UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.

FACULTAD DE INGENIERÍA.

TESIS:

DISEÑO DE UN ALGORITMO EVOLUTIVO PARA OPTIMIZACIÓN NUMÉRICA RESTRINGIDA INSPIRADO EN LA MADURACIÓN DE LA AFINIDAD.

Que para obtener el grado de:

Maestro en Ingeniería Eléctrica

Presenta:
Enrique Villalobos Zavala.

Director de Tesis
M.C. Ricardo Paramont Hernández García.

Septiembre del 2004.

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

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

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

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

Agradecimientos.

Le agradezco a mi padre por darme su apoyo, su cariño y la oportunidad de estudiar.

Agradezco a mi madre por ser la maestra de mi primer aprendizaje, por su amor y su entrega.

Le quiero agradecer a mi esposa Irene por alentarme y apoyarme para hacer los estudios de maestría.

Doy las gracias a mis hijos Mara y Marco Enrique por cederme parte de su tiempo para desarrollar y terminar este trabajo de tesis.

Quiero agradecer muy especialmente al Maestro Ricardo Paramont por sus enseñanzas, su apoyo, su tiempo y su dedicación para la realización de este trabajo.

Índice general.

Resumen. i

Índice general. iii

Índice de tablas. vii

Índice de figuras. ix

Capítulo 1: Introducción. 1

1.1 Planteamiento del problema. 1

1.2 Objetivo. 3

1.3 Hipótesis. 4

1.4 Estructura de la tesis 4

Capítulo 2: Modelos Computacionales Bioinspirados. 5

2.1 Modelos computacionales bioinspirados. 5

2.1.1 Optimización por colonia de hormigas. 6

2.1.2 Sistema inmune artificial. 7

2.1.2.1 La teoría de la red inmune 8

2.1.2.2 Mecanismo de selección negativa. 8

2.1.2.3 El principio de la selección clonal. 9

2.1.3 Redes neuronales. 10

2.1.4 Computación ADN. 12

2.1.5 Optimización por enjambre de partículas. 12

2.1.6 Autómatas celulares. 13

2.2 Computación evolutiva. 15

2.2.1 Selección Natural. 16

2.2.2 Genética. 16

2.2.3 Teoría sintética de la evolución. 17

2.2.4 Terminología. 17

2.3 Programación evolutiva 18

2.4 Programación genética 19

2.5 Clasificadores genéticos 20

2.6 Algoritmos genéticos. 21

2.6.1 La codificación. 22

2.6.2 La evaluación. 23

2.6.3 La selección. 23

2.6.4 El cruzamiento. 24

2.6.5 La mutación. 24

2.7 Estrategias evolutivas. 24

2.8 Optimización numérica de funciones. 25

2.9 Manejo de restricciones. 25

2.10 Técnicas clásicas de optimización. 26

2.10.1 Búsqueda exhaustiva. 26

2.10.2 Búsqueda local. 26

2.10.3 Métodos que usan soluciones parciales. 27

2.10.4 Búsqueda global. 27

2.11 Ventajas del algoritmo evolutivo sobre las técnicas clásicas 28

Capítulo 3: Sistema Inmune. 29

3.1 Sistema inmune. 30

3.1.1 Respuesta inmune innata. 30

3.1.2 Respuesta inmune adquirida. 31

3.2 Características de la respuesta inmune adquirida. 32

3.3 Linfocitos. 33

3.3.1 Linfocitos B. 33

3.3.2 Linfocitos T. 34

3.3.3 Receptores. 34

3.3.4 Antígenos y epitopos. 35

3.3.5 Afinidad. 36

3.4 Generación y maduración de linfocitos. 36

3.5 Maduración de la afinidad. 37

3.6 Respuesta inmune humoral. 38

3.7 Respuesta inmune celular. 38

Capítulo 4: Algoritmo evolutivo propuesto. 41

4.1 El algoritmo MARACoR. 41

4.1.1 La población de anticuerpos. 42

4.1.2 La transgresión. 42

4.1.3 La aptitud. 43

4.1.4 La maduración. 44

4.1.5 La mutación. 44

4.1.6 Cruzamiento. 45

4.1.7 Algoritmo. 47

4.1.8 Analogía entre el Sistema Inmune y el algoritmo propuesto. 55

Capítulo 5: Experimentación. 57

5.1 Lenguaje de programación y equipo usado. 57

5.2 Codificación de anticuerpos. 57

5.3 Prueba del algoritmo. 58

5.4 Los parámetros. 59

5.5 Los experimentos. 61

5.6 Resultados. 63

Capítulo 6: Análisis de resultados y conclusiones. 67

6.1 Análisis de resultados. 67

6.2 Evaluación del algoritmo. 68

6.3 Otros trabajos. 70

6.3.1 Koziel y Michalewicz 1999. 70

6.3.2 Runarsson y Yao 2000. 70

6.3.3 Coello y Cruz 2002. 71

6.3.4 Hernández, Botello, Lizárraga y Coello 2003. 72

6.3.5 Hernández 2004. 72

6.4 Conclusiones 73

Bibliografía 75

Apéndice A. 79

Índice de tablas.

Tabla 2.1. Modelos bioinspirados. 6

Tabla 4.1. Analogías entre el SI y el algoritmo propuesto. 56

Tabla 5.1 Algunas características de los problemas de prueba para los algoritmos propuestos. 59

Tabla 5.2 Valores asignados a los parámetros. 61

Tabla 5.3. Experimento 1. 63

Tabla 5.4. Experimento 2. 63

Tabla 5.5. Experimento 3. 63

Tabla 5.6. Experimento 4. 64

Tabla 5.7. Resultados promedio de los cuatro experimentos. 64

Tabla 5.8. Tabla que presenta los mejores resultados de los cuatro experimentos. 64

Tabla 5.9 Valores de las variables para los mejores vectores obtenidos en los cuatro experimentos. 65

Tabla 6.1. Resultados del promedio de los 4 experimentos de MARACoR. 69

Tabla 6.2. Los mejores resultados de los 4 experimentos . 69

Tabla 6.3 Resultados obtenidos por Koziel y Michalewics en 1999. 70

Tabla 6.4. Resultados obtenidos por Runnarson y Yao en 2000. 71

Tabla 6.5. Resultados obtenidos por Coello y Cruz. 71

Tabla 6.6. Resultados de IS-PAES. 72

Tabla 6.7. Resultados de EAAMRE. 72

Tabla 6.8 Trabajos evaluados con las 13 funciones. 73

Tabla 6.9 Trabajos evaluados con 10 de las 13 funciones. 73

Tabla 6.10 Trabajos evaluados con 3 de las 13 funciones. 73

Índice de figuras.

Figura 1.1. Ejemplo de un problema de programación no lineal. 3

Figura 4.1 Cruzamiento anular. 46

Figura 5.1. En un arreglo unidimensional del tipo double se almacenan las variables que representan las coordenadas de un punto en el espacio de busqueda S. 58

Resumen

En este trabajo se implementa un algoritmo evolutivo para optimización numérica restringida, que busca dar solución a problemas de programación no lineal sujetos a restricciones. Esta técnica heurística busca el valor óptimo de una función multidimensional, en un espacio de búsqueda que se encuentra fraccionado ya sea por igualdades o desigualdades que se conocen como restricciones. El individuo más apto encontrado por el algoritmo evolutivo representa un punto en el espacio de búsqueda.

El algoritmo evolutivo propuesto se inspira en la maduración de la afinidad. La afinidad es la fuerza de atracción que existe entre un anticuerpo y algún agente externo o interno (antígeno) que es considerado ajeno al organismo por el sistema inmune. El proceso de maduración de la afinidad se lleva a cabo cuando al multiplicarse los anticuerpos, por división celular, algunos de los nuevos anticuerpos sufren mutaciones, de tal manera que con este proceso los anticuerpos van aumentando su afinidad por el antígeno.

El algoritmo es sometido a un conjunto de funciones de prueba que ha sido usado en otros trabajos de investigación. Los resultados obtenidos muestran que el algoritmo es bastante robusto. Aunque en algunos casos los mejores valores encontrados son apenas superados por los encontrados en otros trabajos, lo robusto

del algoritmo le permite encontrar buenos resultados en todas las funciones de prueba.

Capítulo 1

Introducción.

En este capítulo haremos una descripción del problema que se quiere resolver y el objetivo que se busca alcanzar con este trabajo así como la hipótesis que sostiene esta tesis.

1.1 Planteamiento del problema.

El problema consiste en optimizar una función n-dimensional $f(x),\ x=(x_1,\ldots,x_n) \in R^n$ donde x pertenece a un espacio n-dimensional S llamado *espacio de búsqueda*.

$S \subseteq R^n$ esta limitado por las restricciones paramétricas li_i, ls_i que son los límites inferior y superior de la componente x_i de $\boldsymbol{x}$.

$f(x)$ es sometida a un conjunto de restricciones que dan como resultado que dentro del dominio de la función se forme una región en la que debe encontrase x para satisfacer las restricciones; esta región F donde se satisfacen todas las restricciones es llamada *región factible*.

La región F' donde una o más de las restricciones no se cumplen, se conoce como *región no factible*.

La optimización consiste en encontrar el valor de x tal que $f(x)$ tenga el mayor valor obtenido cuando se trata de una maximización o encontrar el valor de x tal que $f(x)$ tenga el menor valor obtenido en una minimización.

Al proceso de optimizar una función sujeta a un conjunto de restricciones, se le llama *manejo de restricciones*.

Una manera formal de hacer el planteamiento del problema es como lo hizo Koziel y Michalewicz (1999) [39]:

Optimizar $f(\boldsymbol{x}) \in R$, (1.1)

Donde:

$$\boldsymbol{x} \in S = [li_1, ls_1] \times \ldots \times [li_n, ls_n] \quad (1.2)$$

Sujeta a las restricciones:

$$g_i(\boldsymbol{x}) \leq 0, \text{ para } i = 1, \ldots, p \quad (1.3)$$

$$h_i(\boldsymbol{x}) = 0, \text{ para } i = p+1, \ldots, m \quad (1.4)$$

Donde p es el número de restricciones de desigualdad y m-p el número de restricciones de igualdad.

En (1.3) se definen las restricciones de desigualdad y en (1.4) las restricciones de igualdad. S (1.2) es llamado el espacio de búsqueda. li_i, ls_i representan los límites inferior y superior de la componente x_i de $\boldsymbol{x}$. La región factible F, está en función de las restricciones:

$$F = \{\boldsymbol{x} \mid g_i(\boldsymbol{x}) \leq 0, \text{ para } i = 1, \ldots, p \text{ y } h(x) = 0 \text{ para } i = p+1, \ldots, m\} \quad (1.5)$$

La optimización, se refiere a encontrar los valores de las componentes de x, tal que al evaluarse en $f(x)$ arroje un valor óptimo x_m:

$\boldsymbol{x}_m \in F \cap S$, que es tal que:

$$f(\boldsymbol{x}_m) \text{ es mejor o igual que } f(x) \text{ para } \boldsymbol{x} \in F \cap S \quad (1.6)$$

donde $f(\boldsymbol{x}_m)$ es mejor o igual que $f(x)$ significa:

$$\begin{cases} f(\boldsymbol{x}_m) \geq f(\boldsymbol{x}) \text{ si se está maximizando} \\ f(\boldsymbol{x}_m) \leq f(\boldsymbol{x}) \text{ si se está minimizando} \end{cases} \tag{1.7}$$

A este problema se le llama *problema de programación no lineal* (NLPP por sus siglas en inglés, *non linear programming problem*) sujeto a restricciones. No existe un algoritmo determinista que garantice que se encontrará el óptimo para cualquier caso [39], excepto quizá buscar el óptimo punto por punto en todos los puntos del espacio de búsqueda S. En la figura 1.1 se muestra la gráfica de la función de Ackley para tres dimensiones. La base del cubo es el espacio de búsqueda S, mientras que la gráfica suspendida en la parte superior del cubo es la función objetivo $f(x)$.

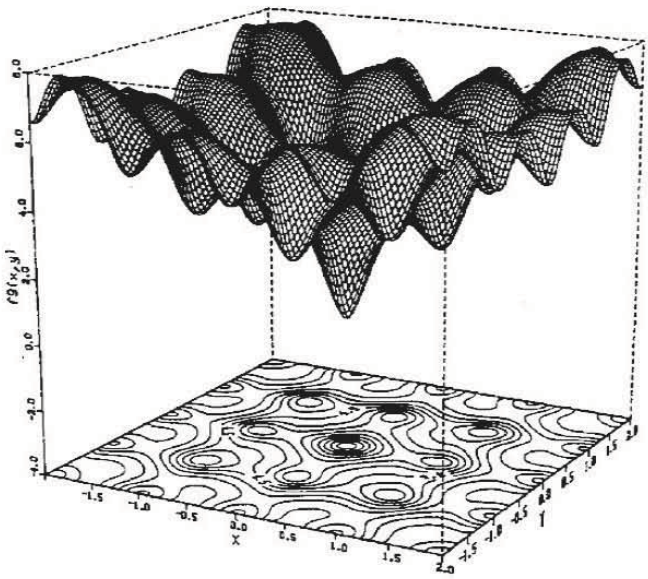


Figura 1.1. Ejemplo de un problema de programación no lineal[1].

1.2 Objetivo.

El objetivo de esta tesis es, **desarrollar un algoritmo robusto de optimización numérica restringida que tome como modelo, el mecanismo de maduración de la afinidad del sistema inmune, para resolver problemas de programación no lineal sujetos a restricciones. Implementándolo a través de un algoritmo evolutivo que utilice genes codificados con vectores de números reales**.

[1] La figura fue tomada del sitio http://tracer.lcc.uma.es/problems/ackley.html

1.3 Hipótesis.

La hipótesis es que se puede desarrollar un algoritmo evolutivo robusto que permita resolver problemas de programación no lineal sujetos a restricciones tomando como modelo la maduración de la afinidad y que al codificar los genes con vectores de números reales se mejoran los resultados obtenidos por algoritmos que usan diferente codificación como el de Runarsoon y Yao [40]; Coello y Cruz [4]; el de Hernández [36]; el de Koziel y Michalewicz [39] y el de Aguirre, Lizarraga, Botello y Coello [22].

1.4 Estructura de la tesis.

El trabajo consta de 6 capítulos, en el primero se hace el planteamiento del problema, se define el objetivo y se establece la hipótesis. En el segundo capítulo se hace una breve descripción de los modelos computacionales bioinspirados y se definen algunos conceptos que serán utilizados más adelante. En el tercer capítulo se describen algunos de los mecanismos del sistema inmune, algunos de los cuales fueron tomados como modelo para el presente trabajo. En el cuarto capítulo se describe el algoritmo propuesto "MARACoR" y se explican algunas de sus rutinas más importantes. En el quinto capítulo se presentan los resultados obtenidos en los experimentos realizados con el algoritmo. En el sexto y último capítulo se hace un análisis de los resultados, se compara con otros trabajos similares y se dan las conclusiones.

Capítulo 2

Modelos Computacionales Bioinspirados.

A partir de los años sesentas se han desarrollado varias técnicas y algoritmos que toman como modelo a la naturaleza para dar solución a problemas que no son fáciles de resolver por métodos convencionales. A estos modelos se les conoce como *modelos computacionales bioinspirados*. En este capítulo se hace un recuento de estas técnicas y se da una breve explicación de cada una de ellas.

2.1 Modelos computacionales bioinspirados.

Los modelos computacionales bioinspirados se basan en hacer analogías con sistemas naturales o sociales para determinar métodos heurísticos no determinístas de búsqueda y optimización para encontrar la solución a problemas difíciles de resolver por técnicas tradicionales. Algunos ejemplos se muestran en la tabla 2.1.

Modelo natural.	Modelo artificial.
Hormigas.	Optimización por colonia de hormigas.
Sistema inmune.	Sistema inmune artificial.
Cerebro.	Redes neuronales.
ADN	Computación de ADN.
Enjambre.	Optimización por enjambre de partículas.
Células	Autómatas Celulares.
Evolución.	Computación evolutiva.

Tabla 2.1. Modelos bioinspirados.

2.1.1 Optimización por colonia de hormigas.

Los algoritmos de colonia de hormigas. Parten principalmente del estudio del comportamiento social de las hormigas, realizado principalmente por los mirmecólogos Pierre-Paul Grassé y Edward O. Wilson, que definen a la colonia de hormigas como un solo organismo, cuyos componentes u órganos son las mismas hormigas. En general, la comunicación entre las hormigas es poca, la mayor parte de la comunicación se realiza a través de los actos que otras hormigas han realizado, o bien a través de la acumulación de feromonas en el suelo.

Casi todos los comportamientos de una colonia de hormigas, desde la construcción de estructuras, hasta la localización de comida, sirven como inspiración para resolver algún tipo de problema de búsqueda; como por ejemplo:

En la búsqueda de comida, al principio el rastro de feromona no es muy fuerte, las primeras hormigas que siguen el camino hacia la comida pueden desviarse y encontrar un camino más corto, con lo que se perfecciona la solución propuesta.

A la larga, la mayor parte de las hormigas siguen el camino hasta que el alimento se agote.

Sin embargo, si por alguna causa, una hormiga encuentra el "camino largo" antes de que otra encuentre el corto (lo cual pude suceder), entonces la colonia adoptará

el camino largo y convergerá a una solución que no es óptima. Aunque el método no es infalible la mayor parte de las veces el resultado es el camino corto que además también es el que tiene mayor probabilidad de ser encontrado mediante una búsqueda aleatoria.

A toda esta familia de algoritmos se le denomina *optimización por colonia de hormigas*.

2.1.2 Sistema inmune artificial.

El principio básico del sistema inmune artificial es diferenciar lo propio de lo ajeno y esto se logra a través de una serie de mecanismos: *Diversidad*, manteniendo librerías de posibles componentes de antígenos o agentes externos que usan para reconocerlos; *especifidad*, porque se puede crear un anticuerpo específico para cada diferente antígeno o agente externo; *memoria*, para guardar las respuestas inmunes llevadas a cabo anteriormente y poder utilizarlas cuando sean requeridas; y *tolerancia*, porque no se ataca al propio organismo, sólo a lo que se considera "extraño".

Es a través de estos mecanismos que el sistema inmune artificial puede utilizarse para encontrar soluciones en grandes espacios de búsqueda.

Dada la complejidad del sistema inmune, existe una gran cantidad de mecanismos que pueden servir como modelo para resolver problemas diversos. Sin embargo los sistemas inmunes artificiales se han inspirado básicamente en tres metáforas diferentes:

- La teoría de la red inmune.

- El mecanismo de selección negativa.

- El principio de selección clonal.

2.1.2.1 La teoría de la red inmune.

La teoría de la red inmune fue propuesta por Jerne en 1974 [30]. La hipótesis se basa en que el sistema inmune mantiene una red idiotípica (que transfiere factores hereditarios) de células B interconectadas para el reconocimiento de antígenos. Las células de alguna manera mantienen regulada la red aun sin la presencia de antígenos. Dos células B son interconectadas a través de sus receptores, si la afinidad que comparten rebasa un cierto umbral y la fuerza de la conexión es directamente proporcional a la afinidad.

La población inicial es generada a partir de un subconjunto de datos de entrenamiento de la red y los datos restantes se utilizan para entrenar a los antígenos. Las poblaciones siguientes de células B se generan a partir de dos subpoblaciones. Una, la población inicial y otra, una población clonada. Los antígenos son seleccionados aleatoriamente del conjunto en entrenamiento y son presentadas a las células B de la red. Si la afinidad es buena, entonces la célula B es clonada y mutada. La mutación produce un conjunto variado de anticuerpos que puede ser usado en el proceso de clasificación. Una vez que una nueva célula B es creada trata de integrarse a la red a través de la célula B más cercana; si no se puede integrar, es removida de la población y una nueva célula B es creada a partir del antígeno y es incorporada a la red.

2.1.2.2 Mecanismo de selección negativa.

El propósito de la selección negativa es proveer tolerancia hacia células propias y lo hace inspirados en la habilidad del sistema inmune para detectar antígenos desconocidos y no reaccionar a células propias. Durante la generación de las células T, los receptores son creados a partir de fragmentos de material genético seleccionados aleatoriamente, durante su proceso de maduración pasan al timo donde son sometidos a la selección negativa. Ahí los linfocitos T que reaccionan contra células propias son eliminados. Las células que sobreviven a este proceso continúan su maduración, dejan el timo y se incorporan al torrente sanguíneo para desempeñar funciones inmunológicas y proteger al individuo ante la presencia de antígenos.

En 1994 Forrest [37] propuso el algoritmo de la selección negativa que se basa en la idea de generar un conjunto de detectores binarios para que de manera aleatoria se hagan candidatos, rechazando aquellos detectores que reconozcan sus propios datos. Estos detectores más tarde son usados para detectar anomalías. El algoritmo del mecanismo de selección negativa consta de tres fases:

- Autodefinición.- Define un conjunto de cadenas de igual longitud a manera de identificadores propios del sistema, el cual será monitoreado por el sistema para su protección.

- Generación de detectores.- Se genera un conjunto de detectores que deben ser diferentes a las cadenas generadas en la autodefinición. Se usa una comparación de coincidencia parcial, donde las cadenas se consideran iguales, si un porcentaje de bits contiguos en dichas cadenas tienen coincidencia.

- Monitoreo de anomalías.- Continuamente se comparan los detectores con las cadenas propias y si hay coincidencia esto querrá decir que se ha presentado una anomalía.

2.1.2.3 El principio de selección clonal.

El principio de selección clonal describe las características básicas de la respuesta inmune al estímulo de un antígeno. La idea principal es que las células que reconozcan al antígeno sean las que proliferen tomando ventaja de las que no reconozcan al antígeno. Las principales características del principio de selección clonal son:

- Las nuevas células son copias exactas de sus padres (clones) y están sometidas a la hipermutación somática (mecanismo con altas tasas de mutación).

- Eliminación de los nuevos linfocitos, cuyos receptores provoquen una reacción auto inmune.

- Proliferación y diferenciación de células (que adquieren una función específica la cual desempeñarán hasta su muerte) maduras al contacto con el antígeno

2.1.3 Redes Neuronales.

Las redes neuronales son modelos computacionales encaminados a emular el funcionamiento del cerebro como la *percepción,* el *razonamiento* y el *aprendizaje*.

Estos modelos se caracterizan por que en ellos el procesamiento se realiza en forma paralela y distribuida a través de muchos procesadores altamente conectados entre sí. Estos procesadores o neuronas artificiales pueden ser virtuales o alambradas y emulan el comportamiento de las neuronas realizando operaciones muy sencillas. Las redes se pueden clasificar en:

Supervisada.- Mediante este tipo se introduce a la red una serie de patrones de entrada y salida. La red es capaz de ajustar los pesos con el fin de memorizar la salida deseada.

No supervisada.- Aquí la red responde clasificando los patrones de entrada en función de las características más adecuadas de cada uno.

Auto supervisada.- En este tipo la propia red corrige los errores en la interpretación a través de una retroalimentación.

El *entrenamiento* de la red es muy importante ya que servirá para que posteriormente la respuesta del sistema sea la adecuada. Las señales de entrada a una neurona artificial son variables continuas en lugar de pulsos discretos, como se presentan en una neurona biológica.

Cada señal de entrada pasa a través de una ganancia o peso, llamado *peso sináptico o fortaleza de la conexión* cuya función es análoga a la de la función sináptica de la neurona biológica. Los pesos pueden ser positivos o negativos.

La aplicación más común de las redes neuronales es el procesamiento de patrones, sin embargo existen modelos que pueden aplicarse a: Problemas de optimización, pronóstico y aprendizaje por recompensa.

Fue en 1943 cuando Warren McCulloch y Walter Pitts propusieron el clásico modelo de neurona en el que se basan las redes neuronales actuales. Seis años después, en 1949, en su libro The Organization of Behavior, Donald Hebb presentaba la regla de aprendizaje. Hebb observó que la sinapsis (conexión entre 2 neuronas reales) se reforzaba sí la neurona de entrada y la neurona de salida eran activadas

de manera continua. De esta forma, las conexiones que se usan son las que se refuerzan.

En 1957, Frank Rosenblatt presentó el *Perceptrón*, una red neuronal con aprendizaje supervisado cuya regla de aprendizaje era una modificación de la propuesta por Hebb. El Perceptrón se trata de una red supervisada que trabaja con patrones de entrada binarios. La principal aportación del Perceptrón es que la adaptación de los pesos se realiza teniendo en cuenta el error entre la salida que da la red y la salida que se desea. Este proceso se realiza en dos etapas: En la primera se presentan las entradas y las salidas deseadas; en esta fase la red aprende que salida debe dar para cada entrada. En la siguiente etapa, la red es capaz de responder adecuadamente cuando se le vuelven a presentar los patrones de entrada.

En los años 60 se propusieron otros dos modelos, también supervisados, basados en el Perceptrón de Rosenblatt denominados *Adaline* y *Madaline*. En estos, la adaptación de los pesos se realiza teniendo en cuenta el error, calculado como la diferencia entre la salida deseada y la dada por la red, al igual que en el Perceptrón. Sin embargo, la regla de aprendizaje empleada es distinta. Aquí se define una función error para cada neurona. Cuando se presenta una entrada a la neurona, ésta indica cual es el error cometido para cada valor posible de los pesos. A esta regla de aprendizaje se la denomina *Delta*.

La era moderna de las redes neuronales artificiales surge con la *técnica de aprendizaje de propagación hacia atrás*. La estructura de las redes citadas anteriormente consta de dos capas: La primera formada por unidades que dejan pasar la entrada y que no tienen aprendizaje, y la segunda capa está formada por una o varias neuronas en el caso del Madaline. Minsky y Papert demostraron que una red del tipo Perceptrón no es capaz de aprender todas las posibles combinaciones entre entradas y salidas. Ellos propusieron añadir capas intermedias de neuronas para resolver el problema.

El algoritmo de propagación hacia atrás tiene la función de enseñar a las capas intermedias. Este algoritmo compara la salida real con la salida deseada. La diferencia entre ambas constituye un error que se propaga hacia atrás desde la capa de salida hasta la de entrada permitiendo así la adaptación de los pesos de las neuronas intermedias mediante una regla de aprendizaje Delta.

Posteriormente se han desarrollado otros modelos que permiten un aprendizaje no supervisado como el *mapa auto-organizativo* de Kohonen, los basados en la *Teoría de Resonancia Adaptativa* (ART) de Grossberg y Carpenter, o los *modelos de control motor* de Bullock, Gaudiano y Grossberg, entre otros.

2.1.4 Computación ADN.

La computación ADN comenzó en 1994 cuando Leonard Adleman publicó "*Molecular computation of solutions to combinatorial problems*".

Adleman, realizó un experimento que consistía en demostrar que es posible hacer computación de una forma diferente a la tradicional, intentó resolver el problema de los caminos hamiltonianos, también conocido como del viajero, usando secuencias de ADN.

Esta técnica como su propio nombre indica, usa el ADN como base para calcular, en vez de algún equipo de cómputo. Aquí de lo que se trata es lograr usar como representación de un problema una cadena de ADN. Una vez que esto se logre, lo único que se tiene que hacer es soltar todas las posibles soluciones en una cuba PCR (Polimerase Chain Reaction). La cuba PCR usa el ADN polimeraza, una enzima, para replicar cadenas de ADN. Después de varias reacciones químicas las cadenas de ADN más aptas serán las que prevalezcan.

2.1.5 Optimización por enjambre de partículas.

La optimización por enjambre de partículas es una técnica de optimización estocástica basada en poblaciones desarrollada por Dr. Eberhart y el Dr. Kennedy en 1995 inspirada en el comportamiento de parvadas de pájaros y cardúmenes de peces.

La optimización por enjambre de partículas comparte muchas similitudes con las técnicas de computación evolutiva como los algoritmos genéticos. El sistema es inicializado con una población de soluciones aleatorias y se busca un óptimo a través de las generaciones. Pero la optimización por enjambre de partículas no tiene operadores de evolución como cruzamiento y mutación, aquí las posibles soluciones

llamadas partículas se desplazan sobre el espacio de búsqueda siguiendo a la partícula óptima.

Se trata de representar las diferentes soluciones a un problema mediante "individuos" de una manada. Cada individuo tiene unas coordenadas en el espacio de búsqueda, así como un valor de evaluación. Las partículas, tomando diferentes caminos, tratan de dirigirse hacia la mejor partícula del grupo, teniendo también en cuenta el mejor valor obtenido por las partículas en la vecindad de cada una. El movimiento se realiza según una aceleración, que es un parámetro del método.

Las tres reglas que rigen una bandada son: S*eparación* (evitar los sitios donde haya muchos individuos), *cohesión* (tratar de ir hacia el centro percibido de la bandada), y *alineación* (tratar de ir en la dirección general de la bandada).

Comparados con los algoritmos genéticos la optimización por enjambre de partículas tiene la ventaja de que es fácil de implementar y tiene pocos parámetros para ajustar. Algunos ejemplos de aplicación son: Optimización de funciones, entrenamiento de redes neuronales artificiales y control de sistemas difusos.

2.1.6 Autómatas celulares.

Un Autómata Celular es una herramienta computacional que está básicamente compuesta por una estructura estática de datos y un conjunto finito de reglas que son aplicadas a cada nodo o elemento de la estructura. Las ventajas que presenta esta técnica son la sencillez para la construcción de los modelos y la particularidad de los patrones de comportamiento presentados por el Autómata en tiempo de ejecución.

Los autómatas celulares fueron introducidos a finales de los años 40 por John von Neumann siguiendo una sugerencia de Stanislav Ulam con el objetivo de crear un modelo real del comportamiento de sistemas extensos y complejos. En un autómata celular los objetos que pueden ser interpretados como datos pasivos y los que hacen la función de dispositivos de computación se encuentran al mismo nivel y sometidos a las mismas leyes.

Los mecanismos que propuso para alcanzar estructuras autorreplicantes se asemejan bastante a los que utilizan los organismos vivos. De forma paralela a von Neumann, el ingeniero alemán Konrad Zuse desarrolló el concepto de autómata celular. Zuse estaba interesado en la construcción de modelos digitales de la mecánica, sin embargo sus trabajos no tuvieron tanto impacto como los de von Neumann.

En los años 60 John Holland comenzó a aplicar los autómatas celulares en problemas de optimización y adaptación. Cuestiones importantes sobre computabilidad y reversibilidad se han estudiado por Alvy Smith, Serafino Amoroso y Victor Aladyev.

En los años 70 John Conway en su etapa de estudiante en Cambridge diseñó "*el juego de la vida"*, donde se simula la evolución de una siembra de microorganismos sometidos a unas sencillas reglas que regulan los nacimientos y la supervivencia. Estas sencillas reglas dan lugar a un comportamiento muy complejo por parte de la colonia de organismos, las reglas son:

Supervivencia.- Una celda en estado "1" sobrevive si tiene 2 ó tres vecinos vivos a su alrededor.

Muerte.- Una celda viva muere por superpoblación si tiene 4 vecinos vivos o más; y muere por aislamiento si tiene uno o ningún vecino vivo.

Nacimiento.- Una celda en estado "0" nace si tiene 3 vecinos vivos a su alrededor.

Este juego de autómatas celulares se hizo muy popular gracias a Martin Gardner con sus artículos en Scientific American.

La cuestión de sí los autómatas celulares pueden modelar sólo aspectos fenomenológicos o si en cambio pueden servir para modelar las propias leyes Físicas fue estudiada por Edward Fredkin y Tommaso Toffoli en los años 80. El tema principal de su investigación es la formulación de que los modelos computacionales

de problemas físicos conservan la información, y de esta forma retienen su carácter reversible.

Los simuladores de autómatas celulares son capaces de actualizar millones de celdas en un tiempo extremadamente corto y son una herramienta ideal en la construcción de modelos simples de ecuaciones diferenciales como las ecuaciones del calor, de propagación de ondas o de Navier-Stoke.

En particular, los modelos de autómatas celulares se utilizan de forma regular en dinámica de fluidos. Finalmente para una de las ramas de la Física que más se está desarrollando como es la de los sistemas dinámicos, la aparición de fenómenos colectivos, la turbulencia, el caos, la fractalidad, los autómatas celulares dan una gran colección de modelos representativos de estos fenómenos, que pueden ser aislados y estudiados de forma sencilla. El uso sistemático de los autómatas celulares en este campo fue iniciado por Stephen Wolfram a mediados de los años 80.

2.2 Computación Evolutiva.

La computación evolutiva se basa en la teoría sintética de la evolución frecuentemente llamada *neo-darwinismo*.

La computación evolutiva es un conjunto de técnicas heurísticas de búsqueda y optimización a través de modelos computacionales evolutivos que generalmente se usan cuando los espacios de búsqueda son muy grandes y que en ocasiones se complican aún más por la aplicación de restricciones a dicho espacio. Los paradigmas de la computación evolutiva son: Los algoritmos genéticos, estrategias evolutivas, programación evolutiva, clasificadores genéticos y programación genética.

El algoritmo básico de la computación evolutiva es [42] [43]:

- Representar las posibles soluciones del problema.
- Generar aleatoriamente una población inicial de posibles soluciones.
- Una función de evaluación que permita identificar las soluciones más aptas.
- Operadores genéticos que varíen las posibles soluciones de generación en generación.

- Una serie de parámetros propios de la técnica usada (la probabilidad de aplicación de los operadores genéticos, la población inicial, el número de descendientes, etc.)

2.2.1 Selección Natural.

Darwin propuso en 1859 su explicación de la evolución mediante la teoría de la selección natural en su libro "El Origen de las especies" [9] que sostiene la tesis de que se producen cambios hereditarios que proporcionan ventajas en la lucha por la sobrevivencia, de modo que, a largo plazo, los organismos poseedores de esos cambios son los que prevalecen.

Después de varias generaciones esos cambios heredados de generación en generación serán típicos en la especie. Por el contrario, si los cambios no son favorables, los individuos de esa especie que posean estos rasgos tenderán a desaparecer hasta que después de varias generaciones no quede ninguno. A este proceso de la supervivencia del más apto y la desaparición del menos apto se le llamó "*Selección natural*". Sin embargo, Darwin no explica cómo son heredados estos cambios.

2.2.2 Genética.

La genética fundada por Mendel y desarrollada a partir de 1900 permitió explicar como se heredaban dichos cambios. Fue Mendel quien descubrió que los caracteres se heredaban de forma discreta, y que se tomaban del padre o de la madre, dependiendo de su carácter dominante o recesivo. A estos caracteres que podían tomar diferentes valores se les llamaron *genes*, y a los valores que podían tomar, *alelos*.

Los cromosomas están compuestos por genes que a su vez están compuestos por ácido desoxirribonucleico (ADN). Cada especie de seres vivos tiene un número fijo y característico de cromosomas. La macromolécula de ADN está compuesta por bases púricas y pirimidínicas, la adenina, citosina, guanina y timina. La combinación y la secuencia de estas bases forma el código genético, único para cada ser vivo.

El código genético codifica todas las proteínas que forman parte de un ser vivo. Mientras que al código genético que heredará las características de generación en generación se le llama *genotipo*, al cuerpo que construye esas proteínas, modificado por la presión ambiental, la historia vital, y otros mecanismos dentro del cromosoma, se le llama *fenotipo*.

2.2.3 Teoría sintética de la evolución.

La combinación de los principios de la genética mendeliana y la teoría de la evolución de Darwin se conoce como teoría *neo darwiniana o Teoría sintética de la evolución*, llamada así por Julián Huxley, en 1942.

Esta teoría dice que la historia de la mayoría de la vida está causada por una serie de procesos que actúan en y dentro de las poblaciones: reproducción, mutación, competición y selección. La evolución se puede definir entonces como cambios en el conjunto genético de una población.

2.2.4 Terminología.

Antes de continuar definiremos algunos términos de uso común en la computación evolutiva:

Cromosoma: Es una estructura de datos que agrupa una serie de parámetros para representar la solución a un problema. Generalmente se representa con una cadena binaria, aunque pueden ser usados números reales o letras del alfabeto entre otras representaciones.

Gen: Un cromosoma está conformado por genes, donde cada gen representa un parámetro del problema.

Alelo: Es el conjunto de valores que puede tomar cada gen. Por ejemplo, en cadenas binarias 0 y 1, en cadenas de reales un número real, etc.

Aptitud: Es un valor que se asigna a un cromosoma de acuerdo a sus genes para indicar que tan bien o que tan mal resuelve el problema al que se enfrenta.

Función de aptitud: Evalúa el desempeño de un cromosoma dándole un valor de acuerdo a su aptitud.

Cruzamiento: Representa la reproducción sexual de los individuos y es una operación genética donde se busca intercambiar genes entre los individuos.

Mutación: Es una operación genética que modifica los genes de un cromosoma de manera aleatoria.

Bloques constructores: Es un conjunto de genes vecinos en la estructura del cromosoma, que después de evolucionar durante varias generaciones permanecen sin cambio. De tal manera que si este bloque se inserta en otro cromosoma el resultado será una mejora en su aptitud [43].

Esquema: Un esquema es un patrón de valores en los genes de un cromosoma y se construye substituyendo genes por comodines, permitiendo la búsqueda de similitudes entre los cromosomas. Los cromosomas con esquemas pequeños, de bajo orden y aptitud arriba del promedio reciben un incremento de representantes en las siguientes generaciones de un algoritmo genético.

Elitismo: Consiste en preservar al mejor individuo de una población poniendo directamente una copia de él en la siguiente población. Si en un algoritmo genético se aplica el elitismo en cada generación, la última población siempre contendrá al mejor individuo encontrado en el proceso.

2.3 Programación evolutiva.

Fogel, Owens y Walsh fueron los creadores de la programación evolutiva en 1966, una técnica en la cual las candidatas a soluciones a tareas determinadas eran representadas por autómatas celulares. Un autómata celular es un conjunto de estados y reglas de transición entre ellos, de forma que, al recibir una entrada, cambia o no de estado y produce una salida. Estos estados de transición se hacían evolucionar mediante mutación aleatoria, seleccionándose el que mejor se aproxime.

El planteamiento es muy similar al de los algoritmos genéticos pero en este caso el método de reproducción no incluye el cruce entre individuos sino únicamente la mutación; es decir, para generar descendientes, un determinado individuo se duplica a sí mismo y posteriormente sufre algunas mutaciones. Se suele trabajar con factores de mutación variables. Eliminar el cruce permite que en programación evolutiva la representación de los individuos pueda ser bastante más compleja y flexible. La codificación de un individuo en un algoritmo genético trata de imitar a un genoma y por tanto una vez codificado la longitud, ésta es constante. En programación evolutiva se pueden utilizar estructuras complejas como redes neuronales, grafos, árboles, etc. para representar a los individuos. Para cada problema en particular se deberá elegir una representación que se adapte convenientemente.

2.4 Programación genética.

La programación genética es una rama de la computación evolutiva en la que los individuos que componen la población son programas ejecutables en vez de secuencias de bits como en los algoritmos genéticos. Esta nueva variante fue presentada por John R. Koza en 1982. Los programas se representan como árboles sintácticos. En estos árboles, una llamada a una función viene representada por un nodo del árbol y los argumentos de la función son sus nodos hijos. Para ello se eligió el lenguaje LISP (List processing), que se puede representar de forma unívoca como un árbol; la raíz es el resultado de la evaluación, y las ramas contienen como símbolos terminales las variables del problema.

El proceso comienza definiendo una serie de funciones con las que se quiere realizar el programa. A continuación se genera una población inicial de árboles que utilizan estas funciones. Sobre esta población se aplican las mismas operaciones que en los algoritmos genéticos, es decir, cruce, mutación y selección. La operación de reproducción funciona intercambiando subárboles entre distintas soluciones, la mutación cambia aleatoriamente alguno de los nodos, y para la selección de la siguiente generación se utiliza, al igual que en los algoritmos genéticos, una función de evaluación capaz de evaluar la aptitud de los programas. Típicamente la función de evaluación suministrará al programa una serie de datos de entrada y comprobará las salidas con los resultados deseados.

Por esta razón, la programación genética no hace evolucionar programas, sino funciones multivariadas, pero con una sola salida, lo cual limita su uso. Se puede usar para generar, por ejemplo, circuitos, o para realizar regresiones simbólicas, es decir, generar una función de formato libre que aproxime los datos obtenidos en un experimento o modelo.

2.5 Clasificadores genéticos.

Los sistemas clasificadores fueron presentados por John Holland en 1975. Básicamente son sistemas basados en reglas (clasificadores) diseñados para interactuar con su entorno, y aprender de él mediante la asignación de pesos a cada regla y la creación de nuevas reglas a partir de las anteriores.

Las reglas tienen la forma condición->acción, identificando una condición con una acción a realizar en caso de que la condición se cumpla. Un clasificador genético está dividido en tres subsistemas, *el sistema de rendimiento, el sistema de asignación de crédito y el sistema de descubrimiento de clasificadores*. Su funcionamiento es el siguiente:

En primer lugar se deben generar una serie de reglas iniciales a partir de los conocimientos que se tengan sobre el problema a tratar, y asignar a estas reglas un peso. Desde este punto el sistema trabajará sólo a partir de la información que reciba de su entorno. Para ello comparará la información recibida con las condiciones de los clasificadores, y seleccionará aquellos cuyas condiciones se cumplan.

Los clasificadores seleccionados indicarán una serie de acciones a realizar; estas acciones pueden contradecirse entre sí por lo que el sistema deberá detectar las incompatibilidades, y en caso de qué se produzcan, seleccionar una de las acciones. A la hora de elegir se estudian los pesos asignados a cada acción dando preferencia al clasificador con mayor peso. Toda esta funcionalidad recae en el subsistema de rendimiento.

Hasta aquí el proceso es totalmente estático. Para ser capaz de evolucionar y aprender dinámicamente, un sistema clasificador aplica dos técnicas: Por un lado el

sistema se realimenta con la información que recibe del exterior, y si considera que las decisiones que se tomaron fueron acertadas, las premia incrementándoles el peso. Esto se realiza a través del subsistema de asignación de crédito quien es el responsable de la asignación y actualización de los pesos. Por otro lado se utiliza un algoritmo genético para crear nuevas reglas a partir de las mejores reglas existentes. Esto será llevado a cabo por el subsistema de descubrimiento de clasificadores. Las nuevas reglas sustituirán a las que tengan los pesos inferiores. El peso de las nuevas reglas se calcula a partir del peso de sus padres; esto será realizado por el subsistema de asignación. Para saber si una regla es útil es necesario observar cómo se comporta en un periodo de tiempo por lo que no se generan nuevas reglas constantemente sino cada cierto intervalo, de esta forma todas las reglas tienen la oportunidad de demostrar su aptitud.

2.6 Algoritmos genéticos.

Los algoritmos genéticos son una rama de los algoritmos evolutivos que surgen a principios de los sesentas como resultado de los trabajos de investigación de Jhon Holland y un grupo de estudiantes en la universidad de Michigan.

Los algoritmos genéticos son métodos sistemáticos para la resolución de problemas de búsqueda y optimización que aplican los mismos métodos de la evolución biológica: Selección basada en la aptitud de la población, reproducción sexual y mutación.

Las posibles soluciones se codifican en un cromosoma. Todos los operadores utilizados por un algoritmo genético se aplicarán sobre estos cromosomas, o sobre poblaciones de ellos. En el algoritmo genético va implícito el método para resolver el problema, a diferencia de otros algoritmos evolutivos como la programación genética. Hay que tener en cuenta que un algoritmo genético es independiente del problema, lo cual lo hace un algoritmo robusto, por ser útil para cualquier problema, pero a la vez débil, pues no está especializado en ninguno.

Las soluciones codificadas en los cromosomas compiten para ver cuál constituye la mejor solución aunque esta no necesariamente sea la mejor posible. De las posibles soluciones sólo las mejores adaptadas sobrevivirán y heredarán su material genético

a las siguientes generaciones, igual que en la evolución de las especies. La diversidad genética se introduce mediante mutaciones y cruzamientos.

Resumiendo, un algoritmo genético consiste en:

- Codificar en una cadena o cromosoma las posibles soluciones al problema generando aleatoriamente una población inicial.

- Tener una función que permita evaluar la aptitud o eficiencia de los cromosomas.

- Se selecciona a los individuos más aptos para que sean los padres de la siguiente generación y se asignan algunas copias a la siguiente generación de manera proporcional.

- Se hace un cruzamiento entre los padres seleccionados (simulando la reproducción sexual).

- A los hijos resultantes se les aplicará un operador de mutación, conformando así la nueva generación.

- El proceso se repite desde el segundo paso hasta que el algoritmo cumpla con algún criterio que lo detenga.

2.6.1 La codificación.

El algoritmo genético requiere que las posibles soluciones se codifiquen en cromosomas. Cada cromosoma tiene varios genes, que corresponden a parámetros del problema. El número de bits usado para cada parámetro dependerá de la precisión que se quiera en el mismo o del número de opciones posibles (alelos) que tenga ese parámetro. La representación puede hacerse de varias maneras y no necesariamente en bits.

2.6.2 La evaluación.

La función de aptitud es donde se evalúan los cromosomas para determinar cual de ellos es el más apto. Durante la evaluación, se decodifica el gen, convirtiéndose en un conjunto de parámetros que da solución al problema. A esta posible solución al problema se le da una puntuación de acuerdo a que tan cerca esté de la mejor solución. Esta puntuación es llamada *aptitud*.

2.6.3 La selección.

Una vez que se evaluaron los cromosomas hay varios métodos para seleccionar a los individuos más aptos. La selección generalmente se hace de manera probabilística de tal manera que hasta los individuos menos aptos tengan oportunidad de sobrevivir. Los métodos más populares son:

Selección por ruleta: Se trata de una ruleta dividida en rebanadas, el tamaño de la rebanada es proporcional a la aptitud de los individuos, es decir, entre más aptos, el tamaño de la rebanada es mayor. La ruleta se lanza tantas veces como individuos tenga la población; a la siguiente generación pasará un individuo con la aptitud señalada por la aguja de la ruleta.

Selección estocástica universal: Propuesto por James Baker en 1987, similar a la de la ruleta pero en este caso tiene tantos apuntadores como individuos vayan a ser seleccionados, distribuidos uniformemente al rededor de la ruleta. Ésta es lanzada y los apuntadores indicarán que individuos formarán la siguiente generación.

Selección por torneo: Se seleccionan aleatoriamente dos individuos, se define un parámetro entre 0.5 y 1, después se genera un número aleatorio entre cero y uno que será comparado con el parámetro. Si el número aleatorio generado es menor que el parámetro, el individuo con mayor aptitud será seleccionado para la siguiente generación, de lo contrario el de menor aptitud será seleccionado.

Selección por estado uniforme: Propuesto por Whitley en 1989[8] donde en cada generación, los menos aptos son reemplazados por descendientes obtenidos de los individuos más aptos.

2.6.4 El cruzamiento.

El cruzamiento equivale a la reproducción sexual de los individuos y consiste en el intercambio de genes entre los cromosomas.

Se puede cruzar en un punto, que se hace seleccionando de manera posicional un gen en el cromosoma. A partir de ese punto, el resto de la cadena es intercambiada con otro individuo.

La cruza en dos puntos consiste en seleccionar dos puntos de manera aleatoria en el cromosoma, el segmento de cromosoma entre los dos puntos es intercambiado con un segmento similar de otro cromosoma. También puede haber cruzamiento múltiple, donde varios segmentos de cromosoma son intercambiados con otro.

2.6.5 La mutación.

La mutación puede ser unitaria, donde se genera un número aleatorio para seleccionar el gen a modificarse y éste es cambiado por alguno de los alelos, o puede ser una mutación múltiple donde se seleccionan al azar varios genes y estos son modificados por alguno de los alelos.

Tanto la mutación como el cruzamiento se suelen aplicar de manera probabilística, por lo que se requieren de parámetros como la probabilidad de cruzamiento y la probabilidad de mutación.

2.7 Estrategias evolutivas.

Esta técnica evolutiva dirigida particularmente a la optimización de funciones continuas de alta complejidad se desarrolló en Alemania, en 1965, por Rechenberg [24] y Schwefel [20] [21]. Esta técnica, se utilizó inicialmente para resolver problemas de ingeniería que resultaban difíciles de resolver por los métodos de

optimización tradicionales, como el gradiente conjugado, y se basa en la modificación sistemática de un vector de números reales mediante operadores probabilísticos, usando ciertos criterios para decidir en qué dirección dirigir la búsqueda. La estrategia evolutiva utiliza como operador principal a la mutación, y en su versión más reciente usa la cruza como operador secundario. La técnica original constaba de dos individuos, un padre y un descendiente mutado desde el padre. El individuo con mejor aptitud era seleccionado para ser el padre de la siguiente generación. La mutación consistía en incrementar o decrementar un valor real de acuerdo a una distribución dada. Los parámetros de la distribución eran codificados como parte del individuo y coevolucionaban con él.

2.8 Optimización numérica de funciones.

Se trata de encontrar el punto o puntos en una función multidimensional que den el máximo, o mínimo, valor para esa función. En algunos casos, tal función puede estar definida en todo el dominio, puede haber zonas del espacio en las cuales no esté definida, o puede que esté definida en secciones. En general, casi todo problema de búsqueda, formulado como un problema de optimización, suele tener varios óptimos, pero uno de ellos es mejor que el resto y se denomina óptimo global. El resto son óptimos locales; es decir, se puede definir una vecindad alrededor de ellos en la cual son óptimos globales.

2.9 Manejo de restricciones.

Una *restricción* es simplemente una condición que tiene que cumplir la solución a un problema. En la optimización numérica las restricciones pueden ser igualdades o desigualdades que acotan el área de búsqueda en algo que se conoce como regiones factibles. El manejo de restricciones es entonces la capacidad del algoritmo para encontrar el óptimo dentro de las regiones factibles.

Existen varias opiniones sobre cómo manejar las restricciones con algoritmos genéticos, Michalewicz [43] las clasifica en:

- Uso de funciones de penalización.
- Mantenimiento de una población factible por representaciones y operadores genéticos especiales.
- Separación de objetivos y restricciones.
- Rechazo de individuos no factibles.

2.10 Técnicas clásicas de optimización.

En general, los métodos de búsqueda se dividen, en métodos *globales* y *locales*.

Los métodos globales tratan de encontrar el óptimo global de un problema, mientras que los locales se concentran en la vecindad de la solución generada inicialmente, y, por tanto, necesitan alguna técnica adicional, como comienzos múltiples, para acercarse al óptimo global.

Los métodos locales no tienen ninguna garantía de que el óptimo encontrado sea global, ni una medida del error en que se incurre dando como bueno el óptimo encontrado; los globales sí suelen tener una de las dos cosas, o las dos.

2.10.1 Búsqueda exhaustiva.

En algunos problemas donde el espacio de búsqueda es suficientemente pequeño y enumerable, se pueden usar métodos de *búsqueda exhaustiva* o *enumerativas*: se examinan una por una todas las posibles soluciones, y se da por buena la que tenga el valor óptimo al evaluarse en la función objetivo.

2.10.2 Búsqueda local.

Los algoritmos de búsqueda local parten de una solución inicial, y, aplicándole operadores de variación, la van alterando; si la solución alterada es mejor que la original, se acepta, si no lo es, se vuelve a la inicial. El procedimiento se repite hasta que no se consigue mejora en la solución.

Los procedimientos de búsqueda local más usados son los basados en el gradiente: En este caso, el operador de variación selecciona una nueva solución teniendo en cuenta la derivada de la función que se quiere optimizar en un punto, tratando de

ascender o descender usando el gradiente hasta llegar a un punto de inflexión donde no se puede obtener ninguna mejora adicional. Estos procedimientos se suelen llamar de *hillclimbing*, o de escalado de colinas.

2.10.3 Métodos que usan soluciones parciales.

Estos métodos van construyendo soluciones variable a variable, no obteniendo una solución completa al problema hasta que todas las variables han sido asignadas. Uno de los métodos más conocidos son los algoritmos *voraces* o *greedy*, donde para cada variable, se asigna aquél valor que maximice alguna función de utilidad.

Los algoritmos *voraces* sustituyen un criterio de optimalidad global por un criterio en cada paso; no siempre los dos se corresponden, y por tanto, no tienen porqué hallar el óptimo global del problema.

Otros métodos consisten en dividir el problema general en problemas más simples, y hallar la solución a dichos problemas.

2.10.4 Búsqueda global.

Uno de los más conocidos es el *recocido simulado*, o *simulated annealing*. A grandes rasgos, consiste en un procedimiento de ascenso de gradiente, pero tal que no siempre se escoge la mejor solución; dependiendo de una *temperatura*, la probabilidad de escoger una solución peor que la actual va descendiendo con el tiempo, hasta que al final del algoritmo, cuando la temperatura es 0, se escoge de forma determinista siempre la mejor solución.

Otros métodos, como el Procedimiento de Búsqueda Adaptable Aleatoria Voraz en ingles Greedy Randomized Adaptive Search Procedure o GRASP, se basan en la combinación de fases de explotación, en los cuales se siguen algoritmos de búsqueda local, con fases de exploración, en los cuales se usan procedimientos aleatorios para expandir el espacio de búsqueda recorrido.

El algoritmo GRASP mantiene una lista de posibles soluciones a un problema, que se genera aleatoriamente. Se escoge una de las soluciones, y se mejora hasta que

no se pueda mejorar más; sí es mejor que la mejor almacenada, se actualiza la lista de soluciones almacenadas.

Un algoritmo similar, denominado *búsqueda tabú*, incluye una memoria que almacena los últimos movimientos realizados, y que se usa para no volver a caer en los mismos óptimos locales. A esta memoria se le denomina *lista tabú*. Pero también tiene un método que permite escapar de los óptimos locales, que se denomina *nivel de aspiración*; consiste en un criterio para aceptar movimientos incluidos en la lista tabú. Aparte de esa memoria, que se podría llamar de corto plazo, diversas variantes de la búsqueda tabú incluyen memorias a mediano plazo, registrando los atributos más comunes de un conjunto de soluciones, para poder explotar esa zona del espacio de búsqueda, y a largo plazo, que diversifique la búsqueda sobre regiones que no se han explorado con anterioridad.

2.11 Ventajas del algoritmo evolutivo sobre las técnicas clásicas.

Las ventajas que tiene el algoritmo evolutivo sobre los métodos clásicos son:

- El algoritmo evolutivo se puede aplicar sobre grandes espacios de búsqueda mientras que el de búsqueda exhaustiva esta orientado a pequeños espacios de búsqueda.
- En la técnica de búsqueda local, a veces se requiere de derivar las funciones y saber de gradientes mientras que en el evolutivo no.
- El de soluciones parciales divide el problema general en problemas más simples, los problemas no siempre pueden ser divididos y las soluciones parciales no siempre dan una buena solución global; con el algoritmo evolutivo el problema no necesita dividirse.
- En el de búsqueda global se requiere de un manejo de listas para saber donde ya se buscó y cuales son los valores óptimos, mientras que en el evolutivo esto no se requiere.
- En la técnica que usa derivadas y en la que se divide el problema en problemas parciales, los procesos son particulares para un problema específico. Por otro lado el algoritmo evolutivo es aplicable a varios problemas y no a uno en particular.

Por estas razones, creo que el algoritmo evolutivo es una mejor opción para resolver problemas de optimización numérica.

Capítulo 3

Sistema Inmune.

En este capítulo se hará una breve descripción del funcionamiento del sistema inmune y algunos de sus mecanismos con el fin de describir la metáfora entre el mecanismo de maduración de la afinidad y el algoritmo propuesto.

En latín, *immunitas e immunis* se refieren a una excepción legal, existente en la Roma antigua, para prestar un servicio o cumplir con una obligación con el Estado; sin embargo, en algunos escritos de la época ya se utilizan estos términos para referirse a la resistencia a ciertos venenos o pestes.

La principal tarea que tienen los sistemas inmunes es defender al organismo del que forman parte, de los ataques que le pueden causar los agentes *patógenos* (que pueden causar una enfermedad).

Los agentes patógenos los podemos dividir en *patógenos exógenos* (que vienen de afuera del organismo) entre los que se pueden poner como ejemplo microorganismos tales como bacterias, hongos, virus, protozoarios y las substancias que secretan

como lo son las toxinas y otras substancias producidas en el medio ambiente como venenos naturales, substancias químicas artificiales, etc. Los *patógenos endógenos* (que se forman dentro del organismo) se pueden poner como ejemplo las células cancerígenas que se generan dentro del mismo organismo o las partes de un virus que son producidas por células infectadas por éste.

Algunos de los mecanismos usados por el sistema inmune de los animales vertebrado con mandíbula (SIM), que es el sistema inmune más evolucionado [19], han sido tomados como modelo para generar algoritmos que permitan resolver problemas de optimización numérica. Estos algoritmos bioinspirados forman parte de los llamados *Sistemas Inmunes Artificiales*.

3.1 Sistema inmune.

El sistema inmune puede ser dividido en dos partes una respuesta inmune evolutivamente más antigua, que se ha denominado *respuesta inmune innata*, y otra respuesta, exclusiva de los vertebrados, que se superpone con la respuesta innata, denominada *respuesta inmune adquirida*, y ambas se complementan para cumplir la función de defender al organismo de agentes patógenos [23].

3.1.1 Respuesta inmune innata.

La inmunidad innata es un conjunto de barreras que no permiten la entrada de materiales nocivos al individuo, estas barreras pueden ser anatómicas, fisiológicas, fagocíticas e inflamatorias [16] formando así la primera línea de defensa de la respuesta inmune.

Algunas de estas barreras son la piel, el ácido estomacal, la mucosa (atrapa microorganismos y partículas pequeñas), el reflejo de la tos y enzimas en las lágrimas y las grasas de la piel. Si un antígeno traspasa las barreras externas, es atacado y destruido por otras partes del sistema inmunológico.

El sistema inmunológico incluye ciertos tipos de glóbulos blancos, al igual que sustancias químicas y proteínas en la sangre (como proteínas del complemento e

interferón), algunas de las cuales atacan directamente a las sustancias extrañas en el individuo y otras trabajan juntas para ayudar a las células del sistema inmunológico.

La respuesta inflamatoria (inflamación) es parte de la inmunidad innata y se presenta cuando los tejidos son lesionados por bacterias, trauma, toxinas, calor o cualquier otra causa. Las sustancias químicas incluyendo histamina, bradiquinina, serotonina y otras, son liberadas por el tejido dañado y hacen que los vasos sanguíneos derramen líquido en los tejidos, lo que deriva en una inflamación localizada. Esto ayuda a aislar la sustancia extraña del contacto con otros tejidos corporales.

Las sustancias químicas también atraen a los glóbulos blancos que se "comen" a los microorganismos y células muertas o dañadas. El proceso por el cual estos glóbulos blancos rodean, sumergen y destruyen las sustancias extrañas se llama *fagocitosis* y las células son colectivamente llamadas *fagocitos*, las cuales finalmente mueren.

3.1.2 Respuesta inmune adquirida.

A diferencia de la inmunidad innata, la inmunidad adquirida se desarrolla cuando el individuo está expuesto a varios antígenos y se construye una defensa que es específica para dicho antígeno. Los *linfocitos*, un tipo especial de glóbulos blancos, contienen subgrupos de linfocitos B y T que son actores claves en las respuestas inmunes adquiridas. Los linfocitos B (también llamados células B) producen anticuerpos, los cuales se adhieren a un antígeno específico y facilitan la destrucción del antígeno por parte de los fagocitos. Los linfocitos T (células T) atacan los antígenos directamente y proporcionan control de la respuesta inmune. Las células B y T se desarrollan específicamente para un tipo de antígeno y cuando hay exposición a un antígeno diferente, se forman células B y T diferentes.

A medida que los linfocitos se desarrollan, aprenden normalmente a reconocer los tejidos que son parte del individuo (propio) y a distinguirlos de los tejidos y partículas que no se encuentran normalmente en él (no propio). Una vez que se

forman las células B y T, algunas de ellas se multiplican y brindan "*memoria*" para el sistema inmunológico, lo que le permite responder más rápida y eficientemente la próxima vez que sea expuesto al mismo antígeno, y en muchos casos previene al individuo de enfermarse.

3.2 Características de la respuesta inmune adquirida.

La respuesta inmune adquirida frente al ataque de agentes extraños, reúne ciertas características que reflejan las propiedades de los linfocitos encargados de dicha respuesta.

• *Especificidad y diversidad*: Las respuestas inmunes son específicas para los diferentes antígenos. Las distintas partes de cada antígeno que son reconocidos de forma específica por linfocitos individualizados reciben el nombre de determinantes antigénicos o epitopos. Esta *especificidad* existe porque cada linfocito presenta receptores de membrana que son capaces de distinguir diferencias estructurales mínimas entre los distintos antígenos.

El número total de especificidades antigénicas de los linfocitos de un individuo se denomina repertorio linfocitario y se calcula que el sistema inmune de un individuo puede discriminar entre diez millones y mil millones de determinantes antigénicos diferentes. Esta propiedad del repertorio se denomina *diversidad*.

• *Memoria*: La exposición del sistema inmune a un antígeno extraño aumenta su capacidad de respuesta ante un nuevo contacto con tal antígeno. Las respuestas a una segunda y subsiguientes exposiciones al mismo antígeno, denominadas respuestas secundarias, suelen ser más rápidas y efectivas.

• *Especialización*: Como ya se mencionó, el sistema inmune responde de manera individualizada y especial a cada diferente microorganismo, lo que aumenta al máximo la eficacia de los mecanismos de defensa ante la presencia de algún antígeno. Se dice que la respuesta es especializada porque no se utilizan todos los mecanismos de que dispone la respuesta inmune en cada infección, sino solamente los que son más eficaces.

• *Autolimitación*: Todas las respuestas inmunes normales disminuyen con el tiempo después de la exposición al antígeno. De esta manera, el sistema inmune recupera su estado normal, este proceso es denominado homeostasis. Si no existieran los mecanismos de homeostasis, el sistema estaría permanentemente activado, lo que llevaría a su desgaste o, podría producir daño en el propio individuo.

• *Ausencia de autorreactividad*: Una de las características más importantes del sistema inmune es su capacidad para reconocer, responder y eliminar diferentes antígenos extraños (no propios) al tiempo que no reacciona de forma dañina frente a las propias sustancias antigénicas del individuo. Esta ausencia de respuesta inmune se denomina también *tolerancia*. La tolerancia a antígenos propios, o auto tolerancia se logra a través de mecanismos, que consisten en la eliminación de linfocitos que presentan receptores específicos para algunos antígenos propios.

3.3 Linfocitos

Los linfocitos provienen de las células madre que se encuentran en la medula ósea y son de tres tipos: Linfocitos B, linfocitos T y linfocitos NK. Estos últimos tienen una función de vigilancia en el sistema. A su vez los linfocitos T pueden ser de tres tipos: Linfocitos T colaboradores (Th), linfocitos T citotóxicos (Tc) y los linfocitos T supresores / reguladores (Ts).

3.3.1 Linfocitos B.

Representan cerca del 5-15% de todos los linfocitos circulantes en el individuo. En el feto, se producen en el hígado y después en la médula ósea. Las células B producen anticuerpos llamados inmunoglobulinas. Estos anticuerpos son destinados a reaccionar contra los antígenos específicos que se introducen en el sistema por medio de varios gérmenes. Los anticuerpos reaccionan contra los virus, la levadura y las bacterias y permiten a los glóbulos blancos eliminarlos. Los anticuerpos, compuestos principalmente de aminoácidos, son proteínas que pueden ser divididas en cinco grandes clases de anticuerpos llamados inmunoglobulinas A, G, M, D y E

(IgA, IgG, IgM, IgD y IgE). Cada uno tiene una estructura química única y una función especifica.

3.3.2 Linfocitos T.

Se crean en la médula ósea y se desarrollan en el timo [29] a partir de células madre linfocíticas de origen embrionario. Después presentan receptores antigénicos específicos y se diferencian en dos subgrupos. Uno presenta el receptor proteico CD4 (células T colaboradoras) y el otro grupo el CD8 (células T citotóxicas). A su vez, constituyen diferentes poblaciones que son: Los linfocitos T colaboradores, los citotóxicos y los supresores. Sus funciones son:

- Ayudar a las células B a producir anticuerpos.
- Reconocer y destruir a los patógenos.
- Controlar el nivel y la calidad de la respuesta inmunológica.

3.3.3 Receptores.

Los receptores son proteínas de forma alargada pegadas a la membrana celular de los linfocitos. Estas proteínas son una expresión del material genético del individuo. El receptor esta conformado por la unión de diferentes proteínas y se dividen en dos regiones una variable en la parte superior del receptor y una constante en la parte inferior.

La respuesta inmune adquirida se basa en el reconocimiento específico de los antígenos, tanto en la respuesta de linfocitos B, como en la de los T. Para este reconocimiento específico hay unas proteínas fundamentales en cada tipo de linfocito, mientras que para los linfocitos T estas cadenas de proteínas se encuentran pegadas a la membrana celular y se conocen como *Receptor de la célula T* (TCR), en los linfocitos B las proteínas son las inmunoglobulinas.

Las inmunoglobulinas son un grupo grande de proteínas presentes en el suero y otros fluidos del organismo. Se pueden encontrar de 2 formas principales: forma

soluble, entonces las denominamos anticuerpos, o pegadas a la membrana de los linfocitos B. En este último caso, forma parte de un complejo de proteínas que en su conjunto se denomina *Receptor de la célula B* (BcR).

3.3.4 Antígenos y epitopos.

- *Antígeno* (Ag) es toda sustancia capaz de inducir una respuesta inmune. Los antígenos pueden enlazarse con los receptores de los linfocitos y se subdividen en 3 tipos principales: tolerógenos, inmunógenos y haptenos.

- *Inmunógenos*, capaces de inducir por sí solos respuesta inmune y convertirse en blanco de dicha respuesta. Exposiciones secundarias al inmunógeno conducen a respuestas más enérgicas y rápidas (respuestas secundarias o de memoria).

- *Haptenos*, no son capaces por sí solos de inducir la respuesta inmune. Al unirse a un inmunógeno mayor, se puede generar respuestas tanto frente al inmunógeno como frente al hapteno. Exposiciones repetidas al complejo inmunógeno-hapteno conducen a respuestas enérgicas y rápidas.

- *Tolerógenos*, capaces de inducir por sí solos respuesta inmune pero sucesivas exposiciones al antígeno van provocando respuestas primarias de menor intensidad posteriormente.

- Epitopo o determinante antigénico, es la unidad más pequeña de un antígeno que puede ser identificada por un anticuerpo específico o por un TCR específico, y es la parte de los antígenos que puede enlazarse con los receptores de los linfocitos o con un anticuerpo. Los epitopos pueden ser:

- *Epitopo lineal*: Una secuencia lineal de aminoácidos dentro de una proteína.

- *Epitopo conformacional*: Aminoácidos que no están contiguos en la estructura lineal, se organizan espacialmente muy próximos, debido al plegamiento de la proteína.

3.3.5 Afinidad.

La afinidad de un anticuerpo por un epitopo es la suma de todas las fuerzas atractivas y repulsivas entre el sitio de unión de ese anticuerpo y el correspondiente epitopo. Al existir una afinidad alta el anticuerpo y el antígeno se atraen entre sí.

3.4 Generación y maduración de los linfocitos.

La generación de los linfocitos T y B se lleva a cabo en la médula ósea que está formada por islotes de células madres situados en el interior de los huesos. Todas las células del sistema inmune se originan a partir de las células madres de la médula ósea a través de los linajes mieloide y linfoide.

Durante la edad fetal estas funciones se realizan por el hígado, que abandona esta actividad después del nacimiento. Además, la médula ósea actúa como órgano linfoide secundario donde se hace la maduración final de células B a células plasmáticas.

Las células B inmaduras tienen una vida corta (entre 3 y 4 días), y durante este período se produce el proceso de *selección negativa* donde son eliminados aquellos linfocitos cuyos receptores reconocen moléculas del propio organismo (auto-antígenos). En respuesta al autoantígeno la célula B inmadura hace un intento de no reaccionar a lo propio codificando los genes en sus receptores, si no lo logran serán destruidas pero de hacerlo se convertirán en linfocitos B maduros.

La maduración de los linfocitos T se lleva a cabo en el timo donde los timocitos o linfocitos T inmaduros (sin receptores) llegan por vía arterial de la medula ósea a la corteza del timo donde comienzan a dividirse activamente y a través de los capilares pasan a la médula del timo. De ahí salen por los capilares venosos hacia el torrente sanguíneo.

Los linfocitos T se maduran en el trayecto de la corteza a la médula del timo. En el proceso de maduración los timocitos interactúan con células que en sus membranas están provistas de una proteína llamada MHC (complejo principal de histocompatibilidad) que es un grupo de genes estrechamente ligados y muy polimórficos.

Aquí se producen dos fases de selección de timocitos:

- Selección positiva: Sólo sobreviven aquellos timocitos que hayan generado receptores TCR capaces de reconocer moléculas MHC propias; los demás mueren por apoptosis (Es un proceso del sistema inmunológico para deshacerse de células innecesarias, mediante el cual se fragmenta el cromosoma de la célula y luego toda la célula se rompe).

- Selección negativa: Se eliminan los timocitos que habiendo superado la selección positiva, hayan resultado autoreactivos a moléculas propias presentadas por el MHC o que tengan una afinidad demasiado alta hacia el MHC propio.

De esta forma sólo salen como linfocitos T maduros aquellas células autotolerantes y capaces de reconocer antígenos.

3.5 Maduración de la afinidad.

Este término se refiere a la forma en la que los linfocitos B aumentan significativamente su afinidad por los epitopos de patógenos al migrar hacia los nodos linfáticos o el bazo. Existe la hipótesis de que la evolución creó una presión selectiva que favoreció la aparición de secuencias tales como las reordenadas que codifican la porción variable de los receptores, que hacen que hacia ellas se dirijan mutaciones con alta frecuencia; dichas mutaciones crean a su vez variantes en las posiciones más útiles para generar variedades de anticuerpos nuevos que luego serán sometidas en el individuo a un proceso selectivo de tipo darwiniano, responsable de la maduración de afinidad de los anticuerpos.

Aquí están involucradas la hipermutación somática donde los linfocitos B se multiplican por división celular en los centros germinales, en la llamada zona obscura de los nodos linfáticos y el bazo, y la selección negativa en los centros germinales de la llamada zona clara de dichas estructuras. De las numerosas variantes aleatorias que van surgiendo en los linfocitos, algunas son de mayor afinidad que el original, y otros serán de menor afinidad. A partir de ahí, el antígeno dirige un proceso selectivo que va enriqueciendo las poblaciones de linfocitos con receptores de mayor afinidad;

proliferando más aquellos linfocitos con versiones de receptores de mayor afinidad. Este fenómeno es lo que se conoce como *maduración de la afinidad*.

3.6 Respuesta inmune humoral.

En la respuesta inmune humoral los linfocitos CD4 activan a los linfocitos B, mismos que están encargados de elaborar anticuerpos neutralizantes dirigidos contra el virus invasor.

Esta se inicia cuando el epitopo de un antígeno se enlaza con gran afinidad a un receptor de un linfocito B. Esto provoca una expansión monoclonal que es el proceso de hacer copias exactas del linfocito B. Las regiones variables de los receptores de un linfocito B clonado, son idénticas a las del linfocito B original.

Estos clones recorren el organismo a través del torrente sanguíneo, si alguno de los receptores de estos clones se enlaza con otro epitopo igual al que provocó la primera expansión monoclonal, se llevará a cabo otra de estas expansiones.

Las células B que sufren la expansión monoclonal, secretan substancias químicas que se difunden en el organismo.

Estas substancias indican a todos los clones que los antígenos siguen atacando al sistema y que la defensa debe continuar. Los clones dan origen a su vez a dos tipos de células: Las *células plasmáticas* y las *células B de memoria*.

Las células plasmáticas se encargan de fabricar inmunoglobulinas, con sus regiones variables idénticas a las de las inmunoglobulinas de la membrana celular de los linfocitos B, pero éstas son moléculas que recorren libremente el organismo.

3.7 Respuesta inmune celular.

En la respuesta inmune celular los linfocitos CD4 activan a los linfocitos CD8 y a los linfocitos llamados linfocitos T citotóxicos específicos a virus, que se encargarán de reconocer y eliminar células cancerosas o infectadas por virus.

Los linfocitos T reconocen el antígeno mediante el receptor sólo cuando el antígeno es degradado y procesado en el interior de las células presentadoras de antígeno (APC) y sus determinantes antigénicos son expuestos en la superficie de estas células en una molécula MHC.

Cuando tiene lugar el reconocimiento antigénico entre el TcR y la molécula MHC que porta el antígeno, el linfocito T queda activado y se desencadena una serie de reacciones bioquímicas en el citoplasma de la célula T, dando así lugar al proceso de activación, proliferación y maduración celular.

La consecuencia final de este tipo de respuesta es la formación de células Th activas productoras de inteleucinas y células citotóxicas que poseen capacidad de atacar a las células que portan el antígeno que indujo su activación.

Capítulo 4

Algoritmo Evolutivo Propuesto.

En capítulos anteriores se han descrito a grandes rasgos algunos de los mecanismos del sistema inmune. Inspirado en la afinidad entre anticuerpos y epitopos, el mecanismo de generación y maduración de los receptores de los linfocitos y la respuesta inmune, se desarrolló un Algoritmo evolutivo llamado algoritmo *"MARACoR"* (Maduración de Afinidad y Receptores de Anticuerpos Codificado en Reales). El Algoritmo Evolutivo propuesto tiene como objeto ser aplicado a la optimización de funciones restringidas en problemas de programación no lineal (*NLPP* por sus siglas en inglés). Este trabajo es propuesto como una línea de investigación en el proyecto de tesis doctoral de Hernández [36]. En [36] se propone el algoritmo genético "AMRE" (Affinity Maturation Receptor Editing).

4.1 El algoritmo MARACoR.

El algoritmo MARACoR toma como inspiración principal la maduración de la afinidad que sufren los receptores de los linfocitos B en los nodos linfáticos y el bazo, que como se menciona en un capítulo anterior, consiste en las mutaciones de las

mínimas unidades de almacenamiento del genoma en la parte variable de los receptores del linfocito B y los anticuerpos, que se suscitan cada vez que se divide un linfocito. La búsqueda de anticuerpos con mayor afinidad a los epitopos de los patógenos en el sistema inmune es el equivalente de la búsqueda del óptimo en un problema de programación no lineal.

4.1.1 La población de anticuerpos.

El MARACoR se basa en la evolución de poblaciones de linfocitos B *artificiales*. Son artificiales porque no sólo se multiplican por división, es decir asexualmente, como ocurre en los Sistemas Inmunes, sino que también por cruzamiento entre sí, sexualmente. Esto no ocurre en los sistemas inmunes naturales.

Los linfocitos evolucionan de manera independiente uno de otro por medio de maduración de la afinidad, hasta que se determina que deben intercambiar información entre ellos por medio del cruzamiento lo que ayuda a la diversidad.

4.1.2 La Transgresión.

Un anticuerpo representa un punto en el espacio de búsqueda *S*. Si dicho punto se encuentra en la región no factible F' (la región en el espacio de búsqueda *S* que no cumple con las restricciones $g_i(\boldsymbol{x})$ y $h_i(\boldsymbol{x})$), esto significa que es afín a un epitopo propio y que el sistema inmune no actuará en su contra o que se producirá una enfermedad autoinmune. Este tipo de anticuerpos son llamados $autoanticuerpos$. Se requiere que la afinidad por un epitopo (de patógeno o propio) sea *suficiente* para que se dispare la respuesta inmune. Se puede establecer la analogía entre la fuerza de la afinidad de un autoanticuerpo por un epitopo y la distancia de un punto no factible a la región factible *F*. La transgresión es una medida de tal distancia.

Bajo restricciones de igualdad $h_i(\boldsymbol{x}) = 0, \text{para } i = p+1,\ldots,m$ (1.4) es difícil establecer la transgresión por lo que la transgresión puede ser manejada de la siguiente manera:

$$\left|h_i(\boldsymbol{x})\right| - \delta_i = 0, \text{para } i = p+1,\ldots,m \qquad (4.1)$$

donde $\delta_i \in R^+$ y $\delta_i << |h_i(x)|$ es una tolerancia, o sea que (4.1) son aproximaciones de las restricciones de igualdad. De tal forma que:

$$g_i(\boldsymbol{x}) = |h_i(\boldsymbol{x})| - \delta_i \text{ para } i = p+1,\ldots,m \qquad (4.2)$$

Donde ahora tenemos m desigualdades, formadas por las p desigualdades iniciales y las m-p igualdades transformadas en desigualdad.

Por tanto, todas las restricciones son entonces restricciones de desigualdad.

Tomando en cuenta las expresiones $g_i(\boldsymbol{x}) \le 0, \text{para } i = 1,\ldots,p$ (1.3) y 4.2 la transgresión, *transg*, se define como [35]:

$$transg = \sum_{i=1}^{m} \tau_i \quad \text{donde} \begin{cases} \tau_i = 0 & \text{si } g_i(\boldsymbol{x}) \le 0 \\ \tau_i = g_i(\boldsymbol{x}) & \text{de otra manera} \end{cases} \qquad (4.3)$$

4.1.3 La aptitud.

Para determinar si un anticuerpo es más apto que otro se hace una comparación entre los dos bajo el siguiente criterio:

Dados dos anticuerpos a_1 y a_2 que representan dos puntos $\boldsymbol{x}_1$ y $\boldsymbol{x}_2 \in S$ respectivamente, a_1 tiene mayor aptitud que a_2 sí:

1) $transg(\boldsymbol{x}_1) = 0$ y $transg(\boldsymbol{x}_2) > 0$

2) $(transg(\boldsymbol{x}_1) > 0$ y $transg(\boldsymbol{x}_2) > 0)$ y $transg(\boldsymbol{x}_1) \le transg(\boldsymbol{x}_2)$

3) $(transg(\boldsymbol{x}_1) = 0$ y $transg(\boldsymbol{x}_2) = 0)$ y $f(\boldsymbol{x}_1) \ge f(\boldsymbol{x}_2)$ si se está maximizando.

4) $(transg(\boldsymbol{x}_1) = 0$ y $transg(\boldsymbol{x}_2) = 0)$ y $f(\boldsymbol{x}_1) \le f(\boldsymbol{x}_2)$ si se está minimizando.

En el sistema inmune esta comparación se haría basándose en la afinidad del anticuerpo por los epitopos de los patógenos y la afinidad del anticuerpo por los epitopos propios. Sus equivalentes artificiales son el valor de la función objetivo evaluada en el punto que representa el anticuerpo, y el valor de la *transgresión* de ese punto, respectivamente.

4.1.4 La maduración.

El proceso de Maduración de los anticuerpos artificiales se lleva a cabo a través de un RMHC (Random Mutation Hill Climbing). Como se mencionó en un capítulo anterior, la maduración de la afinidad de los receptores de los linfocitos B es uno de los mecanismos que ayudan a la creación de receptores cada vez más afines a los epitopos de los patógenos.

La maduración de la afinidad se lleva a cabo mutando la mínima parte del genoma que se encuentra en la región variable de los receptores (mutación puntual) cada vez que se multiplica un linfocito B en la zona obscura de estructuras dentro de los nodos linfáticos y el bazo.

Todos aquellos anticuerpos que al mutar no se enlazan con la fuerza suficiente a un epitopo de patógenos son eliminados, y esto favorece la subsistencia de los anticuerpos que al ser mutados se enlazan con la suficiente fuerza a un epitopo de patógenos. Puede observarse que existe una gran similitud de la maduración de la afinidad con el RMHC.

Cada mutación genera un descendiente. Si este nuevo anticuerpo tiene una mayor aptitud bajo los criterios expuestos en el punto anterior, el progenitor será reemplazado por su descendiente.

4.1.5 Mutación.

La mutación se lleva a cabo sobre la mínima parte del genoma que se encuentra en la región variable de los receptores y es llamada mutación puntual. Dentro del sistema inmune artificial propuesto, los receptores o genes son codificados como vectores de reales.

El dominio de cada variable en el vector es segmentado en un número aleatorio de partes. Estas partes serán sometidas a mutación de tal forma que sea análoga a la mutación sufrida por la mínima parte del genoma que se encuentra en la región variable de los receptores en el sistema inmune biológico.

El operador de mutación utilizado está inspirado en el operador de mutación de Mühlenbein´s [17]. Pero, este operador genera de manera aleatoria, el tamaño de la mutación. En las primeras evaluaciones se usan mutaciones grandes, conforme aumentan las evaluaciones disminuye el tamaño de la mutación.

Después de consumir tres décimas partes de las evaluaciones es utilizado un parámetro variable para determinar el tamaño de la mutación que va desde mutaciones muy grandes hasta mutaciones muy pequeñas. De esta manera, se busca que las mutaciones grandes permitan que el algoritmo pueda salir de los óptimos locales en los cuales se haya podido quedar atorado; mientras que las mutaciones pequeñas le ayuden a encontrar un mejor óptimo. Las mutaciones de impacto medio ayudan a acercarlo o alejarlo de manera importante de los óptimos.

4.1.6 Cruzamiento.

El cruzamiento ayuda a liberar el algoritmo cuando este queda atrapado en óptimos locales, y a hacer grandes desplazamientos dentro del área de búsqueda.

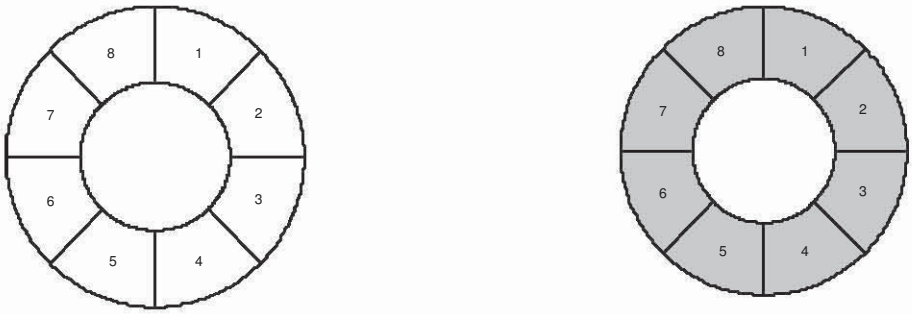
Al principio, cuando hay alta probabilidad de que el vector esté lejos del vector óptimo, la mutación puntual debe usarse hasta que se encuentre dicha cadena o hasta que las cadenas se queden atoradas en óptimos locales. Si éste último es el caso, se debe aplicar una operación de cruzamiento. Así que un ciclo de varias operaciones de mutación puntual y cruzamiento, resultan en una buena estrategia para la optimización de funciones. Dentro del algoritmo propuesto se utiliza el cruzamiento anular [36] (figura 4.1) con objeto de hacer prevalecer los patrones en el gen.

Durante la fase experimental se observó que cuando los descendientes del padre son muy parecidos entre sí, el puro cruzamiento no basta para sacarlo de los óptimos locales en donde el algoritmo queda atorado. Por tanto se implementó una rutina que hace que los extremos del anillo sean sometidos de manera aleatoria ya sea a una mutación múltiple (tres mutaciones puntuales sobre el mismo cromosoma), o a cambiar el valor del cromosoma por otro generado aleatoriamente dentro del rango de la variable. Esta rutina se toma como una analogía de lo que sucede en los extremos del anillo en genes codificados en binario. De esta manera, si se queda atorado en un óptimo local y los cromosomas cruzados son iguales o tienen pocos genes (vector con pocas dimensiones) estas variaciones en los

extremos del anillo permitirán al algoritmo salir de ahí.

Se hicieron algunas otras pruebas quitando el cruzamiento y dejando solamente implementada la rutina de mutación en los extremos del anillo. Se observa que los resultados se degradan un poco, pero que el cruzamiento no es indispensable para el funcionamiento del algoritmo.

Vector 1 Vector 2



Se forma un anillo con los vectores.

Se efectúa el cruzamiento.

Figura 4.1 Cruzamiento anular.

4.1.7 Algoritmo.

A grandes rasgos el algoritmo consiste en lo siguiente:

Se genera una población de anticuerpos de forma aleatoria, cada padre clonará dos descendientes que serán sometidos a maduración. Este proceso consiste en mutar los genes mientras siga dando mejores descendientes. Cuando esto no suceda, los descendientes serán ordenados de acuerdo a su aptitud y serán sometidos a cruzamiento formando las parejas por pareo de Vasconselos [1], es decir, el mejor con el peor el segundo mejor con el penúltimo peor, así sucesivamente.

Después del cruzamiento serán sometidos nuevamente a maduración. Al terminar de madurar se agruparan padres e hijos y serán ordenados de acuerdo a su aptitud. Los menos aptos serán eliminados dejando una población igual a la población inicial y se repite el proceso hasta agotar las evaluaciones y como resultado se obtiene el individuo más apto, su aptitud y su transgresión.

El algoritmo es:

inicio

1 Generar aleatoriamente, una población inicial A^0, de tamaño μ anticuerpos. Los anticuerpos son vectores de *L* números reales que representa un punto en el espacio de búsqueda *S*.

2 Calcular el valor de la función objetivo y de la transgresión del primer anticuerpo y almacenar este anticuerpo junto con los mencionados valores en el registro *MejorAnticuerpo*. Este registro será modificado únicamente por la rutina del Madurador. La función objetivo ha sido evaluada una vez, por lo que el contador del número de evaluaciones debe ser actualizado: $NEv = NEv + 1$.

3 Se invoca la rutina:

Madura*(A^0 , μ , Nev, rng)*

Cuando se regresa de esta rutina la variable *NEv* y la población en el argumento (en este caso A^0) están actualizadas. La rutina invocada,

Madura, monitorea el desempeño de la maduración y determina si continúa la búsqueda del óptimo o si regresa el control a esta rutina principal.

4 $i = 0$

5 **mientras**(*NEv* < *NMaxEv*)

6 Hacer la población intermedia, $A^{\perp}$, de tamaño λ copiando los individuos de la población de padres. El número de copias que cada anticuerpo pone en la población intermedia está dado por el valor del parámetro ω que para este trabajo vale 2. Cada individuo pone por lo menos una copia en $A^{\perp}$, por lo que $\lambda \geq \mu$.

7 Aplicar el pareo *Vasconcelos* y el cruzamiento anular con probabilidad $p_c = 1$. El pareo *Vasconcelos* [1] es una manera determinista de formar las parejas de anticuerpos que se cruzarán: la primera cadena se cruza con la última, la segunda con la penúltima, y así consecutivamente. Todos los cruces se hacen con una probabilidad p_c. La razón de hacerlo de esta manera es el tratar de preservar la variedad genética. En el cruzamiento anular, el genoma no se ve como una estructura lineal de números reales sino como un anillo. Un vector se transforma en un anillo conectando su extremo izquierdo con su extremo derecho. Cuando se aplica el cruzamiento anular hay dos parámetros que se deben tomar en cuenta: 1) el punto de inicio del cruce, que puede ser cualquiera de los L números reales del vector y 2) la longitud del arco que puede tomar los valores de 1 a L-1. El cruzamiento en un punto por su estructura lineal destruye los esquemas de mayor longitud definitoria y tiende a preservar esquemas de menor longitud. El cruzamiento uniforme por su naturaleza de cruzamiento gen a gen tiende a destruir los esquemas. El cruzamiento anular trata de no destruir esquemas y por su estructura circular no favorecer ni a los esquemas de longitud definitoria grande ni a los de longitud definitoria pequeña.

8 Se invoca la rutina: Madura*($A^{\perp}$, μ, Nev, rng)*

9 Ordenar los individuos de la población $A^i + A^{\perp}$ de la siguiente manera. Los anticuerpos son ordenados de acuerdo a:

i) si son factibles van primero que los no factibles,

ii) si son factibles se ordenan de acuerdo a sus valores de la función objetivo;

iii) si se minimiza, el anticuerpo con el menor valor de la función objetivo va primero;

iv) si se maximiza, el anticuerpo con el mayor valor de la función objetivo va primero;

v) si el anticuerpo represente un punto no factible, entonces el individuo con el menor valor de transgresión es el que va primero. Es decir, se siguen los criterios definidos en 4.1.4.

10 La población A^{i+1} se hace con los primeros μ anticuerpos de la población ordenada $A^i + A^{\perp}$.

11 $i = 1 + 1;$

12 **fin de mientras**

13 Regresa al mejor anticuerpo encontrado durante toda la corrida y almacenado en el registro *MejorAnticuerpo.* Donde mejor anticuerpo es aquel que tiene la menor transgresión o una transgresión cero y mejor aptitud.

fin.

La rutina de Maduración es:

Madura*($A^{\perp}$, μ, Nev, rng)*

inicio

1 Calcula el número de evaluaciones de la función objetivo asignado a cada anticuerpo en *A* (que es igual al número de mutaciones puntuales que se aplicará a cada anticuerpo en *A*), N_{ea}, como una función de la longitud de la subcadena del anticuerpo que representa al punto en *S*: $N_{ea} = 2 \cdot rndx$. Donde el parámetro rndx es un número aleatorio entre 1 y x. Para este trabajo se utiliza una x de 25.

2 Calcular el número de evaluaciones que quedan por realizarse:

$NEv_{restantes} = NMaxEv - NEv$;

3 Mientras ($NEv < NMaxEv$ y $NSaltosBuenos/(N \cdot N_{ea}) \geq T_{\text{saltos buenos}}$);

$T_{\text{saltos buenos}} \in [0,1]$ es un umbral dado por el usuario, después de varias pruebas experimentales se encontró que los mejores resultados se obtenían generando de manera aleatoria un entero entre [1,25] y dividiéndolo entre 10000.

4 Revisar si el número de evaluaciones restantes es suficiente para evaluar N_{ea} veces a cada uno de los N miembros de la población; si no es suficiente entonces: $N_{ea} = \left\lceil \frac{NEv_{restantes}}{N} \right\rceil$

si($N_{ea} = 0$) **entonces** $NEv = NMaxEv$;

Con esto se evita que se sobrepase el número máximo de evaluaciones fijado por el usuario. Si es que el número de evaluaciones restantes no alcanza para aplicar N_{ea} veces la mutación puntual a cada anticuerpo se hace *NEv=NmaxEv*.

5 *SaltosBuenos* = 0; inicia el contador de buenos saltos. Este contador es actualizado si el nuevo anticuerpo supera al mejor anticuerpo.

6 **para** *i* = 1 **hasta** *N*

7 **para** j = 1 **hasta** N_{ea}

8 Hacer una copia b del vector, z_i, y escoger al azar uno de los n números reales del vector que representa un punto en *S* y substituir el valor por el obtenido en la rutina *muta(r,z_i,rg)* donde r es la posición de la variable a mutar, z_i es el vector y rg los límites superior e inferior para esa variable. Llamar a la copia con la mutación puntual *b*;

9 Evaluar la función objetivo y las restricciones en *b*.

10 Se compara el nuevo anticuerpo b con MejorAnticuerpo usando la función C*omparaAnticuerpo(fo,tg,foma,tgma)* pasando la evaluación de la función objetivo y la transgresión de ambos anticuerpos como parámetros.

11 **Sí** *(b es MejorQueOIgualA MejorAnticuerpo)*

12 *SaltosBuenos* = *SaltosBuenos* + 1; aquí se actualiza el contador de buenos saltos

13 Substituir z_i con *b*, almacenar *b* en el registro temporal *MejorAnticuerpoTemp* y si (b es *MejorQueOIgualA MejorAnticuerpo*) actualizar el registro *MejorAnticuerpo* almacenando en él, el punto *b*, *f*(*b*) y *transg*(*b*).

14 **Si no** z_i permanece con el valor que tenía.

15 **fin de sí**

16 Se genera de manera aleatoria $T_{\text{saltos buenos}} \in [0,1]$ dando aleatoriamente un entero entre 1 y 25 y dividiéndolo entre 10,000 según se explica en 3.

17 **Si** $NSaltosBuenos/(N \cdot N_{ea}) <$ índice de rango y $NEv > (3/10)NMaxEv$ **entonces** se incrementa la razón de cambio del rango (*rr*). La razón de cambio del rango, es un parámetro que será usado por el algoritmo de mutación para generar un número aleatorio entre cero y *rr*. El valor máximo de rr es 56 y el valor mínimo es 8. Los incrementos de *rr* son de 4 y si valor máximo es rebasado, se iguala con el valor mínimo (*rr*=8) y se incrementa el índice de rango. El índice de rango (*ir*), es usado para incrementar *rr* cuando el porcentaje de saltos buenos es inferior a dicho índice. El valor máximo de *ir* es 0.075 y el valor mínimo es 0.025. Los incrementos de *ir* son de 0.025 y si el valor máximo es rebasado, *ir* se iguala al valor mínimo (*ir*=0.025).

18 **fin de para**

19 **fin de para**

20 $NEv = NEv + N \cdot N_{ea}$

21 **fin de mientras**

fin.

donde la función C*omparaAnticuerpo* es:

C*omparaAnticuerpo(fo,tg,foma,tgma)*
inicio
1 **sí**(el anticuerpo *b* está en la región factible y *a* no
o
tanto el anticuerpo *a* como el *b* no están en región factible y *transg*(*b*)≤*transg*(*a*)
o
ambos anticuerpos *a* y *b* están en región factible y $f(b) \geq f(a)$)
2 **regresa** verdadero;
3 **si no regresa** falso;
fin.

Muta (r,z_i,rng)

inicio

1 De rng se obtiene el rango de la variable r-ésima del vector Z_i.

2 Se selecciona el número máximo para generar la fracción del rango que será adicionada o decrementada al número real de acuerdo al siguiente criterio:

3 Sí NEv <(3/10) $NMaxEv$ r_{max}=8

4 Sí NEv <(2/10) $NMaxEv$ r_{max}=4

5 Sí NEv <(1/10) $NMaxEv$ r_{max}=2

6 Variación= r_{max} + un entero aleatorio entre [1, r_{max}].

7 Si $NEv \geq$ (3/10) $NMaxEv$ r_{max} es igual al generado en Madura punto 17

Variación = un entero aleatorio entre [1, r_{max}].

8 Se obtiene la potencia base dos de –Variación (2^-Variación).

9 Se obtiene aleatoriamente un signo positivo o negativo y se aplica al rango de la variable que a su vez se multiplica por el resultado de la potencia.

10 El resultado se suma algebraicamente con la r-ésima variable del vector Z_i.

11 Si el resultado sobrepasa los límites inferior o superior para dicha variable, se asigna a la variable el valor del límite rebasado.

fin

En pseudo código el algoritmo sería el siguiente:

El proceso principal

1 *ContEvals* ←0
2 *i* ← *0*
3 obtener una población aleatoria, $P_{(0)}$, de w *vectores* de L variables
4 MejorAnticuerpoGlobal ← primera cadena de la población
5 Se aplica el **proceso de maduración** y se actualiza *ContEvals*
6 **mientras** (*ContEvals* < *MaxEva*){
7 Se genera una población intermedia, $P_{(1/2)}$, copiando dos veces cada vector de $P_{(0)}$
8 Se **ordena** ascendentemente la población de acuerdo a la aptitud
9 Se aplica un pareo de vasconselos y aplica el **proceso de cruzamiento** anular
10 Se aplica el **proceso de maduración** y se actualiza *ContEvals*
11 Se unen $P_{(i)}$ U $P_{(1/2)}$ y se ordenan de acuerdo a la aptitud y transgresión.
12 Se toman los primeros w elementos y se hace una población $P_{(i+1)}$
13 **Si** *ContEvals* >= *MaxEva* **termina mientras**
14 *i* ← *i* + 1;
15 }**fin de mientras**
16 Se despliega el MejorAnticuerpoGlobal

El proceso de maduración

1 **para** i ←1,…,*w,* salva el i-*esimo* vector en *el registro* de MejorAnticuerpo,
2 g=w*2*(1+random(25))

```
3 Si numvar>15 → g=w
4 si (MaxEva-ContEvals)<=g entonces g=(MaxEva-ContEvals)/w
                              y  si (g==0) entonces  ContEvals =MaxEva y si (g>0)
entonces g=g*w
5 mientras  afinidad >= LimiteEficiencia  y ContEvals <MaxEva {
6 afinidad=0
7 para i ←1 a w {
8 calcula la aptitud del i-esimo vector;
9 repite g/w veces{
10 selecciona aleatoriamente una variable en el vector y aplica el proceso de
mutación
11 calcular la nueva aptitud.
12 ContEvals ← ContEvals + 1
13 si  la aptitud del  vector mutado es igual o mayor a la del i-esimo vector
                              entonces reemplaza el i-esimo vector
14 si  la aptitud del  vector mutado es igual o mayor a la del MejorAnticuerpo
                              entonces reemplaza el MejorAnticuerpo
15 afinidad = afinidad +1
16}fin de repite
17}fin de para
18 LimiteEficiencia =1+random(25)
19 LimiteEficiencia = LimiteEficiencia / LimiteInferiorEficiencia
20 si afinidad/g)<ir y ContEvals >=(3*MaxEva/10)
                        entonces      rr=rr+4
                                      ir=ir+0.025
                                      si (rr>56) entonces rr=8
                                      si (ir>0.075) entonces ir=0.025
21 }fin de mientras
22 si  la aptidud del  vector mutado es igual o mayor a la del MejorAnticuerpoGlobal
                              entonces reemplaza el MejorAnticuerpoGlobal
23 regresa;
```

El proceso de mutación

```
1 s=aleatorio {0,1};
2 si (s==0) entonces s=-1 si no s=1;
3 ri=rng[func-1][r].max-rng[func-1][r].min;
4 si (CEM<(3*maxeva/10)) entonces rr=8;
5 si (CEM<(2*maxeva/10)) entonces rr=4;
6 si (CEM<(1*maxeva/10)) entonces rr=2;
7 si (CEM>=(3*maxeva/10)) entonces r32=random(rr+1) si no
r32=rr+random(rr+1);
8 sg=2^-r32;
9 y=(s*sg*ri);
10 z=y+z;
11 si ((z)>(rng[func-1][r].max)) z=rng[func-1][r].max;
12 si ((z)<(rng[func-1][r].min)) z=rng[func-1][r].min;
13 regresa (z);
```

El proceso de cruzamiento

1 se seleccionan aleatoriamente dos puntos en el anillo
2 se intercambian los arcos
3 rnc=aleatorio {0,1};
4 **si** rnc=0 las variables en los extremos del arco se cambian por un valor generado aleatoriamente dentro de los limites inferior y superior de dichas variable.
5 **si** rnc=1 se llama tres veces al **proceso de mutación** sobre las variables en los extremos del arco.
6 se actualiza la transgresión y la aptitud de los nuevos vectores.

4.1.8 Analogía entre el Sistema Inmune y el algoritmo propuesto.

La misión del sistema inmune es defender al organismo de los patógenos (exógenos y endógenos) y para poder combatirlos debe tener la capacidad de distinguir lo que es propio de lo que no lo es.

Esto lo logra eliminando a aquellos linfocitos cuyos receptores se enlazan con partes del propio organismo (selección negativa de los linfocitos B en la médula ósea y selección negativa de los linfocitos T en el timo) o eliminando a aquellos linfocitos cuyos receptores no son afines a los epitopos pertenecientes a patógenos (selección negativa de los linfocitos B en los nodos linfáticos o en el bazo), o bien, permitiendo que se "editen" los receptores de los linfocitos que se han enlazado con algún epitopo propio para comprobar si esta modificación impide que se enlacen otra vez con algún epitopo propio (edición de receptores de los linfocitos B en la médula ósea).

De esta manera, el sistema inmune busca generar anticuerpos cuyos receptores sean afines a los epitopos de los antígenos, mientras que en un problema de programación no lineal sujeto a restricciones busca un óptimo dentro de la región factible.

Continuando con la analogía, los epitopos de los antígenos son análogos a la región factible y los genes en el receptor serían el vector que da solución al problema de programación no lineal.

Las analogías pueden resumirse en la Tabla 4.1.

	Mecanismos del Sistema Inmune	**Algoritmo Evolutivo Propuesto**
1	Epitopos de patógenos	Región factible, *F*
2	Epitopos propios	Región no factible $S - F$
3	Los anticuerpos evolucionan dentro del cuerpo de un organismo por maduración de la afinidad.	Los anticuerpos evolucionan por escalamiento por mutación aleatoria y por cruzamiento entre sí, por esta última causa se requiere usar poblaciones de al menos dos anticuerpos.
4	Grado de afinidad entre anticuerpos y epitopos de patógenos.	Valores de la función en regiones factibles.
5	Grado de afinidad entre anticuerpos y epitopos propios	Valores de la transgresión, *transg* (ver ec. 4.3)
6	Entre más afines son los receptores de un linfocito al epitopo de un patógeno más probabilidades tiene de subsistir. Entre más afines son los receptores de un linfocito a epitopos propios más probabilidades tiene de ser eliminado.	Entre mejor sea el valor de la función objetivo evaluada en un punto representado por un anticuerpo, más probabilidades tiene de formar parte de la siguiente generación de padres. Entre mayor sea el valor de la transgresión de un auto anticuerpo, mayor probabilidad tiene de no aparecer en la próxima generación de padres.

Tabla 4.1. Analogías entre el SI y el algoritmo propuesto.

Capítulo 5

Experimentación.

En este capítulo se describirán algunos detalles sobre el algoritmo MARACoR, así como los resultados obtenidos durante la fase experimental.

5.1 Lenguaje de programación y equipo usado.

El programa fue desarrollado en lenguaje C con el compilador Borland C++ Power Builder 6.0. Las pruebas se corrieron en Computadoras Personales con Windows XP. Se utilizaron 3 equipos diferentes; una Lap Top con un procesador AMD Athlon XP 2000+ a 910 MHz y con 192 MB de RAM, una PC de escritorio con un procesador Pentium IV a 1.7 GHz y 256 MB de RAM y por último, una PC de escritorio con un procesador Pentium III a 733 MHz y 512 MB de RAM.

5.2 Codificación de anticuerpos.

Para representar a los anticuerpos se utilizaron arreglos de tipo double. El tipo double para este compilador utiliza 64 bits y tiene una precisión de 15 dígitos. Adicionalmente a las variables que componen el vector se pusieron en el arreglo un

campo para la aptitud, un campo para la transgresión. Aunque estos datos no forman parte del genoma son útiles para poder ordenar los anticuerpos de acuerdo a su aptitud, por esta razón son incluidos en el arreglo. Cada anticuerpo representa un punto en el espacio de búsqueda S. En la Figura 5.1 se esquematiza el anticuerpo.

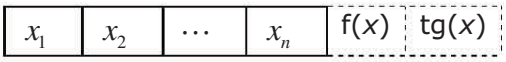


Figura 5.1. En un arreglo unidimensional del tipo double se almacenan las variables que representan las coordenadas de un punto en el espacio de búsqueda S.

5.3 Prueba del algoritmo.

Para ver que tan bien funciona el algoritmo, éste es sometido a una colección de funciones de prueba, que pueden ser aplicaciones reales o un conjunto de funciones seleccionadas para tal propósito.

En 1975 Dixon y Szegö desarrollaron una pequeña colección de funciones compuesta por problemas de baja dimensionalidad, sin restricciones y muy fáciles de resolver.

En 1981 Hock y Schittkowski publicaron una colección de problemas de optimización local.

En 1978 Dixon y Szegö presentaron nuevas funciones para probar sus algoritmos de optimización global no restringida.

En 1990 Floudas y Pardalos publicaron el libro *A Collection of Test Problems for Constrained Global Optimization* [18].

En 1995 Michalewicz propone 5 funciones en *Genetic Algorithms, Numerical Optimization, and Constraints* [41]. Después, en 1999, amplia esta colección de funciones a 11 en *Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization* [39], y en el 2000, Runarsson y Yao en *Stochastic Ranking for Constrained Evolutionary Optimization* [40] la amplían a 13.

Para probar el algoritmo MARACoR se utilizó la colección de 13 funciones usada por Runarsson y Yao. (Ver apéndice A).

En la tabla 5.1 tomada de [36] se muestran algunas características de estas trece funciones. La última columna es el tamaño relativo de la región factible.

Problema	Tipo de función objetivo	No. de variables	No. total de restricciones	No. de restricciones lineales	No. de restricciones de igualdad	$\frac{F \cap S}{S}$
G01	no lineal	13	9	9	0	0.000002
G02	no lineal	20	2	1	0	0.999973
G03	no lineal	10	1	0	1	0.000002
G04	no lineal	5	6	0	0	0.269744
G05	no lineal	4	5	2	3	< 0.000001
G06	no lineal	2	2	0	0	0.000063
G07	no lineal	10	8	3	0	0.000002
G08	no lineal	2	2	0	0	0.008677
G09	no lineal	7	4	0	0	0.005261
G10	lineal	8	6	4	0	0.000003
G11	no lineal	2	1	0	1	0.000102
G12	no lineal	3	1	0	1	0.047771
G13	no lineal	5	3	0	3	< 0.000001

Tabla 5.1 Algunas características de los problemas de prueba para los algoritmos propuestos.

Para efectuar las pruebas se utilizó el mismo tipo de prueba que se propone en [39], [40] y [41], consistente en asignar un valor a cada uno de los parámetros que deben fijarse del algoritmo. A continuación son realizadas 30 corridas independientes con los valores fijados. Este procedimiento debe repetirse para cada función. Una vez hechas las corridas se calcula la media, la mediana, la desviación estándar y se separan el mejor y el peor valor obtenido.

5.4 Los parámetros.

Para la asignación de parámetros, se podría necesitar otro programa de optimización que nos indicara cuales son los parámetros más adecuados. Para la elaboración del programa final de MARACoR se desarrollaron cerca de 50 versiones previas durante las cuales se fueron probando y modificando los parámetros y en algunos casos más extremos, el mismo algoritmo. Para evitar la dependencia de un valor en particular, ya que algunos valores favorecían a algunas funciones y perjudicaban a otras, al final lo que se decidió fue que los parámetros se generaran de manera aleatoria dentro de un rango predefinido. Por tanto, el usuario no tiene que asignar ningún parámetro excepto el número de evaluaciones y la tolerancia.

Los parámetros pueden ser de dos tipos: unos a los que se les puede llamar *constantes*. A éstos se les asigna un solo valor que no cambiará en ninguna de las corridas. Y otros a los que se les puede llamar *variables*, cuyos valores puede cambiar de un experimento a otro y son tomados de un intervalo designado para tal efecto para cada uno de ellos.

Los parámetros que se deben fijar son:

μ Tamaño de la población de padres que se fijo en 1.

ω Número de descendientes se prefija en 2.

f_{nea} El número de mutaciones puntuales que sufrirá cada uno de los anticuerpos antes de probar si debe aplicarse el operador de cruzamiento se genera de manera aleatoria, un número entre 1 y f_{nea} *y el número obtenido es multiplicado por 2*.Para funciones que tienen un número de variables superior a 10, este parámetro se fija en 1 (En el conjunto de pruebas g1 y g2).

$T_{saltos\ buenos}$ Sirve de umbral, si la fracción de saltos a mejores puntos es menor que él se aplica el operador de cruzamiento, si no, se realizan otras f_{nea} mutaciones puntuales a cada uno de los individuos de la población; con un entero entre [1,25] dividido entre 10000.

n_{evals} número de evaluaciones que ejecutará el algoritmo; se usaron 350,000.

p_m Probabilidad de mutación queda fija en 1.

pc Probabilidad de cruzamiento queda fija en 1.

ir El índice de rango se fija en 0.075 máximo y 0.025 mínimo con incrementos de 0.025.

rr La razón de cambio del rango que es aplicado en la mutación se fija en un máximo de 56 y un mínimo de 8 con incrementos de 4.

tol *Se utiliza una tolerancia de 10^-4 para todas aquellas restricciones de igualdad que se convirtieron en desigualdades.(Ver 4.1.2 la transgresión).*

La mayoría de los parámetros se obtuvieron a través de la experimentación.

En la tabla 5.2 se resumen los valores fijados a los parámetros.

Variable	Intervalo asignado
p_m	1
p_c	1
μ	1
ω	2
n_{evals}	350,000
f_{nea}	[1,25]
$T_{saltos\ buenos}$	[1,25]/ 10000
ir	[0.025,0.075]
rr	[8,56]

Tabla 5.2 Valores asignados a los parámetros.

5.5 Los experimentos.

Se realizaron cuatro experimentos. Para cada uno de ellos se utilizaron los mismos parámetros (ver tabla 5.2). El número de evaluaciones se estableció en 350,000 como lo hacen en [39], [40] y [41] (en [36] se usaron 500,000 evaluaciones). Una vez que se han fijado los parámetros constantes y variables, se realizan 30 corridas independientes para cada uno de los trece problemas por experimento. Para cada una de estas corridas se reporta:

1) El mejor valor de la función objetivo obtenido, $f(\boldsymbol{x})$,
2) El número de evaluaciones de la función objetivo y sus restricciones, necesarias para obtenerlo,
3) Indicación de sí el punto correspondiente, $\boldsymbol{x}$, está o no en región factible (si la transgresión es cero, esta en región factible),
4) Número de veces que se invocó el operador de cruzamiento para obtener el mejor individuo.
5) Las coordenadas de cada punto, $\boldsymbol{x} = (x_1, x_2, ..., x_n)$.

En cada corrida el algoritmo se encarga de generar una población inicial de μ anticuerpos que para nuestro caso es solo de un anticuerpo. A partir de este anticuerpo inicial, se generan dos anticuerpos descendientes; los cuales serán sometidos a los algoritmos de maduración y cruzamiento. De entre los padres y los descendientes se seleccionan los μ mejores anticuerpos y se repite el proceso.

Una vez terminados los cuatro experimentos, para cada uno de ellos se toman en cuenta solamente aquéllas corridas cuyo mejor punto encontrado cayó en *región factible*, para obtener la siguiente información:

1) mejor valor de la función objetivo, $f(\boldsymbol{x}_{mejor})$,**"mejor"**,

2) la mediana de los valores de la función objetivo, $f(\boldsymbol{x}_{mediana})$,"**mediana"**,

3) la media (promedio aritmético) de los valores de la función objetivo de cada punto, $f(\boldsymbol{x}_{media})$,**"media"**,

4) la desviación estándar de los valores de la función objetivo,

5) el peor valor de la función objetivo, $f(\boldsymbol{x}_{peor})$,**"peor"**,

6) la media del número de iteraciones para encontrar al mejor punto de cada corrida, **"num evals promedio"**

7) El número de veces que se usó el operador de cruzamiento,**"num cruza promedio"**,
8) El porcentaje de puntos factibles. **"% de factibles"**

Esta información se encuentra concentrada en la sección 5.6. Los datos arriba mencionados de los cuatro experimentos son promediados y con ellos se genera la tabla 5.7.

Se busca cual fue el experimento que obtuvo la menor desviación estándar para cada uno de los problemas de prueba y con esta información se conforma la tabla 5.8.

Por último de entre las 120 corridas para cada problema de prueba, se busca el valor de la función objetivo más cercano al óptimo encontrado por el algoritmo. El valor de la función objetivo, así como el valor para cada una de las variables del mejor vector encontrado por el algoritmo están concentrados en la tabla 5.9

5.6 Resultados.

A continuación se presentan los concentrados de los resultados obtenidos durante los experimentos.

Problema	óptimo	mejor	mediana	media	desv est	peor	num evals promedio	num cruza promedio	% de factibles
g01 (MIN)	-15	-15	-15	-15	0	-15	273609.4333	130012.9	100
g02 (MAX)	0.803619	0.803535272	0.797717937	0.797225429	0.005752325	0.785633661	349487.6	128994.5667	100
g03 (MAX)	1	0.998948701	0.977809203	0.970116087	0.025543152	0.894838564	348878.7	1200.533333	100
g04 (MIN)	-30665.539	-30665.5163	-30665.03597	-30664.52806	1.323280333	-30661.22107	339255.0667	1109.833333	100
g05 (MIN)	5126.4981	5126.632596	5262.269265	5453.196325	333.0819683	6067.816656	336738.7143	919	93.3333
g06 (MIN)	-6961.81388	-6961.813874	-6961.813868	-6961.813836	0.000147752	-6961.813062	309956.4667	1012.166667	100
g07 (MIN)	24.3062091	24.40701361	24.97795646	25.10792202	0.429724988	26.04364365	328600.1	595.3333333	100
g08 (MAX)	0.095825	0.095825041	0.095825041	0.095825041	4.28905E-17	0.095825041	157692.6	1207.433333	100
g09 (MIN)	680.6300573	680.6912633	681.1714915	681.3142071	0.488999773	682.7784174	332729.4667	674.5	100
g10 (MIN)	7049.3307	7049.47025	7061.91724	7083.638259	38.74123477	7196.369655	341897.1	713.4666667	100
g11 (MIN)	0.75	0.7499	0.7499	0.749900002	7.6805E-09	0.749900037	335848.0667	1885.766667	100
g12 (MAX)	1	1	1	1	0	1	65377.93333	606.4	100
g13 (MIN)	0.0539498	0.053944492	0.054656231	0.060312026	0.012908468	0.103193269	335729.4	970.0666667	100

Tabla 5.3. Experimento 1.

Problema	óptimo	mejor	mediana	media	desv est	peor	num evals promedio	num cruza promedio	% de factibles
g01 (MIN)	-15	-15	-15	-15	0	-15	285041.5333	135430.2667	100
g02 (MAX)	0.803619	0.803502175	0.795049001	0.796029727	0.007045501	0.778191044	348825.5	128794.8333	100
g03 (MAX)	1	0.994235299	0.975080171	0.967710732	0.021293606	0.917109395	347356.2333	1204.066667	100
g04 (MIN)	-30665.539	-30665.52478	-30664.95347	-30664.39584	1.310206095	-30660.37602	334429.0667	1108.233333	100
g05 (MIN)	5126.4981	5126.612993	5385.300231	5455.594912	305.8553134	6111.966879	332363.7857	919.8928571	93.3333
g06 (MIN)	-6961.81388	-6961.813875	-6961.813867	-6961.813858	2.41584E-05	-6961.81378	301059.6333	995.2333333	100
g07 (MIN)	24.3062091	24.39194768	24.85465845	25.01587752	0.497973213	26.13656634	331671.1333	619.7	100
g08 (MAX)	0.095825	0.095825041	0.095825041	0.095825041	4.34291E-17	0.095825041	174184.8	1240.333333	100
g09 (MIN)	680.6300573	680.8003906	681.2214529	681.393482	0.466290669	682.9359943	326374.5667	676.4	100
g10 (MIN)	7049.3307	7049.655058	7077.081782	7096.959934	49.56672932	7230.162322	336183.4667	705.8333333	100
g11 (MIN)	0.75	0.7499	0.7499	0.7499	1.21871E-09	0.749900007	329003.3333	1873.3	100
g12 (MAX)	1	1	1	1	0	1	57680.53333	550.1	100
g13 (MIN)	0.0539498	0.053947746	0.054742811	0.061815491	0.024308029	0.182423312	342507	984.7	100

Tabla 5.4. Experimento 2.

Problema	óptimo	mejor	mediana	media	desv est	peor	num evals promedio	num cruza promedio	% de factibles
g01 (MIN)	-15	-15	-15	-15	0	-15	281591.7	133795.6667	100
g02 (MAX)	0.803619	0.803547096	0.794043322	0.795292264	0.00696494	0.777240313	349288.3333	128911.7333	100
g03 (MAX)	1	0.998244999	0.976138296	0.965040987	0.028655426	0.898135479	348318.3	1203.566667	100
g04 (MIN)	-30665.539	-30665.52757	-30665.01318	-30664.43072	2.024405021	-30654.51342	340458.6667	1109.566667	100
g05 (MIN)	5126.4981	5130.210413	5197.662855	5356.302414	274.9260916	6012.657832	328252.2222	926.8888889	90
g06 (MIN)	-6961.81388	-6961.813875	-6961.813867	-6961.813858	2.85754E-05	-6961.813736	313821.6	1000.233333	100
g07 (MIN)	24.3062091	24.34539501	24.80341671	25.05215327	0.811500835	28.27964424	331697.3333	602.8	100
g08 (MAX)	0.095825	0.095825041	0.095825041	0.095825041	4.50061E-17	0.095825041	160995.4667	1199.066667	100
g09 (MIN)	680.6300573	680.6634085	681.2100559	681.2446427	0.313719778	682.0247818	332596.0667	698.6666667	100
g10 (MIN)	7049.3307	7052.387318	7084.890782	7099.98379	53.59501676	7260.358775	336979.8	694.2666667	100
g11 (MIN)	0.75	0.7499	0.7499	0.749900001	7.17921E-09	0.749900039	325272.3333	1849.4	100
g12 (MAX)	1	1	1	1	0	1	43359.13333	418.7666667	100
g13 (MIN)	0.0539498	0.053943575	0.055210855	0.057731319	0.006408347	0.082727945	338207.1667	983.3333333	100

Tabla 5.5. Experimento 3.

Problema	óptimo	mejor	mediana	media	desv est	peor	num evals promedio	num cruza promedio	% de factibles
g01 (MIN)	-15	-15	-15	-15	0	-15	276831.4333	131557.3	100
g02 (MAX)	0.803619	0.803469969	0.794825191	0.796316633	0.00505861	0.782490863	348469.6	128707.5333	100
g03 (MAX)	1	0.996116351	0.980329421	0.968124124	0.044695672	0.749965375	348619.1333	1207.766667	100
g04 (MIN)	-30665.539	-30665.53311	-30664.84703	-30663.95714	1.779028196	-30659.30591	339057.1333	1099.3	100
g05 (MIN)	5126.4981	5126.545496	5367.537479	5527.34258	389.5917878	6110.683263	330117.2963	922.962963	90
g06 (MIN)	-6961.81388	-6961.813875	-6961.81387	-6961.813851	4.15543E-05	-6961.813708	273162.8	991.8333333	100
g07 (MIN)	24.3062091	24.37613074	24.97114985	25.00758768	0.482437991	26.44231081	336089.9333	600.0333333	100
g08 (MAX)	0.095825	0.095825041	0.095825041	0.095825041	2.82301E-17	0.095825041	193013.2667	1300.033333	100
g09 (MIN)	680.6300573	680.9004788	681.3163407	681.412056	0.531843894	683.2766787	324165.1	673.4333333	100
g10 (MIN)	7049.3307	7049.788902	7059.339131	7075.536328	36.1420774	7201.62348	338756.9667	711.1333333	100
g11 (MIN)	0.75	0.7499	0.7499	0.7499	2.51306E-10	0.749900001	330403.5333	1881.966667	100
g12 (MAX)	1	1	1	1	0	1	62208.6	569.6333333	100
g13 (MIN)	0.0539498	0.05394346	0.054125913	0.061406185	0.022012397	0.161665376	344704.5	990.4333333	100

Tabla 5.6. Experimento 4.

La siguiente tabla muestra los resultados promedio de los cuatro experimentos.

Problema	óptimo	mejor	mediana	media	desv est	peor	num evals promedio	num cruza promedio	% de factibles
g01 (MIN)	-15	-15	-15	-15	0	-15	279268.525	132699.0333	100
g02 (MAX)	0.803619	0.803513628	0.795408863	0.796216013	0.006205344	0.78088897	349017.7583	128852.1667	100
g03 (MAX)	1	0.996886337	0.977339273	0.967747982	0.030046964	0.865012203	348293.0917	1203.983333	100
g04 (MIN)	-30665.539	-30665.52544	-30664.96241	-30664.32794	1.609229911	-30658.8541	338299.9833	1106.733333	100
g05 (MIN)	5126.4981	5127.500374	5303.192457	5448.109058	325.8637903	6075.781157	331868.0046	922.1861772	91.667
g06 (MIN)	-6961.81388	-6961.813875	-6961.813868	-6961.813851	6.05099E-05	-6961.813572	299500.125	999.8666667	100
g07 (MIN)	24.3062091	24.38012176	24.90179537	25.04588512	0.555409257	26.72554126	332014.625	604.4666667	100
g08 (MAX)	0.095825	0.095825041	0.095825041	0.095825041	3.98889E-17	0.095825041	171471.5333	1236.716667	100
g09 (MIN)	680.6300573	680.7638853	681.2298353	681.3410969	0.450213529	682.753968	328966.3	680.75	100
g10 (MIN)	7049.3307	7050.325382	7070.807234	7089.029578	44.51126456	7222.128558	338454.3333	706.175	100
g11 (MIN)	0.75	0.7499	0.7499	0.749900001	4.08243E-09	0.749900021	330131.8167	1872.608333	100
g12 (MAX)	1	1	1	1	0	1	57156.55	536.225	100
g13 (MIN)	0.0539498	0.053944818	0.054683952	0.060316255	0.01640931	0.132502476	340287.0167	982.1333333	100

Tabla 5.7. Resultados promedio de los 4 experimentos.

Los mejores resultados de los 4 experimentos se presentan en la tabla 5.8.

Problema	óptimo	mejor	mediana	media	desv est	peor	num evals promedio	num cruza promedio	% de factibles
g01 (MIN)	-15	-15	-15	-15	0	-15	273609.4	130012.9	100
g02 (MAX)	0.803619	0.803469969	0.794825191	0.796316633	0.00505861	0.782490863	348469.6	128707.5	100
g03 (MAX)	1	0.994235299	0.975080171	0.967710732	0.021293606	0.917109395	347356.2	1204.067	100
g04 (MIN)	-30665.539	-30665.52478	-30664.95347	-30664.39584	1.310206095	-30660.37602	334429.1	1108.233	100
g05 (MIN)	5126.4981	5130.210413	5197.662855	5356.302414	274.9260916	6012.657832	328252.2	926.8889	90
g06 (MIN)	-6961.81388	-6961.813875	-6961.813867	-6961.813858	2.41584E-05	-6961.81378	301059.6	995.2333	100
g07 (MIN)	24.3062091	24.40701361	24.97795646	25.10792202	0.429724988	26.04364365	328600.1	595.3333	100
g08 (MAX)	0.095825	0.095825041	0.095825041	0.095825041	2.82301E-17	0.095825041	193013.3	1300.033	100
g09 (MIN)	680.6300573	680.6634085	681.2100559	681.2446427	0.313719778	682.0247818	332596.1	698.6667	100
g10 (MIN)	7049.3307	7049.788902	7059.339131	7075.536328	36.1420774	7201.62348	338757	711.1333	100
g11 (MIN)	0.75	0.7499	0.7499	0.7499	2.51306E-10	0.749900001	330403.5	1881.967	100
g12 (MAX)	1	1	1	1	0	1	65377.93	606.4	100
g13 (MIN)	0.0539498	0.053943575	0.055210855	0.057731319	0.006408347	0.082727945	338207.2	983.3333	100

Tabla 5.8. Tabla que presenta los mejores resultados de los cuatro experimentos.

En la tabla 5.9. se presentan los valores de las variables para los mejores vectores encontrados en los cuatro experimentos.

	Problema g01	Problema g02	Problema g03	Problema g04	Problema g05
$f(\boldsymbol{x})$	-15	0.803535272012731	0.998948700593448	-30665.5331084488000	5126.545496046220000
X_1	1	3.160128771071190	0.313982676167601	33.0000014305115	676.901681400321000
X_2	1	3.124810900908660	0.308808925121724	29.9952915539237	1029.320993204050000
X_3	1	3.091660560967430	0.320953074223690	45.0000000000000	0.121047594086119
X_4	1	3.057149161643980	0.318923829871422	36.7757230647505	-0.395203423396721
X_5	1	3.020954329729590	0.316743128204937	-30665.5331084488000	
X_6	1	2.993019579744230	0.311005843290539		
X_7	1	2.951913265679120	0.321213580183382		
X_8	1	2.918362042123170	0.319140652141224		
X_9	1	0.503493086064654	0.314336477039469		
X_{10}	3	0.497748358545749	0.317082728905861		
X_{11}	3	0.485048970340471			
X_{12}	3	0.477834079431880			
X_{13}	1	0.457393116127484			
X_{14}		0.467430756271233			
X_{15}		0.464389213228163			
X_{16}		0.451255479982566			
X_{17}		0.453467484302678			
X_{18}		0.435251376848919			
X_{19}		0.450028410950517			
X_{20}		0.446297048719333			

	Problema g06	Problema g07	Problema g08	Problema g09	Problema g10
$f(\boldsymbol{x})$	-6961.813875063080000	24.388761996371300	0.0958250414180358	680.663408450305000	7049.470250222910
x_1	14.095000000233400	2.158217105207140	1.2279713525230900	2.302861875701770	582.658839431188
x_2	0.842960789674447	2.401658084782050	4.2453733656562800	1.962681806265540	1334.984706215230
x_3		8.745010567631100		-0.497240010140087	5131.826704576490
x_4		5.018926535446390		4.345264870944380	182.296791983260
x_5		0.955929029614471		-0.630179072266314	294.726935064235
x_6		1.332827800632040		1.041683099075000	217.702872089016
x_7		1.292776409937370		1.564396963802300	287.569809822949
x_8		9.804130042785930			394.726933716528
x_9		8.216628863000840			
x_{10}		8.310361914854270			

	Problema g11	Problema g12	Problema g13
$f(\boldsymbol{x})$	0.749900000000000	1.00000000000000	0.0539434599611295
x_1	0.707036072768515	4.99999994414956	-1.7156417085326400
x_2	0.500000008195924	4.99999994168206	1.5939723244219400
x_3		4.99999999518437	-1.8300446264375800
x_4			-0.7624521802915560
x_5			0.7651987290473280

Tabla 5.9 Valores de las variables para los mejores vectores obtenidos en los cuatro experimentos.

Capítulo 6

Análisis de resultados y conclusiones.

En este capítulo se hace un análisis de los resultados obtenidos con el algoritmo propuesto y se compara con los resultados de otros investigadores, adicionalmente se dan las conclusiones.

6.1 Análisis de resultados.

De los resultados obtenidos se puede observar que para el problema g1, g12 los resultados son óptimos. Al algoritmo le resulta ligeramente difícil g5. Mientras tanto g2, g3, g7, g10 y g13 dan buenos resultados. El resto de las funciones g4, g6, g8, g9 y g11 tienen un comportamiento muy bueno y el algoritmo resulta bastante robusto en estos casos.

6.2 Evaluación del algoritmo.

Tal como lo hace Hernández en [36], para tener medidas cuantitativas de los resultados de los experimentos se pueden usar las siguientes fórmulas:

Las ecuaciones siguientes sirven para medir de manera cuantitativa el desempeño de los métodos.

$$\xi_{mejor} = \begin{cases} \dfrac{min(\left|f(\boldsymbol{x}^*)\right|, \left|f(\boldsymbol{x}_{mejor})\right|)}{max(\left|f(\boldsymbol{x}^*)\right|, \left|f(\boldsymbol{x}_{mejor})\right|)} & \text{si } f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}_{mejor}) \text{ y } max(\left|f(x^*)\right|, \left|f(x_{mejor})\right|) \neq 0 \\ 1 & \text{de otra forma} \end{cases} \tag{6.1}$$

$$\xi_{media} = \begin{cases} \dfrac{min(\left|f(\boldsymbol{x}^*)\right|, \left|f(\boldsymbol{x}_{media})\right|)}{max(\left|f(\boldsymbol{x}^*)\right|, \left|f(\boldsymbol{x}_{media})\right|)} & \text{si } f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}_{media}) \text{ y } \max(\left|f(x^*)\right|, \left|f(x_{media})\right|) \neq 0 \\ 1 & \text{de otra forma} \end{cases} \tag{6.2}$$

$$\xi_{peor} = \begin{cases} \dfrac{min(\left|f(\boldsymbol{x}^*)\right|, \left|f(\boldsymbol{x}_{peor})\right|)}{max(\left|f(\boldsymbol{x}^*)\right|, \left|f(\boldsymbol{x}_{peor})\right|)} & \text{si } f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}_{peor}) \text{ y } \max(\left|f(x^*)\right|, \left|f(x_{peor})\right|) \neq 0 \\ 1 & \text{de otra forma} \end{cases} \tag{6.3}$$

Donde:

$f(\boldsymbol{x}^*)$ Es el mejor valor conocido de la función;

$f(\boldsymbol{x}_{mejor})$ Es el mejor valor de la función $f(\boldsymbol{x})$ obtenida en las 30 corridas independientes de un experimento;

$f(\boldsymbol{x}_{media})$ Es el promedio aritmético de los 30 valores de $f(\boldsymbol{x})$ obtenidas en las corridas independientes de un experimento;

$f(\boldsymbol{x}_{peor})$ Es el peor valor de la función $f(\boldsymbol{x})$ obtenida en las 30 corridas independientes de un experimento;

Obsérvese que $0 \leq \xi_i \leq 1$ para $i \in \{mejor, media, peor\}$. Entre más similar sea a 1 mejor será el resultado y entre más se acerque a 0 será peor.

Los resultados de la evaluación del promedio de los cuatro experimentos se presentan en la tabla 6.1. La calificación obtenida bajo el criterio de 6.2 es de ***37.72030***. El resultado de las mismas 3 funciones evaluadas por Coello y Cruz [4] es **8.99539** y el resultado de las mismas 10 funciones evaluadas por ISPAES [22] es **29.63425**.

Problema	óptimo	mejor	media	peor	ξ mejor	ξ media	ξ peor	
g01 (MIN)	-15	-15	-15	-15	1	1	1	
g02 (MAX)	0.803619	0.803513628	0.796216013	0.78088897	0.999868878	0.990787939	0.971715415	
g03 (MAX)	1	0.996886337	0.967747982	0.865012203	0.996886337	0.967747982	0.865012203	
g04 (MIN)	-30665.539	-30665.52544	-30664.32794	-30658.8541	0.999999558	0.999960508	0.999782006	
g05 (MIN)	5126.4981	5127.500374	5448.109058	6075.781157	0.99980453	0.940968333	0.843759505	
g06 (MIN)	-6961.81388	-6961.813875	-6961.813851	-6961.813572	0.999999999	0.999999996	0.999999956	
g07 (MIN)	24.3062091	24.38012176	25.04588512	26.72554126	0.996968323	0.970467164	0.909474905	
g08 (MAX)	0.095825	0.095825041	0.095825041	0.095825041	0.999999568	0.999999568	0.999999568	
g09 (MIN)	680.6300573	680.7638853	681.3410969	682.753968	0.999803415	0.998956412	0.9968892	
g10 (MIN)	7049.3307	7050.325382	7089.029578	7222.128558	0.999858917	0.994399956	0.976073832	
g11 (MIN)	0.75	0.7499	0.749900001	0.749900021	0.999866667	0.999866668	0.999866694	
g12 (MAX)	1	1	1	1	1	1	1	
g13 (MIN)	0.0539498	0.053944818	0.060316255	0.132502476	0.999907655	0.894448765	0.407160695	Calif total
Calificaciones					12.99296385	12.75760329	11.96973398	37.72030

Tabla 6.1. Resultados del promedio de los 4 experimentos de MARACoR.

Las evaluaciones de los mejores resultados obtenidos en los cuatro experimentos (menor desviación estándar) se presentan en la tabla 6.2. La calificación obtenida bajo el criterio de 6.2 es de ***38.10771***. El resultado de las mismas 3 funciones evaluadas por Coello y Cruz [4] es **8.99680** y el resultado de las mismas 10 funciones evaluadas por ISPAES [22] es **29.71220**.

Problema	óptimo	mejor	media	peor	ξ mejor	ξ media	ξ peor	
g01 (MIN)	-15	-15	-15	-15	1	1	1	
g02 (MAX)	0.803619	0.803469969	0.796316633	0.782490863	0.99981455	0.990913148	0.973708764	
g03 (MAX)	1	0.994235299	0.967710732	0.917109395	0.994235299	0.967710732	0.917109395	
g04 (MIN)	-30665.539	-30665.52478	-30664.39584	-30660.37602	0.999999536	0.999962722	0.999831636	
g05 (MIN)	5126.4981	5130.210413	5356.302414	6012.657832	0.999276382	0.957096464	0.852617635	
g06 (MIN)	-6961.81388	-6961.813875	-6961.813858	-6961.81378	0.999999999	0.999999998	0.999999989	
g07 (MIN)	24.3062091	24.40701361	25.10792202	26.04364365	0.995869855	0.968069324	0.933287578	
g08 (MAX)	0.095825	0.095825041	0.095825041	0.095825041	0.999999568	0.999999568	0.999999568	
g09 (MIN)	680.6300573	680.6634085	681.2446427	682.0247818	0.999951002	0.999097849	0.997955024	
g10 (MIN)	7049.3307	7049.788902	7075.536328	7201.62348	0.999935005	0.996296305	0.978852993	
g11 (MIN)	0.75	0.7499	0.7499	0.749900001	0.999866667	0.999866667	0.999866668	
g12 (MAX)	1	1	1	1	1	1	1	
g13 (MIN)	0.0539498	0.053943575	0.057731319	0.082727945	0.999884612	0.93449796	0.65213514	Calif total
Calificaciones					12.98883247	12.81351074	12.30536439	38.10771

Tabla 6.2. Los mejores resultados de los 4 experimentos.

6.3 Otros Trabajos.

En años recientes se han desarrollado otros trabajos donde han sido sometidos al mismo conjunto de pruebas, algunos incluso son sistemas inmunes artificiales. De tal manera que se pueden comparar los resultados obtenidos con el algoritmo propuesto y los resultados obtenidos en otros trabajos.

6.3.1 Koziel y Michalewicz 1999.

En 1999 Koziel y Michalewicz publican *Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization* [39], que es un algoritmo evolutivo para optimización numérica que es sometido a un conjunto de trece pruebas. En la tabla 6.3 se ven los resultados obtenidos. La calificación obtenida bajo el criterio de 6.2 es de ***33.27785***.

Problema	óptimo	mejor	media	peor	ξ mejor	ξ media	ξ peor	
g01 (MIN)	-15	-14.7864	-14.7082	-14.6154	0.98576	0.980546667	0.97436	
g02 (MAX)	0.803619	0.79953	0.79671	0.79119	0.994911768	0.991402642	0.984533716	
g03 (MAX)	1	0.9997	0.9989	0.9978	0.9997	0.9989	0.9978	
g04 (MIN)	-30665.539	-30664.5	-30655.3	-30645.9	0.999966118	0.999666107	0.999359574	
g05 (MIN)	5126.4981				0	0	0	
g06 (MIN)	-6961.81388	-6952.1	-6342.6	-5473.9	0.998604691	0.911055669	0.78627497	
g07 (MIN)	24.3062091	24.620	24.826	25.069	0.987254634	0.97906264	0.969572344	
g08 (MAX)	0.095825	0.095825	0.0891568	0.0291438	1	0.930412732	0.304135664	
g09 (MIN)	680.6300573	680.91	681.16	683.18	0.99958887	0.999222	0.996267539	
g10 (MIN)	7049.3307	7147.9	8163.6	9659.3	0.986210034	0.863507607	0.729797263	
g11 (MIN)	0.75	0.75	0.75	0.75	1	1	1	
g12 (MAX)	1	0.999999857	0.999134613	0.991950498	0.999999857	0.999134613	0.991950498	
g13 (MIN)	0.0539498	0.054	0.064	0.557	0.99907037	0.842965625	0.09685781	Calif total
Calificaciones					11.95106634	11.4958763	9.830909378	33.27785

Tabla 6.3 Resultados obtenidos por Koziel y Michalewics en 1999.

6.3.2 Runarsson y Yao 2000.

En el 2000 Thomas P. Runarsson y Xin Yao publican *Stochastoic Ranking for Constrained Evolutionary Optimization* [40], que es una estrategia evolutiva que trabaja con penalizaciones y con un ordenamiento de acuerdo a la aptitud. Los resultados se presentan en la tabla 6.4. La calificación obtenida bajo el criterio de 6.2 es de ***37.53328***.

Problema	óptimo	mejor	media	peor	ξ mejor	ξ media	ξ peor	
g01 (MIN)	-15	-15.000	-15.000	-15.000	1	1	1	
g02 (MAX)	0.803619	0.803515	0.781975	0.726288	0.999870585	0.973066839	0.903771563	
g03 (MAX)	1	1.000	1.000	1	1	1	1	
g04 (MIN)	-30665.539	-30665.539	-30665.539	-30665.539	1	1	1	
g05 (MIN)	5126.4981	5126.497	5128.881	5142.472	0.999999785	0.999535396	0.996893731	
g06 (MIN)	-6961.81388	-6961.814	-6875.940	-6350.262	0.999999983	0.987665014	0.912156244	
g07 (MIN)	24.3062091	24.307	24.374	24.642	0.999967462	0.997218721	0.986373229	
g08 (MAX)	0.095825	0.095825	0.095825	0.095825	1	1	1	
g09 (MIN)	680.6300573	680.630	680.656	680.763	0.999999916	0.999961886	0.999804715	
g10 (MIN)	7049.3307	7054.316	7559.192	8835.655	0.999293298	0.93255082	0.797827745	
g11 (MIN)	0.75	0.750	0.750	0.75	1	1	1	
g12 (MAX)	1	1.000000	1.000000	1.000000	1	1	1	
g13 (MIN)	0.0539498	0.053957	0.067543	0.216915	0.99986656	0.798747465	0.248714012	Calif total
Calificaciones					12.99899759	12.68874614	11.84554124	37.53328

Tabla 6.4. Resultados obtenidos por Runnarson y Yao en 2000.

6.3.3 Coello y Cruz 2002.

En 2002 Coello y Cruz presentaron *A parallel implementation of the artificial immune system to handle constraints in genetic algorithms: preliminary results* [4]. Ese trabajo es un algoritmo genético para optimización numérica que utiliza más de un procesador para su ejecución. Este trabajo solo presenta resultados para g1, g4 y g9. Los resultados se presentan en la tabla 6.5. La calificación obtenida bajo el criterio de 6.2 es de ***8.84334***.

Problema	óptimo	mejor	media	peor	ξ mejor	ξ media	ξ peor	calificacion
g01(1p)	-15	-14.7841	-14.5266	-13.8417	0.985606667	0.96844	0.92278	
g01(2p)	-15	-14.9995	-14.7773	-12.9892	0.999966667	0.985153333	0.865946667	
g01(3p)	-15	-14.9995	-14.5922	-11.8787	0.999966667	0.972813333	0.791913333	
g01(4p)	-15	-14.9978	-14.8201	-12.9930	0.999853333	0.988006667	0.8662	2.85406
g04(1p)	-30665.539	-30665.51	-30654.98	-30517.44	0.999999054	0.999655672	0.995170507	2.99483
g04(2p)	-30665.539	-30665.00	-30662.32	-30652.21	0.999982423	0.999895029	0.999565343	
g04(3p)	-30665.539	-30664.19	-30648.86	-30604.85	0.999956009	0.999500068	0.998564058	
g04(4p)	-30665.539	-30664.62	-30657.03	-30634.81	0.999970032	0.999722522	0.998997931	
g09(1p)	680.6300573	680.9599	681.6192	683.7651	0.999515621	0.998548834	0.99541503	
g09(2p)	680.6300573	680.7499	681.6655	683.2584	0.999823955	0.998481011	0.996153223	2.99446
g09(3p)	680.6300573	680.9414	682.5332	687.1950	0.999542776	0.997667806	0.99321619	
g09(4p)	680.6300573	681.2212	682.5125	686.8906	0.999132231	0.99810802	0.993626205	
							calif total	8.84334

Tabla 6.5. Resultados obtenidos por Coello y Cruz en 2002.

6.3.4 Hernández, Botello, Lizárraga y Coello 2003.

En 2003 se presenta *IS-PAES: A Cointraint-Handling Technique Based on Multiobjetive Optimization Concepts* [22]. El IS-PAES es un algoritmo evolutivo que utiliza los frentes de Pareto para hacer optimización numérica. Solo se prueba con 10 de las 13 funciones, excluye g5, g12 y g13. Los resultados se presentan en la tabla 6.6. La calificación obtenida bajo el criterio de 6.2 es de ***29.60567***.

Problema	óptimo	mejor	media	peor	ξ mejor	ξ media	ξ peor	
g01 (MIN)	-15	-14.9997	-14.494	-12.446	0.99998	0.966266667	0.829733333	
g02 (MAX)	0.803619	0.803376	0.793281	0.768291	0.999697618	0.987135695	0.956038869	
g03 (MAX)	1	1.000	1.000	1.000	1	1	1	
g04 (MIN)	-30665.539	-30665.539	-30665.539	-30665.539	1	1	1	
g05 (MIN)	5126.4981				0	0	0	
g06 (MIN)	-6961.81388	-6961.814	-6961.813	-6961.810	0.999999983	0.999999874	0.999999443	
g07 (MIN)	24.3062091	24.338	24.527	24.995	0.998693775	0.990998047	0.972442853	
g08 (MAX)	0.095825	0.095825	0.095825	0.095825	1	1	1	
g09 (MIN)	680.6300573	680.630	680.631	680.631	0.999999916	0.999998615	0.999998615	
g10 (MIN)	7049.3307	7062.019	7342.944	7448.014	0.998203304	0.960014226	0.946471194	
g11 (MIN)	0.75	0.750	0.750	0.750	1	1	1	
g12 (MAX)	1				0	0	0	
g13 (MIN)	0.0539498				0	0	0	Calif total
Calificaciones					9.996574596	9.904413123	9.704684306	29.60567

Tabla 6.6. Resultados de IS-PAES.

6.3.5 Hernández 2004.

En 2004 R. Paramont Hernández propone el algoritmo de AMRE [35], [36], el cual es un algoritmo genético codificado en binario. El algoritmo MARACoR propuesto en esta tesis, es una línea de investigación propuesta en [36]. Tabla 6.7. La calificación obtenida bajo el criterio de 6.7 es de ***36.12761***.

Problema	óptimo	mejor	media	peor	ξ mejor	ξ media	ξ peor	
g01 (MIN)	-15	-15	-15	-15	1	1	1	
g02 (MAX)	0.803619	0.767753	0.694139	0.60775	0.955369398	0.863766287	0.75626634	
g03 (MAX)	1	0.992328	0.938526	0.815843	0.992328	0.938526	0.815843	
g04 (MIN)	-30665.539	-30664.571	-30659.671	-30636.136	0.999968434	0.999808645	0.999041171	
g05 (MIN)	5126.4981	5126.485	5162.439	5761.968	0.999997445	0.993038	0.889713046	
g06 (MIN)	-6961.81388	-6961.812	-6961.584	-6960.393	0.99999973	0.99996698	0.999795904	
g07 (MIN)	24.3062091	24.672	26.727	36.975	0.985173845	0.909425267	0.657368738	
g08 (MAX)	0.095825	0.095825	0.095825	0.095825	1	1	1	
g09 (MIN)	680.6300573	680.806	681.143	681.849	0.999741567	0.999246938	0.998212298	
g10 (MIN)	7049.3307	7052.574	7406.532	9198.724	0.999540125	0.951772125	0.766337886	
g11 (MIN)	0.75	0.75	0.75	0.75	1	1	1	
g12 (MAX)	1	1	1	1	1	1	1	
g13 (MIN)	0.0539498	0.05394	0.10084	0.44026	0.99981835	0.535003967	0.122540771	Calif total
Calificaciones					12.93193689	12.19055421	11.00511915	36.12761

Tabla 6.7. Resultados de EAAMRE.

En la tabla 6.8, 6.9 y 6.10 se presentan los diferentes trabajos ordenados por la calificación obtenida. La calificación máxima posible es 39 para 13 funciones, 30 para el caso de 10 funciones y 9 para el caso de 3 funciones.

Trabajo	Calificación
MARACoR mejores	38.10771
MARACoR promedio	37.72030
Runarsson y Yao	37.53328
AMRE	36.12761
Koziel y Michalewicz	33.27785

Tabla 6.8 Trabajos evaluados con las 13 funciones.

Trabajo	Calificación
MARACoR mejores	29.71220
MARACoR promedio	29.63425
IS-PAES	29.60567

Tabla 6.9 Trabajos evaluados con 10 de las 13 funciones.

Trabajo	Calificación
MARACoR mejores	8.99680
MARACoR promedio	8.99539
Coello y Cruz	8.84334

Tabla 6.10 Trabajos evaluados con 3 de las 13 funciones.

6.4 Conclusiones.

El algoritmo propuesto se inspira en el mecanismo de maduración de la afinidad de los anticuerpos del sistema inmune y éste ha sido aplicado a la solución de problemas de programación no lineal para optimización numérica restringida a través de un algoritmo evolutivo, el cual responde bien al conjunto de pruebas al que fue sometido [40] (Ver apéndice A). Aunque en algunos casos los mejores valores encontrados son ligeramente superados por los encontrados en otros trabajos, lo robusto del algoritmo le da las mejores calificaciones. Además por haber sido codificado en reales nos da más precisión en algunos resultados como en g6 y en g8.

Bibliografía.

[1] A. Kuri, A Comprehensive Approach to Genetic Algorithms in Optimization and Learning, Colección de Ciencia de la Computación, IPN, México 1999.

[2] A. Tarakanov and D. Dasgupta, An immunochip archtecture and its emulation, in 2002 NASA/DoD Conference on Evoluable Hardware, 2002.

[3] Alexander Tarakanov, Dipankar Dasgupta, A formal model of an artificial immune system, BioSystem, 2000.

[4] C. A. Coello and N. Cruz Cortés, A parallel implementation of the artificial immune system to handle constraints in genetic algorithms: preliminary results, in Proceedings of the 2002 Congress on Evolutionary Computation CEC2002, Honolulu, 2002.

[5] Carlos A. Coello Coello, A Survey of Constraint Handling Techniques used with Evolutionary Algorithms, Lania-IR-99-05, Laboratorio Nacional de Informática Avanzada, 1999.

[6] D. Bradley and A. Tyrell, Immunotronics: Hardware fault tolerance inspired by the immune system, in proceedings of the 3rd`International conference on Evoluable Systems (ICES2000), vol. 1801, Springer-verlag, Inc., 2000 GECCO, 2002.

[7] D. Dasgupta, Immune-based intrusion detection system: a general framework, in Proceedings of the 22nd national information systems security conferencfe (NISSC), 1999.

[8] D. Whitley. The GENITOR Algorithm and Selective Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. Proc. 3th International Conf. on Genetic Algorithms. D. Schaffer, ed., pp. 116-121. Morgan Kaufmann, 1989.

[9] Darwin Charles, El origen de las especies, Editorial Porrua, México, 2002.

[10] De Castro, L. N. & Von Zuben, F. J. (2000), "The Clonal Selection Algorithm with Engineering Applications", (full version, pre-print), In Proceedings of GECCO'00, Workshop on Artificial Immune Systems and Their Applications, pp. 36-37.

[11] De Castro, L. N. & Von Zuben, F. J. (2001), "aiNet: An Artificial Immune Network for Data Analysis", (full version, pre-print), Book Chapter in Data Mining: A Heuristic Approach, H. A. Abbass, R. A. Sarker, and C. S. Newton (eds.), Idea Group Publishing, USA, Chapter XII, pp. 231-259.

[12] De Castro, L. N. & Von Zuben, F. J. (2001c), "An Immunological Approach to Initialize Feedforward Neural Network Weights", Proceedings do ICANNGA 2001 (International Conference on Artificial Neural Networks and Genetic Algorithms), pp. 126-129.

[13] Dipankar Dasgupta, F. González, Artificial immune system research in the las five years, Proceedings of the 2003 Congress on Evolutionary Computation CEC2003, pp 123—130, IEEE Press, Canberra, 2003.

[14] Dipankar Dasgupta, Senhua Yu, Nivedita Sumi Majumdar, MILA - Multilevel Immune Learning Algorithm, In the proceedings of GECCO, July 2003.

[15] E. Hart and P Ross, Exploiting the analogy between immunology and Spare Distributed Memories:A System for Clustering Non-stationary data, 1st ICARIS, 2002.

[16] E.L. Cooper, Innate Immunity, Vol. 3, pp. 1387-1389.

[17] F. Herrera, M. Lozano, J.L. Verdegay. Tackling Real-Coded Genetic Algorithms: Operators and tools for the Behaviour Analysis. Artificial Intelligence Review 12 (1998) pp. 265-319.

[18] Floudas, Christodoulos A. and Pardalos, Panos M., A Collection of Test Problems for Constrained Global Optimization Algorithms, Springer, Berlin, Heidelberg, New York 1990.

[19] G. W. Litman, M.K. Anderson and J.P. Rast, Evolution of Antigen Binding Receptors, Annu. Rev. Immunology, Vol. 17, pp. 109-147.

[20] H.-P.Schwefel, Evolution and optimum seeking. Wiley, New York, 1995.

[21] H.-P.Schwefel, Evolutionstrategie und numrische optimierung. PhD Thesis, Technische Universität Berlin, Berlin, 1975.

[22] Hernández Aguirre, Arturo; Botello Rionda, Salvador, Lizárraga Lizárraga, Giovanni and Coello Coello, Carlos A. IS-PAES: A Constraint-Handling Technique Based on Multiobjective Optimization Concepts, in Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb and Lothar Thiele (Eds), Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003, pp. 73--87, Springer, Lecture Notes in Computer Science, Vol.

[23] Hoffmann, J.A., Kafatos, F.C., Janeway, C.A., and Ezekowitz, R.A.B. (1999). Phylogenetic perspectives in innate immunity. Science 284, 1313-1318.

[24] I.Rechenberg, Evolutionsstrategie. Fromman-Holzboog, Stuttgart,1973.

[25] J. Hunt, J. Ti mmis, D. Cooke, M. Neal and C. King, Jisys: The Development of an Artificial Immune System for Real World Applications, in Artificial Immune System and Their Applications, 1999 Function optimization, WCCI 2002, 2002.

[26] J. Kim and P. J. Bentley, Toward an artificial immune system system for network intrusion detection: An investigation of dynamic clonal selection, in Proceedings of the 2002 Congress on Evolutionary Computation CEC2002. Honolulu, 2002.

[27] John H. Holland, Adaptation in Natural and Artificial Systems, The MIT Press, Cambridge, Massachusetts, 1992.

[28] Jon I. Timmis, Artificial Immune Systems: A Novel Data Analysis Technique Inspired by the Immune Network Theory, Dissertation, University of Wales, 2000.

[29] Miller, J.F.A.P. (1999). Discovering the origins of immunological competence. Annual Review of Immunology 17, 1-18.

[30] N. K. Jerne, Towards a network theory of the immune system. Ann. Immunol, (Inst. Pasteur), 125C, pp373-389, 1974. Honolulu, 2002.

[31] O. Narasoui, F. González, C. Cardona, C. Rojas, and D. Dasgupta, A Scalable Artificial Immune System Model for Dynamic Unsupervised Learning, GECCO, 2003.

[32] O. Nasaroui, F. González and D Dasgupta, The Fuzzy AIS: Motivations, Basic Concepts, and Applications to Clustering and Web Profiling, IEEE International Conference on Fuzzy Systems, 2002.

[33] P. D. Williams, K. P. Anchor, J. L. Bebo, G. H. Gunsh, and G. D. Lamont, CDIS: Towards a computer immune system for detecting network intrusions, Lecture notes in Computer Science, vol. 2212, pp. 117-133, 2001 Timmis, Artificial Immune System: A New Computational Intelligence Approach, Springer-Verlag, 2002.

[34] P. Hajela, J. Yoo, Constrained genetic search via schema adaptation an immune network solution, Structural Optimization, 12, pp. 11-15, 1996.

[35] R.P. Hernández García, An Evolutionary Algorithm Partially Inspired in the Vertebrates' Immune System for Performing Numerical Optimization, In the proceedings, of IASTED international conference on artificial intelligence and applications, February 2004.

[36] R.P. Hernández García, Uso de Modelos del Sistema Inmune para Manejo de Restricciones con Algoritmos Genéticos, Proyecto de Tesis para obtener el grado de doctor en ciencias de la computación, Abril 2003.

[37] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-Nonself Discrimination in a Computer. In Proceedings of IEEE Symposium on Research in Security and Privacy, pages202--212, Oakland, May 16-18 1994.

[38] S. Hofmeyr and S. Forrest, Architecture for an artificial immune system, Evolutionary Compuiation, vol. 8, no. 4, pp. 443-473, selection, in Proceedings of the 2002 Congress on Evolutionary Computation CEC2002. Honolulu.

[39] S. Koziel, Z. Michalewicz, Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization, MIT, Evolutionary Computation, vol. 7, no. 1, pp. 19-44, 1999.

[40] T.P. Runarsson, Xin Yao, Stochastic Ramking for Constrained Evolutionary Optimization, IEEE Transactions on Evolutionary Computation, Vol. 4, no. 3, pp.284-294, 2000.

[41] Z. Michalewicz, Genetic Algorithms, Numerical Optimization, and Constraints, en Larry J. Eshelman, ed., Proceedings of the Sixth International Conference on Genetic Algorithms, University of Pittsburgh, pp. 344-350, Morgan Kaufmann Publishers, San Francisco, 1995.

[42] Zbigniew Michalewicz, A perspective on evolutionary computation ,Proceedings of the Workshop on Evolutionary Computation, November 21-22, 1994, University of New England, Armidale, Australia, pp. 76-93.

[43] Zbigniew Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 3rd revision Edition, 1997.

Apéndice A.

Colección de funciones utilizadas para probar los algoritmos propuestos, tomadas de las presentadas por Runarsson y Yao en [40].

Problema g01

Minimizar $f(\boldsymbol{x}) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=1}^{13} x_i$

Sujeta a:

1) $2x_1 + 2x_2 + x_{10} + x_{11} \leq 10$,
2) $-8x_1 + x_{10} \leq 0$,
3) $-2x_4 - x_5 + x_{10'} \leq 0$,
4) $2x_1 + 2x_3 + x_{10} + x_{12} \leq 10$,
5) $-8x_2 + x_{11} \leq 0$,
6) $-2x_6 - x_7 + x_{11} \leq 0$,
7) $2x_2 + 2x_3 + x_{11} + x_{12} \leq 10$,
8) $-8x_3 + x_{12} \leq 0$,
9) $-2x_8 - x_9 + x_{12} \leq 10$

Los límites de las variables son:

$0 \leq x_i \leq 1$, $i = 1, \ldots, 9$,
$0 \leq x_i \leq 100$, $i = 10, 11, 12$,
$0 \leq x_{13} \leq 1$

Óptimo: $f(\boldsymbol{x}) = -15$, $\boldsymbol{x} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$

Problema g02

Maximizar $f(\boldsymbol{x}) = \left| \dfrac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i\, x_i^2}} \right|$

Sujeta a:

$\prod_{i=1}^{n} x_i \geq 0.75$,
$\sum_{i=1}^{n} x_i \leq 7.5\, n$

Los límites de las variables son:

$0 \leq x_i \leq 10$, i=1, . . . , n.

Donde $n = 20$.
Óptimo: $f(\boldsymbol{x}) = 0.803619$

Problema g03

Maximizar $f(\boldsymbol{x}) = (\sqrt{n})^n \cdot \prod_{i=1}^{n} x_i$

Sujeta a:

$\sum_{i=1}^{n} x_i^{\,2} = 1$

Las variables son:

$0 \le x_i \le 1 \qquad \text{para } i = 1, \ldots, n$

Donde $n = 10$

Óptimo: $f(\boldsymbol{x}) = 1$

$$(x_1, \ldots, x_n) = \left(\frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}} \right)$$

Problema g04

Minimizar $f(\boldsymbol{x}) = 5.3578547\, x_3^2 + 0.8356891\, x_1 x_5 + 37.293239\, x_1 - 40792.141$

Sujeta a:

1) $0 \le 85.334407 + 0.0056858\, x_2\, x_5 + 0.0006262 \quad x_1\, x_4 - 0.0022053\, x_3\, x_5$
2) $85.334407 + 0.0056858\, x_2\, x_5 + 0.0006262\, x_1\, x_4 - 0.0022053\, x_3\, x_5 \le 92$
3) $90 \le 80.51249 + 0.0071317\, x_2\, x_5 + 0.0029955\, x_1\, x_2 + 0.0021813\, x_3^{\,2}$
4) $80.51249 + 0.0071317\, x_2\, x_5 + 0.0029955\, x_1\, x_2 + 0.0021813\, x_3^{\,2} \le 110$
5) $20 \le 9.300961 + 0.0047026\, x_3\, x_5 + 0.0012547\, x_1\, x_3 + 0.0019085\, x_3\, x_4$
6) $9.300961 + 0.0047026\, x_3\, x_5 + 0.0012547\, x_1\, x_3 + 0.0019085\, x_3\, x_4 \le 25$

Los límites de las variables son:

$78 \le x_1 \le 102,$
$33 \le x_2 \le 45,$
$27 \le x_i \le 45,\ i = 3, 4, 5$

Óptimo: $f(\boldsymbol{x}) = -30665.539$, $\boldsymbol{x} = (78,\ 33,\ 29.995256025682,\ 45,\ 36.775812905788)$

Problema g05

Minimizar $f(\boldsymbol{x}) = 3\,x_1 + 0.000001\,x_1^3 + 2\,x_2 + 0.000002/3\,x_2^3$

Sujeta a:

1) $x_4 - x_3 + 0.55 \geq 0$
2) $x_3 - x_4 + 0.55 \geq 0$
3) $1000\,\text{sen}(-x_3 - 0.25) + 1000\,\text{sen}(-x_4 - 0.25) + 894.8 - x_1 = 0,$
4) $1000\,\text{sen}(x_3 - 0.25) + 1000\,\text{sen}(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0,$
5) $1000\,\text{sen}(x_4 - 0.25) + 1000\,\text{sen}(x_4 - x_3 - 0.25) + 1294.8 = 0,$

Los límites de las variables son:

$0 \leq x_i \leq 1200,\ i = 1,2$
$-0.55 \leq x_i \leq 0.55,\ i = 3, 4$

Óptimo: $f(\boldsymbol{x}) = 5126.4981$, $\boldsymbol{x} = (679.9453, 1026.067, 0.1188764, -0.3962336)$

Problema g06

Minimizar $f(\boldsymbol{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$

Sujeta a:

1) $(x_1 - 5)^2 + (x_2 - 5)^2 - 100 \geq 0,$
2) $-(x_1 - 6)^2 - (x_2 - 5)^2 + 82.81 \geq 0$

Los límites de las variables son:

$13 \leq x_1 \leq 100$
$0 \leq x_2 \leq 100$

Óptimo: $f(\boldsymbol{x}) = -6961.81388$, $\boldsymbol{x} = (14.095, 0.84296)$

Problema g07

Minimizar

$$f(\boldsymbol{x}) = x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 +$$
$$7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

Sujeta a:

1) $105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0$,
2) $-10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0$,
3) $8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0$,
4) $3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 10$
5) $-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0$,
6) $-x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geq 0$
7) $-5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0$,
8) $-0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0$

Los límites de las variables son:

$-10.0 \leq x_i \leq 10.0$, $i = 1, \ldots, 10$.

Óptimo: $f(\boldsymbol{x}) = 24.3062091$, $\boldsymbol{x} = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$

Problema g08

Maximizar $f(\boldsymbol{x}) = \dfrac{\text{sen}^3(2\pi x_1)\,\text{sen}(2\pi x_2)}{x_1^3(x_1 + x_2)}$,

Sujeta a :

1) $x_1^2 - x_2 + 1 \leq 0$,
2) $1 - x_1 + (x_2 - 4)^2 \leq 0$

Los límites de las variables son:

$0 \leq x_1 \leq 10$
$0 \leq x_2 \leq 10$

Óptimo: $f(\boldsymbol{x}) = 0.0958215$, $\boldsymbol{x} = (1.2279713, 4.2453733)$

Problema g09

Minimizar

$$f(\boldsymbol{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

Sujeta a:

1) $127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0$,
2) $282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 + x_5 \geq 0$,
3) $196 - 23x_1 - x_2^2 - 6x_6^2 + 8x_7 \geq 0$,
4) $-4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0$

Los límites de las variables son:

$-10 \leq x_i \leq 10.0, \text{i} = 1,\ldots,7$

Óptimo: $f(\boldsymbol{x}) = 680.6300573$, $\boldsymbol{x}$ = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)

Problema g10

Minimizar $f(\boldsymbol{x}) = x_1 + x_2 + x_3$

Sujeta a:

1) $1 - 0.0025(x_4 + x_6) \geq 0,$
2) $1 - 0.0025(x_5 + x_7 - x_4) \geq 0,$
3) $1 - 0.01(x_8 - x_5) \geq 0,$
4) $x_1 x_6 - 833.33252\, x_4 - 100x_1 + 83333.333 \geq 0,$
5) $x_2 x_7 - 1250x_5 - x_2 x_4 + 1250x_4 \geq 0,$
6) $x_3 x_8 - 1250000 - x_3 x_5 + 2500x_5 \geq 0.$

Los límites de las variables son:

$100 \leq x_1 \leq 10000,$
$1000 \leq x_i \leq 10000,\ i = 2, 3,$
$10 \leq x_i \leq 1000,\ i = 4,\ldots,8$

Óptimos: $f(\boldsymbol{x}) = 7049.3307$, $\boldsymbol{x}$ = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)

Problema g11

Minimizar $f(\boldsymbol{x}) = x_1^2 + (x_2 - 1)^2$,

Sujeta a:

$x_2 - x_1^2 = 0$

Los límites de las variables son:

$-1 \leq x_i \leq 1,\ i = 1,2.$

Óptimos: $f(\boldsymbol{x}) = 0.75000455$, $\boldsymbol{x}$ = (0.70711, 0.5) y $f(\boldsymbol{x}) = 0.75000455$, $\boldsymbol{x}$ = (-0.70711, 0.5)

Problema g12

Maximizar $f(\boldsymbol{x}) = (100 - (x_1 - 5)^2 + (x_2 - 5)^2 - (x_3 - 5)^2)/100$

Sujeta a:

$(x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$

Donde:

$p, q, r = 1, \ldots, 9.$

Los límites de las variables son:

$0 \leq x_i \leq 10, \; i = 1, 2, 3.$

Óptimo: $f(\boldsymbol{x}) = 1$, $\boldsymbol{x} = (5, 5, 5)$

Problema g13

Minimizar $f(\boldsymbol{x}) = e^{x_1 x_2 x_3 x_4 x_5}$

Sujeta a:

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0$$
$$x_2 x_3 - 5 x_4 x_5 = 0$$
$$x_1^3 + x_2^3 + 1 = 0$$

Los límites de las variables son:

$-2.3 \leq x_i \leq 2.3, \; i = 1, 2$
$-3.2 \leq x_i \leq 3.2, \; i = 3, 4, 5.$

Óptimo: $f(\boldsymbol{x}) = 0.0539498$, $\boldsymbol{x} = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)$