UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

POSGRADO EN CIENCIA E INGENIERIA DE LA COMPUTACION

SISTEMA EN PARALELO PARA CALCULAR LA ESTRUCTURA TRIDIMENSIONAL DE LAS PROTEINAS

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERIA (COMPUTACION)

P R E S E N T A :

LUIS GERMAN PEREZ HERNANDEZ

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

MEXICO, D. F., 2005

Sistema en Paralelo para Calcular la Estructura Tridimensional de las Proteínas

Luis Germán Pérez Hernández

2005

Agradecimientos

Agradezco ha mis directores de tesis, los doctores Luis B. Morales Mendoza y Ramón Garduño Juárez, que me ofrecieron la oportunidad de trabajar con ellos, además del apoyo desinteresado, en todos los aspectos, que me brindaron a través de todos mis estudios. Asimismo a los sinodales Dr. Héctor Benítez Pérez, Dr. Julio Solano Gonzáles y al Mat. Carlos Velarde Velásquez que de forma profesional cedieron parte de su tiempo y aceptaron revisar esta tesis.

De manera muy especial agradezco a mi madre, la Sra. María del Consuelo Hernández Espinosa, el apoyo que me dio para el logro de esta meta.

El presenta trabajo ha sido posible a la generosa donación de tiempo de supercómputo por la Dirección General de Asuntos del Personal Académico de la UNAM. Se agradecen los apoyos otorgados por los proyectos PAPIIT-UNAM IN118903-2 Y PAPIIT-UNAM IN107701 del Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica de la UNAM, así como también al proyecto CONACYT No. 41652-A.

ÍNDICE

RESUMEN

Este trabajo presenta un algoritmo que puede proporcionar una solución "de manera eficiente" al problema denominado plegamiento de proteínas. Este problema está basado en la imposibilidad matemática de explorar totalmente el espacio conformacional de estas moléculas, y su solución 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 en paralelo en donde en cada uno de los procesadores se distribuye un Algoritmos Genéticos Híbridos acoplados con un analizador de clusters. Nuestros algoritmos genéticos tienen el optimizador local numérico SUMSL (que se explicara mas adelante) como un operador genético adicional. En si, el algoritmo es una extensión de los Algoritmos Genéticos tradicionales, que mejoran a los individuos en cada generación para así tener mejores resultados en menor tiempo. Una vez obtenidos todos los resultados de los procesadores, los mejores individuos de cada uno de estos se les aplica un algoritmo de análisis de "cluster" de datos. Para que con esta nueva información se reajusten los espacios conformacionales de los ángulos diedros de la proteína, permitiendo con esto reducir de manera "inteligente" el espacio de búsqueda. Estos intervalos contienen los subespacios donde se ubican los individuos con las mejores propiedades y así realizar una búsqueda más eficiente por cada iteración. Las proteínas de prueba fueron la *Met-encéfalina* y la *Leu-encéfalina*, cuya estructura tridimensional ya es conocida, y que se han usado como banco de pruebas para otros métodos heurísticos de optimización. Nuestro algoritmo predice exitosamente la estructura tridimensional para estas proteínas reportadas por Lee et al. (1997), Androulakis et al. (1997), y Morales et al (1998).

Este procedimiento se diseño pensado en la robustez y la portabilidad del mismo, ya que nuestra meta principal es el que éste se pueda ajustar en cualquier ambiente en paralelo, con el número de procesadores que se tenga disponible.

Capítulo 1

INTRODUCCIÓN

El plegamiento de proteínas ha sido uno de los eventos más estudiados durante el Siglo XX en el campo de la Bioquímica y la Biofísica, y se espera que durante una gran parte del Siglo XXI todavía sea materia de estudio. Matemáticamente, este evento es mejor conocido como el problema de los múltiples mínimos, o como ***el problema del plegado de las proteínas*** [4]. Este problema está 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 * ≈20 átomos por residuo)*3-6 = 5994 grados de libertad. Un sistema de ecuaciones con tal número de variables independientes es imposible de resolver hoy en día. Si para esta "proteína" no hubiera restricciones en su conformación y solo su estructura primaria fuera dada, el número de confórmeros para una proteína promedio (~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 [34].

Las técnicas de muestreo conformacional pueden ser agrupadas en métodos determinístas y métodos estocásticos. Los primeros incluyen cualquier método por el cual la generación y evaluación de la conformación de una molécula dada están determinados por los anteriores. Los métodos estocásticos incluyen a los ampliamente usados métodos de Monte Carlo, donde la conformación se genera al tomar aleatoriamente los valores de cualquier parámetro que define a la conformación. Por lo tanto, la hipersuperficie conformacional [34] 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; este proceso es llamado computación evolutiva. Por lo tanto, la computación evolutiva es esencialmente heurística, i.e. 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

"calculo razonable".

El problema del plegado de proteínas tiene una gran repercusión en el entendimiento de la vida, ya que las proteínas se pueden encontrar tanto dentro como fuera de las células y cada uno de los diversos tipos de proteínas llevan una labor específica.

En la célula las moléculas de proteínas se pliegan para llevar a cabo sus funciones biológicas. 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 especificas a pesar de la gran cantidad de posibles conformaciones. Como las proteínas están formadas de miles de átomos que interaccionan entre si 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 seudo poténciales obtenidos a partir de estructuras conocidas en los bancos de datos. Aún con estos potenciales mínimos, encontrar la estructura con una energía más baja en 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 proceso pertenece a los problemas *NP*-completos [33]. El 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 está muy poco estudiada, y segundo, el espacio de sus posibles conformaciones es muy grande y complejo. Mientras que analizar el primer motivo 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 Monte Carlo, 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 una solución al problema del plegado de proteínas y lograr simular sus estructuras tridimensionales, se han usado diversos métodos [17, 18]. Como este problema

en particular presenta una gran cantidad de elementos combinatorios, se requieren técnicas muy elaboradas para su solución. En el pasado se han empleado varias técnicas heurísticas con buenos resultados, como lo son la Búsqueda Tabú [18], el Recocido Simulado [23-16] y el Método de Umbrales [17] y Algoritmos Genéticos [26] entre otros. Sin embargo, a pesar de que se han tenido avances en la reducción del tiempo en que se obtiene un buen resultado y se dan buenas aproximaciones, todavía se puede mejorar este proceso utilizando otras técnicas alternativas.

En este trabajo se reporta el uso de un sistema en paralelo utilizando MPI en el cual a cada uno de los procesadores se les asigna un procedimiento Híbrido de un algoritmo genético acoplado con el optimizador local SUMLS, y posteriormente particionado con el algoritmo PAM [32] para recalcular la holgura de los ángulos, para así dar otra posible solución a este problema. 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, la portabilidad y la precisión en los resultados obtenidos. Para esto se realizaron pruebas con las moléculas de los neuropéptidos *Met-encéfalina* [7] y *Leu-encéfalina* [12] para los cuales se conocen sus estructuras biológicamente activas, y que han sido empleadas como "benchmaks" por otros trabajos. Emplearemos esta comparación para medir la eficiencia y la exactitud de este método. Posteriormente se realizarán experimentos con otras proteínas menos estudiadas, como la *Thymo-encéfalina.*

Actualmente con los métodos de ingeniería genética se puede producir proteínas con una secuencia de aminoácidos. Si se conociera la estructura final a la que se puede plegar esta proteína, el tener esta información permitiría predecir las propiedades químicas y biológicas de esta, simplificando la labor de interpretación de la información de la proteína. Una de sus mayores aplicaciones sería dentro del proyecto del genoma humano, ya que permitiría entender el mecanismo de algunas enfermedades infecciosas y hereditarias ligadas al mal plegado de las proteínas, permitiendo el diseño de 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 con el propósito de describir la metodología para resolverlo.

Se dará a conocer la terminología que se usara durante la tesis. 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 [22]. 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 interaccionan entre sí, así como con otros miles de moléculas de agua, no es posible calcular su función de energía libre por primeros principios; por lo tanto, es necesario adaptar seudo poténciales obtenidos a partir de proteínas cuya estructura es conocida 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. Algorítmicamente este problema pertenece a la clase de *NP*-Completos [3].

Termodinámicamente hablando, el plegado de una proteína puede visualizarse como un embudo energético [22]. 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. En esta tesis 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 [8,10] que corresponda al mínimo local más bajo de una función matemática apropiada que describa la energía potencial del sistema. Las simulaciones por computadora es la obtención de 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 a través 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 a través 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 en 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 peptídicos [7]. 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 los 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 [1], MM2 [34], ECEPP/2 [1], CHARMM [6], DISCOVER [30], AMBER [9], GROMOS87, MM3 [34] y ECEPP/3 [1]. En este trabajo se utiliza el campo de fuerza ECEPP/3, 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, 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 sobreponer 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) [36], Método de Umbrales (MU) [35], Monte Carlo con Minimización (MCLM) [37], Ensamble Multicanónico (EM) [20] y los Algoritmos Genéticos tradicionales (AG) [21]. Entre los métodos deterministas se encuentran la Dinámica Molecular (DM), el Método de la Ecuación de Difusión (MED) [31], 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 dimensión del problema. Muy recientemente un método llamado αBB [2] ha reclamado tener la 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 tridimensional 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 han 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 [27-28] 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:

$$U = U_{elec} + U_{nb} + U_{nonb} + U_{tor} + U_{loop} + U_{s-s'}$$

Donde:

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

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

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

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

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, 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

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

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.

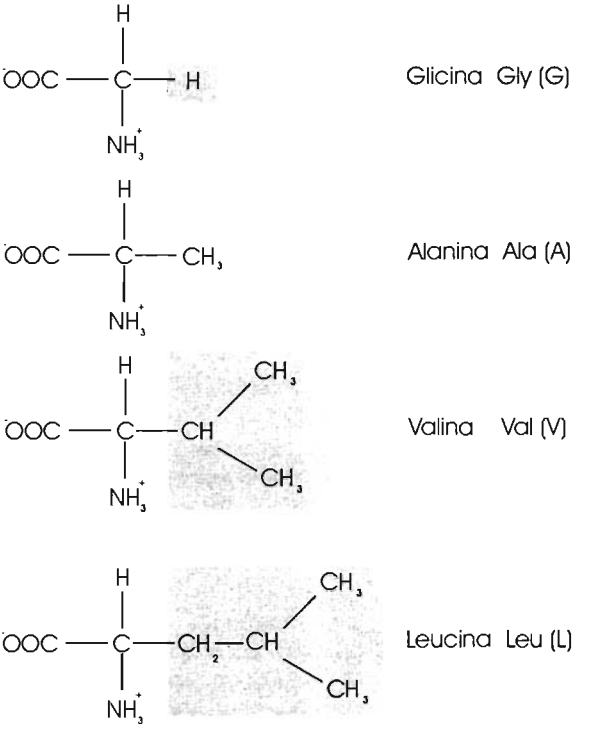

Figura 2.1: Ejemplos de estructuras de Aminoácidos.

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

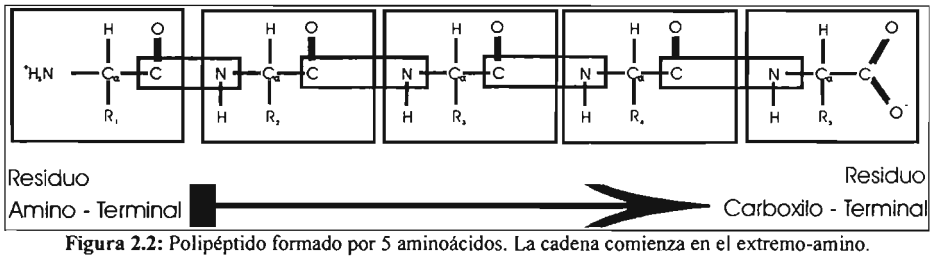

Figura 2.2: Polipéptido formado por 5 aminoácidos. La cadena comienza en el extremo-amino.

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 tiene una dirección porque sus elementos constructores tienen extremos diferentes. Por convención se toma al extremo el 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 plana y 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.

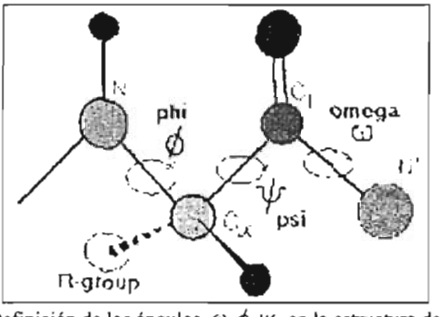

Figura 2.3: Definición de los ángulos ω, ϕ, ψ en la estructura de la proteína.

Existe un enorme grado de libertad torsional alrededor de los enlaces de *carbono α* y el carbono carboxilico (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 por los 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 un grafo con nodos correspondientes a los átomos y con arcos correspondientes a los enlaces atómicos entre los átomos. En este grafo 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 φi, ψi y ωi los ángulos diedros de la columna vertebral correspondiendo a la *i*-ésimo residuo aminoácido (ver Figura 2.3 y 2.4). For i = 1,..., k, sean $\chi_{i1},...,\chi_{iq}$ los ángulos diedros en las cadenas laterales del *i*-ésimo residuo.

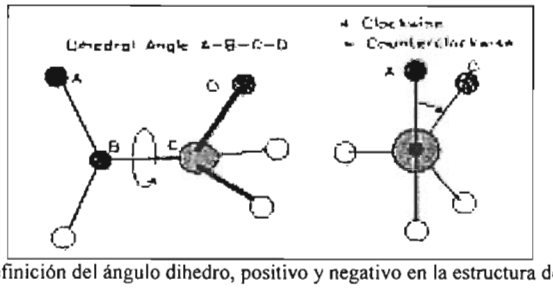

Figura 2.4: Definición del ángulo dihedro, positivo y negativo en la estructura de la proteína.

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:

Minimizar $U(\phi_1,\psi_i,\omega_1,\chi_{i1},\ldots,\chi_{i1},\ldots,\phi_k,\psi_k,\omega_k,\chi_{k1},\ldots,\chi_{kq})$

Sujeta a las siguientes restricciones experimentales

$-180^o \le \phi_i,\psi_i \le 180^o$ i = 1,..., k

$-10^o \le (\omega_i - 180^o) \le 10^o$ i = 1,..., k

$-180^o \le \chi_{ij} \le 180^o$ i = 1,..., k j = 1,..., q

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 solo de los ángulos diedros torsionales. Esta energía es tratada como una suma de energía electrostática entre átomos no enlazados, puentes de hidrógeno y contribuciones torsionales, el término de la formación de un ciclo es agregado si el péptido tiene enlaces de disulfuro intramoleculares, más la energía conformacional de cada una de las dos conformaciones del anillo pirolidino.

Capítulo 3

OPTIMIZACIÓN

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

3.1 PROBLEMAS DE OPTIMIZACIÓN

Varios investigadores han 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 $f(x)$ (1)

sujeta ha:

$g_i(x) \geq b_i$ i = 1,..., m (2)

$h_i(x) \geq b_j$ j = 1,..., n (3)

Donde x es un vector de variables de control y $f(x)$, $g_i(x)$ y $h_i(x)$ son funciones generales.

En los problemas de optimización debe considerarse la diferencia entre el mínimo global o un mínimo local de la función real f(x), donde x es un vector de n variables reales. Un mínimo global se refiere a la existencia de un x* que minimice f para todos los vectores posibles x. Por mínimo local entendemos un vector, x*, tal que $f(x^*) \leq f(x)$ para todo x cercano a x*. La posibilidad de acceder a un mínimo global depende de las características del problema.

Tal como se desprende de las expresiones (1) a (3), los problemas de optimización constan de tres elementos básicos:

Una función objetivo que querríamos maximizar o minimizar. Así, en el caso de la fabricación de hilos, trataríamos de minimizar la desviación entre los valores observados y los estándares establecidos.

Un conjunto de variables o incógnitas que determinan el valor del objetivo. Las

variables pueden referirse a las cantidades de diferentes recursos o al tiempo empleados en una actividad. En la búsqueda de datos experimentales apropiados, las incógnitas son los parámetros que definen el modelo.

Un conjunto de vínculos que delimitan los valores que pueden tomar las variables.

En resumen, un problema de optimización es: encontrar valores de las variables que minimicen o maximicen la función objetivo mientras satisfacen los vínculos.

Sin embargo, en la estructura de un problema de optimización, hay excepciones que amplían o eliminan algunos de estos elementos.

Aunque casi todos los problemas de optimización tiene una función objetivo única, hay dos excepciones importantes:

No hay función objetivo. En algunos casos, por ejemplo en el diseño de circuitos integrados, la meta es encontrar un conjunto de variables que satisfagan los vínculos del modelo. No se trata de optimizar porque no hay razón para definir una función objetivo. Este tipo de problemas reciben el nombre de problemas de factibilidad.

Hay múltiples funciones objetivo. Es bastante común que se quieran optimizar varios objetivos al mismo tiempo, y también es usual que los distintos objetivos no sean compatibles. Las variables que optimizan un objetivo pueden estar muy lejos del óptimo de los otros objetivos. En la práctica, los problemas con múltiples objetivos, se reformulan construyendo un solo objetivo con una combinación pesada de los distintos objetivos o se reemplazan algunos objetivos por vínculos. El problema es conocido como optimización multiobjetivo.

También se encuentran problemas no típicos en cuanto a los vínculos, siendo la optimización no-restringida un campo importante, con algoritmos y software disponible para su resolución.

Los problemas de optimización se dividen, por sus características y por los métodos empleados en su resolución, en dos clases: problemas de optimización continua, en la que todas las variables pueden tomar valores en intervalos de la recta real, y problemas de optimización combinatoria, donde se requiere que algunas o todas las variables estén restringidas a tener valores enteros [31] [32].

3.2 OPTIMIZACIÓN CONTINUA

El campo de la *optimización continua* se divide también en dos áreas, la de la *optimización restringida* y la *optimización no-restringida.*

En los problemas de *optimización no-restringida* se trata de encontrar el mínimo local de una función real $f(x)$.

En la *optimización global* se trata de encontrar un x* que minimice f para todos los vectores posibles x. Sin embargo, para muchas aplicaciones, los mínimos locales resultan suficientemente buenos, en particular cuando se dispone de un punto de partida adecuado [33].

Muchos de los algoritmos que se utilizan para resolver problemas de optimización no-restringida, se basan en el método de Newton [34]. El método de Newton requiere el cálculo del *vector gradiente*

$$\nabla f(x) = (\partial_i f(x))$$

y de la *matriz Hessiano*

$$\nabla^2 f(x) = (\partial_j \partial_i f(x))$$

El método de Newton genera un modelo cuadrático de la función objetivo en torno al valor, x_k, de la k-ésima iteración. El modelo está definido por la función

$$q_k(s) = f(x_k) + \nabla f(x_k)^T s + \tfrac{1}{2}\, s^T \nabla^2 f(x_k) s$$

En el método de Newton básico, la siguiente iteración se obtiene del mínimo de q_k. Cuando el Hessiano es definido positivo, el modelo cuadrático tiene un único mínimo que se puede obtener resolviendo el sistema lineal simétrico n X n

$$\nabla^2 f(x_k) sk = -\nabla f(x_k)$$

La siguiente iteración es entonces

$$x_{k+1} = x_k + s_k$$

Si el punto inicial está suficientemente cerca del mínimo local x* en el cual el Hessiano es definido positivo, la convergencia está garantizada. Más aún, la velocidad de la convergencia es cuadrática, esto es,

$$\| x_{k+1} - x^* \| \leq \beta \| x_k - x^* \|^2,$$

para alguna constante positiva β .

La *optimización restringida* comprende problemas de *optimización restringida no lineal, acotada, redes, estocástica y,* en particular, *programación lineal.*

3.3 OPTIMIZACIÓN COMBINATORIA

Los problemas de optimización combinatoria se refieren a la asignación eficiente de recursos limitados para resolver objetivos deseados cuando los valores de alguna o de todas las variables deben tomar sólo valores enteros. Las limitaciones en recursos básicos, tales como trabajo, materias primas o capital restringen las alternativas factibles. Sin embargo, en la mayoría de estos problemas, hay muchas alternativas posibles a considerar y la meta consiste en determinar cuales son las mejores.

La optimización combinatoria abarca problemas de las áreas más disímiles. Su amplitud proviene de que en muchos problemas prácticos, las actividades y los recursos, tales como máquinas, vehículos, personas, son indivisibles. Además, muchos problemas tienen solamente un número finito de opciones alternativas y por lo tanto, se pueden formular como problemas de optimización combinatoria. La palabra combinatoria se refiere al hecho de que existen sólo un número finito de alternativas factibles.

Así, la optimización combinatoria es el proceso de encontrar una o más soluciones óptimas en un espacio discreto. Problemas de este tipo aparecen en casi todos los campos administrativos (finanzas, comercialización, producción, control de inventarios, localización), y en innumerables ramas de la ingeniería (el diseño óptimo de canales y puentes, diseño de circuitos, determinación de los estados mínimos de energía en la fabricación de aleaciones, logística de la generación y del transporte de corriente eléctrica, programación de líneas de producción en fabricas, problemas de cristalografía).

A continuación, presentamos algunos problemas clásicos de optimización combinatoria.

3.3.1 EL PROBLEMA DE LA MOCHILA

Se dispone de un conjunto de n elementos para llenar una mochila de capacidad V. Cada objeto i tiene asociado un valor ci y ocupa un volumen vi. Se trata de determinar el subconjunto de objetos $I \subseteq \{1, 2, \ldots, n\}$ que se puedan introducir en la mochila y que maximice la función objetivo. El problema tiene un único vínculo lineal, que la suma del peso de todos los artículos no exceda el peso total permitido y una función objetivo lineal, la suma del valor de los artículos colocados en la mochila [40].

$$\max \sum_{i \in I} c_i$$

Sujeto ha:

$$\sum_{i \in I} v_i \leq V$$

El problema de la mochila tiene múltiples aplicaciones en criptografía, en la protección de archivos de computadora, correo electrónico y transferencia electrónica de fondos. La "llave" de seguridad consiste en la combinación lineal de un conjunto de datos que deben igualarse a un determinado valor.

La generalización del problema, con varios vínculos de mochila, tiene también aplicaciones importantes. Un ejemplo es el problema de presupuesto. El problema consiste en determinar un subconjunto de centenares de proyectos bajo consideración que aseguren el mayor retorno de la inversión, satisfaciendo los requerimientos financieros y regulatorios [41].

3.3.2 PROBLEMA DE REDES

Muchos problemas de optimización pueden ser representados por una red o grafo, definida por nodos y arcos que los conectan. Algunos surgen de redes físicas tales como las calles de una ciudad, red de carreteras, sistema ferroviario, redes de comunicación y circuitos integrados. Además, hay otros problemas que se pueden modelar como redes aunque no haya una red física subyacente. Por ejemplo, el problema de asignación donde se desea asignar un conjunto de personas a un conjunto de tareas a mínimo costo. Un sistema de nodos representan a las personas que se asignarán y otro sistema de nodos representan a los trabajos y habrá un arco que conecta a cada persona con cada trabajo que sea capaz de realizar [42] [43].

El problema puede ser establecido como

$$\min \sum_{ij} c_{ij} x_{ij}$$

Sujeto ha:

xij = 1, si el recurso I es asignado a la actividad j
xij = 0, de otra manera

$$\sum x_{ij}=1 \qquad j=1,\ldots, n$$

$$\sum_{j=1}^{i=1} x_{ij} = 1 \qquad\qquad I = 1,\ldots, m$$

Donde cij es el costo de aplicar el recurso i a la actividad j.

El problema de asignación es un caso especial del problema del transporte, el cual, a su vez, es un caso especial del problema de flujo máximo. Todos estos problemas pueden ser resueltos usando el algoritmo del simplex, pero cada problema cuenta con algoritmos más eficientes diseñados para tomar ventaja de su estructura particular.

3.3.3 PROBLEMA DEL AGENTE VIAJERO

El problema del agente viajero, TSP por sus siglas en inglés, es uno de los problemas que ha atraído más la atención dentro del campo de la Optimización Combinatoria. Es muy sencillo de enunciar, pero muy difícil de resolver. Dado un conjunto de n ciudades y el costo de viajar entre dos cualquiera de ellas, se trata de encontrar el camino que pasa por todas las *n* ciudades y que regresa a la de origen con un costo mínimo [49].

En términos de la teoría de grafos la formulación del problema es la siguiente: Dado un grafo completo con pesos, se trata de encontrar el ciclo hamiltoniano de peso mínimo. Aquí, los nodos representan a las ciudades, las aristas representan a los caminos entre ellas y los pesos de cada arista es el costo del camino que representa.

Formalmente, para el caso del agente viajero asimétrico,

$$\min \sum_{I,j} c_{ij}\, x_{ij}$$

Sujeto ha:

xij = 1, si se recorre la arista que va del vértice i al vértice j
xij = 0, de otra manera

$$\sum_{i=1} x_{ij} = 1 \qquad\qquad j = 1,\ldots, n$$

$$\sum_{j=1} x_{ij} = 1 \qquad\qquad I = 1,\ldots, n$$

$$\sum_{i \in S,\, j \in S^*} x_{ij} \geq 1 \qquad \text{para todas las particiones (S,S*) de los } n \text{ puntos}$$

donde **c***ij* es el peso del arco (*I,j*).

El último vínculo evita que se produzcan sub-tours.

3.4 MÉTODOS DE RESOLUCIÓN EN OPTIMIZACIÓN COMBINATORIA

La solución de problemas de optimización combinatoria es una tarea difícil. La dificultad se debe a que, distinto de lo que sucede en algunos problemas de optimiación donde la región factible es convexa y, en consecuencia una solución local es un óptimo global, en los problemas combinatorios se debe buscar en una red de puntos factibles y se tienen múltiples mínimos locales.

3.4.1 TÉCNICAS EXHAUSTIVAS

Un acercamiento obvio para resolver una instancia de un problema de optimización combinatoria es simplemente enlistar todas las soluciones factibles del problema dada, evaluar sus función objetivo y tomar la mejor. Sin embargo, esto es inmediatamente obvio que este acercamiento de completa enumeración es muy ineficiente; además, aunque es posible en principio resolver cualquier problema por este camino, en la práctica no los es, ya que existe un basto número de soluciones para cualquier problema de tamaño razonable. Para ilustrar este punto regresemos al problema del agente viajero.

Problema del Agente Viajero. Un vendedor tiene que buscar una ruta que visite cada una de las N ciudades, una y sólo una vez, y que además minimice el total de la distancia recorrida, 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 entre cada par de ciudades es la misma sin considerar la dirección del viaje.

Supóngase que tenemos una computadora que puede listar todas las posibles soluciones de una instancia; entonces utilizando la fórmula anterior, tendríamos como resultado la Tabla 3.1 [13] que nos muestra el crecimiento del tiempo de cómputo con respecto al tamaño de la entrada del problema. En este ejemplo podemos observar que la enumeración completa es ineficiente para obtener una solución óptima, ya que para 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 este tipo de problemas de manera más eficiente que la enumeración completa. Estos algoritmos fueron capaces de resolver pequeñas instancias, pero no lo fueron para encontrar soluciones óptimas, en una cantidad razonable de tiempo computacional, cuando las instancias 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.

Básicamente, hay cuatro métodos exhaustivos (exactos) generales para resolver problemas de optimización combinatoria, aunque en la práctica se combinan en procedimientos híbridos:

- Técnicas enumerativas
  - (a) Algoritmo de retroceso (backtracking)
  - (b) Rama y Cota (branch and Bound)

- Programación Dinámica
- Técnicas de relajación
- Dual Ascendente

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 [13] 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.

Sin embargo, 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 [21], que es una abreviación de non-deterministic polinomial [10]. De hecho el problema del agente viajero es de los problemas más difíciles en NP-completos, sí 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.

3.4.2 TÉCNICAS HEURÍSTICAS

Con la idea de combatir la enumeración exhaustiva para problemas NP-completos (que es el único método exacto para una serie de problemas combinatorios, para los cuales había de ofrecer una solución) surge la idea de heurística como método de enumeración parcial.

Una técnica 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 sí 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. La causa de este gran interés es doble.

a) Por un lado, el desarrollo del concepto de complejidad computacional ha proporcionado las bases para explorar las heurísticas.

b) Por otro lado, han surgido nuevas técnicas, muy eficientes, para encontrar "buenas" soluciones para problemas de optimización combinatoria en tiempos razonables de cómputo.

Las heurísticas son la respuesta a los problemas que presentan explosión combinatoria[3], sin embargo existen otros argumentos a favor del uso de las técnicas heurísticas.

a) Lo que se está optimizando en general es un modelo de un problema 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.

[3] Explosión Combinatoria: Cuando los números aumentan exponencialmente, un pequeño exponente puede producir resultados astronómicos.

b) Las técnicas Heurísticas son usualmente más flexibles y suelen hacer lo mismo con funciones objetivo y/o restricciones más complicadas con los algoritmos más exactos.
c) Una heurística resulta adecuada cuando se carece de algún método exacto para la solución.
d) Cuando requiere mucho espacio de memoria o mucho tiempo para resolverlo.
e) Cuando no se necesita la solución optima.
f) Se usan como paso intermedio para generar una solución de buena calidad en la aplicación de otro algoritmo. A veces son usadas soluciones heurísticas como punto de partida de algoritmos exactos de tipo iterativo.
g) Las heurísticas generalmente ofrece más de una solución, por lo cual se tienen más opciones de decisión.
h) En contraste de las técnicas exactas, resulta sencillo fundamentar una técnica heurística.

Por el contrario, también presenta inconvenientes el uso de métodos heurísticos. Uno de ellos es que por lo general no es posible conocer la calidad de la solución, es decir, cuan cerca se esta del optimo X^*, la solución X_{heu} que nos ofrecen. Existen métodos que nos pueden orientar con respecto a la calidad de la solución obtenida, por ejemplo podría generarse aleatoriamente varias soluciones y si son similares a X_{heu} cabría poner en duda la efectividad de la heurística.

3.4.3 TIPOS DE HEURÍSTICAS

Dado el carácter no polinomial de la mayoría de los problemas de optimización combinatoria se hace necesario contar con buenas heurísticas entre las cuales destacan:

1. Métodos constructivos. Este tipo de métodos construyen soluciones factibles usando reglas heurísticas en forma determinista y secuencial. Eñ representante más famoso de estos son los Algoritmos Greedy (glotones, voraces o ávidos).

2. Métodos de descomposición. Consisten en dividir el problema en subproblemas más pequeños, siendo la salida de uno la entrada del siguiente, de forma que al resolverlos todos obtengamos una solución para el problema global. Por ejemplo en el caso TSP, se podrían subdividir en viajes y finalmente unirlos.

3. Método de búsqueda por vecindades. Tiene como objetivo perfeccionar una

solución existente s, mediante una búsqueda local en una vecindad bien definida en torno de s en el espacio de factibilidad.

Los algoritmos genéticos se han convertido quizás en los métodos más conocidos entre las técnicas heurísticas. Ellos pueden se vistos como una forma de una búsqueda de vecinos [5], aunque su original inspiración proviene de la genética de la población. Esta técnica es la usada en este trabajo.

En los problemas de optimización combinatoria los métodos heurísticos sólo inspeccionan un pequeño subconjunto del espacio factible. Una heurística bien diseñada deberá explorar exclusivamente las soluciones más interesantes.

Los mayores inconvenientes con los que se enfrentan estas técnicas es la existencia de óptimos locales que no sean absolutos. Si a lo largo de la búsqueda se cae en un óptimo local, en principio la heurística no sabría continuar pues se quedaría "pegada" en ese punto. Una primera posibilidad para salvar esa dificultad consiste en reiniciar la búsqueda desde otra solución inicial y confiar que, es esta ocasión, la exploración siga por otros derroteros.

Se utilizo para este trabajo la técnica heurística de los algoritmos genéticos por que fue el que produjo los mejores resultados para este trabajo [41].

Capítulo 4

ALGORITMOS GENÉTICOS

En este capítulo se describe como funciona 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 este y los diferentes cambios que se han hecho para mejorar su rendimiento. Cabe señalar que los Algoritmos Genéticos son de las técnicas heurísticas más utilizadas recientemente, pues han demostrado su funcionamiento y precisión en diversos problemas tanto de optimización global o dando buenas aproximaciones a la soluciones para algoritmos NP-Completos [19].

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 [19] en su libro *"Adaptation in Natural and Artificial Systems"* [22] y trabajos posteriores, aunque muchas más personas han contribuido para mejorarlos o desarrollar variantes de este.

Los biólogos han estado intrigados con el mecanismo de la evolución desde que la Teoría de la Evolución fue aceptada. Para mucha gente, 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 por lozamientos fósiles prehistóricos.

Este mecanismo de la evolución no ha sido completamente entendido, pero muchas de sus características si. La evolución toma el nombre de cromosoma -como un dispositivo orgánico para codificar la estructura de los seres vivos [15]. 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 sus características de la teoría ha sido extensamente aceptada:

1 La evolución es un proceso que opera con cromosomas de un ser vivo que está codificado.

2 La selección natural es una liga entre cromosomas y el rendimiento de las estructuras decodificadas. El proceso de la selección natural causa que los cromosomas que tienen estructuras codificadas acertadas, se reproduzcan más que

otras que no lo son.

3 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.

4 La evolución biológica no tiene memoria. Cualquiera que sepa sobre la producción de los individuos sabe que su función esta 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 natural intrigó a John Holland en la década de los setentas. Holland creía que incorporando éstas apropiadamente en un algoritmo de computadora, la técnica resultante podría servir para resolver problemas difíciles como la naturaleza lo hace bien en la evolución [20]. Entonces empezó a trabajar con un algoritmo que manipula cadenas de dígitos binarios - 1's y 0's - al que le llamó cromosoma. El algoritmo de Holland trata de simular la evolución de pobladores que tienen cromosomas [19]. Como en la naturaleza, este algoritmo resuelve el problema de encontrar un buen cromosoma que manipula el material en los cromosomas ocultos. De igual forma, no se tiene idea del problema que se esta resolviendo. La única información que se tienen es la evaluación de cada uno de los cromosomas que se producen, y solo esta evaluación hace que en la selección de los cromosomas se escojan las mejores evaluaciones que se tengan y permitir que estas se reproduzcan mucho más que las malas evaluaciones.

Estos algoritmos, usan codificaciones y mecanismos de reproducción simples, un comportamiento complicado de despliegue y son usados para resolver problemas extremadamente difíciles. Como en la naturaleza, esto se hace sin el conocimiento del mundo decodificado. En general estas son simples manipulaciones de cromosomas simples. Hoy en día esto ha propiciado un gran número de algoritmos de este tipo, con el propósito de encontrar mejores diseños, encontrar mejores tiempos y producir mejores soluciones para una variedad de otros problemas también importantes, en los que no se pueden aplicar otras técnicas.

Cuando Holland empezó a estudiar estos algoritmos, el no sabía que nombre ponerles. Y como lo demostraba su potencial, al menos que fuera necesario, decidió ponerle el nombre de Algoritmos Genéticos [20].

Como se sabe, los descubrimientos que provienen de la evolución biológica también se integran a los Algoritmos Genéticos, es decir, la biología y la genética lo continúan influenciando. Esta influencia es en 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 teoría de la biología. En este punto, los algoritmos genéticos se parecen mas a las redes neuronales y al recocido simulado, que son otros algoritmos basados en la metamorfosis del mundo natural.

La descripción del algoritmo genético de la Figura 4.1 describe las bases de muchos de los Algoritmos Genéticos. Sin embargo, las diferentes investigaciones que se realizan hoy en día han modificado estas descripciones por diferentes caminos. Entre estos camino podemos encontrar tanto Algoritmos Genéticos hechos con un lenguaje estructurado, así como también en lenguajes funcionales y actualmente programados en lenguajes orientados a objetos.

4.2 VISTA DE ALTO NIVEL DE UN ALGORITMO GENÉTICO

Un fenómeno natural se puede abstraer 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, y en el de muchos de sus estudiantes, la codificación se realiza usando una cadena de binarios [20]. Probablemente no existe una mejor técnica para resolver todos los problemas, y ciertamente el seleccionar una buena técnica de decodificación es un problema que tiene que ser estudiado más profundamente. Un problema importante es derivado de las consideraciones que se tienen que tomar en cuenta como son las técnicas de selección y la representación del contexto del problema en el mundo real.

ALGORITMO GENÉTICO

- Inicializar la población de cromosomas.
- Evaluar cada uno de los cromosomas in la población.
- Crear nuevos cromosomas por acoplamiento de los cromosomas actuales; aplicando mutación y reconbinación como los cromosomas padres.
- Borrar los miembros de la población por los nuevos cromosomas.
- Evaluar los nuevos cromosomas e insertar estos datos en la población.
- Cuando se termine el tiempo, parar y regresar el mejor cromosoma; En caso contrario, regresar al tercer paso.

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

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 mismo cromosoma en el problema a resolver. La función de evaluación juega el mismo rol 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 una medida de que tan buena es cualquier 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 que son reemplazados por mejores y mejores hijos. El mejor individuo en la población final producida puede ser el que altamente envuelve la solución del problema.

4.3 SELECCIÓN DE LOS PADRES, MUTACIÓN Y CRUZAMIENTO

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

4.3.1 SELECCIÓNDE LOS PADRES POR RULETA

El propósito de la selección de los padres en un algoritmo genético, es el de seleccionar quien tiene más oportunidades de cruzamiento [14], o en este caso, quien de los miembros esta más ajustado. Una de las técnicas más utilizadas es el de la selección de los padres por ruleta. La selección de los padres por ruleta se describirá en la Figura 4.2.

El efecto de la ruleta en la selección de los padres es el de regresar un padre seleccionado aleatoriamente. Aunque esta selección sea aleatoria, cada oportunidad que tienen los padres de ser seleccionados es directamente proporcional a su adaptación - fitness -. En balance, al final del número de generaciones de este proceso conduce a tener los miembros mas adaptados y contribuye a separar el material genético en base al fitness de los miembros de la población. Claro que es posible que se seleccione el peor individuo de la población cada ves que se usa, sin embargo esta ocurrencia no inhibe el rendimiento del algoritmo genético, ya que las probabilidades de esta ocurrencia en una población de cualquier tamaño son despreciables.

De cualquier manera que se vea, esta técnica de selección de padres tiene la ventaja 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.

SELECCIÓN DE LOS PADRES POR RULETA

- Se suma la adaptación "fitness" de todos los miembros de la población; llamando a este resultado "total fitness".
- Se genera "n", un número aleatorio entre 0 y el "total fitness".
- Se regresa el primer miembros de la población cuyo "fitness", sumado a los "fitness" de los miembros de la población que lo precedieron, sea más grande o igual a "n".

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

4.3.2 CRUZAMIENTO Y MUTACIÓN POR BIT

El cruzamiento y la mutación por bit son componentes básicos para los Algoritmos Genéticos tradicionales. Estas funciones causan que la creación de nuevos 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 del cruzamiento y la mutación. Cuando la mutación de bit es aplicada a una cadena de bits, este barre toda la lista remplazando cada uno por un bit seleccionado aleatoriamente solo 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 presentan tres cromosomas padres de longitud 5, 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 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 este 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.

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, con números generados aleatoriamente para cada bit de los cromosomas, nuevos bits para las dos ocasiones 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.

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. Holland ha experimentado con 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 del corte.

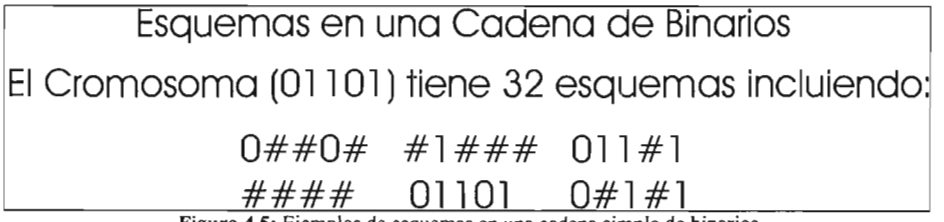

Figura 4.5: Ejemplos de esquemas en una cadena simple de binarios

Una importante característica del cruzamiento por punto es que este puede producir hijos que son radicalmente diferentes de sus padres. El primer ejemplo de la Figura 4.4 es una instancia de esto. Otra importante característica de este proceso es que a partir del punto de cruzamiento no se introducen diferencias de los bits en la posición donde los padres tienen el mismo valor. En el segundo ejemplo tenemos que el bit en las posiciones 2, 3, 4 y 5 tienen el mismo valor en los dos padres y en los dos hijos después de 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 el resultado sería un algoritmo muy poco práctico. Este razonamiento no es válido 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 esta corriendo, ya que, como en la naturaleza, la reproducción sexual es el proceso que más reacciones fuertes ejerce sobre ella. 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 los diversos cromosomas. En la Figura 4.5 muestra como un cromosoma sencillo de longitud 5 con 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 "#" 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.

Holland llama a estos bloques de estructuras como esquemas [15]. En sus investigaciones sucesivas de los Algoritmos Genéticos, Holland concluye que se manipulan los esquemas cuando se ejecuta el algoritmo. En efecto, El Teorema del Esquema de Holland nos dice precisamente esto. Si se usan las técnicas de reproducción, las oportunidades de hacer que se reproduzcan los individuos son directamente proporcionales 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 del algoritmo genético de la manera siguiente. Si tenemos que r es el promedio del fitness de todos los cromosomas en la población conteniendo a s, que n es el número de los cromosomas en la población conteniendo a s, y que a es el promedio del fitness de todos los cromosomas de la población. Entonces el numero 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.

El Teorema de los Esquemas nos dice que si una estructura que esta contenida en un cromosoma con evaluaciones aproximadas al promedio estas tienden a ocurrir en la siguiente generación, y que si una ocurrencia está 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; los diferentes bloques de estructuras que dan una buena solución están disponibles 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 es un Algoritmo Genético Híbrido, la diferencia que tienen estos con los Algoritmos Genéticos tradicionales, y el optimizador local numérico "secant unconstrained minimization solver" (SUMSL) [8] usado como un operador "genético" adicional el cual mejora uno o más individuos. También se mencionarán las diferentes técnicas que existen de Hibridación y particularmente que técnicas se utilizaron para nuestro sistema en particular.

5.1 HIBRIDACIÓN

La meta central que se busca en el desarrollo de un Algoritmo Genético es el de encontrar una forma de que el algoritmo sea robusto [15]. Este desarrollo se ha creado para que los Algoritmos Genéticos resuelvan una gran variedad de diferentes tipos de problemas. Este tipo de implementación ha servido bien para usarse en la representación binaria de los cromosomas y se ha singularizado en los operadores genéricos. La representación binaria puede codificar casi cualquier cosa y los operadores no incluyen el conocimiento del dominio de las optimizaciones que pueden estar haciendo.

Cualquiera de las optimizaciones que se quieran realizar hace que se tengan diferentes metas. Para que se pueda vender una aplicación de un Algoritmo Genético, se puede informar a la persona que tiene un difícil problema de optimización que existe un mejor algoritmo para resolver este problema en particular, optimizando el cómputo y el tiempo de la corrida. Aunque el Algoritmo Genético con 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. Este es un hecho que incluso en la naturaleza, para los individuos de cada una de las especies, hace que para una gran variedad de nichos que se pueden encontrar en un medio ambiente, el mejor individuo de un nicho no siempre será el mejor adaptado para el otro y este fallara en la competencia por recursos.

Para los partidarios de los Algoritmos Genéticos que buscan optimizar la vida de los individuos, estos hacen que la naturaleza deba adaptarse a los nichos que se quieran incorporar. Estos nichos incorporados son difíciles, ya que estos son representativos de los problemas de optimización del mundo real que se quieran intentar de resolver. Si esta adaptación procede, sólo nos servirá para este nicho en particular, ya que en los demás fallarán por que es diferente la competencia por los recursos. La gente del mundo real no

pagaría por múltiples soluciones ya que las comparaciones que se tendrían que hacer para encontrar la mejor solución serian muy caras. Estas preferirán 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 los 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 nuevas y mejoradas herramientas que continuamente son desarrolladas en los campos de las Ciencias de la Computación y de la optimización. Entonces cuando aplicamos cualquiera de estas técnicas para mejorar el funcionamiento del algoritmo genético tradicional, tenemos un algoritmo genético híbrido.

No pueden hacerse predicciones exitosas de este tipo para los algoritmos genéticos que se ha 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. Por lo general el algoritmo que se usará es aquel con el que se está 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 la de poder decodificar técnicas que nos sean poco familiares específicamente para el problema que tenemos que resolver [15], y con esto se obtiene un mayor número de recursos para llegar a esta solución.

Para lograr una buena hibridación se tiene que utilizar los siguientes tres principios básicos:

1 Uso de la codificación actual: Se debe usar la misma técnica de codificación del algoritmo para el híbrido

2 Hibridación cuando sea posible: Incorporar los desarrollos positivos del algoritmo actual para el híbrido.

3 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 las implicaciones que se tiene.

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 le sea natural para el usuario, subsecuentemente el algoritmo híbrido puede ser operado por la misma estructura que el algoritmo actual con el que se esta trabajando.

Usando la codificación actual se pueden producir técnicas de codificación que pueden ser utilizados 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 alternativa que tengamos de optimización para transformar nuestro algoritmo genético en un algoritmo genético híbrido. Esto nos trae una gran variedad de posibles caminos, incluyendo estos:

1 Si el algoritmo actual es rápido, al algoritmo híbrido se le puede adicionar la solución o soluciones como una producción de la población inicial. Por este camino el algoritmo genético híbrido con elitismo es garantizado que 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.

2 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 (1989) [15] 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.

3 Si el algoritmo actual es bueno en la interpretación de la codificación, puede que sea 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 presentar también 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 ser no tan ampliamente aplicada a la familia de los operadores genéticos que manipulan cadenas de números binarios. Entonces se tienen que crear operadores análogos dado la técnica de codificación que ha sido adoptada. Esto no es siempre fácil.

El operador cruzamiento, visto abstractamente, es un operador que combina partes de dos cromosomas de los padres para producir un nuevo hijo. La técnica de codificación adoptada debe tener soporte para operadores de este tipo, lo que está muy apegado al entendimiento del problema, la técnica de codificación, la función de cruzamiento y el cálculo de 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 el mecanismo del cruzamiento es 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 ha 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, siendo este un punto crítico para el algoritmo genético. Aquí 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 puede no estarse acercando a la solución del problema.

Finalmente, estas reglas que se han dado no se sabe si funcionaran siempre para solucionar el problema en su dominio. Como las heurísticas son propensas a fallar, pero estas conducen a las mejores soluciones con mayor frecuencia, estas podrían ser empleadas para incorporar elementos heurísticos en el conjunto de los operadores para poderlos accionar aleatoriamente dentro del proceso de búsqueda.

5.3 OPTIMIZADOR LOCAL SUMSL

En esta sección se describe el funcionamiento del algoritmo de optimización continua (ver sección 3.2) SUMSL, que es un método cuasi Newton bien conocido desarrollado por Gay [13]. SUMSL usa gradientes analíticos y aproximaciones Hessianas a través de actualizaciones de secantes. Este optimizador local es incorporado como un operador adicional al Algoritmo Genético que se implementó, el cual se empleó para la fase de minimización de la energía potencial (ECEPP/3).

Esta rutina interactúa con tres rutinas fundamentalmente las cuales son:

1 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).

2 CALCF. Esta rutina calcula la función con respecto al cromosoma x*

3 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 a 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 por que la cadena *x* es el cromosoma mismo, y es pasado por referencia. De tal forma que cualquier rutina a la que se le pasa por parámetro x* esta es modificada.

En la Figura 5.1 observamos que el cromosoma x* es pasado primeramente por la rutina "SUMIT", como se mencionó anteriormente, esta intenta encontrar un vector tal que este minimice a x* apoyándose en el cálculo del valor de este cromosoma en la línea a3 por medio de la rutina "CALCF". Una vez calculado se redirecciona nuevamente a "SUMIT", si no ha convergido con el mínimo se envía a la rutina "CALCG" para calcularle su gradiente. Todos estos pasos se repiten hasta que se encuentra la convergencia con el mínimo o se halla 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 a  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 de la parte fundamental del funcionamiento del optimizador local "SUMSL".

Capítulo 6

ALGORITMOS GENÉTICOS HÍBRIDOS ACOPLADO CON UN ANÁLISIS DE CLUSTERS

Uno de los problemas más grandes de los algoritmos genéticos es su lenta convergencia. Para aliviar esta dificultad muchos autores han propuesto el uso de un acercamiento híbrido. Una forma común de algoritmos genético híbridos es usar un optimizador local como un nuevo operador genético para mejorar los individuos de la población. Sin embargo, otro tipo de algoritmos híbridos involucra el acoplamiento de algoritmos genéticos con otras técnicas. En nuestro caso, acoplamos a nuestros algoritmos genéticos un algoritmo de Análisis de "Clusters" como un camino para reducir el espacio conformacional de los péptidos para los algoritmos genéticos.

6.1 ALGORITMO DE ANÁLISIS DE CLUSTERS DE DATOS

Los análisis de "Clusters" son métodos para buscar patrones, clusters (grupos) y relaciones presentes en datos dados. La clasificación de objetos similares en grupos es una importante actividad humana. La clasificación siempre ha jugado un importante Rol en la ciencia, como por ejemplo en la biología se clasifican animales, plantas, minerales, etc., en la Química la clasificación de los compuestos, en la medicina los diferentes tipos de cáncer, entre muchos ejemplos más.

Existen muchos diferentes tipos de métodos que se pueden emplear para la clasificación de datos. La elección de que algoritmo se va a emplear depende del tipo de datos que se tiene y del propósito en particular. Algunas veces cuando se tienen algunos algoritmos aplicables no solo basta con saber los argumentos a priori para seleccionar un método en particular. En estos casos se hacen diversas pruebas con estos algoritmos y se selecciona el que mejor represente la solución que deseamos a partir de los datos de entrada.

En los métodos de análisis de "clusters" de datos se consideran por lo regular dos tipos de algoritmos de análisis de "clusters" de datos, estos tipos son por particionamiento y por métodos jerárquicos. Para nuestro caso, el tipo que nos sirve es el de particionamiento.

El método de particionamiento construye k clusters. Esto es, que clasifica los datos en k grupos, los cuales satisfacen los requerimientos de la partición:

1 Cada grupo contiene al menos 1 objeto.
2 Cada objeto debe pertenecer exactamente a un grupo.

Estas condiciones implican que puede haber tantos grupos como objetos:

$$k \leq n$$

La segunda condición nos dice que dos diferentes "clusters" no pueden tener ningún objeto en común, y los k grupos forman el conjunto completo de datos.

Es importante notar que k es dado por el usuario. En efecto, el algoritmo construye una partición con muchos "clusters" como se desea. Claramente, no todos los valores de k conducen a una partición "natural", así es conveniente correr el algoritmo varias veces con diferentes valores de k y seleccionar para qué valor de k se tienen ciertas características o que gráficas son mejores, o para conservar los "clusters" que parecen proporcionar la mejor interpretación.

6.1.1 PARTICIONADOR EN BASE HA MEDOIDES

Las conformaciones moleculares fueron analizados usando el Particionador basado en el algoritmo de Medoides (PAM) proporcionado por el paquete de software del análisis estadístico de SPLUS. El PAM es una variante del bien conocido algoritmo "k-mean" para agrupar datos multidimensionales. Cada conformación molecular y su energía potencial corresponden a un sólo punto en el espacio multidimensional. Las conformaciones moleculares próximas en este espacio multidimensional tendrán perfiles similares. La meta del algoritmo PAM es identificar los grupos de los puntos (conformaciones moleculares) que son cercanas juntas pero lejos de otros grupos. El programa optimiza automáticamente la calidad de miembro de los grupos (clusters) para reducir al mínimo la distancia entre los miembros del grupo y para maximizar la distancia de otros clusters (véase la referencia 15 para los detalles). El programa de "clusters" de objetos PAM se mide por intervalos-escalados de variables, y puede ser aplicado cuando la entrada de los datos están representados en una matriz de disimilitud.

En el caso de nuestro estudio de la estructura tridimensional de las proteínas, lo que nos interesa en clasificarlas en q clusters las conformaciones moleculares en base a la energía termodinámica que desprenden cada una de ellas y la distancia angular que existe entre cada uno de los ángulos de torsión de los diferentes pobladores.

6.2 ALGORITMOS GENÉTICOS ACOPLADOS CON PAM

El algoritmo híbrido acoplado con PAM (AGHPAM) usado para predecir la estructura tridimensional de péptidos esta diseñado de la siguiente manera. AGHPAM consiste en dos fases que son iterativamente ejecutadas. En la primera fase de nuestro algoritmo corremos N algoritmos genéticos con tamaño de población N_{pop}. En cada una de estas corridas de los algoritmos genéticos, cada ángulo dihedro θ_i de cada individuo (molécula) es generado aleatoriamente en un espacio conformacional V_i. En la segunda fase se realiza un análisis de "cluster" de datos con los mejores q individuos de cada corrida de los algoritmos genéticos. El conjunto de estos $q*N$ conformaciones es analizado usando el algoritmo PAM para detectar clusters para cada uno de los ángulo de torsión θ_i. Por medio de estos clusters se generan nuevos espacios conformacionales V_i (regiones más prometedoras) para cada uno de los ángulos de torsión: Estos espacios conformacionales V_i son usados para generar los ángulos de torsión en la primera fase de las siguientes iteraciones. En la primera iteración, el espacio conformacional V_i para los ángulos diedros φ_i, ψi, $\chi_{i1},...,\chi_{iq}$, es [-180,180], mientras que para ω_i, $V_i=[175,180] \cup [-179,-175]$ que corresponden a la forma trans-planar del enlace peptídico [2], para $i = 1,..., k$. Un pseudo código de AGHPAM se muestra en la Figura 6.1

Note que en cada iteración se determinan por medio de PAM los espacios conformacionales de cada ángulo diedro. Con esta información, se reduce el espacio de soluciones para las siguientes generaciones de los algoritmos genéticos, haciendo cada vez una búsqueda más inteligente. Las conformaciones generadas en estos nuevos dominios son por lo general mejores que las generadas anteriormente.

Pseudocódigo del Algoritmo Genético Híbrido

Entrada	ga.seq (datos del péptido), V espacio conformacional, ga.inp (parámetros del Algoritmo)
Paso 1	Inicialice (a)Poner nivel de iteración a k = 1 (b)Poner el espacio conformacional de cada ángulo con intervalos fijos
Paso 2	Correr N veces el algoritmo con una poblacional aleatoria de tamaño fijo N_{pop} sobre el espacio conformacional V (c)Aplicar el operador mejorado (d)Guardar los tres mejores individuos de la generación. (e)Con una probabilidad igual a **pc** escoger a los padres para ser cruzados y crear a los descendientes. (f)Con la probabilidad **pm**, mutar a la población. (g)Con el mejor individuo mas los descendientes hacer la nueva población. (h)Si todavía no se ha ejecutado N veces el Algoritmo, ir a 1(a). En caso contrario ir a 3.
Paso 3	k = k + 1
Paso 4	Si todavía k no es igual al número de vueltas definido ir 5. En otro caso ir a Salida.
Paso 5	Algoritmo de análisis de "cluster" PAM Con los q mejores individuos de cada algoritmo genético obtenidos de 2 ejecutar el algoritmo PAM. Crear el nuevo espacio conformacional V (usado para las siguientes iteraciones)
Ir a 2	
Salida	ga.out (mejor individuo con sus ángulos dihedros)

Figura 6.1: Diagrama de flujo para el Algoritmo Genético Híbrido.

6.3 ELEMENTOS QUE CONFORMAN EL CÓDIGO DEL ALGORITMO GENÉTICO HÍBRIDO

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

1 Cromosoma: Arreglo de ángulos que conforman la conformación de un individuo.

2 Fitness: Variable que contiene el valor de la configuración del cromosoma, este valor es calculado por la función de energía ECEPP/3.

3 Individuo: Estructura que contiene la información necesaria de cada individuo que formará parte de la población. Esta estructura consta de un cromosoma, un fitness y variables que guardan la información de los padres con los que fue formado.

4 Población: Arreglo definido de Individuos (esta definido por el valor de popsize)

5 Bestfitness: Individuo independiente a la población, este conservara todas las características del individuo que tenga el fitness mínimo y se cambiará cuando encuentre a otro mejor.

6 Lchrom: Longitud de los cromosomas.

7 Popsize: Número de pobladores en cada generación.

Nuestro algo algoritmo genético híbrido incluye dos modificaciones a los algoritmos genéticos tradicionales. La primera modificación es que cambiamos el uso de reales en lugar de dígitos binarios para caracterizar los genes o los ángulos de torsión que describen al péptido. La segunda modificación es la inclusión de un operador mejorado que es un optimizador local (SUMSL) que dirige al cromosoma hacía a un mínimo local cercano. La modificación del algoritmo genético es fácil de implementar y facilita el uso del dominio de operadores específicos mientras que el proceso de codificación/decodificación es evitado.

El algoritmo Genético Híbrido puede ser aplicado a cualquier tipo de problema relacionado con el problema de 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 número de ángulos que la conforman, el numero de generaciones, la probabilidad de cruzamiento y mutación. Al final de su operación el sistema dará como resultado un archivo con el valor

mínimo encontrado junto con la estructura tridimensional correspondiente y para propósitos estadísticos se dará a conocer el numero de iteraciones, mutaciones y de cruzamientos que se hicieron para encontrarlo.

A diferencia de los algoritmos comunes, este Algoritmo Genético Híbrido en lugar de buscar el individuo que tenga el máximo peso con respecto a la función objetivo, este los minimiza. Puesto que el objetivo principal es el de buscar es la estructura de la proteína que tenga le menor energía potencial.

6.4 IMPLEMENTACIÓN DE LOS ALGORITMOS GENÉTICOS PARA PREDECIR LA ESTRUCTURA TRIDIMENCIONAL DE LAS PROTEÍNAS

En esta sección se presentará la implementación de las partes principales del algoritmo genético híbrido tales como la implementación de los individuos, la población inicial, la función objetivo, etc. para predecir la conformación tridimensional de las proteínas.
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. Sea k en número de residuos de aminoácidos en la molécula dada. Entonces en la conformación molecular se tiene un conjunto de ángulos diedros que son codificados en un vector variable que sigue la secuencia primaria usando el orden interno convencional $[\phi_i, \psi_i, \omega_i, \chi_{i1}, ..., \chi_{iq}]_{i=1,...,k}$ donde i_q es el número de ángulos diedros laterales de la i-ésimo residuo. Así cualquier vector $x = (\theta_1, ..., \theta_m) = (\phi_1, \psi_i, \omega_1, \chi_{i1}, ..., \chi_{i1}, ..., \phi_k, \psi_k, \omega_k, \chi_{k1}, ..., \chi_{kq})$ determina el cromosoma o al individuo, donde $\theta_1, ..., \theta_m$ son los genes que definen la conformación del péptido.Valores reales con un decimal son asignados a estas variables.

Población Inicial. La población inicial de un tamaño fijo N_{pop} fue generada aleatoriamente como sigue. Los ángulos $\omega_{1,...,}\omega_k$ son forzados a tomar aleatoriamente cualquier valor real en el conjunto $[175,180] \cup [-179,-175]$. Los ángulos diedros remanentes son permitidos a explorar el espacio conformacional de valores reales en el intervalo $[-180^o, +180^o]$ en la primera fase de la primera iteración de AGHPAM. Para las siguientes iteraciones, los rangos de estos ángulos son determinados por el algoritmo PAM. Por lo contrario, el vector de parámetros es siempre escogido en los siguientes intervalos, determinados experimentalmente (ver Tabla 6.1)

Parámetros	Intervalos
Probabilidad de Cruzamiento	[0.7,10]
Probabilidad de Mutación	[0.3,7.0]

Tabla 6.1: Intervalos para los parámetros de los GAs

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

Selección. Ya que la función 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_β con valor de energía ECEPP3, $E(s_\beta)$ es calculada con:

$$p_s(s_\beta) = \frac{(E_{\max} - E(s_\beta))}{N_{pop}(E_{\max} - E_\alpha)}$$

Donde N_{pop} es el tamaño de la población, E_α 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 reproducida el mejor de los individuos encontrados se introducirá en la nueva población.

Cruzamiento. La probabilidad de cruzamiento de una pareja de individuos está dada por el promedio de sus probabilidades individuales de cruzamiento. Si 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 $x = (\theta_1,...,\theta_m)$ 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 algoritmo el usuario determina el número de generaciones que se aplica en cada sesión del algoritmo. Después de terminar estas iteraciones definidas, este presentará un informe de los resultados obtenidos.

6.5 PROGRAMAS PARA LA IMPLEMENTACIÓN DE LOS ALGORITMOS GENÉTICOS HÍBRIDOS

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 describen en forma más detallada.

1 La función Objetivo. La función objetivo del Algoritmo Genético Híbrido es el optimizador local SUMSL. La forma en que funciona consiste es que una vez tenido el cromosoma resultante del proceso de cruzamiento y mutación, se obtiene su energía que desprende por su configuración y posteriormente localmente se optimiza el resultado por medio del cálculo de su gradiente por aproximaciones a través de secantes.

2 La generación de la nueva población. En la primera fase de la primera iteración de AGHPAM, los individuos son elegidos como se muestra en la Sección 6.4. En las siguientes iteraciones de AGHPAM, cada uno de los ángulos de la molécula (individuo) se toma aleatoriamente en los espacios confromacionales obtenidos por el algoritmo de PAM.

3 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 el proceso de minimizar el valor de la función objetivo, luego dentro del proceso de la asignación del fitness se hace una conversión de este por medio de la siguiente regla:

   $Poblador[j].fitness = Máximo - Poblador[j].fitness + 1.00$

   Donde:

   j indica el poblador j-ésimo
   Máximo es el máximo local de esta generación

   Entonces como resultado tendremos que el poblador que tenga el menor fitness será el nuevo máximo local, y viceversa.

4 Proceso de cruzamiento. Aquí una vez escogidos los dos individuos que se hallan eleccionado anteriormente según su probabilidad de cruzamiento se hace el intercambio de información o no. Si se realiza este proceso entonces de procede a escoger un índice i-ésimo al azar entre la longitud del cromosoma y una ves echo

esto, los ángulos del primero al *i*-ésimo del segundo individuo son remplazados con los del primero y los ángulos *i*-ésimo al ultimo del primero son copiados con los del segundo.

5 Proceso de mutación. Después de que dos individuos se hallan sido cruzados, a cada uno de ellos se le aplica este proceso. Este consiste en que según la probabilidad de mutación se escoge un índice al azar del individuo y en este lugar se cambia el ángulo por otro escogido al azar (este número al azar esta entre los intervalos establecidos para la proteína). Esta acción se realiza k veces, donde k es la longitud del cromosoma del poblador.

6 Escalamiento de la población. En este proceso se llevará acabo una vez que se halla creado la población y se halla 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 estos. Esta nos servirá para que el "fitness" sea mejor distribuido, para que en el proceso de selección de la generación siguiente se escoja a diferentes individuos y se desechen a los peores.

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

$Poblador[j].fitness = a * Poblador[j].fitness + b$

Donde:

a y b son los factores lineales de la población.

Capítulo 7

PARALELISMO

En esta sección se presentan las características principales de la computación paralela, como son los diferentes tipos de memoria, y se mencionan algunas herramientas de programación y bibliotecas.

7.1 INTRODUCCION

Sistemas más rápidos y eficientes, tanto en el área de computación científica como en el área comercial, ha ido en constante aumento al correr de los años. Los límites físicos existentes para un aumento de la velocidad en un único procesador y su alto costo nos llevan al surgimiento de nuevas arquitecturas para sistemas de cómputo de alto desempeño. Estas máquinas pueden ser clasificadas en tres tipos:

Multiprocesadores vectoriales: poseen un pequeño número de procesadores, donde cada uno es un procesador vectorial de alto desempeño.

Sistemas MPP (Massively Parallel Processors): poseen de centenas a millares de procesadores interconectados, constituyendo una línea principal de computadoras paralelas. Pueden ser de memoria distribuida o compartida.

Estaciones de trabajo conectadas por redes: máquinas interligadas por red de media y alta velocidad, que pueden trabajar como una máquina virtual de alto desempeño.

7.1.1 ¿QUÉ ES UNA COMPUTACIÓN PARALELA?

La Computación Paralela es el uso de más de una Unidad de Procesamiento Central (CPU) para la solución de un problema, Figura 7.1 muestra esquemáticamente este proceso. La computación paralela abarca el campo entero de las máquinas. Desde dos computadoras personales conectadas via Ethernet a miles de los más poderosos procesadores en una supercompuatdora paralela.

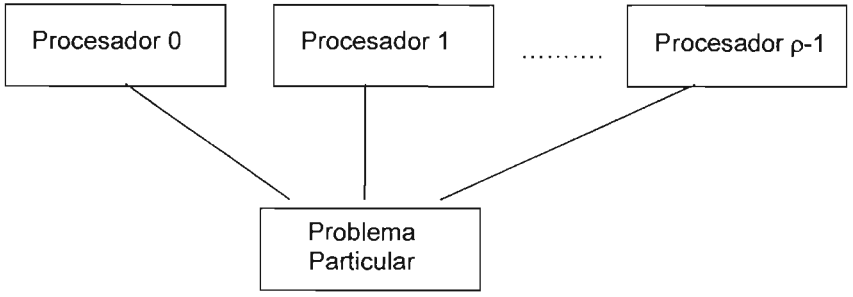

Figura 7.1: Computadora paralela usando mas de un procesador para resolver en conjunto un problema particular

7.1.2 TERMINOLOGÍA

Algunos de los términos utilizados en la técnica del procesamiento en paralelo son descritos a continuación:

1 Tarea o proceso: es la unidad lógica mínima definible de un programa.

2 Ejecución secuencial: ejecución de un programa o tarea en un único procesador, donde una instrucción es procesada independientemente.

3 Paralelización de código: Transformación de un programa secuencial en uno paralelo, a través de la identificación de tareas dependientes para un programa, de manera que se ejecuta de modo paralelo. Generalmente requiere modificaciones al código de un algoritmo utilizado en modo secuencial.

4 Aceleración de un programa (Speedup): La principal medida de la eficiencia de una paralelización es el aceleración (*speedup),* la cual es la razón entre los tiempos de ejecución de la versión secuencial y paralela de un mismo programa o algoritmo.

Para un determinado algoritmo, se considera la mejor implementación secuencial como referencia para comparar con la implementación paralela. Así la aceleración S_ρ para ρ procesadores se define como:

$$S_\rho = \frac{tiempo\_de\_ejecucion\_de\_un\_programa\_secuencial}{tiempo\_de\_ejecución\_de\_la\_versión\_paralela\_con\_\rho\_procesadores}$$

5 Eficiencia de un programa: Razón entre la aceleración de un programa y el número de procesadores utilizados para la ejecución del mismo

$$eficiencia = \frac{aceleración}{número\_de\_procesadores}$$

Intuitivamente la eficiencia está relacionada con la fracción de tiempo que un procesador está siendo utilizado para ejecutar el algoritmo.

Idealmente, si ρ procesadores utilizan el 100 % de su tiempo solamente para operaciones de computación del algoritmo, la eficiencia será igual a 1.

Como un programa paralelo requiere comunicación entre procesadores, parte del tiempo de los procesadores es gastado es esta actividad, llevando a valores de eficiencia menores a 1.

Ley de Andahl: Cantidades posibles de aceleración debido a la paralelización de porciones de código. Es decir, la cantidad que podemos acelerar el proceso de relación al tiempo secuencial, dependiendo de la cantidad de código que es factible paralelizar y el número de procesadores.

La Figura 7.2 nos muestra la aplicación de la Ley de Amdahl, en donde vemos la eficiencia de los procesadores utilizados en relación al porcentaje del código que ha sido paralelizado, ejemplo:

$$T_p = T_s(\frac{P_p}{N_p}) + P_s$$

donde:

T_p	Tiempo de corrida de programa en paralelo
T_s	Tiempo de corrida de programa secuencial
P_p	Porcentaje de tiempo usado corriendo en paralelo
N_p	Número de procesadores
P_s	Porcentaje de tiempo usado en secuencial

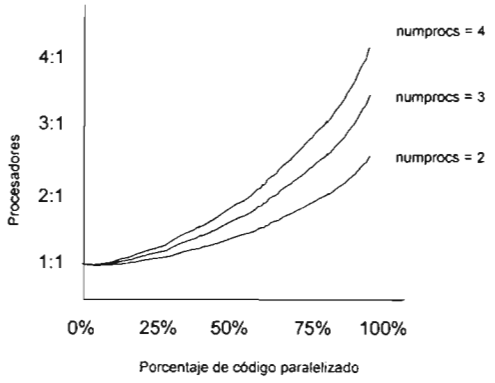

Figura 7.2: Ley de Amdahl

Si la porción de código que es factible programarse en paralelo es el 100% y utilizamos 4 procesadores entonces: el tiempo de CPU sería la cuarta parte del tiempo que utilizo en la versión en serie, hay que recordar que este sería el caso ideal.

1 Sincronización: Coordinación temporal entre procesadores, usada para un paso de información entre ellos. La sincronización es un factor de decremento de la aceleración exigida por un programa en paralelo, debido a que algunos procesadores puede estar inactivos, esperando que otros procesadores terminen sus respectivas tareas. Una comunicación entre procesadores se decide asíncrona cuando exige una coordinación entre tareas.

2 Granulado: Volumen de procesamiento realizado por cada tarea, en relación al volumen de comunicación existente entre las tareas. Un programa de granulado muy fino posee tareas que realizan pocas instrucciones y necesitan comunicarse mucho, en cuanto a un programa de granulado grueso posee tareas que ejecutan muchas instrucciones y se comunican poco. Cuando la comunicación entre los procesadores sea asíncrona, el granulado afecta el tiempo de ejecución de un programa, esto es porque programas con granulado fino necesitan sincronización más frecuente, introduciendo periodos de inactividad de los procesadores y aumentando el tiempo de ejecución.

3 Escalabilidad: Este es un sistema definido por un algoritmo paralelo en una máquina

paralela y escalable cuando la aceleración obtenida crece proporcionalmente al número de procesadores utilizados por el hardware. Los factores que contribuyen para la escalabilidad de un sistema son el hardware, el algoritmo utilizado y el código implementado.

4 Balanceo de carga: Distribución de las tareas entre los procesadores, a modo de garantizar la ejecución lo más eficiente posible del programa en paralelo. Este balance puede ser visto de manera estática, antes del inicio de la ejecución, o de manera dinámica durante el tiempo de la ejecución.

5 Determinismo: Un algoritmo o programa es deterministico si su ejecución con una entrada especifica siempre resulta en una misma salida. Se dice que un programa es no determinístico si existe la posibilidad de obtener salidas diferentes para varias ejecuciones del programa, utilizándose siempre una misma entrada.

7.2 MODELOS DE COMPUTACIÓN PARALELA

En este punto se presenta una breve clasificación de cómputo paralelo.

Clasificación según Flynn [25]: SISD, MISD, SIMD y MIMD

En 1966 Flynn publica sus estudios sobre arquitectura en paralelo, los cuales establecieron la terminología tradicional. Debido a los avances tecnológicos de las últimas dos décadas, la relevancia de esta taxonomía ha disminuido, pero la terminología ha sido muy usada para describir las computadoras en paralelo.

En 1966 Flynn establece las cuatro clases de arquitecturas de máquina

- SISD Single Instruction Single Data
- SIMD Single Intruction Multiple Data
- MISD Multiple Instruction Single Data
- MIMD Multiple Instruction Multiple Data

Hacemos referencia a ellas por sus siglas en inglés. A continuación las describiremos:

La arquitectura SISD es la arquitectura no paralela, por ejemplo la PC convencional.

La arquitectura SIMD consiste de un número de procesadores idénticos procesando un ciclo de pasos sincronizados y realizando el proceso sobre diferentes datos.

El modelo MISD es la arquitectura teórica propuesta por Flynn pero no tuvo uso.

El modelo MIMD cubre un vasto número de esquemas de interconexión, de procesadores y arquitecturas. El punto clave es que cada procesador opera de forma independiente de los otros, potencialmente corriendo programas totalmente diferentes. Los procesadores de MIMD se comunican utilizando una red de alta velocidad. La red permite que los procesadores compartan datos y sincronicen cálculos. Raro es el problema en paralelo que no necesiten ninguna comunicación o sincronización entre procesadores. Las máquinas MIMD modernas utilizan explícitamente un paradigma de paso de mensajes para comunicar los procesadores. Un procesador en específico envía a un procesador destino, el cual recibe el dato.

7.3 MODELOS DE ORGANIZACIÓN DE MEMORIA

Existen dos tipos básicos de organización de memoria: memoria compartida y memoria distribuida, los cuales se describen a continuación.

7.3.1 MEMORIA COMPARTIDA

En las máquinas que utilizan memoria compartida (ver Figura 7.3), todos los procesadores pueden acceder a cualquier dirección de memoria. La sincronización entre tareas es realizada a través de un control de operaciones de lectura y escritura ejecutadas por las tareas de memoria compartida. Una ventaja de este tipo de acceso es que el compartir de datos puede ser de manera rápida. Una desventaja es el nivel de escalabilidad del sistema, es decir, el limitado número de caminos existentes entre los procesadores a la memoria.

Una manera de eliminar la desventaja anterior consiste en proveer a cada procesador con una memoria local. La cual almacena un programa para ser ejecutado y las estructuras de datos que no son compartidas entre los procesadores. Las estructuras globales están almacenadas en memoria compartida, conforme a lo ilustrado.

Una extensión de este tipo de arquitectura anterior consiste en eliminar totalmente el espacio de memoria que se comparte. En ambos casos, estas referencias vistas por un procesador en la memoria de otro procesador son controladas por el hardware de interconexión de memoria con los procesadores. Se debe observar que los tiempos de acceso a memoria locales son típicamente menores que los tiempos a memoria remota. Un factor importante para la aplicación correcta de este modelo y el control cobre el acceso a memoria compartida de las tareas. Por ejemplo una tarea no puede alterar un dato mientras otra tarea esta leyendo el mismo dato.

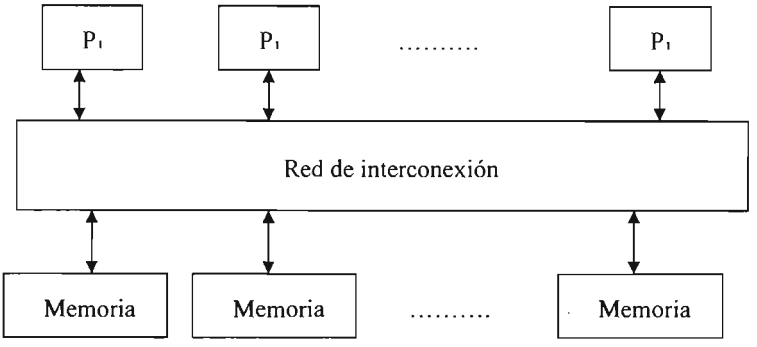

Figura 7.3: Arquitectura de memoria compartida

7.3.2 MEMORIA DISTRIBUIDA

En máquinas que utilizan memoria distribuida (ver Figura 7.4) y distribución de memoria física entre los procesadores, cada memoria local sólo puede dar acceso al procesador al cual está asociado. El intercambio entre las tareas se realiza a través de mensajes entre las redes de comunicación.

El envío y recepción de mensajes entre los procesadores pueden ser vistos en forma con-espera o sin-espera. En el envío con-espera la ejecución de una tarea que genera el envío de un mensaje es interrumpida antes que el mensaje sea recibido por el receptor; en cuanto que en la forma sin-espera inicia el proceso de envío y continúa su ejecución sin aguardar a confirmar que el mensaje fue recibido por el receptor. En el recibimiento con espera, una tarea es interrumpida cuando espera la llegada de un determinado mensaje. En la forma sin-espera se verifica si el mensaje dejado está disponible, en el caso de que no esté, se continúa con el procesamiento.

Las ventajas de utilizar programación distribuida son la escalabilidad del sistema, la posibilidad de mayor confianza en el sistema a través del duplicado de datos entre los

procesadores y consecuentemente, la tolerancia a fallas y la facilidad de la incorporación de procesadores mas especializados. Las desventajas son la difícil implementación de algunos algoritmos, tales como la descomposición de datos de un arreglo y la reestructuración de implementaciones ya existentes, por ejemplo en aplicaciones secuénciales o en aquellas que utilizan memoria compartida.

Debe hacerse notar la semejanza entre la arquitectura de memoria compartida y de la memoria distribuida. La mayor diferencia entre ellas es que, la red de interconexión permite que los procesadores ejecuten operaciones de lectura y de escritura en memorias pertenecientes a otros procesadores. En la segunda el acceso a memoria de otro procesador tiene que ser visto de manera explícita a través del paso de mensajes entre los procesadores.

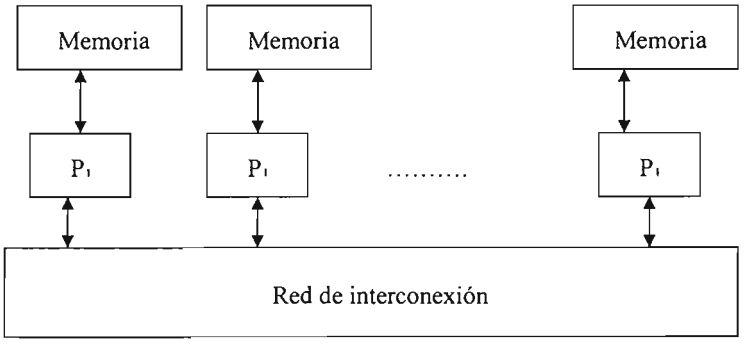

Figura 7.4: Arquitectura de memoria distribuida

7.4 MÉTODOS DE PROGRAMACIÓN

Como ya comentamos anteriormente, la programación paralela es una disciplina relativamente reciente, donde los patrones todavía no están bien definidos. De una forma general los métodos de programación paralela, actualmente utilizados, pueden ser clasificados en tres modelos Lewis [29].

1 Paralelismo de datos: los datos son distribuidos entre los procesadores que ejecutan una misma secuencia entre ellos. Un factor importante para la implementación eficiente de este tipo de programación es un direccionamiento de la localización de datos. Para que un programa de este tipo se ejecute eficientemente en una máquina de memoria distribuida, los datos deben estar alojados en la memoria de procesadores próximos a los que van a utilizar estos datos. Este modelo también es conocido como descomposición de dominio.

2 Paralelismo de tareas: el programa es fraccionado en tareas corporativas. Estas tareas pueden ejecutar diferentes procedimientos entre si, cuyo procesamiento puede ser visto de manera asíncrona. Para mayor eficiencia de la implementación de los programas debe tomarse en cuenta el direccionamiento de la localización de los datos como descomposición funcional.

3 Paralelismo de objetos: en este caso, el paralelismo puede ser realizado de diferentes formas. Por ejemplo, [2] una función miembro de un objeto puede llamar una función pública de otro objeto remoto o no y [3] un tipo abstracto de datos, como un arreglo, puede ser implementado a la eficiencia de una manera distribuida. Son necesarios los mismos cuidados relacionados a la eficiencia discutidos aun anterioridad con relación a la localidad de los datos y a la granularidad de las tareas.

Dependiendo de la aplicación, un modelo podrá ser más adecuado que otro. En muchas aplicaciones, una combinación de los paradigmas puede ser útil para un mejor desempeño del programa.

7.5 MODELOS DE ORGANIZACIÓN DE MEMORIA

Como en el caso de programación secuencial, existen varios lenguajes y herramientas para la implementación de programas en paralelo, siendo que cada una de estas representa una mejor opción para una determinada clase de problemas. Estas están divididas en dos clases: lenguajes paralelos y bibliotecas para paso de mensajes. La primera clase abarca los lenguajes que poseen instrucciones especiales para la creación de aplicaciones paralelas por los usuarios. La segunda clase contempla las herramientas que ofrecen funciones de una biblioteca de funciones para que el usuario realice la comunicación entre las tareas de su programa paralelo.

Lenguajes paralelos: La mayor parte de estos poseen extensiones vistas en lenguajes secuénciales ya existentes, de modo que se tornan transparentes para el usuario las primitivas de comunicación utilizadas en el sistema de computación, como una tentativa de facilitar la implementación de programas paralelos. Estos lenguajes poseen un compilador de lenguaje que transforma el programa del usuario y envía el resultado a un compilador de lenguaje secuencial; como ejemplos de lenguaje paralelo puede citarse a HPF (High performance Fortran), Fortran-M, Network Linda, C, C++, Regis, SR, etc.

Bibliotecas para paso de mensajes: Una manera simple de agregar una nueva funcionalidad a un lenguaje de programación a través de la creación de bibliotecas de software. El usuario realiza llamadas a nuevas rutinas de biblioteca para procesar las nuevas funciones, sin que el lenguaje original tenga que ser modificado. Debido a la facilidad de implementación antes mencionado, han sido desarrolladas varias herramientas basadas en bibliotecas de paso de mensajes para facilitar el desarrollo de programas en paralelo; por ejemplo: PVM (Parallel Virtual Machine), P4, Express y MPI (Message Passing Interface).

7.6 MPI

MPI es una especificación de librerías para el paso de mensajes en arquitecturas de memoria distribuida. Estás librerías están propuestas como un estándar basado en comités de vendedores, desarrolladores y usuarios.
Entre las características más importantes de estas librerías están:

- El MPI estándar es disponible para múltiples plataformas.
- El MPI fue diseñado para un alto rendimiento en máquinas masivamente en paralelo y en Estaciones de Trabajo conectados en "Clusters".
- El MPI esta disponible en software libre y en versiones de paga.
- El MPI fue desarrollado en base a comités de vendedores, desarrolladores y usuarios.
- La información de las librerías para los desarrolladores que utilizan MPI esta disponible
- Existe software disponible de Prueba de MPI para los desarrolladores.
- Portabilidad de los sistemas creados en base a MPI
- Escalabilidad.
- Existen versiones de estas librerías para C y Fortran.

Entre las características anteriores cabe destacar que se puede usar el MPI en implementaciones montadas en máquinas de diferentes tipos de una manera muy eficiente, ya que el MPI se encarga de hacer el intercambio de la información y solo el usuario especifica la lógica del mensaje. Esto es que el MPI es heterogéneo y transparente para el usuario que lo utiliza.

Las librerías MPI son muy eficientes en el intercambio de información, ya que no se

necesita de información extra y en muchos casos la información puede ser movida directamente de la memoria al conductor del mensaje, y del conductor del mensaje a la memoria.

7.7 PARALELIZACIÓN DE LOS ALGORITMOS GENÉTICOS

Cualquier método de algoritmos genéticos requiere de un extenso uso de tiempo de cómputo para tener una buena solución. Esto es generalmente por tener un largo número de generaciones o un intensivo uso de tiempo de cómputo por cada iteración (cruzamiento, mutación, etc.). Sin embargo si el algoritmo genético puede ser optimizado en un solo procesador, entonces es posible tener el uso de múltiples procesadores para acelerar la búsqueda de la solución. Para incrementar el número de generaciones por cada unidad de tiempo podemos hacer una de tres cosas: (a) acelerar el cálculo por cada generación, (b) compartir pobladores entre la población de cada procesador, y (c) ejecutar algoritmos genéticos independientes.

7.7.1 CLASIFICACIÓN DE LOS ALGORITMOS GENÉTICOS EN PARALELO

La idea básica detrás de la mayoría de los programas paralelos es dividir una tarea en pedazos y solucionar los pedazos que usan simultáneamente procesadores múltiples. Este acercamiento del divide y vencerás se puede aplicar al los algoritmos genéticos de muchas diversas maneras, y en la literatura contiene muchos ejemplos de implementaciones de algoritmos genéticos en paralelo acertadas. Algunos métodos del paralelización utilizan una sola población, mientras que otros dividen a población en varias sub-poblaciones. Algunos métodos pueden explotar masivamente la arquitectura de la computadora en paralelo, mientras que otros se enfocan a utilizar las computadoras con múltiples procesadores con pocos elementos de proceso pero que consumen más tiempo de cómputo. Hay tres tipos principales de algoritmos genéticos en paralelo: [2] el algoritmo genético global de maestro-esclavo de población sencilla, [3] algoritmos genéticos de granularidad fina de población sencilla, y [23] algoritmos genéticos de granuralidad gruesa de múltiples poblaciones. En un algoritmo genético basado en la relación maestro-esclavo hay una sola población (como si fuera un algoritmo genético simple), pero la evaluación de la función objetivo se distribuye entre varios procesadores. Puesto que en este tipo de algoritmo genético en paralelo, la selección y el cruzamiento trabaja sobre la población entera, también se les conoce como algoritmos genéticos en paralelo globales. Los algoritmos genéticos de granularidad fina son utilizados para las computadoras masivamente en paralelo y consiste en una estructura espacial de la población. La selección y el acoplamiento se restringen a una vecindad pequeña, pero al traslape de los vecinos permiten una cierta interacción entre todos los individuos. El caso ideal debe tener solamente un individuo para cada elemento de proceso disponible.

Los algoritmos genéticos de múltiple población son más sofisticados, y consisten en tener muchas sub-poblaciones con intercambios ocasionalmente de individuos. Este intercambio de individuos es llamado migración y es controlado por múltiples parámetros. Los algoritmos genéticos de múltiple población son muy populares, pero también es la clase de algoritmos genéticos en paralelo más difíciles de entender, por que el efecto de migración no esta totalmente entendido. Estos algoritmos introducen cambios fundamentales en la operación de los algoritmos genéticos y tienen diferentes comportamientos que los algoritmos genéticos simples.

Los algoritmos genéticos de múltiple población son conocidos con diferentes nombres. Algunas veces son llamados como algoritmos genéticos distribuidos, por que usualmente son implementados computadoras de memoria distribuida MIMD. También como en la computadora el radio de comunicación el alto, ocasionalmente se le llama algoritmo genético de granularidad gruesa. Finalmente, como estos algoritmos genéticos tienen múltiples poblaciones donde relativamente una esta aislada de las otras, entonces también se les conoce como algoritmos genéticos en paralelo de "islas".

7.7.2 PARALELIZACIÓN DE AGHPAM

Nuestro algoritmo fue implementado utilizando un modelo similar al tercer modelo de los algoritmos genéticos en paralelo (algoritmos genéticos de granularidad gruesa de múltiples poblaciones), en donde cada procesador tiene sus propias poblaciones independientes de las demás. El intercambio de material genético de los pobladores se realiza al pasar los q mejores individuos de cada población a PAM, para que la información genética de estos individuos sea distribuida a los demás en forma de un ajuste al espacio conformacional de la proteína.

En nuestro algoritmo en paralelo el primer procesador distribuye a los demás los datos iniciales como son, la probabilidad de mutación, la probabilidad de cruzamiento, el espacio conformacional de cada ángulo de torsión, etc. Posteriormente cada uno de los ρ procesadores ejecuta aproximadamente el mismo número de algoritmos genéticos. Si un procesador es escogido, por ejemplo el cero, este procesador recibe los q mejores individuos de cada algoritmo genético de los otros ρ-1 procesadores. El procesador cero ejecuta PAM con los datos recibidos y con los suyos. Con todos estos datos recalcula los espacios conformacionales para cada ángulo de torsión y con esto reduce de manera "inteligente" el espacio de búsqueda. Después, el procesador cero distribuye el nuevo espacio conformacional a los demás procesadores, los cuales ejecutan de nuevo los algoritmos genéticos sobre este espacio conformacional. La topología usada para el AGHPAM en paralelo es una estructura de árbol en la cual cada procesador difrenete de cero es conectado al procesador cero (ver Figura 7.5).

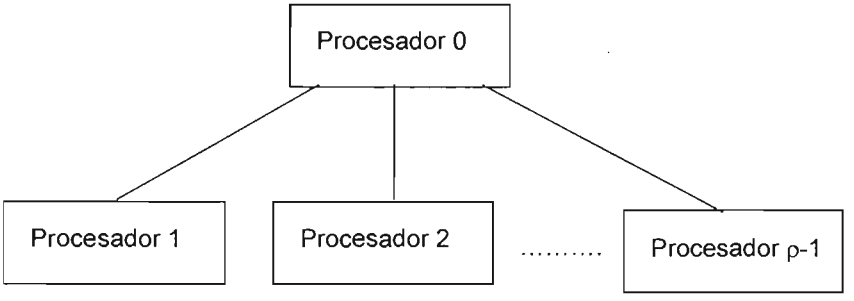

Figura 7.5: Configuración de la red de comunicación entre los procesadores utilizado en AGHPAM

Capítulo 8

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.

8.1 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. En los primeros experimentos computacionales el Algoritmo Genético Híbrido acoplado con PAM fue compilado con la opción -O2 y corrido en un solo procesador de la máquina SGI/CRAY ORIGIN 2000 para analizar el funcionamiento del sistema y medir su tiempo de cómputo. Posteriormente corrimos la versión en paralelo de nuestro algoritmo usando de 2, 4, 8 y 16 procesadores y se realizaron las mediciones del tiempo de CPU.

Para todas las moléculas analizadas, los parámetros para AGHPAM fueron encontrados experimentalmente. El tamaño de la población fue N_{pop} = 16, número de generaciones 8, probabilidad de cruzamiento 0.6 y la probabilidad de mutación 0.2 para los algoritmos genéticos. Por otro lado, el número máximo de iteraciones para AGHPAM fue 4. En número de algoritmos genéticos que corremos en la primera fase de cada iteración de AGHPAM fue $N = 32$ y el parámetro $p = 3$. Entonces el algoritmo PAM fue aplicado a $p*N$ (= 96). Mientras el número de clusters a ser encontrados fue 3.

Después de haber encontrado experimentalmente los parámetros, se procedió a realizar 10 corridas de AGHPAM con un solo procesador para cada una de las moléculas analizadas. Cada simulación para la proteína *Met-encéfalina* tomó cerca de 103 segundos por procesador de tiempo total de CPU y aproximadamente 46,000 evaluaciones de la función de energía ECEPP/3, para la proteína *Leu-encéfalina* tomó en promedio 105 segundos y 45,000 evaluaciones. Cabe recordar que la función objetivo es el optimizador local SUMSL, el cual para cada poblador calculado para la siguiente generación puede llevar hasta 200 evaluaciones de la función de energía ECEPP/3.

En la tabla 8.1 se compara 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 de mínima con mejor mínimo de energía para *Met-encéfalina* con los métodos conocidos como MCM, α 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 [37] reporta un valor de -12.396 kcal/mol, igual Androulakis [11] 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.

En la tabla 8.2 se compara los resultados obtenidos, con los mejores mínimos de energía de conformaciones de la *Leu-encéfalina,* obtenidos por otros autores usando diferentes métodos de optimización. Los tres primeros resultados muestran los ángulos diedros de la estructura de mínima con mejor mínimo de energía para *Leu-encéfalina* con los métodos conocidos como α para los primeros dos resultados y un algoritmo genético acoplado con un optimizador local para el último. El cuarto valor es el obtenido por nuestro AGHPAM. 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. Al igual que el resultado obtenido en la *Met-encéfalina* los valores de energía obtenidos son similares, en donde la diferencia también es derivada de las diferencias en los parámetros de ECEPP. La energía reportada por Floudas es similar a la de este trabajo.

		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.Biomol. Struct.Dynamics,9, 951 (1992).

AGH.- Esta tesis (Algoritmos Genéticos Híbridos).

Tabla 8.1: Ángulos dihedros de las 7 conformaciones con mejores mínimos de energía para la *met-encéfalina*

A continuación se presentara las gráficas de comportamiento del Algoritmo Genético Híbrido para obtener a la energía mínima de la proteína *met-encéfalina*. Aquí se ve que su comportamiento corresponde a una probabilidad Gauseana [17] Figura 8.1. En la Figura 8.2 se muestra la estructura tridimensional resultante del mejor mínimo encontrado de la m*et-encéfalina*.

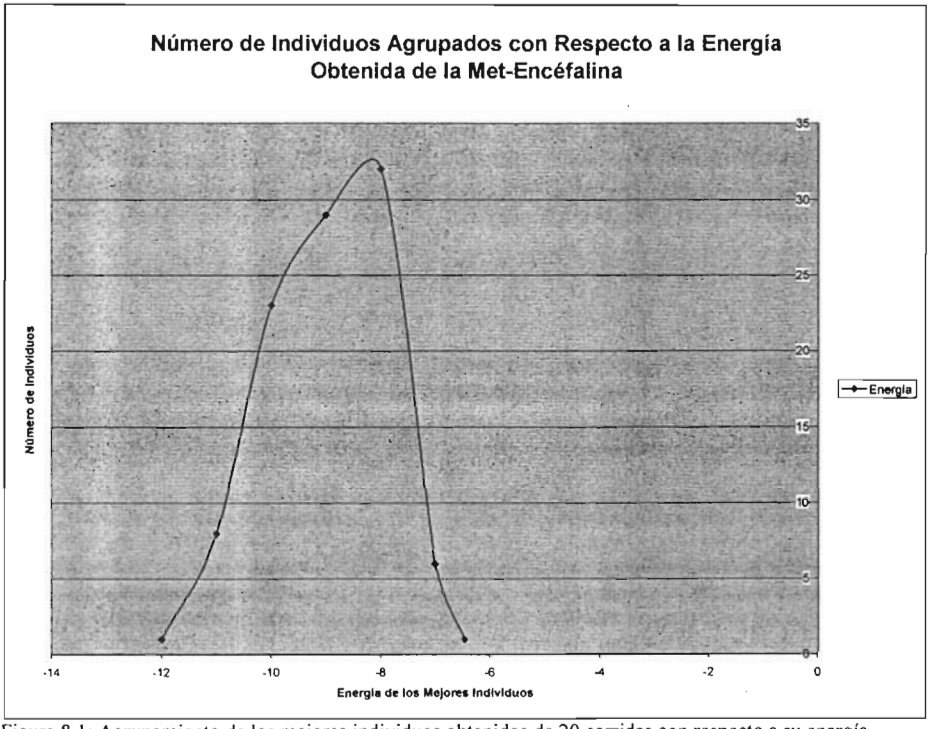

Figura 8.1: Agrupamiento de los mejores individuos obtenidos de 20 corridas con respecto a su energía obtenida.

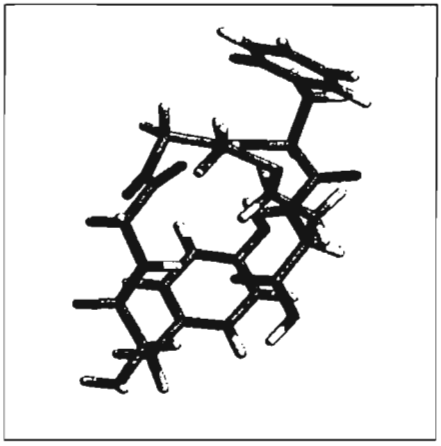

Figura 8.2: Gráfica de la *met-encéfalina*, obtenida por el Algoritmo Genético Híbrido acoplado con PAM

Como en la Figura 8.1 de la *Met-encéfalina*, la gráfica de la Figura 8.3 de la proteína *Leu-encéfalina* también corresponde a una probabilidad Gauseana. En la gráfica de la Figura 8.4 se presenta la estructura tridimensional del mejor mínimo encontrado de la *Leu-encéfalina*.

		A0	A1	BT	AGH
Tyr	ϕ	102.135	-163.07	-82.928	-168.2
	ψ	139.309	-42.29	155.394	-35.7
	ω	-166.873	-177.69	-173.356	-179.4
	χ_1	-163.758	-174.79	-171.896	171
	χ_2	79.92	90.16	-106.47	56.4
	χ_3	-154.795	-177.26	10.974	-158.6
Gly	ω	-140	65.9	-148.177	98
	ϕ	60.323	-88.33	70.332	43.2
	ψ	179.446	174.19	168.48	168.5
Gly	ω	69.574	-150.83	85.373	69.8
	ϕ	-85.997	31.94	-69.33	-91.3
	ψ	174.502	-178.71	-171.218	-177.9
Phe	ϕ	-101.036	-158.73	-135.494	-90.9
	ψ	-24.937	157.23	17.363	-32.4
	ω	-172.496	178.02	-173.684	-173.8
	χ_1	73.661	53.24	58.938	178.9
	χ_2	86.759	84.4	94.179	73.9
Met	ϕ	-81.092	-77.83	-161.513	-148
	ψ	131.465	123.31	160.391	124.8
	ω	177.678	-178.69	179.102	-171.1
	χ_1	-178.432	-179.81	52.272	177.6
	χ_2	65.431	64.47	174.335	62.7
	χ_3	-67.792	172.58	-179.78	52.5
Energía Kcal/mol		2.02E+01	2.75E+01	2.10E+01	2.76E+01

A0.- L.P. Androulakis, C.D. Maranas.
A1.- C.A. Floudas
BT.- R. Garduño Juárez [12][24].
AGH.- Esta Tesis (Algoritmos Genéticos Híbridos)

Tabla 8.2: Ángulos dihedros de las 7 conformaciones con mejores mínimos de energía para la *Leu-encéfalina*

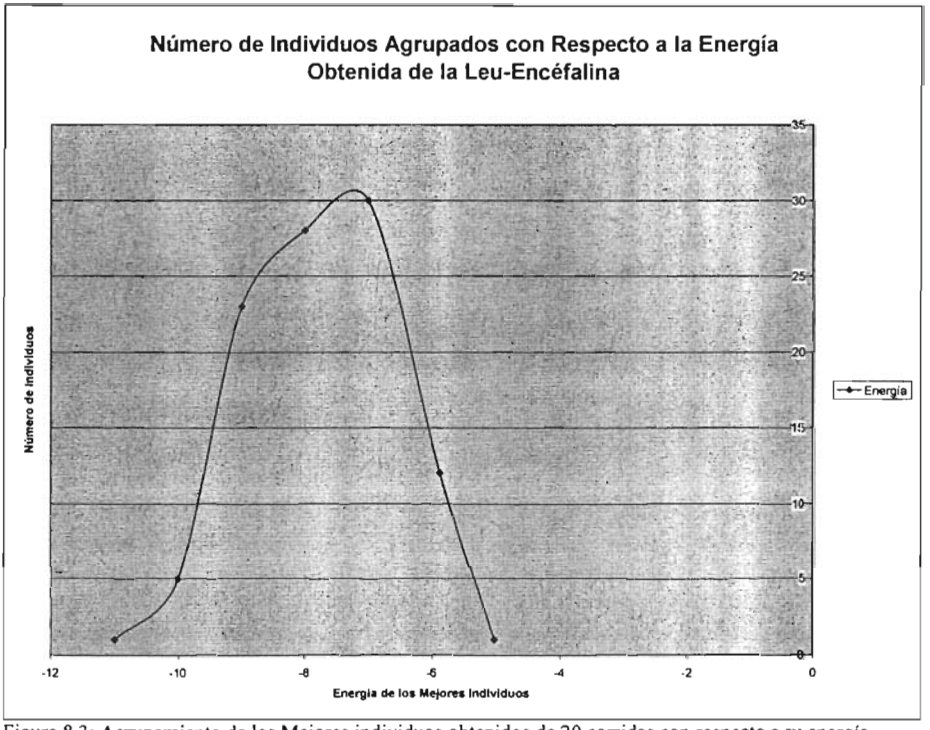

Figura 8.3: Agrupamiento de los Mejores individuos obtenidos de 20 corridas con respecto a su energía obtenida.

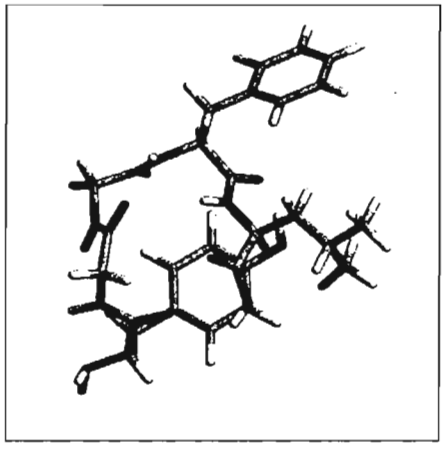

Figura 8.4: Gráfica de la *leu-encéfalina*, obtenida por el Algoritmo Genético Híbrido acoplado con PAM

8.2 RESULTADOS DEL ALGORITMO EN PARALELO

En esta sección se muestran los resultados del diseño en paralelo para el programa AGHPAM en un modelo MIMD. La versión en paralelo fue programada usando el lenguaje C. La comunicación entre procesadores fue ejecutada usando la librería Message Passing Interface Standard, versión 1.2. Los resultados computacionales en paralelo presentados aquí fueron obtenidos sobre la SGI ORIGIN 2000 con 32 procesadores. Dado que hemos usado MPI, nuestro programa en paralelo tiene un alto grado de portabilidad para otros sistemas de múltiples procesadores como los clusters.

El código en paralelo de AGHPAM ha sido probado sobre la *met-encéfalina* y la *Leu-encéfalina.*

Las curvas de aceleración (speed up) del algoritmo en paralelo para la *met-encéfalina* y la *Leu-encéfalina* son descritas en las figuras 8.5 y 8.6 (respectivamente) nos muestran la eficiencia ganada por el uso de múltiples procesadores, en donde el caso ideal es cuando tenemos por cada procesador un AGHPAM. Como se puede ver en las figuras, el speed up para la *Leu-encéfalina* es ligeramente superior al speed up de la *met-encéfalina.* Aunque para fines prácticos podemos considerar que el speed up para ambas moléculas son similares. El factor de la aceleración en estas figuras muestra claramente la ventaja del diseño del algoritmo en paralelo de AGHPAM. Por ejemplo, cuando usamos 16 procesadores obtuvimos un speed up de 10.7 y 11.7 para la *met-encéfalina* y la *Leu-encéfalina,* respectivamente.

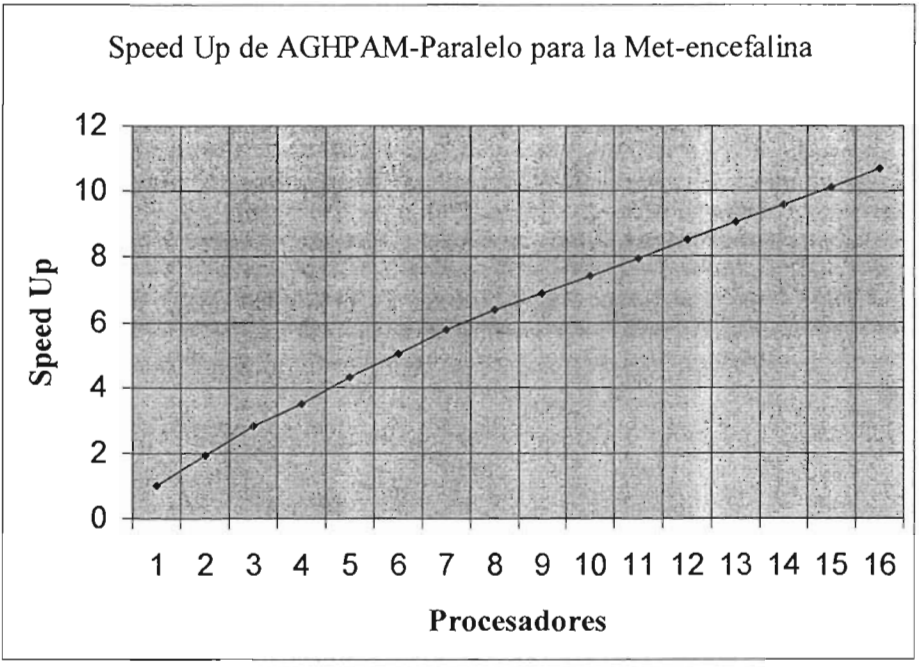

Figura 8.5: Gráfica del Speed Up obtenido de la *Met-encéfalina*

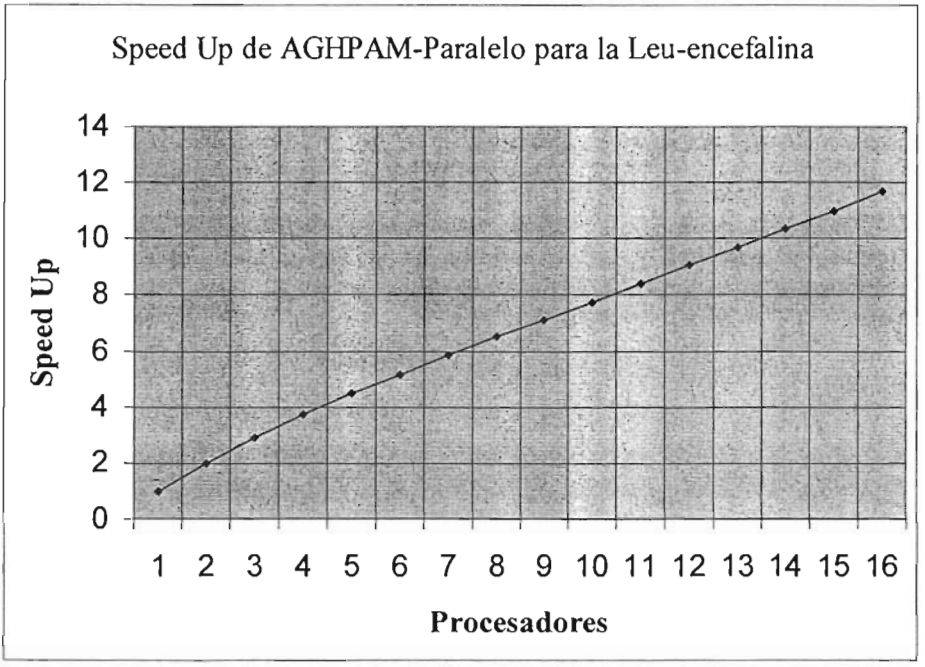

Figura 8.6: Gráfica del Speed Up obtenido de la *Leu-encéfalina*

8.3 CONCLUSIONES

En esta tesis se describieron los aspectos de una implementación de un algoritmo genético híbrido acoplado con un analizador de clusters para atacar el problema del plegado de proteínas. El algoritmo fue probado sobre las moléculas *Met-encéfalina* y la *Leu-encéfalina*, pentapéptidos que por muchos años han sido usadas como un modelo de validación para muchos optimizadores globales. También se presenta una versión en paralelo cuyos resultados nos muestran la bondad del diseño en paralelo para distribuir las cargas de los procesadores y reducir los tiempos de ejecución.

Los valores obtenidos en esta tesis para la proteína *Met-encéfalina* del algoritmo genético híbrido son acordes a los de Nayeem [37] y Androulakis [2] que consecuentemente definen a sus conformaciones como las correspondientes al probable mínimo global. El hecho de que diferentes procedimientos de optimización han obtenido las mismas energías conformacionales, da suficiente evidencia para decir que se obtuvo el mínimo global de energía para *Met-encéfalina*.

Similarmente los valores obtenidos en esta tesis para la proteína *Leu-encéfalina* del algoritmo genético híbrido son acordes a los de Floudas que consecuentemente definen a sus conformaciones como las correspondientes al probable mínimo global.

BIBLIOGRAFÍA

1. A. Scheraga, Harold A; ECEPP V.3.0 User Guide; Departament of Chemistry; Cornell University Ithaca; 1993.

2. 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. C. B. Anfisen, *Science 181*, 223-230 (1973).

5. C.R: Reeves, Genetic algorithms and neighbourhood search. In Evolutionary Computing: AISB Workshop, Leeds, UK, April 1994: Selected Papers, number 865 in Lecture Notes in Computer Science. Springer-Verlag, Berlin.

6. 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. Seibel, U.C. Singh, P.K. Weiner, P.A. Kollman, AMBER 5, University of California, San Francisco, C.A., 1997.

7. D. A. E. Cochran, S. Penel and A. Doig, *J. Protein Sci. 10*, 463-470 (2001).

8. David M. Gay; SUMSL; The National Science Foundation, 1982.

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

10. Hansmann, U.H.E.; Okamoto, Y. J Comp Chem 1993, 14, 1333.

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

12. http://www.fis.unam.mx/CCF/areasinvest/acad/academicos/ramong.html.

13. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Pub. Co. Inn. (1989).

14. K. Soman, R. Franczkiewicz, C. Mumenthaler, B. von Freyberg, T. Schaumann, and W. Braun, FANTOM v4.2, University of Texas Medical Branch, Galveston, Texas,

USA and Institut fur Molekularbiologie und Biophysik, Eidgenossische Technische Hochschule, Zurich, Switzerland, (1998).

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

16. L. B. Morales, R. Garduño-Juárez and D. Romero, *J. Biomol. Struct. Dyn. 8*, 721-735 (1991).

17. L. B. Morales, R. Garduño-Juárez and D. Romero, *J. Biomol. Struct. Dyn. 9*, 951-957 (1992).

18. L. B. Morales, R. Garduño-Juárez, J. M. Aguilar-Alvarado and F. J. Riveros-Castro, *J. Comp. Chem. 21*, 147-156 (2000).

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

20. J.H. Holland, Genetic Algoritms, Scientific American, July, 1992.

21. J.H. Holland, Adaptation in Natural and Artificial Systems: TheUniversity of Michigan Press, Ann Arbor, MI; 1975

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

23. R. Garduño-Juárez, L. B Morales and F. Pérez-Neri, *J. Mol. Struct. (THEOCHEM) 208*, 279-300 (1990).

24. R. Garduño-Juárez and F. Pérez-Neri, *J. Biomol. Struct. Dyn. 8*, 737-758 (1991).

25. P.J. Hatcher, M.J. Quinn. Data-Parallel Programing on MIMD Computers. Scientific and Enginering Computation. Juanus Kowalik, editor, 1991.

26. R. S. Judson, E. P. Jaeger, A. M. Treasurywala and M. L. Peterson, *J. Comp. Chem. 14*, 1407-1414 (1993).

27. R. Kning and T. Dandekar, *J. Mol. Model. 5*, 317-324 (1999).

28. R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*, John Wiley and Sons Inc. (1998).

29. S. Schulze-Kremer, In*Proccedings of the Second Workshop on Parallel Problem Solving from Nature*. (eds.) B. Mnner & B. Manderick. pp.391-400, Elsevier, Amsterdam (1992).

30. T. Brodmeier and E. Pretsch, *J. Comp. Chem. 15*, 588-595 (1994).

31. T.G. Lewis, H. El-Rewini. Introduction to parallel computing. Prentice may, 1992.

32. InsightII/discover'95, Biosym Technologies, Inc., 9685 Scraton Road, San Diego, CA 92121-2777, 1995.

33. Kostrowicki, J.; Cherayil, B.J.; Scheraga, H.A. J Phys Chem 1991, 95, 4113.

34. Ostrowicki, J.; Cherayil, B.J.; Scheraga, H. A.J. Phys Chem 1992, 96, 7442.

35. Kaufman, L. (1990): Finding groups in Data: An introduction to cluster analysis. John Wiley and Sons.

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

37. F. Mohamadi, N.G.J. Richards, W.C. Guida, R. Liskamp, C. Caufiled, G. Heng, T. Hendrikson, W.C. Still, W.C. Still, MACROMODEL, J. Comput. Chem. 11 (1990) 440-467.

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

39. L. B. Morales, R. Garduño-Juárez and D. Romero, J. Biomol. Struct. Dynam. 1991, 8, 721.G Nemethy, K.D. Gibson, K.A. Palmer, C.N. Yoon, G.

40. Nayeem, G.A.; Vila, J.; Scheraga, H. A. J Phys Chem 1992, 96, 4672.

41. Luis Germán Pérez Hernández. Un Algoritmo Genético Híbrido para Resolver el Problema de Plegado de Proteínas. Tesis de licenciatura, UNAM (2002).