Uso de Emulaciones del Sistema Inmune para Manejo de Restricciones en Algoritmos Evolutivos

Nareli Cruz Cortés

Maestría en Inteligencia Artificial

Asesor: Dr. Carlos A. Coello Coello

Laboratorio Nacional de Informática Avanzada

Revisor: Dr. Angel Kuri Morales

Centro de Investigación en Computación-IPN

Revisor: Dr. Edgar Vallejo Clemente

ITESM Campus Estado de México

Diciembre, 2000

Resumen

Los algoritmos evolutivos han sido utilizados muy exitosamente como técnicas de optimización durante los últimos años.

Sin embargo, dado que son técnicas que operan sin restricciones, se les ha tenido que acoplar un mecanismo que permita incorporar restricciones de todo tipo (lineales, no lineales, igualdad y desigualdad) a la función de aptitud.

Aunque tradicionalmente se han utilizado funciones de penalización para incorporar restricciones en los algoritmos evolutivos, existe una cantidad significativa de trabajo en torno a enfoques alternativos. En esta tesis se explora una de ellas que está basada en el uso de emulaciones del sistema inmune.

El sistema inmune biológico es un complejo sistema de procesamiento de información que ha servido de inspiración para realizar simulaciones por computadora que buscan dar solución a diversos problemas planteados por la ciencia y la tecnología, pricipalmente orientados a la protección contra individuos invasores.

Esta simulación se basa en el principio de reconocimiento de los organismos extraños que intentan invadir al organismo. La tarea de reconocimiento se lleva a cabo por los anticuerpos, los cuales deben de combinarse entre sí para lograr la especifidad adecuada al antígeno invasor, es decir, el sistema debe aprender los anticuerpos correctos para tal antígeno.

Al tomar este modelo del sistema inmune al dominio de optimización glo-

bal, el aprendizaje lo realiza un algoritmo genético, que busca crear similitud a nivel del genotipo de los individuos que violan restricciones con aquellos que no lo hacen. De esta manera se disminuye, o elimina en algunos casos, la violación a las restricciones del problema.

El trabajo de esta tesis es una extensión a una propuesta reportada previamente en la literatura [26, 27], la cual adolecía de algunas limitantes. De entre ellas, las más importantes limitaciones del algoritmo original eran el hecho de que requería de al menos una solución factible en la generación inicial del algoritmo genético, y que planteaba mapeos genotípicos aplicables únicamente a alfabetos binarios. Ambas limitantes fueron superadas en el trabajo aquí reportado.

El algoritmo propuesto se aplica a varias funciones de prueba propuestas por investigadores del área de optimización evolutiva, a fin de validar su funcionamiento. Los resultados obtenidos hacen ver que el uso de emulaciones del sistema inmune para manejo de restricciones es una alternativa viable y muy prometedora.

Agradecimientos

Agradezco a mis padres y a mi tía Elvia por todo su apoyo.

Gracias al Dr. Carlos Coello Coello por su ayuda y orientación.

Agradezco el apoyo otorgado por CONACyT a través de una beca para cursar la Maestría en Inteligencia Artificial del LANIA y la Universidad Veracruzana. Asimismo, agradezco la beca terminal otorgada por CONACyT para concluir esta tesis de maestría, a través del proyecto NSF-CONACyT No. 32999-A, el cual se titula "Estudio y Desarrollo de Técnicas Avanzadas de Manejo de Restricciones para Algoritmos Evolutivos en el Contexto de Optimización Numérica".

Índice General

Índice de Figuras

Introducción

Existen en la naturaleza una infinidad de sistemas biológicos que son considerados como complejos sistemas de procesamiento de información. Muchos de ellos han servido como inspiración en la generación de ideas que buscan resolver problemas que se plantean la ciencia y la tecnología.

Algunos de los sistemas biológicos que se han simulado con tales fines son las redes neuronales, los sistemas evolutivos y el sistema inmune, entre otros.

La computación evolutiva ha recibido una atención creciente debido en parte, a que ha demostrado tener un algoritmo de búsqueda con gran robustez, simplicidad conceptual y procesamiento en paralelo.

Asimismo, el sistema inmune es actualmente objeto de gran interés por parte de los investigadores, que buscan comprender su mecanismo de operación, así como crear simulaciones de éste, principalmente enfocados a la protección contra individuos invasores.

La computación evolutiva, particularmente los algoritmos genéticos, y el sistema inmune artificial dan la base al algoritmo que se estudia en esta tesis, combinando los beneficios de ambas técnicas, aplicadas al manejo de restricciones en optimización numérica.

El manejo de restricciones en optimización numérica es un problema que se ha tratado de resolver por diversas técnicas. Esta tesis es un estudio de una alternativa a ellas que es la simulación del sistema inmune para manipular tales restricciones, y que usa como motor de búsqueda un algoritmo genético.

El documento está organizado en cinco capítulos; en el capítulo 1 se da una breve introducción a la computación evolutiva haciendo énfasis en los algoritmos genéticos; el capítulo 2 aborda la optimización numérica con manejo de restricciones y se mencionan algunas de las técnicas que existen. En el capítulo 3 se describe de manera general cuál es el funcionamiento del sistema inmune tal como se lleva a cabo en la naturaleza por los seres vivos (principalmente los vertebrados). El capítulo 4 muestra de manera muy general la forma en que se modela el sistema inmune en una computadora, así como una descripción del algoritmo del sistema inmune para manejo de restricciones; y finalmente, en el capítulo 5 se muestran las motivaciones para realizar la implementación del algoritmo, algunas propuestas que tienen la intención de mejorarlo, así como la descripción de las funciones a las cuales fue aplicado y los resultados arrojados por él. Finalizamos con el análisis de resultados, las conclusiones derivadas de ellos y algunas de las rutas posibles de trabajo futuro.

Capítulo 1

COMPUTACIÓN EVOLUTIVA

La computación evolutiva se refiere a un conjunto de técnicas de búsqueda y optimización inspiradas en el proceso de selección natural (la supervivencia del más apto) que reina en las especies del planeta.

En este capítulo se da una breve introducción a ella, iniciando con algunos antecedentes históricos, y discutiendo luego brevemente el paradigma en el que se basa y se conoce como Neo-Darwinismo. Finalmente, se describen los principales paradigmas que integran la computación evolutiva.

1.1 Antecedentes históricos

A partir del paradigma conocido como Neo-Darwiniano, se puede explicar la vida en el planeta, con base en cuatro principales procesos que son: reproducción, mutación, competencia y selección.

El Neo-Darwinismo es el resultado de tres teorías principalmente: el seleccionismo de Weismann, la genética de Mendel y la teoría de la evolución de Charles Darwin.

1.1.1 El seleccionismo de Weismann

El francés Jean Baptiste Pierre Antoine de Monet (Caballero de Lamarck) [32] en su filosofía zoológica que data de 1809, expuso su teoría de que las características adquiridas por un individuo durante su vida, son codificadas genéticamente y heredadas a sus descendientes de manera directa. Esto es, que el ambiente influye directamente sobre las características físicas de los individuos, pues cuanto más se usa una estructura más se acentúa y vice–versa. Esto se conocería más tarde se conocería como la doctrina del uso y desuso.

Esta teoría Lamarckiana fue refutada de manera implícita por primera vez, por el científico alemán August Weismann (1834-1914), al afirmar que [32, 3] iniciando con el óvulo fertilizado, existen dos procesos independientes de división de la célula. Uno conlleva al cuerpo o 'soma', (hoy conocido como fenotipo), y el otro (la línea germinal, hoy conocido como genotipo) conlleva a los gametos que forman el punto de inicio de la siguiente generación. De estas dos líneas celulares, el soma morirá, pero la línea germinal es potencialmente inmortal. La línea germinal es capaz de transmitir información hereditaria, mientras que el soma no puede hacerlo.

La conclusión de Weismann es conocida como la teoría del germoplasma, y afirma que la selección natural es el único mecanismo capaz de cambiar la línea germinal, mientras que tanto esa línea germinal como el ambiente pueden influenciar al soma.

1.1.2 Charles Darwin

La publicación del libro titulado "El Origen de las Especies" por el biólogo inglés Charles Darwin [12], en el año de 1859, vendría a revolucionar las ciencias biológicas e inclusive la manera de cómo el hombre se ve a sí mismo, ahora como resultado de un largo proceso de adaptación al medio ambiente.

A través de sus largos viajes, Darwin se dio cuenta de que la especie que no sufriera cambios se volvería incompatible con el entorno, puesto que éste tiende a cambiar con el tiempo. Asimismo, las similitudes entre padres e hijos observada en la naturaleza, le sugirieron que ciertas características de las especies eran hereditarias y de generación en generación ocurrían cambios cuya principal motivación era hacer a los nuevos individuos más aptos para sobrevivir.

De tal manera, Darwin infirió que debía existir una competencia por la sobrevivencia, existiendo especies con más capacidad de sobrevivir que otras. Darwin aceptó la teoría Lamarckiana bajo el término de "uso y desuso", pero pensaba que la selección natural es lo más importante.

Sus conclusiones ponen al descubierto la necesidad de tener una explicación a las variaciones en los seres vivos. La respuesta a ello sería encontrada años después por Gregor Mendel.

1.1.3 Genética de Gregor Mendel

Gregor Mendel [3] nació en el año 1822 en Heinzendorf Austria, y es conocido como el padre de la genética, pues su trabajo ha cimentado la base para esta ciencia.

Siendo monje, inició sus experimentos con plantas de chícharos (guisantes) en el año de 1856 (tres años antes de la publicación del origen de las especies de Darwin). Hizo cruzas durante ocho generaciones, seleccionando para hacer sus estudios, siete características de las plantas que eran independientes entre sí, llevando un registro cuidadoso y detallado de sus observaciones.

Fue el primero en señalar que existen factores de herencia que hoy se conocen como genes y cromosomas, los cuales explican la variabilidad en los seres vivos.

Concluyó que no existen mezclas de genes como se creía hasta entonces, sino que sólo se combinan en la reproducción conservando su individualidad a través de las generaciones. El resultado de estas combinaciones puede ser estimado estadísticamente.

1.2 Neo-Darwinismo

El pensamiento resultante del evolucionismo de Weismann, la genética de Mendel y la teoría evolutiva de Darwin forman el Neo-Darwinismo, que explica la evolución como un proceso de adaptación al medio ambiente donde tienen mayor probabilidad de sobrevivir las especies con más aptitud.
Fogel [17, 18] resume el Neo-Darwinismo en siete características:

1. El individuo es el blanco principal de la selección.
2. La variación genética es en gran parte un fenómeno debido al azar. Los procesos estocásticos juegan un papel importante en la evolución. 1
3. La evolución "gradual" puede incorporar discontinuidades fenotípicas.
4. No todos los cambios fenotípicos son necesariamente consecuencia de una selección natural "ad hoc"
5. La evolución es un cambio en la adaptación y la diversidad, y no simplemente un cambio en la frecuencia de los genes.
6. La selección es probabilística y no determinística.

Fogel explica la evolución como un proceso de optimización, con las siguientes palabras [17]:

> "Si consideramos a la evolución natural como un proceso de optimización, debido a que la evolución ha sido capaz de optimizar

> organismos hasta hacerlos aptos para sobrevivir, entonces, de modelarse ésta adecuadamente, puede emplearse para encontrar la mejor solución de un problema, es decir, la solución óptima"

1.3 Computación Evolutiva

La computación evolutiva está inspirada en el Neo-Darwinismo. Intenta simular este proceso evolutivo en la computadora. Para realizar su modelado son necesarios cinco componentes [38]:

1. Una representación para las soluciones potenciales al problema
2. Una manera de crear una población inicial de dichas soluciones potenciales
3. Una función de evaluación que juega el papel del ambiente, comparando las soluciones en términos de su "aptitud"
4. Operadores genéticos que alteran la composición de la descendencia
5. Valores para los parámetros que usa la técnica (tamaño de la población, probabilidades de aplicar los operadores genéticos, etc.)

La computación evolutiva es una técnica de búsqueda heurística, lo que significa, en las palabras de Reeves [42]:

> "Una heurística es una técnica que busca soluciones buenas (es decir casi óptimas) a un costo computacional razonable, aunque sin garantizar factibilidad u optimalidad de las mismas. En algunos casos, ni siquiera puede determinar qué tan cerca del óptimo se encuentra una solución factible en particular"

La computación evolutiva suele usarse en problemas de búsqueda y optimización del mundo real que son difíciles de resolver por técnicas clásicas, debido a que los espacios de búsqueda son extremadamente grandes, no convexos y frecuentemente con restricciones difíciles de satisfacer.

A diferencia de las técnicas tradicionales, en las técnicas evolutivas existe un conjunto de soluciones potenciales, y no sólo una como en aquellas, lo que implica un alto grado de paralelismo, pues se explotan a la vez diferentes regiones del espacio de búsqueda.

Tiene la particularidad adicional de tener operadores probabilísticos; el hecho de que éstos sean probabilísticos y no determinísticos, evita que pueda quedar atrapados fácilmente en óptimos locales.

Dentro de la Inteligencia Artificial se considera a las técnicas evolutivas como sub-simbólicas, debido a que su representación del conocimiento es numérica (como en las redes neuronales) y no simbólicas (como los sistemas expertos)

Terminología

Alguna terminología básica usada por la comunidad de computación evolutiva es la siguiente [6]:

Un **cromosoma** es una estructura de datos que posee una cadena de parámetros o genes. Esta estructura puede ser almacenada, por ejemplo, como una cadena de bits (representación binaria) o como un arreglo de números reales (representación de punto flotante o representación de código real). Este cromosoma es análogo a los cromosomas base 4 presentes en nuestro ADN. Dentro de la comunidad de algoritmos genéticos, suele usarse un modelo haploide de una célula (un cromosoma individual). Sin embargo, los diploides también han sido usados en el pasado. (Ver por ejemplo [29, 28, 24])

Un **gen** es un subconjunto de un cromosoma que usualmente codifica el valor de un solo parámetro del problema (una variable de decisión).

Un **alelo** es el valor de un gen (o una de sus posiciones). Por ejemplo, para la representación binaria cada alelo puede tener un valor de 0 o 1.

Un **esquema** es un patrón de valores de los genes en un cromosoma, el cual puede incluir estados "no importa"(representados por el símbolo #). Así, en un cromosoma binario, cada esquema puede ser especificado por una cadena de la misma longitud que el cromosoma, con cada carácter siendo uno del alfabeto {0, 1, # }. Se dice que un cromosoma particular contiene un esquema particular si éste corresponde con el esquema (por ejemplo, el cromosoma 01101 corresponde al esquema #1#0#)

Si la solución de un problema puede ser representada por un conjunto de N parámetros reales, entonces el trabajo de encontrar esta solución puede concebirse como una búsqueda en un espacio de N dimensiones. Esta región es simplemente referida como el **espacio de búsqueda del problema.**

La **aptitud** de un individuo es el valor que refleja su desempeño (por ejemplo, que tan bien resuelve cierta tarea). Una función de aptitud es un mapeo de los cromosomas en una población a sus correspondientes valores de aptitud. Un horizonte o mapa de aptitud es la hipersuperficie obtenida de aplicar la función de aptitud a cada punto en el espacio de búsqueda.

Un **bloque constructor** es un pequeño grupo de genes contiguos, los cuales co-evolucionan de tal modo que su introducción en cada cromosoma resultará en un incremento en la aptitud de tal cromosoma. [1]

La hipótesis de los bloques constructores establece que los algoritmos genéticos construyen soluciones encontrando primero tantos bloques constructores como sea posible, y combinándolos luego de tal manera que se obtenga una aptitud más elevada a la actual.

Elitismo (o una estrategia elitista) es un mecanismo el cual asegura que los cromosomas del(os) miembro(s) con más alta aptitud de la población son pasados a la siguiente generación sin ser alterados por algún operador genético. El uso del elitismo garantiza que la máxima aptitud de la población nunca decrece de una generación a la siguiente, y ello normalmente produce una convergencia más rápida de la población.

Explotación es el proceso de usar información acumulada de previos puntos visitados en el espacio de búsqueda para determinar cuales lugares podrían ser los siguientes en ser visitados (con una cierta probabilidad). "Escalando la colina" es un ejemplo de explotación, porque investiga puntos adyacentes en el espacio de búsqueda, y se mueve en la dirección señalada por el mayor incremento de aptitud. Las técnicas explotativas son buenas para encontrar mínimos y máximos locales.

Exploración es el proceso de visitar regiones completamente nuevas de un espacio de búsqueda, observando si algo prometedor puede ser encontrado allí. Contrario a la explotación, la exploración abarca saltos hacia regiones desconocidas. Las búsquedas aleatorias son un ejemplo de exploración. Los problemas que contienen muchos mínimos (o máximos) locales algunas veces

[1] Aunque esta hipótesis es comunmente aceptada, es importante enfatizar que no ha sido comprobada, como lo demuestran los *Royal Roads* ideados por Melanie Mitchell et al[40].

pueden ser resueltos sólo utilizando técnicas de exploración tales como la búsqueda aleatoria.

Un **genotipo** representa una solución potencial a un problema, y es básicamente la cadena de valores seleccionada por el usuario, también llamado cromosoma.

Un **fenotipo** es el significado de un cromosoma particular, definido externamente por el usuario.

La **cruza** es el operador por medio del cual los individuos intercambian información entre sí. Este operador es el que realiza la tarea explotativa dentro del espacio de búsqueda.

La **mutación** es el operador responsable por los cambios aleatorios en los individuos de la población. Este operador es el que realiza la exploración del espacio de búsqueda.

La **reproducción** [22] es un proceso en el cual las cadenas de los individuos son copiadas teniendo en cuenta sus valores de función objetivo, lo cual significa que los individuos con valor más alto tienen mayor probabilidad de contribuir en uno o más descendientes en la siguiente generación.

1.4 Principales paradigmas de la computación evolutiva

En la actualidad existen 3 paradigmas principales que conforman la Computación Evolutiva:

1. Estrategias Evolutivas
2. Programación Evolutiva
3. Algoritmos Genéticos

Cada uno de ellos se apega en menor o mayor grado a las características del Neo-Darwinismo. A continuación proporcionamos una breve descripción de cada uno de ellos.

1.4.1 Programación evolutiva

Fue propuesta por Lawrence J. Fogel en 1960. En ella, el aprendizaje se ve como un comportamiento adaptativo [19].

Su interés está centrado principalmente en las relaciones entre padres e hijos, más que en el efecto de los operadores genéticos.

La programación evolutiva es una abstracción de la evolución a nivel de las especies, por lo que no se aplica ningún operador de cruza, ya que individuos de diferentes especies no se pueden cruzar entre sí.

Su operador único es pues la mutación, y la selección es probabilística.

La propuesta original es la siguiente: una población de máquinas de estado finito (máquinas de Mealy [2]) se expone a una serie de símbolos de entrada (el ambiente); y se espera que eventualmente, sea capaz de predecir las secuencias futuras de símbolos que recibirá. Fogel utilizó la función de pago que indicaba qué tan bueno era un cierto autómata para predecir un símbolo.

[2]La máquina de Mealy es un transductor determinístico de estado finito que emite un símbolo de salida en cada movimiento. Para cada posible símbolo de entrada, en cada estado, la máquina puede generar un símbolo de salida y el siguiente estado de transición. A diferencia de las máquinas de Moore, cuya salida está determinada únicamente en función del estado presente, la salida de las máquinas de Mealy está en función del estado presente y la entrada. [19]

Para cada máquina padre, conforme se ofrece cada símbolo de entrada a la máquina, se compara cada símbolo de salida con el siguiente símbolo de entrada. El valor (normalmente numérico) de esta predicción se mide posteriormente con respecto a la función de pago. Después de efectuarse la última predicción, una función de pago de cada símbolo indica la aptitud de la máquina. El algoritmo básico de la programación evolutiva es el siguiente:

1. Generar aleatoriamente una población inicial
2. Aplicar mutación (hay diversas variantes de este operador)
3. Se calcula la aptitud de cada hijo y se usa un proceso de selección mediante torneo (normalmente estocástico) para determinar cuales son las soluciones que se retendrán.

1.4.2 Estrategias evolutivas

Las estrategias evolutivas (EE) fueron desarrolladas por Paul Bienert, Ingo Rechenberg y Hans-Paul Schwefel en la Universidad Técnica de Berlín en el año de 1963 [46]. En este paradigma, la búsqueda progresa debido a las mutaciones que se aplican a los individuos más aptos, por lo que el operador principal es la mutación. Las variables de control son usadas para producir nuevos cambios aleatorios en las variables objetivo y después las propias variables de control son mutadas. La versión original usaba un solo padre y con él se generaba un solo hijo. Este hijo se mantenía si era mejor que el padre, o de lo contrario se eliminaba (selección extintiva). En ella un individuo nuevo es generado usando:esto es, la secuencia de símbolos que se han observado hasta el momento actual $x^{t+1} = x^t + N(0, \sigma)$

donde t se refiere a la generación (o iteración) en la que nos encontramos, y $N(0, \sigma)$ es un vector de números Gaussianos independientes con una media de cero y desviación estándar σ.

Rechenberg introdujo a principios de los 1970s el concepto de población, al proponer una estrategia evolutiva llamada (μ+1), en la cual hay μ padres y se genera un solo hijo, el cual puede reemplazar al peor padre de la población.

Schwefel introdujo a mediados de los 1970s el uso de múltiples hijos en la denominadas ($\mu + \lambda$)-EE y (μ,λ)-EE.
La notación se refiere al mecanismo de selección utilizado:

1. En el primer caso, los μ mejores individuos obtenidos de la unión de padres e hijos sobreviven.

2. En el segundo caso, sólo los μ mejores hijos de la siguiente generación sobreviven.

Las estrategias evolutivas trabajan directamente sobre los fenotipos, por lo que no existe una codificación de las variables. La cruza o recombinación es un operador secundario que se aplica en forma diferente a las variables de decisión que a las variables de control. Las diferentes técnicas existentes para "dos padres" (recombinación sexual) (p.ej. intermedia) son esencialmente diferentes clases de promedios ponderados. También es posible aplicar recombinación panmítica, que involucra a varios padres en la creación de un solo hijo.

Las estrategias evolutivas simulan el proceso evolutivo al nivel de los individuos, de ahí que la recombinación sea posible. Asimismo, usan normalmente selección determinística.

1.4.3 Algoritmos Genéticos

Los algoritmos genéticos fueron desarrollados por John Holland y su grupo de trabajo en la Universidad de Michigan a principios de los 1960s [31].

Koza [36] define los algoritmos genéticos de la siguiente manera:

> “El algoritmo genético es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos (típicamente cadenas de caracteres de longitud fija que se ajustan al modelo de las cadenas de cromosomas), cada uno de los cuales se asocia con una aptitud, en una población nueva (es decir, la siguiente generación) usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y sobrevivencia del más apto y tras haberse presentado de forma natural una serie de operaciones genéticas (notablemente la recombinación sexual)”

El algoritmo genético simple es el que a continuación se muestra [4]:

1. Generar una población inicial de manera aleatoria. Los individuos de tal población serán un conjunto de cromosomas o cadenas que representen las posibles soluciones del problema, aunque en general no es posible representar todo el espacio de las soluciones. (Se requiere normalmente una codificación de las variables del problema).

2. Aplicar una función de aptitud a cada uno de esos cromosomas, calculada de acuerdo a la solución codificada por el cromosoma.

3. Conociendo la aptitud de cada individuo, se lleva a cabo un proceso de selección (normalmente probabilística), que determina a los padres de la siguiente generación (presumiblemente los más aptos).

4. Realizar un proceso de cruza, entre los padres que resultaron seleccionados en el paso anterior.

5. Aplicar un operador de mutación a los hijos resultantes que serán ahora la nueva generación.

6. Este procedimiento se cicla desde el punto 2 hasta que se satisfaga algún criterio de terminación predefinido.

Los algoritmos genéticos también modelan el proceso evolutivo al nivel de los individuos (como en las estrategias evolutivas), aunque normalmente requieren de una codificación de las variables del problema (la representación binaria es la adoptada tradicionalmente, aunque muchos otros esquemas son posibles [5]).

En algoritmos genéticos existen varias maneras de llevar a cabo el proceso de selección. Éste suele realizarse de forma probabilística (es decir, aún los individuos menos aptos tienen una cierta oportunidad de sobrevivir) Los investigadores han estudiado varias formas alternativas en el intento de reducir los errores estocásticos asociados a la selección [22].

Las técnicas usadas pueden clasificarse en tres grandes grupos:

1. Selección Proporcional
2. Selección Mediante Torneo
3. Selección de Estado Uniforme

En la **selección proporcional** originalmente propuesta por Holland [30], los individuos se eligen de acuerdo a su contribución de aptitud con respecto al total de la población. Se suelen considerar cuatro grandes grupos dentro de las técnicas de selección proporcional [23]:

1. Ruleta
2. Sobrante Estocástico
3. Universal Estocástico
4. Muestreo Determinístico

Adicionalmente, las técnicas de selección proporcional pueden tener los siguientes aditamentos:

1. Escalamiento Sigma

2. Jerarquías

3. Selección de Boltzmann

La **selección mediante torneo** fue propuesta por Wetzel [51]. La idea básica es seleccionar con base en comparaciones directas de los individuos. Existen dos versiones de la selección que son la determinística y probabilística.

La **selección de estado uniforme** fue propuesta por Whitley [52] y se usa en los algoritmos geneticos no generacionales, en los cuales sólo unos cuantos individuos son reemplazados en cada generación (los menos aptos).

Técnicas de cruza

Existen básicamente tres técnicas de cruza:

- Cruza de un punto
- Cruza de dos puntos
- Cruza uniforme

La **cruza de un punto** fue propuesta por Holland [31], en ella se selecciona un punto de manera aleatoria para cada pareja de padres, a partir del cual se intercambian sus extremos, para dar origen a los nuevos descendientes.

De Jong [34] fue el primero en implementar una **cruza de dos puntos** como una generalización de la cruza de un punto.

La **cruza uniforme** fue propuesta originalmente por Ackley [2], aunque se suele atribuir a Syswerda [50] consiste de una cruza de n puntos, pero en la cual el número de puntos de cruza no se fija previamente.

Aunque los algoritmos genéticos fueron concebidos originalmente para problemas de aprendizaje de máquina en los que se buscaba mejorar una cierta función de recompensa, se han vuelto muy populares para optimización global a pesar de las críticas esgrimidas en su contra [35].

En esta tesis utilizaremos algoritmos genéticos con representación binaria y entera para optimización global (lineal y no lineal) en espacios de búsqueda restringidos.

Capítulo 2

MANEJO DE RESTRICCIONES

2.1 Optimización numérica con restricciones

El objetivo de cualquier técnica de optimización es encontrar el óptimo (o los óptimos) globales de cualquier problema. Un problema de optimización numérica puede definirse como [26]:
Minimizar $f(x)$
Sujeta a:

$$\begin{array}{ll} g_j(x) \leq 0 & j = 1, \ldots, p \\ h_k(x) = 0 & k = 1, \ldots, m \\ x_i^L \leq x_i \leq x_i^U & i = 1, \ldots, n \end{array}$$

donde $f(x)$ es la función que será minimizada (la función objetivo); $g_j(x)$ son las p-restricciones de desigualdad y $h_k(x)$ son las m-restricciones de igualdad para el problema, respectivamente; x_i son las n-variables de decisión con límites superior e inferior denotados como x_i^L y x_i^U respectivamente.

A estas restricciones de igualdad y desigualdad se les llaman "restricciones explícitas". A las restricciones de los rangos en que las variables de decisión toman sus valores se les llaman "restricciones implícitas".

Cada una de las restricciones determina una porción del espacio de búsqueda donde se puede encontrar una solución.

La intersección de todos ellos determina la región factible. Un punto x que pertenece a esta región es una variable de decisión factible, y es una variable no factible si se encuentra fuera de ella. Por lo tanto, todas las soluciones a un problema que satisfagan las restricciones existentes (de cualquier tipo), se consideran dentro de la región factible.

Se dice que una restricción es activa si cumple con la igualdad cuando alcanza el óptimo.

2.2 Técnicas usadas para el manejo de restricciones

Muchos problemas prácticos contienen una o varias restricciones que deben de ser satisfechas. Los algoritmos evolutivos trabajan como técnicas de optimización sin restricciones, por lo que debe de buscarse la manera adecuada de incorporarlas en la función de aptitud.

Refiriéndonos únicamente a la optimización numérica, las técnicas para el manejo de restricciones usando algoritmos genéticos se clasifican en cuatro grandes grupos [38]:

1. Técnicas que penalizan a los individuos no factibles
2. Técnicas que rechazan a los individuos no factibles
3. Técnicas que mantienen una población factible mediante el uso de representaciones y operadores genéticos especiales

4. Técnicas que separan los objetivos y las restricciones

2.2.1 Las técnicas que penalizan a los individuos no factibles

Las **técnicas que penalizan a los individuos no factibles** son las más utilizadas por los investigadores de computación evolutiva. La idea básica es convertir el problema a uno sin restricciones [39, 7], usando para ello una función de penalización exterior. La conversión adoptada es la siguiente:

Minimizar $F = F \pm \overline{P}$

donde F es la función objetivo modificada que adicionalmente contiene el término de penalización $\overline{P}$, y:

$$\overline{P} = c \times [\sum_{j=1}^{p} g_j(x) + \sum_{k=1}^{m} h_k(x)]$$

para todas las restricciones violadas. La constante c es un factor de penalización definido por el usuario.

Estas técnicas pueden penalizar al individuo al menos de 2 maneras [43]: a) simplemente por ser no factible, sin usar ninguna información sobre qué tan cerca se encuentra de la región factible o, b) puede usarse la cantidad de infactibilidad del individuo para determinar su penalización correspondiente.

Las técnicas que penalizan a los individuos no factibles dependen de los parámetros especificados por el usuario, como son el factor de penalización y cualquier otra constante requerida, por lo que su buen desempeño depende claramente del número de constantes que se definen y el valor asociado a éstas.

Estas técnicas suelen requerir un cuidadoso diseño de la función de penalización y de las constantes asociadas a ésta. Lo que es peor, una mala elección de dicha función o sus valores, puede hacer que el algoritmo evolutivo no converja.

2.2.2 Rechazo de individuos no factibles

El **rechazo de individuos no factibles** es la técnica más fácil para manipular restricciones. Consiste en asignar una aptitud de cero a aquellos individuos que violan alguna restricción, para que no tengan ninguna posibilidad de ser seleccionados (también conocida con el nombre de Pena de Muerte). Esta técnica es útil cuando el espacio de búsqueda es convexo y la región factible constituye una parte relativamente grande de éste.

2.2.3 Técnicas que mantienen una población factible mediante el uso de representaciones y operadores genéticos especiales

Existen básicamente dos propuestas en este sentido, una de ellas realizada por Davis [15] que presenta varias representaciones especializadas desarrolladas para problemas de optimización en particular, las cuales preservan la factibilidad de los individuos en todo momento.

La otra propuesta fue realizada por Michalewicz [38] y es llamada GENOCOP (*Genetic algorithm for Numerical Optimization for Constrained Problems*). GENOCOP es capaz de manejar restricciones lineales y elimina las igualdades aplicando operadores genéticos especiales, los cuales le permiten mantener en todo momento sólo individuos factibles. Algunas desventajas de esta técnica son: supone al menos un punto inicial factible y, puesto que únicamente maneja restricciones lineales, sólo es aplicable a espacios de búsqueda convexos.

2.2.4 Técnicas que separan los objetivos y las restricciones

Existen varias propuestas al respecto. Una de ellas fue desarrollada por Paredis [41] basada en un modelo co-evolutivo (en un modelo co-evolutivo existe más de una población, y la aptitud de un individuo en una población depende del estado de evolución de la otra), donde una población de soluciones posibles co-evoluciona con una población de restricciones.

Existe otra técnica llamada "memoria conductista" (behavioural memory) [45] en donde las restricciones deben seguir cierto orden lineal, que afecta el desempeño del algoritmo. Además, se requiere el uso de un esquema para mantener la diversidad en la población.

Una técnica más común consiste en redefinir el problema de optimizar $f(x)$, (una función con un solo objetivo) en uno con objetivos múltiples, en el que tendremos $m + 1$ objetivos; donde m es el número de restricciones. Posteriormente, se aplica cualquier técnica de optimización con objetivos múltiples.

Finalmente, tenemos la propuesta de Surry et al. [49, 48] que usa clasificación de Pareto y VEGA para manejar restricciones. En esta técnica, llamada COMOGA, la población es jerarquizada con base en su grado de violación de las restricciones (contando el número de individuos dominados por cada solución). Después, una porción de la población es seleccionada con base en estas jerarquías y el resto de acuerdo al costo real (aptitud) de los individuos.

Una alternativa para el manejo de restricciones, que no requiere de factores de penalización, es la simulación del sistema inmune biológico propuesta por Hajela y Lee [26, 27, 53], la cual es explicada a detalle en los capítulos posteriores.

Capítulo 3

SISTEMA INMUNE BIOLÓGICO

3.1 Nociones del sistema inmune biológico

El sistema inmune ha sido de gran interés para los investigadores debido a su eficiencia y gran complejidad, siendo aún en nuestro tiempo, un campo abierto para la investigación con muchas preguntas por responder.

El término *"inmune"* es derivado de la palabra griega *inmunitas* (*"in"* privativo, *"munus"* carga, o sea privado de carga, privilegiado) que denominaba en la antigua Roma a los senadores que eran exentos de algunas obligaciones cívicas y procesos legales. Históricamente, el significado ha cambiado. Hoy significa protección ante las enfermedades, de manera especial contra las infecciosas.

El sistema inmune es el resultado de un proceso evolutivo de cerca de 400 millones de años. La base de la teoría inmunológica es su capacidad de poder distinguir entre lo propio y lo no propio, que hacían capaz al animal multicelular primitivo de distinguir entre la propia especie y otras extrañas.

Una definición completa y moderna de inmunidad es dada por A. Abbas

[1] y es la siguiente:

> "Inmunidad es una reacción frente a sustancias extrañas, incluidos los microorganismos, y macromoléculas como las proteínas y los polisacáridos, sin implicar a la consecuencia fisiológica o patológica de tal reacción"

Las células y tejidos que son responsables de la inmunidad son llamados el sistema inmune.

El sistema inmune es el encargado de mantener al organismo sano, protegiéndolo del ataque de elementos extraños que lo pueden enfermar en un ambiente potencialmente hostil.

Los microorganismos infecciosos u otras macromoléculas extrañas poseen antígenos en su superficie, los cuales pueden dañar al organismo.

Los mecanismos de reconocimiento son la fuente de todo el sistema inmune.

El fenómeno de reconocimiento ocurre con frecuencia en la naturaleza, por ejemplo, entre los granos de polen y el estigma de las plantas. La polinización y la germinación sólo se producen si son genéticamente compatibles, evitando así la autofertilización.

Terminología (Definida en [16]):

Antígenos: substancias extrañas que inducen una inmunidad específica.

Linfocitos: células esféricas (7 a 10 **?**m de diámetro) con un gran núcleo, rodeado por una pequeña cantidad de citoplasma. Están asociados con todos los aspectos de la inmunidad específica. Los linfocitos son los principales constituyentes del tejido linfoidico.

Anticuerpos: son las proteínas de reconocimiento, que se encuentran en el suero y otros cuerpos fluidos de los vertebrados, y que reaccionan específicamente ante los antígenos que inducen su formación.

Macrófagos:. Cualquier célula del diverso grupo de aquellas con la capacidad de fagocitar (engullir) (sin incluir los granulocitos), y destruir substancias extrañas. Los macrófagos juegan un importante papel en la inmunidad no específica (removiendo el material extraño) y la inmunidad específica (procesamiento y presentación de antígenos ante células T).

Paratope: es la parte de una célula o molécula de un anticuerpo que hace contacto con un determinante antigénico (epítope)

Epítope: también llamado determinante antigénico, es un pequeño sitio restringido en una molécula invasora que induce la inmunidad, esto es, el sitio en el cual un anticuerpo o célula T puede unirse al antígeno específicamente.

Células T: se forman en la médula ósea para después ir al timo, donde residen brevemente, abandonándolo cuando están maduras, -de allí el nombre de células T, derivados del timo-. Son las causantes de la inmunidad mediada por células de manera directa, y de manera indirecta de la inmunidad humoral.

Células B: son los linfocitos derivados de la médula ósea en los mamíferos y las aves. Las células B actúan como receptores para antígenos específicos. Las células B juegan el papel principal en la inmunidad humoral debido a que, después de la interacción con un antígeno y otras células linfoidicas, proliferan en anticuerpos formando células llamadas plasma y células efectoras de acuerdo a la estimulación antigénica.

Fagocito: Célula capaz de absorber y asimilar las células orgánicas o inorgánicas vecinas.

3.2 Inmunidad natural y adquirida

Los inmunólogos [18] señalan dos tipos de inmunidad: *natural y adquirida.* La inmunidad natural es inespecífica y no posee memoria, mientras que la adquirida es específica y cuenta con un mecanismo de memoria.

La inmunidad natural, también conocida como innata o nativa, es inespecífica. Esto significa que está activa aún sin ser expuesta a los antígenos, por lo que la respuesta inmune no depende del tipo de antígenos al que se enfrente.

Algunos ejemplos de este tipo de inmunidad son las barreras físicas como la piel y las mucosas, las células fagocíticas, ciertas sustancias químicas y enzimas.

Una característica importante es que en este tipo de inmunidad no hay ningún cambio con la exposición repetida a los antígenos; es decir, no existe memoria. Los invertebrados poseen este tipo de inmunidad.

Por convención, la Inmunología, sólo se enfoca a la inmunidad adquirida, aunque ésta requiera de la inmunidad innata para funcionar con eficiencia.

La inmunidad adquirida está compuesta por *linfocitos* y los productos que secretan, que son los *anticuerpos.* Los linfocitos viajan en el torrente sanguíneo circulando a través de todo el organismo, mientras que los anticuerpos se activan a la llegada de algún antígeno.

Existen dos tipos de linfocitos: *tipo B y tipo T.*

Los linfocitos tipo T, son causantes de la inmunidad mediada por células, en la que estos linfocitos tipo T activan a los macrófagos para que combatan a los microorganismos intracelulares o destruyan a las células infectadas.

Los linfocitos tipo B provocan la inmunidad llamada humoral, y son éstos los que secretan anticuerpos que eliminan a los microorganismos extracelulares y las toxinas que secretan, porque los anticuerpos pueden unírsele para cooperar en su destrucción. Ver figura 3.1.

Una de las principales características de la inmunidad adquirida es su especificidad, ya que responde de maneras diferentes ante distintos antígenos. Es decir, un antígeno a partir de su forma, provoca la activación de un grupo distinto de anticuerpos.

Los anticuerpos contienen una región especializada que se llama paratope, la cual es usada para identificar a otras moléculas. Los aminoácidos a partir de los cuales se construye el paratope, determinan su forma y por lo tanto, también la forma de los antígenos a los cuales el paratope puede adherirse para proceder a su eliminación.

El sistema inmune es capaz, a partir de una limitada variedad de anticuerpos, de generar todos aquellos que sean adecuados a casi cualquier tipo de antígeno.

Esta capacidad de reconocimiento del organismo es impresionante, según Smith et al. [47] es capaz de reconocer 10^{16} antígenos distintos, siendo que un mamífero posee entre 10^7 y 10^8 diferentes anticuerpos.

La información genética de un anticuerpo se codifica en segmentos de gen separados, los cuales actúan de manera cooperativa para el reconocimiento del antígeno.

Ante la presencia de un antígeno, se incrementa la concentración de aquellos anticuerpos que le son específicos. A este fenómeno se le llama expansión clonal.

Otra característica importante de la inmunidad adquirida es que posee memoria. En exposiciones repetidas ante un mismo antígeno el sistema pre-

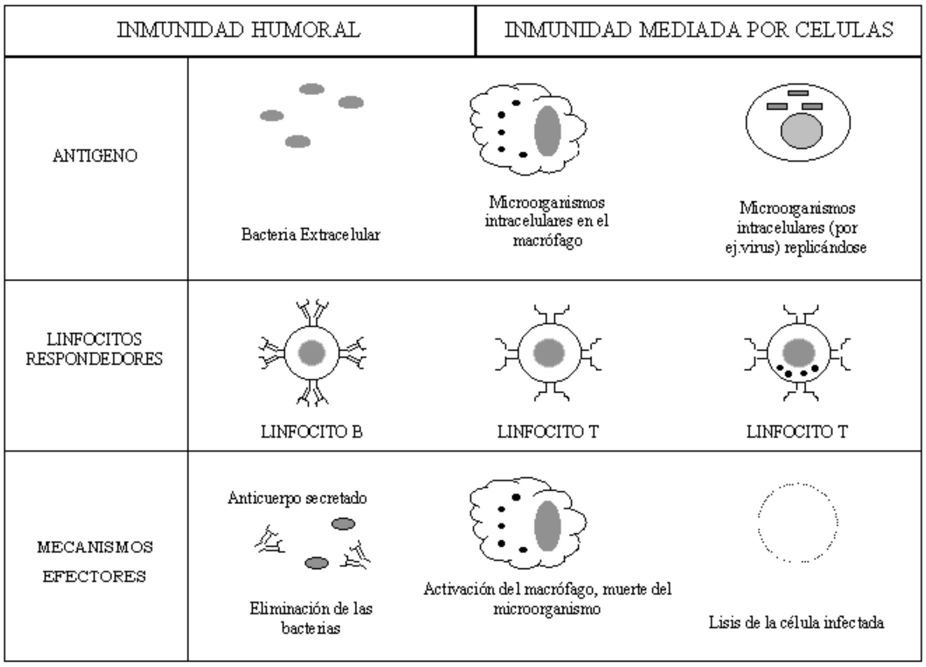

Figura 3.1: Formas de inmunidad específica. En la inmunidad humoral, las células llamadas linfocitos B secretan anticuerpos que eliminan a los microorganismos extracelulares. En la inmunidad mediada por células, los linfocitos T activan a los macrófagos para que maten a los microorganismos intracelulares o destruyan a las células infectadas (p.ej. células infectadas por virus)

senta respuestas más rápidas y eficientes, ya que "recuerda" el tipo de anticuerpos que le fueron presentados con anterioridad.

3.3 Características principales

Abul Abbas [1] resume las características principales del sistema inmune de la siguiente manera:

- **Especificidad:** Antes de la exposición al antígeno existe una gran cantidad de linfocitos B y T, cada uno con diferente especificidad, determinada por la apariencia de su receptor. Las respuestas inmunes son específicas al antígeno que se presenta; los linfocitos B generan anticuerpos dependiendo de la forma del epítope del antígeno. Un antígeno diferente provocará la activación de un grupo distinto de células.

- **Diversidad:** Es increíble la variedad de antígenos que pueden ser reconocidos por el sistema inmune. Distintos clones de linfocitos se diferencian en la estructura de sus receptores para el antígeno, por lo que el repertorio linfocítico es extremadamente grande. El sistema inmune tiene la capacidad para generar al menos 10^7 receptores diferentes.

- **Memoria:** El sistema inmune, cuando es expuesto repetidamente a algún antígeno específico, tiene respuestas diferentes, debido a que "recuerda" cada encuentro con un microorganismo o antígeno extraño, de forma que encuentros posteriores estimularán mecanismos de defensa cada vez más eficaces. El sistema "aprende" los anticuerpos necesarios para tales antígenos. Este fenómeno es debido principalmente a que: Los linfocitos proliferan ante la exposición a cierto antígeno, por lo que los clones de éstos contendrán los receptores adecuados al tipo de antígeno, y por lo tanto, la misma especificidad que el linfocito original. Los linfocitos sobreviven durante largos periodos de tiempo, aún después de la exposición a los antígenos. De esta manera los linfocitos están preparados para responder de manera rápida ante la provocación del antígeno. A esta propiedad de la inmunidad específica se le llama memoria inmunitaria.

- **Regulación o autolimitación:** Todas las respuestas inmunes, decaen después de la eliminación del antígeno. Existen varias razones para ello:

  La más importante de ellas es que las respuestas inmunitarias inducidas por antígenos actúan precisamente para eliminarlos. Esto suprime el estímulo de la activación linfocitaria.

  Después de la estimulación antigénica, los linfocitos continúan con sus funciones durante breves periodos, después de lo cual permanecen en reposo, evolucionan a células de memoria, o se diferencian a células terminales con una vida corta.

- **Discriminación de lo propio y lo ajeno:** Es una de las propiedades más importantes del sistema inmune, y consiste en su capacidad para distinguir entre antígenos extraños y los elementos que le son propios. Esto además evita cualquier atentado que el organismo pudiera tener contra sí mismo. A esta propiedad del sistema se le llama autotolerancia.

3.3.1 Fases de la respuesta inmunitaria

Todas las respuestas inmunitarias específicas inician su proceso con el reconocimiento de algún antígeno invasor que se adhiere a los receptores específicos situados sobre los linfocitos maduros; una vez identificado el epítope del antígeno invasor se estimula la generación de anticuerpos que le son específicos, para finalmente proceder a la eliminación del mismo. De esta forma las respuestas inmunitarias pueden ser divididas en tres fases: 1) la fase de reconocimiento, 2) la fase de activación y 3) la fase efectora.

En el capítulo siguiente veremos la forma en que podemos modelar (aunque de una manera burda) en una computadora estos eficientes mecanismos con que cuenta el sistema inmune biológico.

Capítulo 4

INTRODUCCIÓN AL SISTEMA INMUNE ARTIFICIAL

Los sistemas biológicos, tales como el ser humano, pueden ser vistos como complejos sistemas de procesamiento de información, y han servido como inspiración para generar ideas aplicables a la ciencia y la tecnología [14].

Los sistemas de procesamiento de información motivados biológicamente son clasificados en [14]: sistemas neuronales (redes neuronales), sistemas genéticos (algoritmos evolutivos) y sistema inmune (sistema inmune artificial).

Existe un importante crecimiento en el interés por la investigación del sistema inmune, motivado principalmente por su gran capacidad de procesamiento y alto paralelismo, reconocimiento de patrones, memoria y un aprendizaje implícito.

Para Rowe [44] es una especie de "segundo cerebro" debido a que conserva memoria de experiencias pasadas y además puede construir nuevos patrones a partir de su experiencia para reconocer nuevas estructuras, los antígenos.

Las principales características que podemos observar en el sistema inmune y que son de gran importancia para su simulación y capacidad de resolver problemas son las siguientes [33]: Reconocimiento, extracción de características, diversidad, aprendizaje, memoria, detección distribuida, autoregulación, mecanismo de umbral, co-estimulación, protección dinámica y detección probabilística.

4.1 Simulaciones del sistema inmune

Actualmente se han desarrollado numerosos modelos que emulan el sistema inmune o alguna parte de él, para resolver problemas del mundo real. Entre los modelos computacionales más investigados encontramos los siguientes [13]:

4.1.1 Modelo de red inmune.

El modelo de red inmune fue propuesto por Jerne [33], y se basa en la idea de que el sistema inmune funciona como una red regulada de células con la capacidad de reconocerse unas a otras aún en la ausencia de antígenos. A estas redes se les llama redes idiotípicas. Los linfocitos que componen esta red están interconectados por medio de los anticuerpos.

El reconocimiento de un antígeno no se lleva a cabo únicamente por un conjunto de anticuerpos, sino que a través de un sistema de reconocimiento por niveles, el cual es desatado por la relación anticuerpo-antígeno.

Cuando se presenta un primer conjunto de antígenos, el sistema reacciona generando un conjunto A1 de anticuerpos que le son específicos. Este conjunto de anticuerpos A1 actúan ahora como antígenos y estimularán la producción de un segundo grupo de anticuerpos a los que se les llama anticuerpos idiotipicos A2, y que tiene la propiedad de reconocer idiotopes de

A1. Así de manera sucesiva se creará un tercer grupo de anticuerpos A3 capaz de reconocer anticuerpos de A2.

Jerne discute este modelo como una red formal y como una red funcional.

Desde el punto de vista de red formal se analizan la diversidad, dualismo y supresión. Desde el punto de vista de red funcional se ha trazado un diseño cualitativo.

A partir del modelo de red inmune se concluye que el sistema inmune tiene una sorprendente semejanza con el sistema nervioso desde el punto de vista de red funcional.

4.1.2 El algoritmo de la selección negativa

El algoritmo de la selección negativa propuesto por Forrest [20] para la detección de cambios se basa en el principio de reconocimiento de lo propio y lo que no lo es del sistema inmune.

Trabaja de manera análoga al sistema inmune biológico, generando detectores de manera aleatoria, eliminando a aquellos que sólo son capaces de detectar lo propio, y conservando a los detectores que pueden distinguir a los invasores externos.

El algoritmo funciona de la manera siguiente [20]:

- Define como propio un conjunto de cadenas S de longitud l en un alfabeto finito, que es el conjunto que necesita ser protegido o monitoreado. Por ejemplo, S puede ser un programa, archivo de datos (cualquier software), o un patrón de actividad normal, el cual es segmentado en subcadenas del mismo tamaño.

- Generar un conjunto R de detectores, cada uno de los cuales falla al hacer comparación de coincidencia *(matching)* con los elementos de S. En vez de usar comparación de coincidencia exacta o perfecta, el método usa una regla de comparación parcial, en la cual dos cadenas coinciden si y sólo si son idénticas en al menos r posiciones contiguas, donde r es un parámetro seleccionado expresamente por el usuario.

- Monitorear cambios en S, checando continuamente la coincidencia de los detectores en R contra S. Si algún detector coincide en la comparación, entonces se sabe que un cambio ha ocurrido, porque los detectores son diseñados para no tener correspondencia con alguna de las cadenas originales en S.

Este algoritmo tiene tres principios importantes: 1) cada copia del algoritmo de detección es única, 2) la detección es de manera probabilística y 3) es un sistema robusto (probabilístico) de detección de cualquier actividad invasora, en lugar de una búsqueda para un patrón específico de cambios.

Existen muchas aplicaciones computacionales de estos dos modelos. Entre ellas podemos encontrar por ejemplo [13]:

- La seguridad en computadoras como la detección de virus, monitoreo de procesos UNIX, protección contra usuarios no autorizados.
- Detección de anomalías en series de tiempo, detección de fallas y reconocimiento de patrones, entre otras.

4.2 Manejo de restricciones usando el sistema inmune

Hajela y Lee [26, 27, 53] desarrollaron un algoritmo para manejo de restricciones basado en el sistema inmune, como una alternativa para eliminar el uso de la función de penalización en un problema de optimización. En esta propuesta se usa un algoritmo genético como el motor de búsqueda.

Esta propuesta intenta eliminar la subjetividad que introduce la selección de parámetros en las técnicas que hasta ahora se han venido utilizando de

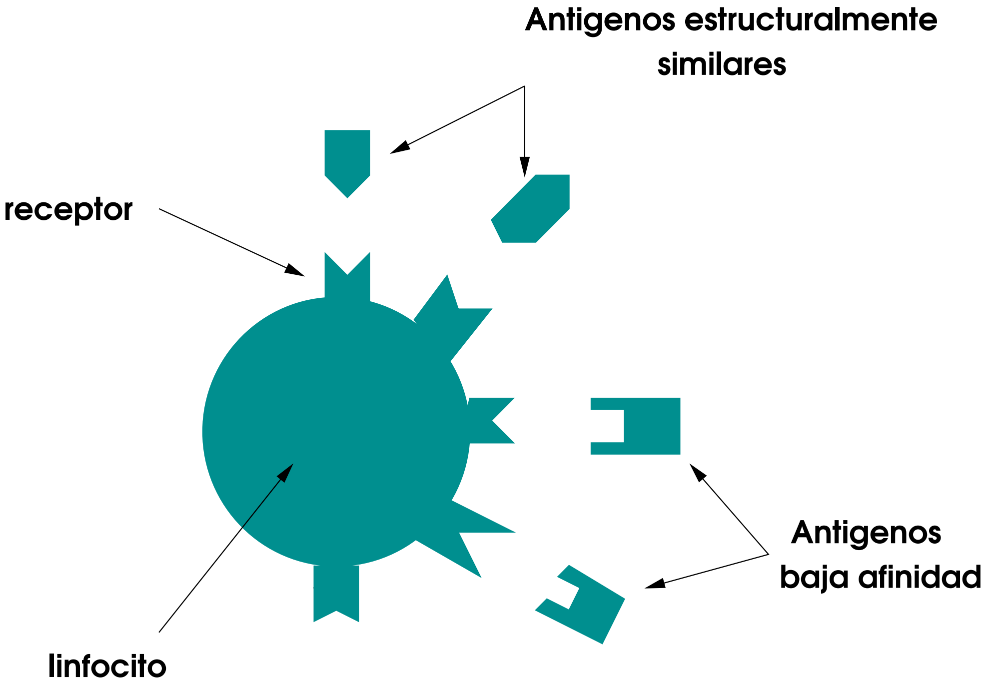

Figura 4.1: Antígenos con baja y alta afinidad

manera convencional, combinando la información tanto de los individuos factibles, como de los no factibles.

La información necesaria para poder reconocer una gran variedad de antígenos posibles se encuentra codificada en segmentos de genes separados de los anticuerpos, cada uno de los cuales forma parte de una biblioteca.

De manera muy simplificada, podemos decir que los anticuerpos de este algoritmo trabajan de manera cooperativa para encontrar la forma adecuada que corresponda a un antígeno determinado. El efecto cooperativo de estos segmentos de gen de los anticuerpos dará como resultado individuos con la forma adecuada para un antígeno específico (ver figura 4.1).

El sistema inmune está orientado hacia la generación de anticuerpos que se adapten a los antígenos.

Este proceso cooperativo requiere alguna organización. Hajela y Lee [26, 27, 53] propusieron el uso de un algoritmo genético como la manera más adecuada de llevar a cabo dicha organización, debido a que un problema de optimización con restricciones es análogo a la evolución dentro de un ecosistema donde la población debe ser co-adaptada a un gran número de requerimientos.

En los algoritmos genéticos con restricciones, los individuos de la población no sólo exhiben su aptitud desde el punto de vista de la función objetivo, sino que además cada individuo debe de ser factible.

Los anticuerpos con capacidad para reconocer una mayor diversidad de antígenos serán los más aptos, mientras que los que reconocen una cantidad menor son individuos menos aptos.

A los anticuerpos que son capaces de distinguir una gran variedad de antígenos se les llama *generalistas*, y a los que pueden reconocer una menor variedad de antígenos se les llama *especialistas*, en analogía al sistema inmune biológico.

El algoritmo supone que existe una población inicial compuesta por una mezcla de individuos factibles y no factibles. Estos individuos son separados en dos grupos: los individuos no factibles, los cuales tienen la función de anticuerpos, y los individuos factibles, que son los antígenos. Los anticuerpos evolucionan hacia el grupo de antígenos.

De manera general el algoritmo es el siguiente:

1. Una población inicial se genera aleatoriamente y se evalúa el valor de la función objetivo y la cantidad de violación de cada miembro de la población.

2. Se divide la población en dos grupos. El primero de ellos contiene a los individuos no factibles que se denominarán anticuerpos, mientras que en el segundo están los individuos factibles, llamados antígenos.

3. La aptitud de todos los anticuerpos es inicializada en cero.

4. La aptitud del conjunto de anticuerpos se computa de acuerdo a su similitud con un conjunto de antígenos predeterminados de la siguiente manera:
    - Un antígeno se selecciona aleatoriamente de la población de antígenos.
    - De la población de N anticuerpos, se selecciona aleatoriamente una muestra de tamaño σ sin reemplazo.
    - Cada anticuerpo de la muestra es comparado contra el antígeno seleccionado, y calculamos el resultado de la comparación, a la que llamaremos Z (cantidad de coincidencia), la cual representa una distancia a nivel del genotipo.
    - Se identifica el anticuerpo de la población con más alta cantidad de similitud Z. Los empates se resuelven de manera aleatoria.
    - La cantidad de similitud del anticuerpo ganador es añadida al valor de su aptitud. La aptitud de los otros anticuerpos permanece sin cambio.
    - Los anticuerpos son retornados a la población, y el proceso es repetido típicamente durante tres veces el número de anticuerpos.
5. Basado en la aptitud calculada en el paso 4, la población de anticuerpos es reproducida en un algoritmo genético convencional (usando cruza y mutación)
6. El proceso es repetido desde el paso 3 hasta la convergencia o hasta que un número máximo de la función de aptitud es alcanzado.
7. Los individuos son retornados al AG externo y se procede de manera convencional.

Este mismo algoritmo se ilustra en la figura 4.2.

La cantidad de similitud Z de un anticuerpo contra el antígeno se calcula bit a bit de la manera siguiente:

$$Z = \sum_{i=1}^{L} t_i$$

donde $t_i = 1$ si hay una coincidencia en la localidad i=1,...,L (donde L es la longitud de ambas cadenas cromosómicas), o cero si no hay coincidencia, como se ilustra en la figura 4.3. Un valor grande para Z indica un alto grado de coincidencia entre las dos cadenas, y por lo tanto, un alto valor de aptitud.

Este es el esquema de un problema de adaptación, ya que los individuos no factibles, que pueden tener altos valores de su función objetivo son forzados a adoptar aquellas características que contribuyen a la no violación de restricciones en los individuos que son factibles.

La premisa implícita de la técnica, bajo ciertas condiciones, es que la reducción de las diferencias genotípicas entre dos individuos traerá como consecuencia una similitud fenotípica, la cual, en nuestro caso, hará que una solución no factible se acerque a la región factible [1].

El tamaño seleccionado para el parámetro σ, determina si los anticuerpos son *generalistas* o *especialistas*.

Si el tamaño de la muestra σ de anticuerpos es lo suficientemente grande, entonces los anticuerpos tienden a ser especialistas, debido a que hay mayor probabilidad de que un anticuerpo corresponda exactamente a algún antígeno.

En cambio, si el tamaño de σ es pequeño, entonces los anticuerpos tienden a ser generalistas. Cuando una muestra de tamaño σ es igual o ligeramente mayor que el número de antígenos, entonces tanto individuos generalistas como especialistas existen en la población.

[1]Esto puede ser debatible, pero existe cierto respaldo teórico a esta hipótesis [21, 47], y el trabajo de Hajela y sus colegas [26, 27, 53], así como esta tesis, son parte del respaldo empírico de la misma

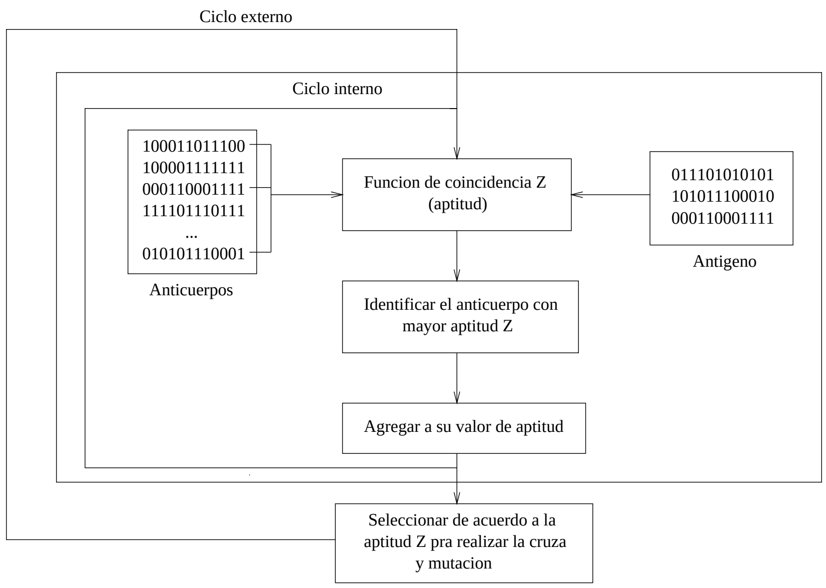

Figura 4.2: Algoritmo de la simulación del sistema inmune

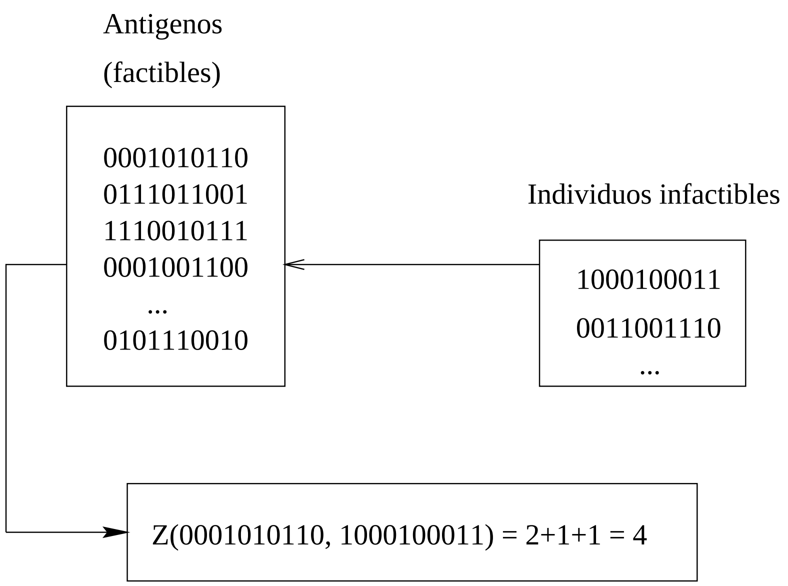

Figura 4.3: Medida de aptitud Z

Estos anticuerpos generalistas contienen fragmentos de información con características de buenas soluciones.

Yoo y Hajela [27, 53] aseguran que el algoritmo propuesto anteriormente funciona mejor si se generan individuos generalistas.

Capítulo 5

ALGORITMO PROPUESTO

En la simulación del sistema inmune para el manejo de restricciones, propuesto por Hajela y Lee [26, 25, 27] fueron detectadas algunas limitantes, las cuales motivaron el desarrollo de este trabajo.

En este capítulo se encuentran descritos tales problemas, las propuestas que pretenden solucionarlos, así como los resultados de su experimentación.

5.1 Motivación

Los problemas detectados en el algoritmo de Hajela y Lee son los siguientes:

1. Supone, de manera implícita, al menos un punto inicial factible,
2. Únicamente está propuesto para representación binaria; y
3. No especifica cómo afecta a la convergencia del algoritmo la selección del parámetro σ, el cual indica la cantidad de anticuerpos que son expuestos ante un antígeno.

Veamos ahora cada uno de estos puntos en mayor detalle:

1. **El algoritmo supone al menos un punto inicial factible**

   Lograr una población inicial que esté compuesta por la mezcla de, tanto individuos factibles como no factibles, es fácil de conseguir cuando la región factible del problema constituye una parte relativamente grande del espacio de búsqueda. Esta condición cambia de manera importante si el espacio de búsqueda está altamente restringido, con zonas factibles pequeñas, donde los individuos factibles son difíciles de encontrar.

2. **Únicamente está propuesto para representación binaria.**

   El principal problema de emplear representación binaria en un algoritmo genético es cuando se desarrollan aplicaciones del mundo real; en ellas la dimensionalidad suele ser alta, pues se tiene un gran número de variables que además requieren muy buena precisión.

3. **Variación en el parámetro sigma**

   Como se expuso en el capítulo 4 y de acuerdo a lo sugerido por Yoo y Hajela [53], el sistema inmune busca la creación de anticuerpos generalistas para mejorar las características de su convergencia (un anticuerpo generalista es aquél capaz de reconocer una mayor variedad de antígenos; uno especialista sólo puede reconocer un tipo de antígeno). La creación de anticuerpos generalistas o especialistas está determinada por el tamaño de σ.

   Por ende, es importante conocer qué tanto afecta al desempeño del programa, la variación de tal parámetro.

5.2 Propuestas

5.2.1 El algoritmo supone al menos un punto inicial factible

Cuando no existe un punto inicial factible (todos los individuos son no factibles); antígeno y anticuerpos se determinan de la siguiente manera:

- El antígeno es el mejor individuo de la población
- Los anticuerpos son una porción de la población seleccionada de manera aleatoria.

Dado que todos los individuos de la población son no factibles, para determinar al mejor de ellos, se busca a aquél que viola en menor cantidad las restricciones. En caso de que exista empate, se selecciona al que viola una cantidad menor de restricciones, y si aún esto no lo resuelve, entonces se decide cuál es el mejor de manera aleatoria.

Por otro lado, para la asignación de los anticuerpos se considera sólo a una parte de la población con el fin de evitar que todos los individuos vayan hacia un mismo punto, pudiendo ocasionar con ello convergencia prematura.

A cada cierto intervalo de generaciones se verifica si algún individuo ha alcanzado la región factible, en cuyo caso, este individuo reemplaza al que estaba haciendo la función de antígeno, alcanzando así la condición deseada para el sistema inmune (población compuesta por la mezcla de individuos factibles y no factibles).

El sistema inmune continúa entonces de manera convencional.

5.2.2 Únicamente está propuesto para representación binaria.

Representación real

Se propone usar una representación de números reales mediante el uso de enteros, esto es, usando un número entero para representar cada dígito, como se muestra en la siguiente figura:

4	2	9	2	1	6

donde se supone una posición fija del punto decimal para cada variable. La cadena completa es decodificada como un solo número real, dividiendo y multiplicando cada valor de acuerdo a su posición. La cadena en la figura representa, por ejemplo el 429.216

La precisión está limitada por la longitud de la cadena. Esta representación pretende ser un compromiso entre un algoritmo genético con codificación real y una representación binaria de números reales, manteniendo lo mejor de ambos esquemas al incrementar la cardinalidad del alfabeto utilizado, pero manteniendo el uso de los operadores genéticos casi sin cambio [5]. Se ha mostrado anteriormente que esta representación ha mejorado el desempeño de un algoritmo genético en diversos problemas de optimización numérica [11, 6].

Cálculo de la aptitud de los anticuerpos para esta representación.

La cantidad de similitud, que hemos denominado Z, de un anticuerpo contra el antígeno, representa una distancia a nivel del genotipo. En la propuesta original de Smith et al. [47] respecto al uso de simulaciones del sistema inmune basadas en algoritmos genéticos (adoptada por Hajela y sus colegas [25, 27], así como en este trabajo de tesis), se planteó el uso de un modelo que trabajara con cadenas binarias, principalmente porque eso facilitaría el chequeo de patrones entre cadenas. Cabe mencionar, sin embargo, que estrictamente hablando, estas cadenas binarias no se interpretan realmente como el "genotipo" de un algoritmo genético, aunque para fines prácticos así lo consideramos en este trabajo de tesis. Las cadenas binarias realmente se utilizan para simplificar el modelo del sistema inmune y se interpretan como la representación tanto de los genes que codifican tanto a los antígenos como a los receptores de las células B y T. Desde el punto de vista teórico no hay ningún impedimento para operar con alfabetos de mayor cardinalidad (usar por ejemplo, el "fenotipo" de cada individuo como punto de partida para la emulación del sistema inmune). Sin embargo, en nuestro caso, se optó por usar el genotipo como el punto de partida, a fin de respetar la propuesta original del sistema inmune antes mencionado.

Para el caso de la representación real, la distancia Z la calculamos de la siguiente manera:

Se obtiene la distancia entre cada alelo del anticuerpo contra cada uno del antígeno, calculando el complemento a 9 de la diferencia aritmética entre ambos alelos:

$$x = 9 - \mid Ab - Ag \mid$$

donde Ab es el alelo del anticuerpo, y Ag es el alelo del antígeno en la misma posición.

Usando estos valores se calcula la distancia Z entre ambas cadenas cromosómicas, computando la distancia euclidiana entre ellas, de la manera siguiente:

$$Z = \sqrt{x_1^2 + \ldots + x_i^2}$$

donde i es la longitud de la cadena cromosómica, y x es la distancia entre cada pareja de alelos anticuerpo-antígeno (que se ha calculado anteriormente)
Ejemplo:
Antígeno:

3	2	8

Anticuerpo:

6	8	1

Calculamos la distancia entre cada pareja de genes:
para la primera pareja de alelos, 3 y 6:

$$x_1 = 9- \mid 3 - 6 \mid = 6$$

para la pareja 2 y 8:

$$x_2 = 9- \mid 2 - 8 \mid = 3$$

el último par 8 y 1:

$$x_3 = 9- \mid 8 - 1 \mid = 2$$

finalmente calculamos la distancia entre las dos cadenas:

$$Z = \sqrt{x_1^2 + x_2^2 + x_3^2} = \sqrt{6^2 + 3^2 + 2^2} = 7$$

de manera que el valor de aptitud (Z) para el anticuerpo 68.1, respecto del antígeno 32.8 es igual a 7.

Representación binaria con código de Gray

El uso de la representación binaria no mapea adecuadamente el espacio de búsqueda con el espacio de representación [5]. Por ejemplo, los enteros 3 y 4, que son contiguos en el espacio de búsqueda, son en representación binaria 011 y 100, respectivamente, difiriendo en 3 bits en el espacio de representación. A este fenómeno se le llama risco de Hamming [5].

Existe una variante de la representación binaria que es el código de Gray, frecuentemente usada por los investigadores de algoritmos genéticos, la cual preserva la propiedad de adyacencia existente entre el espacio de búsqueda y el espacio de representación.

El valor de la aptitud (Z) de un anticuerpo dentro del sistema inmune, se calcula por una verificación bit a bit de las cadenas cromosómicas del antígeno y anticuerpo, y se determina la similitud o distancia entre ellas, por lo cual se considera importante que no existan distancias genotípicas mayores a 1 en la representación.

Para codificar los valores de las variables en cadenas binarias se discretiza el rango de cada variable fijando una precisión deseada y posteriormente se tratan esos valores como si fueran enteros. Se calcula el tamaño que debe tener la cadena aplicando la siguiente fórmula:

Tamaño $= (int)log_2(LimSup - LimInf) * 10^{precision} + 0.9$

La precisión aplicada en cada problema de prueba se describe en la sección de resultados de las funciones.

5.2.3 Variación en el parámetro σ

Para verificar qué tanto afecta al desempeño del algoritmo la selección del parámetro σ, que determina la cantidad de anticuerpos que serán sometidos ante un antígeno, se realizaron varias ejecuciones del programa, comparando los resultados arrojados. De esta manera se pretende conocer la influencia de tal parámetro en la simulación del sistema inmune. Los resultados obtenidos se muestran en el apartado 5.4 de resultados.

5.2.4 Otros cambios

Con el afán de lograr un mejor desempeño del programa se realizaron algunos otros cambios al algoritmo genético externo, en el cual evoluciona el sistema inmune. Estos cambios son en la asignación del valor de aptitud a los individuos y en la adición de un mecanismo elitista.

Asignación de aptitud a los individuos.

En un problema con restricciones, la aptitud de los individuos no sólo está determinada por su valor de la función objetivo, sino que además debe de ser factible, por lo que, cuando en la población se cuenta con individuos no factibles, su valor de aptitud se determina directamente con base en la cantidad de violación de las restricciones, para entonces buscar minimizar tal valor. De manera que cuando un individuo viola en gran cantidad las restricciones, tiene una aptitud menor que aquellos que lo hacen en menor cantidad o que son factibles. Este criterio es aplicado a la técnica de selección proporcional.

Por otro lado, específicamente para la selección mediante torneo binario, el ganador se determina de acuerdo a lo siguiente:

- Si ambos individuos son factibles, gana el que tiene mayor valor de la función objetivo (suponemos problemas de maximización).
- Cuando uno de los individuos es factible, éste gana sobre el que no lo es.
- Si ambos son no factibles:
  - Gana aquél cuya suma de violación a las restricciones es menor.
  - Si violan en igual proporción en el punto anterior, entonces se prefiere al individuo que viole menos restricciones.
  - Si existe empate en el punto anterior, se selecciona al individuo con mejor aptitud (con base en la función objetivo)
  - Si los individuos tienen igual valor en la evaluación de la función objetivo, entonces el ganador se determina aleatoriamente.

Es importante mencionar que la aptitud basada en la cantidad de violación como se ha descrito, sólo se aplica cuando existen individuos no factibles en la población. En caso contrario se utiliza el valor de la función objetivo de manera convencional.

Individuo elitista

El elitismo retiene al mejor individuo de la generación actual, el cual pasa a la siguiente sin ser afectado por ningún operador. El criterio para determinar a tal individuo es el siguiente:

- Si existen individuos factibles en la población, entonces se selecciona aquel cuyo valor de la función objetivo sea más alto.
- Si no existe ningún individuo factible, se escoge al que tiene menor cantidad de violación de las restricciones.
- En caso de existir empate en el punto anterior, se prefiere al que viola un menor número de restricciones.
- Si aún existiera empate, seleccionar al individuo elitista de manera aleatoria.

5.3 Funciones de prueba

Para comprobar la eficiencia del algoritmo, se usaron algunas de las funciones propuestas por los investigadores en manejo de restricciones [8, 39]. Éstas se refieren a problemas de optimización global que previamente han sido resueltas por algoritmos evolutivos y otras técnicas de programación matemática. Se han seleccionado 7 funciones para realizar los experimentos. Buscando incluir a aquellas que tienen restricciones de desigualdad, espacios de búsqueda de alta y baja dimensionalidad, espacios de búsqueda convexos y no convexos, zonas factibles disjuntas y/o muy pequeñas [1]. Los problemas son descritos a continuación:

5.3.1 Problema 1: Diseño de un recipiente de presión [8]

Un recipiente cilíndrico está tapado por ambos lados con cabezas hemisféricas como se muestra en la figura 5.1. El objetivo es minimizar el costo total, incluyendo el costo del material, forma y soldadura. Existen cuatro variables: T_s (ancho de la estructura), T_h (ancho de la cabeza), R (radio interno) y L (longitud de la sección cilíndrica del recipiente, sin incluir la cabeza). T_s y T_h son enteros múltiplos de 0.0625 pulgadas, los cuales son las anchos disponibles de los platos de acero, y R y L son continuas. El problema se define como sigue:
Minimizar:

$$F(x) = x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3 \tag{5.1}$$

sujeto a:

$$g_1(x) = -x_1 + 0.0193 x_3 \leq 0 \tag{5.2}$$

$$g_2(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \leq 0 \tag{5.3}$$

$$g_4(x) = x_4 + 240 \leq 0 \tag{5.4}$$

5.3.2 Problema 2. Diseño de una viga soldada [8]

Una viga soldada es diseñada para minimizar el costo sujeto a restricciones de esfuerzo cortante (τ), esfuerzo de pandeo (σ), carga en la barra (Pc), deflexión

[1] Aunque no se consideran funciones de prueba con restriccciones de igualdad, es posible aplicar el método propuesto en esta tesis a dichos problemas si estas restricciones se transforman a desigualdades como se indica en la literatura especializada [39, 7]

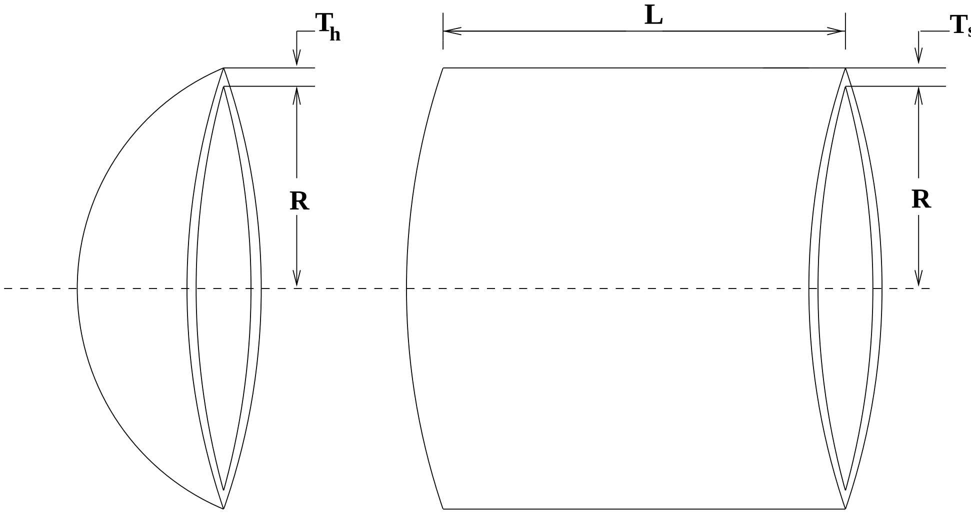

Figura 5.1: Vaso de presión usado para el problema 1

en la viga (δ) y restricciones de límite. Hay cuatro variables mostradas en la figura 5.2 : $h(x_1), l(x_2), t(x_3)$ y $b(x_4)$. El problema se describe como sigue:

Minimizar:

$$F(x) = 1.1074x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \tag{5.5}$$

$$g_1(x) = \tau(x) - \tau_{max} \leq 0 \tag{5.6}$$

$$g_2(x) = \sigma(x) - \sigma_{max} \leq 0 \tag{5.7}$$

$$g_3(x) = x_1 - x_4 \tag{5.8}$$

$$g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2 - 5.0) \leq 0 \tag{5.9}$$

$$g_5(x) = 0.125 - x_1 \leq 0 \tag{5.10}$$

$$g_6(x) = \delta(x) - \delta_{max} \leq 0 \tag{5.11}$$

$$g_7(x) = P - Pc(x) \leq 0 \tag{5.12}$$

donde:

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x^2}{2R} + (\tau'')^2} \tag{5.13}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2},\ \tau'' = \frac{MR}{J},\ M = P\left(L + \frac{x_2}{2}\right) \tag{5.14}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{5.15}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \tag{5.16}$$

$$\sigma(x) = \frac{6PL}{x_4x_3^2},\ \delta(x) = \frac{4PL^3}{Ex_3^3x_4} \tag{5.17}$$

$$Pc(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \tag{5.18}$$

$$P = 6000\,lb,\ L = 14\,in,\ E = 30 \times 10^6\,psi,\ G = 12 \times 10^6\,psi \tag{5.19}$$

$$\tau_{max} = 13,600\,psi,\ \sigma_{max} = 30,000\,psi,\ \delta_{max} = 0.25\,in \tag{5.20}$$

5.3.3 Problema 3 [39]

Minimizar:

$$f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{5.21}$$

sujeto a:

$$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \tag{5.22}$$

$$g_2(x) = 2x_1 + 2x_3 + x_{11} - x_{12} - 10 \leq 0 \tag{5.23}$$

$$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \tag{5.24}$$

$$g_4(x) = -8x_1 + x_{10} \leq 0 \tag{5.25}$$

$$g_5(x) = -8x_2 + x_{11} \leq 0 \tag{5.26}$$

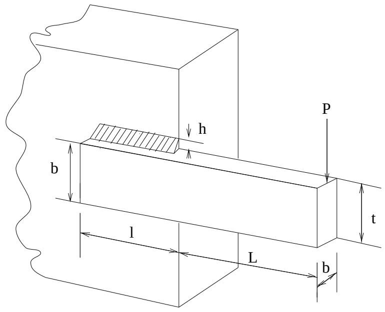

Figura 5.2: Viga soldada para el problema 2

$$g_6(x) = -8x_3 + x_{12} \leq 0 \tag{5.27}$$

$$g_7(x) = 2x_4 - x_5 + x_{10} \leq 0 \tag{5.28}$$

$$g_8(x) = 2x_6 - x_7 + x_{11} \leq 0 \tag{5.29}$$

$$g_9(x) = 2x_8 - x_9 + x_{12} \leq 0 \tag{5.30}$$

donde los límites son $0 \leq x_i \leq 1,\ 1 \leq i \leq 9,\ 0 \leq x_i \leq 100,\ 10 \leq x_i \leq 12$ y $0 \leq x_{13} \leq 1$ el mínimo global está en $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$. Seis restricciones están activas en el óptimo $(g1, g2, g3, g4, g4, g5, g6)$. Se dice que una restricción es activa si cumple con la igualdad cuando alcanza el óptimo.

5.3.4 Problema 4 [39]

Maximizar:

$$f(x) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \tag{5.31}$$

sujeto a:

$$g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0 \tag{5.32}$$

$$g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0 \tag{5.33}$$

donde $n = 20$ y $0 \leq x_i \leq 10$ para $1 \leq i \leq n$. El máximo global es desconocido, la mejor solución reportada en la literatura es $f(x) = 0.803619$.

5.3.5 Problema 5 [39]

Minimizar:

$$f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \tag{5.34}$$

sujeto a:

$$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0 \tag{5.35}$$

$$g_2(x) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053 \leq 0 \tag{5.36}$$

$$g_3(x) = 80.51249 - 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0 \tag{5.37}$$

$$g_4(x) = -80.51249 - 0.0071317x_2x_5 + 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0 \tag{5.38}$$

$$g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0 \tag{5.39}$$

$$g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0 \tag{5.40}$$

donde $78 \leq x_1 \leq 102,\ 33 \leq x_2 \leq 45$ y $27 \leq x_i \leq 45$ para $3 \leq i \leq 5$.

El óptimo global está en $x^* = (78, 33, 29.995256, 45, 36.775812)$ donde $f(x^*) = -30665.539$. Dos restricciones están activas: g_1 y g_6

5.3.6 Problema 6 [39]

Minimizar:

$$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \tag{5.41}$$

sujeto a:

$$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \tag{5.42}$$

$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0 \tag{5.43}$$

donde $13 \leq x_1 \leq 100$ y $0 \leq x_2 \leq 100$. La solución óptima es $x^\star = (14.095, 0.84296)$, donde $f(x^\star) = -6961.81388$. Ambas restricciones están activas en el óptimo.

5.3.7 Problema 7 [39]

Maximizar:

$$f(x) = (100 - (x_1 - 5)^2 - (x_2 - 5)^2)/100 \tag{5.44}$$

sujeto a :

$$g_1(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0 \tag{5.45}$$

donde $0 \leq x_i \leq 10$ para $1 \leq i \leq 3$ y $p, q, r = 1, 2, \ldots, 9$. La región factible del espacio de búsqueda consiste de 9^3 esferas disjuntas. Un punto (x_1, x_2, x_3) es factible, si y sólo si existen p, q y r tales que la desigualdad de arriba se cumple. El óptimo global está localizado en $x^* = (5, 5, 5)$ donde $f(x^*) = 1$.

Cálculo de ρ para los problemas de prueba.

Koziel y Michalewicz [37] sugieren una medida a la que llaman ρ para determinar qué tan difícil resulta encontrar una región factible para los problemas de prueba. Esta medida $\rho = \mid F \mid / \mid S \mid$ fue determinada experimentalmente generando 1,000,000 de puntos de manera aleatoria en el espacio de búsqueda S y determinando si pertenecen o no a la región factible F.

La tabla 5.1 muestra los valores de ρ obtenidos para cada uno de los problemas descritos, así como el número de variables, y tipo de restricciones.

Función	**n**	**Tipo de f**	ρ	DL	DN
1	4	cuadrática	40.9429%	2	1
2	4	cuadrática	0.6116%	6	1
3	13	cuadrática	0.0111%	9	0
4	20	no lineal	99.8474%	0	2
5	5	cuadrática	52.1230%	0	6
6	2	cúbica	0.0066%	0	2
7	3	cuadrática	0.0018%	0	2

Tabla 5.1 Resumen de las funciones seleccionadas para los experimentos.

Donde n es el número de variables, DL son las desigualdades lineales, y DN son las desigualdades no lineales.

5.4 Resultados

Se realizó la implementación del algoritmo propuesto con la finalidad de verificar su eficiencia.

El programa se realizó en lenguaje C, usando el compilador **egcs** versión 1.1.2 en una estación de trabajo Sun Ultra 5 con sistema operativo Linux

(Kernel 2.2.14).

Técnicas de selección disponibles en la implementación

El programa cuenta con las siguientes opciones para realizar el mecanismo de selección:

1. Ruleta
2. Sobrante estocástico con reemplazo
3. Sobrante estocástico sin reemplazo
4. Universal estocástico
5. Escalamiento sigma
6. Torneo binario

Asimismo, para aplicar la cruza se tienen las opciones:

1. Un punto
2. Dos puntos
3. Uniforme

Parámetros del algoritmo genético aplicados

Específicamente para la realización de los experimentos que se reportan se aplicaron los siguientes parámetros (para los casos donde hubo necesidad de modificar alguno de ellos, se especifica en la descripción de los resultados):
Número de generaciones: 5000
Tamaño de población: 90
Porcentaje de cruza: 0.75
Porcentaje de mutación: 0.05
Tipo de selección: Sobrante estocástico con reemplazo

Tipo de cruza: Cruza de un punto

El parámetro σ determina el tamaño de la muestra de anticuerpos, que son sometidos ante los antígenos. Se experimentó utilizando 3 variantes de él con la finalidad de determinar qué tanto afecta a la eficiencia del algortimo tal parámetro. Las tres buscan crear individuos generalistas en la medida de lo posible, los valores se determinan de la siguiente manera:

1. Si la cantidad de antígenos es menor a 5, entonces sigma es igual a 4; en caso contrario, sigma es igual a la cantidad de antígenos menos 4
2. El valor de sigma es igual a la cantidad de anticuerpos dividido entre 3
3. Si no existen individuos factibles en la población (antígenos) entonces sigma es igual a la cantidad de anticuerpos; en caso contrario, sigma es igual a la cantidad de antígenos presentes en la población.

5.4.1 Problema 1

El mejor valor reportado en la literatura es [10]: **f(x)=6288.7445**, para los valores de $x = (0.8125, 0.4375, 40.3239, 200.0000)$ con las restricciones: (-0.034324, -0.052847, -27.105847, -40.00000)
La precisión aplicada es de 4 digitos.

Representación binaria

El mejor valor encontrado por el sistema inmune para esta representación es: **f(x)=6062.253733**, para los valores de x=(0.812500, 0.437500, 42.077800, 176.893600) con las restricciones: (-0.000398, -0.036078, -5.604816, -63.106400)

Ejecución	Sigma1	Sigma2	Sigma3
1	6112.4047	6841.0246	6409.2433
2	6095.1982	6426.5653	6449.1678
3	6136.4491	6199.8326	6161.295
4	6835.809	6455.812	6456.9217
5	6435.0303	6062.5371	6063.8637
6	7333.8196	6109.614	6131.1642
7	6103.8023	6484.7437	7354.2236
8	6411.9759	6453.5133	7336.5341
9	6402.3659	6837.0855	6133.525
10	6142.1521	6103.9382	6442.0066
11	6405.5149	6385.0809	6515.245
12	6101.4332	6467.5316	6778.2688
13	6377.3045	6455.7341	6112.2067
14	6091.0855	6835.6292	6432.6934
15	6453.5558	6413.2073	6097.8869
16	6371.4857	6383.6893	6415.3658
17	6095.3845	6068.6782	6101.815
18	6092.8574	7367.044	6155.2296
19	6091.5044	6062.2537	7457.9
20	6114.3169	6423.9501	6418.085
Mediana	6139.3006	6425.2577	6416.7254
Promedio	6310.1725	6441.87324	6471.13206
Desviación Estándar	317.348035	323.200973	447.415152
Mejor	6091.0855	6062.2537	6063.8637
Peor	7333.8196	7367.044	7457.9

Tabla 5.2 Variación de σ del problema 1 con representación binaria.

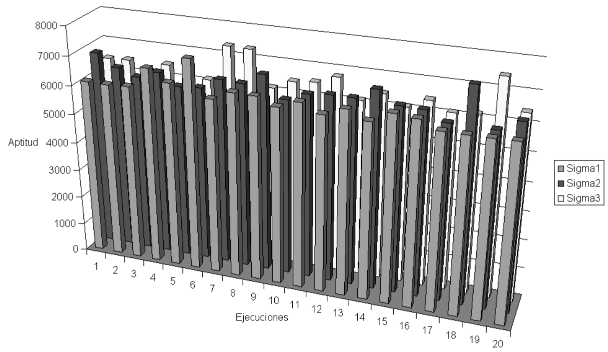

Figura 5.3: Variación de sigma para 20 ejecuciones del problema 1 usando representación binaria.

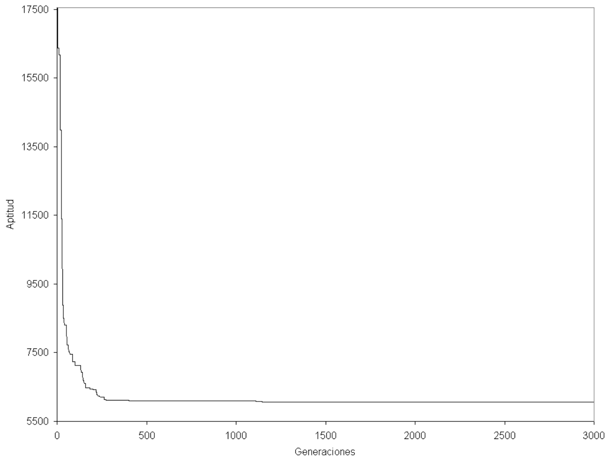

Figura 5.4: Aptitud contra generación del problema 1 para la mejor ejecución encontrada con Sigma2 usando representación binaria.

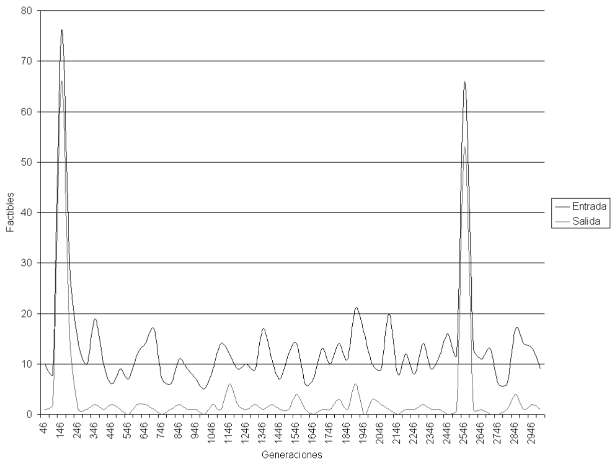

Figura 5.5: Individuos factibles que entran y salen del sistema inmune por generación para la mejor ejecución del problema 1 usando representación binaria.

Representación real

El mejor valor encontrado por el sistema inmune es: **f(x)= 6772.129318** Para los valores de x=(1.062500, 0.562500, 55.045000, 62.757400), con las restricciones: (-0.000131, -0.037371, -1.833963, -177.242600).

Ejecución	Sigma1	Sigma2	Sigma3
1	6795.5219	6782.7664	6799.9188
2	6788.6366	6793.6969	6793.7842
3	6776.6631	6870.4929	6801.2185
4	6779.006	6889.5039	6778.8318
5	6801.1351	6888.0531	6801.337
6	6772.1293	6798.7767	6777.7586
7	6857.0551	6899.2008	6906.4165
8	6788.9027	6781.7544	6804.2225
9	6802.5894	6812.7098	6776.3386
10	6801.3527	6776.2913	6775.1362
11	6787.7489	6817.0065	6804.2245
12	6775.8906	6782.8304	6807.4106
13	6789.3767	6783.348	6791.0956
14	6792.361	6801.4441	6868.1607
15	6779.006	6861.7734	6801.8181
16	6798.6011	6787.405	6879.042
17	6817.898	6803.2928	6801.3527
18	6841.6473	6798.9654	6866.3607
19	6785.651	6865.7997	6791.378
20	6772.9621	6963.6011	6775.1059
Mediana	6788.9027	6803.2928	6801.337
Promedio	6795.19014	6830.31296	6810.57856
Desviación Estándar	22.5794939	52.7879886	39.1062242
Mejor	6772.1293	6776.2913	6775.1059
Peor	6857.0551	6963.6011	6906.4165

Tabla 5.3 Variación de σ del problema 1 usando representación real.

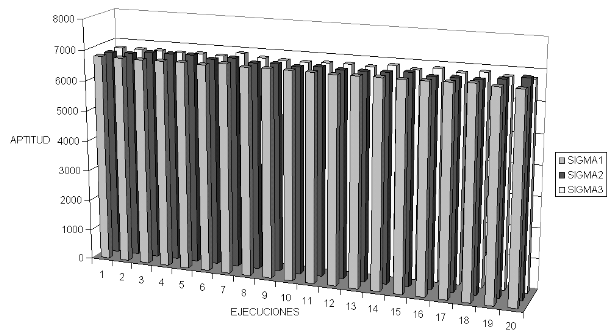

Figura 5.6: Variación de σ para 20 ejecuciones del problema 1 usando representación real.

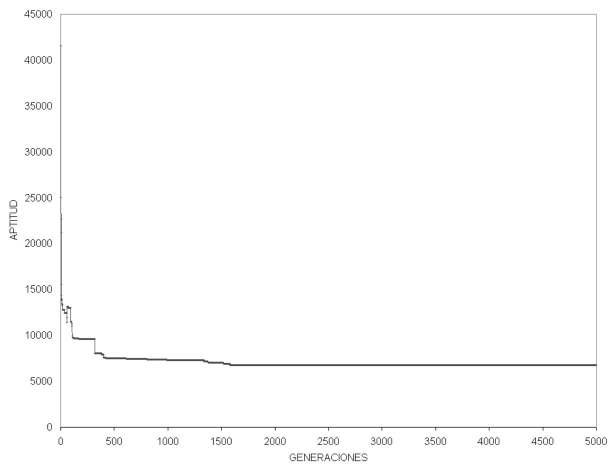

Figura 5.7: Aptitud contra generación de la mejor ejecución del problema 1 con Sigma1 usando representación real

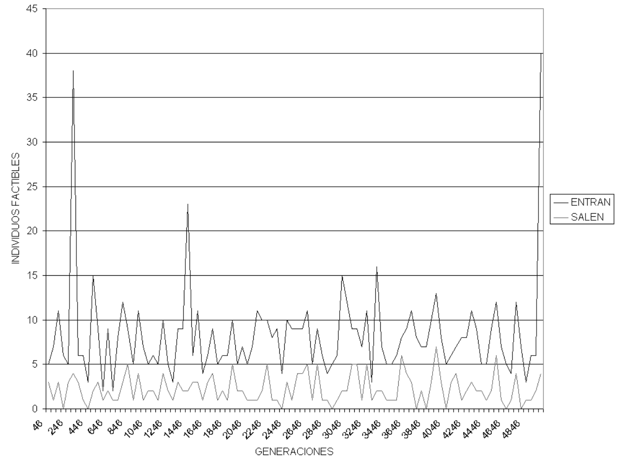

Figura 5.8: Individuos factibles que entran y salen del sistema inmune por generación para la mejor ejecución del problema 1 usando representación real.

5.4.2 Problema 2

El mejor valor reportado por la literatura es [9]: **f(x)=1.74830941**, para los valores de x=(0.2088, 3.4205, 8.9975, 0.2100) Con las restricciones: (-0.337812, -353.902604, -0.00120, -3.411865, -0.08380, -0.235649, 363.232384) La precisión aplicada es de 4 dígitos.

Representación binaria

El mejor valor encontrado por el sistema inmune es: **f(x)=1.725255**, para los valores de $x = (0.205700, 3.471800, 9.035100, 0.205800)$ con las restricciones: (-0.270305, -0.139490, -0.000100, -3.432597, -0.080700, -0.235538, -5.492326)

Ejecución	Sigma1	Sigma2	Sigma3
1	1.7327	1.7455	1.7255
2	1.7368	1.7306	1.7416
3	1.7377	1.7343	1.7257
4	1.7734	1.7346	1.7878
5	1.7712	1.7359	1.7321
6	1.7329	1.7372	1.7291
7	1.7355	1.7267	1.807
8	1.7293	1.7303	1.8007
9	1.7947	1.7312	1.7356
10	1.7349	2.05	1.7342
11	1.7335	1.7445	1.8632
12	2.0703	1.8134	2.1515
13	1.7264	1.8169	1.8951
14	1.8112	2.0875	1.8046
15	1.7257	1.7253	1.7645
16	1.7284	1.7449	1.831
17	1.734	1.7368	1.7323
18	1.734	1.8923	1.8023
19	1.7753	1.731	1.8023
20	1.9014	1.73	1.7327
Mediana	1.7352	1.73635	1.77615
Promedio	1.770965	1.783945	1.79494
Desviación estándar	0.08217001	0.10606214	0.09739714
Mejor	1.7257	1.7253	1.7255
Peor	2.0703	2.0875	2.1515

Tabla 5.4 Variación de σ del problema 2 con representación binaria.

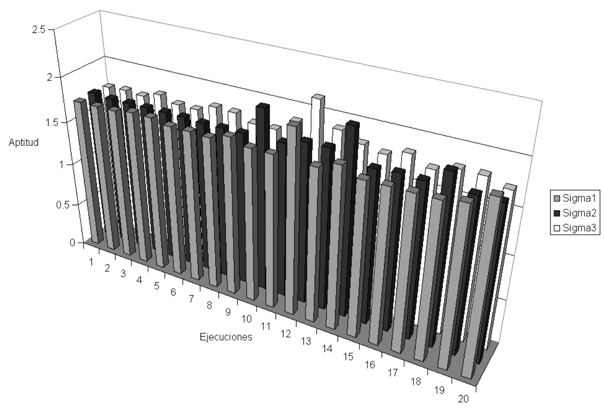

Figura 5.9: Variación de sigma para 20 ejecuciones del problema 2 con representación binaria

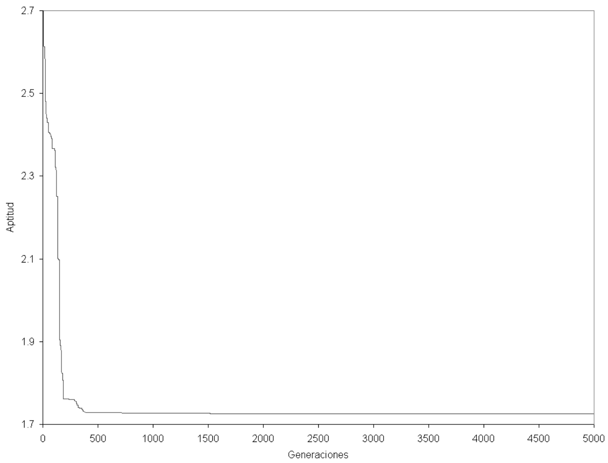

Figura 5.10: Aptitud contra generación de la mejor ejecución del problema 2 con Sigma2 usando representación binaria.

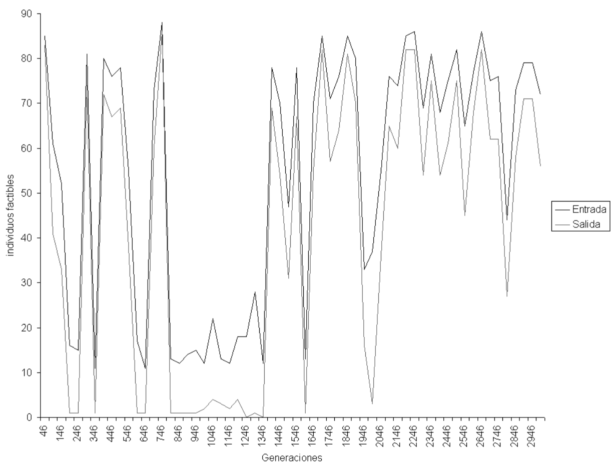

Figura 5.11: Individuos factibles que entran y salen del sistema inmune por generación de la mejor ejecución del problema 2 usando representación binaria.

Representación real

El mejor valor encontrado por el sistema inmune para esta representación es: **f(x)= 1.727616** para los valores de x: (0.204500, 3.500200, 9.033000, 0.205900) con las restricciones: (-5.075303, -0.765967, -0.001400, -3.429711, -0.079500, -0.235535, -13.331683)

Ejecución	Sigma1	Sigma2	Sigma3
1	1.7278	1.7564	1.76
2	1.91	1.7428	1.7419
3	1.8483	1.8008	1.728
4	1.7776	1.7376	1.8178
5	1.7361	1.7549	1.749
6	1.7813	1.8091	1.7867
7	1.7448	1.7768	1.7819
8	1.756	1.7305	1.7797
9	1.7321	1.8304	1.8531
10	1.7522	1.73	1.7287
11	1.7361	1.7815	1.7345
12	1.9199	1.7334	1.7477
13	1.7345	1.7492	1.7276
14	1.7367	1.7413	1.7806
15	1.7353	1.7361	1.7365
16	1.7352	1.732	1.7318
17	1.7463	1.7918	1.7464
18	1.7371	1.7453	2.0619
19	1.7433	1.7351	1.7443
20	1.7281	1.7458	1.7345
Mediana	1.7402	1.74555	1.74705
Promedio	1.765935	1.75804	1.77363
Desviación estándar	0.05786385	0.02981007	0.0754545
Mejor	1.7278	1.73	1.7276
Peor	1.9199	1.8304	2.0619

Tabla 5.5 Variación de σ del problema 2 con representación real.

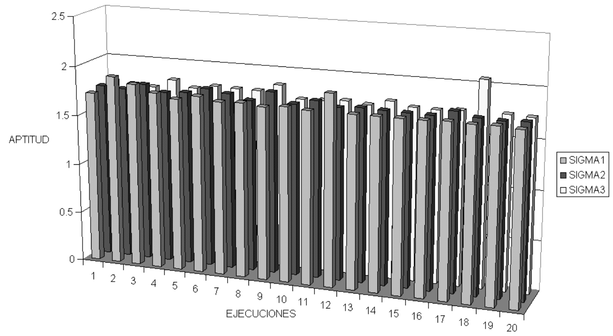

Figura 5.12: Variación de σ para 20 ejecuciones del problema 2 con representación real

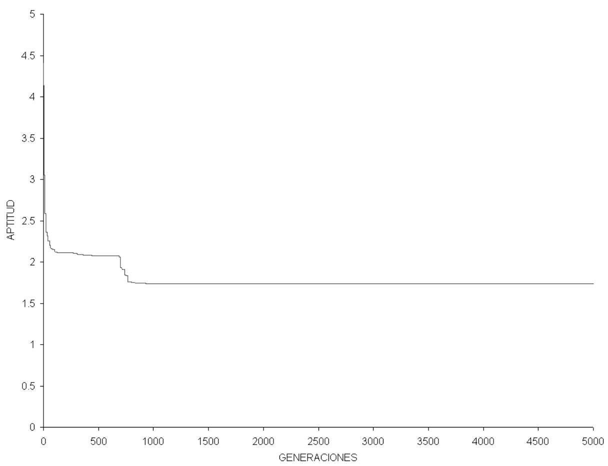

Figura 5.13: Aptitud contra generación de la mejor ejecución del problema 2 con Sigma3 usando representación real.

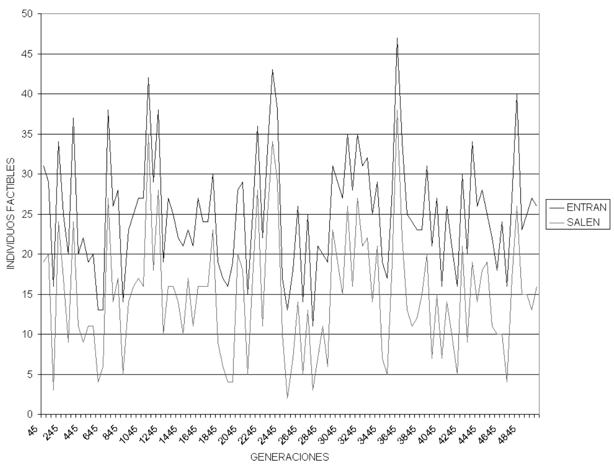

Figura 5.14: Individuos factibles que entran y salen del sistema inmune por generación de la mejor ejecución del problema 2 usando representación real

5.4.3 Problema 3

El óptimo local se encuentra $f(x^*) = -15$, para los valores de: x^*=(1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1). La precisión aplicada es de 4 dígitos.

Representación binaria

El mejor valor encontrado por el sistema inmune para esta representación es: **f(x)=-14.487350**, para los valores de x: (0.993700, 0.999700, 0.994500, 0.999900, 0.984800, 0.991200, 0.981100, 0.979000, 0.967200, 2.957700, 2.897400, 2.800900, 0.988700) con las restricciones: (-0.158100, -0.265000, -0.313300, -4.991900, -5.100200, -5.155100, -0.026900, -0.066100, -0.124300)

Ejecución	Sigma1	Sigma2	Sigma3
1	-12.7339	-14.4874	-11.5978
2	-14.3858	-14.2363	-11.5978
3	-12.0704	-9.1255	-12.4639
4	-14.0594	-14.2522	-13.2072
5	-13.8501	-11.6513	-11.3276
6	-12.0634	-13.8097	-10.1236
7	-11.3657	-12.4932	-11.2266
8	-12.4538	-8.468	-12.3444
9	-9.8763	-11.6649	-13.9531
10	-13.901	-10.2704	-12.7339
11	-10.1236	-12.2198	-14.3858
12	-11.3003	-14.2972	-11.2763
13	-13.8922	-12.0901	-13.8922
14	-14.2276	-12.61	-11.2818
15	-9.9728	-14.097	-13.9356
16	-14.3858	-14.019	-13.8501
17	-14.2363	-11.391	-14.2363
18	-9.1255	-12.6377	-12.4538
19	-14.2522	-11.9966	-11.2763
20	-11.6513	-11.4724	-12.4639
Mediana	-12.59385	-12.3565	-12.45885
Promedio	-12.49637	-12.364485	-12.4814
Desviación Estándar	1.75500158	1.72169428	1.25302711
Mejor	-14.3858	-14.4874	-14.3858
Peor	-9.1255	-8.468	-10.1236

Tabla 5.6 Variación de σ del problema 3 usando representación binaria.

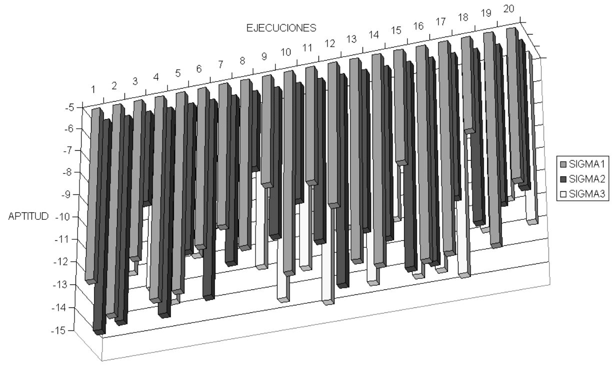

Figura 5.15: Variación de σ para 20 ejecuciones del problema 3 usando representación binaria

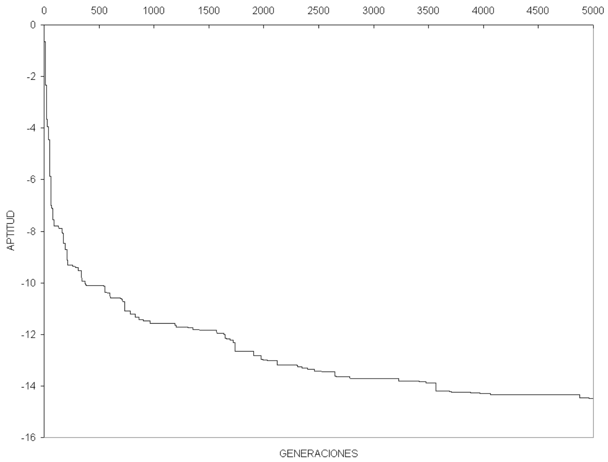

Figura 5.16: Aptitud contra generación de la mejor ejecución del problema 3 con Sigma2 usando representación binaria.

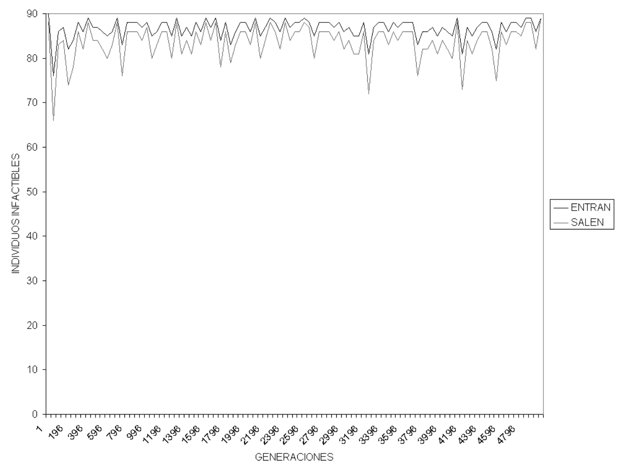

Figura 5.17: Individuos factibles que entran y salen del sistema inmune por generación para la mejor ejecución del problema 3 usando representación binaria

Representación real

El mejor valor encontrado por el sistema inmune para esta representación es: **f(x)=-14.968806**, para los valores de x=(0.999800, 0.998900, 0.999900, 0.999900, 0.999900, 0.999900, 0.999900, 0.999800, 0.999900, 2.998700, 2.989900, 2.989700, 0.998600), con las restricciones: (-0.014000, -0.012200, -0.022800, -4.999700, -5.001300, -5.009500, -0.001000, -0.009800, -0.009800)

Ejecución	Sigma1	Sigma2	Sigma3
1	-14.7507	-14.7682	-14.7694
2	-14.7888	-14.6548	-14.7407
3	-14.7851	-14.8647	-12.7788
4	-14.8634	-13.9733	-14.7696
5	-14.9643	-14.8635	-14.7114
6	-14.867	-13.8377	-14.9544
7	-14.7722	-14.8985	-14.877
8	-14.8793	-13.9356	-14.7086
9	-14.8832	-14.9305	-14.8357
10	-14.7859	-14.8438	-14.6717
11	-14.7773	-14.9592	-14.9688
12	-14.7552	-14.9109	-14.9111
13	-12.8519	-14.7686	-12.7679
14	-14.8703	-14.6739	-12.9235
15	-14.769	-13.9533	-14.9636
16	-14.8565	-13.7851	-14.8551
17	-14.7367	-13.8338	-14.9682
18	-14.7169	-14.7814	-14.8476
19	-14.7949	-14.8647	-14.9622
20	-14.7102	-14.7686	-14.6716
Mediana	-14.7855	-14.7686	-14.80265
Promedio	-14.70894	-14.543505	-14.532845
Desviación estándar	0.44202517	0.44958024	0.74415515
Mejor	-14.9643	-14.9592	-14.9688
Peor	-12.8519	-13.7851	-12.7679

Tabla 5.7 Variación de σ del problema 3 con representación real.

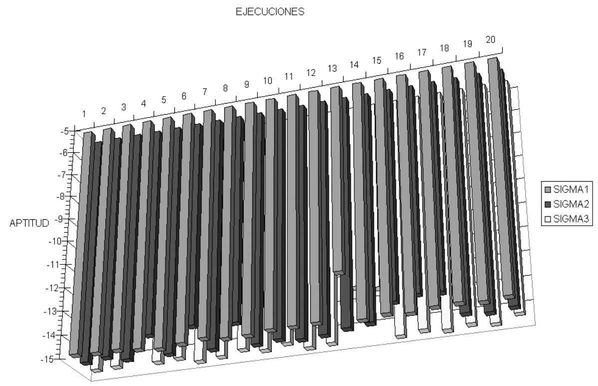

Figura 5.18: Variación de sigma para 20 ejecuciones del problema 3 usando representación real

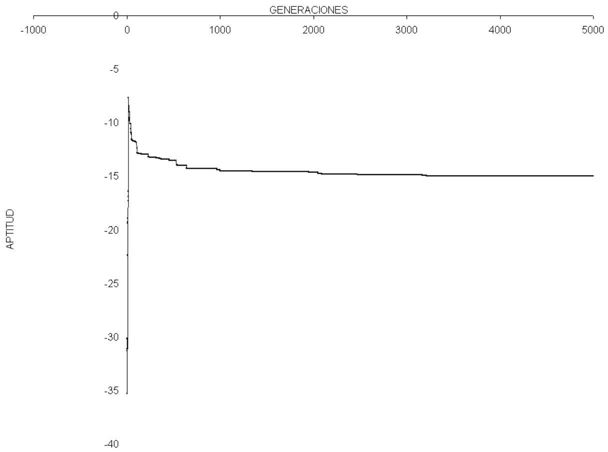

Figura 5.19: Aptitud contra generación de la mejor ejecución del problema 3 con Sigma3 usando representación real.

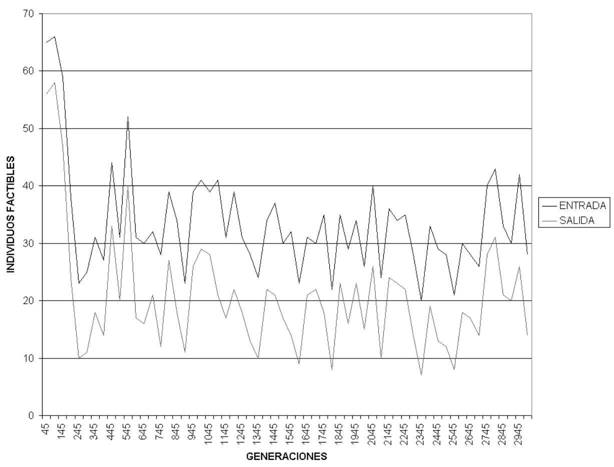

Figura 5.20: Individuos factibles que entran y salen del sistema inmune por generación de la mejor ejecución del problema 3 usando representación real

5.4.4 Problema 4

El mejor valor reportado en la literatura es **f(x)=0.803619**, para los valores de: x^{*}=(1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1).
La precisión aplicada es de 6 dígitos.

Representación binaria

El mejor valor encontrado por el sistema inmune para esta representación es: **f(x)=1.165918**, para los valores de x: (3.373053, 2.290074, 1.433605, 1.656068, 0.967193, 0.845529, 0.613445, 0.653318, 0.897143, 1.662509, 1.509683, 1.501970, 0.916316, 1.131878, 1.205804, 0.174996, 0.810654, 0.995756, 0.858659, 0.395146) con las restricciones: (-0.468461, -126.107201)

Ejecución	Sigma1	Sigma2	Sigma3
1	1.0193	0.9997	1.0356
2	1.0578	0.9617	0.8567
3	0.9148	1.035	1.1659
4	1.018	1.0023	1.0037
5	0.8936	0.9454	1.0146
6	1.059	0.9319	0.9146
7	0.9224	1.067	1.0373
8	0.8861	0.9399	1.0507
9	1.0187	1.0923	1.0809
10	0.9124	1.0748	0.8877
11	0.8789	1.0339	1.0899
12	1.1173	1.0481	1.0323
13	1.0218	0.9336	0.9848
14	1.067	1.0926	0.8022
15	0.9319	0.9375	0.9788
16	1.0187	1.1165	1.0911
17	0.9336	0.919	0.9124
18	1.0356	0.9038	0.8877
19	1.0146	0.9783	0.8861
20	1.0146	0.9522	1.0146
Mediana	1.0146	0.989	1.00915
Promedio	0.986805	0.998275	0.98638
Desviación estándar	0.0703799	0.06687304	0.0937393
Peor	0.8789	0.9038	0.8022
Mejor	1.1173	1.1165	1.1659

Tabla 5.8 Variación de σ del problema 4 con representación binaria.

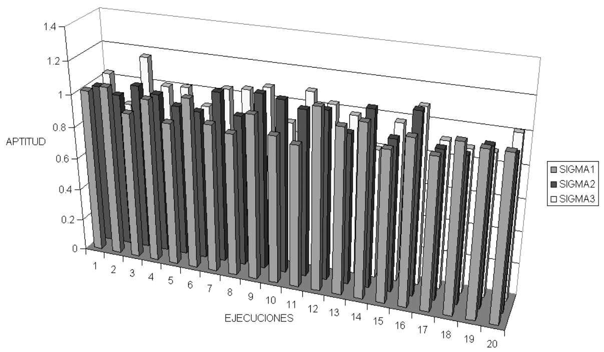

Figura 5.21: Variación de σ para 20 ejecuciones del problema 4 usando representación binaria

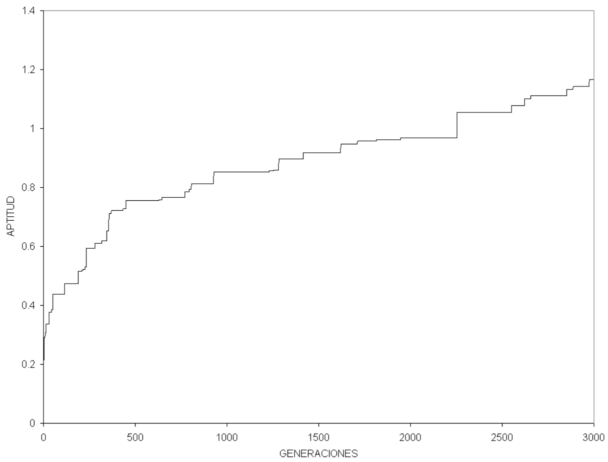

Figura 5.22: Aptitud contra generación de la mejor ejecución del problema 4 con Sigma3 usando representación binaria.

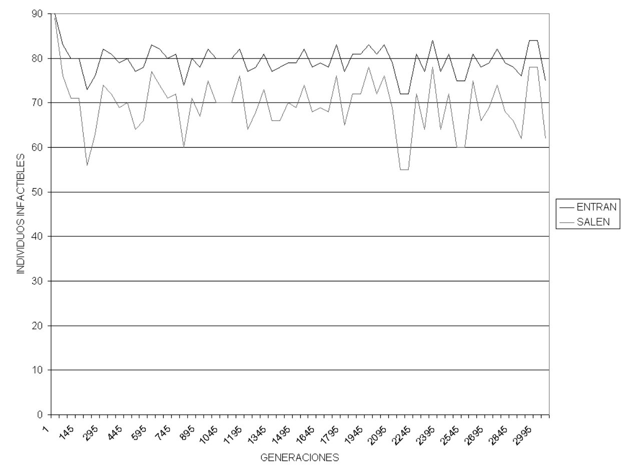

Figura 5.23: Individuos factibles que entran y salen del sistema inmune por generación del la mejor ejecución del problema 4 usando representación binaria

Representación real.

El mejor valor encontrado por el sistema inmune para esta representación es: **f(x)=1.332873**, para los valores de x: (1.889508, 1.288910, 1.492300, 1.558005, 1.211474, 1.742341, 0.845707, 1.123716, 1.010017, 0.768072, 0.982819, 0.904926, 0.685947, 1.019152, 1.005444, 1.021039, 0.464163, 0.844014, 0.511112, 0.668843) con las restricciones: (-0.003230, -128.962491).

Ejecución	Sigma1	Sigma2	Sigma3
1	1.2067	1.102	1.2102
2	1.3048	1.2876	1.0299
3	1.2192	1.2622	1.2899
4	1.0925	1.2975	1.0153
5	1.2935	1.0086	1.1733
6	1.0627	1.3026	1.2579
7	1.2244	1.2626	1.1483
8	1.2832	1.0967	1.2477
9	1.0554	1.2029	1.2207
10	1.22	1.3329	1.1291
11	1.2614	1.1161	1.3147
12	1.1786	1.2741	1.2525
13	1.2338	1.2323	1.1648
14	1.2817	1.2067	1.1064
15	1.2611	1.2997	1.1427
16	1.108	1.1858	1.1885
17	1.0446	1.207	1.2916
18	1.1151	1.0432	1.1357
19	1.1941	1.1482	1.2603
20	1.2276	1.2493	1.2747
Mediana	1.2196	1.21965	1.19935
Promedio	1.19342	1.2059	1.19271
Desviación estándar	0.08426256	0.09259042	0.08451383
Peor	1.0446	1.0086	1.0153
Mejor	1.3048	1.3329	1.3147

Tabla 5.9 Variación de σ del problema 4 usando representación real.

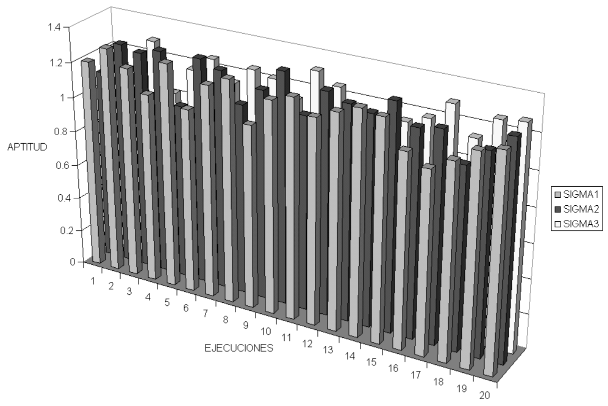

Figura 5.24: Variación de σ para 20 ejecuciones del problema 4 usando representación real

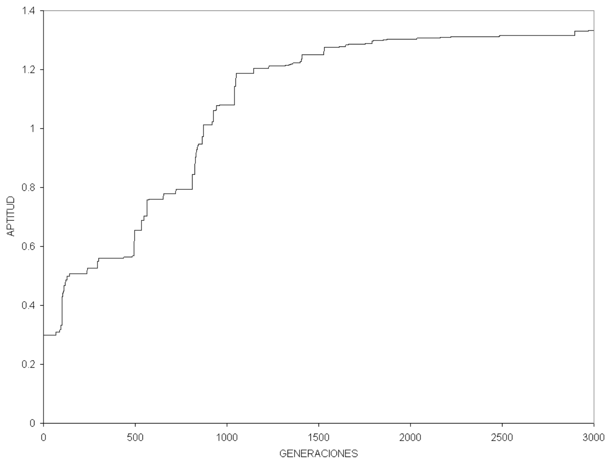

Figura 5.25: Aptitud contra generación de la mejor ejecución del problema 4 con Sigma2 usando representación real.

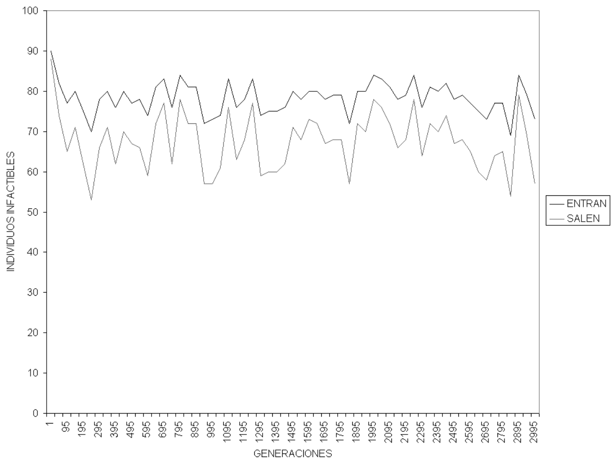

Figura 5.26: Individuos factibles que entran y salen del sistema inmune por generación de la mejor ejecución del problema 4 usando representación real

5.4.5 Problema 5

El mínimo global está en x^*=(78,33,29.995256,45,36.775812) donde $f(x^*)$=**-30665.539.** La precisión aplicada es de 6 dígitos.

Para realizar los experimentos con esta función se aplicaron 10,000 generaciones.

Representación binaria

El mejor valor encontrado por el sistema inmune para esta representación es: $f(x) = -30664.834963$ para los valores de x: (78.000073, 33.007052, 29.999690, 44.999925, 36.764667) con las restricciones: (-0.000241, -91.999759, -11.158039, -8.841961, -4.999992, -0.000008).

Ejecución	Sigma1	Sigma2	Sigma3
1	-30563.335	-30653.9346	-30631.8399
2	-30496.1548	-30663.8753	-30616.3501
3	-30564.608	-30651.7701	-30632.2275
4	-30479.4398	-30643.9223	-30497.1526
5	-30631.4126	-30664.835	-30600.1306
6	-30495.245	-30664.0142	-30614.4131
7	-30496.4674	-30650.4632	-30478.3318
8	-30611.164	-30662.27	-30492.2498
9	-30496.3875	-30641.4488	-30600.0981
10	-30597.4776	-30654.8181	-30496.7137
11	-30459.5211	-30657.9184	-30648.9357
12	-30578.5589	-30654.5066	-30497.381
13	-30598.5637	-30597.4776	-30619.9003
14	-30544.5005	-30493.6606	-30648.9569
15	-30493.6606	-30648.9357	-30625.847
16	-30497.2637	-30643.9223	-30562.7038
17	-30497.4006	-30600.1306	-30625.3234
18	-30479.5383	-30628.1817	-30628.1817
19	-30597.8486	-30562.7038	-30632.8053
20	-30578.2365	-30609.3528	-30609.3528
mediana	-30520.9506	-30649.6995	-30615.3816
promedio	-30537.8392	-30632.4071	-3058
7.9448 desv.est.	53.7230651	42.4753501	59.6728747
mejor	-30631.4126	-30664.835	-30648.9569
peor	-30459.5211	-30493.6606	-30478.3318

Tabla 5.12 Variación de σ del problema 5 usando representación binaria.

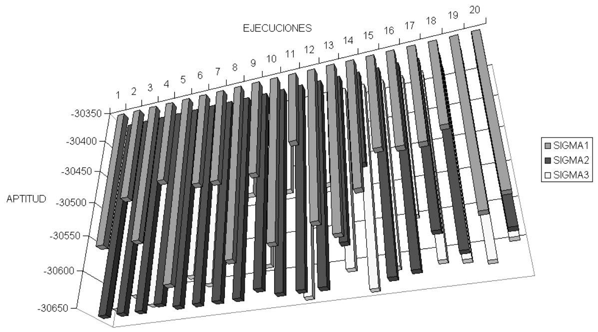

Figura 5.27: Variación de σ para 20 ejecuciones del problema 5 con representación binaria

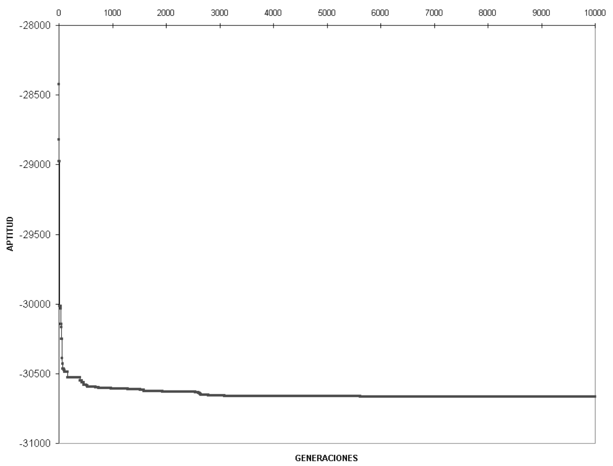

Figura 5.28: Aptitud contra generación de la mejor ejecución del problema 5 con Sigma2 usando representación binaria.

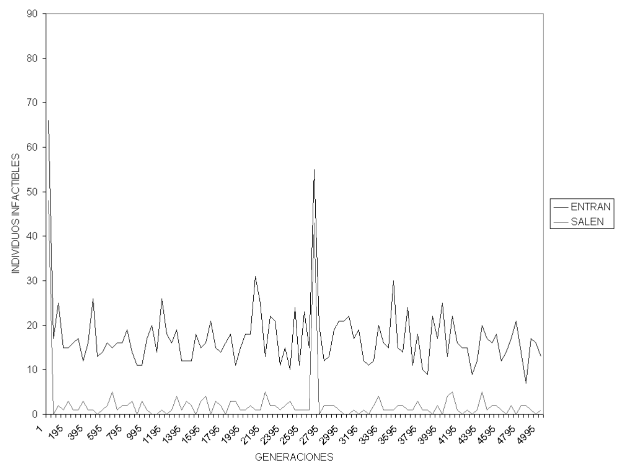

Figura 5.29: Individuos factibles que entran y salen del sistema inmune por generación de la mejor ejecución del problema 5 usando representación binaria

Representación real

El mejor valor encontrado por el sistema inmune para esta representación es: **f(x)=-30648.956939**

Para los valores de x=(78.000100, 33.094745, 30.100000, 44.997928, 36.512700), con las restricciones: (-0.020827, -91.979173, -11.160873, -8.839127, -4.999997, -0.000003)

Ejecución	Sigma1	Sigma2	Sigma3
1	-30563.335	-30580.8131	-30631.8399
2	-30496.1548	-30632.9567	-30616.3501
3	-30564.608	-30566.8994	-30632.2275
4	-30479.4398	-30495.245	-30497.1526
5	-30631.4126	-30497.2629	-30600.1306
6	-30495.245	-30578.5812	-30614.4131
7	-30496.4674	-30627.2828	-30478.3318
8	-30611.164	-30491.894	-30492.2498
9	-30496.3875	-30479.5373	-30600.0981
10	-30597.4776	-30581.8321	-30496.7137
11	-30459.5211	-30497.263	-30648.9357
12	-30578.5589	-30495.6332	-30497.381
13	-30598.5637	-30648.4872	-30619.9003
14	-30544.5005	-30616.6576	-30648.9569
15	-30493.6606	-30614.4124	-30625.847
16	-30497.2637	-30477.6054	-30562.7038
17	-30497.4006	-30600.3255	-30625.3234
18	-30479.5383	-30631.1029	-30628.1817
19	-30597.8486	-30611.7726	-30632.8053
20	-30578.2365	-30576.807	-30609.3528
Mediana	-30520.9506	-30579.6972	-30615.3816
Promedio	-30537.8392	-30565.1186	-30587.9448
Desviación estándar	53.7230651	59.9927566	59.6728747
Mejor	-30631.4126	-30648.4872	-30648.9569
Peor	-30459.5211	-30477.6054	-30478.3318

Tabla 5.13 Variación de σ del problema 5 con representación real.

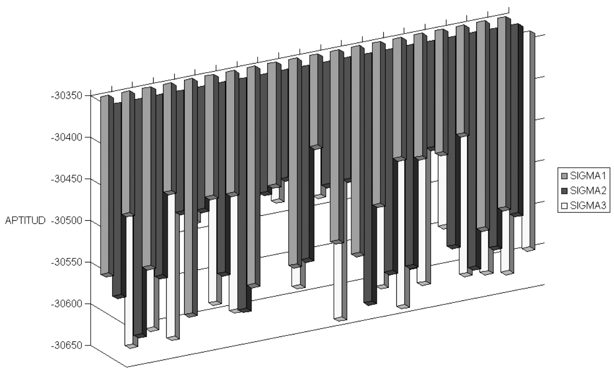

Figura 5.30: Variación de σ para 20 ejecuciones del problema 5 con representación real

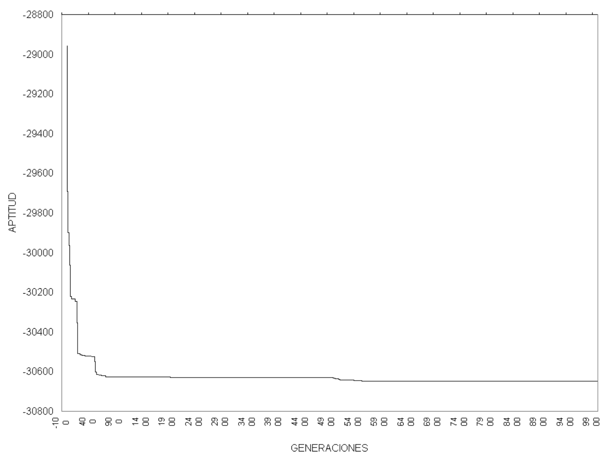

Figura 5.31: Aptitud contra generación para la mejor ejecución del problema 5 con Sigma3 usando representación real.

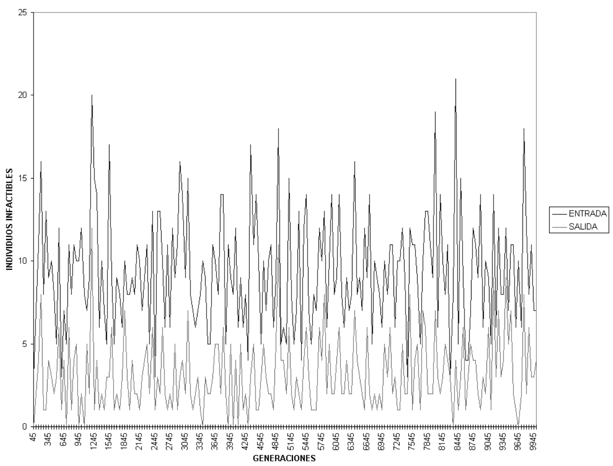

Figura 5.32: Individuos factibles que entran y salen del sistema inmune por generación para la mejor ejecución del problema 5 usando representación real

5.4.6 Problema 6

El mejor valor reportado por la literatura es [37]: **f(x)=-6961.81388**, para los valores de x=(14.095, 0.84296) La precisión aplicada es de 5 dígitos.

Representación binaria

El mejor valor encontrado por el sistema inmune para esta representación es: **f(x)=-6961.724654**, para los valores de x=(14.095040, 0.843040), con las restricciones: (-0.000069, -0.000011)

Ejecución	Sigma1	Sigma2	Sigma3
1	-6940.0099	-6958.5264	-6959.9115
2	-6960.8593	-6944.9281	-6961.3868
3	-6960.9699	-6960.1492	-6961.7021
4	-6954.1461	-6818.9964	-6961.7021
5	-6961.1621	-6957.1275	-6961.7021
6	-6956.199	-6961.1841	-6961.7247
7	-6961.4994	-6961.4994	-6961.7247
8	-6960.2618	-6961.4994	-6930.6892
9	-6961.4994	-6953.2058	-6960.8458
10	-6961.4994	-6960.8458	-6957.4902
11	-6959.2459	-6958.2772	-6961.3077
12	-6947.1923	-6949.7423	-6961.4994
13	-6959.8585	-6956.8403	-6961.4994
14	-6961.567	-6960.1932	-6941.8163
15	-6905.4982	-6948.9994	-6961.7247
16	-6960.9699	-6961.1951	-6961.7247
17	-6958.2607	-6961.7247	-6961.7247
18	-6959.7794	-6961.7247	-6961.7247
19	-6961.7021	-6956.3271	-6960.1382
20	-6960.8573	-6960.8458	-6960.7357
Mediana	**-6960.55955**	**-6959.3378**	**-6961.4994**
Promedio	**-6955.65188**	**-6950.6916**	**-6958.63874**
Desv. Estándar	**13.0287411**	**31.3723605**	**7.93048887**
Mejor	**-6961.7021**	**-6961.7247**	**-6961.7247**
Peor	**-6905.4982**	**-6818.9964**	**-6930.6892**

Tabla 5.14 Variación de σ del problema 6 usando representación binaria.

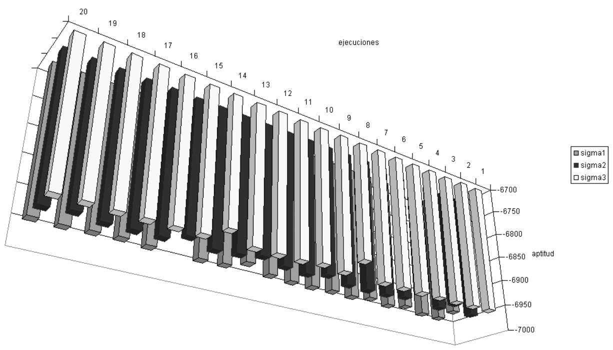

Figura 5.33: Variación de σ para 20 ejecuciones del problema 6 usando representación binaria

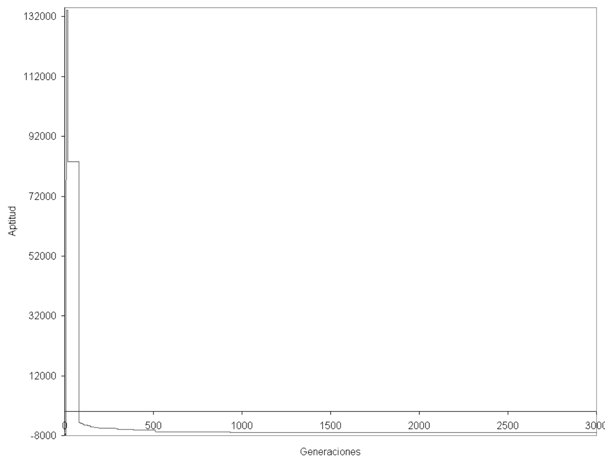

Figura 5.34: Aptitud contra generación de la mejor ejecución del problema 6 con Sigma3 usando representación binaria

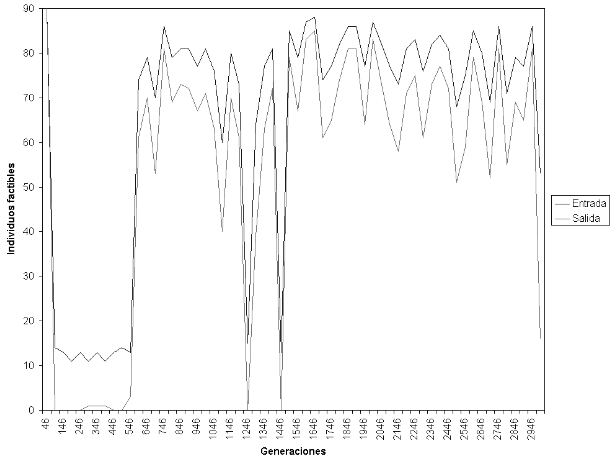

Figura 5.35: Individuos factibles que entran y salen del sistema inmune por generación de la mejor ejecución del problema 6, usando representación binaria

Representación real

El mejor valor encontrado por el sistema inmune para esta representación es: **f(x)=-6960.609596**, para los valores de x:(14.095540, 0.844030), con las restricciones: (-0.000935, -0.000145).

Ejecución	Sigma1	Sigma2	Sigma3
1	-6950.8228	-6955.0159	-6950.5149
2	-6950.8228	-6950.8228	-6950.8228
3	-6950.8228	-6950.8228	-6950.8228
4	-6950.8228	-6950.8228	-6950.8228
5	-6950.8228	-6950.8228	-6950.8228
6	-6950.8228	-6950.8228	-6950.8228
7	-6950.7349	-6950.8228	-6950.8228
8	-6951.64	-6950.8228	-6950.8228
9	-6950.8228	-6954.6307	-6950.8228
10	-6953.9046	-6786.7384	-6958.3979
11	-6950.7459	-6950.5149	-6771.8037
12	-6950.8228	-6950.8228	-6952.9847
13	-6950.8228	-6948.265	-6676.4414
14	-6950.8228	-6950.8228	-6950.8228
15	-6942.6675	-6957.2231	-6950.8228
16	-6952.7146	-6950.8228	-6953.9046
17	-6928.7932	-6950.8228	-6960.6096
18	-6950.8228	-6950.8228	-6950.8228
19	-6786.7405	-6950.8228	-6950.8228
20	-6950.8228	-6955.225	-6953.9046
Mediana	-6950.8228	-6950.822	-6950.8228
Promedio	-6941.39074	-6943.41547	-6929.42175
Desviación Estándar	36.784791	36.9390551	71.9465515
Mejor	-6953.9046	-6957.2231	-6960.6096
Peor	-6786.7405	-6786.7384	-6676.4414

Tabla 5.15 Variación de σ del problema 6 usando representación real.

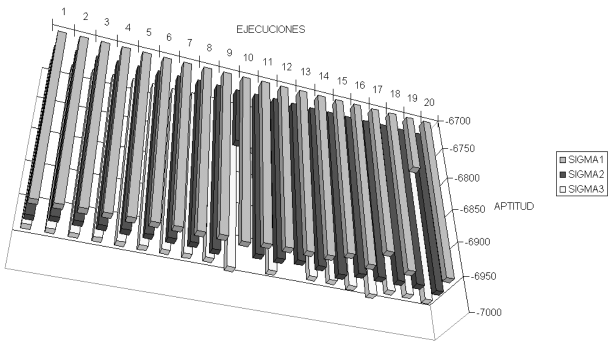

Figura 5.36: Variación de σ para 20 ejecuciones del problema 6 usando representación real

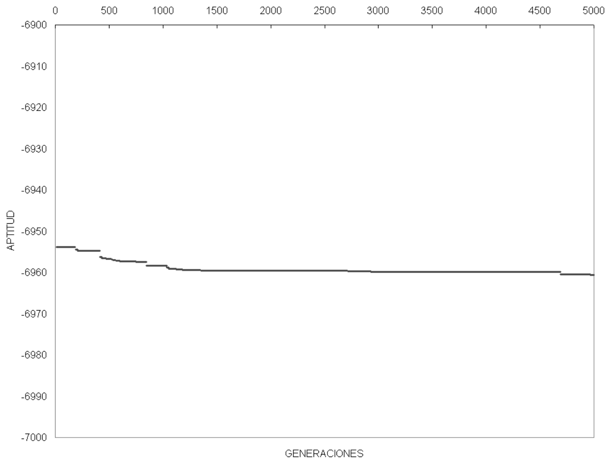

Figura 5.37: Aptitud contra generación de la mejor ejecución del problema 6 con Sigma3 usando representación real

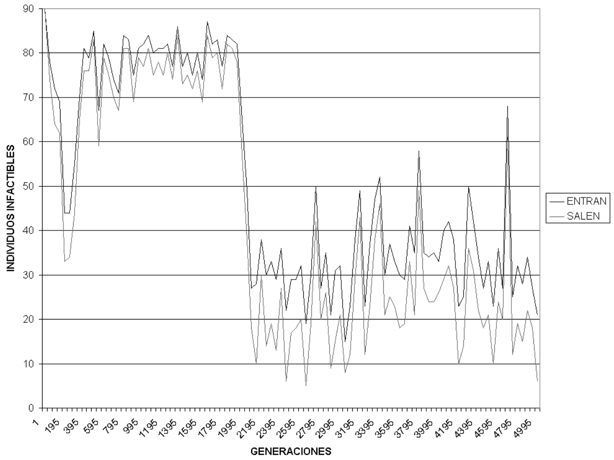

Figura 5.38: Individuos factibles que entran y salen del sistema inmune por generación de la mejor ejecución del problema 6 usando representación real

5.4.7 Resultados del problema 7

El máximo global está localizado en [39] $x^* = (5, 5, 5)$ donde $f(x^*) = 1$. La precisión aplicada es de 4 dígitos.

Para todos los experimentos que se realizaron con esta función, tanto en la representación real como binaria, el algoritmo siempre encontró el valor óptimo.

Representación binaria

Ejecución	**Sigma1**	**Sigma2**	**Sigma3**
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	1.00	1.00	1.00
6	1.00	1.00	1.00
7	1.00	1.00	1.00
8	1.00	1.00	1.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	1.00	1.00	1.00
12	1.00	1.00	1.00
13	1.00	1.00	1.00
14	1.00	1.00	1.00
15	1.00	1.00	1.00
16	1.00	1.00	1.00
17	1.00	1.00	1.00
18	1.00	1.00	1.00
19	1.00	1.00	1.00
20	1.00	1.00	1.00
Mediana	1.00	1.00	1.00
Promedio	1.00	1.00	1.00
Desviación estándar	0	0	0
Mejor	1.00	1.00	1.00
Peor	1.00	1.00	1.00

Tabla 5.16 Variación de σ del problema 7 usando representación binaria.

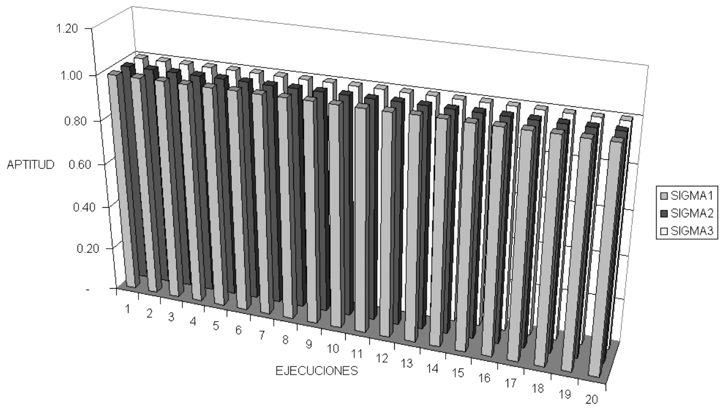

Figura 5.39: Variación de σ para 20 ejecuciones del problema 7 usando representación binaria

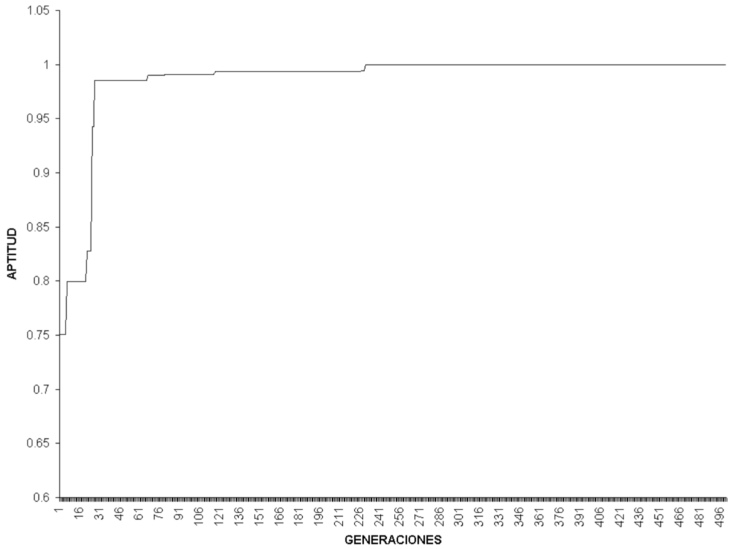

Figura 5.40: Aptitud contra generación para la mejor ejecución del problema 7, usando resentación binaria (para cualquier valor de Sigma

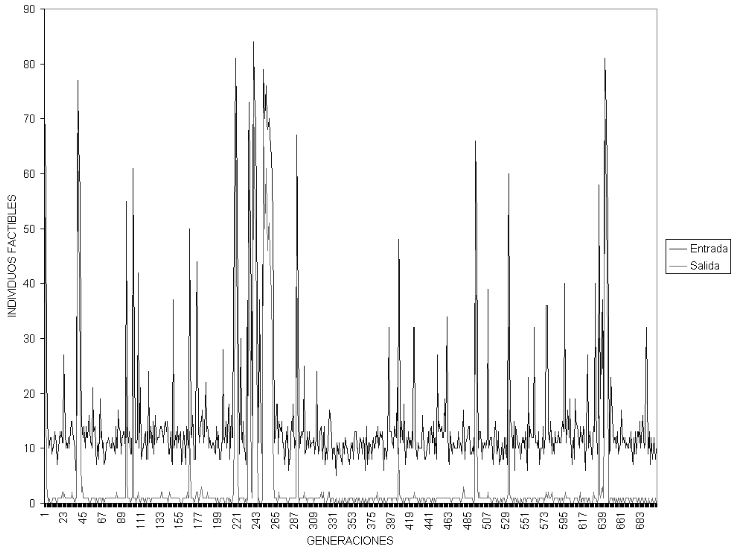

Figura 5.41: Individuos factibles que entran y salen del sistema inmune por generación para el problema 7 usando representación binaria

Representación real

Ejecución	**Sigma1**	**Sigma2**	**Sigma3**
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	1.00	1.00	1.00
6	1.00	1.00	1.00
7	1.00	1.00	1.00
8	1.00	1.00	1.00
9	1.00	1.00	1.00
10	1.00	1.00	1.00
11	1.00	1.00	1.00
12	1.00	1.00	1.00
13	1.00	1.00	1.00
14	1.00	1.00	1.00
15	1.00	1.00	1.00
16	1.00	1.00	1.00
17	1.00	1.00	1.00
18	1.00	1.00	1.00
19	1.00	1.00	1.00
20	1.00	1.00	1.00
Mediana	1.00	1.00	1.00
Promedio	1.00	1.00	1.00
Desviación estándar	0	0	0
Mejor	1.00	1.00	1.00
Peor	1.00	1.00	1.00

Tabla 5.17 Variación de σ del problema 7 usando representación real.

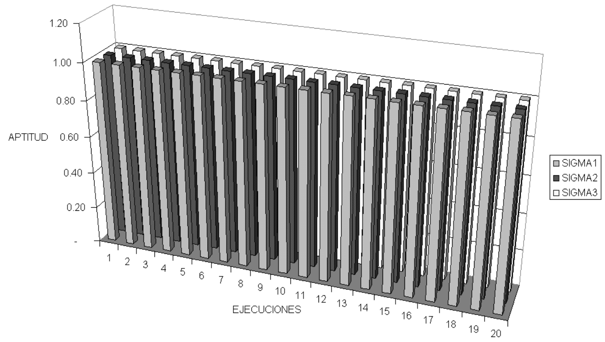

Figura 5.42: Variación de σ para 20 ejecuciones del problema 7 usando representación real

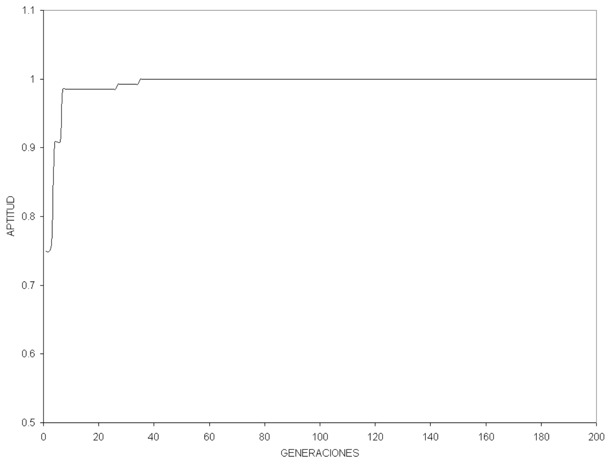

Figura 5.43: Aptitud contra generación para la mejor ejecución del problema 7, usando representación real (para cualquier valor de Sigma)

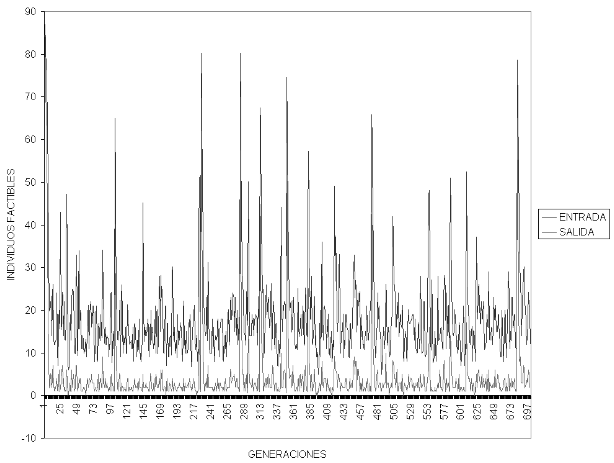

Figura 5.44: Individuos factibles que entran y salen del sistema inmune por generación para el problema 7, usando representación real

5.5 Discusión de resultados

De acuerdo a los resultados presentados anteriormente, podemos observar que el algoritmo funciona de manera aceptable, e inclusive en algunos casos superando los valores que se han reportado previamente en la literatura.

Con lo que respecta al tipo de representación que se aplicó, los problemas con alta dimensionalidad (problemas 3 y 4, con 13 y 20 variables respectivamente) mostraron un desempeño mejor cuando se usa representación real que cuando se utiliza representación binaria.

Para el caso de las funciones que tienen baja dimensionalidad (problemas 1, 2, 5, 6 y 7 que tienen cinco o menos variables) los mejores resultados se obtuvieron al usar representación binaria.

Este efecto puede ser causado por la aplicación de la cruza de un punto, pues presupone bloques constructores con esquemas cortos y de bajo orden, y cuando esto no sucede (como es el caso de las cadenas largas), suele no proporcionar resultados apropiados, de manera que, en esas cadenas más largas el efecto de la cruza se hace menos perceptible y en ese caso se requiere un número mayor de generaciones o aplicar otro tipo de cruza.

La razón por la que en este trabajo de tesis se aplicó el operador de cruza de un punto, es porque se tuvo la intención de experimentar con la forma más simple del algoritmo genético.

En las pruebas con diferentes valores del parámetro σ, que determina la cantidad de anticuerpos que son expuestos ante el antígeno, podemos detectar que su variación no parece afectar realmente al desempeño del algoritmo, pues para todos los casos el comportamiento con uno u otro valor es similar.

Es importante enfatizar, sin embargo, que las tres variantes que fueron usadas en los experimentos están orientadas a buscar la formación de anticuerpos generalistas.

En pruebas previas a estos experimentos, se utilizaron valores de σ, tanto fijos como aquellos que generan anticuerpos especialistas, con la finalidad de comprobar que efectivamente el algoritmo funciona mejor si se buscan anticuerpos generalistas.

Podemos ver que el algoritmo se desempeñó bien para todas las funciones de prueba a las que fue aplicado, sin tener distinción entre las problemas con desigualdades lineales o no lineales, con zonas factibles pequeñas o muy grandes.

Conclusiones

Podemos concluir, con base en los resultados mostrados, que la similitud a nivel del genotipo, implica una similitud a nivel del fenotipo. Esta similitud puede ser explotada por métodos como el aquí presentado para reducir el costo computacional asociado con el manejo de restricciones en algoritmos evolutivos.

Este costo computacional adicional se refiere a que a diferencia de las técnicas tradiccionales para manejo de restricciones en las que se deben recalcular los valores de violación y algunos otros factores; en esta técnica la aptitud de cada individuo se obtiene por medio de un chequeo sencillo de bits, y que además combina los esquemas de tanto individuos factibles como de aquellos que no lo son.

Para problemas de alta dimensionalidad, la representación real usando números enteros, obtiene mejores resultados, y para el caso de funciones de baja dimensionalidad los mejores valores se obtienen usando representación binaria.

El algoritmo demostró ser bastante rubusto al trabajar sobre funciones con características diferentes ya sean cuadráticas, cúbicas, no lineales, en espacios de búsqueda no convexos, y con desigualdades no lineales.

Trabajos futuros

Algunas de las extensiones o trabajos futuros que podrían realizarse sobre la simulación del sistema inmune son las siguientes:

- El algoritmo descrito únicamente emula de manera burda una pequeña parte del sistema inmune biológico, que se refiere al reconocimiento de antígenos extraños para promover la creación de los anticuerpos que le son específicos. Sería interesante ir más allá del proceso de reconocimiento y simular otras funciones del sistema inmune.

- Basado en el modelo de red inmune de Jerne [33], hacer una simulación en la cual el reconocimiento de agentes extraños se realiza por niveles, provocando de esta manera que el sistema posea una respuesta secundaria o memoria, así que en exposiciones repetidas ante un mismo antígeno se tengan respuestas cada vez más rápidas, mejorando con ello la eficiencia del algoritmo. Asimismo, permitiría la creación de nuevos antígenos en diferentes niveles.

- Hacer un análisis detallado de cómo esta simulación podría ser una alternativa para mantener la diversidad en la población de un algoritmo genético, basada en el hecho de que los anticuerpos tienen la capacidad de reconocer casi cualquier tipo de antígeno, y para lograrlo necesitan mantener diversidad.

- Aplicar la simulación del sistema inmune en la resolución de problemas de optimización con múltiples objetivos [7], que sería análogo a exposición ante diversos antígenos a un mismo tiempo. Aunque se ha usado esta técnica para optimización multimodal (para mantener diversidad [47]), no se ha utilizado hasta ahora para optimización multiobjetivo.

Bibliografía

[1] Abul K. Abbas, Andrew H. Lichtman, and Jordan S. Pober. *Inmunología Celular y Molecular.* McGraw Hill Interamericana, 1995.

[2] David H. Ackley. *A Connectionist Machine for Genetic Hillclimbing.* Kluwer Academic Publishers, Boston, Massachusetts, 1995.

[3] E. Baldwin. *Genética Elemental.* Limusa, 1983.

[4] Bill P. Buckles and Fred E. Petry, editors. *Genetic Algorithms.* IEEE Computer Society Press, 1992.

[5] Carlos A. Coello Coello. La importancia de la representación en los algoritmos genéticos (parte I). *Soluciones Avanzadas*, Año 7(69):50–56, Mayo 1999.

[6] Carlos A. Coello Coello. MOSES: A Multiobjetive Optimization Tool for Engineering Design. *Engineering Optimization*, 31(3):337–368, 1999.

[7] Carlos A. Coello Coello. A Survey of Constraint Handling Techniques used with Evolutionary Algorithms. Technical Report Lania-RI-99-05, Laboratorio Nacional de Informática avanzada, Xalapa, Veracruz, México, 1999.

[8] Carlos A. Coello Coello. Functions used to test constraint-handling techniques. Technical Report Lania-RI-2000-04, Laboratorio Nacional de Informática avanzada, Xalapa, Veracruz, México, 2000.

[9] Carlos A. Coello Coello. Treating Constrains as Objetives for Single-Objetive Evolutionary Optimization. *Engineering Optimization*, 32(2):275–308, 2000.

[10] Carlos A. Coello Coello. Use of a self-adaptative penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113–127, 2000.

[11] Carlos A. Coello Coello, Alan D. Christiansen, and Arturo Hernández Aguirre. Using a New GA-Based Multiobjetive Optimization for Robot Arms. *Robotica*, 16(4):401–414, 1998.

[12] Charles Darwin. *The Origin of Species by Means of Natural Selection or the preservation of Favored Races in the Struggle for life.* Random House, New York, 1993. (Publicado originalmente en 1929).

[13] Dipankar Dasgupta. *Artificial Immune Systems.* Springer-Verlag, Berlin, 1998.

[14] Dipankar Dasgupta and Nii Attoh-Okine. Immunity-Based Systems: A Survey. In *IEEE International Conference on Systems, Man and Cybernetics*, Orlando, Florida, October 12-15 1997.

[15] Lawrence Davis, editor. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[16] Klaus D. Elgert. *Immunology, Understanding the Immune System.* Wiley-Liss, USA, 1996.

[17] David B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on neural networks*, 5(1):3–14, January 1994.

[18] David B. Fogel. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*. The Institute of Electrical and Electronic Engineers, New York, 1995.

[19] Lawrence J. Fogel. *Intelligence Through Simulated Evolution. Forty years of Evolutionary Programming*. John Wiley & Sons, New York, 1999.

[20] S. Forrest, A.S. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *IEEE Symposium on Research in Security and Privacy*, pages 202–212, Oakland, CA, May 16-18 1994. Springer-Verlag.

[21] Stephanie Forrest and Alan S. Perelson. Genetic algorithms and the immune system. In Hans-Paul Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, pages 320–325. Springer-Verlag, Berlin, Germany, 1991.

[22] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Alabama, USA, 1953.

[23] David E. Goldberg and Kalyanmoy Deb. A Comparison of Selection Schemes Used in Genetic Algorithms. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93, San Mateo, California, 1989. Morgan Kaufmann.

[24] David E. Goldberg and Robert E. Smith. Nonstationary Function Optimization Using Genetic Algorithms with Dominance and Diploidy. In John J. Grefenstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic*

Algorithms, pages 59–68. Lawrence Erlbaum Associates, Hillsdale, New Jersey, July 1987.

[25] P. Hajela and J. Lee. Constrained Genetic Search via Schema Adaptation. An Immune Network Solution. In Niels Olhoff and George I.Ñ. Rozvany, editors, *Proceedings of the First World Congress of Stuctural and Multidisciplinary Optimization*, pages 915–920, Goslar, Germany, 1995. Pergamon.

[26] P. Hajela and J. Lee. Constrained Genetic Search via Schema Adaptation. An Immune Network Solution. *Structural Optimization*, 12:11–15, 1996.

[27] P. Hajela and J. Yoo. Constraint Handling in Genetic Search Using Expression Strategies. *AIAA Journal*, 34(12):2414–2420, 1996.

[28] W. D. Hillis. Co-evolving parasites improves simulated evolution as an optimization procedure. In C. Langton, C. Taylor, J. Farmer, and S. Rasmussen, editors, *Artificial Life II*, pages 313–324. Addison-Wesley, Reading, Massachusetts, 1992.

[29] W.D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.

[30] John H. Holland. *Adaptation in Natural and Artificial Systems.* Ann Harbor: University of Michigan Press, Michigan USA, 1975.

[31] John H. Holland. *Adaptation in Natural an Artificial Systems.* MIT Press, Cambridge, Massachusetts, second edition, 1992.

[32] John Jenkins, editor. *Genetics.* Houghton Mifflin Company, Boston, Massachusetts, 1984.

[33] N. K. Jerne. The immune system. *Scientific American*, 229(1):52–60, 1973.

[34] A. K. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptative Systems*. PhD thesis, University of Michigan, Michigan, 1975.

[35] Kenneth A. De Jong. Genetic Algorithms are NOT Function Optimizers. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 5–17. Morgan Kaufmann Publishers, San Mateo, California, 1993.

[36] John R. Koza. *Genetic Programming. On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, Massachusetts, 1992.

[37] Slawomir Koziel and Zbigniew Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[38] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, third edition, 1996.

[39] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–36, 1996.

[40] Melanie Mitchell and Stephanie Forrest. Fitness Landscapes: Royal Road Functions. In T. Bäck, D. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*. Oxford University Press, Oxford, USA, 1992.

[41] Jan Paredis. Co-evolutionary constraint satisfaction. In Julian Miller, Adrian Thompson, Peter Thomson, and Terence C. Fogarty, editors,

Proceedings of the 3rd Conference on Parallel Problem Solving from Nature, pages 21–30, Edinburgh, Scotland, April 2000. Springer-Verlag.

[42] Colin B. Reeves. *Modern Heuristic Techniques for Combinatorial Problems.* John Wiley & Sons, Great Britain, 1993.

[43] Jon T. Richardson, Mark R. Palmer, Gunar Liepins, and Mike Hilliard. Some Guidelines for Genetic Algorithms with Penalty Functions. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 191–197, George Mason University, 1989. Morgan Kaufmann Publishers.

[44] Glenn W. Rowe, editor. *The Theoretical Models in Biology.* Oxford University Press, Oxford, first edition, 1994.

[45] Marc Schoenauer and Spyros Xanthakis. Constrained GA Optimization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 573–580, San Mateo, California, 1993. Morgan Kaufmann Publishers.

[46] Hans-Paul Schwefel, editor. *Evolution and Optimization Seeking.* John Wiley & Sons, New York, 1995.

[47] Robert E. Smith, Stephanie Forrest, and Alan S. Perelson. Population Diversity in an Immune System Model: Implications for Genetic Search. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 153–165. Morgan Kaufmann Publishers, San Mateo, California, 1993.

[48] Patrick D. Surry and Nicholas J. Radcliffe. The COMOGA Method: Constrained Optimisation by Multiobjective Genetic Algorithms. *Control and Cybernetics*, 26(3), 1997.

[49] Patrick D. Surry, Nicholas J. Radcliffe, and Ian D. Boyd. A Multi-Objective Approach to Constrained Optimisation of Gas Supply Networks : The COMOGA Method. In Terence C. Fogarty, editor, *Evolutionary Computing. AISB Workshop. Selected Papers*, Lecture Notes in Computer Science, pages 166–180. Springer-Verlag, Sheffield, U.K., 1995.

[50] Gilbert Syswerda. Uniform Crossover in Genetic Algorithms. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, San Mateo, California, 1989. Morgan Kaufmann.

[51] A. Wetzel. *Evaluation of Efectiveness of genetic algorithms in combinational optimization.* University of Pittsburgh, Pittsburgh (unpublished), 1983.

[52] Darrel Whitley. The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121, San Mateo, California, July 1989. Morgan Kaufmann Publishers.

[53] J. Yoo and P. Hajela. Enhanced GA Based Search Through Immune System Modeling. In *3rd World Congress on Structural and Multidisciplinary Optimization*, Niagara Falls, New York, May 1999.