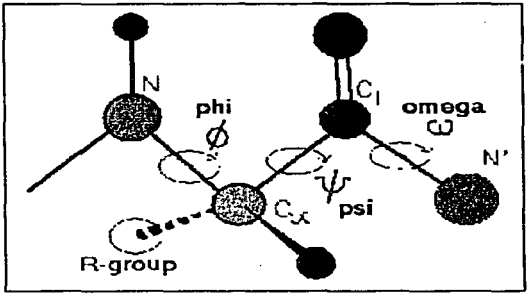

Figura 2.3: Definición de los ángulos ϖ, ϕ, ψ en la estructura de la proteína.

Existe un enorme grado de libertad torsional alrededor de los enlaces entre el carbono α y el *carbono carboxílico* (CO), y entre el *carbono* α y el átomo de *nitrógeno amida* (enlace sencillo puro). A estas rotaciones se les llama ángulos ϕ y ψ.

La conformación de la cadena principal o columna vertebral del polipéptido está definida completamente cuando por valores de los ángulos ϕ y ψ para cada uno de los residuos de los *aminoácidos*.

La secuencia de *aminoácidos* es muy importante, ya que determina la conformación de la proteína específica. Al mismo tiempo, la función de una proteína nace de la conformación que esta tenga.

Una molécula polipeptídica será tomada como una gráfica con nodos correspondientes a los átomos y con arcos correspondientes a los enlaces atómicos entre los átomos. En esta gráfica las distancias acotadas (enlaces atómicos) y los ángulos acotados (ángulos entre dos átomos adyacentes enlazados) son invariantes bajo cambios conformacionales. Sea k el número de residuos de aminoácidos en la molécula. Sean ϕ_i, ψ_i y ω_i los ángulos diedros de la columna vertebral correspondiente a la i-ésimo residuo aminoácido (ver Figuras 2.3 y 2.4). For $i = 1, \ldots, k$, sean $\chi_{i_1}, \ldots, \chi_{i_q}$ los ángulos diedros en las cadenas laterales del i-ésimo residuo.

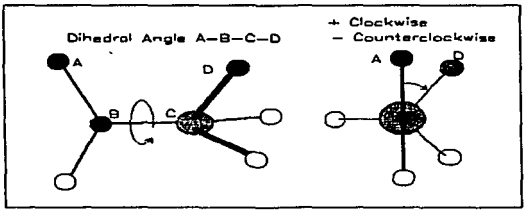

Figura 2.4: Definición del ángulo dihedro, positivo y negativo en la estructura de la proteína.

Dadas estas definiciones y dado el campo de fuerza de ECEPP3, entonces el problema de encontrar la estructura molecular con la energía potencial más baja se puede formular como el siguiente de problema de optimización:

$$\text{Minimizar } U(\phi_1, \psi_1, \omega_1, \chi_{i_1}, \ldots, \chi_{i_1}, \ldots, \phi_k, \psi_k, \omega_k, \chi_{k_1}, \ldots, \chi_{k_q})$$

sujeta a las siguientes restricciones experimentales

$$-180° \leq \phi_i, \psi_i \leq 180° \quad i = 1, \ldots, k.$$

$$-10° \leq (\omega_i - 180°) \leq 10° \quad i = 1, \ldots, k,.$$

$$-180° \leq \chi_{ij} \leq 180° \quad i = 1, \ldots, k, \quad j = 1, \ldots, q_i.$$

Capítulo 3

Optimización Combinatoria

En este capítulo se hará una breve introducción a la optimización combinatoria y los problemas que puede estudiar.

3.1 Problemas de Optimización Combinatoria

Se ha estudiado el problema de buscar soluciones óptimas a problemas que pueden ser estructurados como una función de algunas variables de control, y tal vez con la presencia de algunas restricciones. Tales problemas pueden ser formulados de la siguiente manera:

Minimizar $f(x)$ sujeta a:
$g_i(x) \geq b_i$; $\quad i = 1, ..., m$
$h_j(x) \geq c_j$; $\quad j = 1, ..., n$

Donde x es un vector de variables de control y $f(x)$, $g_i(x)$ y $h_j(x)$ son funciones de cualquier tipo.

Existen diversas clases de estos problemas, obtenidos al aplicar restricciones sobre el tipo de funciones bajo consideración y en los valores que las variables de control puedan tomar. Los problemas más conocidos son los obtenidos al restringir que $f(x)$, $g_i(x)$ y $h_j(x)$ sean funciones lineales de las variables de control, las cuales son continuas. Esta es la clase de problemas de *programación lineal*.

Consideramos aquí otra clase de problemas; aquellos de naturaleza combinatoria. Este término es usualmente reservado para problemas en los cuales

las variables de control son discretas. El problema de buscar una solución óptima a tales problemas es conocido como *optimización combinatoria.*

Los problemas de optimización combinatoria y de programación lineal en algunas ocasiones tienen similitud y muchos de los intentos para resolverlos utilizan estos métodos, generalmente introduciendo variables enteras que toman valores entre 0 y 1, para producir una formulación de programación entera. Tales formulaciones en ocasiones involucran un gran número de variables y restricciones, y no pueden hacer frente a problemas muy grandes.

Un ejemplo de un problema de optimización combinatoria es la siguiente:

Problema de Asignación. Un conjunto de n personas está disponible para realizar n tareas. Si la i-ésima persona realiza la j-ésima tarea, ésta representa un costo de C_{ij} unidades. Entonces el problema consiste en encontrar una asignación $\pi_1, \ldots, \pi_n$ la cual se minimice:

$$\sum_{i=1}^{n} C_{i\pi_j}$$

Aquí la solución está representada por la permutación $\pi_1, \ldots, \pi_n$ de los números $1, \ldots, n$.

Debe quedar claro que, en el ejemplo anterior, utilizamos el término problema en el sentido genérico. Dada una situación en la vida real, tendremos un problema particular, en el cual los símbolos utilizados para describirlo deben tomar valores numéricos específicos. Es común usar el término ejemplar[1] para poder distinguir una situación particular de una general.

3.2 Óptimos Locales y Globales

La mayoría de los problemas de optimización combinatoria presenta una gran cantidad de mínimos globales y mínimos locales, esto también es un problema ya que tenemos el riesgo de quedar atrapados en alguno de los mínimos locales si lo que se desea es encontrar un mínimo global. Podemos hacer más concreta esta idea si introducimos el concepto de *vecindad.*

Estrictamente hablando, la vecindad $V(\chi, \sigma)$ de una solución χ es un conjunto de soluciones que pueden ser alcanzadas a partir de χ aplicando una simple operación σ. Tal operación podría ser quitar o agregar un objeto a la solución o hacer el intercambio de dos objetos de la solución.

[1] Utilizaremos el término ejemplar de un problema como traducción literal de *instance of a problem.*

El algunos casos es posible encontrar un *movimiento* S, tal que un óptimo local sea también un óptimo global. Hay muchos tratamientos existentes para la optimización combinatoria, pero estos tienden a concentrarse en métodos que son exactos en vez de heurísticos. Esto es, están principalmente relacionados con aquellas técnicas que garantizan encontrar una solución óptima al problema establecido. Estos métodos usualmente están en la teoría de la *programación lineal*, o usan métodos de *enumeración implícita* como el *branch and bound* [1].

3.3 Heurísticas

Este término deriva del griego *heuriskein* que significa encontrar o descubrir. Sin embargo, las heurísticas no garantizan encontrar una solución óptima. El término heurístico es usado en contraste a los métodos que si garantizan encontrar una solución óptima y ha llegado a ser común en el contexto de la optimización combinatoria.

Definición. Una heurística es aquella que ayuda a guiar al proceso de búsqueda mejorando en cada intento de aproximación su eficiencia. Sin embargo, no garantiza encontrar una solución óptima, pero si nos proporciona buenas soluciones, cercanas a la óptima, con un costo computacional razonable.

Una gran cantidad de artículos tratan de cómo las técnicas heurísticas se han utilizado para resolver problemas de optimización combinatoria. Pareciendo doble la causa de este gran interés. Por un lado, el desarrollo del concepto de complejidad computacional ha proporcionado las bases para explorar las heurísticas. Por otro lado, han surgido nuevas técnicas, muy eficientes, para resolver problemas de optimización combinatoria en tiempos razonables de cómputo.

El método clásico para resolver ejemplares de un problema de optimización combinatoria es simplemente listar todas las soluciones factibles de un ejemplo dado, evaluar su función objetivo y escoger la mejor solución. Podríamos pensar que la solución de un problema de optimización combinatoria únicamente se limita a buscar de manera exhaustiva el valor que minimice o maximice la función objetivo dentro de un conjunto finito de posibilidades, tal vez utilizando una computadora muy rápida, el problema carecería de interés matemático, sin detenernos a pensar por un momento en el tamaño de este conjunto de soluciones.

Sin embargo, es obvio que este método es muy ineficiente conforme crece el tamaño de la entrada del problema debido a la explosión combinatoria del espacio de soluciones, es decir, dado un conjunto de elementos que pueden obtener diferentes arreglos ordenados de estos, permitiendo una gran cantidad de posibilidades para cualquier entrada de tamaño considerable.

Para ilustrar este punto, considéresc el problema de Agente Viajero (travelling Salesman Problem), el cual se ha utilizado como referencia de estos probablemente por que es muy fácil de describir, pero muy difícil de resolver. Sin embargo, se han utilizado ya técnicas heurísticas junto con técnicas de paralelización para resolver ejemplares de gran tamaño para este problema con éxito [12].

Problema del Agente Viajero. Un vendedor tiene que buscar una ruta que visite cada una de las N ciudades, una y sólo una vez iniciando en cualquiera de ellas y volviendo a la misma ciudad. Como el punto inicial es arbitrario, hay (N-1)! posibles soluciones o (N-1)! / 2 si la distancia de las ciudades es la misma sin considerar la dirección del viaje.

Supóngase que tenemos una computadora que puede listar todas las posibles soluciones de un ejemplar; entonces utilizando la fórmula anterior, tendríamos como resultado la Tabla 3.1 la cual nos muestra el crecimiento del tiempo de cómputo con respecto al tamaño de la entrada del problema, podemos observar que la enumeración completa es ineficiente para obtener una solución óptima, ya que por ejemplo un problema de 25 ciudades tomaría aproximadamente 6 siglos en ser resuelto.

Tabla 3.1 Crecimiento del tiempo de cómputo

Número de ciudades	Tiempo de cómputo
20	1 hora
21	20 horas
22	17.5 días
23	1.05 años
24	24.26 años
25	5.82 siglos

Varios algoritmos exactos han sido inventados para encontrar soluciones óptimas de problemas más eficientes que la *enumeración completa*. El algoritmo más conocido es el *Simplex* para problemas de *programación lineal*.

Estos algoritmos fueron capaces de resolver ejemplares pequeños, pero no lo fueron para encontrar soluciones óptimas, en una cantidad razonable de tiempo computacional, cuando los ejemplares son grandes. Como el poder computacional se ha incrementado en los últimos años, ha sido posible resolver grandes problemas, y los investigadores se han interesado en cómo el tiempo de solución varía con el tamaño de la entrada del problema.

Sin embargo, para problemas tales como el del agente viajero, el esfuerzo computacional sigue siendo exponencial. A finales de los años 60's los investigadores se hacían la siguiente pregunta: ¿Habrá un algoritmo de optimización en tiempo polinomial para un problema como el del Agente Viajero? Nadie ha sido capaz de responder a esta pregunta, sin embargo en 1972, Karp [16] mostró que sí la respuesta es afirmativa para el problema de Agente Viajero, luego entonces hay también un algoritmo en tiempo polinomial para otros problemas equivalentes. Como ningún algoritmo ha sido encontrado para estos problemas, esto indica que la respuesta a la pregunta original es probablemente no. Sin embargo, la respuesta real es desconocida.

Hay problemas que tienen algoritmos *polinomiales conocidos* y se dice que están en *clase* P. Pero ¿qué hay del problema del agente viajero y otros problemas equivalentes? ¿son de complejidad exponencial? Muchos de estos problemas están en la clase NP-completos, que es una abreviación de *Nondeterministic Polinomial* [5]. De hecho el problema del agente viajero es de los problemas más difíciles en NP-completos. Si un algoritmo polinomial fuera encontrado para este problema, esto significaría que existe un algoritmo polinomial para todos los problemas NP-completos; pero como ningún algoritmo polinomial exacto ha sido encontrado para cualquier problema NP-completo, hay fuerte evidencia de que el problema es NP-completo, lo cual es un argumento para buscar formas alternativas de resolver problemas computacionalmente difíciles. Existe la posibilidad de que alguien llegue a probar que P = NP; pero mientras nadie encuentre tal prueba, el uso de las técnicas heurísticas tienen una justificación considerable.

Existen otros argumentos a favor del uso de las heurísticas: lo que realmente se está optimizando es un modelo del mundo real, por eso no hay garantía de que la mejor solución del modelo sea también la mejor solución para el problema del mundo real. Las técnicas heurísticas son usualmente más flexibles y son capaces de hacer lo mismo con funciones objetivo y/o restricciones más complicadas que los algoritmos exactos. Este es el caso de técnicas tales como los Algoritmos Genéticos; donde la función objetivo no necesitan cumplir hipótesis de linealidad. Esto nos permite modelar pro-

blemas del mundo real aún más precisamente que con el uso de algoritmos exactos.

Muchos problemas de optimización combinatoria son problemas específicos, de manera que una técnica heurística que funciona para un problema, puede no ser útil para resolver otro problema diferente. Sin embargo, hay un gran interés en técnicas que se han desarrollado en la última década y que pueden ser aplicables con mayor generalidad.

Algunas de estas técnicas han sido desarrolladas usando como estrategia la *búsqueda local de vecinos* (*local neihborhood search*). El proceso inicia con una solución para un ejemplar y busca una mejor solución dentro de una vecindad definida. Habiendo encontrado una mejor solución, el proceso reinicia la búsqueda local con esta nueva solución y esto continúa iterativamente hasta que no puede mejorar la solución actual encontrada. Esta solución final, probablemente sea un óptimo global, aunque con respecto a su vecindad es un óptimo local.

Una variación a lo anterior que ha ganado recientemente atención, es permitir movimientos ascendentes, con lo cual la solución con la que la búsqueda local reinicia puede ser peor que la anterior, considerando que debe haber algunas restricciones para aceptar tales movimientos, en otro caso el procedimiento se sumaría a la búsqueda del espacio completo de soluciones. Por lo tanto, se da la oportunidad de que el proceso pueda escapar o salir de óptimos locales y busque una mejor solución.

Capítulo 4

Algoritmos Genéticos

En este capítulo se describirá qué es un algoritmo genético, los principios naturales en que se basa su funcionamiento y los operadores que lo conforman. También se dará una breve historia de éste y los diferentes cambios que se han hecho para mejorar su rendimiento. Los Algoritmos Genéticos son de las técnicas heurísticas más utilizadas, pues han demostrado su funcionamiento y precisión en diversos problemas tanto de optimización o dando buenas aproximaciones a la soluciones para algoritmos NP-Completos [7].

4.1 Evolución Natural.

Los Algoritmos Genéticos son un intento de minimizar los procesos observados en la evolución natural. Esta técnica heurística fue desarrollada por John Holland [7] en su libro *"Adaptation in Natural and Artificial Systems"* [11] y trabajos posteriores. Muchos investigadores han contribuido para mejorarlos o desarrollar variantes de estos.

Los biólogos han estado intrigados con el mecanismo de la evolución desde que la Teoría de la Evolución fue aceptada. Varios investigadores, incluso los biólogos, quedan sorprendidos con la vida y el nivel de complejidad que se observa en cuanto al desarrollo de las poblaciones para las diferentes especies en mucho tiempo, principalmente observados en los fósiles prehistóricos.

Este mecanismo de la evolución no ha sido completamente entendido, pero muchas de sus características sí. La evolución toma el nombre de *cromosoma* -como un dispositivo orgánico para codificar la estructura de los seres vivos [6]. Un ser vivo es creado a partir de un proceso de *decodificación*

de los cromosomas. La especificación de la codificación del cromosoma y el proceso de decodificación no a sido totalmente entendida, pero muchas de las características de la teoría han sido extensamente aceptada:

- La evolución es un proceso que opera con cromosomas de un ser vivo que está codificado.
- La selección natural es una liga entre cromosomas y el rendimiento de las estructuras decodificadas. El proceso de la selección natural causa que estos cromosomas que tienen estructuras codificadas acertadas, se reproduzcan más que otros que no lo son.
- El proceso de reproducción es un punto en que la evolución ocurre. La mutación puede causar que los cromosomas del hijo biológico sean diferentes a los de sus padres biológicos y el proceso de recombinación hace que se creen diferentes cromosomas en el hijo por material combinado de los cromosomas de dos padres.
- La evolución biológica no tiene memoria. Cualquiera que sepa sobre la producción de los individuos sabe que su función está a voluntad del medio ambiente sobre sus genes - este conjunto de cromosomas llevados por los individuos actuales- y en la estructura de la decodificación de los cromosomas.

Estas características de la evolución intrigó a John Holland en la década de los setentas. Holland creía que, incorporándola apropiadamente en un algoritmo de computadora, esta técnica puede servir para resolver problemas difíciles como la naturaleza lo hace bien en la evolución [10]. Entonces empezó a trabajar con este algoritmo que manipula cadenas de dígitos binarios -1's y 0's- al las que le llamo *cromosomas* [6]. El algoritmo de Holland trata de simular la evolución de individuos que tienen cromosomas [7]. Como en la naturaleza, este algoritmo resuelve el problema de encontrar un buen cromosoma que manipula el material en los cromosomas ocultos. Como en la naturaleza, no se tiene idea del problema que se está resolviendo. La única información que se tienen es la evaluación de cada uno de los cromosomas que se producen, y sólo esta evaluación hace que en la selección de los cromosomas se seleccione las mejores evaluaciones que se tengan y estas se reproducirán mucho más que las malas evaluaciones.

Estos algoritmos, usan codificaciones simples y mecanismos de reproducción, comportamiento complicado de despliegue y son usados para resolver

problemas extremadamente difíciles. Como en la naturaleza, esto lo hace sin el conocimiento del mundo decodificado. Estas son sencillas manipulaciones de cromosomas simples. Esto ha provocado el desarrollo de un gran número de algoritmos de este tipo hoy en día, en donde se ha querido encontrar mejores diseños, encontrar mejores tiempos y producir mejores soluciones para una variedad de otros problemas también importantes, en donde no se pueden utilizar otras técnicas.

Cuando Holland empezó a estudiar estos algoritmos, no sabía que nombre ponerles. Y como se demostraba su potencial, decidió ponerles el nombre de Algoritmos Genéticos [10].

Como se sabe, los descubrimientos provienen de la evolución biológica y también se integran a los Algoritmos Genéticos, entonces la biología y la genética lo continuan influenciando. Esta influencia es gran parte unidireccional. Se sabe que la aplicación de los algoritmos no genéticos en el área de la genética, no ha tenido un gran impacto en la Biología. En este punto, los algoritmos genéticos se parecen a las redes neuronales y al recocido simulado, que también son algoritmos basados en la metamorfosis del mundo natural.

La descripción del algoritmo genético de la Figura 4.1 es una generalización de los Algoritmos Genéticos. Entre las distintas implementaciones de los Algoritmos Genéticos se encuentran los programados en lenguaje estructurado, así como también en lenguajes funcionales y lenguajes orientados a objetos.

4.2 Vista de Alto Nivel de un Algoritmo Genético

Se puede abstraer el fenómeno natural en un algoritmo de varias maneras. Primero se tiene que considerar el mecanismo que liga al algoritmo genético con el problema que se quiere resolver. Aquí hay dos mecanismos posibles -el camino de las soluciones codificadas al problema en los cromosomas y la función de evaluación que regresa una medida del trabajo de cualquier cromosoma en el contexto del problema.

La técnica para codificar las soluciones varia de problema a problema y de un algoritmo genético a otro. En el algoritmo de Holland (Figura 4.1), él y muchos de sus estudiantes, codifican usando una cadena de binarios [10]. Probablemente no existe una mejor técnica para resolver todos los problemas

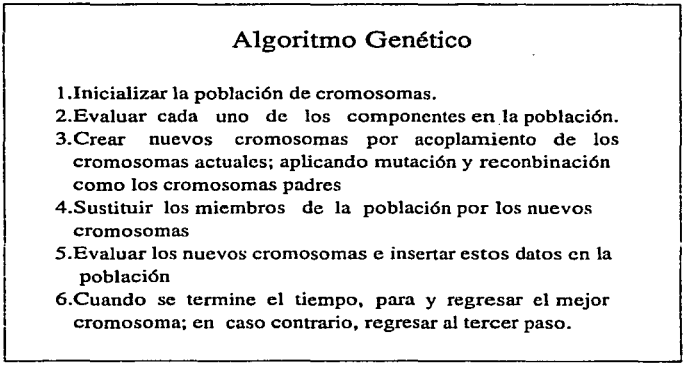

Figura 4.1: Descripción a alto nivel de un algoritmo genético.

y ciertamente el seleccionar una buena técnica de decodificación es un problema que debe ser atacado. Se debe tener en cuenta las técnicas de selección y representación del contexto del problema en el mundo real.

La función de evaluación es una liga entre los algoritmos genéticos y el problema a resolver. Una función de evaluación toma la entrada de un cromosoma y regresa un número que es la medida del rendimiento del cromosoma en el problema a resolver. La función de evaluación juega el mismo papel en un algoritmo genético que el que toma el medio ambiente en la evolución natural. La interacción de los individuos con su medio provee una medida de su aptitud - *fitness* - que el algoritmo genético utiliza para llevar a la reproducción.

Dados los componentes iniciales al problema, el camino para codificar las soluciones y la función que regresa es una medida que indica que tan buena es la codificación. Se puede usar un algoritmo genético para llevar a cabo la evolución simulada en la población de soluciones.

Si todo funciona como en el proceso de la evolución simulada en la población inicial, los cromosomas se mejoran. Los padres son remplazados por mejores hijos. El mejor individuo en la población final producida es el

que contiene la mejor aproximación de la solución del problema.

4.3 Selección de los Padres, Mutación y Cruzamiento

En esta sección se describirá de una manera detallada los principales operadores que tienen los Algoritmos Genéticos y la forma de como funcionan. Estos operadores presentan su funcionamiento en base de cromosomas binarios, en el siguiente capítulo de Algoritmos Genéticos híbridos se dirá cómo son estos operadores con cromosomas con otro tipo de numerología partiendo de los binarios.

4.3.1 Selección de los Padres por Ruleta

El propósito de la selección de los padres en un algoritmo genético es seleccionar quien tiene más oportunidades de cruzamiento, o en este caso, quien de los miembros esta más ajustado. Una de las técnicas más utilizadas es la selección de los padres por ruleta, la cual se describe en la Figura 4.2.

El efecto de la ruleta en la selección de los padres regresa un padre seleccionado aleatoriamente. Aunque esta selección sea al azar, cada oportunidad que tienen los padres de ser seleccionados es directamente proporcional a su adaptación - *fitness*. En balance, sobre el número de generaciones de este algoritmo, este conduce a tener los miembros más adaptados y contribuye a separar el material genético en el *fitness* de los miembros de la población. Es posible que se seleccione el peor individuo de la población cada vez que se usa. Pero esta ocurrencia no inhibe el rendimiento del algoritmo genético seleccionando esta técnica, las probabilidades de esta ocurrencia en la población de cualquier tamaño son despreciables.

Esta técnica de selección de padres tiene las ventajas de que directamente promueve la reproducción de los miembros de la población más ajustados por predisponer la oportunidad de cada uno de los miembros acorde con su evaluación.

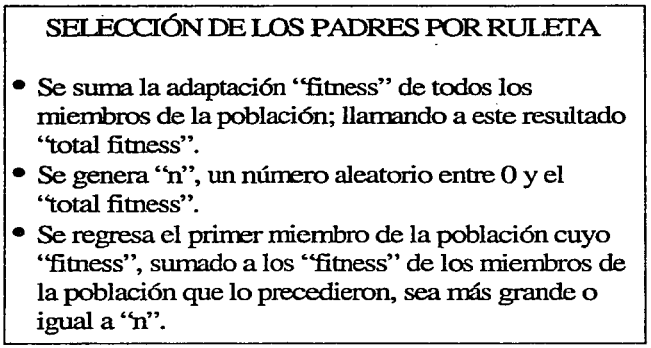

Figura 4.2: La ruleta para el algoritmo de selección de los padres.

4.3.2 Cruzamiento y Mutación por Punto

El cruzamiento y la mutación por punto son componentes básicos para los Algoritmos Genéticos tradicionales. Estas funciones causan que la creación de los cromosomas durante la reproducción de dos diferentes padres. Primero el operador toma y copia a cada uno de los padres -se crean dos hijos. Después se aplican las dos funciones a cada uno de los hijos y estos son alterados. A continuación se detallará más profundamente cada una de estas funciones.

Mutación por bit. La mutación por bit es un procedimiento que se lleva a cabo por cada punto en el cruzamiento y la mutación. Cuando la mutación de bit es aplicada a una cadena de bits este barre toda la lista, sustituyendo cada uno de los bits por un bit seleccionado aleatoriamente si la prueba de probabilidad es pasada. La mutación por bit está asociada a un parámetro de mutación que es típicamente muy bajo.

La Figura 4.3 contiene ejemplos de la operación de mutación por bit. Se tienen tres cromosomas padres de longitud 4, el número generado aleatoriamente por el resultado de la probabilidad de la mutación, el bit generado aleatoriamente que es reemplazado por el bit que ha pasado la prueba de probabilidad y los tres cromosomas resultantes de la acción de mutación por

EJEMPLO DE MUTACIÓN POR BIT												
Cromosomas antiguos				Numeros aleatorios				Bits nuevos	Cromosomas nuevos			
1	0	1	0	0.801	0.102	0.266	0.373	-	1	0	1	0
1	1	0	0	0.120	0.096	0.005	0.840	0	1	1	0	0
0	0	1	0	0.760	0.473	0.994	0.001	1	0	0	1	1

Figura 4.3: Ejemplo de mutación por bit. La tabla muestra tres cromosomas de longitud 4, números generados aleatoriamente para cada bit de los cromosomas, nuevos bits para las dos ocaciones en que la prueba de los números aleatorios a sido pasada y el resultado de los cromosomas. Los dos números aleatorios que causan que se generen nuevos bits están resaltados en el texto.

bit. Cuando vemos el primer cromosoma, la prueba de probabilidad nunca es aprobada y cuando vemos el cromosoma de salida este es el mismo que el de entrada. En el segundo caso, la probabilidad es pasada por el cuarto bit. Algunas veces el bit generado aleatoriamente es el mismo que el bit original y éste no tiene un cambio efectivo. Así, para el tercer cromosoma en la Figura es el único que es cambiado por el operador de mutación por bit.

Aquí se tiene una fuente potencial de confusión por la mutación por bit. Muchos Algoritmos Genéticos usan la mutación de bit por cambio de bit. Usando esta variante, si la probabilidad de mutación es pasada, reemplazan el 1 por el 0 y viceversa. Esta operación resulta ser muy efectiva, ya que la probabilidad de que la mutación genere gemelos es muy alta.

Cruzamiento por punto. Este es otro de los procesos que alteran los cromosomas durante la reproducción y muchos biólogos especialistas en evolución creen que es más importante que la mutación. Este operador es llamado *cruzamiento*. En la naturaleza, el cruzamiento ocurre cuando dos padres intercambian partes de sus correspondientes cromosomas. En un algoritmo genético, el cruzamiento combina el material genético de dos cromosomas padres a dos cromosomas hijos. John Holland creó un operador de cruzamiento llamado *cruzamiento por punto*. El cruzamiento por punto ocurre cuando las partes de dos cromosomas padres son intercambiadas después de que se ha seleccionado un punto aleatoriamente, creando dos hijos. En la Figura 4.4 muestra dos ejemplos de la aplicación del cruzamiento por punto durante el proceso de un algoritmo genético.

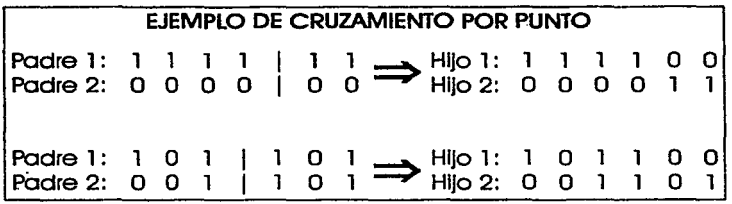

Figura 4.4: Dos ejemplos de cruzamiento por punto, Los hijos son creados por partición de los padres en el punto denotado por la línea vertical e intercambiando material genético de los padres despues del corte.

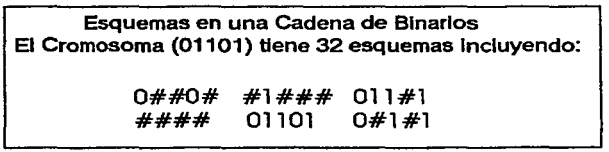

Figura 4.5: Ejemplos de esquemas en una cadena simple de binarios.

Una importante característica del cruzamiento por punto es que puede producir hijos que son radicalmente diferentes de sus padres. En el primer ejemplo de la Figura 4.4 se representa esta situación. Otra importante característica es que a partir del punto de cruzamiento este no introduce diferencias de los bits en la posición donde los padres tienen el mismo valor. En la Figura 4.4, tenemos que el bit de la posición 2, 3, 4 y 5 tiene el mismo valor en los dos padres y en los dos hijos, siempre que el cruzamiento ha ocurrido. Un caso extremo ocurre cuando dos padres son idénticos. En estos casos, el cruzamiento no introduce diversidad en los hijos.

El cruzamiento es un componente extremadamente importante en un algoritmo genético. Muchos partidarios de Algoritmos Genéticos creen que si se borra este operador del algoritmo genético resultaría en un algoritmo muy poco práctico. Esta conclusión no se aplica para el operador de mutación.

Este operador se distingue de los demás por que es un acelerador crítico para el proceso de búsqueda cuando un algoritmo genético está ejecutándose, ya que la reproducción sexual es el proceso que más reacciones fuertes tiene en la naturaleza. La reproducción sexual realiza una combinación muy rápida cuyos beneficios no pueden ser duplicados por la mutación.

El punto central del cruzamiento es la combinación de *bloques de estructuras* de buenas soluciones para diversos cromosomas. La Figura 4.5 muestra un cromosoma sencillo de longitud 5 y varios bloques de estructuras contenidos en este cromosoma. Cada bloque de estructura es representado por una lista creada por tres caracteres - 1, 0 y "#". El 1 y el 0 en cualquier posición del bloque de estructura en el cromosoma que tiene el mismo valor en la posición que lo contiene en el bloque de estructura. El "#" en cualquier posición del bloque de estructura significa que el valor del cromosoma en su posición es irrelevante para determinar que parte del cromosoma contiene el bloque de estructura. La parte del cromosoma que contiene cadenas de 0's y 1's están los bloques de estructuras. Los caracteres "#" no importan.

Holland llama a estos bloques de estructuras como *esquemas* [6], en investigaciones sucesivas de los Algoritmos Genéticos concluyen en la manipulación de los *esquemas* cuando se ejecutan. En efecto, El Teorema del Esquema de Holland [6] nos dice precisamente esto. Si se usan las técnicas de reproducción las oportunidades de hacer que se reproduzcan los individuos es directamente proporcional a la adaptación -*fitness*- de los cromosomas, con esto se puede predecir el relativo incremento o decremento de los esquemas de s en la siguiente generación en el algoritmo genético como sigue. tenemos que r es el promedio del *fitness* de todos los cromosomas en la población con-

teniendo a s. tenemos que n es el número de los cromosomas en la población conteniendo a s. tenemos que a es el promedio del *fitness* de todos los cromosomas de la población. Entonces el número esperado de concurrencias de s en la siguiente generación de la población es $(n * r/a)$, menos las rupturas causadas por la mutación y cruzamiento.

En efecto, el Teorema de los Esquemas nos dice que una estructura que está contenida en un cromosoma con evaluaciones aproximadas al promedio estas tienden a ocurrir en la siguiente generación y una ocurrencia debajo del promedio de las evaluaciones estas tienden a ocurrir menos frecuentemente, ignorando los efectos de la mutación y cruzamiento.

Con esto se concluye que un Algoritmo Genético con cruzamiento tiende a ganar sobre un algoritmo similar sin cruzamiento con tal de que estas condiciones obtengan miembros diversos contenidos en la población; ejemplos representativos de diferentes bloques de estructuras para una buena solución que está disponible en la población; la función de evaluación refleja las contribuciones de los bloques de estructuras; y el cruzamiento es capaz de reunirlos.

Capítulo 5

Algoritmos Genéticos Híbridos

En esté capítulo veremos que son los Algoritmos Genéticos Híbridos, la diferencia que tienen estos con los Algoritmos Genéticos tradicionales y particularmente el optimizador local numérico "SUMSL" [13] usado como un adicional operador "genético" el cual mejora uno o más individuos. También se mecionarán las diferentes técnicas que existen de Hibridación y la técnica que se utilizó para este sistema en particular.

5.1 Hibridación

La meta central en el desarrollo de un Algoritmo Genético es encontrar una forma de que el algoritmo sea robusto [6]. Este desarrollo se ha creado para que los Algoritmos Genéticos resuelvan una gran variedad de diferentes tipos de problemas. Esta implementación ha servido bien para usarse en la representación binaria de los cromosomas y se han singularizado los operadores genéricos. La representación binaria puede codificar casi cualquier cosa y los operadores no incluyen el conocimiento del dominio de la optimización que se pueden estar haciendo.

Cualquiera de las optimizaciones que se quieran realizar ocasiona que se tengan diferentes metas. Para que se pueda pagar una aplicación de un Algoritmo Genético, se puede persuadir a la persona que tiene un difícil problema de optimización que ésta cree el mejor algoritmo para resolver este problema en particular, optimizando el cómputo y el tiempo de la corrida. Aunque para el Algoritmo Genético una representación binaria y cruzamiento por punto y mutación binaria son algoritmos robustos, estos por lo regular no

son los mejores algoritmos para usarse en cualquier problema. Es un hecho incluso en la naturaleza, para los individuos de cada una de las especies, que por una gran variedad de nichos que se pueden encontrar en un medio ambiente, el mejor individuo de un nicho no sea siempre el mejor adaptado para el otro y este fallará en la competencia por recursos.

Para los partidarios de los Algoritmos Genéticos que buscan optimizar la vida de los individuos, hacen que la naturaleza cumpla con la adaptación ha los nichos que se quieran incorporar. Estos nichos incorporados son difíciles, ya que estos son los problemas de optimización del mundo real que se quieran resolver. Si esta adaptación procede, sólo nos servirá para este nicho en particular, ya que en los demás fallarán porque es diferente la competencia por los recursos. Para la gente del mundo real, esta no pagaría por múltiples soluciones ya que las comparaciones que se tendrían que hacer para encontrar la mejor solución serían muy caras. Se preferirá concentrarse en el desarrollo de un algoritmo en particular que se aproxime más a la solución del problema.

Después de que el proyecto de aplicar un Algoritmo Genético para resolver un problema de optimización es aprobado, los partidarios de estos algoritmos a menudo requieren predecir el desempeño de sus técnicas comparadas con las técnicas tradicionales ya desarrolladas sobre el dominio del problema; técnicas de inteligencia artificial para observar comparaciones prometedoras para mejorar la optimización; técnicas de máquinas de aprendizaje para ajustar el dominio del problema; y una gran variedad de herramientas nuevas y mejoradas que continuamente son desarrolladas en los campos de las Ciencias de la Computación y de la Optimización.

No pueden hacerse predicciones exitosas de este tipo para los algoritmos genéticos que se han estado investigando hasta ahora. Ya que no siempre los Algoritmos Genéticos pueden competir con aproximaciones especializadas para un problema.

5.2 Adaptación de un Algoritmo Genético

Cuando se conoce el problema que se quiere resolver, también se tiene una idea del rendimiento del algoritmo a utilizar aunque no sea el óptimo. El algoritmo que se usa en la mayoría de las ocasiones es el más familiarizado y se conoce el camino para resolver su problema. Entonces será muy difícil intentar otro camino utilizando nuevas técnicas para mejorar el rendimiento. Así el objetivo de hibridizar un algoritmo es poder decodificar técnicas que no sean poco familiares específicamente para el problema que tenemos que resolver [6], y con esto se obtienen un mayor número de recursos para llegar a esta solución.

Para lograr una buena hibridación se utilizan los siguientes tres principios básicos:

- Uso de la codificación actual: Se debe usar la misma técnica de codificación del algoritmo para el híbrido
- Hibridación cuando sea posible: Incorporar los desarrollos positivos del algoritmo actual para el híbrido.
- Adaptar los operadores Genéticos: Crear los operadores de cruzamiento y mutación para el nuevo tipo de codificación análogos con los operadores de cruzamiento y mutación de bits. Así como incorporar procedimientos basados en el dominio de la codificación como operadores.

A continuación se describirán con profundidad cada uno de estos principios y sus implicaciones.

5.2.1 Uso de la codificación actual

Usando la codificación actual se tienen dos ventajas. La primera: se garantiza que el dominio especializado en la codificación se continué preservando por el uso del algoritmo actual. La segunda: se garantiza que el algoritmo genético híbrido sea natural para el usuario, subsecuentemente el algoritmo híbrido puede ser operado por la misma estructura que el algoritmo actual con el que se está trabajando.

Usando la codificación actual, por este camino se pueden producir técnicas de codificación que pueden ser utilizadas para resolver problemas del mundo real. Y estas además son efectivas para generar buenos algoritmos de optimización.

5.2.2 Hibridación cuando sea posible

Este principio nos dice que hay que incorporar cualquier técnica de optimización que tengamos para transformar nuestro algoritmo genético en un algoritmo genético híbrido [6]. Esto nos trae una gran variedad de posibles caminos, incluyendo estos:

- Si el algoritmo actual es rápido, al algoritmo híbrido se le puede adicionar la solución o soluciones como una producción de la población inicial. Se garantiza que el algoritmo genético híbrido con elitismo no es peor que el algoritmo actual. En general, cruzando la solución del algoritmo actual sobre cada uno de los otros o sobre otra solución hace que se tenga más rendimiento.

- Si el algoritmo actual realiza transformaciones sucesivas en la codificación, puede ser muy útil incorporar esas transformaciones en su operador "set". Por ejemplo, en Montana and Davis [6] se describe el rendimiento de un algoritmo genético híbrido que usa "propagación hacia atrás" -una técnica de entrenamiento de redes neuronales- como su operador, junto con los operadores mutación y cruzamiento estos se adaptan al dominio de la red neuronal. Este algoritmo híbrido fue desarrollado siguiendo los tres principios de esta sub-sección.

- Si el algoritmo actual es bueno en la interpretación de la codificación, puede ser importante incorporar esta técnica de codificación en la técnica de evaluación. Los algoritmos adaptados al dominio del problema en la mayor parte de las ocasiones contienen codificación coadaptable y estrategias de decodificación. Con esto se incorpora la estrategia de la codificación. Este principio puede otorgar las características de la estrategia actual a la decodificación.

5.2.3 Adaptación de los Operadores Genéticos

Los dos anteriores principios nos dicen como incorporar lo que es bueno del algoritmo actual al algoritmo híbrido. Estos principios también nos dicen lo que hay que incorporar en un algoritmo genético.

Hay que observar que la adaptación de la estrategia de codificación del algoritmo actual, puede no ser tan ampliamente aplicada a la familia de los operadores genéticos que manipulan cadenas de números binarios. Entonces se tienen que crear operadores análogos dada la técnica de codificación que ha sido adoptada. Esto no siempre es fácil.

El operador *cruzamiento*, visto abstractamente, es un operador que combina partes de dos cromosomas de los padres que producen un nuevo hijo. La técnica de codificación adoptada debe tener soporte para operadores de este tipo, pero esto está muy apegado al entendimiento del problema, la técnica de codificación, la función de *cruzamiento* y para calcular cualquiera de los operadores. Si se puede crear el operador de cruzamiento basado en la técnica de codificación del problema, probablemente se puede observar otra técnica de codificación o se cambie el campo actual del algoritmo, por esto es el mecanismo del *cruzamiento* el que generalmente marca la pauta en el mejoramiento del rendimiento del algoritmo actual.

Esta situación es similar al operador de la *mutación*. Se tiene que decidir como usar las técnicas de codificación que han sido adaptadas al dominio del problema. Viéndolo abstractamente, el operador de la *mutación* es un operador que introduce variaciones en los *cromosomas*. Estas variaciones pueden ser locales o globales, pero este es un punto crítico para el algoritmo genético. Se tiene que combinar el conocimiento del problema, la técnica de codificación y la función de mutación en un algoritmo genético para desarrollar uno o muchos operadores de mutación para el dominio del problema. Si no hay operadores de mutación útiles para la técnica de codificación dada, la técnica de codificación puede ser cambiada o el algoritmo genético podría no estarse acercando a la solución del problema.

Finalmente, no se sabe si estas reglas funcionarán siempre para solucionar el problema en su dominio. Las heurísticas son propensas a fallar, pero conducen a las mejores soluciones con mayor frecuencia. Estas pueden ser muy usadas como las heurísticas en el conjunto de los operadores para poderlos accionar aleatoriamente como los otros operadores y éstas son guiadas en base al dominio para el proceso de búsqueda.

5.3 Optimizador Local SUMSL

En esta sección se describe el funcionamiento del optimizador local SUMSL. Este optimizador es incorporado como un operador adicional al Algoritmo Genético que se implementó.

Este optimizador minimiza la función objetivo general (no contraída), en este caso ECEPP/3, usando gradientes analíticos y aproximaciones Hessianas a través de actualizaciones de secantes.

Esta rutina interactúa con tres rutinas fundamentalmente las cuales son:

- SUMIT. Esta rutina que encuentran un vector enésimo $x*$ que minimiza la función objetivo general (sin restricciones) computada por CALF (por lo general la solución encontrada $x*$ un mínimo local en lugar de un mínimo global).
- CALCF. Esta rutina calcula la función con respecto al cromosoma $x*$.
- CALCG. Esta rutina calcula el gradiente de $x*$ con respecto a la función objetivo ECEPP3.

Si el optimizador local numérico SUMSL realiza 200 llamadas a "SUMIT" y no ha logrado que converja $x*$ al mínimo de la función, entonces se le asigna un valor por *default* y termina el cálculo.

El cromosoma $x*$ es denominado así porque la cadena x, que es el cromosoma, es pasado por referencia. Esto es que cualquier rutina a la que se le pasa por parámetro $x*$, ésta es modificada.

En la Figura 5.1 observamos que el cromosoma $x*$ es pasado primeramente por la rutina "SUMIT", esta como se mencionó anteriormente, intenta encontrar un vector tal que este minimice a $x*$ apoyándose del cálculo del valor de este cromosoma en la línea a3 por medio de la rutina "CALCF". Despues de que es calculado el gradiente por la rutina "CALG" regresa a "SUMIT", pero si nuevamente no ha convergido con el mínimo, entonces ahora pasa por la rutina "CALCG" para calcularle su gradiente. Todos estos pasos se repiten hasta que se encuentra la convergencia con el mínimo o se haya llamado 200 veces a la rutina "SUMIT".

```
Entrada: Cromosoma x*
Salida:   Cromosoma x*

    Begin
    ...
a1  Call SUMIT (x*)
a2  if (x* no converge con el mínimo y en el paso
       anterior no le ha  calculado su valor por "CALF")
a3     then Call CALF(x*)
a4          Go to a1
a5     else if (el cromosoma x* no converge con
            mínimo)
a6          then Call CALG(x*)
a7               GO to a1
a8          else Continua
    ...
    End
```

Figura 5.1: Pseudocódigo la parte fundamental del funcionamiento del optimizador local "SUMSL".

Capítulo 6

Implementación del Algoritmo Genético Híbrido al problema de plegado de proteínas

El algoritmo Genético Híbrido puede ser aplicado a cualquier tipo de problema relacionado con el plegado de proteínas, ya que lo único que se tiene que hacer es alimentar el sistema con los datos de la proteína que se quiera plegar, el numero de ángulos que la conforman, el número de generaciones, la probabilidad de cruzamiento y mutación. El sistema dará al final de su operación como resultado un archivo con el valor mínimo encontrado de calorías junto con la estructura tridimensional correspondiente y se dará a conocer para propósitos estadísticos el número de iteraciones, mutaciones y de cruzamientos que se hicieron para encontrarlo.

Este Algoritmo Genético Híbrido a diferencia de los comunes, en lugar de buscar el individuo que tenga el máximo peso con respecto a la función objetivo, éste los minimiza. Ya que lo que se quiere buscar es la estructura de la proteína que tenga la menor energía potencial.

El código fuente fue diseñado en el lenguaje de programación C de la computadora ORIGIN 2000 para aprovechar las cualidades que tiene esta como la arquitectura en paralelo. Las rutinas ECEPP/3 y SUMSL con sus respectivas subrutinas fueron traducidas del lenguaje original en que fueron programadas (FORTRAN) al lenguaje utilizado.

Parámetros	Intervalos
Probabilidad de cruzamiento:	[0.7,1.0]
Repetición de cruzamiento:	[6,10]
Probabilidad de mutación:	[0.3,0.5]
Repetición de mutación:	[1,2]

6.1 Predicción de la conformación de la Met-encéfalina y Leu-encéfalina

En esta sección se presentará la implementación de la partes principales del algoritmo genético híbrido como es la implementación de los individuos, la población inicial, la función objetivo. Para predecir la conformación de la Met-encéfalina y la Leu-encéfalina, para comprobar que tan rápido y preciso es este sistema para posteriormente aplicárlo a otras proteínas menos estudiadas y más grandes.

A continuación se describirán estas implantaciones y las adaptaciones que se realizaron al algoritmo genético híbrido.

Codificación. La estrategia de hibridación dada en el Capítulo 5 sugiere que reemplacemos en nuestro algoritmo la codificación binaria por otra. La Met-encéfalina y la Leu-encéfalina tienen 24 ángulos diedros dependientes que los conforman. Para cada conformación se tiene un conjunto de ángulos diedros que son codificados en un vector variable que sigue la secuencia primaria usando el orden interno convencional $[\phi_i, \psi_i, \omega_i, \chi_{i_1}, \cdots, \chi_{i_q}]_{i=1,k}$ donde k es el número de residuos de aminoácidos y i_q es el número de ángulos diedros laterales de la i-ésimo residuo. Valores reales con un decimal son asignados a estas variables.

Población Inicial. Los ángulos 4, 5, 6, 16, 17, 21, 22, 23 y 24 son ángulos diedros de la cadena lateral; el resto son ángulos diedros de la espina dorsal. Al final, los ángulos 3, 9, 12, 15 y 20 son ángulos diedros omega forzados a tomar aleatoriamente cualquier valor real entre $180° \pm 10°$ que corresponden a la forma **trans-planar** del enlace péptido.Los diedros remanentes son permitidos a explorar el espacio conformacional de valores reales en el intervalo $[+180°, -180°]$. Por el contrario, el vector de parámetros es siempre escogido de los siguientes intervalos determinados experimentalmente.

Función Conformacional. En este trabajo se utilizó la función de energía potencial **ECEPP/3** como la función objetivo a minimizar. La función

de potencial ECEPP/3 supone que la longitud del enlace covalente y los ángulos de enlace son constantes de tal manera que la energía conformacional sea dependiente de los ángulos diedros torsionales. Esta energía es tratada como una suma de energía electrostática no enlazada, puente de hidrógeno y contribuciones torsionales, el término de la finalización del ciclo es agregado si el péptido tiene enlaces de disulfuro intramolecular, más la energía conformacional de cada uno de las dos conformaciones del anillo pirolidino. La forma general de este potencial puede ser encontrada en otra parte de los ángulos 5, 7, 8 y 16.

Selección. Ya que la función que deseamos minimizar es la función de energía empírica ECEPP/3, establecemos que la probabilidad p_s de selección del vector de ángulos (o individuos) S_i con valor de energía ECEPP3, $E(S_i)$ es calculada con:

$$p_s(S_B) = \frac{(E_{\max} - E(S_B))}{p(E_{\max} - E_a)},$$

donde p es el tamaño de la población, E_a es el promedio de la energía y $E_{\max}$ es el máximo de energía de todos los individuos de la población considerada. Por comparación con la selección normal de la ruleta, el numerador puede ser considerado como la adaptación -*fitness*- individual. Más adelante, el proceso de *elitismo* se introducirá en el proceso de selección, esto es, que por cada vez que en la población sea reproducido el mejor de los individuos encontrados se introducirá en la nueva población.

Cruzamiento. La probabilidad de cruzamiento de cada individuo es directamente proporcional a su adaptación en el medio, en este caso, entre menos calorías tenga, habrá más posibilidades de ser escogidos para el cruzamiento. Cuando un par de individuos van a ser cruzados, el número de puntos de cruzamiento en el vector de parámetros son escogidos aleatoriamente entre 1 y el promedio de sus promedios de repeticiones.

Mutación. El operador mutación hace una modificación local de los cromosomas. Si la mutación es aceptada, el valor del un ángulo diedro del vector $[\phi_i, \psi_i, \omega_i, \chi_{i_1}, \ldots, \chi_{i_q}]_{i=1,k}$ es remplazado por otro valor escogido aleatoriamente en el intervalo asignado.

operador genético adicional. Aquí el optimizador local numérico SUMSL (explicado en el capítulo anterior) es el nuevo operador "genético" incorporado al algoritmo.

Condición de Término. En nuestro algortimo el usuario determina el número de generaciones que tendrá el algoritmo. Después de terminar estas iteraciones definidas, se presentará un informe de los resultados obtenidos.

6.2 Elementos que conforman el código del Algoritmo Genético Híbrido

Los principales elementos que integran este sistema se describirán a continuación:

- *Cromosoma*: Arreglo de ángulos que determinan la conformación de un individuo.
- *Fitness*: Variable que contiene el valor de la conFiguración del cromosoma, este valor es calculado por la función.
- *Individuo*: Estructura que contendría la información necesaria de cada ser que estará conformando a la población. Esta estructura consta de un cromosoma, un fitness y variables que guardarán la información de los padres con los que fue formado.
- *Población*: Arreglo definido de individuos, está definida por el valor de popsize.
- *bestfitness*: Individuo independiente a la población, éste conservará todas las características del individuo que tenga el fitness mínimo y se cambiará cuando encuentre a otro mejor.
- *lchrom*: Longitud de los cromosomas.
- *popsize*: Número de individuos en cada generación.

6.3 Programas

Los programas fundamentales para la implantación del Algoritmo Genético Híbrido son: El espacio de búsqueda X, la función objetivo, la generación de la nueva población, la selección de los individuos mejor adaptados, el proceso de cruzamiento, el proceso de mutación y el proceso de escalamiento de la población. A continuación se describirán en forma más detallada.

- **La función Objetivo.** Es el optimizador local SUMSL con respecto a la función de energía ECEPP/3. Su forma de funcionamiento consiste

en que una vez tenido el cromosoma resultante del proceso de cruzamiento y mutación. A éstos se les calcula su valor correspondiente por medio de este optimizador y almacenándolo en una variable contenida en la estructura del individuo (*fitness*).

- **La generación de la nueva población.** Para que se cree la primera generación se crean individuos. Cada uno de sus ángulos son generados dentro del intervalo establecido en los datos de entrada. Después se calcula a cada uno su *fitness* y posteriormente en la población recién creada se escoge al individuo que tenga al mínimo *fitness* como el *bestfitness*. También en este proceso se inicializan las variables necesarias para todo el proceso y una vez designado el *bestfitness*, se escala a la población.

- **La selección de los individuos mejor adaptados.** Este proceso utiliza el método de la ruleta. Esto es que el individuo que tenga el *fitness* más pequeño tiene más posibilidades de ser seleccionado. Recordemos que como lo que queremos es minimizar, entonces dentro del proceso de la asignación del *fitness* se hace una conversión de éste por medio de la siguiente regla:

individuo[j].fitness = Máximo - individuo[j].fitness + 1.00
Donde:
j indica el individuo j-ésimo.
Máximo es el máximo local de esta generación.

Entonces como resultado tendremos que el individuo que tenga el menor *fitness* será el nuevo máximo local y viceversa. Es importante señalar que el *bestfitness* se escogió antes de hacer esta conversión para evitar conversiones posteriores para obtenerlo.

- **Proceso de cruzamiento.** Una vez escogidos los dos individuos que se hayan seleccionado anteriormente según su probabilidad de cruzamiento se hace el intercambio de información o no. Si se realiza este proceso entonces se procede a escoger un índice i al azar entre la longitud del cromosoma hecho esto, los ángulos del primero al i-ésimo elemento del segundo individuo son remplazados con los del primero y los ángulos i-ésimo al último del primero son copiados con los del segundo.

- **Proceso de mutación.** Después de que dos individuos hayan sido cruzados, a cada uno de ellos se le aplica el siguiente proceso. Según la probabilidad de mutación se escoge un índice al azar del individuo y en este lugar se cambia el ángulo por otro elegido al azar, este número al azar está entre los intervalos establecidos para la proteína. Esta acción se realiza k veces, donde k es la longitud del cromosoma del individuo.
- **Escalamiento de la población.** Este proceso se llevará acabo una vez que se haya creado la población y se haya asignado el *fitness* a cada uno de ellos. Por medio del máximo, el mínimo y el promedio de esta se calcula el factor lineal de la distribución del *fitness* y se le aplica a cada uno de éstos. Esta nos servirá para que el *fitness* esté mejor distribuido, para que en el proceso de selección de la generación siguiente escoja a diferentes individuos y se desechen a los peores.

La manera en que se aplicará en los fitness será de la siguiente forma:

individuo[j].fitness = a * individuo[j].fitness + b
Donde: a y b son los factores lineales de la población.

Capítulo 7

Resultados y Conclusiones

En el presente capítulo se presentan los resultados obtenidos y son comparados con los obtenidos por otros autores utilizando diferentes métodos de optimización.

7.1 Resultados

El algoritmo genético híbrido fue usado para determinar la estructura de mínimo de energía para la *Met-encéfalina* y la *Leu-encéfalina*. Estas estructuras tienen un total de 24 ángulos diedros, incluidos los ángulos ω, con movimiento acorde a la relación anteriormente citada. El procedimiento de optimización considera valores reales para todas las variables tomadas desde un punto aleatorio en un intervalo de $[-180°, +180°]$ para todos los ángulos diedros. Excepto aquellos valores para los ángulos ω los cuales pertenecen al intervalo $-10° \leq \omega - 180° \leq +10°$. Con estas restricciones se redujo drásticamente la dimensión del espacio conformacional, pero conservando diferencias básicas entre las conformaciones.

El algoritmo genético híbrido compilado con la opción -O2 fue ejecutado empleando un procesador en la máquina SGI/CRAY ORIGIN 2000, existe una versión de este algoritmo que aprovecha la arquitectura en paralelo de esta máquina para que en un mismo tiempo de cómputo realice de uno hasta el límite de procesadores disponibles el programa. Con esta versión se obtienen resultados diferentes.

En el algoritmo genético híbrido tanto el número de ejecuciones (generaciones), el número de individuos, la probabilidad de cruzamiento y de mutación son definidas por el usuario. Además el usuario también puede definir el numero de ángulos que puede tener un individuo, para este caso en particular se definieron individuos con 24 ángulos.

Se realizaron numerosos experimentos con la proteína Met-encéfalina para determinar la mejor probabilidad de cruzamiento y de mutación, así como del número de individuos apropiado. Con esto se concluyó que el algoritmo genético híbrido para el caso de la Met-encéfalina daba mejores resultado con 20 individuos, con probabilidad de 0.6 de cruzamiento y 0.3 de mutación.

Una vez teniendo la información necesaria se procedió a realizar 100 corridas del algoritmo, cada una con 20 generaciones. Todo este proceso se realizó con un procesador. Cada simulación para la proteína Met-encéfalina tomó cerca de 107 segundos de tiempo total de CPU y aproximadamente 51,000 evaluaciones de la función de energía ECEPP/3, para la proteína Leu-encéfalina tomó en promedio 109 segundos y 50,000 evaluaciones. Cabe recordar que el operador genético adicional es el optimizador local SUMSL, el cual para cada individuo puede usar hasta 200 evaluaciones de la función de energía ECEPP/3. Es claro que teniendo la ventaja de una máquina en paralelo se puede tener, en caso de la ORIGIN 2000 con 32 procesadores, 32 resultados diferentes en relativamente un mismo tiempo de proceso. Un aspecto atractivo de este algoritmo, es que se obtuvo una gran precisión en el resultado en poco tiempo, todo esto sin realizar ninguna optimización o restricción en los ángulos de las proteínas.

En la Tabla 6.1 se comparan los resultados obtenidos, con los mejores mínimos de energía de conformaciones de la Met-encéfalina obtenidos por otros autores usando diferentes métodos de optimización. Los primeros tres resultados muestran los ángulos diedros de la estructura mínima con mejor mínimo de energía para Met-encéfalina con los métodos conocidos como MCM, αBB y CSA, respectivamente. El cuarto valor muestra los ángulos diedros obtenidos con BT. El quinto valor muestra los ángulos diedros obtenidos después de realizar una optimización local con SUMSL, usando como datos de entrada los obtenidos por BT. El resultado número seis fue obtenido en 1992 con el algoritmo MU. El resultado siete es el algoritmo genético híbrido. Todos estos valores de energías se obtuvieron a partir de un solo punto de cálculo con la más reciente versión de ECEPP/3, esta acción normaliza todas las conformaciones a los mismos parámetros obteniendo un valor de energía común. Por ejemplo, Nayeem reporta un valor de -12.396 kcal/mol,

igual Androulakis y Lee, pero reportan un valor de energía -11.77 kcal/mol para este mínimo global de energía de la estructura. Estos valores son claramente una consecuencia derivada de ligeras diferencias en los parámetros de ECEPP usados en las citas anteriores.

Tabla 6.1 Ángulos diedros de las 7 conformaciones con mejores mínimos de energía para la Met-encéfalina

		A0	A1†	A2†	BT	BT*	TA	AGH
Tyr	ϕ	-86.1	-83.5	-83.5	-80.0	-94.0	-94.0	-83.0
	ψ	156.2	155.8	155.8	148.0	155.7	155.0	155.8
	ω	-176.9	-177.1	-177.2	-176.0	-177.1	-175.0	-177.2
	χ_1	-172.6	-173.2	-173.2	-172.0	-173.2	-172.0	-173.2
	χ_2	-101.3	-100.5	79.4	-98.0	-100.7	-104.0	-100.6
	χ_3	14.1	13.6	-166.4	23.0	13.7	27.0	13.6
Gly	ϕ	-154.5	-154.3	-154.3	-151.0	-154.2	-154.0	-154.4
	ψ	83.7	86.0	86.0	86.0	85.8	74.0	86.0
	ω	168.6	168.5	168.5	173.0	168.5	170.0	168.5
Gly	ϕ	83.7	82.9	82.9	82.0	82.9	83.0	83.0
	ψ	-73.9	-75.1	-72.5	-79.0	-75.0	-70.0	-75.1
	ω	-170.0	-169.9	-170.0	-171.0	-169.9	-170.0	-169.9
Phe	ϕ	-137.0	-136.9	-136.9	-137.0	-136.8	-136.0	-136.9
	ψ	19.3	19.1	19.1	23.0	19.1	18.0	19.1
	ω	-174.1	-174.1	-174.1	-176.0	-174.1	-174.0	-174.1
	χ_1	58.8	58.8	58.9	58.0	58.8	59.0	58.9
	χ_2	-85.4	-85.5	94.6	95.0	94.5	-86.0	-85.4
Met	ϕ	-163.6	-163.5	-163.5	-163.0	-163.4	-163.0	196.5
	ψ	160.5	160.9	161.2	172.0	160.8	166.0	-199.2
	ω	-179.7	-179.8	-179.8	180.0	-179.8	178.0	-179.8
	χ_1	52.8	52.9	52.9	51.0	52.8	52.0	52.9
	χ_2	175.3	175.3	175.3	176.0	175.3	174.0	-184.7
	χ_3	-179.8	-179.8	-179.8	177.0	-179.8	-175.0	180.2
	χ_4	61.4	61.4	-58.6	-60.0	-58.6	61.0	-58.6
energía Kcal/mol		-12.389	-12.389	-12.389	-11.246	-12.389	-10.672	-12.389

A0.- A. Nayeem, J.Vila y H.A. Scheraga, J.Compt Chem., 12,594(1991).

A1.- I.P.Androulakis, C.D. Maranas y C.A. Floudas, J.global Opt., 11, 1 (1997).

A2.- J.Lee, H.A. Scheraga y S.Rackovsky, J.Compt. Chem., 18, 1222(1997).

BT.- L.B.Morales, R.Garduño Juárez y Riveros Castro F. (1999).

BT*.- Geometría Optimizada desde BT con SUMSL[2].

TA.- L.B.Morales, R.Garduño Juárez y D.Romero, J.Fiomol. Struct.Dynamics,9, 951 (1992).

AGH.- Esta tesis (Algoritmos Genéticos Híbridos).

(†).—Los valores mostrados de energía son calculados con la versión más reciente de ECEPP/3. Originalmente se reporta una energía de -11.707 Kcal/mol debido a ligeras diferencias en los parámetros de ECEPP usados en la nota computacional.

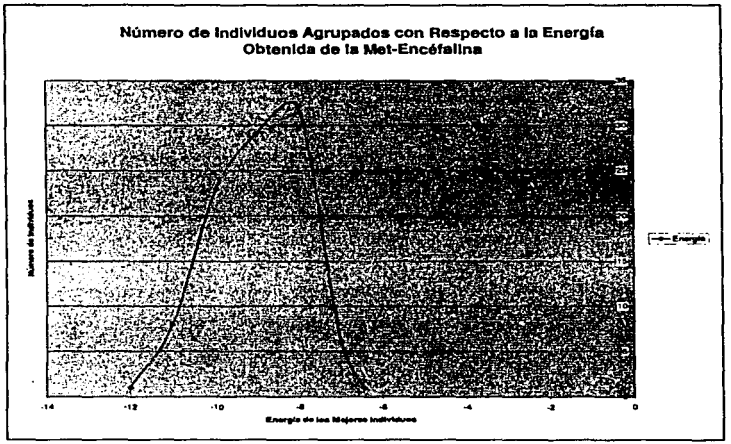

Figura 7.1: Agrupamiento de los Mejores individuos obtenidos de 20 corridas con respecto a su energía obtenida.

Aquí se presentan las gráficas de comportamiento del Algoritmo Genético Híbrido para obtener la energía mínima de la proteína Met-encéfalina. La gráfica de la Figura 7.1 corresponde a una probabilidad Gauseana. En la Figura 7.2 se muestra la estructura tridimencional resultante del mejor mínimo encontrado de la Met-encéfalina.

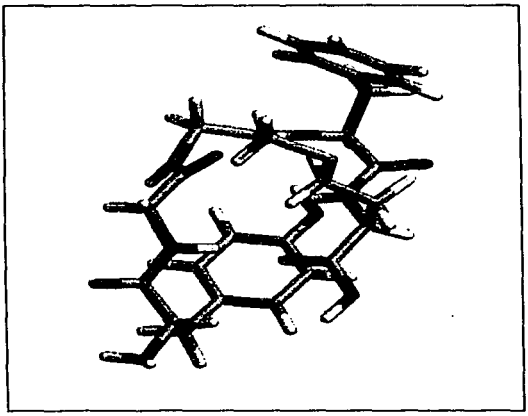

Figura 7.2: Gráfica de la proteína Met-encéfalina, obtenida por medio del Algoritmo Genético Híbrido.

ESTA TESIS NO SALE
DE LA BIBLIOTECA

Tabla 6.2 Ángulos diedros de las 2 conformaciones con mejores mínimos de energía para la Leu-encéfalina

		SA	AGH
Tyr	ϕ	-88	-168.4
	ψ	153	-35.5
	ω	-170	180.7
	χ_1	-175	189.0
	χ_2	-96	56.3
	χ_3	44	-158.6
Gly	ϕ	-155	97.9
	ψ	73	43.1
	ω	170	168.5
Gly	ϕ	73	69.8
	ψ	-93	-91.3
	ω	180	182.1
Phe	ϕ	-90	-90.9
	ψ	-18	-92.4
	ω	180	-173.8
	χ_1	-178	178.9
	χ_2	76	-106.2
Leu	ϕ	-92	-147.9
	ψ	108	124.8
	ω	180	188.9
	χ_1	-158	177.6
	χ_2	165	-297.3
	χ_3	-61	52.5
	χ_4	66	-60.8
energía Kcal/mol		-11.10	-10.77

SA.- L.B.Morales, R.Garduño Juárez y D.Romero, J.Biomol. Struct.Dynamics, 8, (1991), 721 (ECEPP/2).

AGH.- Esta tesis (Algoritmos Genéticos Híbridos).

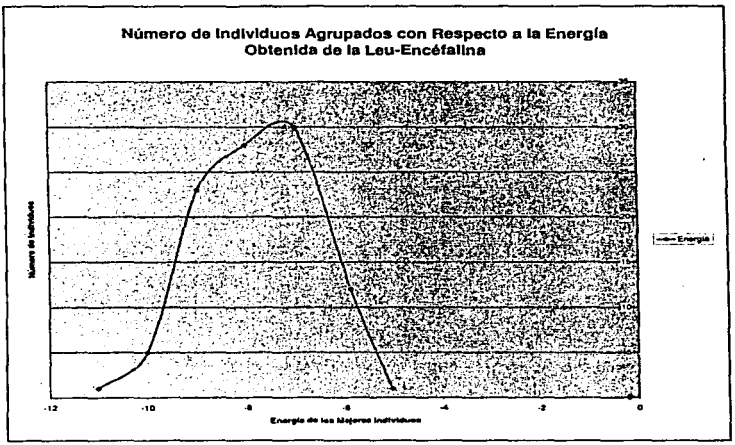

Figura 7.3: Agrupamiento de los Mejores individuos obtenidos de 20 corridas con respecto a su energía obtenida.

Como en la Figura 7.1 de la met-encéfalina , la gráfica de la Figura 7.3 de la proteína leu-encéfalina también corresponde a una probabilidad Gausseana. En la gráfica de la Figura 7.4 se presenta la estructura tridimensional del mejor mínimo encontrado de la leu-encéfalina.

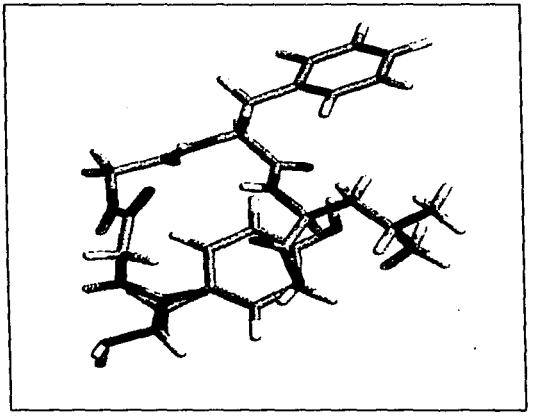

Figura 7.4: Gráfica de la proteína Leu-encéfalina, obtenida por medio del Algoritmo Genético Híbrido.

Tabla 6.3 Resultados Computacionales para Met-encéfalina

Método	N_{var}	CPU	Computadora
Minimización Montecarlo [19]	19	2-3 Hrs.	IBM 3090
	24	10 Hrs.	IBM 3090
Manejador Electrostático Montecarlo	19	2-3 Hrs.	IBM 3090
	24	10 Hrs.	IBM 3090
Ecuación de difusión [17]	19	20 min.	IBM 3090
Self-Consistent Multitorsional Field	10	100 min.	IBM 3090
Recosido Simulado Multicanonical [8]	19	6.0 hrs.	IBM RS600
Recosido Simulado [25]	24	2.5 hrs.	Apollo DN1000
Umbrales Recosido Simulado [22]	24	1.5 hrs.	Apollo DN1000
Recosido Simulado [25] con Minimizador Montecarlo[19]	24	2 hrs.	CRAY X-MP
Recosido Simulado [25] con Minimizador Montecarlo [19]	24	1.2 hrs.	CRAY-2S 4 procesadores
Minimizador Montecarlo [19] vs Recosido Simulado [25]	24	1.5-4 hrs.	IBM 30090
αBB [1]	24	1.3 Hrs.	HP-730
Búsqueda Tabú [26]	24	34.3 min.	ORIGIN2000 con 1 procesador
	24	1.19 min.	ORIGIN2000 con 32 procesadores
Algoritmo Genético Híbrido	24	1.7 min.	ORIGIN2000 con 1 procesador

7.2 Conclusiones

Este trabajo ha demostrado los beneficios y bondades de los Algoritmos Genéticos. A diferencia de otros heurísticos, como la Búsqueda Tabú, que únicamente trabaja en la mejora de un solo conformero, los Algorítmos Genéticos trabajan en un grupo de conformeros que intercambian información entre sí. La diversidad así generada permite que la hipersuperficie de energía potencial sea explorada rápidamente, seleccionando regiones donde los conformeros estéricamente permitidos existen y por lo tanto evita la búsqueda en regiones donde existen conflictos estéricos. En otras palabras, la búsqueda en regiones no permitidas es una pérdida de tiempo en muchos heurísticos, con los Algoritmos Genéticos existe la posibilidad de ahorrar tiempo de cálculo y se abre la posibilidad de mejorar la búsqueda con implantaciones de técnicas de optimización local. Es necesario destacar que para las moléculas prueba que aquí se han presentado, los Algoritmos Genéticos producen la estructura de energía más baja en un tiempo de CPU mucho menor que otros heurísticos. En la Tabla 6.2 se comparan los diferentes métodos que se han empleado para la localización de la misma estructura de más baja energía, así como el número de evaluaciones que estos han requerido para esta tarea. No se menciona el tiempo de CPU empleado ya que muchas de estas técnicas se han desarrollado para diferentes máquinas con procesadores de diferentes velocidades. Es necesario que a futuro se realice un trabajo de refinamiento sobre el procedimiento del Algoritmo Genético aquí presentado con el propósito de reducir aún más el tiempo de corrida. Una opción obvia es la de expandir su acción a un número mayor de procesadores, y para ello es necesario implementar este código para trabajo en paralelo. Como podemos observar, para el caso de un solo procesador, la estructura de más baja energía para la Met-encéfalina fue encontrada en 1.7 minutos en promedio. Una arquitectura como la de la ORIGIN2000 con 36 procesadores daría la oportunidad ideal de probar que estos tiempos pueden reducirse aún más. El hecho de que la estructura de Met-encéfalina con la energía más baja obtenida con nuestro Algoritmo Genético es la misma que la reportada por otros autores quienes han empleado otras técnicas de búsqueda heurística, la más reciente reportada por Nayeem [27] y Androulakis [1], da suficiente evidencia para decir que esta estructura pertenece a la del mínimo global de energía para esta molécula. El trabajo aquí reportado ha puesto las bases para futuras investigaciones en el área de predicción de estructura terciaria en péptidos y proteínas. El permitir que cada individuo sea optimizado parcialmente antes

de decidir si éste es conservado o no dentro de la población que continuará en la siguiente generación nos permite realizar un muestreo rápido de la hipersuperficie de energía potencial de estas moléculas. El código que se ha escrito se ha hecho teniendo en mente la posibilidad de expandir su uso en máquinas en paralelo.

Bibliografía

[1] I.P. Androulakis, C.D. Maranas and C.A. Floudas, Journal global Opt., 11,1 (1997).

[2] B. Berger and T. Leighton, J. Comput. Biol., 5(1), 27 (1998).

[3] B.R. Brooks, R.E. Bruccoleri, B.D. Olafson, D.J. States, S. Swaminathan, M. Karplus, CHARMM: a program for macromolecular energy, minimization, and dynamics calculations, J. Comput. Chem. 4 (1983) 187-217.

[4] D.A. Case, D.A., Pearlman, J. W. Caldwell, T.E. Cheatham III, W.S. Ross, C.L. Simmerling, T.A. Darden, K.M. Mertz, R.V. Staton, A.L. Cheng, J.J. Vincent, M. Crowley, D.M. Fergunson, R.J. Radmer, G.L. Seibel, U.C. Singh, P.K. Weiner, P.A. Kollman, AMBER 5, University of California, San Francisco, CA, 1997.

[5] M.R. Garey and D.S. Johnson, (1979) Computers and intractability. Freeman and Co. NY.

[6] L. Davis (1991). Handbook of Genetic Algorithms. Publish by Van Nostrand Reinhold.

[7] D.E. Goldberg., (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley (Eds.)

[8] Hansmann, U. H. E.; Okamoto, Y. J Comp Chem 1993, 14, 1333.

[9] W. E. Hart and S. Istrail, J. Comput. Biol., 4(1), 1 (1997).

[10] J.H. Holland, Genetic Algoritms, Scientific American, July, 1992.

[11] J.H. Holland, Adaptation in Natural and Artificial Systems: The University of Michigan Press, Ann Arbor, MI; 1975

[12] C. N. Fiechter, Discrete Appl. Math., 51, 243 (1994)

[13] David M. Gay; SUMSL; The National Science Foundation,1982

[14] R. Garduño-Juárez, L.B. Morales, P. Flores-Pérez, R. Mexicana de Física 46 Suplemento 2, 135 (2000).

[15] InsightII/Discover'95, Biosym Technologies, Inc., 9685 Scraton Road, San Diego, CA 92121-2777, 1995.

[16] R.M. Karp (1972). Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher (Eds.) Complexity of Computer Computations, Plenum Press, NY.

[17] Kostrowicki, J.; Cherayil, B. J.; Scheraga, H. A. J Phys Chem 1991, 95, 4113. Kostrowicki, J.; Cherayil, B. J.; Scheraga, H. A.J Phys Chem 1992, 96, 7442.

[18] J. Lee, H. A. Scheraga and S. Rackovsky, J. Compt. Chem., 18, 1222 (1997).

[19] Li, Z; Scheraga, H. A. Proc Nalt Acad Sci USA 1987, 84, 611. Li, Z.; Scheraga, H. A. J Mol Struct (Theochem) 1988, 179, 333.

[20] F. Mohamadi, N.G.J. Richards, W.C. Guida, R. Liskamp, C. Caufiled, G. Cheng, T. Hendrickson, W. C. Still, MACROMODEL, J. Comput. Chem. 11 (1990) 440-467.

[21] L. B. Morales, R. Garduño-Juárez and D. Romero, J. Biomol. Struct. Dyn., 8, 721 (1991).

[22] L. B. Morales, R. Garduño-Juárez and D. Romero, J. Biomol. Struct. Dyn., 9, 951 (1992).

[23] L.B. Morales, R. Garduño-Juárez, J.M. Aguilar-Alvarado and F.J. Riveros-Castro, J. of Computational Chemistry, 21(2), 147 (2000).

[24] L.B. Morales, R. Garduño, P. Flores-Pérez, Algoritmos Genéticos con Memorias Conformacionales.

[25] Morales, L. B.; Garduño-Juárez; Romero, D. J Biomol Struct Dynam 1991, 8, 721.G. Nemethy, K. D. Gibson, K. A. Palmer, C. N. Yoon, G.

[26] L.B.Morales, R.Garduño Juárez y Riveros Castro F. (1999).

[27] Nayeem, G. A.; Vila, J.; Scheraga, H. A. J Comput Chem 1991, 12, 594.

[28] Olszewski, K. A. ; Piela, L.; Scheraga, H. A. J Phys Chem 1992, 96, 4672.

[29] Paterlini, A. Zagari, S. Rumsey and H. A. Scheraga, J. Phys. Chem., 96, 6472 (1992).

[30] A. Scheraga, Harold A; ECEPP (V.1, V.2, V.3) User Guide; Departament of Chemistry; Cornell University Ithaca; 1993