Centro de Investigación y de Estudios
Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Ingeniería Eléctrica

Sección de Computación

Sistema inmune artificial para solucionar problemas de optimización.

Tesis que presenta

Nareli Cruz Cortés

para obtener el Grado de

Doctora en Ciencias

en la Especialidad de

Ingeniería Eléctrica

con Opción en Computación

Director de la Tesis

Dr. Carlos A. Coello Coello

México, D.F. Octubre de 2004

Este trabajo esta dedicado a:
Mi hija Ana Iremi, mi esposo Francisco José y mi madre Enriqueta.

Este trabajo de tesis se derivó del proyecto NSF-CONACyT titulado "Artificial Immune Systems for Multiobjective Optimization" (Ref. 42435-Y), cuyo responsable es el Dr. Carlos A. Coello Coello. Se agradece el generoso apoyo obtenido a través de dicho proyecto.

Un agradecimiento especial para el Dr. Carlos A. Coello Coello por su constante orientación y motivación.

Gracias al Dr. Manuel Martínez Morales por su orientación para la realización del estudio estadístico efectuado en esta tesis.

Agradezco a los matemáticos Margarita Reyes Sierra, Mario Villalobos Arias y Adriana López Lara por toda su ayuda y amistad.

Gracias a la Dra. Katya Rodríguez, al Dr. Oscar Olmedo, al Dr. Arturo Díaz y al Dr. Carlos Mariano por sus valiosos comentarios acerca de este trabajo.

Agradezco a mis hermanos Alma Lilia y Carlos por su amor. Gracias también a mis primos Elida y Francisco y mi tía Elvia.

Gracias a Gregorio Toscano Pulido por su ayuda y amistad.

Gracias a mis compañeros y amigos Margarita Reyes, Efrén Mezura, Ricardo Landa, Abdiel Cáceres y Manuel Gómez por su amistad.

Índice general

Índice de figuras

Índice de cuadros

Abstract

Artificial immune systems are a relatively new research area that can be classified within the group of biologically inspired computational systems, such as neural networks and evolutionary computation, among others.

Artificial immune systems are adaptive computational systems which are inspired on different mechanisms from biological immune systems (specially from mammals) with the aim of solving complex engineering problems in which researchers of this area have found successful results.

In this dissertation, we show how an artificial immune system can be used to solve two important problems within numerical optimization.

The first problem that we tackle is the handling of constraints in genetic algorithms used for global optimization. The proposed artificial immune system is based on a mechanism through which the immune system must "learn" the antibodies appropriate according to the type of antigen that it faces. This learning process is conducted by a genetic algorithm which is on charge of handling the constraints of the problem. Furthermore, a parallel version of this algorithm was also developed. Both versions of the algorithm found competitive results with respect to the state-of-the-art in the area.

The second problem that we address is the optimization of multiple objective functions (called multiobjective optimization). The proposed algorithm is based on the clonal selection principle of the immune system and constitutes the first proposal of this type. The algorithm is capable of reaching the true Pareto front of problems with different degrees of difficulty (with and without constraints) achieving a good distribution of solutions along such front.

Resumen

El sistema inmune artificial es un área relativamente nueva que podemos clasificar dentro del grupo de los sistemas computacionales con inspiración biológica, como son las redes neuronales y la computación evolutiva, entre otros.

Los sistemas inmunes artificiales son sistemas computacionales adaptativos cuya inspiración está basada en los diferentes mecanismos del sistema inmune biológico, especialmente de los mamíferos, con la finalidad de solucionar problemas de ingeniería complejos, en los que los investigadores del área han mostrado resultados exitosos.

En esta tesis presentamos un sistema inmune artificial para solucionar dos problemas importantes dentro del área de optimización numérica.

El primer problema que atacamos es el manejo de restricciones en algoritmos genéticos usados para optimización global. El sistema inmune artificial propuesto está basado en el mecanismo mediante el cual el sistema inmune debe "aprender" los anticuerpos adecuados según el tipo de antígeno que se presente. Este proceso de aprendizaje es conducido por un algoritmo genético el cual es el encargado de manejar las restricciones del problema. Además, se realizó una versión paralela de este algoritmo. Ambas versiones del algoritmo mostraron resultados competitivos con respecto al estado del arte del área.

El segundo problema que se atacó, es el de optimización multiobjetivo. El algoritmo propuesto se basa en el principio de selección clonal del sistema inmune y constituye la primera propuesta de este tipo. El algoritmo es capaz de alcanzar el verdadero frente de Pareto de problemas con distintos grados de complejidad (con y sin restricciones) logrando una buena distribución de las soluciones a lo largo de éste. Los resultados obtenidos son competitivos con el estado del arte en el área.

Dado que el sistema inmune artificial es una heurística, consideramos importante efectuar un estudio teórico sobre la convergencia del algoritmo. Es por ello que en esta tesis se efectúa una demostración matemática de la convergencia del algoritmo multiobjetivo que se propuso.

Capítulo 1

Introducción

1.1. Motivación

Desde sus inicios, los seres humanos se han inspirado en diferentes fenómenos de la naturaleza para solucionar los problemas que se les presentan.

Uno de los sistemas biológicos que ha recibido gran atención en los últimos años es el sistema inmune debido, entre otras causas, a las interesantes características que presenta, tales como aprendizaje, memoria, reconocimiento de patrones y robustez, las cuales son el producto de interacciones locales sencillas.

En la última década se ha desarrollado notablemente un área que se inspira en el sistema inmune de los mamíferos para desarrollar modelos computacionales adaptativos, cuya finalidad es solucionar problemas complejos de ingeniería y aprendizaje de máquina como son la robótica, sistemas clasificadores, planeación de horarios, seguridad en sistemas de cómputo y redes, entre otros. Los investigadores del área han reportado soluciones exitosas en dichas aplicaciones.

En esta tesis exploramos la posibilidad de usar un sistema inmune artificial para solucionar problemas de optimización numérica no lineal.

Los algoritmos genéticos han demostrado ser heurísticas muy eficientes en la solución de problemas de optimización. Sin embargo, tienen algunas desventajas; una de ellas es su incapacidad para manejar restricciones de forma natural, lo que hace necesario incorporarles algún mecanismo adi-

cional con tal finalidad. La importancia de manejar restricciones es debida, entre otras causas, a que la mayoría de los problemas de optimización de mundo real las posee y aunque se ha desarrollado un número importante de algoritmos para manejarlas, aún existe la necesidad de técnicas alternativas.

En la primera parte de esta tesis nos enfocamos sobre este problema incorporando un mecanismo basado en un sistema inmune artificial para manejar restricciones en algoritmos genéticos para solucionar problemas de optimación global.

Otro aspecto interesante de muchos de los problemas de optimización que se presentan en aplicaciones del mundo real tales como la ingeniería y las ciencias en general, es la necesidad de manipular varios objetivos a la vez, mismos que se encuentran en conflicto. En el área de la Investigación de Operaciones existen numerosas técnicas para solucionar problemas de optimización multiobjetivo, sin embargo, debido a que muchas de ellas tienen una complejidad muy alta, es necesario contar con algoritmos alternativos que superen esta desventaja.

En la segunda parte de la tesis proponemos un algoritmo basado en un mecanismo observado en el sistema inmune llamado el principio de selección clonal para solucionar problemas de optimización multiobjetivo.

1.2. Objetivos

El objetivo principal de este trabajo es explorar la factibilidad de utilizar emulaciones del sistema inmune para solucionar problemas de optimización tanto global como multiobjetivo, teniendo en cuenta aspectos que son muy importantes dentro de ella, tales como: el manejo de restricciones de todo tipo (lineales, no lineales, igualdad y desigualdad), mantenimiento de la diversidad de las soluciones codificadas en la población y el costo computacional implicado, el cual se busca que no sea excesivamente elevado.

Los objetivos particulares de esta tesis los resumimos de la siguiente manera:

- Desarrollar mejoras al algoritmo del sistema inmune para manejo de restricciones en algoritmos genéticos y validarlo.

- Desarrollar un nuevo algoritmo de un sistema inmune artificial para optimización con múltiples objetivos que sea capaz de manejar restricciones. Experimentar con diferentes funciones de prueba y comparar contra otros algoritmos existentes utilizando métricas que proporcionen información sobre las soluciones encontradas.

- Realizar un estudio formal de algún aspecto de uno de los algoritmos propuestos.

1.3. Alcance y contribuciones

Las principales contribuciones de esta tesis son las siguientes:

- Se realizaron mejoras al algoritmo de un sistema inmune artificial para manejar restricciones en algoritmos genéticos.

- Los resultados obtenidos por nuestro algoritmo del punto anterior superaron a los encontrados por la versión original en la que éste se inspiró.

- Se desarrolló una versión paralela de dicho algoritmo, utilizando un esquema de memoria distribuida llamado de grano grueso. En general, los resultados obtenidos por esta versión paralela tienen aproximaciones más cercanas al valor óptimo que las de la versión secuencial.

- Ambas versiones del algoritmo se compararon contra técnicas representativas del estado del arte de optimización con computación evolutiva, concluyendo que nuestro algoritmo ésta a la altura de éstos.

- Se efectuó un análisis de varianza con la finalidad de conocer la sensibilidad del algoritmo a sus parámetros. Concluimos que el porcentaje de mutación y tipo de cruza son los que tienen mayor efecto. Además, se realizaron sugerencias acerca de los valores que éstos deben tomar.

- Se desarrolló la primera propuesta de un sistema inmune artificial para optimización con múltiples objetivos capaz de manejar restricciones. Este algoritmo está basado en el principio de selección clonal.

- Se compararon los resultados obtenidos por nuestro algoritmo multiobjetivo contra 3 técnicas representativas del estado del arte del área utilizando métricas que proporcionan información cuantitativa.
- Se llevó a cabo un estudio formal sobre la convergencia del algoritmo del sistema inmune artificial para optimización multiobjetivo. En él demostramos matemáticameente que el algoritmo es capaz de converger al frente de Pareto cuando se cumplen ciertas condiciones determinadas.

1.4. Estructura general del documento

La organización del resto del documento es la siguiente:
En el capítulo 2 se explican los conceptos generales de los algoritmos genéticos y el sistema inmune tanto biológico como artificial.

El capítulo 3 está dedicado a la definición del problema general que deseamos solucionar, tanto de optimización mono-objetivo como multiobjetivo.

En el capítulo 4 se realiza una descripción de la propuesta para solucionar el problema de manejo de restricciones en algoritmos genéticos usando un sistema inmune artificial.

En el capítulo 5 se presenta el algoritmo propuesto para optimización multiobjetivo, los resultados y comparaciones efectuadas.

El capítulo 6 está dedicado a un estudio formal acerca de la convergencia del algoritmo del sistema inmune artificial multiobjetivo.

Capítulo 2

Antecedentes

2.1. Introduccción

Este capítulo está dedicado a proporcionar algunos conceptos básicos de la computación evolutiva que se consideran importantes para establecer claramente el problema que deseamos solucionar. Nos enfocaremos en uno de sus paradigmas más conocidos que son los algoritmos genéticos.

Además, abordaremos el tema del sistema inmune artificial, que es la herramienta con la que pretendemos solucionar dichos problemas, por lo que, la segunda parte de este capítulo está compuesta de una introducción al área del sistema inmune artificial, así como de sus bases biológicas más interesantes desde esta perspectiva.

2.2. Algoritmos Evolutivos

Los *algoritmos evolutivos* son sistemas computacionales diseñados para solucionar problemas altamente no lineales, tomando ideas del proceso evolutivo y de adaptación de la naturaleza, las cuales se agrupan bajo la denominada teoría Neo-Darwiniana de la evolución [41].

Esta teoría Neo-Darwiniana está compuesta por las ideas clave de tres importantes científicos:

1. El principio de supervivencia de los individuos más aptos del naturalista inglés Charles Darwin [25],

2. La teoría de la herencia propuesta por el monje austriaco Gregor Mendel, y

3. El evolucionismo de August Weismann.

Gran parte de la terminología utilizada en algoritmos evolutivos se ha tomada prestada de los conceptos biológicos en los que se inspira (por ejemplo, cromosoma, alelos, etc.).

Una característica importante de los algoritmos evolutivos es que son poblacionales, lo que significa que manipulan simultáneamente un conjunto de soluciones potenciales al problema y no una solución única, como suelen hacer las técnicas clásicas de optimización. El uso de una población de soluciones potenciales a un problema hace a los algoritmos evolutivos menos susceptibles de quedar atrapados en óptimos locales.

De manera general, los algoritmos evolutivos parten de una población inicial (normalmente generada aleatoriamente) a la cual se le aplica un operador de *selección* para obtener a aquellos individuos (o soluciones) que resulten más aptos con respecto a una medida de aptitud; a este conjunto de individuos seleccionados se les llama padres, los cuales son recombinados entre sí para obtener una población de hijos (*cruza o recombinación*). Los hijos sufren un proceso de *mutación*, y reemplazan a los padres formando la siguiente generación. Este proceso se repite generalmente un número predeterminado de iteraciones o generaciones [91].

Los operadores genéticos básicos son los siguientes:

- **Selección**: Consiste de un mecanismo (probabilístico o determinista) que permite elegir a los individuos que fungirán como padres de la siguiente

  generación.

- **Cruza o recombinación**: Se refiere al intercambio de información (material genético) entre dos padres que han sido seleccionados con base en su aptitud.

- **Mutación**: Consiste en hacer pequeñas perturbaciones a los individuos recién creados para la nueva población con la finalidad de explorar zonas del espacio de búsqueda que la cruza pudiera no alcanzar.

Existen 3 grandes paradigmas dentro de los algoritmos evolutivos, los cuales fueron desarrollados de manera independiente a lo largo de la historia del área:

- Programación evolutiva [42]
- Estrategias evolutivas [122, 123]
- Algoritmos genéticos [57]

La filosofía de los 3 paradigmas es esencialmente la misma, sus diferencias estriban en la representación que adoptan o en la importancia que se da a los operadores que se aplican en cada una de ellas.

De estas tres áreas, la que ha adquirido mayor importancia en los últimos años son los *algoritmos genéticos* [48].

De manera general, podemos afirmar que para diseñar un algoritmo evolutivo

es necesario establecer al menos los siguientes componentes:

- Un esquema de representación de soluciones potenciales al problema
- Una medida de aptitud (llamada *fitness* en inglés), que indica qué tan bueno es un individuo con respecto al objetivo que deseamos alcanzar.
- Un conjunto de operadores genéticos básicos: selección, cruza y mutación

Aunque los algoritmos evolutivos pueden ser usados para tareas diversas, su área de aplicación más popular ha sido, sin duda, la optimización [48].

En este capítulo nos concentraremos en los algoritmos genéticos usados para solucionar problemas de optimización global.

2.2.1. Algoritmos genéticos

Los algoritmos genéticos (AG) son algoritmos evolutivos que se han utilizado principalmente para solucionar problemas de optimización.
Las características generales de los algoritmos genéticos son las siguientes:

1. Suelen usar una codificación binaria de las soluciones potenciales al problema en cadenas binarias de longitud fija (también llamadas cromosomas).

2. El operador de cruza tiene mayor importancia que el de mutación.

3. Suelen adoptar métodos probabilísticos de selección (por ejemplo, los esquemas de selección proporcional [49]).

Se han desarrollado una gran cantidad de variaciones de los algoritmos genéticos, y diferentes operadores [3]. Sin embargo, los algoritmos genéticos simples (también llamados canónicos) suelen tener las siguientes características: Representación binaria, selección mediante ruleta (proporcional al valor de aptitud), cruza de un punto y mutación uniforme.

A continuación mostraremos la estructura general para un algoritmo genético simple:

1. Se genera la población inicial de manera aleatoria.

2. Se calcula el valor de aptitud (*fitness*) para cada individuo de la población.

3. Se aplica selección basada en aptitud, para obtener la población de padres.

4. Se aplica cruza (o recombinación) de los padres seleccionados (por parejas) para obtener una población de hijos (cada pareja de padres genera dos hijos).

5. Se aplica mutación uniforme a los hijos.

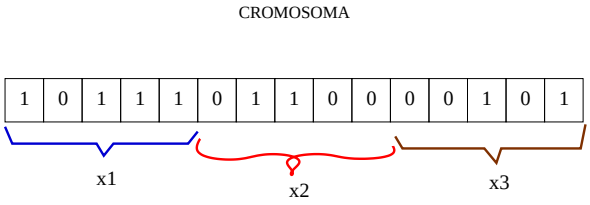

Figura 2.1: Ejemplo de representación binaria para un problema de 3 variables.

6. Los hijos forman la nueva generación, desechando por completo a la población de la generación anterior, excepto por el mejor individuo, el cual pasa intacto a la siguiente generación (a esto se le conoce como elitismo).

7. Se repite desde el paso 2 un número predeterminado de veces o hasta alcanzar convergencia.

A continuación mencionaremos cómo se efectúan cada uno de estos pasos.

En la **representación binaria**, las variables del problema son codificadas en una cadena binaria, como se muestra en la figura 2.1. Cada variable se discretiza fijando una precisión deseada, y posteriormente los valores se tratan como si fueran enteros. Por ejemplo, si deseamos representar a la variable x, cuyos valores varían entre -1.5 y 2.0 y aplicamos una precisión de 3 dígitos, el tamaño de la cadena para representar la variable x se calcula con:

$$\text{Tamaño} = \lceil \log_2[(l_{\sup} - l_{\inf}) * 10^{\text{precision}}] \rceil$$

donde $l_{\inf}$ y $l_{\sup}$ son los límites inferior y superior respectivamente. Para este ejemplo en particular obtenemos que la longitud de la cadena es igual a 12 bits.

La medida de **aptitud** mencionada en el paso 2, se refiere a qué tan bueno es un individuo con respecto a la función objetivo del problema que estamos solucionando. Por ejemplo, si estamos maximizando una función, la mejor aptitud la tendrán los individuos que, al evaluar sus variables en

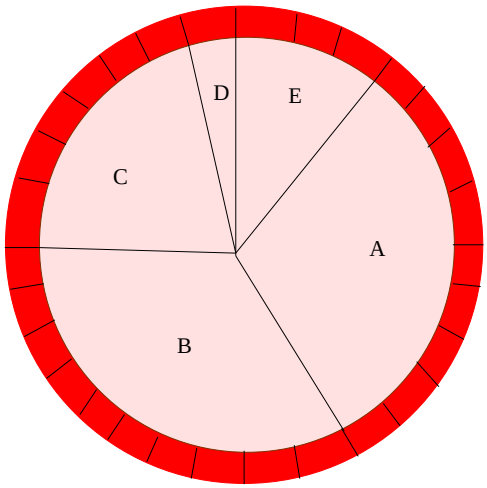

Figura 2.2: Selección proporcional por medio de ruleta. En este ejemplo de 5 individuos, los marcados con A y B son los más aptos y tienen una mayor probabilidad de ser seleccionados. Sin embargo, el individuo marcado con D podría también ser seleccionado (aunque con una probabilidad muy baja).

la función objetivo, obtengan los valores más altos, y viceversa.

La **selección mediante ruleta** es una técnica proporcional, lo que significa que la probabilidad de un individuo de ser seleccionado está en proporción a su aptitud. En esta selección se simula que a cada individuo se le asigna una porción de una ruleta, de forma que al girarla, los individuos con una porción mayor tienen mayor probabilidad de ser seleccionados. Sin embargo, debido a su naturaleza probabilística, este esquema permite que aún los individuos menos aptos tengan alguna probabilidad de ser seleccionados para reproducirse. Ver figura 2.2.

En la **cruza de un punto**, dos padres se cruzan para generar dos hijos, y cada uno de ellos aporta un segmento de su cromosoma. El punto a partir del cual se realiza la cruza se define aleatoriamente. Cada hijo recibe un segmento de cada uno de los padres. La figura 2.3 muestra este mecanismo.

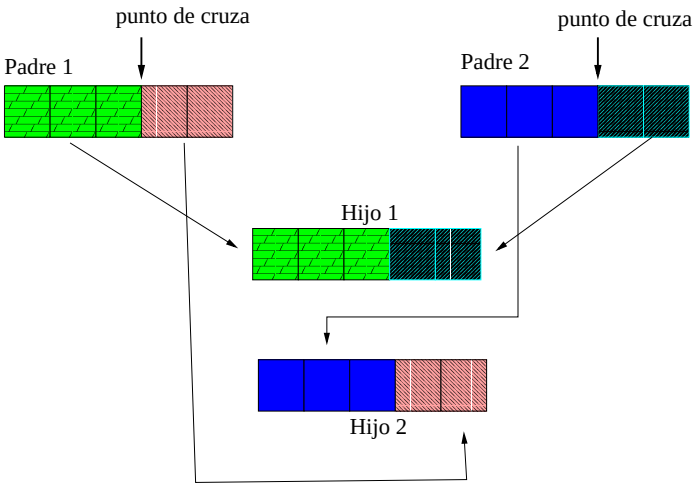

Figura 2.3: Representación de la cruza de un punto. Dos padres generan dos hijos.

La **mutación uniforme** consiste en cambiar el valor de cada bit de la cadena cromosómica con una probabilidad determinada, (se recomienda que sea pequeña, entre 0.01 y 0.001). La finalidad de la mutación es explorar regiones del espacio de búsqueda que el operador de cruza no pudiera alcanzar.

Con la finalidad de superar algunas de las desventajas que poseen los operadores genéticos utilizados en el algoritmo genético simple, se han desarrollado una serie de operadores genéticos alternativos que buscan hacer más eficiente el desempeño de los algoritmos. A continuación listaremos algunos de los más conocidos.
Tipo de representación:

- Binaria
- Real
- Entera
- Punto flotante

Operadores de selección:

- Ruleta [48, 57]
- Sobrante estocástico con y sin reemplazo [8]
- Universal estocástica [4]
- Torneo (binario o mayor), probabilístico o determinista [9]

Operadores de cruza:

- Cruza de un punto o simple [57]
- Cruza de dos puntos [48]
- Cruza uniforme [130]

Cada uno de los operadores genéticos presenta diferentes ventajas y desventajas por lo que no podemos afirmar que alguno de ellos sea más eficiente que los demás. En general, su desempeño suele depender del tipo de problema a resolverse.

2.2.2. Algoritmos genéticos paralelos

A continuación se muestra un panorama general de los algoritmos genéticos paralelos, centrando la discusión en los llamados AG's multipoblacionales o de grano grueso.

El objetivo de los algoritmos genéticos paralelos es dividir un problema en varios subproblemas y resolver éstos de manera simultánea en diferentes procesadores. De esta manera se busca disminuir el tiempo de procesamiento de los algoritmos.

En general, cuando se aplica paralelización a un algoritmo cualquiera, el comportamiento de éste es el mismo que el del algoritmo secuencial. Sin embargo, ese no es necesariamente el caso de la paralelización de algoritmos genéticos [11].

Dentro de la literatura especializada podemos encontrar que existen cuatro estrategias principales para paralelizar un AG, pero sólo una de ellas cumple con la característica de que la versión serial del algoritmo sea igual

a la paralela, es decir, que arrojan exactamente la misma salida, mientras que las otras tres estrategias de paralelización producen, en general, diferentes salidas a la versión secuencial del AG. Las cuatro estrategias de paralelización del AG son las siguientes [11]:

- AG de población única o global o maestro-esclavo
- AG de múltiples poblaciones o de grano grueso
- AG de grano fino o AG celular
- AG maestro-esclavo jerárquico

El AG global o de población única tiene el mismo comportamiento que su correspondiente versión secuencial. En éste únicamente se paraleliza la evaluación de la función de aptitud entre los procesadores, mientras que los procesos de selección, cruza y mutación siguen siendo centralizados. De manera que nos referimos al mismo algoritmo pero con la carga de trabajo es distribuida.

En el AG de múltiples poblaciones o de grano grueso, la población es dividida en múltiples subpoblaciones (llamadas demes) que son distribuidas entre los procesadores, de manera que cada procesador aplica los operadores genéticos sobre la subpoblación que le fue asignada. Este tipo de estrategia implica que el algoritmo secuencial es diferente al paralelo y, aún cuando usaran los mismos números aleatorios, obtendrían salidas diferentes.

El AG de grano fino o AG celular consiste en una población con una estructura espacial simple (usualmente una rejilla bidimensional) en la cual hay un individuo por cada punto de la rejilla. Idealmente se asigna un punto por procesador. La selección y la cruza están restringidas a un pequeño vecindario.

El último tipo de AG paralelo es llamado híbrido jerárquico y combina el esquema de múltiples demes con un AG global o uno de grano fino. Estos algoritmos son llamados así porque pueden ser vistos, a alto nivel como un AG de múltiples poblaciones, y a bajo nivel, son algoritmos con el esquema de población única (ya sea global o de grano fino). Este tipo de estrategia pretende obtener las ventajas tanto de la arquitectura del AG global como de la de grano fino.

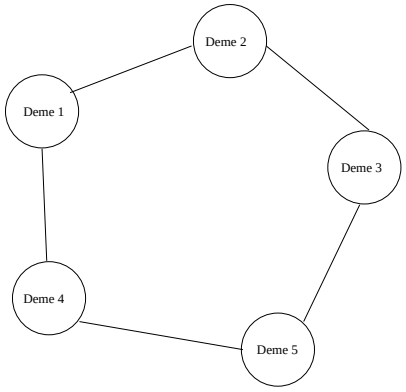

Figura 2.4: Representación de un AG de múltiples poblaciones. Cada círculo representa un AG que tiene comunicación ocasional con sus vecinos. Las rectas indican la topología de interconexión entre las poblaciones.

Para este trabajo se utilizó una estrategia de múltiples poblaciones o grano grueso. Es por

ello que a continuación mencionaremos más detalladamente la manera en que se aplica esta estrategia de paralelización.

Algoritmo genético de múltiples poblaciones o grano grueso

En este tipo de AG paralelo se tiene, como su nombre lo indica, más de una población. A cada una de esas subpoblaciones se le conoce con el nombre de *deme*. Generalmente los demes se distribuyen entre los procesadores. Cada deme evoluciona de manera independiente, pero intercambian individuos a cada cierta cantidad de generaciones. A este intercambio de individuos se le llama *migración* y a la cantidad de generaciones transcurrida entre cada migración se le conoce como *época*. Ver figura 2.4.

La migración es controlada por varios parámetros que debe definir el usuario. Debe decidirse la cantidad de demes y el tamaño de cada uno de ellos, así como la manera en que se interconectan entre sí.

Para el proceso de migración es necesario definir cuántos y cuáles serán los individuos que van a migrar; asímismo, debe definirse cuál es el deme que los va a recibir, determinar quiénes serán reemplazados en dicho deme,

el tamaño de la época y hacia dónde se ha de realizar la migración (o sea la topología de la interconexión).

Debido a que el tamaño de los demes es menor al de la población global, se espera que éstos tengan una convergencia más rápida. Sin embargo, debe tenerse en cuenta que la calidad de la solución obtenida en estos demes podría no ser tan buena. Aunque ésta es la estrategia más utilizada, observamos que requiere la definición de una gran cantidad de parámetros. Generalmente el desempeño del algoritmo dependerá de la selección de tales parámetros.

La aceleración lograda por un algoritmo paralelo es la razón que existe entre el mejor tiempo logrado por la versión secuencial, dividido entre el tiempo de procesamiento requerido por la versión paralela, de la siguiente manera:

$$\text{Aceleración} = \frac{\text{Tiempo}-\text{serial}}{\text{Tiempo}-\text{paralelo}}$$

Este valor indica qué tanto se ganó en tiempo al paralelizar el algoritmo.

Es importante tener en mente, que en nuestro caso, por tratarse de un AG de grano grueso, al compararlo con su versión secuencial, estamos comparando en realidad dos algoritmos diferentes [11].

2.3. Sistema inmune biológico

2.3.1. Introducción

Las soluciones que se proponen en este trabajo a los dos principales problemas planteados (manejo de restricciones en algoritmos genéticos (AG's) y optimización multiobjetivo) están inspirados por algún mecanismo que presenta el sistema inmune natural; es por ello que esta sección está dedicada a proporcionar una descripción del funcionamiento y principales componentes del sistema inmune biológico. Es muy importante aclarar que la descripción aquí presentada está muy simplificada con la finalidad de hacerla comprensible, pues la complejidad del sistema inmune es enorme.

Aunque algunos de los modelos computacionales que se han desarrolla-

do en el área conocida como "sistema inmune artificial" son biológicamente plausibles y han aportado cierta ayuda al entendimiento de los procesos naturales, el objetivo principal dentro del área no es ese, sino el de crear soluciones a problemas computacionales complejos inspirados en la naturaleza, aún cuando desde el punto de vista biológico no sean apropiados. Por esta razón, y siguiendo los objetivos de este trabajo, resulta suficiente con tener una visión global del funcionamiento del sistema inmune.

2.3.2. Características generales

El principal objetivo del sistema inmune es proteger al organismo de la presencia de agentes infecciosos, así como de reparar las células dañadas o eliminarlas cuando sea necesario. De aquí, es fácil deducir la importancia que el sistema inmune tiene para nuestra vida.

El sistema inmune es capaz de detectar la presencia de agentes externos al organismo. Tal detección desencadena una serie de procesos que llevan a la neutralización y eliminación de los invasores de manera adecuada. Una de las características más importantes del sistema inmune es su capacidad de memoria, pues sabemos que a cada encuentro con el mismo tipo de invasor (o uno similar) la respuesta del sistema inmune es cada vez mejor.

De manera general, a cualquier organismo dañino ajeno al sistema lo llamaremos *antígeno*; ejemplos de éstos son las bacterias, los virus y los parásitos.

El sistema inmune trabaja por capas o niveles de defensa que se ilustran en la figura 2.5. A continuación se mencionará a qué se refieren cada uno de ellos.

- **Barrera física**. El primer nivel de defensa está formado por las barreras físicas como la piel y las mucosas. Ésta no es específica, lo que significa que es igual (fijo) para cualquier tipo de antígeno que se presente.

- **Condiciones fisiológicas**. Se refiere a las condiciones fisiológicas que son desfavorables para la sobrevivencia de los antígenos, tales como la temperatura y el ph del órgano en que se encuentra el antígeno.

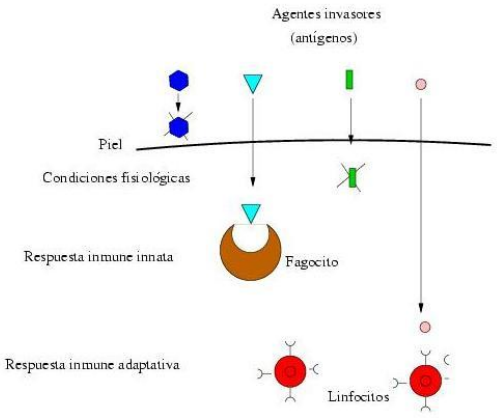

Figura 2.5: Niveles de defensa del sistema inmune

- **Respuesta inmune innata**. La respuesta innata del sistema inmune no tiene memoria. Los principales actores de este tipo de respuesta son las células llamadas fagocitos, específicamente los macrófagos, los cuales fagocitan (o ingieren) a los antígenos para hacer más fácil su eliminación. Una de sus principales funciones es activar al siguiente nivel de defensa (la respuesta adaptativa o aprendida).

- **Respuesta inmune adaptativa o aprendida**. Esta respuesta está representada principalmente por las células llamadas *linfocitos*. Sus características más interesantes son su capacidad de memoria, aprendizaje y especifidad. En la siguiente sección se hablará de manera más detallada de dicha respuesta adaptativa.

2.3.3. Respuesta inmune adaptativa o aprendida

Esta es la respuesta más interesante desde la perspectiva del sistema inmune artificial. Su principal característica es que posee aprendizaje y

memoria [1], además de ser específica al tipo de antígeno que se presente.

Las células representantes de esta respuesta inmune son las células blancas llamadas *linfocitos* que se encuentran circulando en el torrente sanguíneo; existen dos tipos de linfocitos: **B** y **T**. Los linfocitos B maduran en la médula ósea (*bone marrow*, en inglés). Este proceso de maduración de los linfocitos consiste principalmente en hacer un reclutamiento de aquellos que sean capaces de reaccionar ante agentes externos, y en cambio no lo hagan ante las células propias del sistema. De manera que, los linfocitos son expuestos ante un grupo de células propias del organismo, y se permite sobrevivir a aquellos que no presenten ninguna reacción, y por otro lado, se elimina a los que presenten alguna reacción (a este mecanismo se le conoce con el nombre de *selección negativa*). De esta manera, las células maduras serán capaces de reconocer y reaccionar únicamente ante agentes externos, evitando así posibles autoataques.

Los detectores del sistema inmune son capaces de reconocer una variedad impresionante de diferentes antígenos. El genoma humano posee una variedad de 10^6 detectores, mientras que la variedad de antígenos que pueden reconocer es del orden de 10^{16}. La diversidad que presentan los detectores del sistema inmune, se debe principalmente a que son creados mediante la combinación de segmentos de ADN que se seleccionan de manera aleatoria, y posteriormente al proceso de mutación al que son sometidos.

Son muchas las células que intervienen en la activación y ejecución de la respuesta adaptativa. Sin embargo, por sencillez en esta sección nos concentraremos únicamente en algunos de los mecanismos seguidos por los *linfocitos B* y las moléculas que secretan llamadas *anticuerpos*. Ver figura 2.6.

Ante la presencia de un antígeno, únicamente los linfocitos con anticuerpos que posean la forma específica al antígeno, es decir los más afines, serán estimulados para proceder a la eliminación del antígeno. Ver figura 2.7.

Los anticuerpos poseen en su estructura una región específica que es capaz de reconocer a los antígenos, conocida con el nombre de *paratope*; a

[1] Las definiciones se pueden consultar en el Apéndice A de este documento

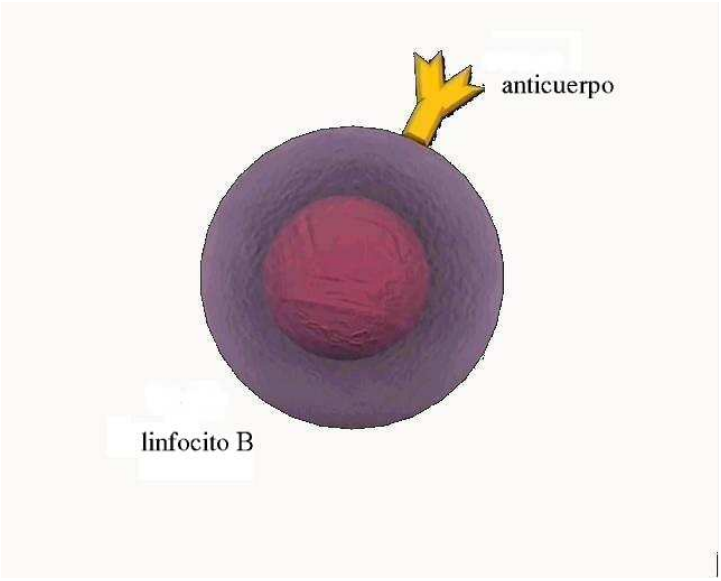

Figura 2.6: Anticuerpos expresados en un linfocito B

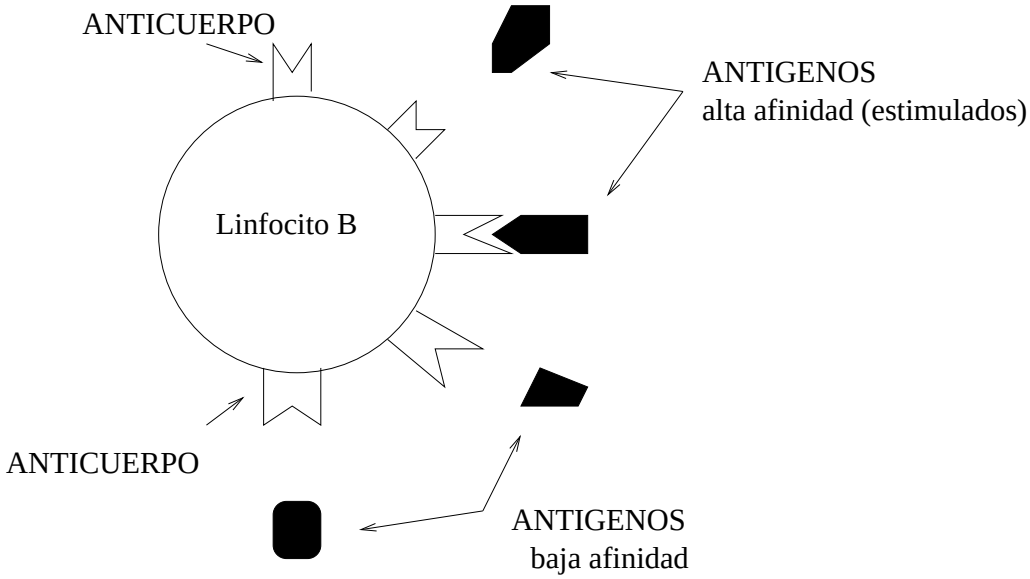

Figura 2.7: Anticuerpos estimulados por un antígeno determinado.

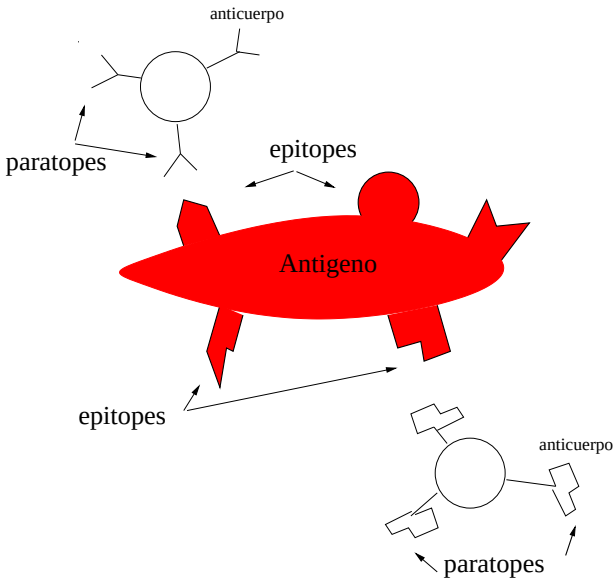

Figura 2.8: Relación paratope-epitope

su vez, la parte del antígeno que puede ser reconocida por los anticuerpos es llamada *epitope*. Cada linfocito B posee en su superficie anticuerpos con el mismo tipo de paratope. Por otro lado, los antígenos poseen diferentes tipos de epitopes, de manera que, un mismo antígeno puede activar varios linfocitos a la vez, o sea que puede ser reconocido por diferentes linfocitos. Ver figura 2.8.
El paratope de un anticuerpo, puede ser estimulado por otro anticuerpo, y no solamente por la presencia de un antígeno. La **teoría de red inmune** explica las interrelaciones que existen entre anticuerpos, aún en la ausencia de antígenos [64, 101].

Principio de selección clonal

Los linfocitos que tienen anticuerpos con la afinidad adecuada para el antígeno (o una similar) son estimulados y sometidos a un proceso de reproducción por clonación, es decir, se crean múltiples copias de ellos. Estos nuevos clones sufren un proceso de mutación a gran escala (llamada hi-

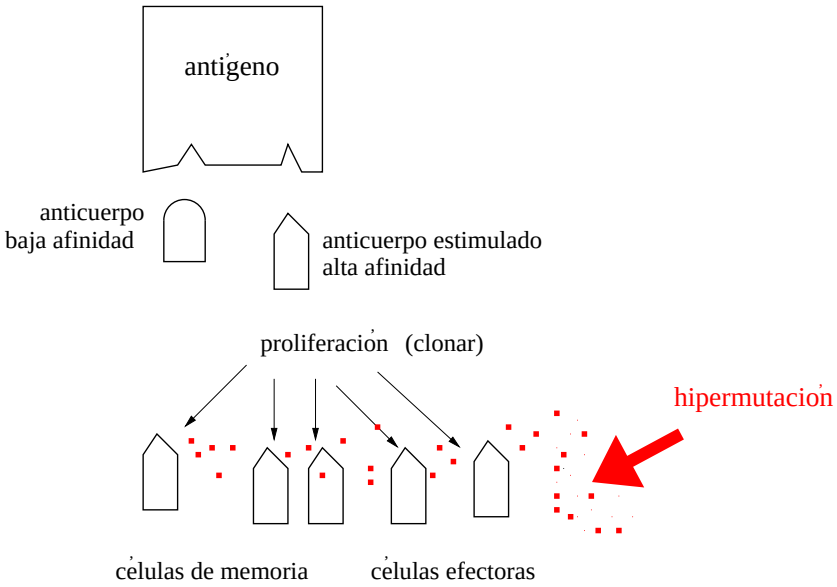

Figura 2.9: Principio de selección clonal.

permutación) con lo que se incrementa considerablemente la variedad de su repertorio. Esto permite que algunos de ellos incrementen aún más su afinidad hacia el antígeno. Los clones que ahora resulten ser más afines al antígeno son denominados células efectoras y se adhieren al antígeno procediendo a su neutralización y eliminación.

El porcentaje de mutación de los clones es inversamente proporcional a su afinidad, de manera que, a mayor afinidad, la mutación es menor. En cambio, si el clon es de menor afinidad, entonces el porcentaje de mutación se incrementa [71]. Este mecanismo se representa en la figura 2.9.

Una vez que los antígenos han sido eliminados del organismo, existirá un excedente de células y el sistema debe regresar a sus niveles normales, es por ello que tales células son eliminadas (autoregulación); sin embargo, algunas se conservan circulando a través del organismo como células de memoria. Así la próxima vez que el mismo tipo de antígeno sea encontrado (o uno similar), entonces el sistema inmune utilizará sus células de memoria, y la respuesta será cada vez más rápida y eficiente. A ésta se le conoce con el nombre de *respuesta secundaria*.

A este proceso de estimulación y activación de los anticuerpos más afines, clonación y mutación se le conoce con el nombre de *principio de selección clonal.*

2.4. Sistema inmune artificial

Análogamente a la computación evolutiva, el área llamada "sistema inmune artificial" se refiere al diseño de sistemas computacionales basados en un mecanismo observado en la naturaleza, con la finalidad de solucionar problemas complejos.

En este caso la fuente de inspiración es nuestro sistema inmune, cuyo funcionamiento fue descrito brevemente en la sección anterior. Ahora intentaremos explicar cómo se utilizan esos mecanismos del sistema inmune en la solución de problemas.

Esta es un área relativamente nueva. Los primeros trabajos que podemos encontrar sobre este tema datan de mediados de la década de los 80's, aunque la importancia que ha adquirido el área ha crecido de manera importante en los últimos cinco años.

El sistema inmune (SI) puede ser visto como un poderoso sistema de procesamiento de información, cuyas características más importantes se muestran a continuación [28]:

- **Memoria**. El SI es capaz de recordar antígenos que se han presentado en el pasado.
- **Aprendizaje**. La respuesta del SI, ante repetidas apariciones de un antígeno es cada vez más rápida y eficiente, por eso se dice que posee aprendizaje.
- **Robusto y tolerante a fallas**. Esto significa que el sistema es capaz de responder adecuadamente aún en el caso de que falten algunos de sus elementos. Además, tiene la capacidad de recuperarse de errores.
- **Diversidad**. El SI es capaz de generar una gran diversidad y reconocer casi cualquier tipo de antígeno que se le presente.

- **Distribuido y descentralizado**. La detección del SI es eminentemente distribuida; las células inmunes se encuentran circulando por todo el organismo. Las interacciones entre los componentes del SI son locales y no existe un procesador central.

Es quizá gracias a éstas y otras características que el área del sistema inmune artificial se ha expandido rápidamente.

2.4.1. Definición de un sistema inmune artificial

Para definir lo que es un sistema inmune artificial recurriremos a una de las definiciones más completa propuesta por Leandro Nunes de Castro y Jonathan Timmis [34]:

> "Los sistemas inmunes artificiales son sistemas adaptativos, inspirados por la teoría inmunológica, funciones, principios y modelos inmunológicos observados, los cuales son aplicados a la solución de problemas."

Aunque la cantidad de modelos y aplicaciones del sistema inmune va en ascenso, no existe un esquema general de cuáles son los elementos esenciales que un sistema inmune artificial debe poseer.

Nunes De Castro y Timmis sugieren en su libro [33] utilizar un esquema general de sistemas computacionales con inspiración biológica, tal como los algoritmos evolutivos o las redes neuronales. Este esquema consta de tres partes principales que deben definirse, y que a continuación se muestran:

- Representación de los componentes del sistema.
- Un conjunto de mecanismos para evaluar las interacciones de los individuos con el ambiente y entre ellos.
- Un proceso de adaptación que gobierne las dinámicas del sistema, es decir, el algoritmo en sí.

A diferencia de otras técnicas bio-inspiradas (por ejemplo los algoritmos genéticos), el sistema inmune artificial no tiene un algoritmo general único.

Existen varios modelos que se basan en diferentes mecanismos del sistema inmune biológico, como el proceso de creación de defensores del sistema, el reconocimiento de invasores, o en cualquier otro mecanismo observado. Esto implica que, obviamente, se tienen algoritmos muy diversos.

A continuación mostramos algunos ejemplos de cómo un sistema inmune artificial puede adaptarse al esquema general de los sistemas con inspiración biológica antes mencionado:

- **Representación de los componentes del sistema**: Es necesario definir una representación para crear los modelos abstractos del sistema inmune, tales como los linfocitos, anticuerpos, fagocitos, etcétera. Entre los tipos de representación más comunes encontramos la binaria, la real y la simbólica. En general, la representación depende del problema con que se está tratando.

- **Un conjunto de mecanismos para evaluar las interacciones de los individuos con el ambiente y entre ellos**: Deben definirse funciones de afinidad para medir la interacción entre los componentes, como la medida de afinidad antígeno-anticuerpo, interacciones entre células B, etcétera. Estas funciones dependen de la representación que se haya seleccionado. Por ejemplo, si se usa representación binaria las interacciones pueden ser distancias de Hamming, o distancias Euclidianas para el caso de representación con números reales.

- **Un proceso de adaptación que gobierne las dinámicas del sistema, es decir, el algoritmo en sí**: Por ejemplo, la estimulación y clonación de linfocitos para el caso del principio de selección clonal.

2.4.2. Modelos y algoritmos del sistema inmune artificial

Existen varios modelos del sistema inmune artificial, quizá debido a que su campo de aplicación es muy amplio y a la gran complejidad del sistema inmune biológico. De Castro y Timmis resumen los modelos en tres grandes grupos que a continuación se enumeran [34]:

- Modelos de timo y de médula ósea (*bone marrow*)
- Algoritmo de selección clonal
- Teoría de red inmune

Los **modelos de timo** son aquellos basados en el proceso del sistema inmune mediante el cual se crea la población de células detectoras.

Los **modelos de médula ósea** se refieren a la creación y maduración de los detectores del sistema inmune. Una de las características más importantes del los linfocitos maduros es su capacidad para poder distinguir entre los elementos que pertenecen al sistema de los agentes externos. La mayoría de estos modelos se basan en el principio de selección negativa y/o positiva.

Los modelos que utilizan el **algoritmo de selección clonal** emulan el proceso mediante el cual el sistema inmune, ante la presencia de un antígeno específico, estimula únicamente a aquellos linfocitos que sean más afines, para después ser clonados y mutados. Ver sección 2.3.3.

Y finalmente, los modelos que se basan en la **teoría de red inmune**, emulan las interrelaciones que existen entre anticuerpos, aún en una ausencia de antígenos, es decir, la estimulación y supresión entre anticuerpos.

Los campos de aplicación del sistema inmune artificial son muy variados. A continuación mencionaremos algunos de los principales en donde se han reportado resultados exitosos:

- Robótica [63, 69, 83, 139]
- Seguridad en sistemas de cómputo y redes [70, 44, 46, 96, 53, 74, 27, 56]
- Optimización [50, 51, 35, 85, 134]
- Reconocimiento de patrones [45, 12, 133, 30]
- Planeación de horarios [54, 55, 94, 18]

- Aprendizaje máquina [6, 29, 40, 47, 61, 62, 126]
- Control [7, 80, 97, 131]

2.5. Resumen

En este capítulo se han revisado algunos de los conceptos básicos de la computación evolutiva, haciendo principal énfasis en los algoritmos genéticos (AG). Se mostraron las características de un AG simple, tales como su estructura general, representación, selección, cruza y mutación. Además, se mostró un panorama general de las estrategias que existen para paralelizar un algoritmo genético, centrándonos en los AG's paralelos de múltiples poblaciones (también llamados de grano grueso). Aunque esta estrategia es sencilla conceptualmente, requiere de la definición de una gran cantidad de parámetros adicionales a los de un algoritmo genético secuencial. También se hizo ver que el resultado arrojado por este tipo de versión paralela del AG no es igual al de la versión secuencial.

Por otro lado, se describieron de manera muy general, los principales componentes del sistema inmune y algunos de sus mecanismos más interesantes.

El sistema inmune puede ser visto como un poderoso sistema de procesamiento de información con características muy interesantes, tales como memoria, aprendizaje y tolerancia a fallas, entre otras. El área llamada "sistema inmune artificial" toma ideas de éste, con la finalidad de solucionar de manera eficiente problemas complejos, creando modelos computacionales adaptativos.

En el siguiente capítulo se describe el estado del arte de los dos principales problemas de optimización que se abordan en esta tesis: el manejo de restricciones y la optimización multiobjetivo.

Capítulo 3

Optimización con Algoritmos Evolutivos

3.1. Introducción

Aún cuando los algoritmos evolutivos han mostrado un desempeño satisfactorio en la solución de diversos problemas de optimización [3], tienen ciertas limitantes cuando se debe lidiar con problemas del mundo real. Uno de ellos es su incapacidad para manejar restricciones de manera natural, así que es necesario incorporarles algún mecanismo adicional con tal finalidad [16]. Por otro lado, una característica inherente a los problemas de optimización multiobjetivo es que no tienen una solución única, sino que un conjunto de ellas.

Adicionalmente sabemos que, debido a ruido estocástico, la población de un algoritmo evolutivo tiende a converger a un punto único si se itera un número excesivo de veces [48]. Por lo tanto, debe buscarse la manera de mantener diversidad para que se logre encontrar la mayor cantidad de puntos posible usando una sola ejecución algorítmica.

Existe un número considerable de algoritmos evolutivos propuestos para solucionar ambos problemas (manejo de restricciones [16] y optimización multiobjetivo [90]). En este capítulo revisaremos algunos de los más importantes. Iniciaremos con el manejo de restricciones para continuar con las propuestas existentes para optimización multiobjetivo.

3.2. Optimización mono objetivo con restricciones

Los problemas de optimización del mundo real suelen tener restricciones que generalmente resultan difíciles de satisfacer. A pesar de haber sido concebidos para tareas de aprendizaje de máquina, el principal uso de los algoritmos genéticos ha sido para solucionar problemas de optimización [67, 68]. Aún cuando han demostrado ser muy eficientes, presentan un problema importante que es no poder manejar restricciones de manera natural; y más aún su incorporación en el mecanismo de selección no resulta fácil.

A continuación daremos una definición formal del primer tipo de problema que deseamos solucionar y, una visión global del estado del arte en esta área.

3.2.1. Definición

El problema que nos interesa solucionar, es el problema general de la optimización numérica global, que podemos definir de la siguiente manera:

$$\text{Encontrar } \vec{x} \text{ que optimice } f(\vec{x}) \tag{3.1}$$

sujeto a:

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, n \tag{3.2}$$

$$h_j(\vec{x}) = 0, \quad j = 1, \ldots, p \tag{3.3}$$

donde $\vec{x}$ es el vector de soluciones $\vec{x} = [x_1, x_2, \ldots, x_r]^T$, n es el número de restricciones de desigualdad y p es el número de restricciones de igualdad (en ambos casos las restricciones pueden ser lineales o no lineales).

Se les llama restricciones activas a aquellas que cumplen con la igualdad cuando alcanzan el valor óptimo de la función. Ver figura 3.1.

3.2.2. Estado del arte

Durante la última década, se ha desarrollado un numeroso grupo de técnicas para incorporar restricciones en computación evolutiva, principal-

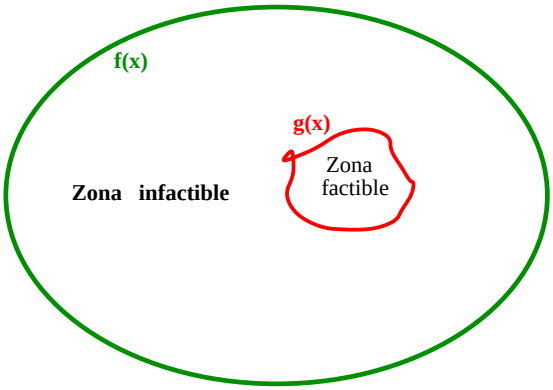

Figura 3.1: Zona factible e infactible en un problema restringido

mente en algoritmos genéticos. Estas técnicas las podemos dividir en cinco grandes grupos que son [116] [16]:

1. Técnicas que rechazan a los individuos infactibles
2. Técnicas que mantienen una población factible por medio de representaciones y operadores genéticos especiales
3. Técnicas que separan objetivos y restricciones
4. Técnicas que penalizan a los individuos infactibles
5. Híbridos

A continuación se verán algunos ejemplos de cada una de ellas:

Rechazo de individuos infactibles

A la técnica en la cual se desechan los individuos infactibles sin importar su cercanía a la zona factible, se le suele conocer con el nombre de "pena de muerte", y en general, podemos decir que es la más fácil de usar e implementar. Sin embargo, tiene fuertes desventajas, porque la búsqueda puede

estancarse en cualquier momento, sobre todo si el problema tiene regiones factibles pequeñas.

Técnicas que mantienen una población factible

Dentro del grupo de técnicas que mantienen una población factible mediante el uso de representaciones y operadores genéticos especiales tenemos varios ejemplos en la literatura especializada [32, 31]. Su rasgo distintivo es que suelen usar una representación especial para el tipo de problema que se está solucionando; éste es el caso de la propuesta de Davidor que se usa en un problema de robótica [31].

Otro ejemplo es GENOCOP propuesto por Michalewicz [89] que es capaz de manejar únicamente restricciones lineales, por lo que su aplicabilidad es muy limitada.

En este grupo también podemos clasificar la técnica propuesta por Schoenauer & Michalewicz [119] en la cual se enfatiza la exploración en el límite entre la región factible y la infactible, dado que muchos problemas de optimización no lineal tienen al menos una restricción activa (Se les llama restricciones activas a aquellas que cumplen con la igualdad cuando alcanzan el valor óptimo de la función).

Probablemente la técnica más conocida dentro de este grupo, sea la propuesta de Koziel & Michalewicz llamada mapas homomorfos [78], en la cual se mantiene la factibilidad de las soluciones por medio de un mapeo entre la zona factible del problema y un cuerpo geométrico de forma conocida (por ejemplo, un hipercubo). El codificador establece una relación entre la solución factible y la solución codificada. La idea principal es transformar el problema original en uno topológicamente equivalente, en el cual la exploración de la zona factible resulte más fácil. Una última propuesta en este grupo es la de Kim & Husbands [72, 73], la cual usa mapas de Riemann para transformar la región factible a una forma que facilite la búsqueda al algoritmo genético.

Las principales desventajas de este grupo de técnicas es que, en general, son muy complejas y su costo computacional se eleva considerablemente conforme crece el número de variables del problema. Además, algunas de ellas están orientadas a la aplicación y requieren un punto inicial factible, que suele ser una condición dificil de satisfacer en muchas aplicaciones del mundo real.

Técnicas que separan objetivos y restricciones

Dentro del grupo de las técnicas que separan los objetivos de las restricciones tenemos las siguientes:

Paredis [98] propuso un algoritmo co-evolutivo con dos subpoblaciones, donde la evolución de una de ellas depende del estado de la otra y viceversa.

Schoenauer & Xanthakis [120] propusieron una técnica llamada *behavioural memory*, en donde las restricciones se ordenan de una manera determinada para satisfacerlas una a una de manera secuencial. El desempeño de este algoritmo se ve afectado entonces por el orden que se imponga a las restricciones.

Coello [14, 15] ha hecho propuestas en las que las restricciones del problema se convierten en objetivos y el nuevo problema se trata como uno multiobjetivo. La aptitud de los individuos se asigna de acuerdo a ciertas reglas que toman en consideración la factibilidad y en su caso, la cantidad restricciones que son violadas. Esta técnica, como todas las técnicas para manejo de restricciones basadas en conceptos multiobjetivo, tiende a ser más eficiente cuando se lidia con espacios altamente restringidos porque tiende a localizar rápidamente la región factible. Sin embargo, tiene problemas aproximando la solución óptima, pues se le dificulta muestrear adecuadamente la zona factible.

Parmee & Purchase [100] proponen una variante del algoritmo multiobjetivo denominado VEGA [118], la cual usa operadores genéticos especiales.

Surry et al. [129, 128] usan jerarquización de Pareto y VEGA en un esquema llamado COMOGA. En esta técnica, la población es ordenada de acuerdo a la violación a las restricciones. Una porción de la población es seleccionada con base en este ordenamiento y el resto con base en la aptitud real de los individuos. Una de las desventajas de esta propuesta es que requiere la definición de varios parámetros. Adicionalmente, su costo computacional es bastante elevado.

Técnicas de penalización

Dentro de las técnicas que penalizan a los individuos infactibles, tenemos las siguientes:

Las penalizaciones estáticas, propuestas por Homaifar et al. [58] que define niveles de penalización que se van usando a lo largo de la corrida del algoritmo.

Un esquema sumamente interesante y que adopta una penalización estática es la propuesta de Runarsson & Yao [113] basada en un ordenamiento estocástico; en ella la población es clasificada usando una versión estocástica del método de ordenamiento de la burbuja. Este algoritmo utiliza una estrategia evolutiva multi-miembro y su desempeño es altamente dependiente de un parámetro que regula el criterio de selección y permite conservar individuos infactibles en la población.

Las penalizaciones dinámicas, propuesta por Joines [66] consisten en definir factores de penalización dinámicos que dependen del número de generación en que se encuentre el algoritmo.

Las penalizaciones basadas en el recocido simulado, fueron propuestas por Michalewicz & Attia [88]. Este método usa coeficientes de penalización que cambian a lo largo de muchas generaciones, con base en la fórmula usada en el recocido simulado [75].

Las penalizaciones adaptativas en las cuales los factores de penalización se van adaptando según el estado actual de la población mediante un proceso de retroalimentación, fueron propuestas por Bean & Hadj-Alouane [5].

El algoritmo genético segregado, el cual consiste en tener 2 parámetros de penalización por cada restricción, buscando con ello lograr un equilibrio entre un factor de

penalización alto y uno bajo. Esta técnica fue propuesta por Le Riche et al. [108]

Híbridos

Este grupo está compuesto por algoritmos que combinan diferentes técnicas, incluyendo aquellas técnicas alternativas que tienen inspiración biológica. En este grupo tenemos:

Un algoritmo que usa principios de lógica difusa combinada con programación evolutiva, propuesto por Van Le [82].

La propuesta de Hajela & Lee [50, 51], la cual está basada en el sistema inmune natural. En este enfoque se separan los individuos factibles de los infactibles y se busca reducir la distancia genotípica entre estos dos grupos, intentando con ello reducir la cantidad de violación a las restricciones.

Los algoritmos culturales [105, 106, 19, 65, 13], en los cuales la cultura es vista como un proceso hereditario evolutivo que trabaja tanto a nivel microevolutivo como macroevolutivo.

Finalmente, tenemos la propuesta de Adeli & Cheng [1] que es un híbrido de una técnica de penalización con una técnica de programación matemática (los multiplicadores de Lagrange).

3.3. Optimización multiobjetivo

En la naturaleza podemos encontrar que la mayoría de los problemas poseen varios objetivos que deben optimizarse a la vez, y en la mayoría de los casos, se tienen además, múltiples restricciones.

La optimización multiobjetivo es un área que ha tomado gran importancia en los últimos años y que tiene su origen en la Investigación de Operaciones [93].

En este trabajo se desarrolla un algoritmo para solucionar problemas multiobjetivo. Es por ello que esta sección está dedicada a proporcionar algunos conceptos básicos que son necesarios para el correcto entendimiento del problema, así como una revisión breve del estado de arte.

3.3.1. Definición

De manera formal, el problema general de optimización multiobjetivo es el siguiente:

$$\text{Encontrar} \quad \vec{x} \text{ que optimice } \vec{f}(\vec{x}) = [f_1\vec{x}, f_2\vec{x}, \ldots, f_k\vec{x}] \tag{3.4}$$

sujeto a:

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, n \tag{3.5}$$

$$h_j(\vec{x}) = 0, \quad j = 1, \ldots, p \tag{3.6}$$

donde $\vec{x}$ es el vector de soluciones $\vec{x} = [x_1, x_2, \ldots, x_r]^T$, n es el número de restricciones de desigualdad y p es el número de restricciones de igualdad (en ambos casos las restricciones pueden ser lineales o no lineales).

En problemas con múltiples objetivos el concepto de óptimo cambia, pues en este caso no existe una única solución al problema, sino un conjunto de soluciones compromiso. La noción más aceptada de "óptimo" en problemas multiobjetivo es aquella introducida por el economista Vilfredo Pareto en 1896 [99].

De manera formal lo definimos de la siguiente manera [17]:

Optimalidad de Pareto:

Un punto $\vec{x}^* \in \Omega$ (Ω es la región factible) es un **óptimo de Pareto** si para toda $\vec{x} \in \Omega$ y $I = \{1, 2, \ldots, k\}$

$$f_i(\vec{x}^*) \leq f_i(\vec{x}) \tag{3.7}$$

y, existe al menos un $i \in I$ tal que

$$f_i(\vec{x}^*) < f_i(\vec{x}) \tag{3.8}$$

Dominancia de Pareto

Un vector $\vec{u} = (u_1, \ldots, u_k)$ se dice que domina a $\vec{v} = (v_1, \ldots, v_k)$ (denotado por $\vec{u} \preceq \vec{v}$) si y sólo si $\vec{u}$ es parcialmente menor que $\vec{v}$, o sea, $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$.

Conjunto de óptimos de Pareto

Para un problema de optimización multiobjetivo dado $\vec{f}(x)$, el conjunto de óptimos de Pareto($\mathcal{P}^*$) se define como:

$$\mathcal{P}^* := \{x \in \Omega \mid \neg\exists\, x' \in \Omega \ \ \vec{f}(x') \preceq \vec{f}(x)\}. \tag{3.9}$$

Frente de Pareto:

Para un problema de optimización multiobjetivo dado $\vec{f}(x)$ y un conjunto de óptimos de Pareto $\mathcal{P}^*$, el frente de Pareto ($\mathcal{PF}^*$) se define como:

$$\mathcal{PF}^* := \{\vec{u} = \vec{f} = (f_1(x), \ldots, f_k(x)) \mid x \in \mathcal{P}^*\}. \tag{3.10}$$

3.3.2. Estado del arte

Las técnicas evolutivas que se han desarrollado para solucionar problemas de

optimización multiobjetivo, las podemos clasificar en dos grandes grupos [17]:

- Enfoques no basados en optimalidad de Pareto
- Enfoques basados en optimalidad de Pareto

Técnicas no basadas en optimalidad de Pareto

Algunas de las **técnicas no basadas en Pareto** más conocidas se mencionarán a continuación:

- **Técnicas de muestreo independiente**: Existen pocos algoritmos evolutivos de este tipo reportados en la literatura especializada [92, 109, 84]. En ellos se usa una combinación de pesos para la función de aptitud; tales pesos se varían de manera uniforme en varias corridas del algoritmo evolutivo. Con estas técnicas se logran encontrar bastantes soluciones no dominadas después de varias corridas. Sin embargo, su principal desventaja es su elevado costo computacional.
- **Técnicas de criterio de selección**: Un ejemplo de estas técnicas es el *Vector Evaluated Genetic Algorithm* (VEGA) propuesto por

Schaffer [118] en el que la población se divide en un número de subpoblaciones igual a la cantidad de objetivos que se desean optimizar. La idea es hacer que cada subpoblación seleccione con base en un solo objetivo, y posteriormente las subpoblaciones se mezclan en una sola a la que se le aplican los operadores genéticos convencionales (cruza y mutación). Este algoritmo es sencillo, pero no garantiza la generación de soluciones no dominadas.

- **Técnicas de selección agregativas**: En ellas se aplica un criterio de selección de acuerdo a la combinación ponderada de los objetivos, ya sea usando una función agregativa lineal o una no lineal [81, 95]. Los pesos varían entre generaciones y/o en cada función de evaluación. Estas técnicas tienen la ventaja principal de que en una sola ejecución se pueden alcanzar varios segmentos del frente de Pareto; su principal desventaja es, que en ciertos casos (cuando se usan funciones agregativas lineales), existen puntos del frente de Pareto que no pueden generarse.

Técnicas que usan selección basada en Pareto

La idea principal de éstas es asignar la aptitud de los individuos con base en el concepto de optimalidad de Pareto, para identificar los vectores no dominados de la población. Dentro de este grupo tenemos una gran cantidad de propuestas, de entre las que destacan las siguientes:

- **MOGA** (*Multi Objective Genetic Algorithm*): Fue propuesta por Fonseca y Fleming ([43]). En este algoritmo la aptitud de un individuo está relacionada a la cantidad de individuos que lo dominan (es decir, los individuos no dominados tienen la aptitud más elevada de la población) y todos los individuos no dominados reciben el mismo valor de aptitud.

- **NSGA** (*Nondominated Sorting Genetic Algorithm*): Propuesto por Srinivas y Deb ([127]). En este algoritmo se clasifica primero a los individuos no dominados globales (o sea, con respecto a toda la población), y se les asigna una aptitud "falsa". Estos individuos se remueven

de la población para continuar el proceso de clasificación con los individuos restantes. Recientemente se propuso una versión más eficiente de este algoritmo llamada NSGA-II (Deb et al.) [36] [37] en la que se usa un operador de *crowding*, y un mecanismo elitista.

- **NPGA** (*Niched Pareto Genetic Algorithm*), propuesto por Horn y Nafpliotis [59]. Usa una versión modificada de selección por torneo con dominancia de Pareto de la siguiente manera: Se seleccionan dos posibles padres y se comparan contra un subconjunto (seleccionado aleatoriamente) de la población; aquél que resulte no dominado con respecto al subconjunto será el ganador. Si existe un empate (ambos son dominados o no dominados), entonces el resultado del torneo se decide usando compartición

  de aptitud (el individuo con menos vecinos gana).

  Una versión mejorada de este algoritmo llamada NPGA2, fue propuesta por Erickson et al. [39]. En esta nueva versión se aplica ordenamiento usando el concepto de Pareto y se sigue realizando selección mediante torneo, aunque se usa un esquema diferente de compartición de aptitud.

- **SPEA** (*Strength Pareto Evolutionary Algorithm*) propuesto por Zitzler y Thiele [141]. Este algoritmo usa un archivo externo que contiene todas las soluciones no dominadas encontradas previamente. Dicho archivo se actualiza a cada generación. Se calcula la "fuerza" (*strength*) de cada uno de sus individuos de manera similar a la clasificación de Pareto que hace MOGA, pues depende de la cantidad de individuos a los que domina. La aptitud de los individuos de la generación actual se calcula de acuerdo a la "fuerza" de las soluciones del archivo externo que lo dominan. Para lograr una buena distribución de soluciones no dominadas, SPEA usa el denominado "método de los enlaces promedio" (*average linkage method*).

  La versión mejorada de este algoritmo se llama SPEA2, y fue propuesta por Zitzler et al. [143]. SPEA2 tiene tres diferencias principales con respecto a su versión original: 1) en la asignación de aptitud se toma

en cuenta tanto a la cantidad de individuos que lo dominan, como a los que domina, 2) se usa una estimación de densidad de los vecinos para eficientar más la búsqueda y, 3) se usa un método de truncado del archivo externo que asegura que no se pierdan las soluciones de los extremos del frente de Pareto.

- **PAES** (*Pareto Archived Evolution Strategy*) propuesto por Knowles y Corne [77]. PAES es una estrategia evolutiva (1+1) (significa que se tiene una población de un individuo padre que genera un solo hijo por mutación). PAES usa el archivo externo para retener las soluciones no dominadas obtenidas a lo largo del proceso evolutivo. Para mantener la diversidad se usa un mecanismo de "*crowding*" que se basa en una división recursiva del espacio de las funciones objetivo, de manera que los individuos quedan localizados en una retícula que facilita determinar y controlar la distribución de las soluciones obtenidas.

- **PESA** (*Pareto Envelope-Based Algorithm*) propuesto por Corne et al. [23], usa una población interna pequeña y una externa (o secundaria) grande. Al igual que PAES, también usa una rejilla adaptativa aplicada en el espacio de las funciones objetivo para mantener la diversidad, con la diferencia de que la selección, en este caso, está basada en la medida de "crowding" adoptada por PAES. La versión mejorada de este algoritmo (PESA-II [22]) es básicamente igual, sólo que posee un mecanismo de selección más eficiente que reduce el costo computacional asociado con los algoritmos multiobjetivo basados en ordenamiento de Pareto.

- **Micro algoritmo genético** (micro-AG), propuesto por Coello y Toscano [20, 21]. Este es un algoritmo con una población inusualmente pequeña (sólo 4 individuos) y un proceso de re-inicialización. La técnica posee dos memorias, una fija y otra reemplazable. También usa un archivo externo o histórico (similar al de PAES) donde se retienen las mejores soluciones encontradas. El micro-AG usa tres formas de elitismo que son: 1) Retiene las soluciones no dominadas encontradas en el ciclo interno del micro-AG, 2) Usa una memoria

reemplazable que es "refrescada" de manera parcial a ciertos intervalos de tiempo y, 3) Sustituye individuos de la memoria reemplazable del micro-AG usando soluciones no dominadas producidas por el micro-AG.

Manejo de restricciones en optimización multiobjetivo

Cuando se trata de resolver problemas de optimización multiobjetivo con restricciones, es necesario tomar en cuenta los conceptos de factiblidad e infactibilidad de los individuos. La manera más común de tratar las restricciones en optimización multiobjetivo es manejándolas como objetivos [15].

Algunas técnicas aplican penalizaciones a las soluciones infactibles o intentan repararlas (es decir, hacerlas factibles). En la sección dedicada a manejo de restricciones se mencionan las desventajas que presentan estos mecanismos.

Otra técnica empleada suele ser la selección mediante torneo donde las reglas para ganar están en función directa de la factibilidad o cercanía a la zona factible de los individuos [87, 103, 38].

En general, podemos decir que existe poco trabajo en este sentido dentro del área, y no son muchos los algoritmos que cuentan con un mecanismo explícito para lidiar con las restricciones basado en conceptos multiobjetivo (ver ejemplo [104, 14, 128]).

3.3.3. Sistemas inmune artificiales para problemas de optimización

Hay poco trabajo previo que se relacione con lo realizado en esta tesis. A continuación, lo describiremos brevemente.

Existe una propuesta de un sistema inmune artificial (SIA) para problemas de optimización multimodal realizada por De Castro y Timmis [33]. Esta propuesta se basa en el principio de selección clonal [10] y en la teoría de red inmune [64, 101].

Hajela y Lee [50, 51] propusieron un SIA para manejo de restricciones acoplado a un AG, en el cual se basa nuestra propuesta. El algoritmo se

describirá a detalle en el siguiente capítulo.

Yoo y Hajela [140] presentaron la primera propuesta de un SIA para optimización multiobjetivo en el que se usa una función agregativa que combina la función objetivo y la cantidad de violación de restricciones, la cual se utiliza como aptitud. Este esquema consiste realmente de un algoritmo genético simple al que se le acopla un sistema inmune artificial cuya búsqueda se basa en similitudes entre cadenas binarias. Al usarse una función agregativa lineal, se tiene la limitante de no poder generar porciones cóncavas del frente de Pareto [26]. Adicionalmente, se depende de la variabilidad de los pesos usados en la función agregativa para determinar la distribución de soluciones obtenida en el frente de Pareto.

Anchor y Lamont [2] propusieron un SIA multiobjetivo que usa selección basada en ordenamiento lexicográfico y optimalidad de Pareto en un algoritmo evolutivo, para detectar intrusos. El algoritmo está enfocado hacia la aplicación y no a la solución de problemas multiobjetivo en general, por lo que no se realiza ninguna comparación con ningún algoritmo evolutivo multiobjetivo.

Finalmente, hay una propuesta muy reciente [85] de un algoritmo para optimización multiobjetivo que aplica el principio de selección clonal, recombinación de segmentos de ADN y diversificación de anticuerpos. En esta propuesta los objetivos del problema toman el papel de los antígenos, mientras que las posibles soluciones representan a los anticuerpos. Se aplica clonación y mutación a los anticuerpos no dominados, con parámetros definidos por el usuario. Se define una medida de "avidez" que es proporcional a la jerarquía del anticuerpo (de acuerdo a dominancia de Pareto) y a su distancia Euclidiana con respecto a los demás anticuerpos. Esta medida de avidez se utiliza para aplicar selección por torneo. A continuación se crea una biblioteca utilizando fragmentos (de tamaño definido por el usuario) de los anticuerpos ganadores del torneo, la cual es usada para formar la siguiente generación de anticuerpos. Una desventaja clara de este algoritmo es la gran cantidad de parámetros que introduce. Además, no se especifica cómo manejar los problemas con restricciones. Los autores argumentan que su algoritmo produce resultados competitivos con respecto a SPEA [77].

Capítulo 4

Sistema inmune artificial para manejo de restricciones en optimización mono-objetivo

4.1. Descripción del problema

La incorporación de manejo de restricciones a los algoritmos genéticos (AG) no es una tarea fácil. Existen varias propuestas que podemos clasificar en cinco grandes grupos [16, 91]: técnicas que rechazan a los individuos infactibles, técnicas que mantienen una población factible por medio de representaciones y operadores genéticos especiales, técnicas que separan objetivos y restricciones, técnicas que penalizan a los individuos infactibles y técnicas híbridas. Nuestra propuesta puede ser clasificada dentro de este último grupo (ver Sec. 3.1.2).

Las técnicas más utilizadas por los investigadores del área son las técnicas de penalización, cuya principal desventaja es que su desempeño depende de la función de penalización y de los factores de penalización adoptados, los cuales en general, son dependientes del problema [107]. Otras técnicas más eficientes propuestas recientemente parecen ser buenas alternativas. De entre ellas se destacan los mapas homomorfos [79], el ordenamiento estocástico (*stochastic ranking* [113]) y el *Adaptive Segregational Constraint Handling Evolutionary Algorithm* (ASCHEA [52]).

En esta sección describiremos un algoritmo para manejar restricciones en algoritmos genéticos (AG) usando una metáfora del sistema inmune. Este sistema inmune artificial propuesto intenta superar las principales desventajas de las técnicas de penalización, siendo un algoritmo que no requiere la definición de ningún factor de penalización y que introduce una cantidad relativamente pequeña de parámetros que el usuario debe definir, con un costo computacional bajo en términos del número de evaluaciones de la función de aptitud.

La definición formal del problema que deseamos solucionar es la siguiente (ver sección 3.2).

$$\text{Encontrar } \vec{x} \text{ que optimice } f(\vec{x}) \tag{4.1}$$

sujeto a:

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, n \tag{4.2}$$

$$h_j(\vec{x}) = 0, \quad j = 1, \ldots, p \tag{4.3}$$

donde $\vec{x}$ es el vector de soluciones $\vec{x} = [x_1, x_2, \ldots, x_r]^T$, n es el número de restricciones de desigualdad y p es el número de restricciones de igualdad (en ambos casos las restricciones pueden ser lineales o no lineales).
Llamaremos Q al conjunto de restricciones tanto de igualdad ($g(\vec{x})$) como de desigualdad ($h(\vec{x})$).

Nuestro algoritmo está basado en la propuesta de Hajela et al. [50, 51] y es una mejora al trabajo desarrollado en [24]. Los resultados reportados en esta tesis muestran que se lograron mejoras con respecto al trabajo preliminar [24], esto es, tanto en mejores aproximaciones a los valores óptimos como en cantidad de evaluaciones de la función de aptitud.

4.2. Principio inmunológico utilizado

El sistema inmune crea a sus detectores (anticuerpos) a partir de la combinación aleatoria de segmentos de genes que se encuentran distribuidos a lo largo del genoma, con lo que logra gran parte de su diversidad. Después de haber sido creados, los detectores son probados para sólo permitir la

supervivencia de los que posean mayor capacidad para reconocer antígenos dañinos.

Cuando el organismo se ve invadido por algún antígeno determinado, los anticuerpos que le son más afines sufren mutaciones, con lo que algunos de ellos podrían incrementar su afinidad.

Este proceso de búsqueda de los anticuerpos más afines para un antígeno determinado a través de la combinación de segmentos (bloques constructores) y mutación del sistema inmune es el que utilizaremos para nuestro algoritmo.

Estos conceptos se explicaron en la sección 2.2.1 con mayor detalle.

4.3. Adaptación del sistema inmune artificial

El sistema inmune artificial que manejará las restricciones, es acoplado a un algoritmo genético, y éste a su vez es insertado dentro de otro algoritmo genético que optimiza una función objetivo $f(\vec{x})$. De manera que tenemos un algoritmo genético insertado dentro de otro. El externo es el algoritmo genético convencional que optimiza $f(\vec{x})$, mientras que el interno es el encargado de manejar las restricciones y es el que realiza la emulación del sistema inmune.

En analogía con el sistema inmune biológico, el objetivo de nuestro sistema inmune artificial (AG interno), es construir los anticuerpos adecuados para un antígeno específico que se presenta, es decir, los anticuerpos con mayor afinidad a éste.

Los individuos factibles juegan el papel de los antígenos, mientras que los infactibles son los anticuerpos. De esta manera se busca que los anticuerpos se parezcan más a los antígenos, es decir, que la población de individuos infactibles se parezca más a la población de individuos factibles, (ver figura 4.1).

Para saber qué tan afines son los anticuerpos con respecto a la población de antígenos, utilizaremos una distancia genotípica (a nivel de bits) que indicará el grado de similitud entre las cadenas binarias de un antígeno y un anticuerpo. Entonces diremos que a mayor similitud, mayor afinidad. Utilizaremos este valor como la función de aptitud para los anticuerpos

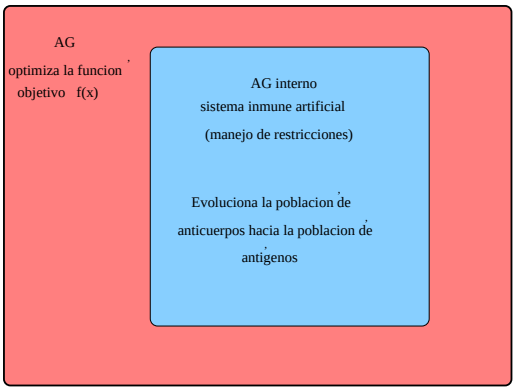

Figura 4.1: Sistema inmune artificial para manejo de restricciones en AG's

dentro del AG interno. A continuación se mencionarán los detalles.

Elementos básicos de un sistema inmune artificial

Aplicando la definición de los elementos básicos de un sistema inmune artificial mostrada en la sección 2.3.2, definimos éstos para nuestro algoritmo de la siguiente manera:

- **Representación de los elementos del sistema:** Utilizamos antígenos (individuos factibles) y anticuerpos (individuos infactibles) representados mediante cadenas binarias de la misma longitud (se adoptarán códigos de Gray), ver sección 2.2.1.
- **Interacción de los elementos entre sí y con el ambiente:** La interación entre antígenos y anticuerpos, es decir, la afinidad entre ellos, se efectúa a través de una medida de similitud a nivel de bits.
- **Proceso de adaptación que gobierna las dinámicas del sistema:** Utilizamos el mecanismo mediante el cual el sistema inmune logra crear los anticuerpos correctos para el antígeno que se le presenta.

La representación binaria presenta problemas con el mapeo entre genotipo y fenotipo debido a que no mantiene la propiedad de adyacencia entre 2 valores consecutivos, es por esto que utilizamos códigos de Gray dado que corrige este problema.

4.4. Algoritmo propuesto

A continuación describiremos el algoritmo completo:

Parámetros de entrada:

- Función objetivo ($f(\vec{x})$)
- Restricciones del problema (Q)
- Tamaño N de la población
- Generaciones del AG externo (Text)
- Generaciones del AG interno (Tint)
- Porcentaje de cruza (Pc)
- Tipo de cruza (Tc) (1 punto, 2 puntos o uniforme).
- Porcentaje de mutación (Pm)

INICIO

1. Generar la población inicial del algoritmo genético (POB) de tamaño N de manera aleatoria.
   INICIA AG INTERNO (SIA)

   a) Asignar las poblaciones de *antígenos* (Ag) y *anticuerpos* (Ac) de la siguiente manera:

   - Si POB contiene individuos factibles e infactibles entonces, los individuos factibles forman la población de Ag y los infactibles la población Ac.
   - Si no, el mejor individuo es asignado como único miembro de Ag y el resto de los individuos son asignados a Ac. El mejor individuo se determina de la siguiente manera:
     - El que viole restricciones,

- en caso de empate en el punto anterior: entonces gana aquel cuyo valor de violación a las restricciones Q sea menor.
- Si el empate permanece entonces, se selecciona al ganador de manera aleatoria.

b) Se calcula la aptitud (Z) de cada uno de los individuos de Ac, la cual es una medida de similitud genotípica con respecto a un individuo tomado aleatoriamente de Ag. Se explica a detalle más adelante.

c) Basados en la aptitud Z calculada en el paso anterior seleccionar a los σ anticuerpos que serán *padres* de la siguiente generación.

d) Aplicar el operador de cruza Tc con probabilidad Pc a los *padres* para crear a los *hijos*. Si existen individuos factibles dentro de la población, entonces cada padre se cruza con un individuo infactible.

e) Aplicar mutación uniforme con probabilidad Pm a la población de *hijos* y formar con ellos la siguiente generación.

f) Se repite el proceso Tint veces desde el paso (b)

g) Ambas poblaciones (Ag y Ac) se unen para formar POB

FIN DEL AG INTERNO (SIA)

2. Se seleccionan a los N *Padres* de POB por medio de torneo binario utilizando reglas especiales que consideran los valores de la función objetivo ($f(\vec{x})$) y las restricciones (Q), (se explican a detalle más adelante).

3. Se aplica cruza Tc con probabilidad Pc a la población de *Padres* seleccionados en el paso anterior para crear la población de *Hijos*

4. Aplicar mutación uniforme con probabilidad Pm a los *Hijos*

5. Se repite el proceso Text veces desde el paso 1(a)

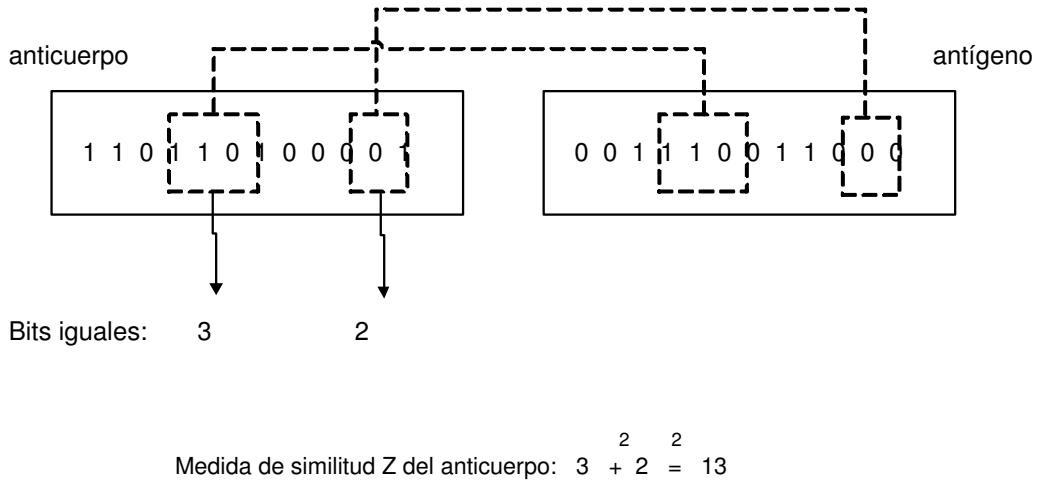

Figura 4.2: Medida de similitud Z entre un antígeno y un anticuerpo. Es utilizada como valor de aptitud de los anticuerpos. Se calcula mediante la sumatoria de los cuadrados de las longitudes de las subcadenas que son iguales en las cadenas binarias del antígeno y el anticuerpo. En este ejemplo tenemos 2 subcadenas iguales: la primera de longitud igual a 3 y la segunda de longitud igual a 2, entonces sumamos el cuadrado de ambas longitudes $Z = 3^2 + 2^2$.

FIN
Salida entregada por el algoritmo: Valores de $\vec{x}$ que optimizan la función $f(\vec{x})$ sujeto a las restricciones Q.

En el paso 1(b) del algoritmo, se calcula la aptitud (Z) de los anticuerpos que consiste en una medida de similitud entre la cadena del anticuerpo con un antígeno. Esta medida tiene en consideración las subcadenas que son iguales y la longitud de ellas [60]. Se calcula mediante la sumatoria de los cuadrados de las longitudes de las subcadenas que son iguales del antígeno y el anticuerpo [1]. Ver figura 4.2.

El operador de selección de torneo binario, al que nos referimos en el paso 2 del algoritmo, está definido por las siguientes reglas:

[1] Se probaron diferentes medidas como distancias de Hamming y otras estrategias, sin embargo, la aquí descrita es la que reportó mejores resultados.

- Si ambos individuos son factibles, gana el que tenga mejor valor de función de aptitud $f(\vec{x})$

- Si uno es factible y el otro no, siempre gana el factible

- Si ambos individuos son infactibles,

  gana aquél cuyo valor de violación a la restricciones Q es menor.

Este algoritmo también puede verse en la figura 4.3.

El algoritmo trabaja con una población de individuos factibles e infactibles, lo que le permite realizar una exploración de los límites de la región factible, que como sabemos, en muchos casos es donde suele encontrarse el valor óptimo de la función objetivo.

El objetivo del sistema inmune artificial (SIA) es evolucionar a la población de anticuerpos hacia la población de antígenos, en similitud con el sistema inmune biológico. Es decir, intenta que los individuos infactibles tengan mayor similitud con la población de individuos factibles. Este esquema no requiere la definición de ninguna función de penalización.

El algoritmo introduce únicamente dos parámetros adicionales a los que deben definirse en un AG convencional, que son la cantidad de generaciones para el AG interno y el tamaño de la muestra de los anticuerpos que será sometido al proceso evolutivo. El AG interno no evalúa la función objetivo del problema, por lo que el costo computacional en este aspecto no se incrementa significativamente. [2]

Es importante hacer notar que el algoritmo supone que similitudes a nivel del genotipo resultarán en similitudes a nivel del fenotipo, lo cual pudiera ser debatible. Sin embargo, los resultados obtenidos por Hajela [51], los reportados en [24] y lo presentado en esta tesis sustentan esta hipótesis.

Algoritmo de Hajela et al

[2] Se está presuponiendo que el costo computacional más significativo dentro del algoritmo genético es el de la evaluación de la función objetivo.

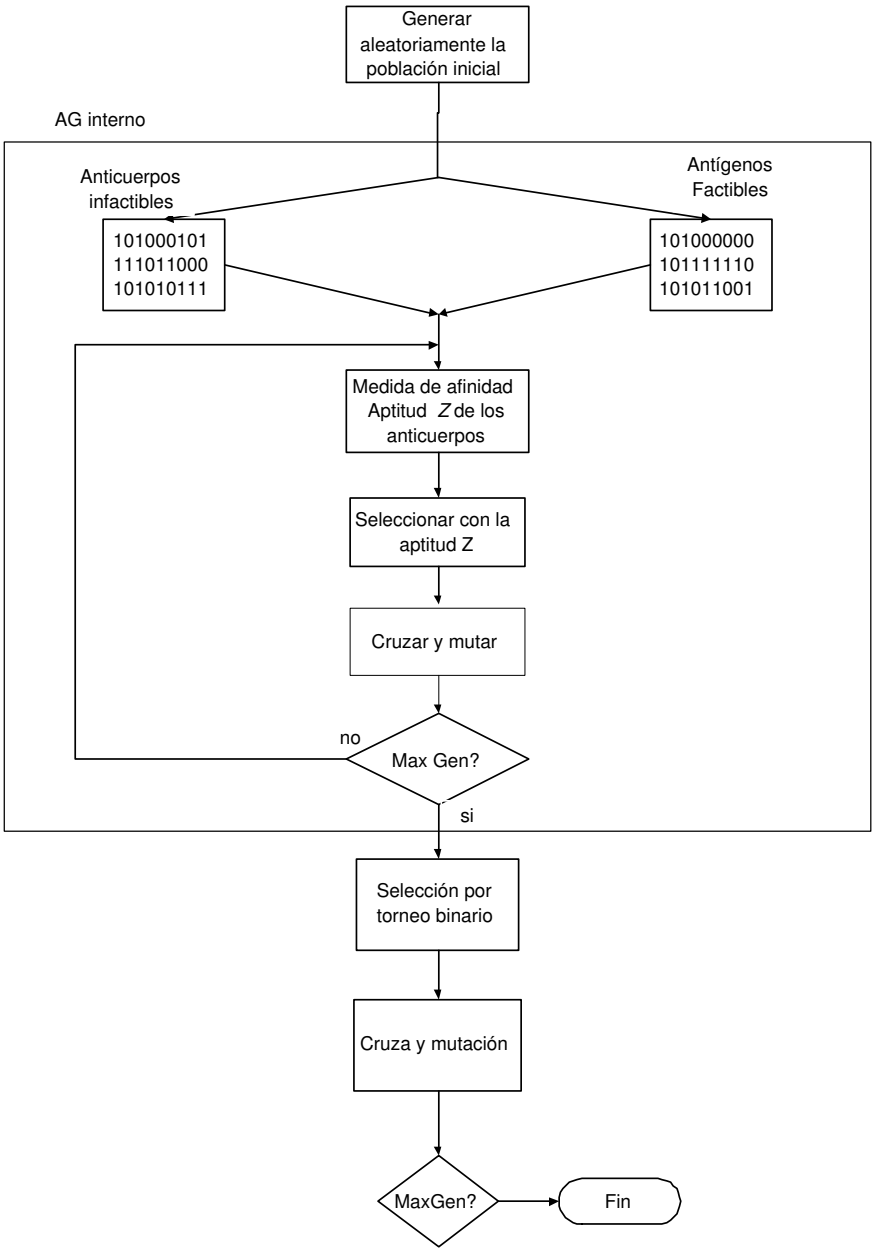

Figura 4.3: Algoritmo del sistema inmune artificial para manejo de restricciones

Nuestra propuesta está basada en el algoritmo de Hajela et al. [50, 51], que se ilustra en la figura 4.4. En ese algoritmo Hajela realiza el cálculo de una función de penalización, y después la población se ordena de acuerdo a ese criterio. Con ello se determina cuáles individuos formarán la población de anticuerpos y cuáles la de antígenos. Los antígenos están formados por el 3 % de mejores individuos, mientras que el resto de la población formarán los anticuerpos. La función de penalización se define de la siguiente manera:

$\sum [max(0, Q)]^2$

El cálculo de la aptitud Z de la población de anticuerpos se efectúa a través de varias operaciones sencillas dentro de un ciclo. En este algoritmo, además de los parámetros propios de los AG's convencionales, deben definirse los siguientes parámetros:

- Número de generaciones para el AG interno.
- Número de repeticiones para el ciclo que efectúa el cálculo de Z.
- Tamaño de la población de anticuerpos que serán sometidos al proceso (σ).

En este caso, el cálculo de la función objetivo del problema $f(\vec{x})$ debe calcularse 2 veces por cada individuo en cada generación del AG externo, la primera para realizar el cálculo de la función de penalización y la segunda para llevar a cabo el proceso de selección del AG externo (asignar el valor de aptitud del AG externo). En la validación de su algoritmo, Hajela et al. no especifican los valores de los parámetros utilizados, tales como porcentaje de cruza y mutación, tipo de selección y cruza, tamaño de la población, número de generaciones para el AG interno y externo y tamaño de la población de anticuerpos sometido al proceso inmune. Ellos aplican su algoritmo en algunos problemas de ingeniería y no proveen información acerca del costo computacional de éste.

Diferencias de nuestro algoritmo con el de Hajela et al.

En nuestro algoritmo intentamos superar las principales desventajas del algoritmo de Hajela et al. A continuación mencionamos cuáles son las diferencias entre ellos:

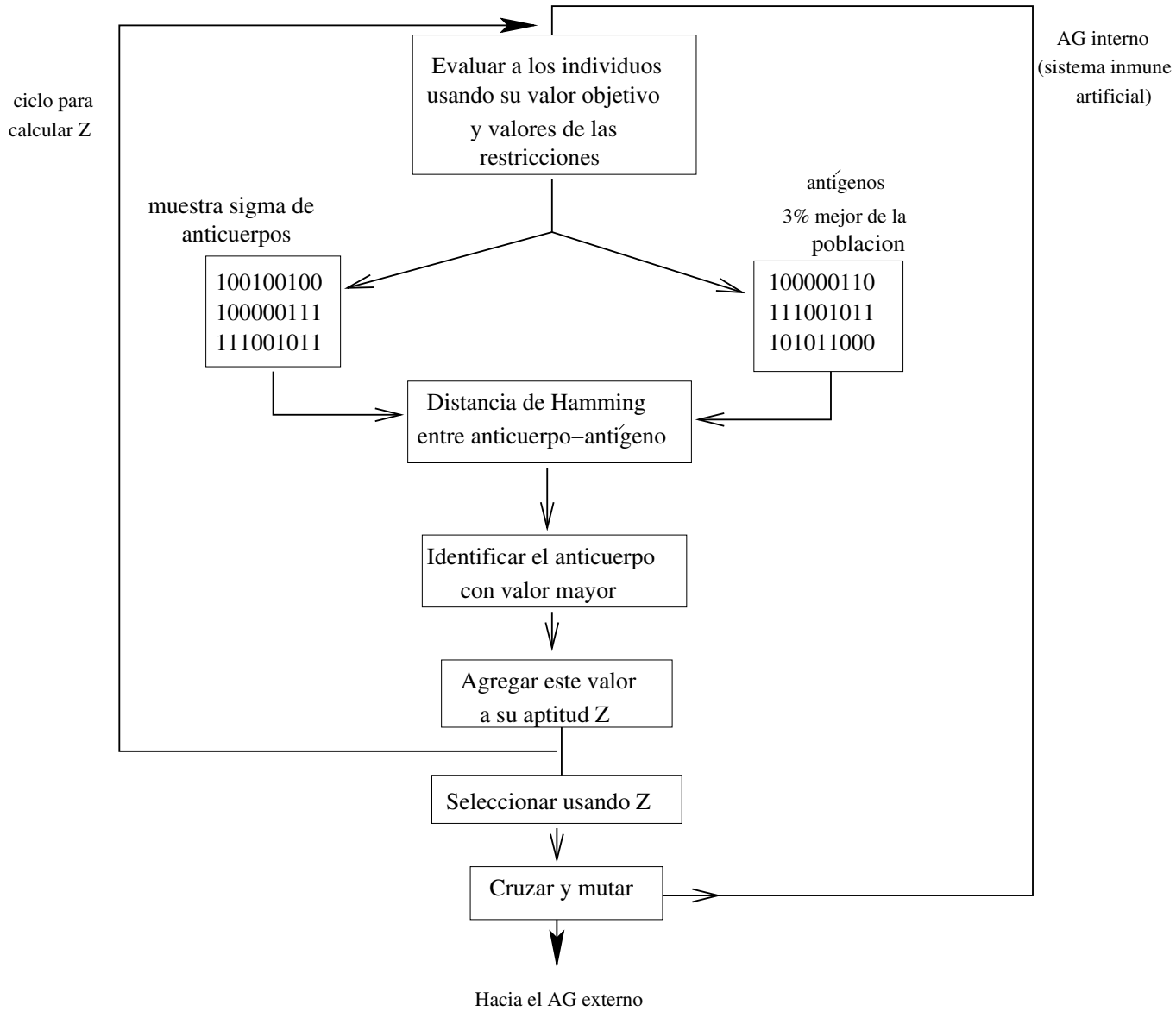

Figura 4.4: Sistema inmune artificial para manejo de restricciones en AG's propuesto por Hajela et al. [50, 51]

- A diferencia de éste, nuestro algoritmo no usa ninguna función de penalización.
- Nuestro algoritmo sólo requiere una evaluación de la función objetivo por cada individuo en cada generación, a diferencia del de Hajela que requiere dos.
- No se requiere ordenar la población.
- Evitamos el bucle que calcula el valor de la función de aptitud para la población de anticuerpos.
- El cálculo del valor de aptitud para la población de anticuerpos en nuestro algoritmo, además de la cantidad de subcadenas de bits que coinciden entre antígeno-anticuerpo, considera la longitud de éstas.
- En el AG interno, intentamos que la cruza se realice entre un individuo factible y uno no factible.

Para validar nuestro algoritmo, hemos utilizado el conjunto de funciones de prueba que suele adoptarse en el área de computación evolutiva para validar nuevos esquemas de manejo de restricciones [90] y comparamos nuestros resultados contra los obtenidos por técnicas que son representativas del estado del arte en el área. Realizamos también la programación del algoritmo propuesto por Hajela et al. y lo comparamos contra nuestro algoritmo a fin de verificar si los cambios realizados producen diferencias significativas de desempeño. Los resultados obtenidos para una función de prueba se muestran en la sección de resultados.

Tiempo de procesamiento del algoritmo en términos de la evaluación de la función objetivo

Se considera que la tarea que más consume tiempo en un AG simple (o canónico) es la evaluación de la función de aptitud $f(\vec{x})$ y las restricciones Q para el caso de espacios de búsqueda restringidos, mientras que el de las operaciones de selección, cruza y mutación se considera despreciable. Llamaremos $T(f)$ a el tiempo de evaluación de la función objetivo $f(\vec{x})$. En cada generación del algoritmo cada individuo evalúa una vez $f(\vec{x})$, entonces

el tiempo de procesamiento (Γ) se determina de la siguiente manera:

$\Gamma = G * N * T(f)$

donde G es el número de generaciones del AG, N es el tamaño de la población y $T(f)$ a el tiempo de evaluación de la función objetivo.

Al tratarse de espacios de búsqueda restringidos, debemos considerar además el tiempo de evaluación de las restricciones del problema $T(Q)$ que en nuestro caso particular del sistema inmune artificial, se evalúan 2 veces en cada ciclo (antes y después de ir al AG interno). Con respecto al AG interno, el tiempo de evaluación está en función de la cantidad de anticuerpos existentes en la población (individuos infactibles) y la cantidad de generaciones de éste. Tenemos que la función de aptitud del AG interno Z es una distancia genotípica entre dos cadenas binarias, y llamaremos $T(Z)$, al tiempo que toma su evaluación. Entonces el tiempo de evaluación de nuestro sistema inmune artificial, medido en términos de la función de aptitud y las restricciones se calcula de la siguiente manera:

$\Gamma_{SIA} = G * N * (T(f) + 2T(Q)) + (G * C * n * T(Z))$

donde: C es la cantidad de generaciones del AG interno, n es la población de anticuerpos que evoluciona, $T(Q)$ es el tiempo de evaluación de las restricciones y $T(Z)$ es el tiempo de evaluación de la función de aptitud Z del AG interno.

4.5. Versión paralela del algoritmo propuesto

La versión paralela de este algoritmo se realizó utilizando el modelo de múltiples poblaciones o de grano grueso (ver sección 2.1.2); éste es un modelo de memoria distribuida en el que la población se distribuye en subpoblaciones (demes) que evolucionan independientemente e intercambian individuos a intervalos (épocas). Cada subpoblación del AG (deme) es asignado a un procesador. Los demes evolucionan de manera independiente y, al cumplirse una época, cada uno de ellos copia un grupo de individuos hacia otro deme. A su vez cada deme, recibe a otro grupo de individuos (migración).

A cada deme le aplicamos diferentes operadores de selección, tipo y porcentaje de cruza, y mutación (ver sección 2.1.1), con la finalidad de

explorar diferentes regiones del espacio de búsqueda. La solución entregada por el algoritmo es la mejor encontrada en todos los demes.

La definición de los parámetros de este AG paralelo es como se indica a continuación:

- Tamaño de cada deme: se obtiene dividiendo el tamaño de la población global entre la cantidad de demes deseados.

- Época: Se efectúa la migración cuatro veces a lo largo del proceso evolutivo.

- Porcentaje de migración: se migra al 25 % de la población del deme

- Política de migración: el grupo de individuos que se copian hacia otro deme está formado por el mejor individuo del deme más un conjunto de individuos seleccionados aleatoriamente.
  Los individuos que son reemplazados en el deme receptor, son seleccionados de manera aleatoria.

- Interconexión de los demes: Los demes están totalmente conectados. Se permite ir de un nodo a cualquier otro. A cada época un deme migra hacia un deme diferente cada vez.

- La cantidad de generaciones del AG externo, se establece de forma que la cantidad de evaluaciones de la función objetivo $f(x)$ sea la misma que en la versión paralela (350,000).

- La cantidad de generaciones del sistema inmune artificial (AG interno) permanece constante en cada deme (30 generaciones cada uno).

El algoritmo fue implantado en una computadora homogénea con 8 procesadores bajo ambiente Linux y lenguaje C con compilador GNU C usando librerías de MPI (*Message Passing Interface*).

Funciones	Tint	N	Pm	σ	Tc
g02	.161659	**.000431**	**.000004**	.324372	**.0000001**
g03	.533344	.026185	**0.0000001**	.929089	.086741
g04	.234967	.957471	**0.0000001**	.777990	.148463
g05	.023031	**.000076**	**.000232**	.025897	.292550
g06	.750688	**0.0000001**	**0.0000001**	**.000001**	**.000949**
g07	.487986	.032380	**.0000001**	.480795	.745745
g09	.100541	.010627	.0000001	.770738	.016831
g10	.005904	.428648	**.001181**	**.000955**	.508680
g11	**.000197**	**.0000001**	**.0000001**	.060648	**.000629**

Cuadro 4.1: Valores de probabilidad de que el efecto de los parámetros (Tint, N, Pm, σ y Tc) sea producto del azar en las funciones de prueba.

4.6. Análisis de varianza

Con la finalidad de conocer el efecto de los parámetros del algoritmo sobre su desempeño, se llevó a cabo un estudio estadístico de análisis de varianza. A continuación explicaremos el diseño del experimento.
Los parámetros (o variables independientes) considerados para este estudio son:

1. Número de generaciones del AG interno (Tint)
2. Tamaño de la población (N)
3. Porcentaje de mutación (Pm)
4. Tamaño de la muestra de anticuerpos (σ)
5. Tipo de cruza (Tc)

El papel de la variable dependiente lo toma el resultado arrojado por el algoritmo, es decir, la solución encontrada.
Cada una de las variables independientes puede tomar diferentes valores (niveles), que a continuación se mencionan:

- Para las generaciones el AG interno, definimos dos niveles posibles: **15 ó 30**
- Tamaño de la población, dos niveles: **30 ó 90**
- Porcentaje de mutación, tenemos tres niveles que son:
  1. 2 / (longitud de la cadena binaria)
  2. Porcentaje de mutación dinámico, inicia en 0.4 y termina en 1/(longitud de la cadena) en ambos AG's (interno y externo)
  3. Porcentaje dinámico, inicia en 0.4 y termina en 1/(longitud de la cadena binaria) para el AG externo; y porcentaje fijo para el AG interno igual a 1/(longitud de la cadena binaria)
- Tamaño de la muestra σ, definimos dos niveles: **Todos los individuos infactibles o la mitad de ellos**.
- Tipo de cruza, tenemos tres niveles: **cruza de un punto, de dos puntos o uniforme**.

El experimento consistió en combinar de todas las maneras posibles los niveles que pueden tomar las variables independientes, de manera que, mediante un análisis de estadístico (ANOVA: *Analysis of Variance*) se pueda determinar, cuál de estas variables (parámetros) es la que tiene mayor efecto en el desempeño del algoritmo.

Con la finalidad de comprobar la correctitud del mecanismo generador de números aleatorios, realizamos un experimento piloto con dimensiones pequeñas, esto es, ejecutando solamente 5 corridas de todas las permutaciones posibles de los niveles de las variables.

El diseño del experimento completo queda de la siguiente manera: Para cada combinación de niveles, de cada variable, se hicieron 30 experimentos independientes (corridas), usando diferente semilla aleatoria, y lo aplicamos a la versión secuencial del algoritmo, utilizando el conjunto de 13 funciones de prueba definidas en el Apéndice B de este documento. Al comparar estos resultados con los obtenidos por el experimento piloto, comprobamos

la consistencia de éstos, de manera que podemos concluir que el mecanismo generador de números aleatorios de nuestro programa no está sesgado.

Resultados obtenidos

A continuación mostraremos los resultados obtenidos por este análisis para cada una de las funciones de prueba definidos en el Apéndice B de este documento.

Consideramos que un determinado parámetro tiene efecto real sobre el desempeño del algoritmo si la probabilidad de que este sea debido al azar es menor a 0.001 En la tabla 4.1 mostramos estas probabilidades para las funciones de prueba. Es importante notar que para las funciones g01, g08 y g12 no se realizó el estudio debido a que el algoritmo siempre llega a la misma solución, por lo que no existe una varianza. Ver tabla 4.3.

En las figuras 4.5, 4.6, 4.9, 4.10, 4.11, 4.12, 4.13, 4.7 y 4.8 mostramos únicamente los parámetros cuyo efecto sobre el desempeño del algoritmo no es causa del azar.

En el eje de las ordenadas se muestra que tan lejos del valor óptimo están en promedio las soluciones encontradas por nuestro algoritmo. En el eje de las abscisas se muestran los valores (niveles) que puede tomar el parámetro.

Realizando una generalización de estos resultados, podemos enumerar las siguientes conclusiones:

1. El valor asignado al porcentaje de mutación es el que tiene mayor efecto en el desempeño del algoritmo. La probabilidad de que su efecto sea producto del azar es menor a 0.00001.

2. Los parámetros tamaño de población y tipo de cruza, ocupan el segundo lugar en importancia del efecto que tienen en el desempeño del algoritmo.

3. Finalmente, los dos parámetros: tamaño σ de la muestra de anticuerpos y generaciones del AG interno parecen no tener un efecto significativo sobre el desempeño del algoritmo.

De acuerdo al resultado obtenido de este estudio estadístico, podemos afirmar que en general, nuestro algoritmo tiene un mejor desempeño cuando

el porcentaje de mutación es dinámico (valores 2 y 3 en el experimento). Por otro lado, los tipos de cruza de 2 puntos y uniforme parecen mejorar el desempeño del algoritmo.

4.7. Resultados

Comparamos los resultados de la versión secuencial y paralela de nuestro algoritmo contra tres técnicas que son de las más competitivas en el área (ver sección 3.1.2).

Esas técnicas son:

- Los mapas homomorfos, que es una técnica basada en un AG, propuesta por Koziel y Michalewicz [79]
- El método de ordenamiento estocástico (*stochastic ranking*) basado en una estrategia evolutiva, propuesto por Runarsson y Yao [113] y,
- ASCHEA (*Adaptive Segregational Constraint Handling Evolutionary Algorithm*) que es una estrategia evolutiva propuesta por Hamida y Schoenauer [52].

Además, presentaremos los resultados obtenidos por el algoritmo de Hajela y Yoo [51] en el cual se basa este trabajo.

Los algoritmos se aplicaron al conjunto de 13 funciones de prueba definidos en el Apéndice B de este documento. Este conjunto de funciones pretende tener una mezcla de varios tipos de funciones (lineales, cúbicas, cuadráticas, polinomiales, no lineales) con diferentes cantidades de variables y restricciones.

Para tener una idea de la dificultad de generar puntos factibles en estas funciones de prueba, se realiza el cálculo del valor ρ, propuesto por Koziel y Michalewicz [79]. El procedimiento consiste en generar aleatoriamente un millón de puntos y contar el porcentaje de ellos que está dentro de la zona factible. Los valores para ρ y las características generales de las funciones se muestran en la tabla 4.2.

Nuestro algoritmo evalúa 350,000 veces la función de aptitud al igual que el ordenamiento estocástico. Por su parte, ASCHEA realiza 1,500,000

y los mapas homomorfos 1,400,000 evaluaciones de la función objetivo del problema. El algoritmo de Hajela y Yoo [51] evalua 350,000 veces la función objetivo.

De acuerdo a las conclusiones del análisis de varianza, los parámetros aplicados en todas las corridas de nuestro algoritmo secuencial son:

- Generaciones del AG interno=30,
- tamaño de la población=30,
- tipo de cruza: dos puntos,
- porcentaje de mutación: dinámico, iniciando en 0.4 y finalizando en (1/L), donde L es la longitud de la cadena binaria,
- porcentaje de cruza=0.8.

Los resultados obtenidos por cada uno de ellos se muestran en la tablas 4.3, 4.4, 4.5, 4.6 y 4.7.

4.8. Análisis de resultados

Para las 13 funciones de prueba el sistema inmune artificial (SIA) logró encontrar soluciones factibles.

En siete funciones de prueba el programa del sistema inmune artificial logró encontrar la solución óptima, ellas son las funciones g01, g03, g05, g07, g08, g11 y g12 (tabla 4.3). Para las funciones g02, g04, g06, g09 y g13 el algoritmo obtiene buenas aproximaciones al valor óptimo. Los peores resultados se obtienen para el caso de la función g10, donde nuestro algoritmo tiene un desempeño pobre; ésta es la única función lineal del conjunto.

En 5 funciones (g01, g03, g05, g06, g10) el SIA supera, tanto en promedio como en mejor valor, a los mapas homomorfos (MH) (tabla 4.4), mientras que se ve superado por éste en una función (g02). Para la función g07 el sistema inmune artificial (SIA) obtiene una mejor solución, aunque se ve superado, en promedio por los mapas homomorfos (MH). Para la función g05 los MH no logran encontrar ninguna solución factible. En 5 funciones

Función	n	**Tipo de** f	ρ	DL	IN	DN
g01	13	cuadrática	0.0111 %	9	0	0
g02	20	no lineal	99.8474 %	0	0	2
g03	10	polinomial	0.0000 %	0	1	0
g04	5	cuadrática	52.1230 %	0	0	6
g05	4	cúbica	0.0000 %	2	3	0
g06	2	cúbica	0.0066 %	0	0	2
g07	10	cuadrática	0.0003 %	3	0	5
g08	2	no lineal	0.856 %	0	0	2
g09	7	polinomial	0.5121 %	0	0	4
g10	8	lineal	0.0010 %	3	0	3
g11	2	cuadrática	0.0000 %	0	1	0
g12	3	cuadrática	0.0654 %	0	0	1
g13	5	no lineal	0.0000 %	0	3	0

Cuadro 4.2: Cálculo de ρ para las funciones de prueba. DL=restricciones con desigualdades lineales, IN=restricciones de igualdad no lineales y DN=restricciones con desigualdades no lineales, n es el número de variables del problema.

Función	Óptimo	Mejor	Promedio	Peor	Desv.Est.
g01	-15.0	**-15.0**	-15.0	-15.0	0.0
g02	0.803619	0.770337	0.704021	0.595632	0.0449
g03	1.0	**1.00**	0.997	0.981	0.006
g04	-30665.5	-30665.0815	-30648.175046	-30613.442569	14.0335
g05	5126.4981	5126.686105	5307.201594	5927.367117	249.6318
g06	-6961.814	-6961.179206	-6959.550307	-6955.603388	1.2393
g07	24.306	24.332397	31.514070	47.214202	4.3983
g08	0.095825	**0.095825**	0.095825	0.095825	0.0
g09	680.63	680.831650	682.193733	688.603687	1.4476
g10	7049.33	7133.1280	8158.9658	9493.8894	677.29
g11	0.75	**0.7500**	0.7505	0.7516	0.0004
g12	-1.0	**-1.0**	-1.0	-1.0	0.0
g13	0.05395	0.06716	1.29267	14.71544	2.57983

Cuadro 4.3: Resultados obtenidos por el sistema inmune artificial

el desempeño de ambos algoritmos es similar (funciones g04, g08, g09, g11). De manera que, en general, el SIA tiene un mejor desempeño que los mapas homomorfos, realizando apenas una cuarta parte de las evaluaciones de la función objetivo que éste efectua.

El SIA es superado por la técnica de ordenamiento estocástico (OE) en las 4 funciones g02, g09, g10 y g13, y tiene un desempeño similar a éste en otras 4 funciones: g01, g08, g11 y g12. (tabla 4.5). En 4 funciones más (g03, g04, g05, g07), ambos algoritmos logran encontrar la solución óptima. Sin embargo, el OE obtiene mejores soluciones promedio que el SIA. Por otro lado, para la función g06, el OE obtiene una mejor solución que el SIA, pero se ve superado por éste en promedio. La técnica de OE supera el desempeño del SIA, realizando la misma cantidad de evaluaciones de la función objetivo (350,000).

ASCHEA supera al SIA en tres funciones que son g04, g05 y g10 (tabla 4.6). Para el caso de la función g01 ambos logran llegar al valor óptimo, aunque el SIA obtiene mejores valores en promedio. En g02, en prome-

Función	Óptimo	Mejor	Promedio	Peor	Desv.Est.
g01	-15.0	-14.7864	-14.7082	-14.6154	N.D.
g02	0.803619	0.79953	0.79671	0.79119	N.D.
g03	1.0	0.9997	0.9989	0.9978	N.D.
g04	-30665.5	-30664.5	-30655.3	-30645.9	N.D.
g05	5126.4981	-	-	-	N.D.
g06	-6961.814	-6952.1	-6342.6	-5473.9	N.D.
g07	24.306	24.620	24.826	25.069	N.D.
g08	0.095825	**0.095825**	0.0891568	0.0291438	N.D.
g09	680.63	680.91	681.16	683.18	N.D.
g10	7049.33	7147.9	8163.6	9659.3	N.D.
g11	0.75	**0.75**	0.75	0.75	N.D.
g12	-1.0	0.9999	0.99913	.9919	N.D.
g13	0.05395	N.D.	N.D.	N.D.	N.D.

Cuadro 4.4: Resultados obtenidos por los mapas homomorfos. N.D. = no disponible

Función	Óptimo	Mejor	Promedio	Peor	Desv.Est.
g01	-15.0	**-15.0**	-15.0	-15.0	0.0E+00
g02	0.803619	0.803515	0.7858	0.726288	2.0E-02
g03	1.0	**1.0**	1.0	1.0	1.9E-04
g04	-30665.539	**-30665.539**	-30665.539	-30665.539	2.0E-05
g05	5126.498	5126.497	5128.881	5142.472	3.5E+00
g06	-6961.814	**-6961.814**	-6875.940	-6350.262	1.6E+02
g07	24.306	24.307	24.374	24.642	6.6E-02
g08	0.095825	**0.095825**	0.095825	0.095825	2.6E-17
g09	680.63	**680.630**	680.656	680.763	3.4E-02
g10	7049.33	7054.316	7559.192	8835.655	5.3E+02
g11	0.75	**0.75**	0.75	0.75	8.0E-05
g12	-1.0	**-1.0**	-1.0	-1.0	0.0E+00
g13	0.05395	**0.053957**	0.067543	0.216915	3.1E-02

Cuadro 4.5: Resultados obtenidos por el método de ordenamiento estocástico.

Función	Óptimo	Mejor	Promedio	Peor
g01	-15.0	**-15.0**	-14.84	N.D.
g02	0.803619	0.785	0.59	N.D.
g03	1.0	**1.0**	0.99989	N.D.
g04	-30665.5	**-30665.5**	-30665.5	N.D.
g05	5126.4981	5126.5	5141.65	N.D.
g06	-6961.814	**-6961.81**	-6961.81	N.D.
g07	24.306	24.3323	24.6636	N.D.
g08	0.095825	**0.095825**	0.095825	N.D.
g09	680.63	**680.630**	680.641	N.D.
g10	7049.33	7061.13	7497.434	N.D.
g11	0.75	**0.75**	0.75	N.D.
g12	-1.0	N.D.	N.D.	N.D.
g13	0.05395	N.D.	N.D.	N.D.

Cuadro 4.6: Resultados obtenidos por ASCHEA. N.D. = no disponible.

dio es mejor el SIA, pero ASCHEA logra encontrar una mejor solución. Ambos algoritmos tienen un desempeño similar en las funciones g03, g08 y g11. En las funciones g06, g07 y g09 ambos encuentran el valor óptimo de la función, aunque ASCHEA obtiene un mejor promedio.

En general, ASCHEA tiene un desempeño similar al del SIA, aunque para algunas funciones, ASCHEA logra un mejor promedio en las soluciones encontradas. Sin embargo, se hace notar que el SIA realiza aproximadamente una cuarta parte de las evaluaciones de la función objetivo que las efectuadas por ASCHEA.

Nótese que los autores de los mapas homomorfos no reportan resultados para la función g13. Esto es debido a que g13 se introdujo en el artículo de Runarsson y Yao [113], cuya fecha de publicación es posterior a la de los mapas homomorfos.

En la técnica de ASCHEA, las funciones g12 y g13 tampoco fueron incluidas, aunque en este caso por razones desconocidas.

Función	Óptimo	Mejor	Promedio	Peor	Desv.Est.	P.F.
g01	-15.0	-5.4000	-3.7845	-1.7458	1.037	10 %
g02	0.803619	0.1322	0.2455	0.3746	0.0611	35 %
g03	1.0	0.00	0.00	0.00	0.00	50 %
g04	-30665.5	-30324.81	-26684.73	-24493.35	1986.1	60 %
g05	5126.4981	INF	INF	INF	INF	0 %
g06	-6961.814	INF	INF	INF	INF	0 %
g07	24.306	37.1127	99.9452	379.7398	97.3613	33 %
g08	0.095825	INF	INF	INF	INF	0 %
g09	680.63	685.4961	875.6067	1931.3254	305.8862	90 %
g10	7049.33	INF	INF	INF	INF	0 %
g11	0.75	0.8129	0.9713	1.0000	0.0643	20 %
g12	-1.0	-1.0	-1.0	-1.0	0.0	100 %
g13	0.05395	INF	INF	INF	INF	0 %

Cuadro 4.7: Resultados obtenidos por el algoritmo de Hajela y Yoo. INF=resultados infactibles, P.F.=Porcentaje de soluciones factibles encontradas.

En la tabla 4.7 observamos que el algoritmo de Hajela y Yoo tiene dificultad para encontrar soluciones factibles. Unicamente en la función g12 encuentra soluciones factibles óptimas en todas las ocasiones. Para las funciones g04, g09 y g11 cuando se logra encontrar soluciones factibles, éstas tienen una aproximación al valor óptimo relativamente buena. En las funciones g05, g06, g08, g10 y g13 nunca se encuentran soluciones factibles. En general, este algoritmo no obtiene buenos resultados.

Versión paralela del algoritmo.

Los resultados obtenidos por la versión paralela del SIA se muestran en las tablas de la 4.8 a la 4.24.

En las tablas podemos observar que las funciones g01,g08, g11 y g12 para la versión paralela del SIA, tienen exactamente las mismas soluciones que las de la versión secuencial al encontrar siempre la solución óptima. La ganancia del paralelo es solamente en términos del tiempo de ejecución.

Para las funciones g02 y g07, observamos que la versión paralela tiene mejores soluciones promedio, mientras que el mejor valor se obtiene con la versión secuencial.

En el caso de las funciones g09 y g13 la mejor solución encontrada por ambas versiones es similar. Sin embargo, la versión paralela mejora las soluciones en promedio. Con la función g13, al incrementar el número de procesadores la solución encontrada está más alejada del óptimo.

En las funciones g03, g04, g05 y g06, las soluciones (mejores y promedio) mejoran con la versión paralela del algoritmo. Sin embargo, a medida que se incrementan los procesadores, la calidad de los resultados se decrementa.

La versión paralela del algoritmo, para la función g10, encuentra consistentemente mejores soluciones que la versión secuencial.

Con respecto a las aceleraciones alcanzadas por el algoritmo paralelo, podemos observar que ésta se ve disminuida de manera importante para el caso de la función g10. Esto puede deberse a que el algoritmo tiene mayor dificultad en encontrar buenas soluciones, lo que provoca que la población de anticuerpos sea grande aumentando así el tiempo de procesamiento del

	2 P A=1.63	3 P A=2.24	4 P A=3.44	5 P A=4.21	6 P A=5.07	7 P A=6.12	8 P A=7.21
Mejor	-15.00	-15.00	-15.00	-15.00	-15.00	-15.00	-15.00
Promedio	-15.00	-15.00	-15.00	-15.00	-15.00	-15.00	-15.00
Peor	-15.00	-15.00	-15.00	-15.00	-15.00	-15.00	-15.00
Desv.Est.	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Cuadro 4.8: Resultados del SIA paralelo para la función g01. P=procesadores, A=aceleración

	2 P A=1.59	3 P A=2.31	4 P A=2.72	5 P A=3.85	6 P A=4.45	7 P A=5.92	8 P A=6.64
Mejor	0.7179	0.7611	0.7740	0.7431	0.7660	0.7737	0.7666
Promedio	0.5998	0.6594	0.6882	0.6367	0.6747	0.7141	0.6779
Peor	0.3564	0.5237	0.5484	0.5104	0.5718	0.6244	0.6092
Desv.Est.	0.0913	0.0709	0.0546	0.0668	0.0560	0.0400	0.0409

Cuadro 4.9: Resultados del SIA paralelo para la función g02

AG interno. Para el resto de las funciones las aceleraciones se encuentran dentro de los rangos esperados.

4.9. Conclusiones

En este capítulo se desarrolló un sistema inmune artificial para manejo de restricciones dentro de un algoritmo genético. Este sistema inmune a su vez, también está basado en un algoritmo genético y es conceptualmente muy simple de implementar.

Nuestra propuesta se basa en el algoritmo propuesto por Hajela et al. [50, 51], intentando superar algunas de sus principales desventajas. Las diferencias entre ambos algoritmos fueron descritas en el capítulo enfatizándose la forma en que el algoritmo aquí propuesto mejora al de Hajela et al.

	2 P A=1.69	3 P A=2.94	4 P A=3.15	5 P A=4.28	6 P A=5.66	7 P A=6.78	8 P A=7.02
Mejor	1.0046	1.0024	1.0032	1.0036	1.0012	0.9936	0.9798
Promedio	1.0001	0.9844	0.9816	0.9820	0.9632	0.9249	0.8528
Peor	0.9794	0.9226	0.9177	0.8777	0.8323	0.6528	0.3364
Desv.Est.	0.0058	0.0240	0.0298	0.0316	0.0499	0.0913	0.1683

Cuadro 4.10: Resultados del SIA paralelo para la función g03

	2 P A=1.80	3 P A=2.56	4 P A=3.15	5 P A=4.20	6 P A=5.32
Mejor	-30665.06	-30665.058	-30663.903	-30664.712	-30663.099
Promedio	-30632.13	-30645.19	-30634.395	-30646.213	-30639.039
Peor	-30418.344	-30565.848	-30553.826	-30577.953	-30578.147
Desv.Est.	64.764	28.583	35.366	25.316	24.799

Cuadro 4.11: Resultados del SIA paralelo para la función g04 (Parte I)

	7 P A=6.24	8 P A=7.35
Mejor	-30662.218	-30658.698
Promedio	-30616.10	-30597.545
Peor	-30500.113	-30469.519
Desv.Est.	43.816	55.563

Cuadro 4.12: Resultados del SIA paralelo para la función g04 (Parte II).

	2 P A=1.93	3 P A=2.17	4 P (60 %) A=3.48	5 P	6 P	7 P	8 P
Mejor	5126.557	5127.106	5126.716	Inf	Inf	Inf	Inf
Promedio	5306.958	5262.486	5162.507	Inf	Inf	Inf	Inf
Peor	6088.653	5776.449	5311.989	Inf	Inf	Inf	Inf
Desv.Est.	290.947	187.875	53.238	Inf	Inf	Inf	Inf

Cuadro 4.13: Resultados del SIA paralelo para la función g05. Inf=soluciones infactibles. 60 % (en 4P) indica que sólo ese porcentaje de soluciones alcanzaron la zona factible.

	2 P A=1.94	3 P A=2.46	4 P A=3.45	5 P A=4.46
Mejor	-6961.723	-6961.755	-6961.262	-6960.077
Promedio	-6961.074	-6961.175	-6954.458	-6953.078
Peor	-6959.779	-6960.261	-6945.559	-6936.408
Desv.Est.	0.474	0.437	4.963	7.702

Cuadro 4.14: Resultados del SIA paralelo para la función g06. Parte I

	6 P A=5.39	7 P A=6.40	8 P A=7.18
Mejor	-6959.167	-6961.138	-6960.990
Promedio	-6954.370	-6954.175	-6955.098
Peor	-6945.253	-6931.428	-6941.946
Desv.Est.	4.135	6.551	5.058

Cuadro 4.15: Resultados del SIA paralelo para la función g06. Parte II

	2 P A=1.48	3 P A=2.66	4 P A=3.38	5 P A=4.44	6 P A=5.42	7 P A=6.31	8 P A=7.11
Mejor	24.665	25.056	25.147	24.684	25.115	24.728	24.720
Promedio	27.748	26.810	26.776	27.923	28.298	26.963	27.191
Peor	31.801	30.256	33.343	33.109	41.789	34.658	33.050
Desv.Est.	2.241	1.220	1.986	2.095	3.697	2.203	2.388

Cuadro 4.16: Resultados del SIA paralelo para la función g07

	2 P A=1.74	3 P A=2.68	4 P A=3.64	5 P A=4.23
Mejor	0.095825	0.095825	0.095825	0.095825
Promedio	0.095825	0.095825	0.095825	0.095825
Peor	0.095825	0.095825	0.095825	0.095825
Desv.Est.	0.00	0.00	0.00	0.00

Cuadro 4.17: Resultados del SIA paralelo para la función g08. Parte I.

Con la finalidad de determinar cuáles parámetros tienen un mayor efecto en el desempeño del algoritmo, llevamos a cabo un análisis de varianza. De aquí concluimos que el parámetro que estadísticamente tiene mayor efecto es el porcentaje de mutación. El segundo lugar lo ocupan el tipo de cruza y el tamaño de la población.

Nuestra versión secuencial del algoritmo del SIA fue comparada contra otras tres técnicas que son representantivas del estado del arte del área, que son los mapas homomorfos, ordenamiento estocástico y ASCHEA, usando un conjunto de 13 funciones de prueba que se suelen adoptar en el área. De esta comparación podemos concluir que la mejor técnica es el ordenamiento estocástico. El segundo lugar lo ocupa nuestro sistema inmune artificial. El siguiente lugar lo ocupa ASCHEA, que aunque obtiene buenas soluciones, tiene un costo computacional elevado (en términos de la evaluación de la función objetivo). Y el último lugar lo ocupa la técnica de los mapas homomorfos.

	6 P A=5.48	7 P A=6.67	8 P A=7.39
Mejor	0.095825	0.095825	0.095825
Promedio	0.095825	0.095825	0.095825
Peor	0.095825	0.095825	0.095825
Desv.Est.	0.00	0.00	0.00

Cuadro 4.18: Resultados del SIA paralelo para la función g08. Parte II.

	2 P A=1.25	3 P A=2.01	4 P A=2.79	5 P A=3.55	6 P A=4.01	7 P A=4.99	8 P A=5.98
Mejor	680.689	680.845	680.807	680.740	680.703	680.809	680.987
Promedio	681.524	681.540	681.360	681.831	681.685	681.542	681.904
Peor	683.281	683.410	682.206	684.376	683.072	684.030	684.424
Desv.Est.	0.590	0.644	0.382	0.943	0.616	0.733	0.957

Cuadro 4.19: Resultados del SIA paralelo para la función g09

	2 P A=1.38	3 P A=1.64	4 P A=1.94	5 P A=2.15
Mejor	7141.0371	7071.880	7053.991	7070.715
Promedio	8476.618	7640.843	7341.393	7364.321
Peor	11677.299	8754.897	8191.945	9035.333
Desv.Est.	1145.37	534.96	316.43	480.834

Cuadro 4.20: Resultados del SIA paralelo para la función g10. Parte I

A la fecha, los mapas homorfos son la mejor técnica basada en un algoritmo genético para manejo de restricciones (recordemos que el ordenamiento estocástico y ASCHEA se basan en una estrategia evolutiva). Al igual que éste, nuestro SIA también está basado en un algoritmo genético y, como observamos en los resultados, nuestra técnica supera a los mapas homomorfos con un costo computacional menor, siendo además un algoritmo bastante más simple.

Finalmente, se realizó una versión paralela del algoritmo, usando un modelo de AG de múltiples poblaciones (también llamado AG de grano grueso). En este caso se usaron diferentes operadores genéticos en cada una de las subpoblaciones y se experimentó usando desde 2 hasta 8 procesadores paralelos.

En los resultados obtenidos por nuestra versión paralela del algoritmo notamos que la calidad de las soluciones encontradas es mejor. Esto se debe, principalmente, a la diversidad en los operadores genéticos que se aplican a cada deme, ya que cada uno de ellos explora diferentes regiones del espacio de búsqueda. Además, el proceso de migración entre los demes ayuda a que las buenas soluciones se expandan rápidamente a través de sus vecinos.

	6 P A=2.44	7 P A=2.96	8 P A=3.21
Mejor	7050.136	7098.774	7053.796
Promedio	7194.541	7335.460	7305.224
Peor	7473.153	7911.420	7997.591
Desv.Est.	119.269	270.723	257.168

Cuadro 4.21: Resultados del SIA paralelo para la función g10. Parte II

	2 P A=1.47	3 P A=1.88	4 P A=2.44	5 P A=2.99	6 P A=3.57	7 P A=4.98	8 P A=6.01
Mejor	0.761	0.754	0.750	0.751	0.750	0.753	0.751
Promedio	0.750	0.749	0.749	0.749	0.749	0.749	0.749
Peor	0.749	0.749	0.749	0.749	0.749	0.749	0.749
Desv.Est.	0.003	0.001	0.000	0.000	0.000	0.001	0.000

Cuadro 4.22: Resultados del SIA paralelo para la función g11

	2 P A=1.77	3 P A=2.49	4 P A=3.31	5 P A=4.35	6 P A=5.26	7 P A=6.44	8 P A=7.51
Mejor	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
Promedio	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
Peor	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
Desv.Est.	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Cuadro 4.23: Resultados del SIA paralelo para la función g12

	2P A=1.89	3P A=2.12	4P A=2.99	5P (90 %) A=3.48	6P (30 %) A=4.95	7P(30 %) A=6.07	8P
Mejor	0.110653	0.059516	0.061356	0.091489	0.056475	0.229090	Inf
Promedio	0.568839	0.516819	0.414923	0.678472	0.694915	0.668966	Inf
Peor	1.948252	1.200161	0.854664	3.268368	0.981620	1.010560	Inf
Desv.Est.	0.3977	0.3267	0.2716	0.7211	0.3482	0.3009	Inf

Cuadro 4.24: Resultados del SIA paralelo para la función g13. Inf=soluciones infactibles. Los valores de % (junto a 5P, 6P y 7P) indican que sólo ese porcentaje de soluciones alcanzaron la zona factible.

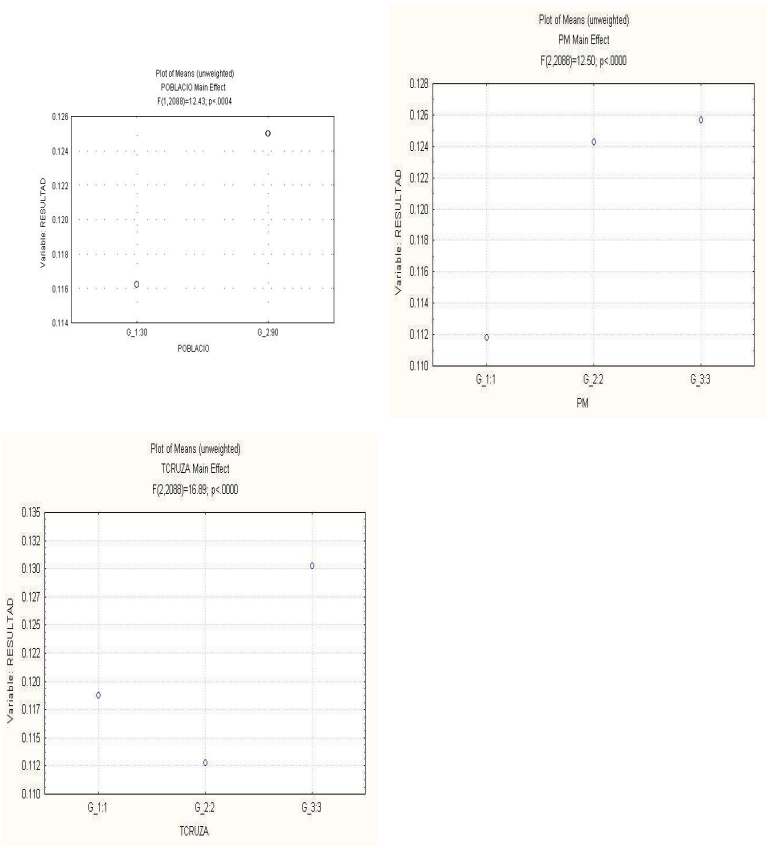

Figura 4.5: Gráficas de los parámetros N Pm y Tc que tienen mayor efecto sobre el desempeño del algoritmo en la función g02. En el eje de las ordenadas se muestra la diferencia de las soluciones encontradas con el valor óptimo (en promedio). En las abscisas los niveles para el parámetro.

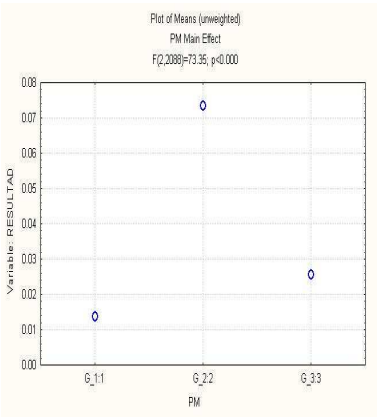

Figura 4.6: Gráfica del parámetro Pm que tiene mayor efecto sobre el desempeño del algoritmo en la función g03. En el eje de las ordenadas se muestra la diferencia de las soluciones encontradas con el valor óptimo (en promedio). En las abscisas los niveles para el parámetro.

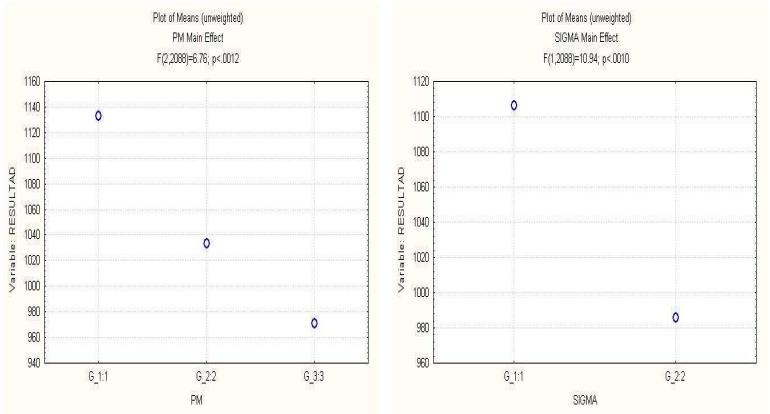

Figura 4.7: Gráficas de los parámetros Pm y σ que tienen mayor efecto sobre el desempeño del algoritmo en la función g10. En el eje de las ordenadas se muestra la diferencia de las soluciones encontradas con el valor óptimo (en promedio). En las abscisas los niveles para el parámetro.

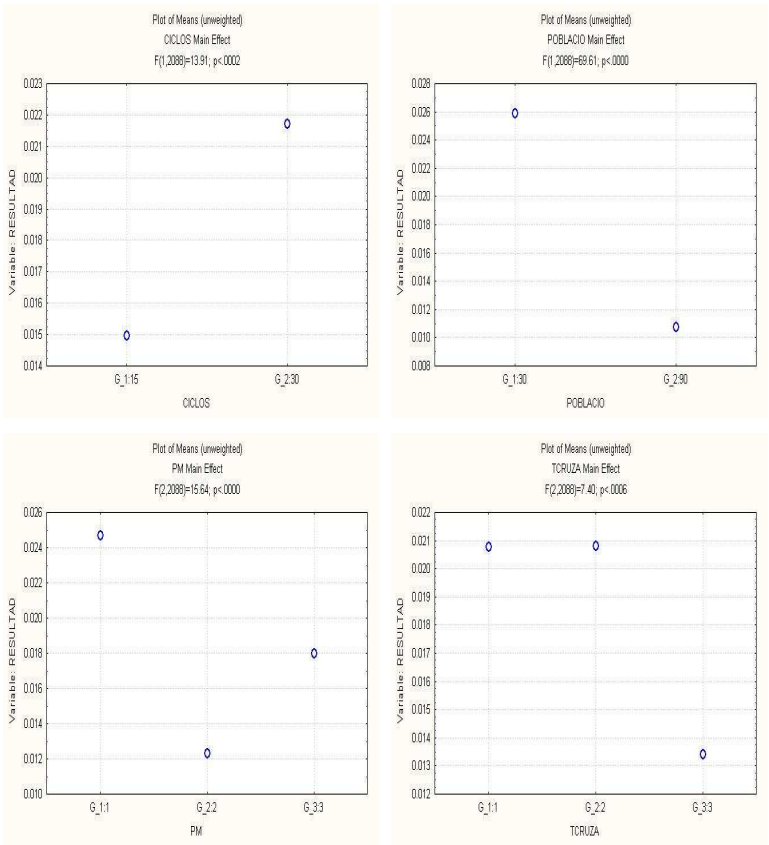

Figura 4.8: Gráficas de los parámetros Tint, N Pm y Tc que tienen mayor efecto sobre el desempeño del algoritmo en la función g11. En el eje de las ordenadas se muestra la diferencia de las soluciones encontradas con el valor óptimo (en promedio). En las abscisas los niveles para el parámetro.

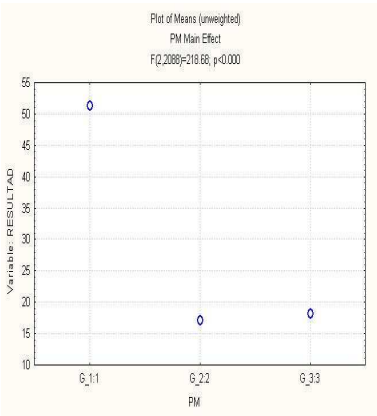

Figura 4.9: Gráfica del parámetro Pm que tiene mayor efecto sobre el desempeño del algoritmo en la función g04. En el eje de las ordenadas se muestra la diferencia de las soluciones encontradas con el valor óptimo (en promedio). En las abscisas los niveles para el parámetro.

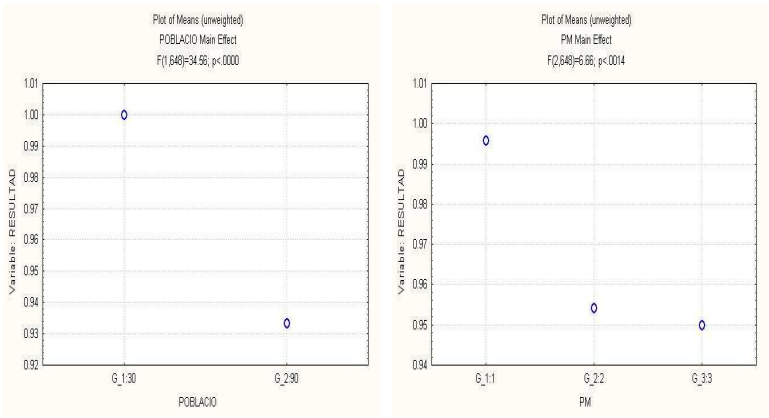

Figura 4.10: Gráficas de los parámetros N y Pm que tienen mayor efecto sobre el desempeño del algoritmo en la función g05. En el eje de las ordenadas se muestra la diferencia de las soluciones encontradas con el valor óptimo (en promedio). En las abscisas los niveles para el parámetro.

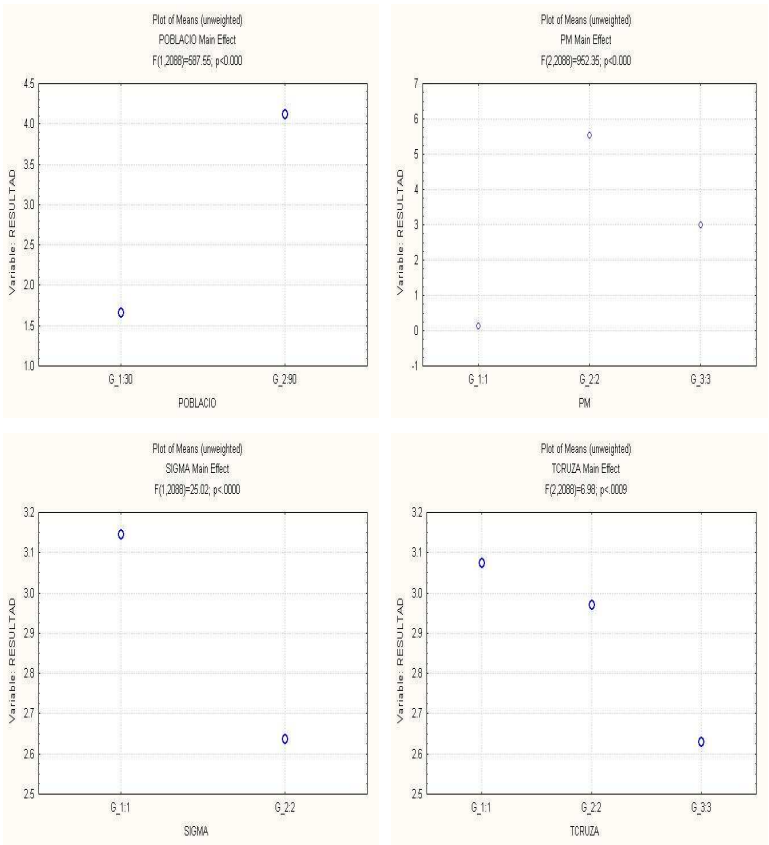

Figura 4.11: Gráficas de los parámetros N, Pm σ y Tc que tienen mayor efecto sobre el desempeño del algoritmo en la función g06. En el eje de las ordenadas se muestra la diferencia de las soluciones encontradas con el valor óptimo (en promedio). En las abscisas los niveles para el parámetro.

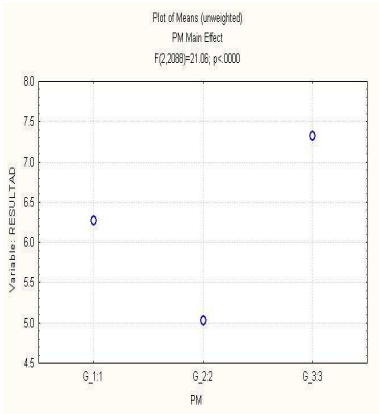

Figura 4.12: Gráfica del parámetro Pm que tiene mayor efecto sobre el desempeño del algoritmo en la función g07. En el eje de las ordenadas se muestra la diferencia de las soluciones encontradas con el valor óptimo (en promedio). En las abscisas los niveles para el parámetro.

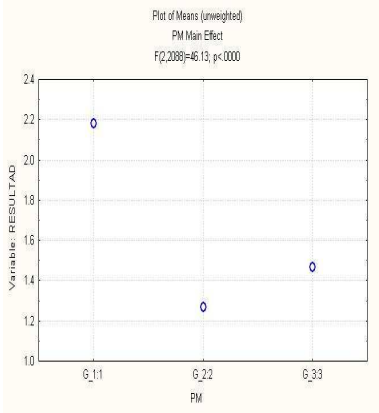

Figura 4.13: Gráfica del parámetro Pm que tiene mayor efecto sobre el desempeño del algoritmo en la función g09. En el eje de las ordenadas se muestra la diferencia de las soluciones encontradas con el valor óptimo (en promedio). En las abscisas los niveles para el parámetro.

Capítulo 5

Sistema inmune artificial para optimización multiobjetivo

5.1. Descripción del problema

En optimización multiobjetivo deseamos optimizar de manera simultánea varias funciones que generalmente se encuentran en conflicto entre sí. Estas funciones pueden o no tener restricciones. La solución para los problemas de optimización multiobjetivo no es única, sino que está compuesta por un conjunto de posibles soluciones al que se le conoce con el nombre de conjunto de óptimos de Pareto. La definición formal del problema que deseamos solucionar es la siguiente ver sección 3.3:

$$\text{Encontrar} \quad \vec{x} \text{ que optimice } \vec{f}(\vec{x}) = [f_1\vec{x}, f_2\vec{x}, \ldots, f_k\vec{x}] \tag{5.1}$$

sujeto a:

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, n \tag{5.2}$$

$$h_j(\vec{x}) = 0, \quad j = 1, \ldots, p \tag{5.3}$$

donde $\vec{x}$ es el vector de soluciones $\vec{x} = [x_1, x_2, \ldots, x_r]^T$, n es el número de restricciones de desigualdad y p es el número de restricciones de igualdad (en ambos casos las restricciones pueden ser lineales o no lineales). Llamaremos Q al conjunto de restricciones tanto de igualdad ($g(\vec{x})$) como de desigualdad ($h(\vec{x})$).

Esencialmente deseamos superar tres dificultades principales dentro de optimización multiobjetivo que son:

- Mantener diversidad
- Manejar restricciones
- Lidiar con varios objetivos a la vez.

5.2. Principio inmunológico utilizado

Para solucionar el problema de optimización multiobjetivo utilizamos el principio de selección clonal del sistema inmune [10] que se describió en la sección 2.3.3.

Cuando la presencia de un antígeno es detectada por el organismo, únicamente los anticuerpos que le son más afines (los de mayor aptitud) son estimulados. Estos anticuerpos estimulados se reproducen por clonación, es decir, crean múltiples copias de sí mismos. Los nuevos clones son sometidos a un proceso de mutación (llamada hipermutación) de lo que puede resultar que algunos de ellos incrementen su afinidad hacia el antígeno, siendo éstos los que proceden a neutralizar y eliminar al antígeno. Al finalizar este proceso, el sistema inmune debe regresar a sus niveles normales eliminando el excedente de células.

Por otro lado, los anticuerpos que no son estimulados, mueren y son reemplazados por nuevos anticuerpos creados de manera aleatoria, con la finalidad de mantener la diversidad en la población.

5.3. Adaptación del sistema inmune artificial

Elementos básicos de un SIA

Los principales componentes de nuestro sistema inmune artificial los definimos de la siguiente manera (ver sección 2.3.2):

- **Representación de los componentes**: Los componentes del sistema son una población de anticuerpos representados mediante cadenas

binarias que codifican las variables del problema, usando códigos de Gray y son soluciones potenciales al problema.

Las funciones objetivo del problema toman el papel del antígeno, dado que son la meta que deseamos que los anticuerpos alcancen.

- **Mecanismos para evaluar las interrelaciones entre individuos y el ambiente**: Existen dos tipos de interacciones entre anticuerpos en el algoritmo propuesto. La primera de ellas es una relación de dominancia de Pareto y la segunda es de acuerdo a su distribución y proximidad medida en el espacio de las funciones objetivo.

- **Proceso de adaptación que domine las dinámicas del sistema**: Principio de selección clonal (selección, clonación, hipermutación, memoria y autoregulación)

El algoritmo del sistema inmune artificial para optimización multiobjetivo propuesto es el siguiente:

Entrada del algoritmo:

- El antígeno representado por las funciones objetivo del problema $\vec{f}(\vec{x})$.
- Restricciones Q del problema.
- Tamaño de la población de anticuerpos n.
- T iteraciones del algoritmo.
- Divisiones de la rejilla de la memoria secundaria ν

INICIO

1. La población inicial tamaño (n) de anticuerpos se genera de manera aleatoria.

2. La memoria secundaria se inicializa en cero, es decir, vacía (esta memoria secundaria se explicará más adelante).

3. Se evalúan las funciones objetivo $f(\vec{x})$ y las restricciones del problema Q para los n anticuerpos para determinar dominancia de Pareto y factibilidad. En el caso de espacios de búsqueda restringidos, la dominancia se mide comparando únicamente anticuerpos factibles contra factibles, e infactibles contra infactibles.

4. Se ordenan los n anticuerpos de mejores a peores. El criterio para este ordenamiento es el siguiente (de más importante a menos importante):

   - Anticuerpos factibles no dominados.
   - Factibles dominados, colocando primero aquellos que son dominados por una menor cantidad de anticuerpos.
   - Infactibles no dominados, colocando primero a los que violan en menor cantidad a las restricciones del problema.
   - Infactibles dominados, colocando primero a los que son dominados por una menor cantidad de anticuerpos.

5. Asignar a los mejores anticuerpos para clonarse ($Ac.Estimulados$). Usar el criterio siguiente:

   - Si (cantidad de anticuerpos factibles no dominados $< 5\,\%$ de n) entonces, seleccionar al $5\,\%$ de los mejores anticuerpos.
   - Si no, seleccionar a todos los anticuerpos factibles no dominados.

6. $Ac.Estimulados$ se copian a la memoria secundaria.

7. Para cada miembro de $Ac.Estimulados$ determinar su cantidad de clones ($\#clones_i$) de la siguiente manera:

   - Calcular $temporal$ utilizando : $temporal = \frac{n \times 0{,}60}{Ac.Estimulados}$. Tenemos dos casos posibles:
   - Si la memoria secundaria no se ha saturado:

     a) El espacio ocupado por $Ac.Estimulados$ se divide en hipercubos de dimensión d, donde d=número de funciones objetivo del problema.

b) Determinar el promedio de población de los hipercubos.

c) Para cada ($Ac.Estimulado_i$), determinar:

- Si $Ac.Estimulado_i$ pertenece a un hipercubo con población menor a promedio entonces, $\#clones_i = 1{,}5 \times temporal$
- si no, $\#clones_i = 0{,}5 \times temporal$

- Si la memoria secundaria está saturada:

a) Si $Ac.Estimulado_i$ pertenece a una celda de la memoria secundaria muy saturada (arriba del promedio) entonces $\#clones_i = 0{,}5 \times temporal$

b) si no entonces, $\#clones_i = 1{,}5 \times temporal$

8. Cada $Ac.Estimulado_i$ crea $\#clones_i$ copias de sí mismo. Llamaremos $Clones$ al conjunto de todos ellos.

9. Se aplica hipermutación a los $Clones$ de la siguiente manera:

   - A los clones que son copias de anticuerpos factibles no dominados, se les muta b bits seleccionados aleatoriamente, donde b es igual al número de variables del problema.
   - A los clones copiados de anticuerpos factibles, y que son dominados por q individuos, se les muta $b+q$ bits.
   - Los clones de anticuerpos infactibles no dominados se mutan en $b+q+1$ bits.
   - Los clones de anticuerpos infactibles dominados por m individuos se mutan en $b+q+1+m$ bits.

10. Aplicar mutación uniforme a los anticuerpos que no fueron estimulados ($Ac.Noestimulados$). El porcentaje de mutación al inicio de la ejecución del algoritmo es $pm = 0{,}6$ y al final $pm = 1/L$, donde L es la longitud de la cadena binaria. Los decrementos en pm se hacen de manera uniforme a lo largo de la ejecución.

11. La población en este momento está compuesta por $(Ac.Estimulados + Clones + Ac.Noestimulados)$. Ordenarla de mejores a peores de la siguiente manera:

    - Factibles no dominados.
    - Factibles dominados, colocando primero aquellos que son dominados por una menor cantidad de individuos.
    - Infactibles no dominados, colocando primero a los que violan en menor cantidad a las restricciones del problema.
    - Infactibles dominados, colocando primero a los que son dominados por una menor cantidad de individuos.

12. Seleccionar a los mejores n individuos para regresar la población a su tamaño original.

13. Ir al paso 5 T veces.

FIN
Salida del algoritmo: Conjunto de vectores $\vec{x}$ de la memoria secundaria.

En este esquema, los mecanismos de *clonación* e *hipermutación* realizan la explotación del espacio de búsqueda, es decir promueven la búsqueda local, mientras que la mutación de los anticuerpos no estimulados realiza la búsqueda global del espacio y mantiene diversidad en la población.

El número de clones que se crean de cada anticuerpo estimulado, así como el porcentaje de hipermutación que se les aplica, está en relación a qué tan bueno es el anticuerpo. De tal manera, creamos una mayor cantidad de clones para los mejores anticuerpos y les aplicamos hipermutación con valores menores [71].

En el paso (7) del algoritmo se calcula la cantidad de clones que se deben crear para cada anticuerpo estimulado. En este punto establecemos una relación de proximidad entre anticuerpos, de manera que consideramos que son mejores aquellos anticuerpos que están menos próximos entre sí, de los que creamos una cantidad mayor de clones. Asímismo, para los anticuerpos

de regiones más pobladas creamos menos anticuerpos. Cuando la memoria secundaria no se ha llenado esta información se extrae de la población principal del algoritmo, mientras que cuando la memoria secundaria se ha saturado, la información se extrae de ésta, evitando así la repetición del cálculo.

En el paso (9) se calcula el número de bits que se mutan por cada clon. Mutamos menos bits para los clones de los mejores anticuerpos. La cantidad de bits que se mutan en los clones va aumentando a medida que los anticuerpos estimulados son peores.

A continuación explicaremos la memoria secundaria a la que se hizo referencia en los pasos 2, 6 y 7 del algoritmo.

Memoria secundaria

En nuestro algoritmo utilizamos un archivo externo con la finalidad de guardar las mejores soluciones globales encontradas a lo largo de su ejecución. A este archivo externo le denominamos memoria secundaria. Al mecanismo de mantener las mejores soluciones globales se le conoce con el nombre de *elitismo*.

El contenido del archivo externo representa la solución del algoritmo para el problema. Todos los puntos almacenados son no dominados entre ellos y representan la aproximación final del frente de Pareto que el algoritmo reporta al usuario.

Idealmente, el tamaño de la memoria secundaria debiera ser infinito, pero al no ser esto posible en la práctica, debemos imponerle un límite. De manera que cuando la rejilla se satura, es necesario definir algún criterio para saber a cuáles individuos se les permite la entrada y a cuáles no. Para ello adoptamos la *rejilla adaptativa* propuesta por Knowles y Corne [77]. Esta rejilla busca una buena distribución de las soluciones a lo largo del frente de Pareto. La rejilla adaptativa se ilustra en la figura 5.1. La idea principal es dividir el espacio ocupado por las soluciones existentes (medido en el espacio de las funciones objetivo) en cierto número de celdas o hipercubos para determinar cuáles regiones son las más pobladas.

Decimos que la rejilla es adaptativa porque su tamaño no es fijo sino que

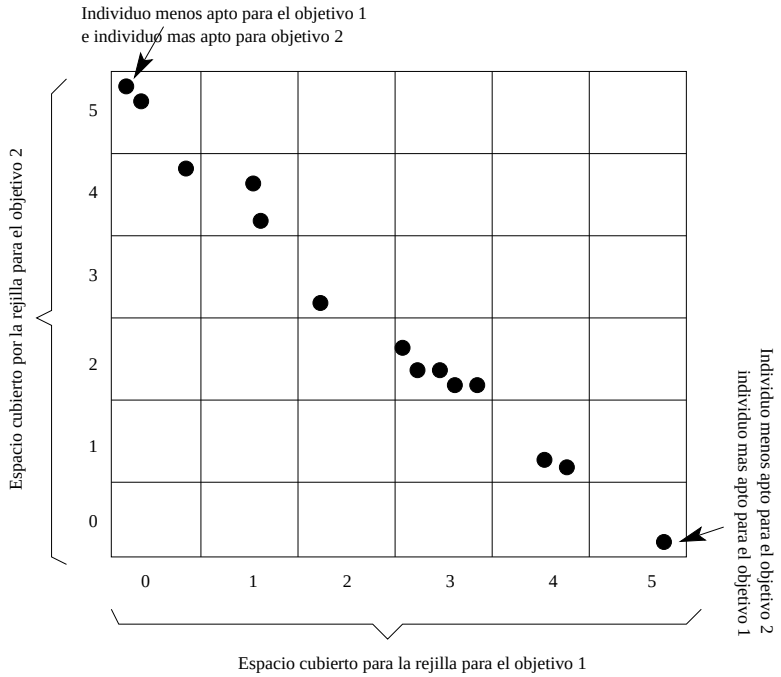

Figura 5.1: Rejilla adaptativa de la memoria secundaria adoptada para nuestro algoritmo de optimización multiobjetivo.

se ajusta a los valores que almacena. Es decir, crece y decrece de acuerdo a los valores extremos de su contenido.

El algoritmo para la implementación de la rejilla adaptativa es el siguiente:

1. Dividir el espacio de las funciones objetivo de acuerdo a un número de subdivisiones establecido por el usuario, digamos ν.

2. Para cada individuo en la memoria secundaria, determinar a qué celda pertenece.

3. Para determinar si a un cierto anticuerpo le es permitido ingresar en la memoria secundaria cuando ésta se encuentra llena, hacer lo siguiente:

    - Si pertenece a la celda más saturada, entonces no se le permite

ingresar.

- En otro caso, sí se le permite ingresar. El individuo que será reemplazado es seleccionado de manera aleatoria de la casilla más saturada.

En la siguiente sección mostraremos los resultados obtenidos de realizar pruebas utilizando diferentes valores para ν que es el parámetro que determina las divisiones de la rejilla adaptativa (paso (1) del algoritmo de la rejilla).

5.4. Experimentos

Cuando se mide el desempeño de un algoritmo evolutivo multiobjetivo, son tres los aspectos más relevantes de cuantificar [142]:

- Cercanía al verdadero frente de Pareto.
- Dispersión uniforme.
- Maximizar la cantidad de elementos diferentes en el verdadero frente de Pareto.

Para comparar diferentes algoritmos de manera cuantitativa, utilizaremos tres métricas que miden cada uno de los aspectos mencionados. Estas métricas son: tasa de error, dispersión y distancia generacional, que definiremos a continuación:

Tasa de error (TE): Propuesta por Van Veldhuizen [135] para indicar el porcentaje de soluciones del frente Pareto actual (FP_{actual}) que no son miembros del verdadero frente de Pareto ($FP_{verdadero}$):

$$TE = \frac{\sum_{i=1}^{n} e_i}{n} \tag{5.4}$$

donde n es el número de vectores en FP_{actual}, $e_i = 0$ si el vector i es un miembro de $FP_{verdadero}$ y $e_i = 1$ de lo contrario. Un valor de $TE = 0$

indica el comportamiento ideal del algoritmo.

Dispersión (D): Esta métrica fue propuesta por Schott [121], y mide la varianza de la distancia de cada miembro del

conjunto de óptimos de Pareto con respecto a su vecino más cercano:

$$D = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2}\,, \tag{5.5}$$

donde $d_i = \text{mín}_j(|\, f_1^i(\vec{x}) - f_1^j(\vec{x}) \,| + |\, f_2^i(\vec{x}) - f_2^j(\vec{x}) \,|)$, $i, j = 1, \ldots, n$, $\bar{d}$ es el promedio de todos d_i, y n. f_1 y f_2 son dos funciones objetivo del problema, sin embargo, en caso de existir, se deben incluir todas. es el número de elementos del conjunto de Pareto obtenidos hasta el momento. Si $D = 0$, significa que nuestro algoritmo ha encontrado la distribución ideal de vectores no dominados.

Distancia Generacional (DG): El concepto de distancia generacional fue introducido por Van Veldhuizen y Lamont [135] como una manera de estimar qué tan lejos están los elementos FP_{actual} de $FP_{verdadero}$ y se define como:

$$DG = \frac{(\sum_{i=1}^{n} d_i^p)^{1/p}}{n} \tag{5.6}$$

donde n es el número de vectores no dominados encontrados por el algoritmo que está siendo analizado, $p = 2$ y d_i es la distancia Euclidiana entre cada uno de ellos y el miembro más cercano del verdadero frente de Pareto. Si $GD = 0$ indica que todos los elementos generados están en el verdadero frente de Pareto de la función.

La validación de algoritmo se llevó a cabo aplicándolo a cinco funciones de prueba. La descripción de las funciones de ejemplo se proporciona en el Apéndice C de este documento.

Realizamos una comparación entre los resultados arrojados por nuestra propuesta contra tres algoritmos representativos del estado del arte del área que son: NSGA-II [127], PAES [77] y el micro-AG [20] [21]. Realizamos 20 ejecuciones de todos los algoritmos y les aplicamos las métricas definidas

anteriormente. Realizamos el mismo número de evaluaciones de la función de aptitud para todos los casos.

Para evitar alguna confusión con la métrica de distancia generacional la aplicamos, tanto del frente encontrado por el algoritmo hacia el verdadero frente de Pareto como en sentido inverso, es decir del verdadero frente hacia la solución del algoritmo. Esto es importante porque la métrica mide la distancia que existe entre un punto y aquél del verdadero frente que se encuentre más cercano. De manera que, si el frente encontrado por el algoritmo se encuentra cubriendo

únicamente una porción del verdadero frente de Pareto, entonces el valor de la métrica será muy bajo, lo que haría suponer que el desempeño del algoritmo fue muy bueno, a pesar de ser esto erróneo. Esto lo podemos detectar, al hacer la comparación de manera inversa (del verdadero frente hacia el algoritmo) pues mediremos la distancia que existe entre los puntos del verdadero frente con aquellos más cercanos producidos por el algoritmo, lo cual implica que éstos se encuentran distribuidos a lo largo del verdadero frente.

Para realizar una comparación más justa, en todos los experimentos efectuamos el mismo número de evaluaciones de la función de aptitud para todos los algoritmos (12,000).

Además, realizamos una comparación gráfica del verdadero frente de Pareto (el cual fue obtenido por enumeración) contra la corrida con valor más cercano al promedio de la métrica de distancia generacional. Para nuestro sistema inmune artificial (SIA) mostramos los resultados obtenidos al probar diferentes cantidades de divisiones o celdas para la memoria secundaria. Esto tiene el propósito de evaluar el impacto de este parámetro en el desempeño del algoritmo.

A continuación se muestran las gráficas y las estadísticas de los resultados obtenidos en este experimento.

5.5. Análisis de resultados

Los valores para la métricas del ejemplo 1 (definido en el Apéndice C) se encuentran en los cuadros 5.1 y 5.2. La comparación gráfica entre el

Métrica de tasa de error

# divisiones	5	15	25	35	45
Promedio	0.3475	0.381	0.428	0.404	0.519
Mejor	0.2	0.22	0.34	0.15	0.41
Peor	0.47	0.49	0.51	0.56	0.65
Desv. Est.	0.07866	0.0733	0.0464	0.09304	0.0583

Métrica de dispersión

# divisiones	5	15	25	35	45
Promedio	0.0112	0.0091	0.0082	**0.0075**	0.0190
Mejor	0.0084	0.0072	0.0072	0.0061	0.0063
Peor	0.0146	0.0115	0.0090	0.0086	0.2335
Desv. Est.	0.0019	0.0010	0.0006	0.0007	0.0505

Métrica de distancia generacional

# divisiones	5	15	25	35	45
Promedio	0.0002	0.0002	0.0002	0.0002	0.0014
Mejor	0.0002	0.0002	0.0002	0.0002	0.0002
Peor	0.0003	0.0003	0.0003	0.0003	0.0238
Desv. Est.	0.0000	0.0000	0.0000	0.0000	0.0053

Métrica de distancia generacional inversa (del verdadero frente de Pareto hacia el SIA)

# divisiones	5	15	25	35	45
Promedio	0.0006	0.0005	0.0004	0.0004	0.0004
Mejor	0.0005	0.0004	0.0004	0.0004	0.0003
Peor	0.0009	0.0007	0.0006	0.0005	0.0005
Desv. Est.	0.0001	0.0001	0.0006	0.0000	0.0000

Cuadro 5.1: SIA con diferente número de divisiones de la rejilla adaptativa para el ejemplo 1.

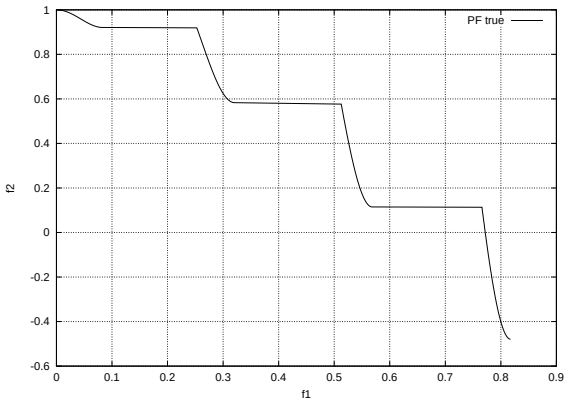

Figura 5.2: Verdadero frente de Pareto para el ejemplo 1 (las líneas horizontales no forman parte del frente, sino únicamente las líneas verticales)

verdadero frente de Pareto y la solución promedio de los

algoritmos aplicados se encuentra en las figuras 5.2, 5.3, 5.4, 5.5 y 5.6.

Para el ejemplo 1 observamos que en la métrica de tasa de error, en promedio, el mejor desempeño lo obtuvieron PAES y el NSGA-II, mientras que el segundo lugar lo ocupa el SIA y el último lugar es ocupado por el micro-AG.

Para la métrica de dispersión el primer lugar lo ocupan las técnicas de SIA y el NSGA-II con valores de desempeño en promedio y desviación estándar muy similares (menores a 0.0001), el segundo el micro-AG y el último lugar PAES. Observamos que utilizando rejillas con 35 y 25 divisiones para el SIA los valores de esta métrica mejoran, sin embargo cuando se usa 45 divisiones el valor empeora. Para el resto de las métricas el valor de las divisiones de la rejilla no parece afectar el desempeño del algoritmo.

En la métrica de distancia generacional el mejor desempeño lo tiene el SIA en promedio y desviación estándar, seguido por NSGA-II y PAES y finalmente el micro-AG.

En la inversa de la distancia generacional tenemos que, el primer lugar lo tiene el NSGA-II y el segundo el SIA, siguiéndoles el micro-AG y PAES. De estos valores podemos concluir que una gran parte de puntos del SIA no logran llegar al verdadero frente de Pareto, aunque quedan muy cercanos a

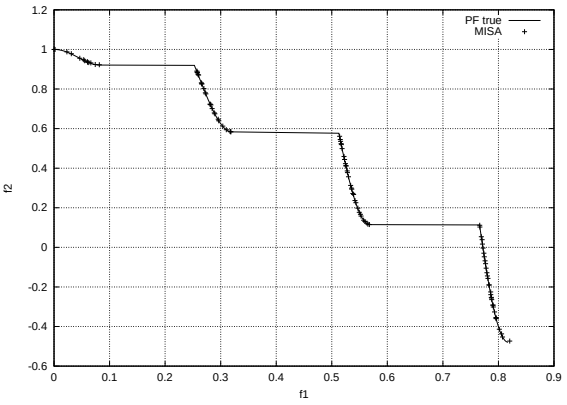

Figura 5.3: Frente de Pareto obtenido por el sistema inmune artificial (SIA) para el ejemplo 1

éste; es por ello que los valores de la distancia generacional son muy bajos. El micro-AG también tiene una gran cantidad de soluciones que no logran exactamente el óptimo, sin embargo quedan muy cerca, lo cual podemos comprobar gráficamente. Gráficamente observamos que tanto PAES como el micro-AG tienen bastantes espacios sin cubrir, y algunos puntos se ven separados del verdadero frente de Pareto.

En términos generales, para este ejemplo 1 el mejor desempeño lo tuvo el NSGA-II y el segundo lugar el SIA.

Los valores para la métricas del ejemplo 2 se encuentran en las tablas 5.3 y 5.4. La comparación gráfica entre el verdadero frente de Pareto y la solución promedio de los algoritmos aplicados se encuentra en las figuras 5.7, 5.8, 5.9, 5.10 y 5.11.

En el ejemplo 2, para la métrica de tasa de error los cuatro algoritmos se comportan de manera muy similar, con diferencias menores a 0.001.

La métrica de dispersión muestra que la mejor distribución la logra el SIA, mientras que PAES presenta el peor valor. Cuando la rejilla tiene 25 divisiones vemos que el valor de esta métrica mejora de manera considerable. Por otro lado, si usamos sólo 5 divisiones el valor empeora.

En la distancia generacional, observamos valores

muy cercanos (con diferencias menores a 0.001), aunque el micro-AG

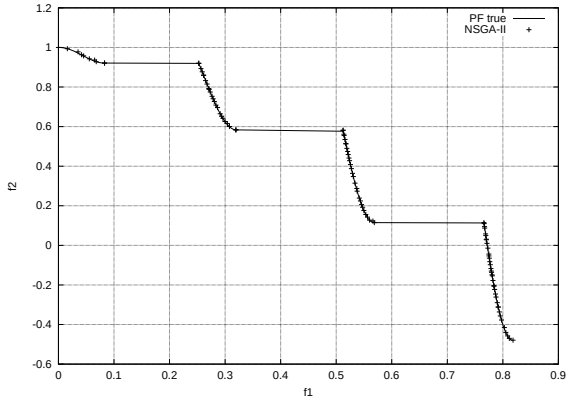

Figura 5.4: Frente de Pareto obtenido por el NSGA-II para el ejemplo 1

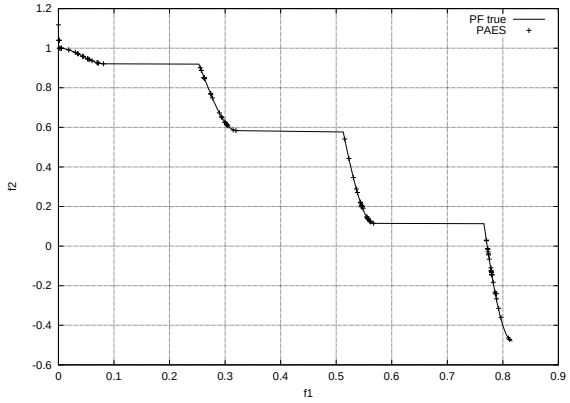

Figura 5.5: Frente de Pareto obtenido por PAES para el ejemplo 1

presenta una desviación estándar mayor.

En lo que respecta a la métrica inversa de la distancia generacional NSGA-II, SIA y el micro-AG presentan un desempeño muy similar. PAES presenta los peores valores. Esto lo podemos comprobar gráficamente, pues PAES sólo es capaz de producir un segmento del verdadero frente de Pareto.

Los valores para la métricas del ejemplo 3 se encuentran en los cuadros 5.5 y 5.6. La comparación gráfica entre el verdadero frente de Pareto y la solución promedio de los algoritmos aplicados se encuentra en las figuras 5.12, 5.13, 5.14, 5.15 y 5.16.

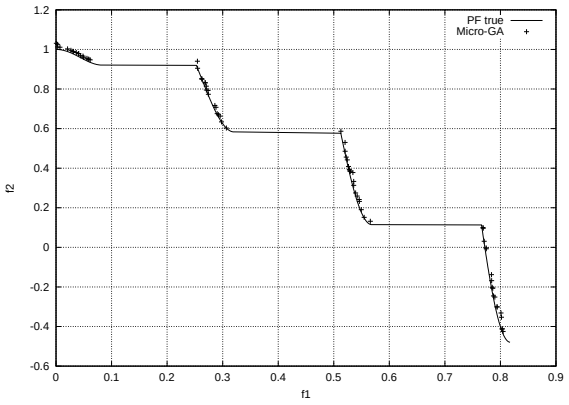

Figura 5.6: Frente de Pareto obtenido por el micro-AG para el ejemplo 1

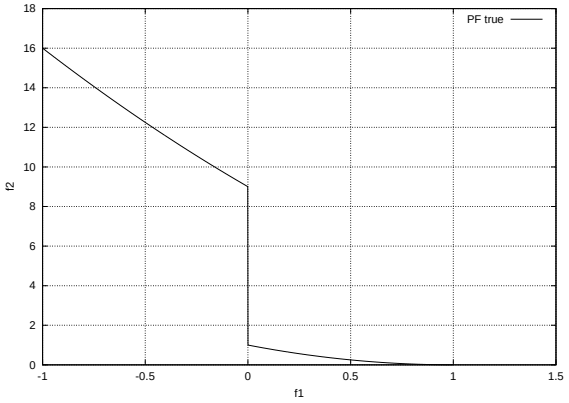

Figura 5.7: Verdadero frente de Pareto para el ejemplo 2

Para el ejemplo 3 la métrica de tasa de error da los mejores resultados para el algoritmo de NSGA-II y el SIA con valores muy cercanos. El siguiente lugar lo ocupa el micro-AG y finalmente PAES.

Para la métrica de dispersión el mejor valor en promedio se obtiene con NSGA-II seguido por los algoritmos SIA, micro-AG y PAES con valores muy similares. El SIA obtiene mejores distribuciones cuando se usan 25 ó 35 divisiones de la rejilla de la memoria secundaria.

En lo que respecta a la distancia generacional, el SIA obtiene las mejores aproximaciones, después PAES, seguidos del micro-AG y finalmente NSGA-II, aunque los valores para los cuatro algoritmos son muy cercanos.

	Tasa de error			Dispersión		
	NSGA-II	PAES	MicroAG	NSGA-II	PAES	MicroAG
Promedio	0.14250	0.13300	0.58251	0.00760	0.03229	0.19539
Mejor	0.08000	0.06000	0.46753	0.00657	0.01246	0.04934
Peor	0.29000	0.26000	0.93333	0.00920	0.25179	0.50707
Desv. Est.	0.05035	0.05121	0.17467	0.00065	0.05240	0.13807
	Distancia generacional			**Distancia generacional inversa**		
	NSGA-II	PAES	MicroAG	NSGA-II	PAES	MicroAG
Promedio	0.00044	0.00304	0.03724	0.00030	0.00105	0.00431
Mejor	0.00021	0.00012	0.03047	0.00029	0.00052	0.00057
Peor	0.00443	0.02520	0.05079	0.00033	0.00308	0.00976
Desv. Est.	0.00094	0.00758	0.00766	0.00033	0.00308	0.00457

Cuadro 5.2: Valores de las métricas del ejemplo 1 para NSGA-II, PAES y MicroAG.

Para la inversa de la distancia generacional, los cuatro algoritmos obtienen valores muy próximos.

En general, para esta función de ejemplo podemos decir que los cuatro algoritmos tienen desempeños muy similares.

Los valores para la métricas de la función del ejemplo 4 se encuentran en los cuadros 5.7 y 5.8. La comparación gráfica entre el verdadero frente de Pareto y la solución promedio de los algoritmos aplicados se encuentra en las figuras 5.17, 5.18, 5.19, 5.20 y 5.21.

En el ejemplo 4 el mejor valor de la tasa de error lo obtiene el SIA, con un valor de casi el doble, el segundo lugar lo ocupa PAES, y las técnicas NSGA-II y micro-AG obtienen valores muy elevados, por lo que quedan en último lugar.

La mejor dispersión es para el NSGA-II, el segundo lugar para SIA y luego PAES, quedando en último lugar el micro-AG. Los mejores valores para el SIA se obtienen cuando la rejilla tiene 35 y 25 divisiones.

Con respecto a la distancia generacional el mejor valor en promedio

Métrica de tasa de error

# divisiones	5	15	25	35	45
Promedio	0.0100	0.0120	0.0110	0.0105	0.0150
Mejor	0.0000	0.0000	0.0000	0.0000	0.0000
Peor	0.0200	0.0300	0.0500	0.0300	0.0300
Desv. Est.	0.0073	0.0089	0.0120	0.0089	0.0100

Métrica de dispersión

# divisiones	5	15	25	35	45
Promedio	0.1272	0.0862	0.857	0.0888	0.0877
Mejor	0.0787	0.0428	0.0401	0.0414	0.0387
Peor	0.1928	0.1959	0.225	0.2142	0.2061
Desv. Est.	0.0399	0.0356	0.046	0.0479	0.0456

Métrica de distancia generacional

# divisiones	5	15	25	35	45
Promedio	0.0046	0.0035	0.0038	0.0042	0.0037
Mejor	0.0002	0.0002	0.0002	0.0002	0.0002
Peor	0.0131	0.0100	0.0143	0.0146	0.0157
Desv. Est.	0.0050	0.0043	0.0048	0.0050	0.0050

Métrica de distancia generacional inversa (del verdadero frente de Pareto hacia el SIA)

# divisiones	5	15	25	35	45
Promedio	0.0044	0.0025	0.0023	0.0021	0.0020
Mejor	0.0031	0.0019	0.0016	0.0016	0.0011
Peor	0.0078	0.0031	0.0027	0.0026	0.0023
Desv. Est.	0.0014	0.0003	0.0003	0.0002	0.0002

Cuadro 5.3: SIA con diferente número de divisiones de la rejilla adaptativa para el ejemplo 2.

	Tasa de error			Dispersión		
	NSGA-II	PAES	MicroAG	NSGA-II	PAES	MicroAG
Promedio	0.01500	0.01800	0.01550	0.04422	0.17232	0.06556
Mejor	0.00000	0.00000	0.01000	0.03906	0.01003	0.05073
Peor	0.05000	0.10000	0.03000	0.05035	0.53812	0.16065
Desv. Est.	0.01277	0.02726	0.00605	0.00292	0.13835	0.02466
	Distancia generacional			**Distancia generacional inversa**		
	NSGA-II	PAES	MicroAG	NSGA-II	PAES	MicroAG
Promedio	0.00246	0.00359	0.00283	0.00135	0.03580	0.00217
Mejor	0.00027	0.00005	0.00028	0.00089	0.00404	0.00202
Peor	0.01309	0.03494	0.01562	0.00214	0.23517	0.00219
Desv. Est.	0.00344	0.00812	0.01562	0.00044	0.06743	0.00004

Cuadro 5.4: Valores de las métricas del ejemplo 2 para NSGA-II, PAES y MicroAG.

	Tasa de error			Dispersión		
	NSGA-II	PAES	MicroAG	NSGA-II	PAES	MicroAG
Promedio	0.9370	0.8920	0.9280	0.3709	0.4816	0.4990
Mejor	0.8600	0.7000	0.8900	0.3001	0.3184	0.4310
Peor	0.9800	1.0000	0.9700	0.4997	0.6560	0.6223
Desv. Est.	0.0279	0.0914	0.0194	0.0818	0.0628	0.0481
	Distancia generacional			**Distancia generacional inversa**		
	NSGA-II	PAES	MicroAG	NSGA-II	PAES	MicroAG
Promedio	0.1607	0.1254	0.1399	0.0384	0.0389	0.0354
Mejor	0.1328	0.0875	0.1315	0.0350	0.0361	0.0345
Peor	0.1722	0.1621	0.1475	0.0474	0.0457	0.0364
Desv. Est.	0.0105	0.0175	0.0050	0.0027	0.0021	0.0005

Cuadro 5.5: Valores de las métricas del ejemplo 3 para NSGA-II, PAES y MicroAG.

Métrica de tasa de error

# divisiones	5	15	25	35	45
Promedio	0.9190	0.9515	0.9550	0.9450	0.9465
Mejor	0.8800	0.9000	0.9400	0.9000	0.8900
Peor	0.9600	0.9900	0.9800	0.9800	0.9800
Desv. Est.	0.0288	0.0243	0.0124	0.0209	0.0252

Métrica de dispersión

# divisiones	5	15	25	35	45
Promedio	0.4447	0.4710	0.4611	0.4317	0.4838
Mejor	0.3276	0.3749	0.3383	0.3166	0.3129
Peor	0.5428	0.5572	0.5154	0.6188	0.6812
Desv. Est.	0.0658	0.0557	0.0491	0.0666	0.0786

Métrica de distancia generacional

# divisiones	5	15	25	35	45
Promedio	0.0374	0.0399	0.0373	0.0367	0.0389
Mejor	0.0290	0.0298	0.0295	0.0294	0.0296
Peor	0.0444	0.0452	0.0430	0.0438	0.0448
Desv. Est.	0.0053	0.0041	0.0034	0.0044	0.0039

Métrica de distancia generacional inversa (del verdadero frente de Pareto hacia el SIA)

# divisiones	5	15	25	35	45
Promedio	0.0336	0.0336	0.0338	0.0339	0.0338
Mejor	0.0328	0.0330	0.0332	0.0332	0.0330
Peor	0.0340	0.0343	0.0344	0.0349	0.0348
Desv. Est.	0.0003	0.0003	0.0003	0.0004	0.0004

Cuadro 5.6: SIA con diferente número de divisiones de la rejilla adaptativa para el ejemplo 3.

Métrica de tasa de error

# divisiones	5	15	25	35	45
Promedio	0.3275	0.3915	0.3785	0.4255	0.4085
Mejor	0.2300	0.2800	0.2900	0.3300	0.2700
Peor	0.4300	0.4700	0.5000	0.5500	0.5400
Desv. Est.	0.0527	0.0453	0.0544	0.0638	0.0610

Métrica de dispersión

# divisiones	5	15	25	35	45
Promedio	0.1814	0.2332	0.1795	0.0904	0.2215
Mejor	0.0683	0.0512	0.0593	0.0530	0.0454
Peor	1.6486	1.4950	0.7178	0.1984	1.2302
Desv. Est.	0.3468	0.3575	0.2184	0.0386	0.3173

Métrica de distancia generacional

# divisiones	5	15	25	35	45
Promedio	0.0608	0.0612	0.0530	0.0228	0.0235
Mejor	0.0039	0.0035	0.0028	0.0033	0.0027
Peor	0.3162	0.1783	0.1745	0.1173	0.1236
Desv. Est.	0.0763	0.0597	0.0512	0.0334	0.0344

Métrica de distancia generacional inversa

# divisiones	5	15	25	35	45
Promedio	0.0078	0.0052	0.0057	0.0048	0.0040
Mejor	0.0039	0.0041	0.0039	0.0033	0.0031
Peor	0.0164	0.0075	0.0070	0.0178	0.0069
Desv. Est.	0.0029	0.0010	0.0010	0.0033	0.0009

Cuadro 5.7: SIA con diferente número de divisiones de la rejilla adaptativa para el ejemplo 4

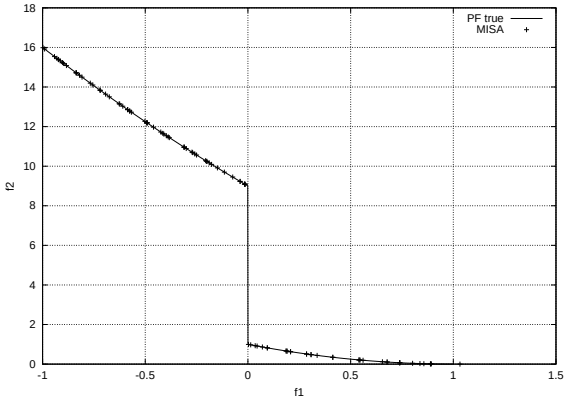

Figura 5.8: Frente de Pareto obtenido por el SIA para el ejemplo 2

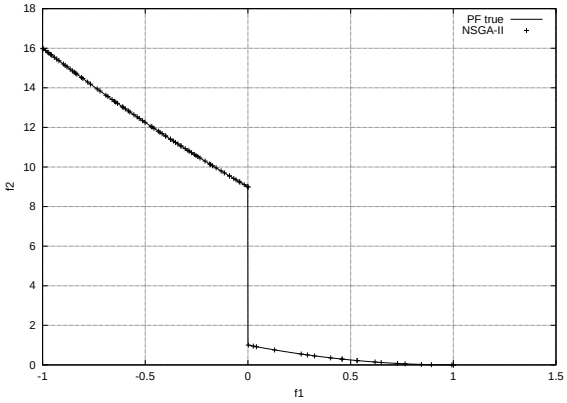

Figura 5.9: Frente de Pareto obtenido por el NSGA-II para el ejemplo 2

lo tiene el SIA seguido muy de cerca por el NSGA-II. Sin embargo, la desviación estándar de NSGA-II es muy grande. Las gráficas nos muestran la corrida promedio de esta métrica por lo que en ella observamos una gran cantidad de puntos lejos del verdadero frente de Pareto. El tercer lugar lo ocupa PAES y luego el micro-AG. Nótese que la desviación estándar del SIA es más pequeña que la de las otras técnicas (casi la mitad).

Para la distancia generacional inversa el mejor valor es para SIA, el segundo lugar lo ocupan el micro-AG y PAES aunque los valores de éstos están bastante alejados de SIA. El último lugar es para el NSGA-II, cuyo valor es casi 5 veces mayor al del micro-AG y PAES. De aquí concluimos

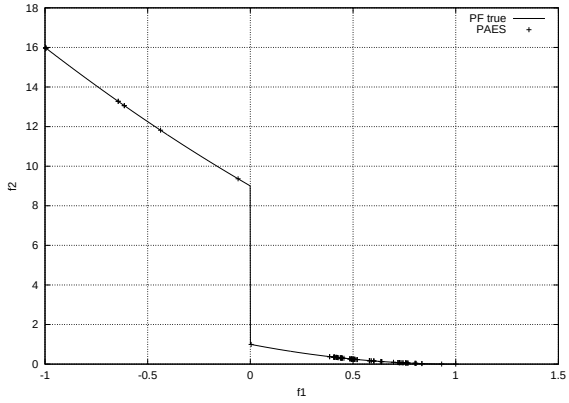

Figura 5.10: Frente de Pareto obtenido por PAES para el ejemplo 2

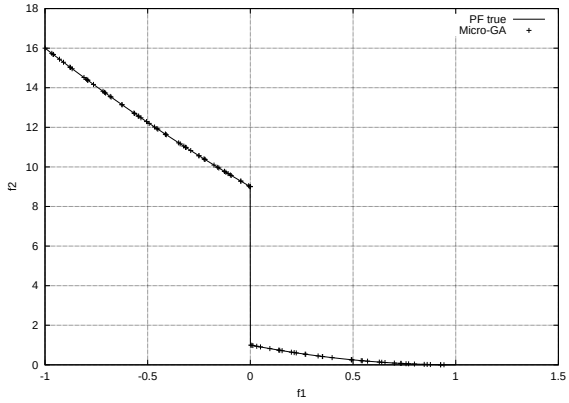

Figura 5.11: Frente de Pareto obtenido por el micro-AG para el ejemplo 2

que NSGA-II pierde una parte del verdadero frente de Pareto, lo cual lo verificamos en las gráficas.

Para esta función de ejemplo número 4, el SIA es el que muestra un mejor desempeño, y el peor lo tiene el NSGA-II.

Los valores para la métricas del ejemplo 5 se encuentran en los cuadros 5.9 y 5.10. La comparación gráfica entre el verdadero frente de Pareto y la solución promedio de los algoritmos aplicados se encuentra en las figuras 5.22, 5.23, 5.24, 5.25 y 5.26.

Para el ejemplo 5, el mejor valor de la tasa de error lo tiene el NSGA-II,

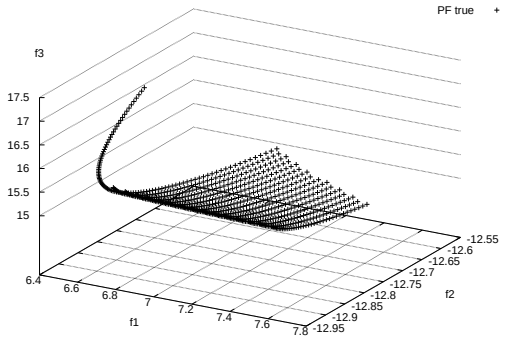

Figura 5.12: Verdadero frente de Pareto para el ejemplo 3

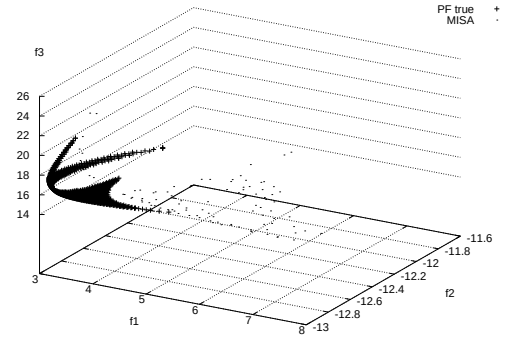

Figura 5.13: Frente de Pareto obtenido por el SIA para el ejemplo 3

seguido por PAES. El tercer lugar es para el SIA y el último para el micro-AG. La desviación estándar de PAES, sin embargo, es muy superior a la de los demás.

La mejor distribución la obtienen PAES y el SIA, mientras que el segundo lugar lo ocupan NSGA-II y el micro-AG, aunque sus valores son muy cercanos. El SIA obtiene los mejores valores para esta métrica cuando la rejilla tiene 15 y 25 divisiones.

En la distancia generacional el primer lugar lo

ocupan PAES y el SIA, seguido muy cercanamente por el NSGA-II y el micro-AG. Sus desviaciones estándar tienen valores muy similares.

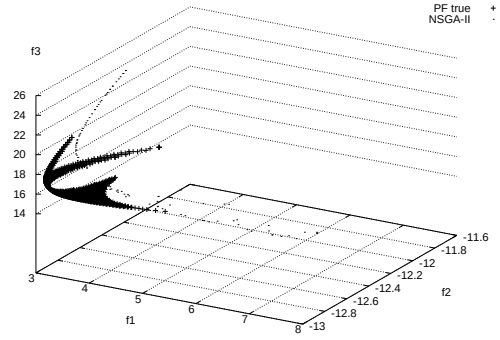

Figura 5.14: Frente de Pareto obtenido por el NSGA-II para el ejemplo 3

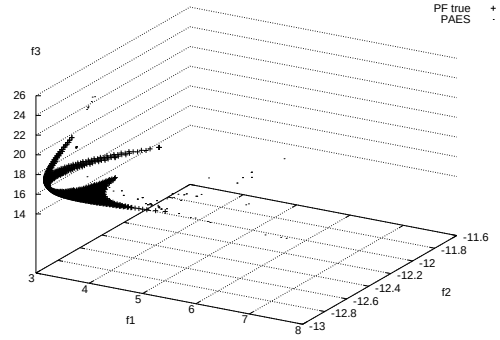

Figura 5.15: Frente de Pareto obtenido por PAES para el ejemplo 3

Para la métrica inversa de la distancia generacional el mejor valor es para SIA y el segundo para las otras tres técnicas. Sin embargo, debe notarse que la desviación estándar de PAES es mayor (casi el doble de las demás). En las gráficas observamos que el SIA y el micro-AG son las que cubren mejor el verdadero frente de Pareto, aunque muchos de sus puntos sólo están aproximados a éste. Por otro lado, el NSGA-II y PAES pierden una parte del verdadero frente de Pareto, sin embargo, los puntos que llegan parecen estar exactamente sobre éste.

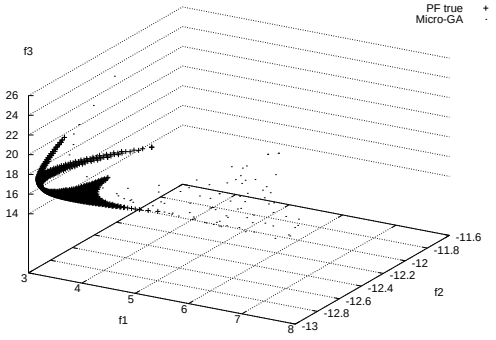

Figura 5.16: Frente de Pareto obtenido por el micro-AG para el ejemplo 3

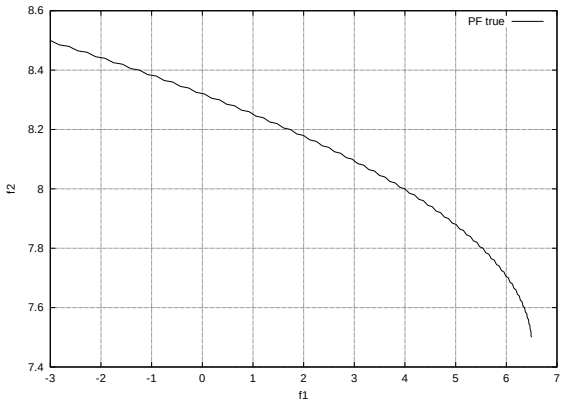

Figura 5.17: Verdadero frente de Pareto para el ejemplo 4

5.6. Conclusiones

En este capítulo hemos presentado un sistema inmune artificial para solucionar problemas de optimización con múltiples objetivos basado en el principio de selección clonal del sistema inmune.

El algoritmo fue probado con cinco funciones de ejemplo, dos ellas con restricciones. Comparamos los resultados obtenidos contra tres algoritmos que son representativos del estado del arte en optimización multiobjetivo que son NSGA-II, PAES y el micro-AG.

Para realizar las comparaciones aplicamos tres métricas que miden de

	Tasa de error			Dispersión		
	NSGA-II	PAES	MicroAG	NSGA-II	PAES	MicroAG
Promedio	0.85364	0.65850	0.80238	0.02201	0.12157	0.22306
Mejor	0.70000	0.48000	0.60000	0.00724	0.04166	0.08349
Peor	1.00000	0.80000	0.93182	0.03504	.22073	.9431
Desv. Est.	0.10118	0.08022	0.09208	0.00742	0.04998	0.23221
	Distancia generacional			**Distancia generacional inversa**		
	NSGA-II	PAES	MicroAG	NSGA-II	PAES	MicroAG
Promedio	0.04528	0.09560	0.11904	0.05553	0.01803	0.01508
Mejor	0.00422	0.00261	0.00538	0.01362	0.00929	0.01074
Peor	0.47043	0.22451	0.45925	0.14823	0.02294	0.02530
Desv. Est.	0.11135	0.10450	0.11895	0.03256	0.00447	0.00428

Cuadro 5.8: Valores de las métricas del ejemplo 4 para NSGA-II, PAES y MicroAG

manera cuantitativa la calidad de las soluciones arrojadas por los algoritmos, además de valernos de una comparación gráfica.

En general, podemos afirmar que el mejor desempeño lo obtienen los algoritmos NSGA-II y SIA. El siguiente lugar lo ocupa PAES y finalmente el micro-AG.

Nuestro algoritmo presenta una mejor distribución de las soluciones a lo largo del frente de Pareto cuando se utilizan entre 25 y 35 divisiones de la rejilla adaptativa que utilizamos en la memoria secundaria.

Métrica de tasa de error

# divisiones	5	15	25	35	45
Promedio	0.4250	0.4738	0.4420	0.4860	0.4477
Mejor	0.3000	0.3100	0.3000	0.3300	0.3400
Peor	0.5700	0.6768	0.5900	0.6500	0.5400
Desv. Est.	0.0826	0.0747	0.0830	0.0845	0.0702

Métrica de dispersión

# divisiones	5	15	25	35	45
Promedio	3.3650	3.106	3.16609	3.2056	3.2988
Mejor	2.9572	2.9281	2.89850	2.8970	2.9184
Peor	3.8963	3.8995	3.73149	3.8953	3.8992
Desv. Est.	0.2552	0.3409	0.24809	0.2679	0.2685

Métrica de distancia generacional

# divisiones	5	15	25	35	45
Promedio	0.0681	0.0570	0.0533	0.0462	0.0588
Mejor	0.0079	0.0050	0.0047	0.0046	0.0038
Peor	0.1292	0.1298	0.1071	0.1293	0.1293
Desv. Est.	0.0335	0.0446	0.0332	0.0366	0.0343

Métrica de distancia generacional inversa

# divisiones	5	15	25	35	45
Promedio	0.0301	0.0254	0.0234	0.0204	0.0254
Mejor	0.0068	0.0052	0.0046	0.0039	0.0043
Peor	0.0551	0.0554	0.0453	0.0550	0.0550
Desv. Est.	0.0136	0.0180	0.0133	0.0149	0.0141

Cuadro 5.9: SIA con diferente número de divisiones de la rejilla adaptativa para el ejemplo 5

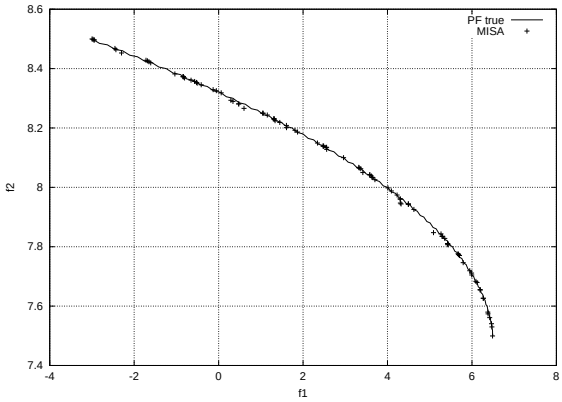

Figura 5.18: Frente de Pareto obtenido por el sistema inmune artificial (SIA) para el ejemplo 4

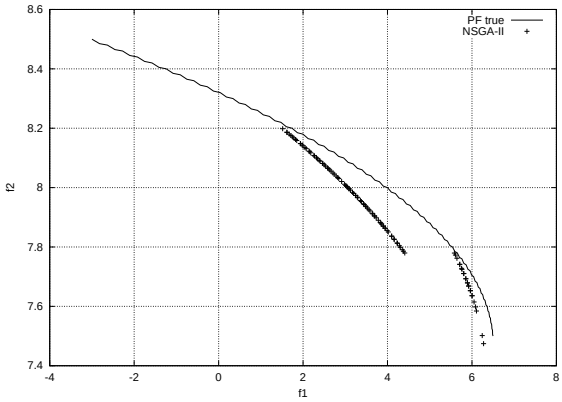

Figura 5.19: Frente de Pareto obtenido por el NSGA-II para el ejemplo 4

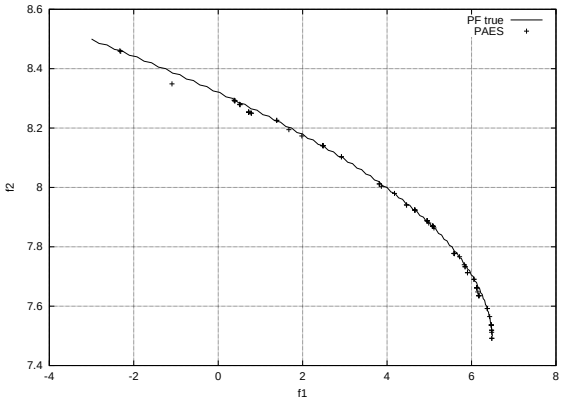

Figura 5.20: Frente de Pareto obtenido por PAES para el ejemplo 4

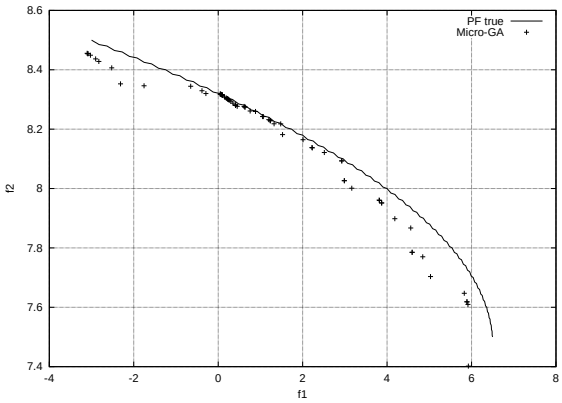

Figura 5.21: Frente de Pareto obtenido por el micro-AG para el ejemplo 4

	Tasa de error			Dispersión		
	NSGA-II	PAES	MicroAG	NSGA-II	PAES	MicroAG
Promedio	0.26250	0.37150	0.60564	3.33918995	3.1667038	3.31062
Mejor	0.18000	0.06000	0.52525	3.019572	2.763833	2.81993
Peor	0.39000	0.88000	0.71429	3.728558	3.711165	3.90077
Desv. Est.	0.05486	0.21475	0.06707	0.25094	0.248873	0.41243

	Distancia generacional			Distancia generacional inversa		
	NSGA-II	PAES	MicroAG	NSGA-II	PAES	MicroAG
Promedio	0.06557	0.04557	0.06662	0.03319	0.03418	0.03112
Mejor	0.01996	0.00285	0.03003	0.01269	0.00648	0.01650
Peor	0.11411	0.11076	0.12919	0.05001	0.0978	0.05530
Desv. Est.	0.03109	0.03048	0.04145	0.01206	0.02234	0.01587

Cuadro 5.10: Valores de las métricas del ejemplo 5 para NSGA-II, PAES y MicroAG

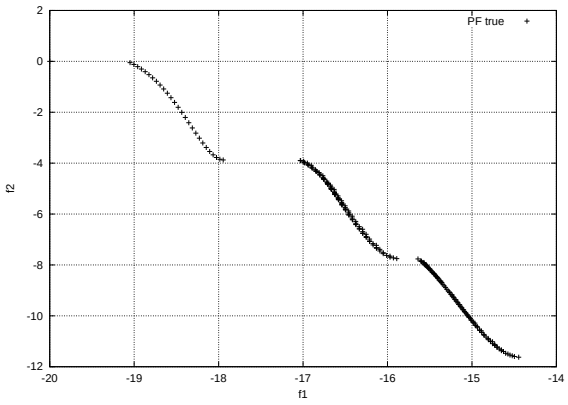

Figura 5.22: Verdadero frente de Pareto para el ejemplo 5

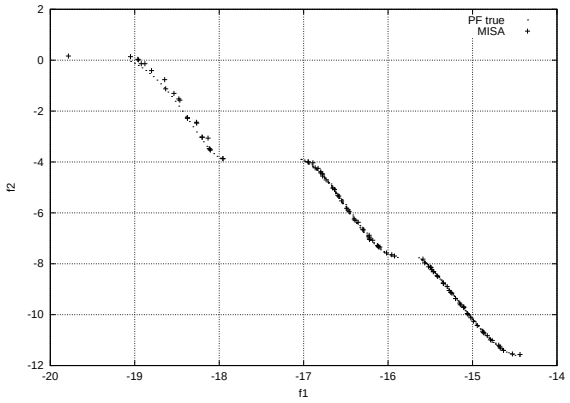

Figura 5.23: Frente de Pareto obtenido por el SIA para el ejemplo 5

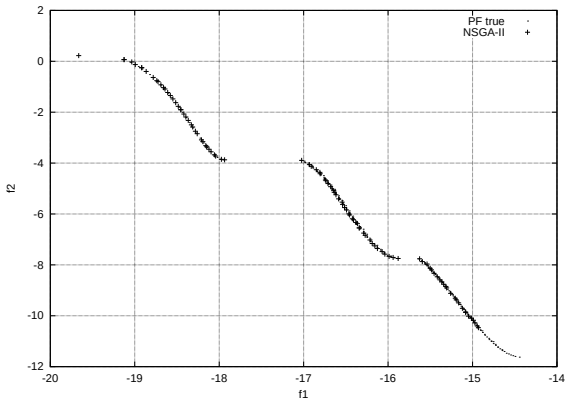

Figura 5.24: Frente de Pareto obtenido por el NSGA-II para el ejemplo 5

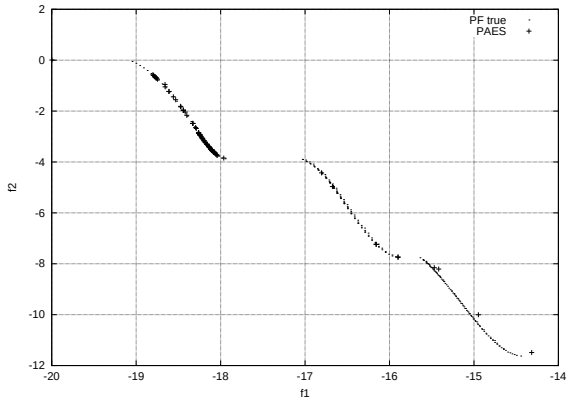

Figura 5.25: Frente de Pareto obtenido por PAES para el ejemplo 5

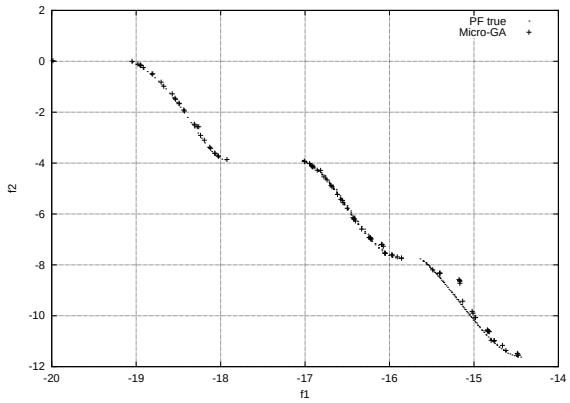

Figura 5.26: Frente de Pareto obtenido por el micro-AG para la función 5

Capítulo 6

Convergencia del sistema inmune artificial multiobjetivo

6.1. Introducción

Existe pocos estudios formales sobre la convergencia de algoritmos evolutivos y particularmente aquellos inspirados en sistemas inmunes artificiales. En este capítulo presentamos un estudio matemático que demuestra la convergencia de nuestro algoritmo del sistema inmune artificial hacia el verdadero frente de Pareto en problemas de optimización multiobjetivo.

Para realizar esta demostración consideramos que la tarea de encontrar los elementos minimales de un conjunto parcialmente ordenado es equivalente a encontrar el verdadero frente de Pareto de un problema de optimización multiobjetivo. Dicha consideración es la misma adoptada previamente en otros trabajos similares (ver por ejemplo [110]).

Iniciaremos el capítulo dando una breve descripción del trabajo relacionado. Posteriormente, proporcionaremos algunas definiciones preliminares necesarias para luego proceder a realizar la demostración. Finalmente, se proporcionan las conclusiones de este capítulo.

6.2. Trabajo relacionado

El primer trabajo que relaciona algún aspecto matemático del sistema inmune con un modelo computacional es el desarrollado por Perelson y Oster [102] quienes realizaron un estudio cualitativo en el que introdujeron el concepto de forma-espacio para describir la interacción entre moléculas del sistema inmune con los antígenos. Su principal enfoque es determinar qué tan grande debe ser el repertorio de células inmunes para reaccionar eficientemente contra los antígenos en donde el sistema inmune es tratado como un sistema de reconocimiento de patrones que identifica *formas*.

Tarakanov y Dasgupta [132] propusieron un modelo formal de un sistema inmune artificial en el que realizan una abstracción matemática de las proteínas expresadas en la superficie de las células inmunológicas (T y B principalmente) que son las responsables de las interacciones entre células propias y las provenientes del exterior (antígenos) que determinan la respuesta inmune y el proceso de reconocimiento. Introducen el concepto de proteína formal como una abstracción matemática de los mecanismos biofísicos en que se basa el comportamiento de las proteínas. Su modelo se basa en las dinámicas definidas en la teoría de red inmune [101], y argumentan que éste provee la base matemática para el desarrollo de algoritmos computacionales para solucionar problemas.

Saab [115] y Saab et al. [114] desarrollaron un modelo que emula un sistema inmune mediante álgebra de procesos. En este trabajo se identifican los principales componentes del sistema inmune y se define un álgebra específica mediante la que se modela el comportamiento individual de los componentes y sus interrelaciones.

Villalobos-Arias et al. [138] proporcionan un estudio matemático de convergencia basado en cadenas de Markov de un sistema inmune artificial multiobjetivo cuyo operador es la mutación uniforme.

Con respecto a análisis de convergencia de algoritmos evolutivos en general, encontramos el trabajo de Rudolph [110, 111, 112] que demuestra la convergencia de algoritmos evolutivos multiobjetivo bajo ciertas condiciones.

Reyes-Sierra [125] proporciona una demostración para un algoritmo

genético multiobjetivo, y Lara-López [86] hace lo mismo para una estrategia evolutiva multiobjetivo.

Van Veldhuizen y Lamont también hicieron una demostración para algoritmos evolutivos multiobjetivo [136]

Nuestra demostración se basa en el análisis de convergencia efectuado en [138].

6.3. Definiciones

Desde luego que sabemos que en un procesador digital es ineludible el proceso de discretización de los puntos reales del espacio de búsqueda. Ello involucra una pérdida implícita en la precisión infinita de los valores en R^n, la mejor o peor aproximación a los valores exactos del espacio de búsqueda dependerá de la cantidad de bits asignados para representar dichos valores y del problema en particular que se está examinando. En este trabajo se decidió utilizar representación de números reales por punto fijo asignando una cantidad específica de bits. La demostración asume que los puntos en R^n del espacio de búsqueda han sido representados en una forma adecuada de bits que permite, para fines prácticos, suponer que dicha aproximación es indistinguible de los valores reales.

Para definir el concepto de conjunto parcialmente ordenado, es necesario antes conocer las nociones de *relación* y sus propiedades.
(Asumiremos minimización en el resto del capítulo).

Definición **1** *Sea A un conjunto de vectores no vacío. El subconjunto $\mathfrak{R} \subseteq A \times A$ es llamado una relación binaria en A. Sea $x, y \in A$. Si $(x, y) \in \mathfrak{R}$ (también denotado como $x\mathfrak{R}y$), entonces decimos que x está en relación $\mathfrak{R}$ con y. Decimos que una relación $\mathfrak{R}$ en A es:*

(a) reflexiva si $x\mathfrak{R}x$ es cierto para toda $x \in A$

(b) antireflexiva si $x\mathfrak{R}y \Rightarrow x \neq y$ es cierto para toda $x, y \in A$

(c) simétrica si $x\mathfrak{R}y \Rightarrow y\mathfrak{R}x$ es cierto para toda $x, y \in A$

(d) antisimétrica si $x\Re y \wedge y\Re x \Rightarrow x = y$ *es cierto para toda* $x, y \in A$

(e) asimétrica si $x\Re y \Rightarrow \neg(y\Re x)$ *es cierto para toda* $x, y \in A$

(f) transitiva si $x\Re y \wedge y\Re z \Rightarrow x\Re z$ *es cierto para toda* $x, y, z \in A$

Definición **2** *Una* **relación de orden parcial** *es reflexiva, antisimétrica y transitiva. Se representa mediante el símbolo* $\preceq$. *Sea* $x = (x_1, x_2, \ldots, x_d)$ *y* $y = (y_1, y_2, \ldots, y_d)$ *vectores en* R^d *entonces definimos una relación de orden parcial de la siguiente manera:*
$x \preceq y$ *si y sólo si* $x_i \leq y_i \ \forall i \in \{1, \ldots, d\}$

Definición **3** *Una relación antireflexiva, antisimétrica y transitiva es llamada de* **orden parcial estricto** *y se representa mediante el símbolo* $\prec$. *Se define de la siguiente manera:*
$x \prec y \Longleftrightarrow (x \preceq y) \wedge (x \neq y)$

Por ejemplo, si: $x = (8, 5, 7)$ y $y = (8, 6, 7)$ entonces decimos que $x \prec y$.

Definición **4** *Sea* X *un conjunto y* $\preceq$ *una relación de orden parcial sobre éste, llamamos a la pareja* $(X, \preceq)$ *un* **conjunto parcialmente ordenado** *(o* poset *por sus siglas en inglés).*

Definición **5** *Un elemento* $\vec{x^*} \in S$ *es llamado un* **elemento minimal** *del poset* $(S, \preceq)$ *si no existe un elemento* $\vec{x} \in S$ *tal que* $\vec{x} \prec \vec{x^*}$. *Denotamos con* $M(S, \preceq)$ *al conjunto de todos los elementos minimales* $(S, \preceq)$.

Definición **6** *Un punto* $\vec{x^*} \in S$ *es llamado una* **solución óptima de Pareto** *al problema de optimización multiobjetivo si* $\vec{x^*} \in M(S, \preceq)$.

El conjunto

$$P^* = \{x \in X : x \text{ es una solución óptima de Pareto}\}$$

es llamado el conjunto óptimo de Pareto, y su imagen en F

$$F(P^*) := \{F(x) : x \in P^*\}$$

es el frente de Pareto.

6.4. Convergencia del SIA

A continuación presentamos el algoritmo del SIA multiobjetivo de una manera resumida, sólo mostrando los pasos que se consideran importantes para la realización de este estudio. (Los detalles del algoritmo fueron explicados en la sección 5.3).
A representa la población de tamaño r de anticuerpos.
E son el conjunto de anticuerpos estimulados para clonarse, estos son no dominados en la iteración actual.
$Clones$ son los clones generados a partir de E
N los anticuerpos no estimulados (los que no se clonan), estos son dominados.
D son los elementos de la memoria secundaria, son no dominados globales, es decir a lo largo del algoritmo (elitismo).

Algoritmo de SIA multiobjetivo resumido.
INICIO

I) Se inicia A^0 aleatoriamente

II) $t \leftarrow 0$

III) $D^0 \leftarrow \emptyset$

IV) **Repetir:**

- *a*) $E^t \leftarrow M(A^t, \preceq)$
- *b*) $N^t \leftarrow (A^t \setminus E^t)$
- *c*) $D^{t+1} \leftarrow M(D^t \cup E^t, \preceq)$
- *d*) $Clones^t \leftarrow$ clonación(E^t)
- *e*) $Clones^t \leftarrow$ hipermutación$(Clones^t)$
- *f*) $N^t \leftarrow$ mutación(N^t)
- *g*) $A^{t+1} \leftarrow \{M((E^t \cup N^t \cup Clones), \preceq) \cup X\}$ donde X es un conjunto seleccionado aleatoriamente de tamaño necesario para completar r anticuerpos.

h) $t \leftarrow t + 1$

i) Ir al paso VI.a un número predeterminado de veces.

FIN

La hipermutación aplicada en el paso IV.*e* y la mutación del paso IV.f fueron definidas en la sección 5.3 de este documento.

Lo que deseamos demostrar es que el algoritmo es capaz de converger al conjunto de elementos minimales con probabilidad igual a 1, lo cual es equivalente a encontrar los elementos del verdadero frente de Pareto.

La población de r anticuerpos *A* es dividida en dos grupos (pasos IV a y b). El primero de ellos son los anticuerpos estimulados E para clonarse y formar el grupo de clones C. El segundo grupo está formado por el resto de los anticuerpos, es decir aquellos que no fueron estimulados N.

Se aplica hipermutación sobre el primer grupo: $hipermutación(C)$, y mutación sobre el segundo grupo: $mutación(N)$. Analizaremos los dos grupos de manera separada.

Hipermutación

Llamaremos w a la cantidad de bits que se mutan en cada clon. Entonces, tenemos que la probabilidad de que la hipermutación aplicada a un clon $\hat{i}$ genere al individuo i^* se define de la siguiente manera:

$$P(hipermutación(\hat{i}), i^*) = \begin{cases} \frac{1}{l^w} & \text{si } H(\hat{i}, i^*) = w \\ 0 & \text{en otro caso.} \end{cases}$$

donde $H(\hat{i}, i^*)$ es la distancia de Hamming entre el clon $\hat{i}$ y la cadena binaria i^* que deseamos obtener, l es la longitud de las cadenas binarias. Entonces tenemos que, en general $P(hipermutación(\hat{i}), i^*) \geq 0$ de manera que la hipermutación no asegura que el algoritmo pueda alcanzar cualquier punto.

Mutación

Ahora analizaremos la mutación uniforme aplicada a los anticuerpos no es-

timulados (N) del paso IV.f. La mutación opera de manera independiente en cada individuo, y consiste en la variación probabilísticas de cada bit de la cadena que representa a los individuos. Tenemos que la probabilidad de mutación pm aplicada es: $1 > pm > 0$. Por ejemplo la probabilidad de que un anticuerpo: $\hat{i} = 11111$ se convierta en i^*=10101 por mutación es $(1-pm)\cdot pm\cdot(1-pm)\cdot pm\cdot(1-pm) = pm^g\cdot(1-pm)^(l-g)$ para $g = 2$ y $l = 5$. El valor de g es la distancia de Hamming entre las cadenas $\hat{i}$ y i^*. Entonces la probabilidad de que un anticuerpo cualquiera $\hat{i}$ llegue a la cadena i^* por mutación es:

$$P_{mutación}(\hat{i}, i^*) \;=\; pm^{H(\hat{i},i^*)}(1-pm)^{l-H(\hat{i},i^*)}$$

donde $H(\hat{i}, i^*)$ es la distancia de Hamming entre las cadenas $H(\hat{i}, i^*)$.
De aquí concluimos que $P_{mutación}(\hat{i}, i^*) > 0$ por lo que la mutación asegura que el algoritmo pueda alcanzar cualquier punto, en particular los elementos minimales.

Convergencia del SIA

Este algoritmo puede ser modelado como una cadena de Markov $\{X_k : k \leq 0\}$ cuyo espacio de búsqueda S es el conjunto de todas las poblaciones posibles de n individuos, cada uno representado por una cadena binaria de longitud l. Por lo tanto $S = (B^l)^n = B^{nl}$, donde $B = \{0, 1\}$; de manera que S es el conjunto de todos los vectores posibles de n elementos, cada uno de los cuales es una cadena binaria de longitud l cuyos elementos son 0 o 1.

Definición **7** *Sea* $\{X_k : k \leq 0\}$ *una cadena de Markov asociada al algoritmo. Diremos que el algoritmo converge con probabilidad 1 si,*

$$(\{X_k\} \subset P^*) \to 1 \textit{ para } k \to \infty$$

Sean i y j dos estados. La probabilidad de transición de la cadena está dada por:

$$P_{ij} = (X_{k+1} = j | X_k = i)$$

Definición **8** *Sean* $i, j \in S$. *Decimos que* i *es* **no esencial** *si existe un estado* j *tal que* $i \to j$ *pero* $i \not\to j$.
Si la condición no se cumple entonces i *es llamado un estado* **esencial**. *Por lo tanto si* i *es esencial,* $i \to j$ *implica* $i \leftrightarrow j$ *y existe al menos una* j *tal que* $i \to j$

Denotaremos al conjunto de estados esenciales con ε y los no esenciales con I. Entonces tenemos que $S = \varepsilon \cup I$.

Definición **9** *Una matriz* $M : n \times m$ *se dice que es no negativa si* $p_{ij} \geq 0$ *y positiva si* $p_{ij} > 0$ *para todos* $i = 1, 2, ..., n$ *y* $j = 1, 2, ..., m$.
Una matriz no negativa se dice ser **estocástica** *si la suma de cada renglón es exactamente 1.*

Lema **1** *Sea* P *una matriz estocástica, y* Q *una submatriz de* P *asociada con las transiciones entre estados no esenciales. Entonces, como* $k \to \infty$*:* $Q^k \to 0$

Demostración ver [124] pag. 120

Corolario **1** *Para cualquier distribución inicial,*
$(X_k \in I) \to 0$ *para* $k \to \infty$

Demostración:
Para cualquier vector con distribución inicial p_o, sea $p_o(I)$ el subvector que corresponde a los estados no esenciales. Entonces por el lema 1 tenemos,
$(x_k \in I) = p_o(I)'Q^k 1 \to 0$ para $k \to \infty$

Lema **2** *Si un estado en el algoritmo del sistema inmune contiene en su conjunto élite un elemento que no es óptimo de Pareto este estado es no esencial.*

Demostración:
De acuerdo a la notación utilizada en el algoritmo del SIA resumido mostrado al inicio de esta sección, D es el conjunto élite, E son los anticuerpos estimulados (no dominados en la iteración actual) y N es el conjunto de anticuerpos no estimulados (dominados).
Sea $\hat{i} = (D^i, E^i, N^i)$ un estado en el cual el conjunto élite D contiene elementos que no son óptimos de Pareto.

1. De E^i se generan clones y luego estos se mutan un número fijo de la entradas al azar (hipermutación). Entonces mútese las primeras posiciones en todos los elementos de todos los clones, existe un probabilidad positiva de hacer esto, llame al conjunto obtenido $Clones(E^i)$

2. Como en N^i se aplica mutación uniforme, mute todos los elementos de éste para obtener el peor elemento de $Clones(E^i)$, de nuevo existe un probabilidad positiva de hacer esto, para que ninguno de estos pase al conjunto E^i.

3. Ahora se debe asignar los nuevos individuos estimulados a los que llamaremos E^j como los no dominados y N^j una cantidad de los restantes para completar los r individuos de la población.

4. Al aplicar elitismo se obtendrá el conjunto D^j

5. A los clones de E^j mútesele las mismas primeras entradas de los elementos. Entonces $Clones(E^j) \subseteq Clones(E^j)$ con lo que los mejores elementos de $Clones(E^j)$ serán de nuevo E^j y al aplicar elitismo los elementos de E^j no modificará al conjunto D^j

6. Sean $D^j_{s_1}, \ldots, D^j_{s_k}$ los elementos de D^j que no son óptimos de Pareto. Como X es completo existen elementos $i^\star_{s_1}, \ldots, j^\star_{s_k} \in P^*$, que dominan a $D^j_{s_1}, \ldots, D^j_{s_k}$, respectivamente.

7. Ahora como en N^j se aplica mutación uniforme a partir de $N^j_1, \ldots, N^j_k$ se pueden obtener $i^\star_{s_1}, \ldots, j^\star_{s_k}$ respectivamente, y los demás elementos de D^j se dejan igual.

8. Como $ClonesM(E^j)$ y $\{j_{k+1}, \ldots, j_{n_2}\}$ ya habían modificado a D^j al aplicar elitismo en esta parte ya no modificarán a D^j. Así los únicos que modifican a j^e serán los $i^\star_{s_1}, \ldots, j^\star_{s_k}$ que reemplazarán a los elementos no dominados de éste.

9. Sea $i^\dagger$ el estado resultante de este proceso, con lo anterior se puede ir de i a $i^\dagger$ $(i \longrightarrow i^\dagger)$, pero como en $i^{\dagger e}$ sólo hay óptimos de Pareto, $P_{i^\dagger i} = 0$ $(i.e \quad i^\dagger \not\longrightarrow i)$, Lo que demuestra que i en un estado no esencial.

Teorema **1** *El algoritmo del sistema inmune artificial con elitismo converge.*

Demostración:
Del lema 2 y corolario 1 tenemos que

$$(\{D_k\} \subset P^*) = (D_k \in \varepsilon) = 1 - (D_k \in I) \to 1 - 0 = 1$$

con lo que queda demostrado.

6.5. Conclusiones

Algunos detalles del algoritmo que no se consideran importantes para este estudio fueron eliminados, como por ejemplo el hecho de que el tamaño de la memoria secundaria es finito, así como la acción de la rejilla adaptativa sobre ella. Además tampoco se consideró la selección de los individuos minimales en el paso IX, en el que podría suceder que éstos constituyan una cantidad mayor a r.

Estos dos procesos del algoritmo son resueltos de manera que los individuos que se seleccionan son aquellos que mejoran la distribución de los puntos a lo largo del frente de Pareto.

Una limitante del modelo es que no se considera el manejo de restricciones, ya que se presupone que el algoritmo es capaz de encontrar soluciones factibles desde el inicio.

El sistema inmune artificial es una heurística por lo que, debido a su naturaleza estocástica resulta importante realizar este tipo de demostraciones de convergencia. Esto nos ayuda a comprender en cierta medida la dinámica del algoritmo, así como a intentar nuevos caminos con la finalidad de realizar mejoras.

Como conclusión del estudio realizado podemos asegurar que el algoritmo converge completamente al conjunto de óptimos de Pareto.

Capítulo 7

Conclusiones y trabajo futuro

En esta tesis se exploraron dos modelos del sistema inmune artificial para solucionar problemas de optimización global.

Realizamos una mejora significativa al algoritmo del sistema inmune artificial que es un híbrido con un algoritmo genético para manejo de restricciones en optimización global. Una ventaja importante es que este algoritmo no necesita definir ningún factor de penalización y es conceptualmente sencillo. Concluimos que los resultados obtenidos por nuestro algoritmo, tanto en su versión secuencial como la paralela, están a la altura del estado del arte en el área.

También realizamos la primera propuesta de un sistema inmune artificial para solucionar problemas de optimización con múltiples objetivos. Este algoritmo es capaz de manejar restricciones de todo tipo y está basado en un mecanismo observado en el sistema inmune al que se le conoce con el nombre de selección clonal. Después de realizar una comparación contra técnicas representativas del estado del arte del área, utilizando métricas que cuantifican diferentes aspectos de los frentes encontrados por los algoritmos, podemos concluir que nuestra propuesta arroja resultados altamente competitivos. Además, adaptamos este algoritmo a un modelo matemático que nos permite confirmar que, teóricamente éste converge al frente de Pareto.

De manera general, el sistema inmune artificial es una opción competitiva para solucionar problemas difíciles de ingeniería, específicamente en optimización numérica como se evidenció en los resultados obtenidos en

esta tesis.

Como posibles rutas para trabajo futuro de esta tesis podemos señalar los siguientes puntos:

- Experimentar con diferentes modelos de paralelismo aplicados al algoritmo del sistema inmune artificial para manejo de restricciones en algoritmos genéticos.
- Aplicar técnicas de paralelismo al sistema inmune artificial multiobjetivo.
- Mejorar las propiedades de distribución del algoritmo multiobjetivo a lo largo del frente de Pareto.
- Realizar un sistema inmune artificial para optimización dinámica.
- Eliminar los parámetros en los algoritmos, haciendo uso de mecanismos de auto-adaptación y/o adaptación en línea.
- Experimentar con diferentes modelos del sistema inmune artificial para solucionar problemas de optimización global.

Apéndice A

Terminología utilizada en computación evolutiva:

- **ADN**: Ácido desoxirribonucléico; es el material genético fundamental de todos los organismos vivos.
- **Aptitud**: Valor que se asigna a cada individuo y que indica qué tan bueno es éste con respecto a las demás soluciones de un problema.
- **Cromosoma**: Estructura de datos que contiene una cadena de parámetros de diseño o genes. Por ejemplo, puede ser una cadena de bits.
- **Cruza**: Es un operador que forma un nuevo cromosoma combinando parte de cada uno de los cromosomas padres.
- **Elitismo**: Mecanismo utilizado en algunos algoritmos evolutivos para asegurar que los cromosomas de los miembros más aptos de una población pasen a la siguiente generación sin ser alterados.
- **Época**: Define la cantidad de generaciones entre una migración y otra en los algoritmos genéticos paralelos.
- **Gene**: Es una subsección de un cromosoma. Usualmente codifica el valor de un solo parámetro (variable de decisión).
- **Generación**: Es una iteración que involucra la creación de una nueva población por medio de los operadores básicos de reproducción.

- **Individuo**: Un solo miembro de la población de soluciones potenciales a un problema. Cada individuo contiene un cromosoma que representa una solución posible al problema.
- **Migración**: Transferencia de un individuo de una subpoblación a otra.
- **Mutación**: Operador que forma un nuevo cromosoma a través de alteraciones a los genes del cromosoma.
- **Genotipo**: Se denomina genotipo a la codificación (por ejemplo binaria) de los parámetros que representan una solución del problema a resolverse.
- **Fenotipo**: Se denomina fenotipo a la decodificación del cromosoma. Es decir, a los valores obtenidos al pasar de la representación (binaria) a la usada por la función objetivo.

Terminología utilizada en sistema inmune artificial:

- **Afinidad**: Medida de similitud entre un antígeno y un anticuerpo o entre anticuerpos.
- **Anticuerpo**: Moléculas segregadas durante la respuesta inmune aprendida, encargadas de neutralizar y eliminar a los antígenos.
- **Antígeno**: Cualquier molécula externa capaz de iniciar una respuesta inmune.
- **Clon**: Una copia exacta de un linfocito, que puede ser sometida a mutación.
- **Linfocito**: Células representantes de la respuesta inmune aprendida. Los linfocitos B tienen la capacidad de segregar anticuerpos, que son los principales actores de la respuesta inmune.
- **Memoria**: El sistema inmune es capaz de recordar antígenos que se han presentado en el pasado.

- **Aprendizaje**: La respuesta del sistema inmune, ante repetidas apariciones de un antígeno es cada vez más rápida y eficiente, por eso se dice que posee aprendizaje.
- **Respuesta inmune innata**: Respuesta del sistema inmune que es inespecífica y no posee memoria.
- **Respuesta inmune adaptativa**: Respuesta del sistema inmune que posee memoria y aprendizaje, realizada principalmente por los anticuerpos.

Apéndice B

Función de prueba 1

Minimizar:

$$f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \tag{7.1}$$

sujeto a:

$$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \tag{7.2}$$

$$g_2(x) = 2x_1 + 2x_3 + x_{11} - x_{12} - 10 \leq 0 \tag{7.3}$$

$$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \tag{7.4}$$

$$g_4(x) = -8x_1 + x_{10} \leq 0 \tag{7.5}$$

$$g_5(x) = -8x_2 + x_{11} \leq 0 \tag{7.6}$$

$$g_6(x) = -8x_3 + x_{12} \leq 0 \tag{7.7}$$

$$g_7(x) = 2x_4 - x_5 + x_{10} \leq 0 \tag{7.8}$$

$$g_8(x) = 2x_6 - x_7 + x_{11} \leq 0 \tag{7.9}$$

$$g_9(x) = 2x_8 - x_9 + x_{12} \leq 0 \tag{7.10}$$

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 $(g1, g2, g3, g4, g5, g6)$. $f^* = -15{,}0$

Función de prueba 2

Maximizar:

$$f(x) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} ix_i^2}} \right| \tag{7.11}$$

sujeto a:

$$g_1(x) = 0{,}75 - \prod_{i=1}^{n} x_i \leq 0 \tag{7.12}$$

$$g_2(x) = \sum_{i=1}^{n} x_i - 7{,}5n \leq 0 \tag{7.13}$$

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 $f(x) = 0{,}803619$.

Función de prueba 3

Maximizar:

$$f(x) = \left(\sqrt{n}\right)^n \prod_{i=1}^{n} x_i \tag{7.14}$$

sujeto a:

$$h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0 \tag{7.15}$$

donde $n = 10$ y $0 \leq x_i \leq 1$ para $1 \leq i \leq n$. El máximo global está localizado en $x_i^* = 1/\sqrt{n}$, $1 \leq i \leq n$ donde $f(x^*) = 1$

Función de prueba 4

Minimizar:

$$f(x) = 5{,}3578547x_3^2 + 0{,}8356891x_1x_5 + 37{,}293239x_1 - 40792{,}141 \tag{7.16}$$

sujeto a:

$$g_1(x) = 85{,}334407 + 0{,}0056858x_2x_5 + 0{,}0006262x_1x_4 - 0{,}0022053x_3x_5 - 92 \leq 0 \tag{7.17}$$

$$g_2(x) = -85{,}334407 - 0{,}0056858x_2x_5 - 0{,}0006262x_1x_4 + 0{,}0022053 \leq 0 \tag{7.18}$$

$$g_3(x) = 80{,}51249 - 0{,}0071317x_2x_5 + 0{,}0029955x_1x_2 + 0{,}0021813x_3^2 - 110 \leq 0 \tag{7.19}$$

$$g_4(x) = -80{,}51249 - 0{,}0071317x_2x_5 + 0{,}0029955x_1x_2 - 0{,}0021813x_3^2 + 90 \leq 0 \tag{7.20}$$

$$g_5(x) = 9{,}300961 + 0{,}0047026x_3x_5 + 0{,}0012547x_1x_3 + 0{,}0019085x_3x_4 - 25 \leq 0 \tag{7.21}$$

$$g_6(x) = -9{,}300961 - 0{,}0047026x_3x_5 - 0{,}0012547x_1x_3 - 0{,}0019085x_3x_4 + 20 \leq 0 \tag{7.22}$$

donde $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$ y $27 \leq x_i \leq 45$ para $3 \leq i \leq 5$. La solución óptima es $f(x^*) = -30665{,}539$

Función de prueba 5

Maximizar:

$$f(x) = 3x_1 + 0{,}000001x_1^3 + 2x_2 + (0{,}000002/3)x_2^3 \tag{7.23}$$

sujeto a:

$$g_1(x) = -x_4 + x_3 - 0{,}55 \leq 0 \tag{7.24}$$

$$g_2(x) = -x_3 + x_3 - 0{,}55 \leq 0 \tag{7.25}$$

$$h_3(x) = (1000\sin(-x_3 - 0{,}25) + 1000\sin(-x_4 - 0{,}25) + 894{,}8 - x_1 = 0 \tag{7.26}$$

$$h_4(x) = 1000\sin(x_3 - 0{,}25) + 1000\sin(x_3 - x_4 - 0{,}25) + 894{,}8 - x_2 = 0 \tag{7.27}$$

$$h_5(x) = 1000\sin(x_4 - 0{,}25) + 1000\sin(x_4 - x_3 - 0{,}25) + 1294{,}8 = 0 \tag{7.28}$$

donde $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0{,}55 \leq x_3 \leq 0{,}55$, y $-0{,}55 \leq x_4 \leq 0{,}55$. La mejor solución conocida es $x^\star = (679{,}9453, 1026{,}067, 0{,}1188764)$ donde $f(x^\star) = 5126{,}4981$

Función de prueba 6

Minimizar:

$$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \tag{7.29}$$

sujeto a:

$$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \tag{7.30}$$

$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82{,}81 \leq 0 \tag{7.31}$$

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.

Función de prueba 7
Minimizar:

$$f(\vec{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2)) + (5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2) + (x_10 - 7)^2 + 45$$

sujeto a:

$$g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7) + 9x_8 \leq 0 \tag{7.32}$$

$$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0 \tag{7.33}$$

$$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_10 - 12 \leq 0 \tag{7.34}$$

$$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \tag{7.35}$$

$$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \tag{7.36}$$

$$g_6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0 \tag{7.37}$$

$$g_7(x) = 0{,}5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_7 - 30 \leq 0 \tag{7.38}$$

$$g_8(x) = -3x_2 + 6x_2 + 12(x_9 - 8)^2 - 7x_10 \leq 0 \tag{7.39}$$

donde $-10 \leq x_i \leq 10, (i = 1, \ldots, 10)$. La solución óptima es $x^\star =$ (2,171996,
2,363683, 8,773926, 5,095984, 0,9906548, 1,430574, 1,321644, 9,828726,
5,095984, 8,375927), donde $f(x^\star) = 24{,}3062091$. Seis restricciones están activas en el óptimo.

Función de prueba 8

Maximizar:

$$f(x) = \frac{\sin(2\pi x_1)^3 \sin(2\pi x_2)}{x_1^3(x_1 + x_2)} \tag{7.40}$$

sujeto a:

$$g_1(x) = x_1^2 - x_2 + 1 \leq 0 \tag{7.41}$$

$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0 \tag{7.42}$$

donde $0 \leq x_1 \leq 10$ y $0 \leq x_2 \leq 10$. El óptimo está localizado en $x^\star = (1{,}2279713, 4{,}2453733)$ donde $f(x^\star) = 0{,}095825$.

Función de prueba 9

Minimizar:

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

sujeto a:

$$\begin{aligned} g_1(\vec{x}) &= -127 - 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0 \\ g_2(\vec{x}) &= -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0 \\ g_3(\vec{x}) &= -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0 \\ g_4(\vec{x}) &= 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0 \end{aligned}$$

donde $-10 \leq x_i \leq 10$ para $(i = 1, \ldots, 7)$. El óptimo global es $\vec{x}^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$ donde $f(\vec{x}^*) = 680.6300573$. Dos restricciones están activas en el óptimo (g_1 y g_4).

Función de prueba 10

Minimizar:

$$f(x) = x_1 + x_2 + x_3 \tag{7.43}$$

sujeto a:

$$g_1(x) = -1 + 0{,}0025(x_4 + x_6) \leq 0 \tag{7.44}$$

$$g_2(x) = -1 + 0{,}0025(x_5 + x_7 - x_4) \leq 0 \tag{7.45}$$

$$g_3(x) = -1 + 0{,}01(x_8 - x_5) \leq 0 \tag{7.46}$$

$$g_4(x) = -x_1x_6) + 833{,}33252x_4 + 100x_1 - 83333{,}333 \leq 0 \tag{7.47}$$

$$g_5(x) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0 \tag{7.48}$$

$$g_6(x) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0 \tag{7.49}$$

donde $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000 (i = 2, 3)$, y $10 \leq x_i \leq 1000 (i = 4, \ldots, 8)$ El óptimo es $x^\star = (579{,}3167, 1359{,}943, 5110{,}071)$ donde $f(x^\star) = 7049{,}3307$.

Función de prueba 11

Minimizar:

$$f(\vec{x}) = x_1^2 + (x_2 - 1)^2 \tag{7.50}$$

sujeto a:

$$h_1(\vec{x}) = x_2 - x_1^2 = 0 \tag{7.51}$$

donde $-1 \leq x_1 \leq 1$ y $-1 \leq x_2 \leq 1$. El óptimo global es $\vec{x}^* = (\pm 1/\sqrt{2}, \frac{1}{2})$ donde $f(\vec{x}^*) = 0{,}75$.

Función de prueba 12

Maximizar:

$$f(x) = (100 - (x_1 - 5)^2 - (x_2 - 5)^2)/100 \tag{7.52}$$

sujeto a :

$$g_1(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0{,}0625 \leq 0 \tag{7.53}$$

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 $f(x^*) = 1$.

Función de prueba 13

Minimizar:

$$f(x) = e^{(x_1 x_2 x_3 x_4 x_5)} \tag{7.54}$$

sujeto a :

$$h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \tag{7.55}$$

$$h_1(x) = x_2 x_3 - 5x_4 x_5 = 0 \tag{7.56}$$

$$h_1(x) = x_1^3 + x_2^3 + 1 = 0 \tag{7.57}$$

donde $-2{,}3 \leq x_i \leq 2{,}3 (i = 1,2)$ y $-3{,}2 \leq x_i \leq 3{,}2 (i = 3,4,5)$. La solución óptima es $x^* = (-1{,}717143, 1{,}595709, 1{,}827247, -0{,}7636413, -0{,}763645)$ donde $f(x^*) = 0{,}0539498$.

Apéndice C

Ejemplo 1

El primer ejemplo es una función con 2 objetivos cuyo frente de Pareto es desconectado formando 4 segmentos.
Minimizar: $F = (f_1(x, y), f_2(x, y))$, donde
$f_1(x, y) = x$,
$f_2(x, y) = (1 + 10y) * [1 - (\frac{x}{1+10y})^{\alpha} - \frac{x}{1+10y} \sin(2\pi q x)]$
y $0 \leq x, y \leq 1$, $q = 4$, $\alpha = 2$.

Ejemplo 2

El segundo problema es una función de 2 objetivos, donde el frente de Pareto está fragmentado en 2 segmentos. Esta función fue propuesta por Schaffer [117]:

$$\text{Minimizar} f_1(x) = \begin{cases} -x & \text{if } x \leq 1 \\ -2 + x & \text{if } 1 < x \leq 3 \\ 4 - x & \text{if } 3 < x \leq 4 \\ -4 + x & \text{if } x > 4 \end{cases} \tag{7.58}$$

$$\text{Minimizar} f_2(x) = (x - 5)^2 \tag{7.59}$$

y $-5 \leq x \leq 10$.

Ejemplo 3

Fue propuesto por Viennet [137]:
Minimizar: $F = (f_1(x, y), f_2(x, y), f_3(x, y))$

donde:

$$\begin{aligned} f_1(x,y) &= \frac{(x-2)^2}{2} + \frac{(y+1)^2}{13} + 3, \\ f_2(x,y) &= \frac{(x+y-3)^2}{175} + \frac{(2y-x)^2}{17} - 13, \\ f_3(x,y) &= \frac{(3x-2y+4)^2}{8} + \frac{(x-y+1)^2}{27} \\ &\quad +15 \end{aligned}$$

y: $-4 \leq x, y \leq 4$, $y < -4x + 4$, $x > -1$, $y > x - 2$.

Ejemplo 4
El cuarto ejemplo es una función de prueba propuesta por Kita [76]:
Maximizar $F = (f_1(x,y), f_2(x,y))$
donde:

$$\begin{aligned} f_1(x,y) &= -x^2 + y, \\ f_2(x,y) &= \frac{1}{2}x + y + 1 \end{aligned}$$

$x, y \geq 0$, $0 \geq \frac{1}{6}x + y - \frac{13}{2}$, $0 \geq \frac{1}{2}x + y - \frac{15}{2}$, $0 \geq 5x + y - 30$.

Ejemplo 5
Este ejemplo fue propuesto por Kursawe [81]:

$$\text{Minimizar } f_1(\vec{x}) = \sum_{i=1}^{n-1} \left(-10 \exp\left(-0{,}2\sqrt{x_i^2 + x_{i+1}^2} \right) \right) \tag{7.60}$$

$$\text{Minimizar } f_2(\vec{x}) = \sum_{i=1}^{n} \left(|x_i|^{0,8} + 5\sin(x_i)^3 \right) \tag{7.61}$$

donde: $-5 \leq x_1, x_2, x_3 \leq 5$

Bibliografía

[1] Hojjat Adeli and Nai-Tsang Cheng. Augmented Lagrangian Genetic Algorithm for Structural Optimization. *Journal of Aerospace Engineering*, 7(1):104–118, January 1994.

[2] Kevin Anchor, Jesse B. Zydallis, Gregg H. Gunsch, and Gary B. Lamont. Extending the Computer Defense Immune System: Network Intrusion Detection with a Multiobjective Evolutionary Programming Approach. In Jonathan Timmis and Peter J. Bentley, editors, *First International Conference on Artificial Immune Systems (ICARIS-2002)*, University of Kent at Catenbury, UK, September 2002.

[3] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, New York, 1997.

[4] James Edward Baker. Adaptive selection methods for genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pages 100–111, Hillsdale, New Jersey, 1985. Lawrence Erlbaum.

[5] James C. Bean and Atidel Ben Hadj-Alouane. A Dual Genetic Algorithm for Bounded Integer Programs. Technical Report TR 92-53, Department of Industrial and Operations Engineering, The University of Michigan, 1992. To appear in R.A.I.R.O.-R.O. (invited submission to special issue on GAs and OR).

[6] Bersini and Varela. The immune recruitment mechanism: A selective evolutionary strategy. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 520–526. Morgan Kaufmann, 1991.

[7] H. Bersini and F. J. Varela. The immune learning mechanisms: Reinforcement, recruiment and their applications. In R. Paton, editor, *Computing with Biological Metaphors*. Chapman & Hall, 1994.

[8] L.B. Booker. *Intelligent behavior as an adaptation to the task environment*. PhD thesis, The University of Michigan, Ann Arbor, MI, 1982.

[9] A. Brindle. *Genetic agorithms for function optimization*. PhD thesis, Department of Computer Science, University of Alberta, Edmonton, Canada, 1981.

[10] F. M. Burnet. Clonal selection and after. In G. I. Bell, A. S. Perelson, and G. H. Pimgley Jr., editors, *Theoretical Immunology*, pages 63–85. Marcel Dekker Inc., 1978.

[11] Erick Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, Massachusetts, USA, 2000.

[12] J. H. Carter. The Immune System as a Model for Pattern Recognition and Classification. *Journal of American Medical Informatics Association*, 7(1):28–41, 2000.

[13] Chan-Jin Chung and Robert G. Reynolds. CAEP: An Evolution-based Tool for Real-Valued Function Optimization using Cultural Algorithms. *Journal on Artificial Intelligence Tools*, 7(3):239–292, 1998.

[14] Carlos A. Coello Coello. Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Enviromental Systems*, 17:319–346, 2000.

[15] Carlos A. Coello Coello. Treating Constraints as Objectives for Single-Objective Evolutionary Optimization. *Engineering Optimization*, 32(3):275–308, 2000.

[16] Carlos A. Coello Coello. Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002.

[17] Carlos Coello Coello. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, 2002.

[18] Carlos Coello-Coello, Daniel Cortés-Rivas, and Nareli Cruz-Cortés. Job shop scheduling using the clonal selection principle. In I.C. Parmee, editor, *Adaptive Computing in Design and Manufacture VI*, pages 113–124. Springer London, Abril 2004.

[19] Carlos A. Coello Coello and Ricardo Landa Becerra. Adding Knowledge and Efficient Data Structures to Evolutionary Programming: A Cultural Algorithm for Constrained Optimization. In W.B. Langdon, E.Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A.C. Schultz, J. F. Miller, E. Burke, and N.Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 201–209, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[20] Carlos A. Coello Coello and Gregorio Toscano Pulido. A Micro-Genetic Algorithm for Multiobjective Optimization. In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 126–140. Springer-Verlag, Lecture Notes in Computer Science No. 1993, Marzo 2001.

[21] Carlos A. Coello Coello and Gregorio Toscano Pulido. Multiobjective Optimization using a Micro-Genetic Algorithm. In L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen,

M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 274–282, San Francisco, California, July 2001. Morgan Kaufmann Publishers.

[22] D. Corne, N. R. Jerram, J. Knowles, and M. J. Oates. PESA-II: Region-Based Selection in Evolutionary Multiobjective Optimization. In L. Spector, E. D. Goodman, A. Wu, W. Langdon, H. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 283–290, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[23] D. Corne, J. Knowles, and M. J. Oates. The Pareto Envelope-Based Selection Algorithm for Multiobjective Optimization. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H. Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, volume 1, pages 839–848, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[24] Nareli Cruz Cortés. Uso de emulaciones del sistema inmune para manejo de restricciones en algoritmos evolutivos. Master's thesis, Maestría en Inteligencia Artificial, UV-LANIA, Xalapa, Veracruz, 2000.

[25] Charles Darwin. *The Origin of Species by Means of Natural Selection or the Preservation of Favored Races in the Struggle for Life*. The Book League of America, 1929. Originally published in 1859.

[26] Indraneel Das and John Dennis. A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems. *Structural Optimization*, 14(1):63–69, 1997.

[27] D. Dasgupta and F. Gonzalez. An Immunity-Based Technique to Characterize Intrisions in Computer Networks. *IEEE Transactions*

on Evolutionary Computation, Special Issue on Artificial Immune Systems, 6(3):281–291, June 2002.

[28] Dipankar Dasgupta, editor. *Artificial Immune Systems and Their Applications*. Springer-Verlag, Berlin, 1999.

[29] Dipankar Dasgupta. Information processing in the immune system. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, 1999.

[30] F. Dasgupta, D. and Nino. A comparison of Negative and Positive Selection Algorithms in Novel Pattern Detection. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Nashville, October 2000.

[31] Yuval Davidor. *Genetic Algorithms and Robotics : A Heuristic Strategy for Optimization*. World Scientific Publishing Co., Singapore, 1990.

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

[33] Leandro Nunes de Castro and Jon Timmis. An artificial immune network for multimodal function optimization. In *Proceedings of the special sessions on artificial immune systems in the 2002 Congress on Evolutionary Computation, 2002 IEEE World Congress on Computational Intelligence*, volume I, pages 669–674, Honolulu, Hawaii, May 2002.

[34] Leandro Nunes de Castro and Jonathan Timmis. *An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm*. Springer-Verlag, 2002.

[35] Leandro Nunes de Castro and F. J. Von Zuben. Learning and Optimization Using the Clonal Selection Principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251, 2002.

[36] K. Deb, S. Agrawl, A. Pratab, and T. Meyerivan. A Fast Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization:NSGA-II. Technical Report KanGAL 200001, Indian Institute of Technology, Kanput, India, 2000.

[37] K. Deb, S. Agrawl, A. Pratab, and T. Meyerivan. A Fast Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization:NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, and H. Schwefel, editors, *Proceedings of the Parallel Problem Solbing from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer, Lecture Notes in Computer Science No. 1917.

[38] Kalyanmoy Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, 2000.

[39] M. Erickson, A. Mayer, and J. Horn. The Niched Pareto Algorithm 2 Applied to the Design of Groundwater Remediation Systems. In E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, and D. Corne, editors, *First International Conference on Artificial Neural Networks*, pages 681–695. Springer-Verlag, Lecture notes on Computer Science No. 1993, 2001.

[40] J. D. Farmer, N. H. Packard, and A. S. Perelson. The Immune System, Adaptation, and Machine Learning. *Physica D*, 22:187–204, 1986.

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

[42] Lawrence J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley & Sons, New York, 1966.

[43] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and General-

ization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[44] S. Forrest, A. Hofmeyr, A. Somayaji, and A. Longstaff. A sense of self for unix processes. In *Proceedings of 1996 IEEE symposium on Computer Security and Privacy*, 1996.

[45] S. Forrest, B. Javornik, R. Smith, and A.S. Perelson. Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3):191–211, 1993.

[46] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 202–212, Oakland, CA, 1994.

[47] Stephanie Forrest and Steven A. Hofmeyr. Immunology as Information Processing. In L.A. Segel and I. Cohen, editors, *Design Principles for the Immune System and Other Distributed Autonomous Systems*, Santa Fe Institute Studies in the Sciences of Complexity, pages 361–387. Oxford University Press, 2000.

[48] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co., Reading, Massachusetts, 1989.

[49] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann Publishers, San Mateo, California, 1991.

[50] P. Hajela and J. Lee. Constrained Genetic Search via Schema Adaptation. An Immune Network Solution. In Niels Olhoff and George I. N. Rozvany, editors, *Proceedings of the First World Congress*

of Stuctural and Multidisciplinary Optimization, pages 915–920, Goslar, Germany, 1996. Pergamon.

[51] Prabhat Hajela and Jun Sun Yoo. Immune network modelling in design optimization. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 167–183. Mc Graw-Hill, 1999.

[52] S. B. Hamida and M. Schoenauer. ASCHEA: New results using adaptive segregationsl constraint handling. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'02)*, volume 1, pages 884–889, Piscataway, New Jersey, 2002. IEEE Service Center.

[53] P. K. Harmer and G. B. Lamont. An Agent Based Arquitecture for a Computer Virus Immune System. In *Proceedings of the Genetic and Evolutionary Computation Conference, Workshop on Artificial Immune Systems and Their Applications*, 2000.

[54] E. Hart, P. Ross, and J. Nelson. Producing robust schedules via an artificial immune system. In *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, Alaska, 1998. IEEE Press.

[55] Emma Hart. The evolution and analysis of a potential antibody library for use in job-shop scheduling. In D. Corne Dorigo and F. Glover, editors, *New Ideas in Optimization*, pages 185–202. McGraw Hill, 1999.

[56] S. A. Hofmeyr and S. Forrrest. Intrusion Detection Using Sequences of System Calls. *Journal of Computer Security*, 1998.

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

[58] A. Homaifar, S. H. Y. Lai, and X. Qi. Constrained Optimization via Genetic Algorithms. *Simulation*, 62(4):242–254, 1994.

[59] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, June 1994. IEEE Service Center.

[60] John E. Hunt and Denise E. Cooke. An adaptative, distributed learning systems based on the immune system. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 2494–2499, 1995.

[61] John E. Hunt and Denise E. Cooke. Learning using and artificial immune systems. *Journal of Network and Computer Applications: Special Issue on Intelligent Systems: Design and Applications*, 19:189–212, 1996.

[62] Y. Ishida and F. Mizessyn. Learning algorithms on an immune network model. In *Proceedings of the International Joint Conference on Neural Networks*, volume I, pages 33–38, China, November 3-6 1992.

[63] A. Ishiguru, Y. Watanable, T. Kondo, Y. Shirai, and H. Uchikawa. Immunoid: a robot with descentralized behavior arbitration mechanisms based on the immune system. In *ICMAS Workshop on Immunity-Based Systems*, December 10 1996.

[64] N. K. Jerne. Towards a network theory of the immune system. *Ann. Immunol.*, 125:373–389, 1974.

[65] Xidong Jin and Robert G. Reynolds. Using Knowledge-Based Evolutionary Computation to Solve Nonlinear Constraint Optimization Problems: a Cultural Algorithm Approach. In *1999 Congress on Evolutionary Computation*, pages 1672–1678, Washington, D.C., July 1999. IEEE Service Center.

[66] J. Joines and C. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In David Fogel, editor, *Proceedings of the first IEEE Conference on Evolutionary Computation*, pages 579–584, Orlando, Florida, 1994. IEEE Press.

[67] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[68] 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.

[69] A. H. Jun, Lee D. W, and Sim K. B. Realization of cooperative and swarm behavior in distributed autonomous robotic systems using artificial immune system. In *Proceedings IEEE SMC'99*, volume 4, pages 614–619, 1999.

[70] Jeffrey O. Kephart, M. Swimmer, and S. R. White. Blueprint for a Computer Immune System. In Dipankar Dasgupta, editor, *Artificial Immune Systems and their Applications*, pages 242–259. Springer-Verlag, 1999.

[71] T. B. Kepler and Perelson A. S. Somatic Hypermutation in B Cells: An Optimal Control Treatment. *Theoretical biology*, 164:37–64, 1993.

[72] Dae Gyu Kim and Phil Husbands. Riemann Mapping Constraint Handling Method for Genetic Algorithms. Technical Report CSRP 469, COGS, University of Sussex, UK, 1997.

[73] Dae Gyu Kim and Phil Husbands. Mapping Based Constraint Handling for Evolutionary Search; Thurston's Circle Packing and Grid Generation. In Ian Parmee, editor, *The Integration of Evolutionary and Adaptive Computing Technologies with Product/System*

Design and Realisation, pages 161–173. Springer-Verlag, Plymouth, United Kingdom, April 1998.

[74] J. Kim and P. Bentley. Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Dynamic Clonal Selection. In *Proceedings of the special sessions on artificial immune systems in the 2002 Congress on Evolutionary Computation, 2002 IEEE World Congress on Computational Intelligence*, 2002.

[75] S. Kirkpatrick, Jr. C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.

[76] H. Kita, Y. Yabumoto, N. Mori, and Y. Nishikawa. Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm. In H. Voigt, W. Ebeling, I. Rechenberg, and H. Scwefel, editors, *Parallel Problem Solving from Nature-PPSN IV*, pages 504–512, Berlin, Germany, 1994. Springer-Verlag. Lecture Notes in Computer Science No. 1141.

[77] J. Knowles and D. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[78] Slawomir Koziel and Zbigniew Michalewicz. A Decoder-based Evolutionary Algorithm for Constrained Parameter Optimization Problems. In T. Bäck, A. E. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th Parallel Problem Solving from Nature (PPSN V)*, pages 231–240, Amsterdam, September 1998. Springer-Verlag.

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

[80] J. Krishnakumar, K. and Neidhoefer. Immunized adaptive critic for an autonomous aircraft control application. In Dipankar Dasgupta,

editor, *Artificial Immune Systems and Their Applications*, pages 221–241. Springer-Verlag, 1999.

[81] Frank Kursawe. A Variant of Evolution Strategies for Vector Optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, Germany, oct 1991. Springer-Verlag.

[82] T. Van Le. A Fuzzy Evolutionary Approach to Constrained Optimization Problems. In *Proceedings of the Second IEEE Conference on Evolutionary Computation*, pages 274–278, Perth, November 1995. IEEE.

[83] D.-W. Lee, Jun H.-B., and Sim K-B. Artificial immune system for realization of cooperative strategies and group behavior in collective autonomous mobiel robots. In *Proceedings of theAROB'99 (International Symposium on Artificial Life and Robotics*, pages 232–235, 1999.

[84] M. A. Lee and H. Esbensen. Fuzzy/Multiobjective Genetic Systems for Intelligent Systems Design Tools and Components. In Witold Pedrycz, editor, *Fuzzy Evolutionary Computation*, pages 57–80. Kluwer Academic Publishers, Boston, Massachusetts, 1997.

[85] G. C. Luh, C. H. Chueh, and W. W. Liu. MOIA: Multi-Objective Immune Algorithm. *Engeneering Optimization*, 35(2):143–164, 2003.

[86] Adriana Lara López. Un estudio de las estrategias evolutivas para problemas multiobjetivo. Master's thesis, CINVESTAV-IPN, Departamento de Ingeniería Eléctrica, Sección de Computación, México, D. F., 2003.

[87] Efrén Mezura-Montes and Carloa A. Coello Coello. Adding a Diversity Mechanism to a Simple Evolution Strategy to Solve Constrained Optimization Problems. In *Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003)*, volume 1, pages 6–13,

Piscataway, New Jersey, December 2003. Canberra, Australia, IEEE Service Center.

[88] Z. Michalewicz and N. Attia. Evolutionary Optimization of Constrained Problems. In *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 98–108. World Scientific, 1994.

[89] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, second edition, 1992.

[90] Zbigniew Michalewicz, Dipankar Dasgupta, R. Le Riche, and Marc Schoenauer. Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering Journal*, 30(4):851–870, September 1996.

[91] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[92] Eric Michielssen, Jean-Michel Sajer, S. Ranjithan, and Raj Mittra. Design of Lightweight, Broad-Band Microwave Absorbers Using Genetic Algorithms. *IEEE Transactions on Microwave Theory and Techniques*, 41(6/7):1024–1031, 1993.

[93] K. M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, Massachusetts, 1998.

[94] K. Mori, M. Tsukiyama, and T. Fukuda. Adaptive scheduling system inspired by immune system. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, 1998.

[95] Stephen R. Norris and William A. Crossley. Pareto-Optimal Controller Gains Generated by a Genetic Algorithm. In *AIAA 36th Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, January 1998. AIAA Paper 98-0010.

[96] T. Okamoto and Y. Ishida. A distributed approach against computer virus inspired by the immune system. In *IEICE Transactions on Communications*, volume E83-B, pages 908–915, Tokyo, 2000.

[97] J. T. Ootsuki and T. Sekiguchi. Application of the immune system network concept to sequential control. In *Proceedings of the IEEE System, Man, and Cybernetics (SMC'99)*, pages 869–874, 1999.

[98] J. Paredis. Co-evolutionary Constraint Satisfaction. In *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, pages 46–55, New York, 1994. Springer Verlag.

[99] Vilfredo Pareto. *Cours D' Economie Politique, volume I and II.* F. Rouge, Lausanne, 1896.

[100] I. C. Parmee and G. Purchase. The development of a directed genetic search technique for heavily constrained design spaces. In I. C. Parmee, editor, *Adaptive Computing in Engineering Design and Control-'94*, pages 97–102, Plymouth, UK, 1994. University of Plymouth.

[101] Alan S. Perelson. Immune network theory. *Immunological Reviews*, 10:5–36, 1989.

[102] Alan S. Perelson and G. F. Oster. Theoretical Studies of Clonal Selection: Minimal Antibody Repertoire Size and Reability of Self-Nonself Discrimination. *Journal theor. Biol.*, 81:645–670, 1979.

[103] David Powell and Michael M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 424–431, San Mateo, California, jul 1993. University of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers.

[104] Tapabrata Ray, Tai Kang, and Seow Kian Chye. An Evolutionary Algorithm for Constrained Optimization. In Darrell Whitley, David

Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 771–777, San Francisco, California, July 2000. Morgan Kaufmann.

[105] Robert G. Reynolds. An Introduction to Cultural Algorithms. In A. V. Sebald, , and L. J. Fogel, editors, *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 131–139. World Scientific, River Edge, New Jersey, 1994.

[106] Robert G. Reynolds, Zbigniew Michalewicz, and M. Cavaretta. Using cultural algorithms for constraint handling in GENOCOP. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 298–305. MIT Press, Cambridge, Massachusetts, 1995.

[107] 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.

[108] Rodolphe G. Le Riche, Catherine Knopf-Lenoir, and Raphael T. Haftka. A Segregated Genetic Algorithm for Constrained Structural Optimization. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 558–565, San Mateo, California, July 1995. University of Pittsburgh, Morgan Kaufmann Publishers.

[109] Brian J. Ritzel, J. Wayland Eheart, and S. Ranjithan. Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. *Water Resources Research*, 30(5):1589–1603, may 1994.

[110] Günter Rudolph. Evolutionary Search for Minimal Elements in Partially Ordered Finite Sets. In V.W. Porto, N. Saravanan, D. Waagen, and A.E. Eiben, editors, *Proceedings of the 7th Annual Conference*

on Evolutionary Programming, pages 345–353. Berlin: Springer, 1998.

[111] Günter Rudolph. Convergency properties of some multiobjective evolutionary algorithms. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000)*, volume 2, pages 1010–1016, Piscataway (NJ), 2000. IEEE Press.

[112] Günter Rudolph. Evolutionary search under partially ordered fitness sets. In *Proceedings of the International Symposium on Information Science Innovations (ISI 2001) in Engineering of Natural and Artificial Intelligent Systems ENAIS*, pages 818–822. ICSC Academic Press: Millet/Sliedrecht, 2001.

[113] Thomas P. Runarsson and Xin Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionay Computation*, 4(3):284–294, 2000.

[114] Rosa Saab, Raúl Monroy, and Fernando Godínez. Towards a model for an immune system. In *MICAI 2002: Advances in Artificial Intelligence, Second Mexican International Conference on Artificial Intelligence*, pages 401–410, Mérida, Yucatán, México, April 2002. Springer.

[115] Rosa Saab-Asbun. Modelo de un sistema inmunológico natural en álgebra de procesos. Master's thesis, Maestría en Ciencias de la Computación, Instituto Tecnológico de Estudios Superiores de Monterrey Campus Estado de México, México., 2002.

[116] R. Sarker, M. Mohammadian, and X. Yao, editors. *Evolutionary Optimization*. Kluwer Academic Publishers, Estados Unidos de America, 2002.

[117] J. David. Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.

[118] J. David Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[119] Marc Schoenauer and Zbigniew Michalewicz. Evolutionary Computation at the Edge of Feasibility. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the Fourth Conference on Parallel Problem Solving from Nature*, pages 245–254. Springer-Verlag, Berlin, September 1996.

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

[121] Jason R. Schott. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology,Cambridge, Massachusetts, 1995.

[122] Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Great Britain, 1981.

[123] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, New York, 1995.

[124] E. Seneta. *Non-Negative Matrices and Markov Chains*. Springer-Verlag, New York, second edition, 1981.

[125] Margarita Reyes Sierra. Estudio de algunos aspectos teóricos de los algoritmos genéticos. Master's thesis, Maestría en Inteligencia Artificial, UV-LANIA, Xalapa, Veracruz, México, 2002.

[126] Derek Smith. Towards a Model of Associative Recall in Immunological Memory. Technical Report 94-9, University of New Mexico, Albuquerque, NM, 1994.

[127] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, fall 1994.

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

[129] 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.

[130] 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, jun 1989. George Mason University, Morgan Kaufmann Publishers.

[131] A. Takahashi and T. Yamada. A self-tuning immune feedback controller for controlling mechanical systems. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, CD-ROM No.1*, 1997.

[132] A. Tarakanov and D. Dasgupta. A formal model of an artificial immune system. *BioSystems*, 55:151–158, 2000.

[133] A. Tarakanov, S. Sokolava, B. Abramov, and A. Aikimbayev. Immunocomputing of the Natural Plague Foci. In *Proceedings of the Genetic and Evolutionary Computation Conference, Workshop on Artificial Immune Systems and Their Applications*, pages 38–39, 2000.

[134] N. Toma, S. Endo, and K. Yamada. The Proposal and Evaluation of an Adaptive Memorizing Immune Algorithm with Memory Mech-

anisms. *Journal of Japanese Society for Artificial Intelligence*, 15(6):1097–1106, 2000.

[135] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithm Research: A History and Analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.

[136] David Van Veldhuizen and Gary Lamont. Evolutionary Computation and Convergence to a Pareto Front. In *Late Breaking Papers at the Genetic Programming*, pages 221–228, Stanford University, California, 1998. Stanford University Bookstore.

[137] Rémy Viennet, Christian Fontiex, and Ivan Marc. New Multicriteria Optimization Method Based on the Use of a Diploid Genetic Algorithm: Example of an Industrial Problem. In J. M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Proceedings of Artificial Evolution (European Conference, selected papers)*, pages 120–127, Brest, France, September 1995. Springer-Verlag.

[138] Mario Villalobos-Arias, Carlos A. Coello Coello, and Onésimo Hernández-Lerma. Convergence analysis of a multiobjective artificial immune system. In *Proceedings of the Third International Conference on Artificial Immune Systems (ICARIS'2004)*. Springer. Lecture Notes in Computer Science, September 2004.

[139] Y. Wantanabe, A. Ishiguro, Y. Shirai, and Y. Uchikawa. Emergent construction of a behavior arbitration mechanism based on the immune system. *Advanced Robotics*, 12(3):227–242, 1998.

[140] J. Yoo and P. Hajela. Immune network simulations in multicriterion design. *Structural Optimization*, 18:85–94, 1999.

[141] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[142] E. Zitzler and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms:Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

[143] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. In K. Giannakoglou, D. Tsahalis, J. Periauxand, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002.