INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

SISTEMA INMUNE ARTIFICIAL CON POBLACIÓN REDUCIDA PARA OPTIMIZACIÓN NUMÉRICA

T E S I S

QUE PARA OBTENER EL GRADO DE
DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A
JUAN CARLOS HERRERA LOZADA

DIRECTORES DE TESIS:
DR. FRANCISCO HIRAM CALVO CASTRO
DRA. HIND TAUD

MÉXICO, D.F. JULIO DE 2011

A Santiago, mi hijo, quien es mi mejor maestro.
A Ana, mi esposa, con quien comparto mi vida.
A mis padres y a mis hermanos.

Sirva el presente espacio para agradecer a todas las personas que directa o indirectamente colaboraron en la realización de esta tesis.

A los Dres. Hiram Calvo Castro y Hind Taud, mis asesores, quienes tuvieron a bien confiar en mi persona y en el tema de investigación. Sin ellos no se hubiera podido concluir satisfactoriamente este trabajo.

A los Dres. Marco A. Moreno Armendáriz, Jesús G. Figueroa Nazuno y Edgar A. Portilla Flores, quienes fungieron como revisores de esta tesis, y a quienes respeto y admiro.

A los Dres. Oscar Camacho Nieto, Grigori Sidorov y Víctor Ponce Ponce, por todo el apoyo brindado durante mi estancia como alumno del doctorado.

Al Departamento de Tecnologías Educativas del CIC, por el excelente trabajo que desempeña, particularmente a la Lic. Lourdes Olvera Cárdenas, a María del Carmen Morales Pitayo y a Silvia Arteaga Cadena.

A las autoridades del CIDETEC, por el apoyo administrativo brindadoes para estudiar el doctorado, especialmente al Dr. Víctor M. Silva García y a los Mtros. Eduardo Rodríguez Escobar y Juan C. González Robles.

A los Mtros. Jesús A. Álvarez Cedillo y Mauricio Olguín Carbajal, por compartir esta etapa de formación conmigo.

A la Secretaría Académica del IPN por permitirme, a través del Programa Institucional de Año Sabático, concluir esta tesis.

Índice general

Capítulo 4

Capítulo 5

Referencias

Índice de figuras y tablas

SIP-14 BIS

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 12:00 horas del día 21 del mes de Junio de 2011 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"SISTEMA INMUNE ARTIFICIAL CON POBLACIÓN REDUCIDA PARA OPTIMIZACIÓN NUMÉRICA"

Presentada por el alumno:

HERRERA	**LOZADA**	**JUAN CARLOS**
Apellido paterno	Apellido materno	Nombre(s)

Con registro: **B 0 7 1 2 3 4**

aspirante de: **DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

Dr. Francisco Hiram Calvo Castro

Dra. Hind Taud

Dr. Jesús Guillermo Figueroa Nazuno

Dr. Marco Antonio Moreno Armendáriz

Dr. Edgar Alfredo Portilla Flores

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de ***México, D.F.*** el día ***23*** del mes de ***Junio*** del año ***2011***, el que suscribe ***Juan Carlos Herrera Lozada***, alumno del Programa de ***Doctorado en Ciencias de la Computación*** con número de registro ***B071234***, adscrito al ***Centro de Investigación en Computación***, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del ***Dr. Francisco Hiram Calvo Castro*** y ***Dra. Hind Taud***, cede los derechos del trabajo intitulado ***Sistema Inmune Artificial con Población Reducida para Optimización Numérica***, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección ***jlozada@ipn.mx***. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Juan Carlos Herrera Lozada

Resumen

En este trabajo de tesis se presenta un nuevo algoritmo, se trata de un micro-sistema inmune artificial (denominado micro-SIA), basado en la teoría de la selección clonal, para resolver problemas de optimización numérica. Para este estudio se empleó el algoritmo CLONALG en su versión estándar, que es un sistema inmune artificial ampliamente utilizado para resolver problemas de optimización y de reconocimiento de patrones. Durante la fase de clonación, CLONALG incrementa drásticamente el tamaño de su población, por lo que esta característica es atractiva para proponer una versión con una población de individuos reducida. La hipótesis que se estableció en este trabajo fue que al reducir el número de individuos en la población, también se reduce el número de evaluaciones a la función objetivo, con lo que fue posible incrementar la velocidad de convergencia y decrementar el uso de la memoria de datos. El micro-SIA propuesto usa una población de 5 individuos (anticuerpos) de los cuales se obtienen sólo 15 clones. En el proceso de maduración de estos clones, dos simples y rápidos operadores de mutación fueron diseñados y utilizados dentro de una convergencia nominal que trabaja en conjunto con un proceso de reinicialización para preservar la diversidad.

Se realizaron dos versiones del micro-SIA para optimización numérica, la primera de éstas no considera el manejo de restricciones, sin embargo la mayoría de los problemas de optimización en el mundo real tienen restricciones, por lo que una segunda versión del micro-SIA, sí permite lidiar con éstas. También se presentan dos aproximaciones del micro-SIA embebidas en hardware y que se ejecutan intrínsecamente para resolver el problema de la maximización de unos en una cadena finita (MaxOne Problem), la primera se implementó en un microcontrolador comercial de 8 bits y la segunda en un dispositivo de lógica reconfigurable (FPGA) estimando cadenas de 16 bits. La experimentación con el micro-SIA finalizó con una aplicación de reconocimiento de caracteres.

Abstract

This thesis presents a new algorithm, it is a micro-artificial immune system (micro-SIA) based on the clonal selection theory to solve numerical optimization. For this study it was considered the algorithm named CLONALG, it is a widely used artificial immune system. During the process of cloning, CLONALG greatly increases the size of its population, so this feature is attractive to propose a version with a reduced population. The established hypothesis is that by reducing the number of individuals in a population will decrease the number of evaluations to the objective function, increasing the speed of convergence and reducing the use of data memory. The proposed micro-SIA uses a population of 5 individuals (antibodies) which are obtained only 15 clones. In the maturation stage of the clones, two simple and fast mutation operators are used in a nominal convergence that works together with a reinitialization process to preserve the diversity.

Two versions of the micro-SIA to solve global numerical optimization problems were realized. The first version does not consider constraints handling. However, the majority of optimization problems in the real world have constraints, so a second version of the micro-SIA, it does allow dealing with constraints.

This thesis also presented two approaches of micro-SIA implemented in hardware. These architectures allow the intrinsic execution of the algorithm to solve the MaxOne problem in a finite string. The first approach is implemented in a commercial 8-bit microcontroller. The second architecture was implemented on a reconfigurable logic device (FPGA). Finally, experimentation with the micro-SIA concluded with the design and implementation of a character recognition application.

Capítulo 1

Introducción

Los algoritmos con inspiración biológica, también conocidos como algoritmos bioinspirados, han adquirido gran importancia dentro del área de la inteligencia artificial debido a que han demostrado ser exitosos en la solución de ciertos problemas complejos de aprendizaje de máquina, reconocimiento de patrones, detección de fallas y optimización. El sistema inmune artificial (SIA), inspirado por algunos procesos observados principalmente en el sistema inmune biológico de los mamíferos, es un área emergente en los sistemas computacionales adaptativos que incluye algoritmos que han sido empleados satisfactoriamente, entre otros, a problemas de optimización global como combinatoria, tratándose de una técnica heurística relativamente nueva y que ha resultado altamente competitiva. En el sistema inmune biológico, el principio de selección clonal es uno de los procesos nativos más estudiados y que es abstraído directamente como un algoritmo particular del SIA, utilizándose principalmente para resolver problemas de optimización numérica y reconocimiento de patrones.

Al igual que sucede con los algoritmos evolutivos, el principio de selección clonal del SIA es *poblacional*, lo que significa que manipula simultáneamente un conjunto de soluciones potenciales del problema, implicando en la mayoría de los casos un tiempo de ejecución elevado y un amplio uso de la memoria de datos que se traduce en un alto costo computacional. Una alternativa para reducir tal costo ha sido el diseño de algoritmos con poblaciones extremadamente pequeñas (micro-algoritmos) que al trabajar con pocos individuos realizan menos evaluaciones de la función objetivo utilizando por ende un menor espacio en la memoria de datos del sistema de cómputo. Cumpliendo con una elección correcta de los parámetros sensibles en este tipo algoritmos, por ejemplo el tamaño de la población o la probabilidad de

mutación, entre otras variables, es posible acelerar la convergencia en comparación a las versiones extendidas de los mismos.

El principio de selección clonal se caracteriza por aumentar el tamaño de la población durante la etapa de clonación, lo que exige la manipulación de más individuos que los existentes al inicio del algoritmo, y sólo permite aplicar operadores de mutación (no hay operador de cruza), por lo que mantener la diversidad de la población y asegurar la convergencia del algoritmo, son retos que hacen interesante el estudio de esta heurística bioinspirada.

En este trabajo de tesis se presenta un algoritmo basado en el principio de selección clonal del sistema inmune artificial, nombrado *micro-SIA* (micro-sistema inmune artificial), que utiliza una población reducida en su número de individuos (anticuerpos, en la terminología propia del SIA) y que fue diseñado primeramente en software para solucionar problemas de optimización numérica y que posteriormente se implementó intrínsecamente en hardware. Las versiones en software del micro-SIA se presentan en las dos vertientes abordadas en la literatura especializada para optimización de un solo objetivo: sin manejo de restricciones y con manejo de espacios restringidos.

Se realizaron dos arquitecturas del micro-SIA propuesto, la primera en un microcontrolador comercial y la segunda en un dispositivo de lógica reconfigurable (FPGA). El *hardware evolutivo*, llamado así por ser una migración de la versión en software del algoritmo bioinspirado a una versión en alguna plataforma en hardware, puede ayudar a solucionar los problemas de diseño de circuitos, de tolerancia a fallos y adaptación a entornos cambiantes. Lo anterior se debe a que hardware evolutivo no usa reglas de diseño rígidas, sólo se guía por el comportamiento del circuito y por lo tanto el espacio de soluciones sobre el que trabaja es mayor, además un diseño complejo permite que el sistema se rediseñe de manera autónoma, cuando se presente una falla o bien cuando las características del entorno varíen.

1.1 Algoritmos evolutivos y bioinspirados

Las técnicas heurísticas con inspiración biológica, también referidas en la literatura especializada como algoritmos evolutivos y bioinspirados [10, 11, 12] son procedimientos de búsqueda, optimización y aprendizaje, cuyo modelo se obtiene del mecanismo de la selección natural y las teorías de la evolución, basadas en el paradigma *Neo-Darwiniano* que aplica sobre esquemas poblacionales. Estas técnicas son muy populares, debido principalmente a su alto poder exploratorio y a su paralelismo implícito.

El Neo-Darwinismo es el conjunto de teorías que explican el origen de las especies en el planeta y cómo sus procesos de reproducción, mutación, competencia y selección permiten la adaptación y evolución de las especies; las teorías involucradas en este modelo son: el *seleccionismo de Weismann*, el *origen de las especies* formulado por Darwin y la *genética de Mendel* [10]. Todos estos algoritmos tienen en común el hecho de utilizar una población o conjunto de soluciones potenciales, generada comúnmente de manera aleatoria, y someterla a un proceso iterativo utilizando diferentes esquemas, operadores y estrategias en función del tipo de algoritmo.

En la particularidad de los algoritmos evolutivos, los operadores genéticos básicos son tres [11]: la *selección*, la *cruza* (reproducción o recombinación) y la *mutación*. La selección consiste de un mecanismo (puede ser probabilístico o determinista) que permite elegir a los individuos que fungirán como padres de la siguiente generación o iteración. La cruza se refiere al intercambio de información (material genético) entre dos padres que han sido seleccionados con base en su aptitud de acuerdo a una función objetivo. El último de los operadores listados, la mutación, se encarga de realizar pequeñas perturbaciones o cambios mínimos 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, manteniendo la diversidad de los individuos.

La secuenciación de los pasos implicados en un algoritmo evolutivo estándar se lista en el algoritmo mostrado en la figura 1.1.

1. *Inicialización:* Población inicial generada aleatoriamente.

2. *Generación:* Aplicar los siguientes procedimientos evolutivos de adaptación,
   - 2.1. *Selección:* Se seleccionan los individuos que sobrevivirán y se reproducirán, de acuerdo a su aptitud con base en la función objetivo.
   - 2.2. *Reproducción y Variación Genética:* Se crean nuevos individuos por medio de la recombinación y/o introduciendo variaciones genéticas (mutación) en individuos seleccionados.
   - 2.3. *Evaluación de Aptitud:* evaluar nuevamente la aptitud de cada individuo modificado.

3. *Ciclo:* Repetir el paso 2 hasta que se alcance el criterio de convergencia.

Figura 1.1. Secuenciación de un algoritmo evolutivo estándar.

En la figura 1.2 se describe gráficamente el comportamiento y la ejecución del algoritmo evolutivo estándar, obsérvese que los operadores genéticos modifican las características de los individuos en el modelo poblacional.

Existen básicamente dos etapas de evaluación de la función objetivo, la primera en la *Selección* y posteriormente en la *Reselección* después de haber aplicado los operadores genéticos.

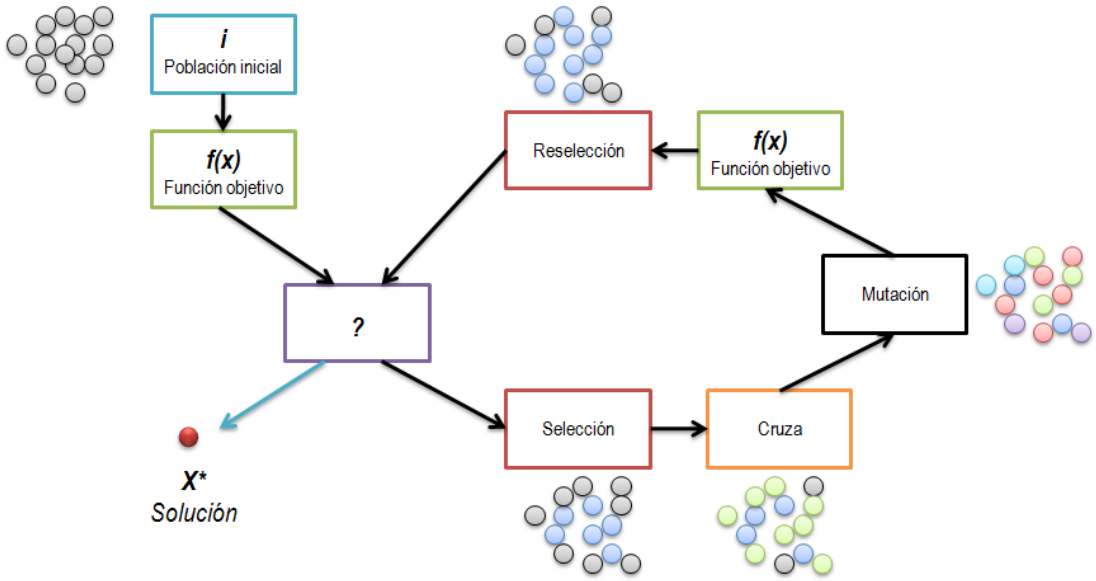


Figura 1.2. Comportamiento de un algoritmo evolutivo estándar.

Han surgido diferentes disciplinas que intentan imitar a través de medios computacionales, los procesos antes indicados, con la iniciativa de resolver problemas complejos *NP*, lineales y no lineales. Una de estas disciplinas es la denominada *Computación Evolutiva* [12] que conjunta a su vez a la *Programación Genética*, a los *Algoritmos Genéticos* y a las *Estrategias Evolutivas*, que durante años han representado la viabilidad de utilizar este tipo de recursos para solucionar satisfactoriamente problemas de optimización a un coste computacional razonable.

Fogel explica en [11] que los tres paradigmas antes mencionados tienen diferencias entre sí, específicamente en la importancia que cobran los operadores genéticos de cruza y mutación, sin embargo, comparten funciones comunes como la reproducción, la variación aleatoria, el ímpetu para simular la evolución, la competencia y la selección, además de ser algoritmos de naturaleza estocástica.

Otra disciplina que surge como una respuesta al análisis del comportamiento social y cultural de un sistema visto como una colectividad o colonia, es la denominada *Inteligencia de Enjambre* [14] que básicamente trata de explicar

cómo se relacionan e intercambian información entre sí los individuos de estas sociedades, con la premisa de resolver una tarea común.

La integración y la coordinación de tareas de los individuos no requiere de ningún supervisor, es decir, los individuos resuelven los problemas ligados a su supervivencia de una manera distribuida y paralela. En la optimización con este tipo de técnicas, la población demanda una memoria, refiriéndose al hecho de que el proceso de mejora se dirige y encauza influenciado por la historia grupal, por la memoria de cada individuo y por el estado presente en el que cada uno se encuentra.

Aunque la inteligencia de enjambre es aplicable al comportamiento humano, se ha estudiado más a fondo sobre insectos. Los paradigmas más conocidos son la *Colonia de Hormigas* [15] y el *Cúmulo de Partículas* [16], este último inspirado en la coreografía de las parvadas de pájaros o los bancos de peces. Existen algunos otros de reciente creación y estudio, como la *Comunicación entre Luciérnagas* [17]. En la figura 1.3 se ilustra el paradigma de la Colonia de Hormigas, en donde se emula el comportamiento de las hormigas para encontrar el camino o ruta más corta entre el alimento y su nido, apoyándose de la concentración de feromonas para guiar la búsqueda.

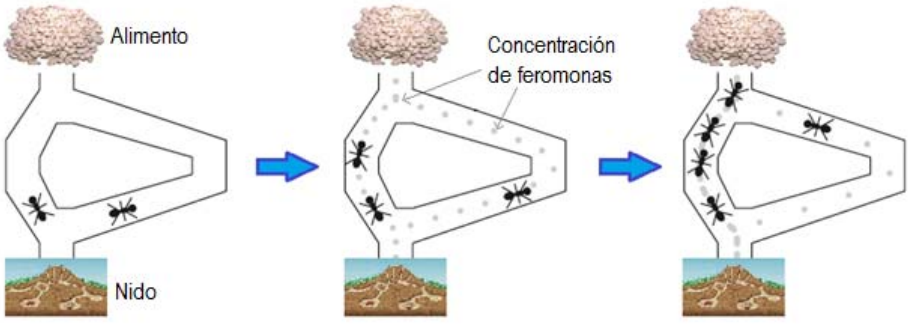


Figura 1.3. Paradigma de la colonia de hormigas.

Dentro de la literatura del área, se pueden encontrar otros modelos bioinspirados que no es sencillo clasificar, por ejemplo, la *Molécula de ADN* que se fundamenta de manera más formal en la llamada *Computación Molecular* y que básicamente ordena secuencias de cadenas de ADN para posteriormente compararlas buscando la hibridación de las mismas cuando sean complementarias [18].

Otro ejemplo son las *Redes Neuronales* que representan un modelo de aprendizaje y procesamiento automático que está inspirado en la forma en que funciona el sistema nervioso de los animales [19]. En la figura 1.4 se muestra una *red neuronal artificial (ANN)* totalmente conectada; en ésta se observa la interconexión de neuronas (elementos básicos de procesamiento) que colaboran para producir *n* estímulos de salida, emulando el razonamiento humano.

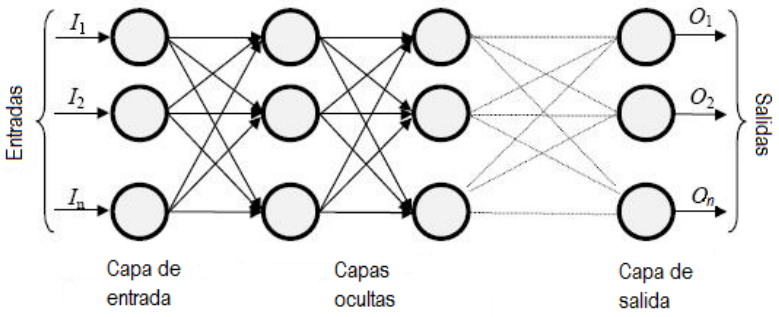


Figura 1.4. Red neuronal artificial.

Para algunos autores como Dasgupta [20], los sistemas de procesamiento de información inspirados biológicamente pueden clasificarse de la siguiente manera: sistema nervioso cerebral (redes neuronales artificiales), sistemas genéticos (algoritmos evolutivos) y sistemas inmunes (sistema inmune artificial).

1.2 Hardware evolutivo

Cuando se trasladan las técnicas evolutivas y bioinspiradas a alguna plataforma en Hardware, surge el término de *Hardware Evolutivo* [1, 3]. Es posible percibir dos ramas principales, concernientes a la operatividad del hardware, la primera de ellas es la reconfiguración automática que persigue una adaptación autónoma del hardware y, por otra parte, la implantación física de algoritmos evolutivos para acelerar algún proceso de búsqueda [2].
En la reconfiguración automática se pretende que un dispositivo electrónico pueda recibir datos del exterior, y de manera autónoma, pueda éste llevar a cabo una recombinación de su estructura interna para adaptarse al medio y mejorar su desempeño.

La implantación de algoritmos, que es la aplicación que se aborda en este trabajo de tesis, surge como una alternativa para acelerar los procesos internos y validar acciones que el hardware debe realizar, mejorando considerablemente las soluciones realizadas en software y procesadas a través de una computadora con secuenciación de instrucciones. Las opciones de implantación conllevan aproximaciones electrónicas digitales y analógicas, y en algunos casos, los menos, híbridas que combinan ambas vertientes.

El hardware evolutivo se puede implementar de dos maneras sobre silicio [35]: de forma *extrínseca*, con actualización fuera de línea, es decir, se simula todo el algoritmo y posteriormente se busca la manera de implantarlo en la plataforma hardware, o bien, de modo *intrínseco* en donde es posible integrar el algoritmo de forma embebida en el mismo dispositivo, lo cual repercute en una mayor velocidad de procesamiento y se puede hablar de reconfiguración y adaptación al medio de manera autónoma, aunque aún no se han logrado resultados totalmente independientes. Para algunos autores, es posible concebir estructuras combinadas [3].

Los dispositivos de lógica programable, con arquitecturas flexibles y robustas, como por ejemplo los *FPGAs* (*Arreglos de Compuertas Programables en Campo*) han sido durante años los recursos más utilizados para probar el

funcionamiento de toda clase de algoritmos evolutivos y bioinspirados, en un circuito integrado digital [3, 29, 35]. Lo anterior se debe a la disposición física de su arquitectura, que de manera natural está construida como un arreglo de bloques que pueden interconectarse de manera creciente entre sí para generar circuitos lógicos más complejos, aunado al gran número de pines de propósito general. En la figura 1.5 se muestra la arquitectura básica de un FPGA con sus tres elementos fundamentales: los *bloques de E/S* que propiamente se encargan del intercambio de datos con el exterior, los *bloques lógicos* (también conocidos como módulos lógicos configurables) que son las unidades en donde se implanta la lógica de solución y las *interconexiones* que son controladas por el tipo de tecnología programable que a la vez facilita la conexión entre los bloques lógicos. Entre las tecnologías programables más importantes se tienen las tecnologías SRAM y anti-fusible. La primera es reprogramable y la segunda es programable en una sola ocasión. Es claro que para prototipos experimentales es más socorrido el uso de la tecnología SRAM.

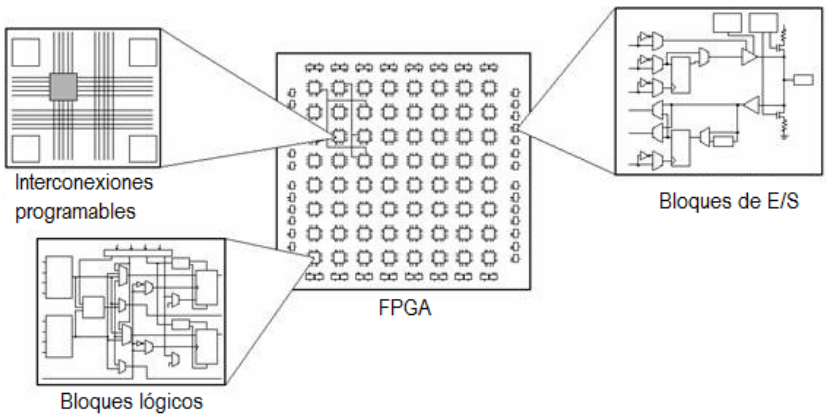


Figura 1.5. Disposición interna de un FPGA.

Las aplicaciones relativas al hardware evolutivo vertido en dispositivos FPGA son muy variadas, encontrándose desarrollos en robótica [36], en el diseño de filtros digitales [37, 38], y en el reconocimiento de patrones [39], entre otras tantas referidas principalmente a la ingeniería. La implementación analógica se realiza a nivel sustrato, proponiendo circuitos en base a transistores, resistencias y capacitores, en donde se busca evolucionar estos componentes para diseñar circuitos tolerantes a fallas y autoadaptativos [40].

1.3 Objetivo General

Diseñar un algoritmo basado en el sistema inmune artificial, con un tamaño de población reducido, para solucionar problemas de optimización numérica.

1.3.1 Objetivos Particulares

1. Realizar un estudio de los diferentes micro - algoritmos evolutivos y bioinspirados existentes en la literatura especializada, para encontrar las características que permitan diseñar un sistema inmune artificial con población reducida.

2. De los procesos biológicos del sistema inmune, se dirigirá el estudio al principio de selección clonal, considerando sus principales características: clonación y mutación.

3. Diseñar los operadores de mutación que permitan el funcionamiento correcto del algoritmo.

4. Realizar los análisis y pruebas pertinentes que permitan validar el desempeño del algoritmo.

5. Dirigir los resultados hacia arquitecturas hardware y plantear posibles aplicaciones futuras.

1.4 Justificación

Existe un interés creciente por utilizar algoritmos evolutivos y bioinspirados para optimizar procesos. La adecuación de las soluciones antes exploradas en software y ahora trasladadas a hardware, deja entrever la inmediata ventaja de la velocidad de procesamiento. Considerando las herramientas actuales en el diseño electrónico, tanto analógico como digital, es posible realizar diversos estudios y abstracciones de las técnicas conocidas.

El paralelismo inherente de los algoritmos ha sido ampliamente abordado en el diseño de *Hardware Evolutivo*, especialmente cuando se utiliza un dispositivo lógico programable en donde ha quedado de manifiesto que las velocidades de procesamiento y de convergencia son muy aceptables y no se escatima en la cantidad de recursos dentro de la arquitectura [1],[2],[3]; sin embargo, son muy pocos los estudios que pretenden reducir la arquitectura en función de los módulos lógicos, o bien, que involucren en estas tendencias a los dispositivos comerciales con restricciones de memoria y con ejecución procedural, como sucede con los microcontroladores comerciales.

La hipótesis planteada en esta tesis es que al disminuir la cantidad de individuos en la población de trabajo de un algoritmo bioinspirado, bajo algún criterio que considere los parámetros sensibles involucrados, también se reduzca el número de evaluaciones de la función objetivo, permitiendo un uso más eficiente de la memoria de datos, sin afectar de manera importante el funcionamiento del algoritmo. Con base a esta suposición, la literatura especializada reporta algunos mecanismos que buscan reducir la cantidad de evaluaciones de la función objetivo caracterizando alguna variante en el manejo de la población, como sucede con el micro-algoritmo genético [6] que representa nuestra principal referencia y que como se indicará posteriormente utiliza una convergencia nominal, un proceso de reinicialización, considera elitismo y genera ruido estocástico al completar una nueva población con nuevos individuos obtenidos aleatoriamente.

Con respecto al sistema inmune artificial para optimización, no existe referencia alguna de algoritmos con población reducida y en consecuencia tampoco se tienen evidencias de implementaciones en hardware, por lo que las aportaciones de esta tesis serán las primeras sobre el tema.

1.5 Contribuciones de la tesis

Se diseñó e implementó en software, un micro-SIA con una población de sólo 5 individuos (anticuerpos), para resolver problemas de optimización numérica global sin restricciones y con restricciones. Cabe mencionar que no existe en la literatura especializada ningún trabajo previo que aborde al SIA y que utilice una población reducida en tamaño. Para ambos casos se presentan los criterios que permitieron el funcionamiento correcto del algoritmo. Se pudo demostrar que el algoritmo propuesto presenta una convergencia más rápida en comparación a la versión estándar o extendida del SIA denominada CLONALG, que al igual que el micro-SIA propuesto, está inspirado en el principio de la selección clonal.

Debido a que se logró la meta del diseño del micro-SIA y funcionó correctamente en software con base en los experimentos realizados, se procedió a la segunda etapa de la investigación doctoral que fue la implantación del micro-SIA en una plataforma de hardware y que éste se ejecutara de forma intrínseca. El carácter intrínseco está definido por la particularidad de que el algoritmo está embebido en el dispositivo y se ejecuta de forma autónoma y no sólo recibe datos provenientes de una PC que previamente ejecuta el algoritmo y entrega resultados para configurar el hardware, como sucedería con una implementación extrínseca.

Es importante mencionar que para las aproximaciones en hardware fue necesario adecuar el micro-SIA para trabajar con una representación numérica binaria y no real, como sucedía con las versiones en software. Se realizaron dos aproximaciones para resolver el problema de la maximización de unos en una cadena finita (*maxone problem*); la primera se realizó con éxito en un

microcontrolador comercial, con recursos internos limitados, utilizando registros de 8 bits (que a la vez es el tamaño máximo de la cadena) y la segunda se programó en un dispositivo con arquitectura flexible, en este caso en un FPGA, definiendo registros de 16 bits. En este último se logró una arquitectura simplificada y modular.

El problema de la maximización de unos es una aplicación clásica en el área de reconocimiento de patrones; no obstante resulta interesante su implementación en hardware, no siendo una tarea trivial debido a las características modulares de un algoritmo bioinspirado, es decir, la realización física de módulos que generen números aleatorios, ordenen datos y apliquen los respectivos operadores genéticos, además se requiere de un bloque adicional que permita el conteo eficiente de los bits puestos a uno para medir la aptitud de un individuo. Finalmente, se presenta una aplicación de reconocimiento de caracteres que hace uso del micro-SIA, implementado en el FPGA, para obtener el caracter más parecido a otro corrupto que se utiliza como entrada en el sistema de reconocimiento, utilizando una matriz de 35 bits para cada caracter (7 filas y 5 columnas).

1.6 Organización de la tesis

El presente documento está organizado en 5 capítulos. En el Capítulo 1 se presenta la introducción al trabajo realizado. En este mismo capítulo se hace mención de los objetivos general y particulares de la tesis y se expone la justificación de la investigación. Al final de este capítulo se indican cuáles son las contribuciones de la tesis.

El segundo capítulo del documento, Capítulo 2, versa en las generalidades del sistema inmune artificial y se presenta el estado del arte sobre micro-algoritmos.

El Capítulo 3 se explica la propuesta presentada y cómo funciona. Se hace hincapié en la manera en la que se concibió el micro-algoritmo, denominado micro-SIA, para resolver problemas de optimización numérica, y cuáles son sus características. En este mismo apartado se incluyen y comentan las pruebas y resultados experimentales de la realización en software.

En el Capítulo 4 se presentan las arquitecturas en hardware que se diseñaron e implementaron para el micro-SIA, con sus respectivos experimentos.

El último capítulo, Capítulo 5, se presentan las conclusiones derivadas de la investigación doctoral y se detallan las propuestas de trabajos a futuro que surgen a partir del desarrollo de esta tesis.

Finalmente se incluyen las referencias estudiadas y se presentan 4 anexos. El Anexo A contiene el listado de las funciones de prueba utilizadas para los experimentos del Capítulo 3, correspondientes a la optimización numérica sin restricciones. El contenido del Anexo B es el listado de las funciones de prueba utilizadas en la optimización numérica con manejo de restricciones, de los experimentos realizados en el mismo Capítulo 3.
Los anexos C y D sirven para mostrar dos trabajos que surgieron de manera paralela al desarrollo de esta tesis y que fortalecieron el conocimiento adquirido durante la investigación. El Anexo C presenta el diseño de un micro-algoritmo de Evolución Diferencial, siguiendo la misma metodología que se utilizó para el diseño del micro-SIA.
Finalmente, en el Anexo D se incluye el diseño de una arquitectura bioinspirada para una unidad de modulación por ancho de pulso (PWM), que se utiliza como una alternativa para la conversión de un dato digital a su contraparte analógica. Esta arquitectura fue abstraída de la hibridación de cadenas de ADN.

Capítulo 2

Sistema inmune artificial

El *sistema inmune artificial (SIA)* puede procesar información de manera muy significativa siendo un sistema adaptativo, distribuido, paralelo y descentralizado; algunas de sus principales características son: memoria, aprendizaje, robustez, tolerancia a fallas, diversidad y autoregulación [20], [21][22].

Para Nunes De Castro y Timmis en [21], los sistemas inmunes artificiales son sistemas adaptativos, inspirados en la teoría inmunológica, funciones, principios y modelos estudiados, los cuales son aplicados a la solución de diversos problemas. En la actualidad existen varios modelos algorítmicos del sistema inmune artificial y diversas aplicaciones de éste, por lo mismo el SIA no tiene un algoritmo habitual único (como sería el caso de un algoritmo genético). Los mismos autores sugieren en [21] que para diseñar un SIA es admisible considerar un esquema similar al que utiliza cualquier otro sistema bioinspirado, es decir, se requiere una representación predefinida de los componentes que intervienen en el sistema (simbólica, real o binaria), un conjunto de mecanismos que permitan evolucionar a éstos (midiendo la afinidad o aptitud) y un proceso que controle las dinámicas del sistema (el algoritmo).

La principal encomienda del sistema inmune en los seres vivos, es mantener al organismo libre de agentes infecciosos extraños a él, así como reparar las células dañadas o eliminarlas en el momento que sea necesario. Los microorganismos *patógenos*, que en su superficie tienen *antígenos (Ag)*, que penetran al organismo pueden resultar dañinos para éste. Los antígenos son moléculas que pueden ser reconocidas por el sistema inmune y que además

son capaces de dar inicio a la respuesta inmune para eliminarlos, propiciando una serie de procesos que neutralizarán y acabarán a los invasores.

El sistema inmune biológico tiene cuatro capas o niveles de defensa principales; cada uno de éstas antepone diferentes tipos de protección para la detección, reconocimiento y respuesta. La primera de estas capas es la *barrera física* o *anatómica*, como la piel y las mucosas. La segunda está conformada por las *condiciones fisiológicas* como la temperatura y el *ph* del órgano en que se encuentra el antígeno. La tercera capa es la *respuesta innata* (no específica) del sistema inmune, la cual no tiene memoria y permite actuar a las células macrófagas, las cuales ingieren a los antígenos para facilitar su eliminación.

Si un antígeno sobrepasa la defensa innata, automáticamente se activa el cuarto nivel de defensa del sistema inmune, conocido como *respuesta adaptativa o aprendida* (específica) y es el más estudiado para adecuar este modelo bioinspirado a un modelo computacional [22], [23]. Las células más representativas de esta respuesta son los *linfocitos B* y *T*. Los primeros maduran en la *médula ósea* y los segundos en el *timo*. Los linfocitos B secretan a su vez, otro tipo de células denominadas *anticuerpos (Ab)*. 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. La figura 2.1 muestra los cuatro niveles de defensa del sistema inmune biológico.

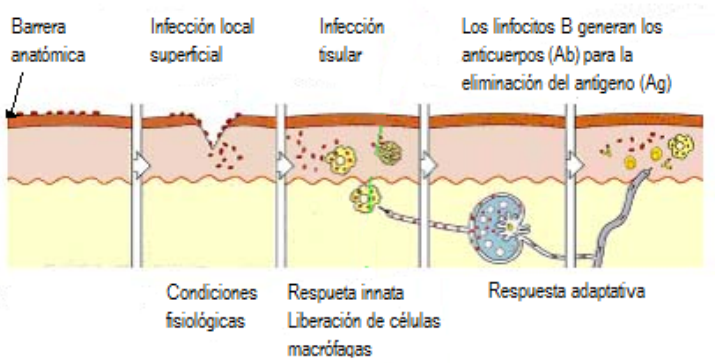


Figura 2.1. Niveles de defensa del sistema inmune biológico.

Los anticuerpos poseen en su estructura externa una región con una forma específica denominada *paratope* que se acopla con su contraparte *epitope* localizada en la estructura del antígeno, para reconocer al invasor. Cada linfocito B posee en su superficie anticuerpos con el mismo tipo de paratope y los antígenos poseen diferentes tipos de epitopes, de manera que un mismo antígeno puede activar varios linfocitos a la vez siendo reconocido por éstos. En la figura 2.2 se aprecia el acoplamiento entre un epitope y un paratope.

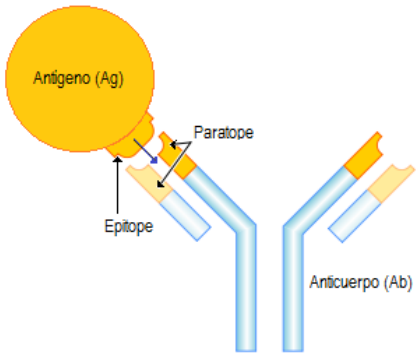


Figura 2.2. Acoplamiento del epitope de un antígeno con el paratope de un anticuerpo.

Nunes de Castro *et al* [21] y Dasgupta [22], coinciden en que debido a su gran complejidad, existen varios modelos del sistema inmune artificial que se abstraen directamente de los mecanismos de respuesta adaptativa; sin embargo, son tres los modelos más estudiados: *modelos de médula ósea y de timo*, *modelo de la selección clonal* y el *modelo de la red inmune*. Los dos últimos modelos, son los que se han utilizado mayormente para resolver problemas de optimización [23],[24], [25], [26], [27] y hacen uso del principio de *selección negativa* (también existe su complemento) que a grandes rasgos hace referencia a la propiedad de exponer a los linfocitos maduros ante un grupo de células propias del organismo, permitiéndoles sobrevivir a aquellos que no presenten ninguna reacción, y en contra sentido, se elimina a los que presenten alguna reacción.

El *modelo de médula ósea* infiere la creación y maduración de los linfocitos detectores del sistema inmune, aumentando su capacidad para poder

distinguir entre los agentes externos. El *modelo de timo* se fundamenta en el proceso mediante el cual se crea la población de linfocitos como células detectoras.

El *modelo de selección clonal* emula 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. La figura 2.3 muestra el principio de la selección clonal, en donde el *anticuerpo B* es el que tiene mayor afinidad con el antígeno, por lo que será clonado. Posteriormente, los nuevos clones sufrirán un proceso de mutación a gran escala (maduración).

Una vez que los antígenos han sido eliminados del organismo, existirá un excedente de células (anticuerpos) que deben ser exterminados para que el sistema regrese a sus niveles normales, a este proceso de le denomina *autoregulación*; sin embargo, algunas de estas células se conservan circulando a través del organismo como células de memoria con la encomienda de que la próxima vez que el mismo tipo de antígeno sea reconocido (o uno similar), el sistema inmune utilizará sus células de memoria y su respuesta será cada vez más rápida y eficiente (*respuesta secundaria*).

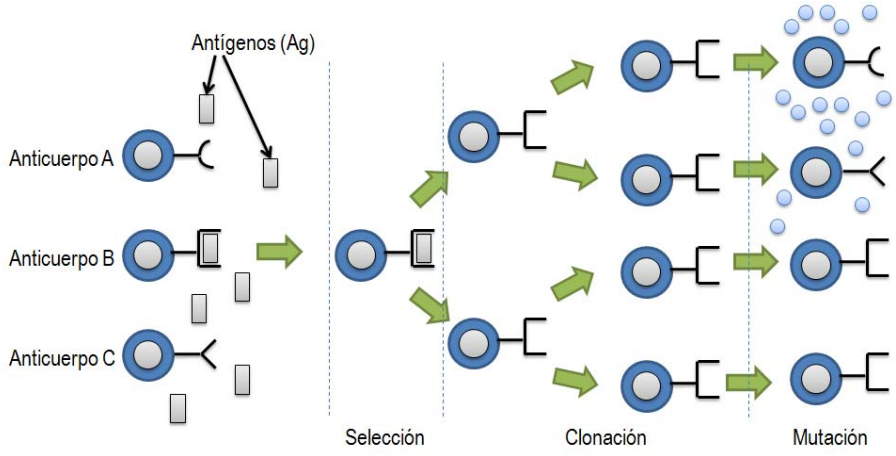


Figura 2.3. Principio de Selección Clonal.

La teoría de la *red inmune* explica las interrelaciones que existen entre anticuerpos, aún en la ausencia de antígenos, considerando que un paratope de un anticuerpo puede ser estimulado por el paratope de otro anticuerpo, llegando a suprimirse.

En la terminología empleada en el área emergente del SIA, entiéndase lo siguiente: un *antígeno* es la función objetivo del problema a resolver y los *anticuerpos* son las posibles soluciones al mismo, siendo estos últimos el equivalente a los *cromosomas* (individuos) en un algoritmo genético. La calidad de un individuo se determina evaluándolo en la función objetivo, por lo que a esta medida se le conoce como *aptitud* o *fitness* cuando se está trabajando con un algoritmo genético, en contraparte, para el SIA se le conoce como la *afinidad* entre los anticuerpos y el antígeno.

Para resolver problemas de optimización numérica y de reconocimiento de patrones, la literatura especializada reporta varios algoritmos derivados de los procesos observados en el sistema inmune biológico, sin embargo, son dos los que se consideran básicos y a partir de éstos surgen otros: *Clonal Selection Algorithm (CLONALG)* y *Artificial Immune Network (AiNet)*.

2.1 CLONALG

CLONALG está basado en el *principio de selección clonal* [24], [25] que se comentó al principio de este capítulo y cuyo funcionamiento considera la respuesta inmune ante el estímulo de un antígeno. El funcionamiento del algoritmo es el siguiente: dada una población de anticuerpos, donde cada uno de ellos tiene asociado un valor de afinidad (correspondiente al valor obtenido de la evaluación en la función objetivo, representada por el antígeno), se seleccionan para ser clonados aquellos individuos con mejor afinidad. Los anticuerpos seleccionados serán clonados proporcionalmente a su afinidad generando un repertorio de clones. Cada uno de los clones es hipermutado somáticamente (mutación a gran escala) en forma inversamente proporcional a

su afinidad. Para luego seleccionar los mejores individuos entre la población de clones y la población de anticuerpos.

Por último se reemplazan los individuos de menor afinidad por individuos generados aleatoriamente. En la figura 2.4 se lista el algoritmo CLONALG propuesto por Nunes y Von Zuben [24]. Originalmente fue diseñado para aplicaciones de reconocimiento de caracteres y posteriormente lo modificaron para que pudiera resolver problemas de optimización numérica multimodal (las funciones presentan varios mínimos locales). En [23] fue extendido para trabajar con problemas de optimización con restricciones.

1. *Inicialización*. Generar la población inicial (anticuerpos) del algoritmo, aleatoriamente.

2. *Manejo de la Población*. Para cada antígeno hacer:

   2.1 *Selección*. Seleccionar a los anticuerpos con mayor afinidad con respecto al antígeno, considerando las soluciones como anticuerpos y el antígeno como la función objetivo.

   2.2 *Clonación y Variación Genética*. Clonación de los anticuerpos estimulados en forma directamente proporcional a la afinidad: el más alto en afinidad, clonará más. Cada uno de los clones es mutado en forma inversamente proporcional a su afinidad: el más alto en afinidad mutará menos.

   2.3 *Evaluación de la afinidad*. Evaluar la afinidad de cada anticuerpo mutado con el antígeno.

   2.4 *Autorregulación*. Una vez exterminados los antígenos, el sistema inmune artificial debe regresar a sus valores normales, eliminando el exceso de anticuerpos.

3. *Ciclo.* Repetir el paso 2 hasta que se alcance el criterio de convergencia.

Figura 2.4. Algoritmo CLONALG.

En la figura 2.5 se muestra el diagrama de flujo de CLONALG. En este diagrama, C representa la población temporal de clones y C^* es la población de clones que mutó (maduró). Para obtener el número de clones de cada anticuerpo de la población ya seleccionada por ranking, se utiliza:

$$Nc = \sum_{i=1}^{n} round\left(\frac{\beta . N}{i}\right) \qquad (2.1)$$

en donde Nc es el número de clones, N es el número de anticuerpos de la población, β es un factor multiplicativo (generalmente igual a 1), y finalmente, i es la posición del anticuerpo en el ranking, siendo $i=1$ el mejor de los anticuerpos.

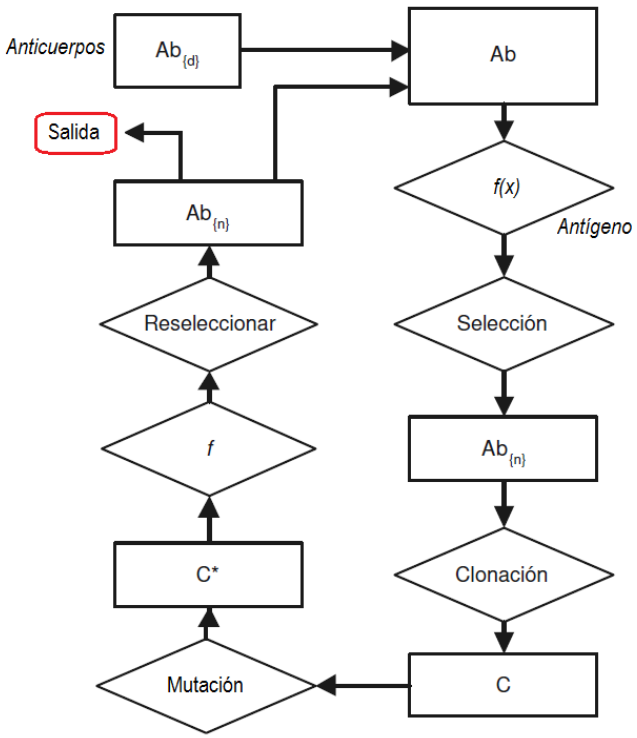


Figura 2.5. Diagrama de flujo de CLONALG.

2.2 AiNet

El algoritmo *AiNet* está basado en la teoría de la red inmune que hace mención a la capacidad que tiene un anticuerpo de un linfocito de estimular a otro anticuerpo de otro linfocito diferente, de manera similar al reconocimiento de un antígeno [26], [27]. Se dice que existe un mecanismo de retroalimentación que mantiene la memoria del sistema inmune, recordando que estas células de memoria tienden a morir si no se estimulan constantemente; el resultado es una *red idiotípica* de comunicación entre los linfocitos o células inmunes del sistema para reconocer los antígenos.

El principio fundamental para lograr la estabilidad de la red, indica que cuando un linfocito reconoce a un antígeno o a algún otro linfocito, el primero es estimulado, sin embargo, cuando un linfocito es reconocido, éste es suprimido de la red. La suma de la estimulación y supresión recibida por la red de linfocitos, más la estimulación provocada por el reconocimiento de antígenos, corresponden al nivel de estimulación total S de una célula, tal y como lo describe la ecuación 2.2.

$$S = N_{st} - N_{\sup} + A_s \qquad (2.2)$$

donde N_{st} corresponde a la estimulación de la red, N_{sup} es la supresión de la red y A_s corresponde a la estimulación antígena. En la figura 2.6 se explica la abstracción del algoritmo AiNet de acuerdo a [26].

1. *Inicialización*. Inicializar una red de células inmunes (linfocitos con sus anticuerpos) aleatoriamente.

2. *Manejo de la Población*. Para cada antígeno hacer:

2.1 *Reconocimiento Antígeno*. Relacionar la red de células con el antígeno.

2.2 *Interacciones de la Red*. Relacionar las células de la red con otras células de la misma red.

2.3 *Evaluación de la afinidad*. Introducir nuevas células dentro de la red y eliminar las menos útiles de acuerdo a algún criterio específico.

2.4 *Nivel de Estimulación*. Evaluar el nivel de estimulación de la red, considerando el conteo de resultados de los pasos previos de acuerdo a la ecuación 3.4.

2.5 *Dinámica de la Red*. Actualizar la estructura de la red y liberar los parámetros de acuerdo al nivel de estimulación de los anticuerpos individuales.

3. *Ciclo.* Repetir el paso 2 hasta que se alcance el criterio de convergencia.

Figura 2.6. Algoritmo AiNet.

2.3 Estado del Arte

Los problemas de optimización del mundo real presentan frecuentemente restricciones que generalmente resultan difíciles de satisfacer. Considerando que matemáticamente el problema de minimizar $f(\vec{x})$ es equivalente al de maximizar $-f(\vec{x})$, el problema general de la optimización numérica global se define de la siguiente manera [23]:

Encontrar $\vec{x}$ que optimice (maximice o minimice) $f(\vec{x})$ (2.3)
sujeto a:

$$g_i(\vec{x}) \leq 0, \quad i = 1,\ldots,n \tag{2.4}$$

$$h_j(\vec{x}) = 0, \quad j = 1,\ldots,p \tag{2.5}$$

donde $\vec{x}$ es el vector de r variables del problema $\vec{x} = [x_1, x_2, \ldots, x_r]^T$, n es el número de restricciones de desigualdad $g_i(\vec{x})$ y p es el número de restricciones de igualdad $h_j(\vec{x})$; en ambos casos las restricciones pueden ser lineales o no lineales. En la figura 2.7 se aprecia el espacio de búsqueda en un problema de optimización global. Una solución factible es aquella que satisface todas las restricciones o condiciones específicas del problema, es decir, si cumple las ecuaciones (2.4) y (2.5). Es mejor solución aquella que se encuentre dentro de la región factible sin importar su valor en la función objetivo. Se les denomina *restricciones activas* a aquellas que cumplen con la igualdad cuando alcanzan el valor óptimo de la función.

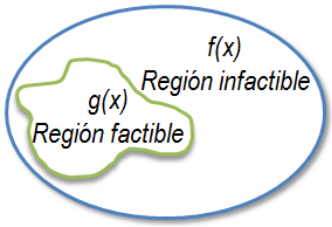


Figura 2.7. Espacio de búsqueda en un problema con restricciones.

En [5] Goldberg realizó varios experimentos utilizando un *algoritmo genético simple* (*AG*) con representación binaria. Probó una población de sólo 3 individuos, cromosomas propiamente, y afirmó que éstos eran suficientes para asegurar la convergencia del algoritmo sin importar la dimensión del cromosoma. Goldberg indicó que la mecánica consistía en aplicar los operadores genéticos hasta alcanzar una *convergencia nominal*. Esta convergencia nominal es un ciclo interno que concluye cuando los individuos son muy similares entre sí o cuando se alcanza cierto número predefinido de iteraciones. Al finalizar esta convergencia limitada, se obtiene un nuevo

individuo (el de mejor aptitud), para posteriormente generar de manera aleatoria los otros dos individuos que completarán la nueva población.

Bajo este esquema, Golberg precisó un criterio para *reinicializar* el algoritmo al concluir la convergencia nominal, manteniendo al menos al mejor individuo (elitismo) y generando ruido estocástico al completar la nueva población con individuos concebidos aleatoriamente. De esta manera se incrementa una iteración del ciclo externo del algoritmo y se utiliza la nueva población. En sus resultados, Goldberg indicó que manteniendo elitismo y con el ruido estocástico habría convergencia aunque el número de generaciones fuera muy grande. El algoritmo básico planteado por Goldberg se puede resumir como se muestra en el algoritmo de la figura 2.8.

1. *Generar aleatoriamente una micro-población*. Considerar tres individuos (cromosomas).

2. *Aptitud y elitismo.* Determinar la *aptitud* de cada individuo y el mejor se mantiene para la próxima generación (estrategia de *elitismo*).

3. *Selección.* Los padres de los restantes individuos, menos aptos, se determinan utilizando alguna estrategia de selección, por lo general, torneo binario.

4. *Analizar la convergencia de la micro-población*. Si la población converge (haciendo referencia a la convergencia nominal), regresar al paso 1, manteniendo al mejor individuo y generando aleatoriamente a los restantes. Si la población no converge, regresar al paso 2.

Figura 2.8. Micro - Algoritmo Genético propuesto por Goldberg.

Existe una población aleatoria de tres cromosomas que se copia a la población inicial. Después de haber aplicado los operadores genéticos se itera el procedimiento de evolución de acuerdo a la convergencia nominal. Si no se ha concluido la convergencia nominal, la población de trabajo siguen siendo los tres cromosomas que se están evolucionando, si ya concluyó la convergencia, el mejor cromosoma evolucionado se copia a la población inicial y ésta se completa con dos cromosomas generados aleatoriamente. En la figura 2.9 se aprecian los ciclos interno y externo del micro-AG; el ciclo interno acaba cuando se alcanza la convergencia nominal y el ciclo externo concluirá hasta que se haya alcanzado la convergencia general.

Krishnakumar en [6] diseñó un AG con una población reducida a sólo 5 individuos, a su algoritmo de representación binaria lo nombró *micro algoritmo genético (micro-AG).* Al igual que Goldberg, Krishnakumar utilizó elitismo para preservar la mejor cadena encontrada al término de la convergencia nominal, como uno de los individuos obligatorios para la siguiente generación. Al comparar el desempeño del micro-AG contra un AG simple con una población de 50 individuos, se obtuvieron mejores resultados sobre funciones de un solo objetivo, además de que se comprobó que el AG de población reducida convergía más rápido.

Dozier *et al* en [7] presentaron dos aproximaciones basadas en el micro-AG que rápidamente encuentran soluciones para problemas de optimización con restricciones. Los autores aportaron algunos mecanismos para el manejo de espacios restringidos bajo una técnica de no penalización.

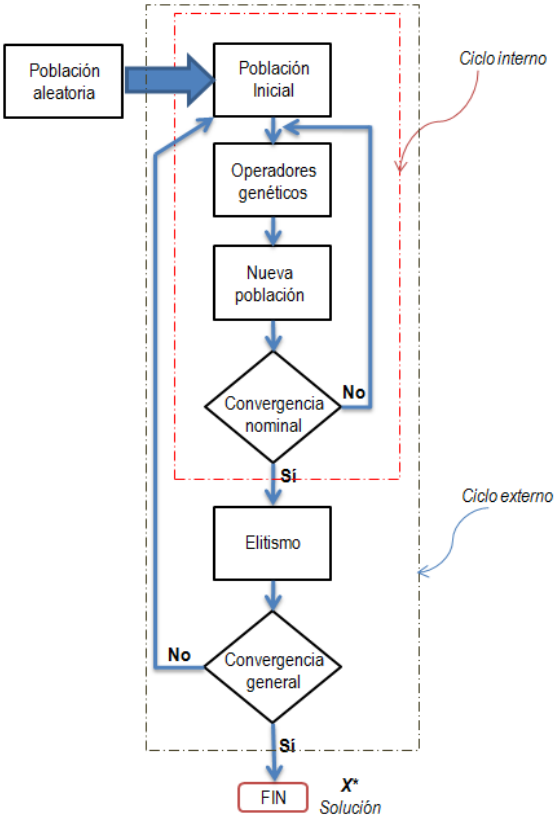


Figura 2.9. Ciclos interno y externo del Micro - Algoritmo Genético propuesto por Goldberg.

Coello y Toscano [8], diseñaron un micro AG para resolver problemas de optimización de múltiples objetivos, aportando criterios para el manejo de restricciones de igualdad y desigualdad, además de proponer un esquema de *dominancia de Pareto* con un posicionamiento geográfico para mantener la diversidad y distribuir uniformemente las soluciones del *frente de Pareto*. Este algoritmo trabaja con una población de únicamente 4 individuos (cromosomas) y utiliza una memoria secundaria que almacena las soluciones potenciales a lo largo de la búsqueda.

El funcionamiento de este micro-AG multi-objetivo comienza generando aleatoriamente la micro-población que se almacena en una memoria interna dividida en dos partes: una de ellas se mantiene sin cambio durante todo el proceso, suministrando la diversidad; la otra parte de la memoria actualiza su contenido en cada ciclo del micro-algoritmo. Con cierta probabilidad, la

población se conforma en cada ciclo tomando valores de ambas porciones de memoria y aplicando los operadores genéticos de manera convencional. Terminado un ciclo, bajo algún criterio específico, se seleccionan dos vectores no dominados de la población final, es decir, aquellos que presenten las mejores soluciones obtenidas hasta el momento y se comparan con el contenido de la memoria externa.

Al realizar la comparación, si uno de ellos o ambos siguen siendo no dominados, se incluyen en la misma memoria externa y se eliminan todos los individuos de menor aptitud o dominados. La figura 2.10, muestra una aproximación al diagrama de flujo del micro-AG multi-objetivo.

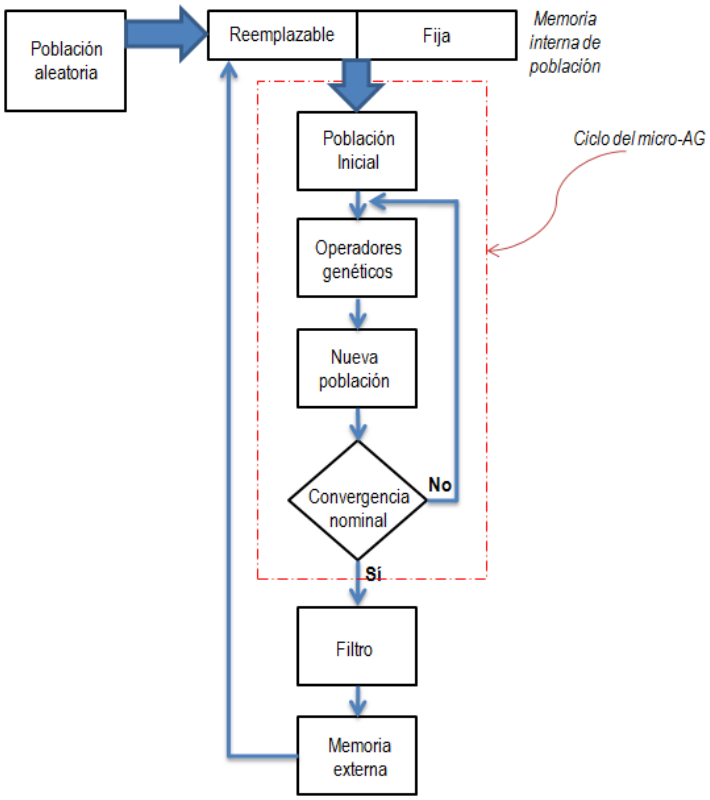


Figura 2.10. Micro-AG multi-objetivo propuesto por Toscano.

Recientemente, Fuentes y Coello en [9], diseñaron un micro algoritmo *PSO* (*Particle Swarm Optimization – Optimización por Cúmulo de Partículas*) para resolver problemas de optimización de un solo objetivo con manejo de restricciones. Utilizan 5 partículas (individuos) y se apoyan de una convergencia nominal.

En 1999 Harik *et al* [4] propusieron un *algoritmo genético compacto* (*AGc*). En vez de trabajar con la población total, como lo hace un algoritmo genético simple, el AGc únicamente simula su existencia, es decir, representa la población mediante un vector de probabilidades de valores $p_i \in [0,1] \ \ \forall i = 1,...,l$, donde l es el número de elementos del alfabeto necesarios para representar las soluciones.

Cada valor p_i indica la proporción de individuos en la población simulada que tienen un cero (uno) en la posición i de su representación. Si estos valores se toman como probabilidades, se pueden generar nuevos individuos y actualizar el vector, favoreciendo a los mejores individuos para realizar este proceso. Los valores de las probabilidades p_i, se fijan inicialmente a 0.5 para representar una población generada aleatoriamente en el que el valor de cada alelo tiene la misma probabilidad de pertenecer a una solución. En cada iteración el AGc se generan dos individuos, basándose en la representación elegida y compara los valores de sus funciones de aptitud. Se le denomina *W* a la representación del individuo con mejor aptitud y *L* a la del peor. Las representaciones de los competidores se usan para actualizar el vector de probabilidad de la iteración k a la k + 1 según la ecuación 2.6:

$$p_{i_i}^{k+1} = \begin{cases} p_i^k + 1/n & si \ \ W_i = 1 \wedge L_i = 0 \\ p_i^k - 1/n & si \ \ W_i = 0 \wedge L_i = 1 \\ p_i^k & si \ \ W_i = L_i \end{cases} \qquad (2.6)$$

siendo n la dimensión de la población simulada y $W_i(L_i)$ el valor del alelo *iésimo* de $W(L)$. El AGc finaliza cuando todos los valores del vector son iguales a 0 ó 1. En este momento, el propio vector p, representa la solución final. Para representar n individuos, el AGc actualiza el vector de probabilidad mediante un valor constante igual a 1/n. De esta forma, sólo hacen falta $\log_2 n$ bits para almacenar cada valor de p_i. Por lo tanto, el AGc sólo necesita $\log_2 n*l$ bits con respecto a los $n*l$ bits necesarios de un AG convencional. De esta manera se pueden explorar poblaciones de una dimensión mayor sin un aumento significativo de la memoria utilizada, aunque se hace más lenta la convergencia del algoritmo.

Aunque este algoritmo plantea una modificación al manejo de una población estándar, no utiliza una micro – población. Sin embargo, se ha demostrado que la implementación en plataformas hardware del AGc es realizable. Aporntewan *et al* [28] y Gallagher *et al*[29], abordan la implementación de este tipo de algoritmos en dispositivos FPGA, en donde se establece claramente el alcance de esta técnica: sustituir la población actual por un vector de dimensión l, donde l se conoce como la longitud del cromosoma, lo que reduce la cantidad de bits necesarios para almacenar una población, aunque ésta, a diferencia de los micro-algoritmos genéticos que tienen una población de máximo 5 individuos, consta de 250 individuos para sus experimentos con funciones estándares de prueba.

Por su parte, Cupertino en [30] utilizó un AGc para controlar la velocidad y la posición de un motor de inducción, que en un esquema de trabajo normal, requiere controladores independientes y que no se pueden activar al mismo tiempo, es decir, primero se tiene que modificar la velocidad y posteriormente la posición del motor; el AGc permite optimizar los tiempos de conmutación, mejorando la linealidad del sistema. Lo interesante de esta realización es que Cupertino busca reducir el coste computacional sin demeritar el desempeño de su controlador, el cual fue implementado en un dispositivo microcontrolador. Debido a las características de los algoritmos genéticos compactos, Cupertino obtiene ventaja de la subpoblación que se evalúa menos

veces en la función objetivo, pudiendo encontrar un buen desempeño comparado contra el propio del algoritmo genético normal. En este trabajo se justifica la implementación debido a la proliferación de los sistemas de control embebidos en los cuales los actuadores vienen equipados con microcontroladores de muy bajo costo.

2.4 Problemática a resolver

Considerando el estado del arte en la realización de micro-algoritmos para resolver problemas de optimización numérica es posible advertir que sólo el algoritmo genético ha sido ampliamente estudiado. El modelo del AG simple ha sido principalmente abordado por su sencillez de programación. Se tienen modelos estándares que facilitan la comparación en desempeño con los modelos de población de tamaño reducido. Los operadores genéticos de mutación, cruza y selección son estudiados utilizando métricas estadísticas, lo que repercute en una simplicidad en las pruebas experimentales.

También es posible aseverar, tras haber revisado el estado del arte, que existe un interés por realizar implementaciones en plataformas en hardware de estos algoritmos. Los principales esfuerzos en este rubro están dirigidos a lograr arquitecturas que de manera intrínseca puedan soportar la ejecución de algoritmos, como es el caso del algoritmo genético compacto, aunque como ya se indicó, éste no utiliza una población reducida sino un criterio que define un vector de probabilidades como un registro lineal de datos binarios.

Entonces, el planteamiento del problema que originó este trabajo de tesis fue el siguiente: ¿cómo implementar un algoritmo basado en el principio de selección clonal del SIA, que utilice una población reducida drásticamente en su cantidad de individuos?
Si se reduce la cantidad de individuos de dicha población, ¿será posible realizar arquitecturas en hardware que de manera embebida ejecuten este micro-algoritmo?

En el algoritmo estándar del principio de selección clonal crece el número de individuos de la población durante la fase de clonación hasta en un 600% por lo que si se consideran poblaciones estándares de 20 individuos (en la práctica pueden ser hasta 100 ó más, dependiendo del problema a resolver) se está infiriendo un número muy grande de datos que se tienen que manipular de manera simultánea. En este trabajo de tesis se propuso utilizar sólo 5 anticuerpos y mantener un crecimiento fijo de sólo 15 clones durante la fase de clonación, lo que permitió realizar menos evaluaciones a la función objetivo. Con ayuda de los operadores de mutación diseñados también fue posible acelerar la convergencia del micro-algoritmo en comparación con la versión estándar del mismo.

La micro-población facilita la migración del algoritmo para optimización numérica implementado y probado inicialmente en software, a una plataforma en hardware, pudiéndose realizar aplicaciones clásicas en el reconocimiento de patrones, como la maximización de unos en una cadena finita y el reconocimiento de caracteres.

Capítulo 3

Micro-Sistema inmune artificial

Puntualmente, el objetivo primordial de esta tesis fue diseñar e implementar un micro-sistema inmune artificial que solucione problemas de optimización. A este algoritmo se le denominó micro-SIA. Los algoritmos abstraídos de los modelos observados en la respuesta adaptativa del sistema inmune biológico son varios, no obstante el que se utilizó en esta tesis fue el del principio inmunológico de la selección clonal en donde se establece que los mejores anticuerpos, seleccionados de acuerdo a su afinidad, clonarán más y mutarán menos que los peores. Las etapas de clonación y mutación del SIA son procesos medulares en el funcionamiento del algoritmo, debido a que ante la ausencia de un operador de cruza, la proliferación de las mejores soluciones y la maduración de las mismas permiten un balance entre la exploración del espacio de búsqueda y la explotación de las soluciones más prometedoras. Considerando lo anteriormente expuesto, en este nuevo algoritmo se aportaron los mecanismos que permitieron reducir drásticamente el número de individuos en la población inicial del SIA sin afectar su desempeño.

3.1 Propuesta de solución

Realizando una revisión de los trabajos citados en el estado del arte, numeral 2.3 del capítulo anterior, es posible encontrar similitudes en el diseño de un algoritmo con un tamaño de población reducido:

1. Población inicial de 3 a 5 individuos. Ninguno de estos trabajos considera un crecimiento de la población en alguna etapa del algoritmo, como sucede con la clonación del SIA.

2. Se requiere de una convergencia nominal y de un proceso de reinicialización. La convergencia nominal se alcanza cuando se ha cumplido cierto número de iteraciones, generalmente predefinidas, funcionando como un ciclo interno en el micro-algoritmo. El proceso de reinicialización permite incorporar las mejores soluciones encontradas, una vez concluida la convergencia nominal, a la nueva población de trabajo que se ejecutará al incrementarse una iteración en el ciclo externo.

3. Es necesario considerar elitismo para preservar, al menos, al mejor individuo obtenido al término de la convergencia nominal.

4. Otro punto en común que comparten estas realizaciones es que el micro-algoritmo que se utiliza para optimización mono-objetivo puede adecuarse para manejar restricciones y para optimización con múltiples objetivos.

Considerando las similitudes anteriores, se propone un esquema general para diseñar (o adaptar en su caso) un algoritmo estándar a un modelo de micro población. Este esquema se muestra en la figura 3.1. Se puede apreciar que existen dos ciclos de funcionamiento: uno interno que se ejecutará mientras no se haya alcanzado la convergencia nominal y uno externo que se efectuará hasta que se alcance el criterio de paro general del algoritmo.

Todos los diferentes parámetros involucrados son importantes para el algoritmo, por ejemplo, Goldberg en [5] asevera que son tres los parámetros de control fundamentales de un algoritmo genético y que deben sintonizarse para garantizar su buen funcionamiento: la probabilidad de mutación, la probabilidad de cruza y el tamaño de la población inicial. En el SIA no hay operador de cruza, por lo que la mutación es uno de los parámetros más sensibles, especialmente si se reduce drásticamente el número de individuos de una población.

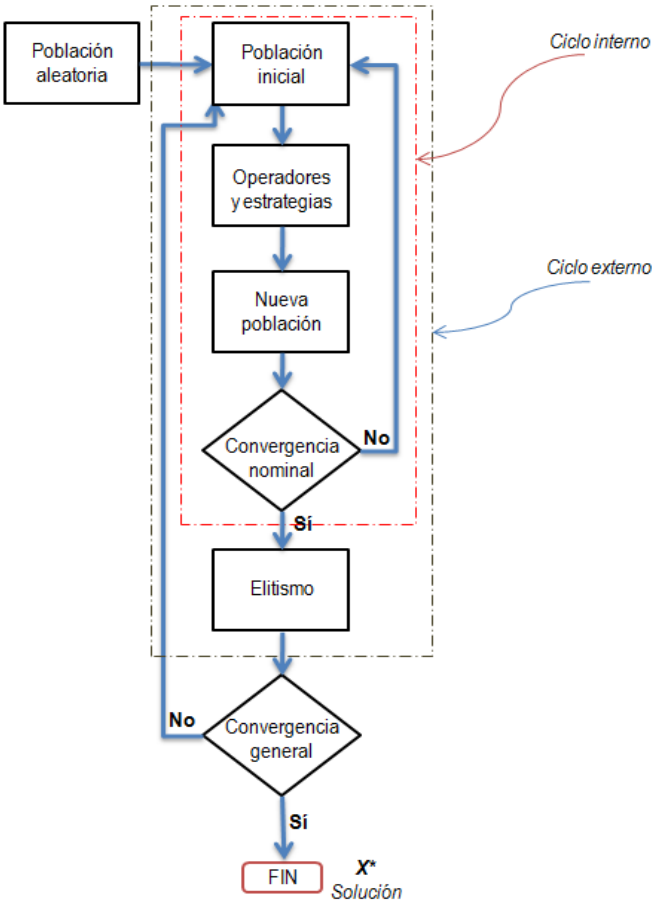


Figura 3.1. Esquema general de un micro algoritmo bioinspirado.

La metodología generalizada consta de 3 partes fundamentales:

1. *Definir el número de individuos de la población*. En dependencia a la complejidad del problema a resolver, será necesario realizar ajustes en el tamaño de la micro población. En la primera generación del algoritmo, los individuos de la población se obtienen de manera aleatoria, éstos se copian a la población inicial del ciclo interno del micro algoritmo (refiérase a la figura 3.1).

Durante el proceso de evolución, generación tras generación, la población se mantendrá con un número estático, es decir, el tamaño de la población no crece ni se disminuye de manera dinámica (en este modelo que no es particular de algún algoritmo), aunque se debe considerar la inclusión de individuos nuevos en cada

generación; éstos pudieran ser obtenidos aleatoriamente, o bien, pudieran ser tomados del conjunto de individuos evolucionados dentro del ciclo interno.

2. *Definir el criterio de la convergencia nominal y aplicar los operadores y estrategias del algoritmo bioinspirado en su versión estándar*. El criterio más utilizado para la convergencia nominal es definir un número de generaciones máximas a realizar. Al alcanzar la convergencia nominal es necesario un proceso de reinicialización para conformar una nueva población de trabajo inicial.

En el ciclo interno controlado por la convergencia nominal se utilizan los operadores y estrategias particulares definidas para el algoritmo bioinspirado en su versión estándar siendo posible implementarlo sin cambios o en otro caso realizar modificaciones que mejoren el desempeño del algoritmo con respecto a un problema en particular. Debido a que la calidad de las soluciones generadas al término de la convergencia nominal depende de los operadores y estrategias, el número de generaciones para alcanzar la convergencia nominal seguramente será diferente al cambiar de un algoritmo bioinspirado a otro.

3. *Aplicar elitismo para garantizar la convergencia*. Algunos de los individuos obtenidos al término de la convergencia nominal, generalmente los mejores o al menos el mejor en base a la aptitud, deben ser copiados a la población inicial que se completa con individuos generados aleatoriamente (para mantener la diversidad) y se incrementa una generación más en el ciclo externo hasta que se alcance la condición de paro general del algoritmo. Es significativo mencionar que depende mucho de la naturaleza de los algoritmos bioinspirados, con respecto a los operadores y estrategias internas, para determinar cuántos individuos deben ser copiados como parte del elitismo a la población inicial después, además de que éste garantiza la convergencia del micro algoritmo según lo comprobó Goldberg en [5].

3.2. Micro–Sistema Inmune Artificial para optimización sin manejo de restricciones

La propuesta presentada en este trabajo de tesis para la optimización mono-objetivo y sin manejo de restricciones se muestra en el diagrama de la figura 3.2. Está basada en el principio de selección clonal del SIA y toma algunos conceptos funcionales de CLONALG [24], aunque los operadores sensibles del algoritmo son totalmente diferentes.

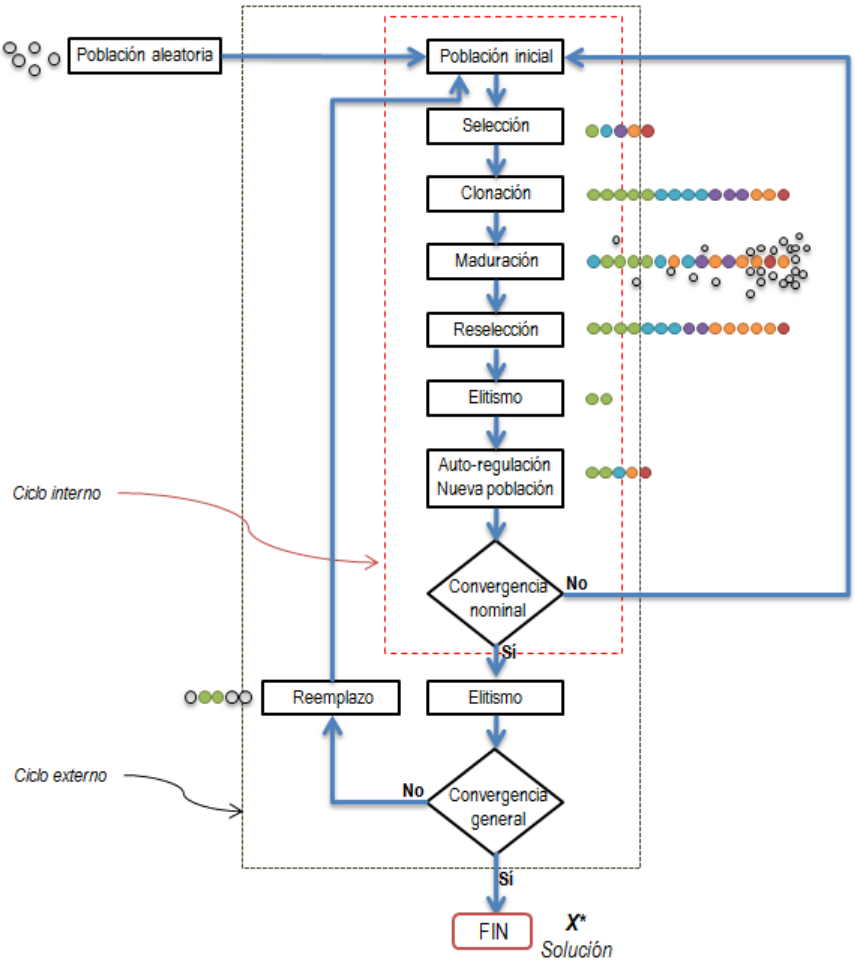


Figura 3.2. Micro – Sistema Inmune Artificial (micro-SIA) sin manejo de restricciones.

El algoritmo funciona de la siguiente manera:

1. *Generar aleatoriamente una población de 5 anticuerpos (individuos).* Al comienzo del algoritmo, para la primera generación, los 5 anticuerpos (*Ab*) generados aleatoriamente se copian directamente a la población inicial. Previamente se fijó que para alcanzar la convergencia nominal se realizaran 10 iteraciones (generaciones). Para alcanzar la convergencia general o criterio de paro del algoritmo, se utilizaron 1000 ó 2000 iteraciones (generaciones) de acuerdo a la función de prueba, tal y como se comentará en los resultados experimentales.

2. *Utilizar una selección basada en ranking.* Para ordenarlos con respecto a afinidad, cada uno de estos anticuerpos se evaluará en la función objetivo, que en la terminología del SIA es propiamente el antígeno (*Ag*). El anticuerpo de mayor afinidad será el mejor individuo, debido a que en la evaluación inicial es el que representa la mejor solución. En el micro-SIA a este individuo se le nombró *BestAb*.

3. *Realizar la clonación de todos los anticuerpos*. Para tal propósito se utiliza

$$N_C = \sum_{i=1}^{n} \left(n - (i-1)\right) \qquad (3.1)$$

en donde N_C es el número de clones que se generarán para cada anticuerpo, n es el número total de anticuerpos de la población, i es el anticuerpo corriente, comenzando con el de mayor afinidad (*BestAb*). Considerando una población de 5 anticuerpos se obtendrá una población con crecimiento controlado de 15 clones: el anticuerpo *BestAb* obtendrá 5 clones, el segundo anticuerpo del ranking clonará 4 veces, y así sucesivamente hasta llegar al peor anticuerpo que sólo obtendrá un clon.

4. *Realizar la maduración de los clones a través de un proceso de mutación*. La probabilidad de mutación se fijará al principio de la convergencia nominal para cada grupo de clones obtenidos del mismo anticuerpo. Esta probabilidad se determina de manera proporcional a la afinidad del anticuerpo que originó los clones y disminuirá en cada generación, así el grupo de clones del anticuerpo *BestAb* mutará menos que los demás grupos de clones que se generaron de los restantes anticuerpos. El único clon que se

obtuvo del peor anticuerpo tendrá la mayor posibilidad de mutar. Obsérvese la siguiente relación matemática

$$prob\_mutation(i) = \frac{Aff(i)}{\sum_{i=1}^{n} Aff(i)} \tag{3.2}$$

en donde *i* es el anticuerpo que permitirá establecer la probabilidad de mutación para el grupo de clones que se obtuvieron de él mismo y *n* es el total de anticuerpos de la población.

Para disminuir uniformemente la probabilidad de mutación en cada generación dentro de la convergencia nominal se utiliza la siguiente ecuación:

$$if\ prob\ \leq \frac{prob\_mutation(i)}{generation}\ \ then\ apply\ mutation \tag{3.3}$$

en donde $random\ prob \in [0,1]$ y *generation* es la generación corriente dentro de la convergencia nominal, es decir, $int\ generation \in [1,10]$ y no se desea dividir entre 0.

Para la variación que afecta los clones, se diseñaron dos operadores sencillos y que permiten explotar mayormente el espacio de búsqueda al realizar diferentes tamaños de paso en el proceso de mutación. Varios aspectos fueron considerados para implementar estos operadores: el número de clones, la generación corriente dentro de la convergencia nominal y el rango permisible de valores de las variables de decisión. La aplicación de estos operadores de mutación se realiza con una probabilidad de 0.5 sobre cada variable de decisión de un clon (en el micro-SIA, todo el vector de soluciones es mutado):

$$x' = x + \frac{(\alpha \cdot range \cdot generation)}{N_C} \tag{3.4}$$

o bien,

$$x^{'} = x + \frac{(\alpha \cdot range)}{(generation \cdot N_C)} \tag{3.5}$$

donde x' es la variable de decisión mutada, x es la variable de decisión a mutar, α es un número aleatorio con distribución uniforme donde $random\ \alpha \in [0,1]$, *generation* es la generación corriente dentro de la convergencia nominal y *Nc* es el número total de clones. El valor de α se debe computar para cada variable de decisión del clon.

En el caso de los 5 clones obtenidos de *BestAb*, $range \in [LB, UB]$ es un número aleatorio entre el límite inferior (LB) y el límite superior (UB) de los valores que pueden tomar las variables de decisión y se mantiene constante para toda la dimensión del clon a mutar, es decir, *range* será el mismo para todas las variables de decisión del clon. Para los restantes clones que se obtuvieron de los 4 anticuerpos del ranking, *range* es cualquier variable de decisión del anticuerpo *BestAb*, cuya posición dentro de la dimensión del vector se obtiene aleatoriamente.

5. *Efectuar nuevamente una selección basada en ranking*. Los 15 clones se ordenan utilizando ranking con respecto a su afinidad evaluando a cada uno de éstos en el antígeno (*Ag*), siendo el mejor clon el que tenga la mayor afinidad. Sí aún no se ha alcanzado la convergencia nominal, los dos mejores clones se seleccionarán (elitismo) y la nueva población se completará con otros 3 clones seleccionados aleatoriamente de la población de clones maduros. Los restantes clones se eliminarán, proveyendo una auto-regulación dentro de la convergencia nominal, manteniendo una población de trabajo de 5 anticuerpos (*Ab*). La nueva población será ahora la población inicial y el algoritmo repetirá sus pasos hasta concluir las 10 iteraciones de la convergencia nominal que a su vez representan el ciclo interno del micro-SIA.

6. *Reinicialización del algoritmo*. Si se alcanzó la convergencia nominal, se mantendrán los dos mejores clones madurados y se generarán otros tres anticuerpos de manera aleatoria (*reemplazo*) para completar la población inicial y reiniciar la convergencia nominal hasta que el ciclo externo del algoritmo cumple con la condición de parada general.

3.2.1 Primera fase de experimentación

El análisis y comparación de algoritmos bioinspirados debe realizarse considerando cuáles son las propiedades que se buscan. No es posible mejorar todas las propiedades al mismo tiempo, ya que por lo general están relacionadas entre sí y puede ocurrir que la mejora de una provoque que otra se vea afectada. De todas las posibles propiedades que pudiera presentar un algoritmo bioinspirado, generalmente se buscan las cuantificables: eficiencia, efectividad, eficacia y robustez.
La eficiencia se mide en términos de la cantidad de recursos empleados, es decir, el costo computacional. La eficacia se define como la probabilidad de alcanzar una solución óptima. La efectividad representa la calidad de las soluciones entregadas por el algoritmo. Por último, la robustez es la variabilidad del comportamiento de un algoritmo dado, es decir, qué tan estable es para resolver un problema y qué tan sensible es al momento de variar sus parámetros.

Los operadores de mutación diseñados para el micro-SIA tienen características de uniformidad y permiten pasos pequeños y grandes. Al aplicar el operador de la ecuación 3.5 la variación será mayor que cuando se aplica el operador de la ecuación 3.6. Estos operadores aceleran la convergencia y permiten una mayor exploración del espacio de búsqueda, así como una explotación de las soluciones prometedoras.

Las primeras aproximaciones probadas en el micro-SIA utilizaban el único operador de mutación no uniforme propuesto por Nunes de Castro *et al* en [24], sin embargo para el algoritmo con población reducida no entregó buenos resultados debido a que no converge al óptimo aún aumentando el número de individuos hasta 10.

Para la primera experimentación y con respecto al micro-SIA sin manejo de restricciones, se utilizaron las funciones citadas en [32]. Se trata de funciones con múltiples variables (dimensión 30) y óptimo en cero, a excepción de *f08*. Se recomienda referirse al anexo A para revisar estas funciones. La elección de este test de prueba se debió a que las funciones abordadas en [32] se resuelven utilizando el algoritmo de evolución diferencial lo que presupone que las funciones son complejas y a la vez ejemplifican un referente actual de experimentación.

En la tabla 3.1 se reportan los resultados obtenidos experimentalmente para esta primera fase. Se listan los resultados de 20 corridas para cada función. Para todos los casos se utilizó una población fija de 5 anticuerpos y 10 generaciones para alcanzar la convergencia nominal, así como una representación real. Para realizar estas pruebas se utilizó una PC con procesador *Quad Core* a 2.66 MHz, con 2MB de memoria. La intención de este experimento fue probar estadísticamente la convergencia del micro-SIA al óptimo conocido.

Tabla 3.1. Resultados experimentales del micro-SIA para problemas de optimización sin restricciones.

Función	*Ciclo externo*	*Generaciones de la convergencia nominal*	*Óptimo conocido*	*Mejor solución Micro-SIA*	*Peor solución Micro-SIA*	*Media Micro-SIA*
f01	1000	10	0.0	0.0	0.000022	0.000009
f02	1000	10	0.0	0.0	0.000017	0.000008
f03	1000	10	0.0	0.0	0.000002	0.000001
f04	1000	10	0.0	0.0	0.000012	0.000005
f05	1000	10	0.0	0.0	0.000028	0.000012
f06	2000	10	0.0	0.0	0.000032	0.000015
f07	2000	10	0.0	0.0	0.000027	0.000013
f08	2000	10	-12596.5	-12569.5	-12569.57	-12569.496
f09	2000	10	0.0	0.0	0.000033	0.000013
f10	2000	10	0.0	0.0	0.000011	0.000007
f11	2000	10	0.0	0.0	0.000013	0.000004

En la tabla 3.1, la columna *Ciclo externo* se refiere al número de generaciones que se fijaron para alcanzar la convergencia general del micro-SIA, que a su vez es la condición de paro del mismo. La columna *Generaciones de la convergencia nominal* presenta el número de generaciones del ciclo interno del micro-SIA que se fijaron para alcanzar esta convergencia limitada. La columna *Óptimo conocido* es la mejor solución documentada en [32]. Se observa que en todas las funciones utilizadas se convergió al óptimo. La medida del error promedio es mínima en todos los casos. Obsérvese que para las funciones *f06*, *f07*, *f08*, *f09*, *f10* y *f11* fue necesario aumentar el número de generaciones en el ciclo externo para converger al óptimo.

En la figura 3.3 se estima el comportamiento del micro-SIA resolviendo solamente *f01* en 5 corridas diferentes; se hizo de esta manera para facilitar la legibilidad de la gráfica. En el eje de las abscisas se asentaron las 1000 generaciones del ciclo externo.

Recordando que el óptimo conocido es 0.0, se puede apreciar cómo el algoritmo al principio realiza pasos grandes y posteriormente se realiza una búsqueda más exhaustiva con pasos más reducidos. El comportamiento reflejado es el esperado considerando que los operadores de mutación se diseñaron para cumplir con este propósito. En esta tesis se maneja el término *aptitud* y *afinidad* de manera equivalente para las figuras gráficas, recordando que en la terminología del SIA propiamente debe utilizarse *afinidad*.

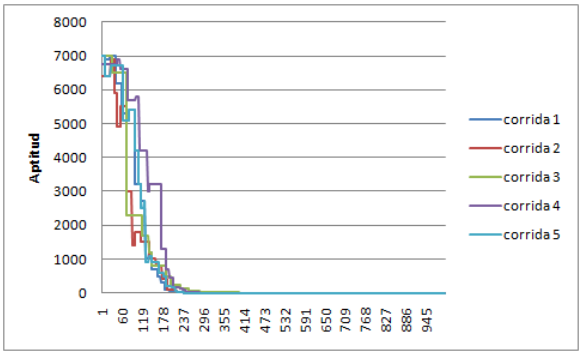


Figura 3.3. Comportamiento del micro-SIA con 5 corridas diferentes para f01.

En la figura 3.4 se detalla cómo el micro-SIA se aproxima al óptimo de *f01* en las últimas 100 generaciones de una corrida. Al paso de las generaciones, la probabilidad de mutación disminuye de acuerdo a la ecuación 3.2, por lo que en la figura 3.4 se aprecia que la aptitud (se trata de un problema de minimización para el caso de *f01*) tiende hacia el óptimo particular (0.0) y ya no efectúa cambios abruptos que obliguen al micro-SIA a explorar nuevamente el espacio de búsqueda, más bien, se explotan las mejores soluciones al acercarse a la condición de paro del algoritmo. El comportamiento anterior es muy conveniente para escapar de los óptimos locales.

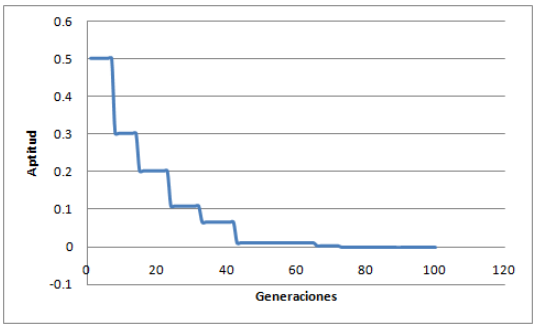


Figura 3.4. Detalle del funcionamiento del micro-SIA en las últimas 100 generaciones de una corrida para resolver f01.

Con respecto al comportamiento del micro-SIA durante las 10 generaciones establecidas para alcanzar la convergencia nominal, en la figura 3.5 se muestran 5 corridas diferentes para resolver *f01*. Esta figura sólo hace referencia a la primera convergencia nominal del algoritmo. Se observan las variaciones grandes debido a que la probabilidad de mutación es alta, de acuerdo a la ecuación 3.2, cuando inicia el algoritmo.

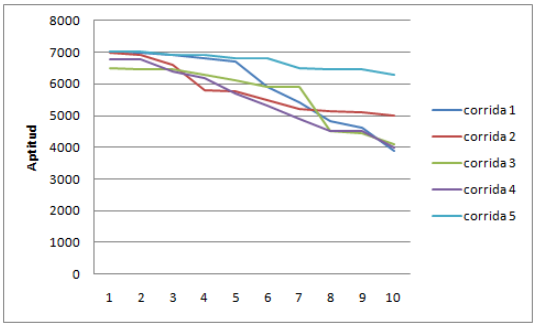
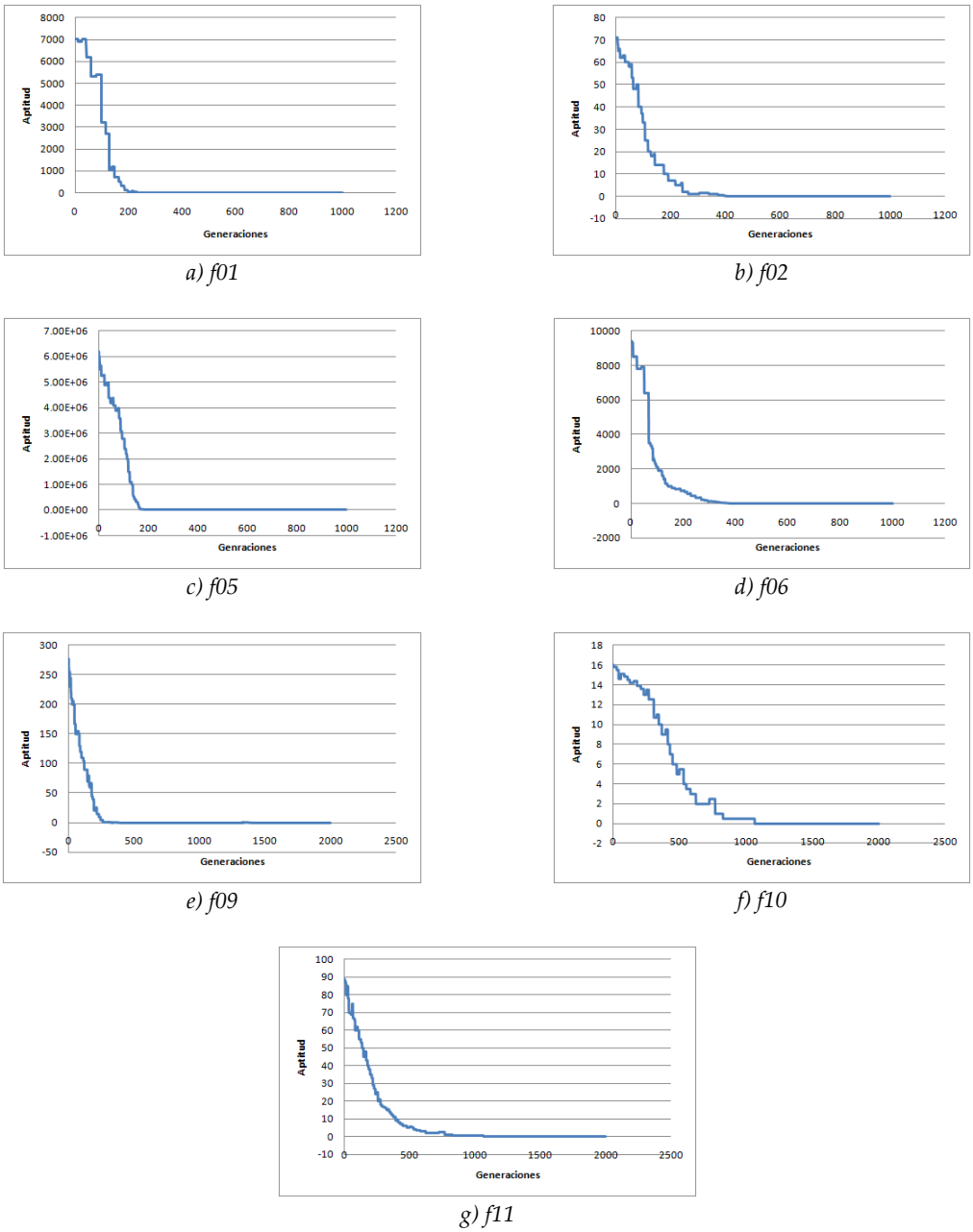


Figura 3.5. Primera convergencia nominal del micro-SIA con 5 corridas diferentes para resolver f01.

En las 5 corridas presentadas en la gráfica, se observa que no existe la posibilidad de un error de convergencia prematura, dado que en las pruebas realizadas se obtuvieron en todos los casos individuos diferentes. Este apartado es muy importante debido a que se aprecia que se mantiene la diversidad de los individuos y no se propicia un estancamiento de la búsqueda. Aún obteniendo individuos idénticos o muy parecidos al término de la convergencia nominal, el ruido estocástico generado antes de conformar la nueva población ayuda al algoritmo a escapar de una convergencia prematura. Una diversidad muy amplia, es decir, que los individuos varíen su

información en gran medida, también representa una problemática que obligaría a la búsqueda a quedarse en un óptimo local. Con este experimento se observa que la diversidad propiciada en el micro-SIA mantiene un balance correcto entre la exploración y la explotación. En la figura 3.6 se muestran los diferentes comportamientos del micro-SIA para resolver f01, f02, f05, f06, f09, f10 y f11, en una sola corrida.



a) f01 *b) f02*

c) f05 *d) f06*

e) f09 *f) f10*

g) f11

Figura 3.6. Comportamiento del micro-SIA para resolver f01, f02, f05, f06, f09, f10 y f11.

3.2.2 Segunda fase de experimentación

Para la segunda fase de los experimentos, se procedió a realizar una comparación entre el SIA en su versión estándar (CLONALG) y el micro-SIA. Sólo se utilizaron tres funciones del test de prueba del numeral 3.2.1. Para esta comparación se ejecutaron ambos algoritmos en la misma PC de los experimentos anteriores, ésta tiene un procesador *Quad Core* a 2.66 MHz y 2MB de memoria. La intención de este experimento fue revisar el número de evaluaciones a la función objetivo y el tiempo de ejecución del algoritmo para converger al óptimo. En la tabla 3.2 se listan los resultados experimentales obtenidos en esta fase.

Tabla 3.2. Resultados experimentales de la comparación entre CLONALG y el micro-SIA, ambos casos sin manejo de restricciones.

	Ab	*Clones*	*Generaciones de la convergencia nominal*	*Ciclo externo*	*Evaluaciones a la función objetivo*	*Tiempo (segundos)*
CLONALG						
f01	50	256	N/A	1000	1,280,000	47.2
f05	70	312	N/A	1000	21,840,000	78.6
f07	70	312	N/A	1200	26,208,000	103.7
Micro-SIA						
f01	5	15	10	1000	750,000	14.8
f05	5	15	10	1000	750,000	14.2
f07	5	15	10	2000	1,500,000	48.3

Nótese en la tabla 3.2, que el número de anticuerpos (individuos) en la población inicial del algoritmo en su versión estándar para *f01* es de 50, mientras que para el micro-SIA en la misma función se utilizan sólo 5 individuos. Así mismo, en la etapa de clonación, para *f01* en la versión estándar la población crece hasta 256 clones, mientras que en el micro-SIA se tiene un crecimiento controlado de 15 clones. Con lo anterior, el uso de la memoria de datos en el micro-SIA se reduce al 5.85% de la utilizada por la versión estándar de CLONALG. Para el caso de *f05* y *f07*, el micro-SIA utiliza el 4.80% de los 312 clones generados en la etapa de clonación.

Con respecto al número de evaluaciones a la función objetivo, comparando los resultados, es evidente que el micro-SIA realiza menos evaluaciones para alcanzar el óptimo. Nótese que si se aumenta el ciclo externo del micro-algoritmo también crece el número de evaluaciones a la función objetivo. Considerando *f07*, el micro-SIA ejecuta 1, 500,000 evaluaciones a la función objetivo y en contraparte CLONALG verifica 26, 208,000 evaluaciones, lo que representa un menor coste computacional para la versión con micro-población. En este mismo sentido, el tiempo de ejecución del micro-SIA es también menor en tres casos reportados.

3.2.3 Tercera fase de experimentación

Esta etapa experimental de la tesis consistió en modificar el número de anticuerpos (*Ab*) en la población inicial. De acuerdo a la ecuación 3.1, si se varía el número de individuos iniciales, implícitamente se modificará también el número de clones a generar. El objetivo de este experimento fue observar el comportamiento del micro-SIA ante diferentes tamaños de micro-población.

Se utilizaron sólo las tres funciones del test de prueba del numeral 3.2.2: *f01*, *f05* y *f07*. Para reportar estos resultados se mantuvo en 10 el número de generaciones para alcanzar la convergencia nominal; para alcanzar la convergencia general, en el ciclo externo del micro-SIA, se utilizaron 1000 generaciones para f01 y f05, así como 2000 para f07, tal y como se hizo en la primera etapa de experimentación. Se decidió fijar estos valores para las generaciones debido a que en los experimentos anteriores se obtuvieron los mejores resultados cuando éstos se utilizaron.

En la tabla 3.3 se aprecian los resultados obtenidos al finalizar este experimento. Se realizaron 20 corridas del micro-SIA y se reportan solamente los mejores y los peores resultados para las funciones citadas.

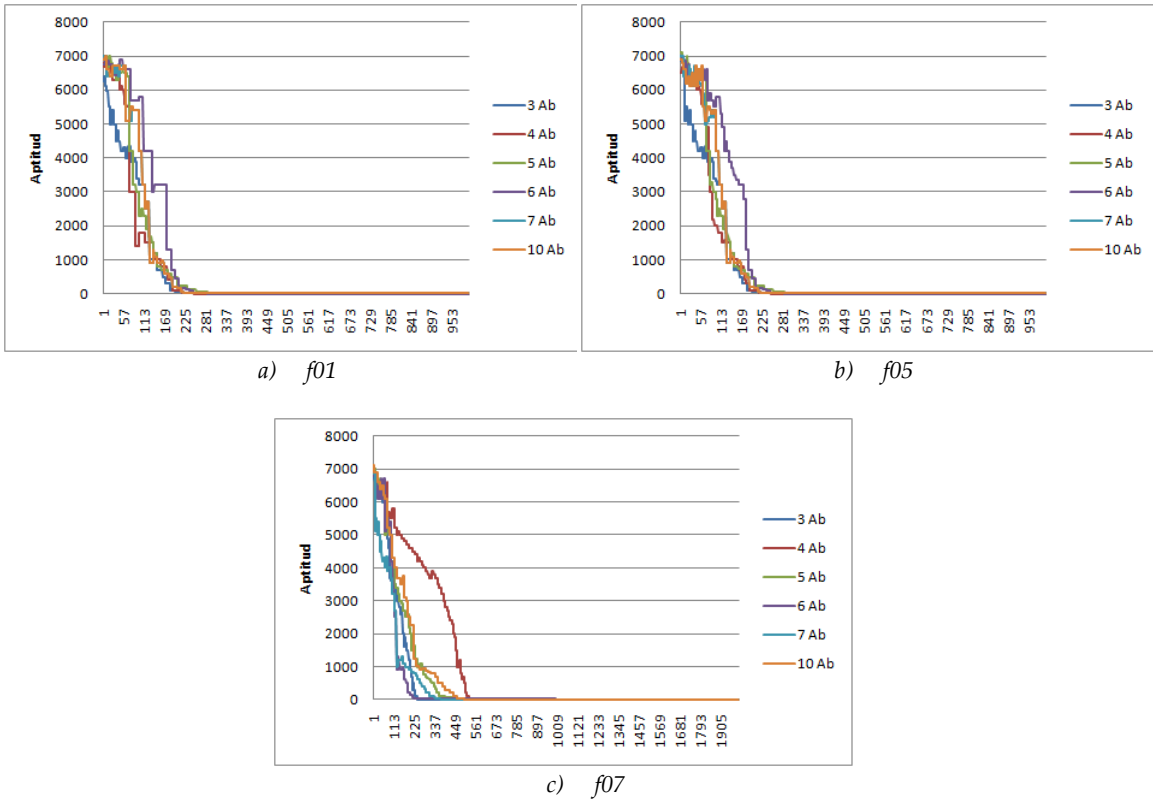
Tabla 3.3. Resultados experimentales al variar el tamaño de la micro-población del micro-SIA.

Ab	*Clones*	*Mejor* ***f01***	*Peor* ***f01***	*Media* ***f01***	*Mejor* ***f05***	*Peor* ***f05***	*Media* ***f05***	*Mejor* ***f07***	*Peor* ***f07***	*Media* ***f07***
3	6	1.782333	24.301121	4.001221	1.478011	4.899233	2.345671	0.834078	2.665235	2.006686
4	10	0.0	0.0830000	0.0456682	0.000012	0.932121	0.0202242	0.000044	0.000421	0.000256
5	15	0.0	0.0000022	0.0000008	0.0	0.000028	0.0000011	0.0	0.000027	0.0000010
6	21	0.006789	1.004601	0.0780502	0.000333	1.004902	0.000567	0.003021	2.301202	0.009786
7	28	0.820006	8.023331	3.029821	0.670012	4.023210	2.147681	1.003415	2.550015	1.700494
10	55	2.893011	30.900189	14.221124	2.453312	18.788210	10.331881	1.899301	23.871131	9.111244

Se distingue que con una población de 5 anticuerpos y 15 clones, se convergió al óptimo en los tres casos. Las peores soluciones en las tres funciones utilizadas no son muy diferentes en valor cuando se comparan con las mejores, considerando la micro-población de 5 anticuerpos. Cuando se utilizan 4 ó 6 anticuerpos se obtienen buenas soluciones, siendo mejor utilizar 4 anticuerpos en vez de 6, por lo menos para las funciones utilizadas en este experimento. La media obtenida de las 20 corridas para cada función muestra que las mejores soluciones se obtuvieron con 5, 4 y 6 anticuerpos, respectivamente. Cuando se creció la población inicial a 7 y 10 anticuerpos, el micro-SIA no obtiene buenas soluciones. Cuando la población inicial es de sólo 3 anticuerpos, tampoco se convergió al óptimo conocido, sin embargo las soluciones encontradas no son tan malas como las que se lograron con 7 y 10 anticuerpos.

Es notorio que cuando se trabaja con muy pocos anticuerpos, durante el ciclo interno del micro-SIA la población tienen muy poca variabilidad entre individuos, por lo que la búsqueda se detiene antes de llegar al óptimo. Los clones generados bajo este esquema son muy pocos, y aún aplicándoles la mutación, no se logra un funcionamiento adecuado del algoritmo. En contraparte, cuando se utilizan más individuos en la población de trabajo, propiciando un aumento considerable en la población de clones, en la convergencia nominal se obtienen soluciones muy variadas y la búsqueda se estanca en la parte final del algoritmo. Esto se debe principalmente a que aún conservando a las mejores soluciones a través del elitismo, la mutabilidad no permite tener elementos representativos que guíen la búsqueda.

En la figura 3.7 se presentan las tres gráficas correspondientes a este experimento. Se muestran las corridas con 3, 4, 5, 6, 7 y 10 anticuerpos para f01, f05 y f07.



a) f01 *b) f05*

c) f07

Figura 3.7. Comportamiento del micro-SIA ante diferentes tamaños de micro-población.

3.3 Micro–Sistema Inmune Artificial para optimización con manejo de restricciones

Debido a que los problemas de optimización en la vida real contienen restricciones de diferente índole, en este último experimento se decidió incluir una aproximación al manejo de restricciones para saber cómo se comporta el micro-SIA ante esta situación. Para que el algoritmo cumpliera con esta

encomienda, fue necesario incorporar un mecanismo que permitió el manejo de restricciones de igualdad, desigualdad, lineales y no lineales.

Existen mecanismos estudiados y probados, principalmente en algoritmos genéticos y estrategias evolutivas, que soportan el manejo de restricciones [31]. El más simple de estos mecanismos es el denominado *pena de muerte* que consiste en rechazar a todos los individuos infactibles, es decir, se le asignará una aptitud de cero a un individuo que no sea factible y se asignará el valor de la función de aptitud al que sí lo sea. Esta técnica es muy sencilla de implementar, no obstante la búsqueda se estanca cuando las regiones factibles del problema a optimizar son muy pequeñas o si la población inicial no tiene ningún individuo factible.

Para el manejo de espacios restringidos en el micro-SIA se utilizaron dos parámetros para seleccionar a los mejores individuos: la afinidad con respecto a la función objetivo y la cantidad de restricciones violadas. Esta metodología fue propuesta por Deb en [31]. La adecuación para el micro–algoritmo aquí propuesto sólo considera en la etapa del primer ranking (refiérase al numeral 3.2), en el ciclo de la convergencia nominal, seleccionar al mejor individuo bajo las siguientes características: como primera opción se buscará al anticuerpo factible sobre el infactible; si fuera el caso que hubiera varios factibles, seleccionar al de la mayor afinidad. Para los infactibles, se seleccionará al que tenga el menor número de restricciones violadas y mayor afinidad.

Una vez realizada la selección inicial de anticuerpos (refiérase al paso 2 del algoritmo explicado en el subtema anterior), utilizamos los mismos criterios para la segunda selección después de haber realizado la mutación de los clones (refiérase al paso 5 del mismo algoritmo).

$$fitness_i(x) = \begin{cases} f_i(x) & \text{si la solución } es \text{ factible} \\ f_{peor} + \sum_{j=1}^{n} g_j(x) & \text{de lo contrario} \end{cases} \tag{3.6}$$

donde f_{peor} es el valor de la función objetivo de la peor solución factible de la población. Si no hay ninguna solución factible en la población, entonces f_{peor} se hace igual a cero.

3.3.1 Cuarta fase de experimentación

Se probaron dos funciones: *g01* y *g02*, presentadas en [33]. En el anexo B de este documento de tesis es posible revisar estas funciones. En la tabla 3.4 se resumen las características de las dos funciones de prueba citadas. La función *g01* tiene 13 variables y la función *g02* tiene 20 variables. El óptimo global para *g01* es -15.0. Para *g02* el mejor óptimo global encontrado (el óptimo es desconocido) es −0.803619. En esta misma tabla, *n* es el número de variables de decisión, ρ el porcentaje de individuos factibles encontrados entre un millón de individuos generados aleatoriamente, *DL* es el número de desigualdades lineales, *DNL* es el número de desigualdades no lineales, *IL* es el número de igualdades lineales y, por último, la etiqueta *INL* representa el número de igualdades no lineales.

Tabla 3.4. Características de las dos funciones de prueba con manejo de restricciones.

Función	*n*	*Tipo de función*	*Óptimo conocido*	ρ	*DL*	*DNL*	*IL*	*INL*
g01	13	Cuadrática	-15.0	0.0111%	9	0	0	0
g02	20	No lineal	−0.803619	99.8477%	1	1	0	0

Revisando ρ en esta tabla, la función *g01* presenta una zona factible muy pequeña, es decir, encontrar individuos que cumplan todas las restricciones es complicado para este problema de minimización. Por el contrario, *g02* tiene una zona factible muy amplia, sin embargo llegar al óptimo conocido resulta difícil debido a que este problema de maximización tiene un gran número de óptimos locales.

En la tabla 3.5 se muestran los resultados de 20 corridas para cada función, utilizando el micro-SIA.

Tabla 3.5. Resultados experimentales del micro-SIA para problemas de optimización con restricciones.

Función	*Ciclo Externo*	*Generaciones de la convergencia nominal*	*Óptimo conocido*	*Mejor micro-SIA*	*Peor micro-SIA*	*Media micro-SIA*	*Error (considerando el mejor resultado)*
g01	2000	10	-15.0	-15.0	-14.978	-14.989	0.0
g02	2000	10	-0.803619	-0.804090	-0.83023	-0.81025	0.000471

El algoritmo diseñado tiene un desempeño satisfactorio para ambos casos. Para el caso particular de *g01*, el micro-SIA sí obtiene el óptimo global conocido. Con respecto a *g02*, el micro-SIA se aproxima en gran medida al óptimo conocido aunque prevalece un error mínimo. En las figuras 3.8 y 3.9 se muestran las gráficas de convergencia al óptimo global del micro-SIA para *g01* y *g02*, respectivamente.

Ambas gráficas comienzan cuando se encuentra el primer individuo factible. Puede apreciarse que la convergencia en ambos casos es rápida. El micro-SIA logró escapar correctamente de los óptimos locales que tiene *g02*, lo anterior se debe a que los operadores de mutación diseñados, así como el elitismo utilizado, permiten un balance correcto entre la exploración del espacio de búsqueda y la explotación de las soluciones potenciales.

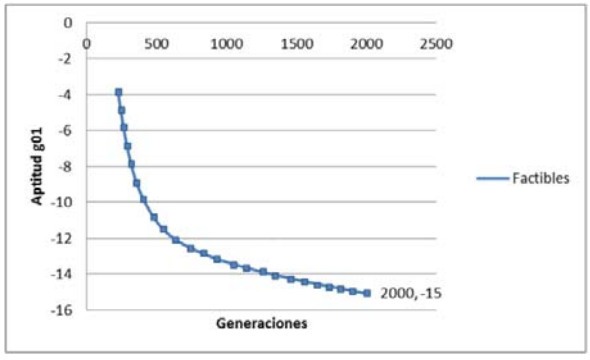


Figura 3.8. Convergencia al óptimo global del micro-SIA para g01.

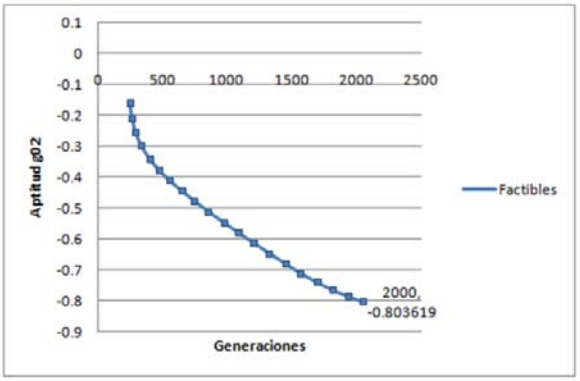


Figura 3.9. Convergencia al óptimo global del micro-SIA para g02.

En las figuras 3.10 y 3.11 se presentan las aproximaciones a la densidad de individuos factibles e infactibles encontrados por el micro-SIA, para *g01* y *g02*, al paso de las generaciones. Estas gráficas se diseñaron para reportar los individuos encontrados cada 50 generaciones.

Es visible que en el caso de *g01* son pocos los individuos factibles encontrados, sin embargo, a partir de que se encuentra el primero de éstos, la proliferación de buenas soluciones se mantiene, demostrando que el criterio utilizado para el manejo de restricciones funciona adecuadamente. Como se esperaba, de acuerdo a las características expuestas en la tabla 3.4, *g02* tiene una zona factible amplia por lo que desde el principio fue posible encontrar individuos que cumplen todas las restricciones.

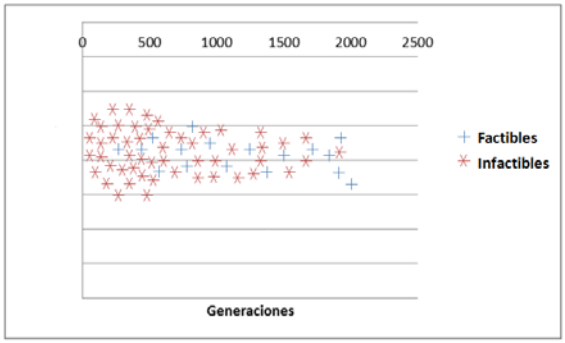


Figura 3.10. Individuos factibles e infactibles encontrados por el micro-SIA para g01.

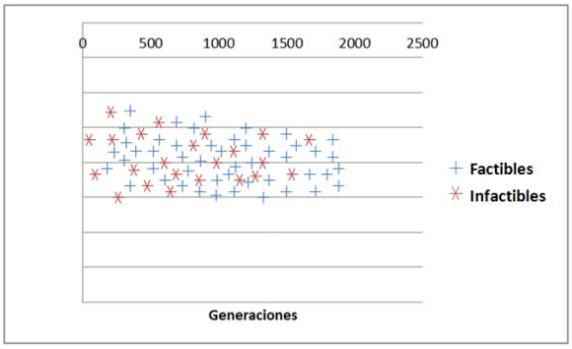


Figura 3.11. Individuos factibles e infactibles encontrados por el micro-SIA para g02.

Capítulo 4

Arquitectura del micro-SIA

En el capítulo 3 se demostró que el micro-SIA puede resolver problemas de optimización numérica global. Aunque al reducir el tamaño de la población resulta obvio que también se reduce el uso de la memoria de datos, resulta interesante analizar cómo se comporta el micro-SIA en una plataforma hardware. En primera instancia, el manejo de pocos individuos parece facilitar la implementación del algoritmo de manera intrínseca en un dispositivo electrónico digital. En esta tesis se dirigieron los esfuerzos hacia dos tipos de dispositivos: un microcontrolador comercial de 8 bits y un dispositivo de lógica reconfigurable (FPGA). El microcontrolador tiene recursos limitados; por otra parte, el FPGA tiene una arquitectura más robusta que la del microcontrolador.

4.1 Micro-SIA en hardware

Para estas realizaciones en hardware se propuso resolver el problema de maximización de unos (*maxone problem*) [34], que consiste en encontrar el número máximo de bits puestos a uno (nivel lógico alto) en una cadena finita. Este problema es clásico en el área del reconocimiento de patrones. Matemáticamente el problema se describe como encontrar un vector $\vec{x} = \{x_1, x_2, \ldots, x_n\}$ en donde $x \in \{0,1\}$ que maximice la ecuación

$$F(\vec{x}) = \sum_{i=1}^{N} x_i \qquad (4.1)$$

En esta ecuación, *N* representa el número de bits de la cadena, por lo que el máximo se obtendrá cuando los *N* bits de la cadena sean unos. Este problema no es sencillo de resolver en hardware, debido a que un número mayor que otro, no necesariamente representa un mejor resultado, por ejemplo, el dato binario 1000 es peor solución que el dato 0011 debido a que su cadena de bits tiene menos unos. A medida que aumenta el número de bits de la cadena, la complejidad de la búsqueda también aumenta tal y como se comenta en [34].

La adecuación general del micro-SIA para las realizaciones en hardware funciona de la siguiente forma:

1. *Generar aleatoriamente 5 anticuerpos de N bits*. En la generación inicial estos anticuerpos se copian directamente a la población inicial para comenzar la convergencia nominal que está controlada por un número de generaciones igual a 5. Obsérvese que a diferencia del micro-SIA utilizado para optimización numérica, ahora se utilizan 5 generaciones en vez de 10, para alcanzar la convergencia nominal. La representación de los individuos ahora es binaria y no real, como sucedió en el capítulo 3 de este trabajo.

2. *Utilizar una selección basada en ranking*. Tal y como se hizo con el micro-SIA para optimización numérica. El anticuerpo de mayor afinidad será el mejor individuo.

3. *Realizar la clonación de todos los anticuerpos*. Para tal propósito se utiliza la ecuación 3.1 del capítulo 3 de esta misma tesis. De igual manera a la versión de optimización numérica, considerando una población de 5 anticuerpos se obtendrá una población de 15 clones.

4. *Realizar la maduración de los clones a través de un proceso de mutación considerando que se manejan cadenas binarias*. La probabilidad de mutación se fijará al principio de la convergencia nominal para cada grupo de clones obtenidos del mismo anticuerpo. Esta probabilidad se determina de manera proporcional a la afinidad del anticuerpo que originó los clones y disminuirá en cada generación, así el grupo de clones del mejor anticuerpo, nombrado *BestAb* en la versión en software, mutará menos que los demás grupos de clones que se generaron de los restantes anticuerpos.

El único clon que se obtuvo del peor anticuerpo tendrá la mayor posibilidad de mutar. Para tal propósito se aplican las mismas ecuaciones 3.2 y 3.3 del capítulo 3. Con respecto a la variación que presenta cada uno de los clones al efectuar la mutación, se utilizó un operador sencillo que invierte un bit en determinada posición de la cadena. Para determinar qué bit debe mutarse se aplica una probabilidad de 0.5 considerando que un bit de una cadena que representa un individuo que es una buena solución tendrá menos posibilidades de cambiar que uno que pertenece a una mala solución.

5. *Efectuar nuevamente una selección basada en ranking*. Esta ocasión, se ordenarán los 15 clones con respecto a su afinidad. Los dos mejores clones se seleccionarán (elitismo) y la nueva población se completará con otros 3 clones seleccionados aleatoriamente de la población de clones maduros. Los restantes clones se eliminarán, proveyendo una auto-regulación dentro de la convergencia nominal.

6. *Aplicar elitismo*. Si se alcanzan las 10 generaciones de la convergencia nominal, se mantendrán los dos mejores clones y se generarán otros tres anticuerpos de manera aleatoria para completar la nueva población y comienza nuevamente la convergencia nominal hasta que el ciclo externo del algoritmo cumpla con la condición de parada.

4.2 Implementación en el microcontrolador

Se decidió dirigir la implementación del diseño hacia un dispositivo comercial, en este caso un microcontrolador de 8 bits de bajo coste. Se utilizó un *PIC16F628A* CUYA arquitectura obliga a que se realice un procesamiento procedural de las instrucciones, además se tiene un número de pines de E/S restringido a dos puertos de 8 bits, así como una memoria de datos que es de sólo 224 bytes. Para este caso, las etapas del diseño incluyen:

1. *Generador de números aleatorios*. Se utiliza el oscilador interno del microcontrolador (4MHz) y un *push button* para accionar el módulo generador que como elemento base tiene un contador binario. Este módulo genera 5 cadenas binarias (anticuerpos) para copiarlos a la población inicial.

Existen diversos métodos para generar números aleatorios. Un generador de verdaderos números aleatorios (*RRNG*) en hardware utiliza características no predecibles de elementos físicos (radiación de un núcleo atómico, inestabilidad de un oscilador libre, ruido de una resistencia eléctrica, resonancia de las partículas de un rayo láser, entre otros). Los generadores de números pseudoaleatorios (*PRNG*) son más comunes en hardware debido principalmente, en la mayoría de los casos, a su facilidad de implementación. Los números pseudoaleatorios se generan a partir de una función determinista pero aparentan ser aleatorios. Estos generadores tienen el inconveniente de trabajar con sucesiones periódicas y que a partir de un mismo valor inicial (semilla) se genera siempre la misma sucesión.

En esta tesis se utilizó un *registro de desplazamiento con realimentación lineal* (*LSFR*) como generador de números pseudoaleatorios. Este circuito sirvió tanto para el microcontrolador como para el FPGA, variando la longitud del registro y la función de retroalimentación. Un circuito LFSR genera secuencias pseudoaleatorias con un período garantizado arbitrariamente largo, de forma simple y eficiente, siempre y cuando se inicialicen con valores diferentes de cero. El registro de desplazamiento se implementa con elementos de memoria que comúnmente son *flip-flops tipo D*. La señal común que produce la transferencia de datos entre los flip-flops, está sincronizada con la señal de reloj (CLK) del circuito. Al realizar una combinación de la salida de dos registros y asignándola a la entrada de otro registro, se obtiene un circuito LFSR. Esta realimentación se realiza utilizando compuertas XOR.

Dado que el estado interno de un registro de desplazamiento tiene L bits, un LFSR puede generar secuencias pseudo-aleatorias de tamaño máximo $2^L - 1$ (no 2^L, porque un registro de desplazamiento que se llena con ceros, provoca que la secuencia de salida del LFSR sea una secuencia infinita de ceros). Un LFSR tendrá longitud máxima si su polinomio característico (polinomio de realimentación) es primitivo en $GF(2^n)$, es decir, si se encuentra este polinomio (pudieran existir varios) es posible

determinar cuántas operaciones XOR se requieren para la retroalimentación y en qué etapas se deben conectar para cumplir con todas las combinaciones binarias de 2^n bits sin repetirlas. Existen tablas que indican los polinomios característicos para realizar la conexión de las compuertas XOR, para diferentes tamaños de registros.

El esquema más habitual de un circuito LFSR de 8 bits, se muestra en la figura 4.1. Es el mismo esquema que se empleó en esta tesis para generar los números pseudoaleatorios en el microcontrolador.

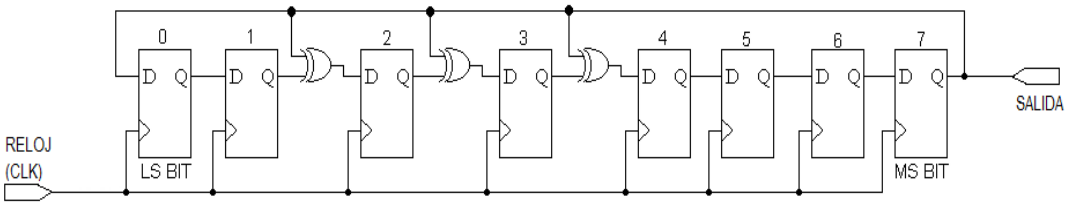


Figura 4.1. LFSR de 8 bits, utilizado para la generación de números pseudoaleatorios.

2. *Módulo de ordenamiento (ranking)*. Este módulo cuenta los unos de cada cadena y asigna una calificación a cada individuo. Posteriormente los ordena de mejor a peor considerando que los mejores individuos son los que tienen el mayor número de unos. Para realizar el ordenamiento se utilizó el conocido método de la burbuja, cuyo pseudocódigo se muestra en la figura 4.2. En éste, *LIMITE* es el número de datos a ordenar.

```
for (i=1; i<LIMITE; i++)
    for j=0 ; j<LIMITE - 1; j++)
          if (vector[j] > vector[j+1])
               temp = vector[j];
               vector[j] = vector[j+1];
               vector[j+1] = temp;
```

Figura 4.2. Pseudocódigo del ordenamiento burbuja.

Para contar los unos de una cadena se utiliza un procedimiento iterativo en donde se recorre un registro de datos (palabra) de un extremo hasta el otro y se incrementa un contador cada vez que se encuentra un *"1"*. El esquema simple utilizado en esta tesis consta de un registro de desplazamiento, que con ayuda de una compuerta XOR y un contador, permite detectar la presencia de un uno y así incrementar el conteo. En el micro-SIA adecuado para hardware, esta tarea es el equivalente a la evaluación de la función objetivo. El mismo número de unos contado es el valor en calificación que se le asigna a cada individuo, equivalente a la aptitud o *afinidad* del SIA. Para ordenarlos de acuerdo a su afinidad, se utiliza un comparador que detecta quién o quiénes tienen el mayor número de unos y los clasifica como los mejores individuos.

3. *Módulo de clonación*. Este módulo se encarga de generar 5 clones del mejor individuo, 4 del segundo mejor y así sucesivamente hasta obtener los 15 clones que caracterizan a nuestro algoritmo.

4. *Módulo de mutación*. Cada uno de los 15 clones tiene probabilidad de mutar, sin embargo, los 5 clones obtenidos del mejor individuo tienen menos probabilidad de mutar que el único clon obtenido del peor individuo, quien tiene la más alta probabilidad de mutar. Toda la cadena de bits de cada clon se recorre completamente para aplicar el operador de mutación en donde corresponda.

Para los 5 clones del mejor anticuerpo se utiliza mutación uniforme, es decir, se invertirá un solo bit de la cadena. La posición de este bit se elige aleatoriamente. Para los 4 clones del segundo mejor anticuerpo se aplica mutación en dos puntos, es decir, se invertirán dos bits en dos posiciones elegidas aleatoriamente. Sucesivamente se utiliza este criterio hasta llegar al peor anticuerpo que sólo obtuvo un clon, a éste se le aplicará mutación en 5 puntos, lo que indica que se invertirán 5 bits de su cadena cuyas posiciones en ésta también se eligen aleatoriamente. En la figura 4.3 se muestra cómo funciona la mutación para el mejor y el peor clon generado.

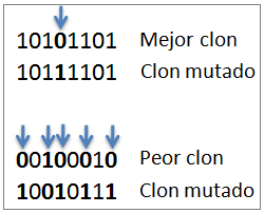


Figura 4.3. Mutación aplicada a los clones.

5. *Módulo de ordenamiento de clones*. Este módulo funciona de manera similar al primer módulo de ordenamiento para 5 anticuerpos, con la observación de que ahora debe clasificar 15 clones. Tras haber mutado, los individuos en el *ranking* han cambiado su calidad.

6. *Módulo de remplazo*. Este módulo se encarga de mantener a los dos mejores clones de los 15 generados y completa la población de trabajo con otros tres que se generan de manera aleatoria. Este módulo hace uso del módulo generador de números aleatorios para cumplir con su encomienda. Este mismo módulo lleva el control de la convergencia nominal por lo que al alcanzar las 5 generaciones del ciclo interno, mantendrá a los dos mejores individuos, que a su vez son las dos mejores soluciones, y los copia a la población inicial para comenzar nuevamente un ciclo externo del micro-SIA.

En la figura 4.4 se pueden observar los módulos concebidos para la implementación embebida, que complementan la explicación anterior. Las etapas del micro-SIA implementado en hardware son las mismas que se presentaron en la figura 3.2 del capítulo 3 de esta tesis.

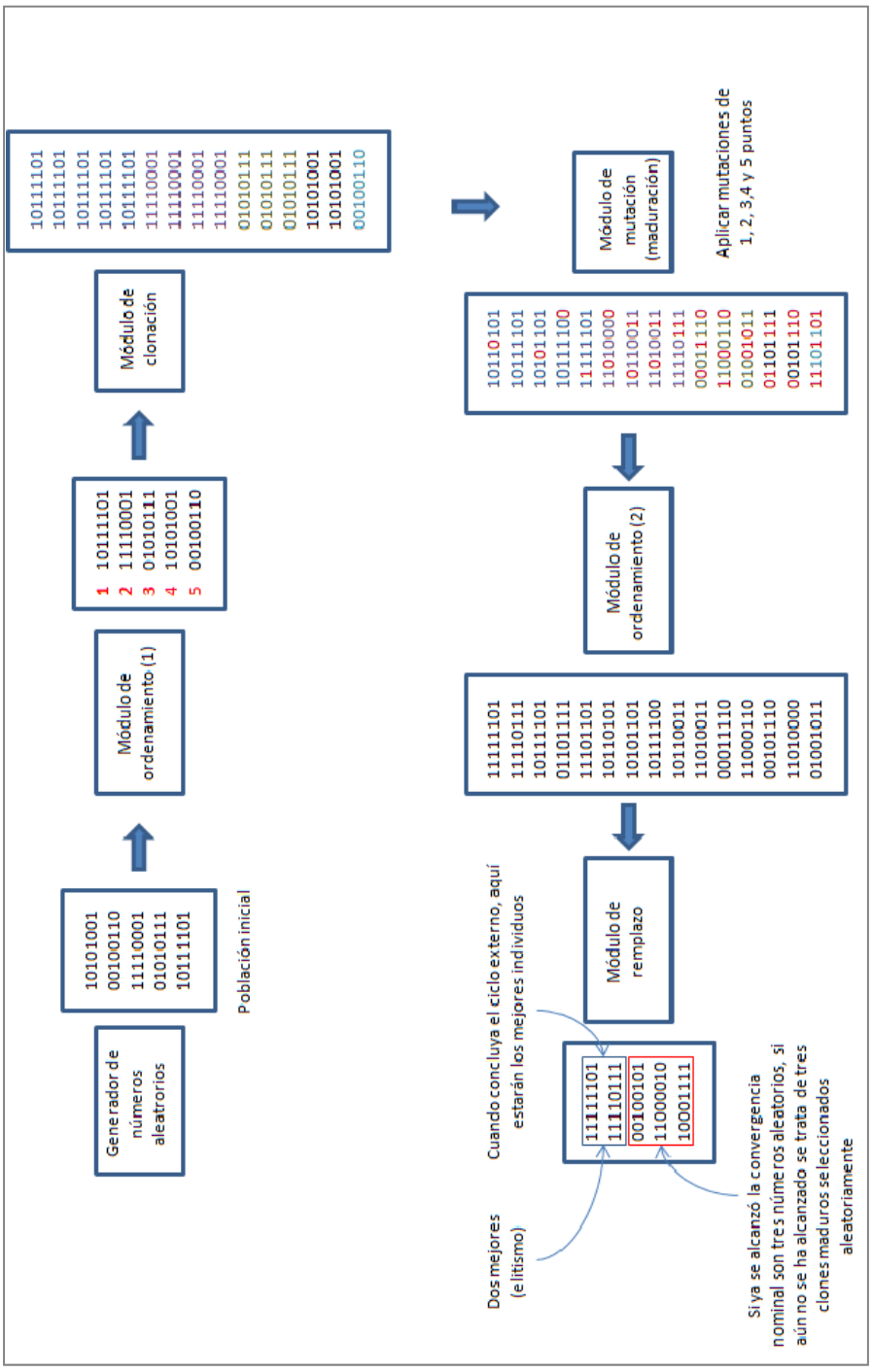


Figura 4.4. Diagrama a bloques del micro-SIA implementado en hardware, para solucionar el problema de la maximización de unos en una cadena de 8 bits.

4.2.1 Experimentación con el microcontrolador

Se realizó una interfaz que permitió comunicar el microcontrolador con una PC, vía el puerto serial (RS-232). Esta interfaz sólo se utilizó para monitorear (leer) los datos entregados por el microcontrolador, dado que el micro-SIA se ejecuta intrínsecamente. El circuito de prueba que se construyó se muestra en el diagrama de la figura 4.5. El *push-button* permite iniciar la ejecución del algoritmo. La terminal en la PC recibe los datos iniciales y finales del algoritmo; estos datos son: los 5 primeros individuos generados aleatoriamente, el número de generaciones (ciclo externo del micro-SIA) de la convergencia y la solución obtenida.

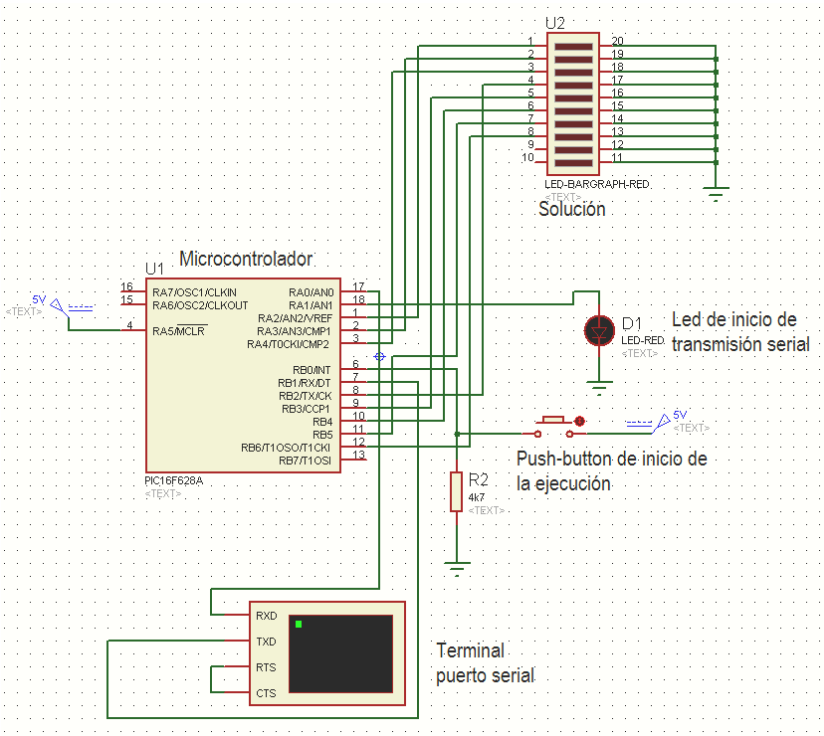


Figura 4.5. Implementación del micro-SIA en el microcontrolador PIC16F628, para resolver el problema de maximización de unos.

En la tabla 4.1 se reportan los 5 anticuerpos que el microcontrolador generó para la población inicial, en 5 corridas diferentes. En esta misma tabla se listan las

generaciones necesarias para lograr el máximo número de unos en la cadena de 8 bits, así como el tiempo de convergencia. En este primer experimento se buscó el número máximo de generaciones para converger al óptimo (11111111, con afinidad 8) y el tiempo utilizado por el microcontrolador. La frecuencia de trabajo del PIC16F628A fue de 4MHz correspondiente al oscilador interno del dispositivo.

Se observa que en todos los casos se convergió al óptimo, lo cual demuestra en forma práctica que los operadores de mutación y de elitismo se seleccionaron adecuadamente para la micro-población y el crecimiento controlado a sólo 15 clones. El tiempo de ejecución de la corrida 5 fue el menor y el de la corrida 1 resultó el más alto. Si se revisan los anticuerpos generados en la población inicial de la corrida 5, se puede advertir que desde el principio se obtuvo un muy buen candidato con afinidad 7, el cual guió la búsqueda y aceleró la convergencia.

Tabla 4.1. Población inicial del micro-SIA implementado en el microcontrolador, generaciones y tiempo de convergencia.

Corrida	*Población inicial*	*Afinidad inicial*	*Generaciones necesarias*	*Mejor encontrado*	*Tiempo (segundos)*
1	10101011 11101001 00110101 11001000 00010010	5 5 4 3 2	1346	11111111	238
2	00110000 00010010 10111101 11001111 10011010	2 2 6 6 4	1102	11111111	194
3	10000011 10010010 00001001 10000001 00101111	3 3 2 2 5	1210	11111111	214
4	11101001 10111101 11000001 11000011 00111110	5 6 3 4 5	1322	11111111	234
5	11111011 01100110 00000001 11101101 10110110	7 4 1 6 5	746	11111111	132

El siguiente experimento realizado con el microcontrolador, consistió en mantener un número fijo de generaciones en el ciclo externo del algoritmo y observar cuál fue el mejor resultado obtenido al alcanzar el criterio de paro general. Para este experimento se realizaron 10 corridas del micro-SIA con un número de generaciones preestablecido y seleccionado considerando los resultados reportados en la tabla 4.1.

En la tabla 4.2 se reportan los resultados obtenidos en este experimento. Entre 1000 y 1500 generaciones se convergió al óptimo. Cuando se utilizaron pocas generaciones (100 y 500) no se obtuvieron buenas soluciones, no obstante, es notorio que el micro-SIA comienza a maximizar (en la particularidad de este problema) desde que inicia su ejecución, dado que hay un crecimiento estable de la afinidad al paso de las generaciones.

Tabla 4.2. Resultados experimentales del micro-SIA implementado en el microcontrolador con diferente número de generaciones.

Generaciones	*Mejor encontrado*	*Afinidad del mejor encontrado*	*Peor encontrado*	*Afinidad del peor encontrado*	*Media de la afinidad*
100	01101001	4	10000001	2	3
500	11001111	6	01001110	4	5.25
750	01111111	7	10110111	6	6.5
1000	11111111	8	11110111	7	7.8
1500	11111111	8	11111111	8	8

4.3 Implementación en el FPGA

El micro-SIA para el problema de maximización de unos, se implementó en un FPGA *Spartan 3A*, con número de parte *3S700A* del fabricante *Xilinx*. Los módulos diseñados en el numeral 4.2 para el microcontrolador son los mismos que se utilizan para el FPGA, con la observación de que en este último dispositivo es posible generar una arquitectura más robusta.

Los módulos fueron realizados en VHDL que es uno de los lenguajes descriptores de hardware más utilizados actualmente. Para este diseño se consideraron cadenas binarias de 16 bits, por lo que aplica el diagrama a bloques de la figura 4.4., asumiendo registros de datos de 16 bits. En la figura 4.6 se muestra una aproximación al código en VHDL que implementa el LFSR que genera los números pseudoaleatorios de 16 bits.

```
ARCHITECTURE arch_lfsr  OF lfsr_pseudo IS
 SIGNAL   q     : std_logic_vector(15 DOWNTO  0);
 SIGNAL   reset : std_logic;
 CONSTANT semilla  : std_logic_vector(15 DOWNTO  0):= (OTHERS  => '1');
BEGIN
 lfsr : PROCESS(clk,reset)
 BEGIN
  IF(reset='0') THEN
   q <= semilla;                        -- Inicio
  ELSIF (clk'EVENT AND  clk='1') THEN
   q(0) <= q(15);                       -- Realimentación
   q(1) <= q(0);
   q(2) <= q(1) XOR q(15);              -- Etapa 1
   q(3) <= q(2) XOR q(15);              -- Etapa 2
   q(4) <= q(3) XOR q(15);              -- Etapa 3
   q(15 DOWNTO 5)<= q(14 DOWNTO 4);
  END IF;

 END PROCESS lfsr;
END ARCHITECTURE arch_lfsr;
```

Figura 4.6. Arquitectura en VHDL para un LFSR de 16 bits.

En la figura 4.7 se presenta la arquitectura para el micro-SIA implementada en el FPGA. Obsérvese que las entidades diseñadas corresponden a las explicadas en el numeral 4.2. La síntesis lógica se realizó utilizando *ISE Webpack 12.4i* de *xilinx*. El paso de los individuos entre cada entidad se realiza de manera paralela.

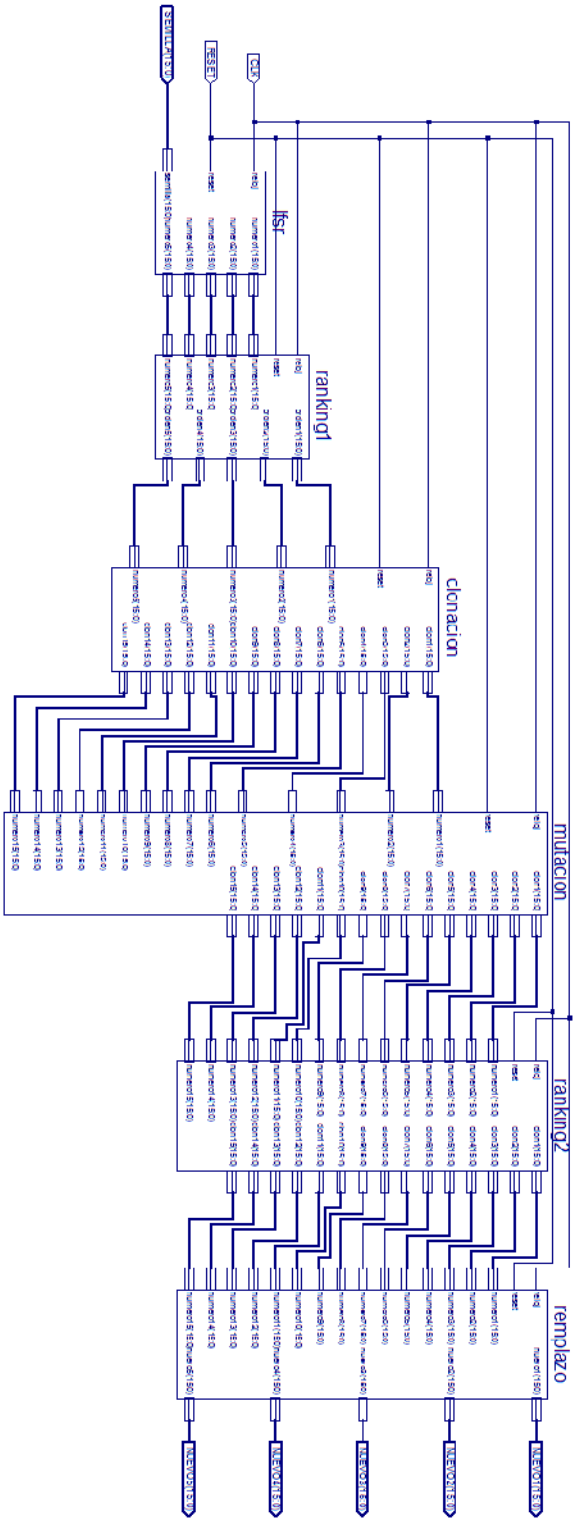


Figura 4.7. Arquitectura del micro-SIA en el FPGA.

4.3.1 Experimentación con el FPGA

El FPGA *3S700A* de Xilinx es un dispositivo con tecnología SRAM y 700,000 compuertas lógicas. La síntesis lógica del diagrama de la figura 4.7 utilizó el 32% de los recursos del FPGA y con el controlador para la comunicación con el puerto serial de la PC, designado para el monitoreo de datos, se utilizó el 37% de los recursos. De igual forma que en el caso del microcontrolador, se predispuso el diseño de una interfaz de comunicación serial (RS-232) sólo para leer los datos entregados por el FPGA. Del lado de la PC se habilitó una terminal que se sincronizó a 9600 baudios con el FPGA.

La frecuencia de trabajo de la arquitectura bioinspirada en el FPGA, es la misma frecuencia del oscilador incorporado en la tarjeta de desarrollo, es decir 50MHz.

El primer experimento con el micro-SIA en el FPGA consistió en revisar la probabilidad de convergencia a través del número de generaciones. Para tal propósito se realizaron 5 corridas del micro-SIA buscando el óptimo de la maximización de unos para una cadena de 16 bits.

En la tabla 4.3 se reportan los resultados de este experimento. En la última columna de esta misma tabla se anotó el tiempo que le tomó al FPGA resolver el problema. A diferencia de la tabla 4.1, referente al experimento equivalente con el microcontrolador, en la tabla 4.3 no se anotó la mejor solución obtenida, esto se hizo arbitrariamente debido a que todas las corridas convergieron al óptimo, que en este caso fue *1111111111111111*, es decir, con afinidad 16.

Este ejercicio considera 65535 combinaciones a razón de 2^{16}, por lo que no es un ejercicio sencillo para el micro-SIA en el FPGA. En la tabla 4.3 se aprecia que en la corrida 3, el micro-SIA convergió más rápidamente al óptimo, sin embargo, en la corrida 1 se utilizó el mayor número de generaciones y por ende el mayor tiempo de convergencia. Al igual que como se precisó en la primera experimentación con el microcontrolador, cuando se obtienen soluciones potenciales en la población inicial, la convergencia es más rápida.

Tabla 4.3. Resultados experimentales del micro-SIA implementado en el FPGA: generaciones y tiempo de convergencia.

Corrida	*Población inicial*	*Afinidad inicial*	*Generaciones necesarias*	*Tiempo (segundos)*
1	0111000000000001 1010100100110101 1001101000110101 1000100010100001 0001111001111101	4 8 8 5 10	55226	135
2	1100111101110101 0101001010101100 0000110010010000 0101000000010000 0001000000001111	11 7 4 3 5	10014	25
3	1001111110111101 0001001011111101 0001100110001100 1111000111111101 0110011110000000	12 9 6 12 6	7230	18
4	0100100100110101 0000001000001000 1100100000000001 1100100000111111 0010011100111110	7 2 4 9 9	12378	31
5	0011111011111011 1011111111011111 0000000110000000 0110101010101001 1010110010010011	7 14 2 8 8	17783	45

Para observar el efecto de las generaciones en el ciclo externo, se decidió realizar un segundo experimento. Cabe mencionar que tanto en el microcontrolador como en el FPGA, no se modificaron las características de la convergencia nominal, manteniendo 5 generaciones para alcanzarla; esto se debe a que por tratarse de parámetros sensibles, la variación de ambos números de generaciones repercutiría en resultados que no sería sencillo interpretar.

El micro-SIA para optimización numérica global que se diseñó en el capítulo 3, utiliza 10 generaciones para alcanzar la convergencia nominal; en contraparte, el micro-SIA en hardware sólo utiliza 5 generaciones. Se decidió fijar este número de generaciones debido a que experimentalmente se obtuvieron excelentes resultados.

En la tabla 4.4., se reportan los resultados obtenidos para este experimento. El micro-SIA se ejecutó 10 veces para cada número de generaciones preestablecidas, como se indica en la primera columna de la tabla.

Tabla 4.4. Resultados experimentales del micro-SIA implementado en el FPGA con diferente número de generaciones.

Generaciones	*Mejor encontrado*	*Afinidad del mejor encontrado*	*Peor encontrado*	*Afinidad del peor encontrado*	*Media de la afinidad*
100	0001001110010001	6	0001000000010001	3	3.75
500	0111000101010011	8	1000001000010100	4	4.5
1000	1111000011001100	8	0011001110010001	6	6.4
5000	1010011111100010	9	0011110011000000	6	7.75
10000	1111110011001111	12	0000110000111111	8	10.5
15000	1011111111111101	14	1101010111101000	9	12.8
20000	1111111111111111	16	1011111100111101	12	14.25

En los resultados de la tabla 4.4., se observa que entre las 15000 y 20000 generaciones, el micro-SIA convergió al óptimo. Existe un proceso de maximización gradual al paso de las generaciones, similar a lo que se observó en el ejercicio equivalente con el microcontrolador. Se tiene un comportamiento lineal que acelera la búsqueda en las últimas generaciones del algoritmo, tal y como se aprecia cuando se revisa la media de la corrida con 20000 generaciones en el ciclo externo del micro-SIA.

4.3.2 Aplicación para el reconocimiento de caracteres

El SIA ha sido aplicado con éxito dentro del área del reconocimiento de patrones y específicamente en el reconocimiento de caracteres. Por tal motivo se decidió realizar una aplicación que permitió la ejecución intrínseca del micro-SIA en un FPGA con el propósito de reconocer 14 caracteres predefinidos: 1, *2*, *3*, *4*, *5*, *6*, *7*, *8*, *9*, *A*, *E*, *I*, *O*, *U*. Nótese que no se consideró el caracter correspondiente al número cero (0), para no confundirse con el caracter de la letra *O*.

Se utilizó una matriz de 7 filas y 5 columnas (35 puntos) para dibujar cada caracter. Se predispuso esta configuración dado que en el mercado nacional de componentes electrónicos es común el uso de matrices de LEDs que concuerdan con estas características.

La matriz es fácilmente codificable en una cadena de 35 bits de la forma $Ag =< Ag_1, Ag_2, ..., Ag_L >$, en donde Ag representa los antígenos o caracteres preestablecidos, y en contraparte, para el caracter a reconocer se codifica de la forma $Ab =< Ab_1, Ab_2, ..., Ab_L >$, en donde Ab representa un anticuerpo. En la figura 4.8 se muestra un ejemplo de codificación para el caracter E.

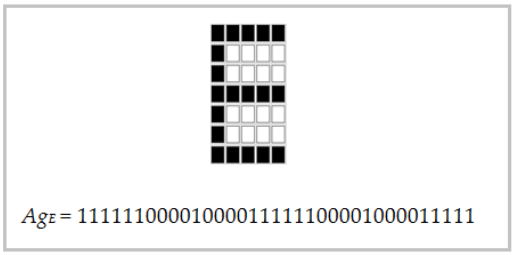


Figura 4.8. Ejemplo de codificación para la matriz de 35 puntos.

La aplicación diseñada conforma un sistema simple para el reconocimiento de los caracteres antes indicados. De inicio, se introduce un carácter corrupto que puede tener grandes variaciones con respecto a los caracteres originales, con ruido aditivo o sustractivo generado de manera arbitraria. Utilizando una distancia de *Hamming*, implementada con compuertas XOR, es sencillo comparar el caracter corrupto con cada uno de los caracteres predefinidos y asignar una medida de afinidad para cada una de estas comparaciones. En este sentido, este procedimiento de comparación es equivalente a la evaluación de la función objetivo.

En la figura 4.9 se ejemplifica el procedimiento para determinar la afinidad de un anticuerpo o caracter corrupto, es decir, cómo se le califica para saber qué tan bueno es el individuo como solución. Nótese que la comparación entre las cadenas de 35 bits se realiza bit por bit con una compuerta XOR, por lo que cuando los bits sean iguales se obtendrá un 0 lógico y cuando sean diferentes se obtendrá un 1 lógico. Se trata de un problema de minimización, dado que el óptimo de una comparación se obtendrá

cuando la cadena del resultado de comparación obtenga sus 35 bits ($C_{34},\ldots,C_0$) puestos a 0 lógico, que determina que ambos caracteres son idénticos.

El *módulo de conteo de unos* se utiliza para asignar el valor numérico de la afinidad, por ejemplo, la cadena del resultado de comparación (considerando los 35 bits) 00000111110000011111000001111100000 tiene una afinidad igual a 15; la cadena 00000000000000011111000000000000000 tiene una afinidad de 5 y sería una mejor solución.

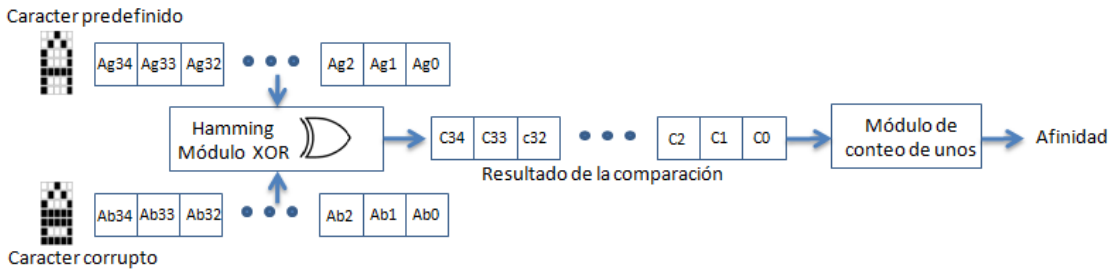


Figura 4.9. Procedimiento para asignar un valor de afinidad en la aplicación de reconocimiento de caracteres.

Se compara entonces el caracter corrupto con los 14 caracteres predefinidos y se obtendrá un valor de afinidad para cada comparación. Las comparaciones que obtienen las dos afinidades menores, serán las dos mejores soluciones. Para conformar la población inicial del micro-SIA se utilizan estos dos mejores individuos con otros tres que se generan aleatoriamente. La encomienda del micro-SIA es encontrar la mejor solución para el reconocimiento del caracter corrupto. En la figura 4.10 se presenta una aproximación al sistema de reconocimiento implementado.

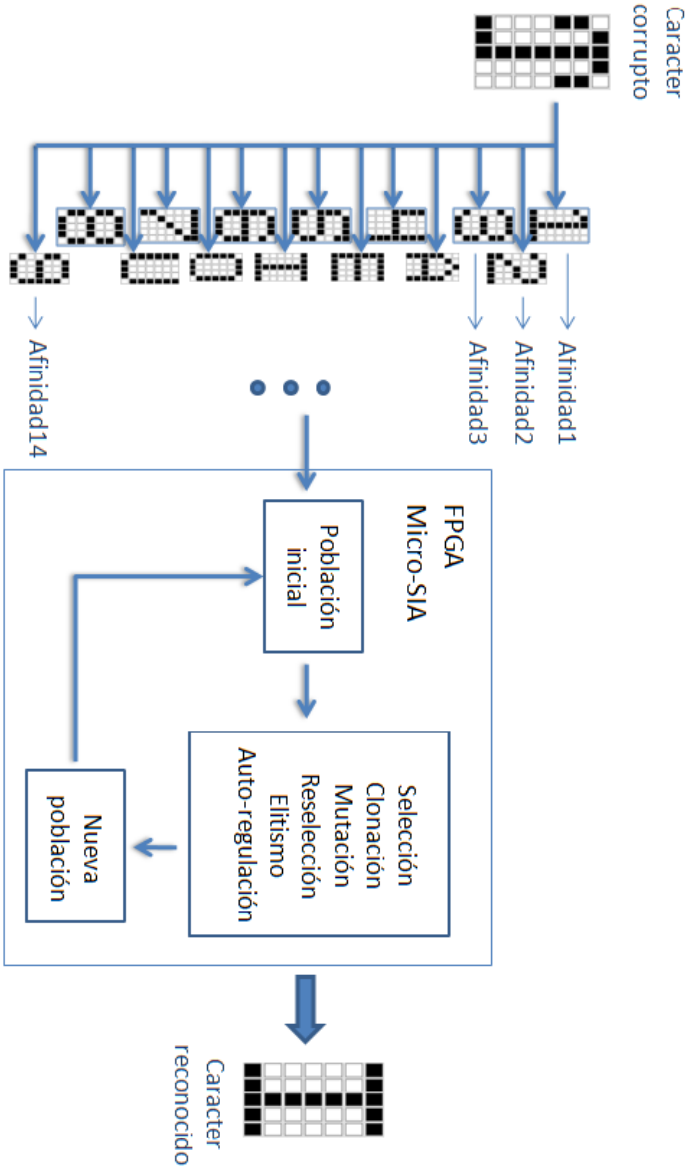


Figura 4.10. Reconocimiento de caracteres con el micro-SIA con ejecución intrínseca.

Las características del micro-SIA embebido en el FPGA son las siguientes: micro-población de 5 individuos (anticuerpos), individuos de 35 bits, 10 generaciones para alcanzar la convergencia nominal, 50 generaciones para la convergencia general (criterio de paro del algoritmo), 15 clones, mutación uniforme, elitismo para las dos mejores soluciones y ruido estocástico integrando tres clones elegidos aleatoriamente dentro de la convergencia nominal o tres individuos generados aleatoriamente si se

alcanzó la convergencia nominal. Puede observarse que se mantienen los parámetros y operadores utilizados en los experimentos anteriores, con la excepción de las 10 generaciones para alcanzar la convergencia nominal.

Para enviar el caracter corrupto al FPGA y leer los datos resultantes, se implementó una aplicación que permite desde una PC la interacción con el FPGA. Lo anterior se logra vía el puerto serial (RS-232). En la figura 4.11 se muestra la aplicación en software en tiempo de ejecución. Esta aplicación permite codificar el caracter, enviar la cadena de 35 bits por el puerto serial y decodificar una cadena de 35 bits que el FPGA devuelve como resultado del reconocimiento de caracteres.

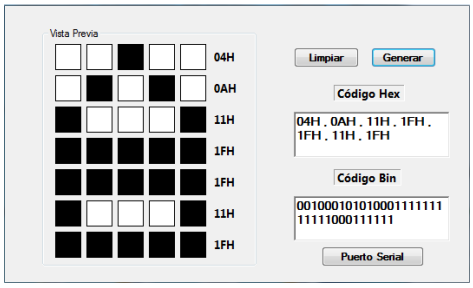
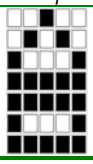
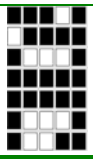
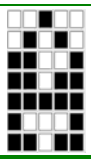
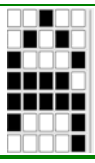
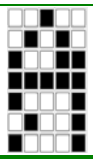
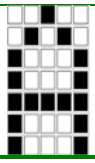


Figura 4.11. Aplicación para codificar y decodificar que permite la interacción con el FPGA.

En la tabla 4.5 se muestra a detalle cómo el micro-SIA evoluciona un caracter. En la primera columna se aprecia el caracter corrupto y las columnas posteriores muestran las modificaciones sufridas en las generaciones indicadas. Estas generaciones hacen referencia al ciclo externo del micro-SIA.

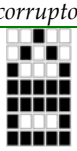
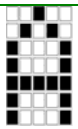
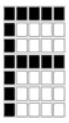
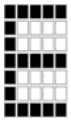
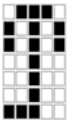
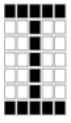
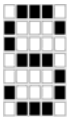
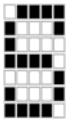
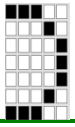
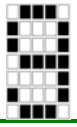
Tabla 4.5. Evolución de un caracter por medio del micro-SIA.

Caracter corrupto	*Generación 10*	*Generación 20*	*Generación 30*	*Generación 40*	*Generación 50 Salida*
					

El micro-SIA convergió a las 50 generaciones, encontrando la mejor solución. El caracter corrupto utilizado para este experimento tiene ruido aditivo generado de manera arbitraria.

En la tabla 4.6 se reportan los resultados de 5 caracteres corruptos introducidos al sistema de reconocimiento. Se realizaron 5 ejecuciones de cada uno de éstos y los resultados fueron siempre los mismos. Los caracteres utilizados tienen ruido aditivo o sustractivo, también generado de manera arbitraria sin alguna tendencia. El último de los caracteres es muy distinto a cualquiera de los caracteres predefinidos en el sistema, sin embargo, el caracter reconocido fue reiteradamente el mismo que se reporta en la tabla 4.6.

Tabla 4.6. Resultados experimentales para 5 caracteres corruptos.

Carácter corrupto	*Generación 50 Salida*
	
	
	
	
	

Para estos experimentos no se midió el tiempo de ejecución debido a que intencionalmente se incluyeron retardos para monitorear las modificaciones de los caracteres al paso de las generaciones. De acuerdo con los resultados reportados en la tabla 4.6 el micro-SIA aplicado al reconocimiento de caracteres funcionó correctamente al obtener el caracter más parecido a uno corrupto. Aún tratándose de una cadena de 35 bits, con 2^{35} posibles combinaciones diferentes, el micro-SIA encontró muy buenas soluciones. Resultó ventajoso que desde la primera población generada se utilizó elitismo, las mejores soluciones son prometedoras y guían la búsqueda, acelerando la convergencia sin la necesidad de explorar la totalidad de las combinaciones posibles.

La arquitectura bioinspirada que se diseñó es robusta y permite cambios sencillos y drásticos, ya sea para reducir o aumentar el número de caracteres predefinidos. La ejecución paralela en el manejo de datos se conserva en similitud al diagrama de la figura 4.7. La única limitante es la cantidad de recursos internos del FPGA, que para este diseño y considerando la interfaz con el puerto serial, se utilizó el 63% de la capacidad total del dispositivo.

Capítulo 5

5.1 Conclusiones

En este trabajo de tesis se diseñó e implementó un algoritmo bioinspirado con un tamaño de población reducido, basado en el principio de selección clonal del sistema inmune artificial, para solucionar problemas de optimización numérica; a este nuevo algoritmo se le denominó *micro-SIA*. El principio de selección clonal extrapolado del sistema inmune biológico, se caracteriza por aumentar el tamaño de la población durante la etapa de clonación y sólo permite aplicar operadores de mutación, por lo que mantener la diversidad de la población y asegurar la convergencia al óptimo global del algoritmo, han sido los principales motivos que hicieron interesante el estudio de esta heurística.

Los elementos principales en la metodología empleada consisten en incorporar una convergencia nominal que se alcanza después de un número fijo de generaciones, un elitismo que preserva al mejor o mejores individuos encontrados al término de la convergencia nominal y un criterio para incorporarlos como parte de una nueva población al momento de reinicializar el algoritmo que como condición de paro debe cumplir un ciclo externo. Debido a que el proceso inmune simulado por este algoritmo no incluye un operador de cruza, la clonación con crecimiento controlado (fijo de 15 clones en esta tesis) y los operadores de mutación diseñados son primordiales para mantener la diversidad y acelerar la convergencia. Experimentalmente se probaron diferentes micro-poblaciones, sin embargo utilizando 5 anticuerpos y 15 clones se obtuvieron los mejores resultados.

El número de generaciones en el micro-SIA para alcanzar la convergencia nominal se planteó de 10, debido a que menos generaciones no aceleran la búsqueda y más generaciones no mejoran los resultados.

Se probaron varios operadores de mutación, no obstante los mejores resultados se obtuvieron con los que aquí se presentaron y que involucran al número de clones, al rango de las variables y a la generación corriente o actual dentro de la ejecución del algoritmo. Se demostró que estos operadores introducen nueva información a la búsqueda, repercutiendo en una mayor explotación de las soluciones prometedoras a la vez que coadyuvan en la exploración del espacio de búsqueda al permitir pasos grandes al principio del algoritmo y pasos muy pequeños al final del mismo. El alto grado de diversidad permitió una amplia exploración del espacio de búsqueda y evitó el estancamiento en óptimos locales.

En las funciones de prueba utilizadas, el elitismo designado (preservando a los dos mejores individuos), tanto en el ciclo interno de la convergencia nominal como en el ciclo externo, coadyuva a asegurar la convergencia del algoritmo. No obstante que la selección por *ranking* utilizada en esta tesis es un operador con alta presión selectiva (al hacer que los mejores individuos sean elegidos con mucha mayor probabilidad), el ruido estocástico generado evita la disminución en la diversidad de la población.

Se comparó el desempeño del micro-SIA con la versión estándar (CLONALG), en este rubro se comprobó que el micro-SIA converge más rápido al óptimo, realizando menos evaluaciones a la función objetivo, utilizando un número mínimo de individuos que a la vez permitió reducir el número de clones generados, que es un proceso crítico en la versión estándar. Aunque CLONALG también tiene parámetros sensibles, en el micro-SIA estos son más dependientes entre sí, por lo que sintonizarlos correctamente es una desventaja que no fue notoria para las funciones de prueba utilizadas pero que seguramente influirá en otro tipo de funciones.

El micro-SIA funciona correctamente, convergiendo al óptimo conocido, para problemas de optimización mono-objetivo sin restricciones y con restricciones, por lo menos para las funciones de prueba seleccionadas. El mecanismo para el manejo de restricciones que se presentó en esta tesis, permitió que el micro-SIA pudiera lidiar satisfactoriamente con este tipo de problemas. Sólo se resolvieron dos problemas de esta clase, por lo que seguramente será necesario adecuar este mecanismo para resolver otro tipo de problemas que requieran el manejo de espacios restringidos.

Aprovechando que el manejo poblacional del micro-SIA está reducido, al igual que el costo computacional, se realizó la migración de la versión diseñada para optimización numérica hacia una plataforma en hardware. La representación de los individuos tuvo que cambiarse del dominio de los números reales a los binarios. Los esfuerzos se dirigieron hacia dos vertientes: un microcontrolador comercial de 8 bits y un FPGA. El primero de estos dispositivos tiene una arquitectura limitada en cuanto al número de recursos, por lo que la implementación exitosa del problema de la *maximización de unos* (problema clásico en el área del reconocimiento de patrones) demostró no sólo que el micro-SIA se puede ejecutar de manera intrínseca, sino que es viable considerar a los microcontroladores para este tipo de aplicaciones. Experimentalmente se comprobó que cuando en la población inicial se obtienen soluciones potenciales, el algoritmo converge más rápidamente al óptimo.

La técnica implementada para resolver el problema de la maximización de unos en el microcontrolador, se extendió para manejar individuos de 16 bits y se resolvió con éxito en un FPGA. Para la particularidad de esta implementación se diseñó una arquitectura modular que tiene la capacidad de ejecutar paralelamente el algoritmo y es robusta, por lo que se puede modificar sin inconvenientes.

Se decidió aplicar el micro-SIA en un sistema de reconocimiento de caracteres, con el propósito de revisar cómo se comporta el algoritmo ante este tipo de problemas. Aprovechando la arquitectura embebida diseñada para la resolver la maximización de unos en el FPGA, se realizó una adecuación para que pudiera reconocer caracteres corruptos en un conjunto de 14 caracteres predefinidos. Cada individuo de la población estuvo representado por una cadena de 35 bits, en congruencia con una matriz de 7 filas y 5 columnas. El micro-SIA se ejecutó intrínsecamente para reconocer el caracter de entrada al sistema, obteniendo buenos resultados sin importar si el caracter corrupto presentaba ruido aditivo o sustractivo. Quedó comprobado que los parámetros definidos para esta aplicación, estuvieron correctamente seleccionados.

5.2 Trabajos a futuro

Es posible afirmar que el desempeño del micro-SIA es tan bueno o mejor, para los casos aquí presentados, que el del SIA en su versión estándar, aunque con menores requerimientos de espacio en memoria de datos, lo que permite enfocar el diseño hacia aplicaciones con un bajo costo computacional o a implementaciones en circuitos embebidos, refiriéndose al denominado hardware evolutivo. En este campo faltaría aplicarlo a problemas reales de ingeniería que requieran la ejecución intrínseca de un algoritmo bioinspirado, como es el caso de la elección de una ruta más corta en el desplazamiento de un robot móvil o el reconocimiento de caracteres en sistemas más complejos con un número muy grande de datos.

Será necesario realizar un análisis de covarianza para determinar el efecto de los parámetros sensibles en el funcionamiento del micro-SIA. También es importante realizar un análisis estadístico de la uniformidad en la distribución de las soluciones encontradas. Probar el algoritmo con otras funciones dará un panorama más certero de la eficiencia del micro-SIA, especialmente cuando estas funciones infieran un manejo de restricciones.

La optimización con múltiples objetivos se puede incorporar a la propuesta aportada en esta tesis, seguramente éste será un trabajo a futuro, al igual que explorar la adaptación de otros algoritmos bioinspirados a un esquema de micro-población.

Referencias

[1] J. Koza and M. Keane. The importance of reuse and development in evolvable hardware. In Proceedings of 2003 NASA/DoD Conference on Evolvable Hardware. Los Alamitos, CA: IEEE Computer Society.

[2] R. Zebulum and A. Stoica. Mixtrinsic evolution. In J. F. Miller, A. Thompson, P. Thomson, and T. C. Fogarty, editors, Proc. of Third International Conference on Evolvable System: From Biology to Hardware (ICES 2000), pages 208–217, Edinburgh, Scotland, April 2000. Springer-Verlag.

[3] P.C. Haddow. An Evolvable Hardware FPGA for Adaptive Hardware. Congress on Evolutionary Computation pp. 553-560, 2000.

[4] G. Harik, F. Lobo, D. Goldberg. The compact genetic algorithm. Evolutionary Computation, IEEE Transactions on Volume 3, Issue 4, Nov. 1999 Page(s):287 – 297.

[5] David E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading, MA, 1989.

[6] K. Krishnakumar. Micro-genetic algorithms for stationary and non-stationary function optimization. In SPIE Proceedings: Intelligent Control and Adaptive systems, pages 289–296, 1989.

[7] G. Dozier, J. Bowen and D. Bahler. Solving Small and Large Scale Constraint Satisfaction Problems Using a Heuristic-Based Microgenetic Algorithm, in Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel and H. Kitano (editors), Proceedings of the First IEEE Conference on Evolutionary Computation (ICEC'94), pages 306-311.

[8] C. A. Coello and G. Toscano. A Micro-Genetic Algorithm for multiobjective optimization. First International Conference on Evolutionary Multi-criterion Optimization. Springer – Verlag, Lecture Notes in Computer Science number 1993, 2001, pp. 126 – 140.

[9] J. C. Fuentes and C. A. Coello. Handling Constraints in Particle Swarm Optimization Using a Small Population Size, in Lecture Notes in Computer Science, Springer. MICAI 2007: Advances in Artificial Intelligence, Volume 4827/2007.

[10] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. Handbook of Evolutionary Computation. Institute of Physics Publishing and Oxford University Press, New York, 1997.

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

[12] C. A. Coello. Introducción a la computación evolutiva. Notas del curso. CINVESTAV-IPN, Sección de Computación, Departamento de Ingeniería Eléctrica, 2003.

[13] D. E. Goldberg and K. 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.

[14] E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm Intelligence: From natural to artificial systems. Oxford University Press, 1999.

[15] M. Dorigo & T. Stützle. Ant Colony Optimization, MIT Press, 2004.

[16] J. Kennedy and R. C. Eberhart. Swarm Intelligence. Morgan Kaufmann, 2001.

[17] A. Tyrrell, G. Auer and C. Bettstetter. Bio-inspired networks and communication systems: Fireflies as role models for synchronization in ad hoc networks. December 2006. Proceedings of the 1st international conference on Bio inspired models of network, information and computing systems BIONETICS '06. Publisher: ACM.

[18] A. Delgado. DNA chips as lookup tables for rule based systems. IEE Computing and Control Engineering Journal, Vol. 13, No. 3, pp. 113-119, 2002.

[19] R. Sun and L. Bookman. Computational Architectures Integrating Neural and Symbolic Processes. Kluwer Academic Publishers, 1994.

[20] D. Dasgupta and N. Attoh-Okine. Immunity-Based Systems: A Survey. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. October 1997.

[21] L. N. de Castro and J. Timmis. An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm. Springer-Verlag, 2002.

[22] D. Dasgupta. Advances in artificial immune systems. Computational Intelligence Magazine, IEEE. Volume 1, Issue 4, Nov. 2006 Page(s):40 – 49.

[23] N. Cruz. Sistema inmune artificial para solucionar problemas de optimización. Tesis Doctoral. CINVESTAV- IPN. México, 2004.

[24] L. Nunes de Castro and F. J. Von Zuben. Learning and optimization using the clonal selection principle. IEEE Trans. Evol. Comput., vol. 6, no. 3, pp. 239–251, Jun. 2002.

[25] L. Nunes de Castro and F. J. Von Zuben. The clonal selection algorithm with engineering applications. Proceedings of Genetic and Evolutionary Computation Conference, Workshop on AISAA, pp. 36-37, July 2000.

[26] L. Nunes de Castro. The immune response of an artificial immune network (aiNet). Evolutionary Computation, 2003. CEC '03. The 2003 Congress on Volume 1. Dec. 2003 Page(s):146 - 153 Vol.1.

[27] L. Nunes de Castro and J. Timmis. An Artificial Immune Network for Multimodal Function Optimization. Proceedings of IEEE Congress on Evolutionary Computation (CEC'02), Hawaii, pp. 699- 674, 2002.

[28] C. Aporntewan and P. Chongstitvatana. A hardware implementation of the Compact Genetic Algorithm. Evolutionary Computation, 2001. Proceedings of the 2001 Congress on Volume 1, 27-30 May 2001 Page(s):624 - 629 vol. 1.

[29] J. Gallagher and G. Kramer. A family of compact genetic algorithms for intrinsic evolvable hardware. Evolutionary Computation, IEEE Transactions on Volume 8, Issue 2, April 2004 Page(s):111 – 126.

[30] F. Cupertino and E. Mininno. Optimization of Position Control of Induction Motors using Compact Genetic Algorithms. IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on Nov. 2006. Page(s):55 – 60.

[31] K. Deb. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, 186:311–338, 2000.

[32] E. Mezura, J. Velázquez, C. A. Coello. A comparative study of differential evolution variants for global optimization. ACM, GECCO 2006: 485-492.

[33] S. Koziel and Zbigniew Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrainted parameter optimization. Evolutionary Computation, 7(1):19–14, 1999.

[34] J.D. Schaffer and L.J. Eshelman. On crossover as an evolutionary viable strategy. R.K. Belew and L.B. Booker, editors. Proceedings of the 4th International Conference on Genetic Algorithms. Page(s): 61-68, Morgan Kaufmann, 1991.

[35] A. Tyrrell. Evolutionary strategies and intrinsic fault tolerance. Dept. of Electron., York Univ. Evolvable Hardware. Proceedings. The Third NASA/DoD Workshop on 2001, Long Beach, CA, USA, on page(s): 98-106. 2001.

[36] P.J. Fleming and R. C. Purshouse, Evolutionary algorithms in control system engineering: a survey, Control Engineering Practice, Vol. 10, pp. 1223-1241, 2002.

[37] D. Sucley. Genetic Algorithms in the Design of FIR Filters, IEEE Proceedings, Vol.138, pp. 234-238,1991.

[38] K. Arne and J. Torresen. Implementing Evolution of FIR-Filters Efficiently in an FPGA. paper.pdf. In proc. of 2003 NASA/DoD Conference on Evolvable Hardware (EH-2003), July, 2003, Chicago, Illinois, USA.

[39] M. Yasunaga, T. Nakamura and J. H. Kim. Genetic algorithm-based design methodology for pattern recognition hardware. In Evolvable Systems: From Biology to Hardware. Third Int. Conf., ICES 2000, volume 1801 of Lecture Notes in Computer Science, J. Miller et al. (eds.), Springer-Verlag, 2000, pp. 264–273.

[40] Y. Thoma, E. Sanchez, D. Roggen and C. Hetherington. Prototyping with a bio-inspired reconfigurable chip. Rapid System Prototyping, 2004. Proceedings. 15th IEEE International Workshop on 28-30 June 2004 Page(s):239 – 246.

Anexo A

f01 – Sphere Model

$$f_1(x) = \sum_{i=1}^{30} (x_i)^2$$

$-100 \leq x_i \leq 100$

$min\ (f_1) = f_1\ (0, \ldots, 0) = 0$

f02 – Schwefel's Problem

$$f_2(x) = \sum_{i=1}^{30} |x_i| + \prod_{i=1}^{30} |x_i|$$

$-10 \leq x_i \leq 10$

$min\ (f_2) = f_2\ (0, \ldots, 0) = 0$

f03 – Schwefel's Problem

$$f_3(x) = \sum_{i=1}^{30} \left(\sum_{j=1}^{i} x_j \right)^2$$

$-100 \leq x_i \leq 100$

$min\ (f_3) = f_3\ (0, \ldots, 0) = 0$

f04 – Schwefel's Problem

$$f_4(x) = max_i \{ [x_i], 1 \leq i \leq 30 \}$$

$-100 \leq x_i \leq 100$

$min\ (f_4) = f_4\ (0, \ldots, 0) = 0$

f05 – Generalized Rosenbrock's Function

$$f_5(x) = \sum_{i=1}^{29} \left| 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right|$$

$-30 \leq x_i \leq 30$

$min\ (f_5) = f_5\ (1, \ldots, 1) = 0$

f06 – Step Function

$$f_6(x) = \sum_{i=1}^{30} (\lfloor x_i + 0.5 \rfloor)^2$$

$-100 \leq x_i \leq 100$

$min\ (f_6\ ) = f_6\ (0, \ldots, 0) = 0$

f07 – Quartic Function with Noise

$$f_7(x) = \sum_{i=1}^{30} i x_i^4 + random[0,1)$$

$-1.28 \leq x_i \leq 1.28$

$min\ (f_7\ ) = f_7\ (0, \ldots, 0) = 0$

f08 – Generalized Schwefel's Problem

$$f_8(x) = \sum_{i=1}^{30} \left(x_i \sin\left(\sqrt{|x_i|} \right) \right)$$

$-500 \leq x_i \leq 500$

$min\ (f_8\ ) = f_8\ (420.9687, \ldots, 420.9687) = -12596.5$

f09 – Generalized Rastrigin's Problem

$$f_9(x) = \sum_{i=1}^{30} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$$

$-5.12 \leq x_i \leq 5.12$

$min\ (f_9\ ) = f_9\ (0, \ldots, 0) = 0$

f10 – Ackley's Function

$$f_{10}(x) = -20e\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2} \right) - e\left(\frac{1}{30}\sum_{i=1}^{30} \cos(2\pi x_i) \right) + 20 + e$$

$-32 \leq x_i \leq 32$

$min\ (f_{10}\ ) = f_{10}\ (0, \ldots, 0) = 0$

f11 – Generalized Griewank's Function

$$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$-600 \leq x_i \leq 600$

$min\ (f_{11}\ ) = f_{11}\ (0, \ldots, 0) = 0$

Anexo B

g01

Min

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

$$\begin{aligned}
g(x)_1 &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\
g(x)_2 &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\
g(x)_3 &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\
g(x)_4 &= -8x_1 + x_{10} \leq 0 \\
g(x)_5 &= -8x_2 + x_{11} \leq 0 \\
g(x)_6 &= -8x_3 + x_{12} \leq 0 \\
g(x)_7 &= -2x_4 - x_5 + x_{10} \leq 0 \\
g(x)_8 &= -2x_6 - x_7 + x_{11} \leq 0 \\
g(x)_9 &= -2x_8 - x_9 + x_{12} \leq 0
\end{aligned}$$

$0 \leq x_i \leq 1 (i = 1, ..., 9)$, $0 \leq x_i \leq 100 (i = 10, 11, 12)$ y $0 \leq x_{13} \leq 1$

$min(f) = f(1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1) = -15$

g02

Max

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

$$\begin{aligned}
g(x)_1 &= 0.75 - \prod_{i=1}^{n} x_i \leq 0 \\
g(x)_1 &= \sum_{i=1}^{n} x_i - 7.5n \leq 0
\end{aligned}$$

$n = 20$, $0 \leq x_i \leq 10 (i = 1, ..., n)$

$max(f) = f(x*) = -0.803619$

Anexo C

Micro-algoritmo de evolución diferencial.

La Evolución Diferencial es un algoritmo evolutivo propuesto por Storn y Price [10] para resolver problemas de optimización principalmente en espacios continuos y en el cual las variables se representan mediante números reales. Actualmente se ha utilizado en problemas de alta complejidad con muy buenos resultados [11].

En la ED se genera de forma aleatoria una población inicial de individuos (vectores originales), de los cuales se seleccionan tres, también aleatoriamente y que sean distintos entre sí. A cada uno de estos individuos se le denomina vector objetivo y se denotan por *r1*, *r2* y *r3*, en donde *r3* es el vector base o principal. Con ayuda de un operador especial (*F*) se realiza una combinación lineal con las diferencias entre *r1*, *r2* y *r3*, con la intención de generar nuevos vectores ruidosos, a esta etapa se le conoce como mutación. *F* es entonces, un factor que escala la diferencia entre vectores. Posteriormente se generan vectores de prueba a partir de la recombinación de vectores ruidosos y vectores originales, lo anterior se logra a través de un parámetro denominado tasa de recombinación (*CR*). Para finalizar el algoritmo se procede a seleccionar el vector que se copiará a la generación siguiente, comparando los vectores de prueba con los originales en base a su aptitud.

La versión más común de la ED es la denominada **DE/rand/1/bin**; que fue la que se abordó en este trabajo y se lista en Algoritmo 1, se recomienda referirse a [12] para mayores detalles del mismo. La cantidad de generaciones, el tamaño de la población y los parámetros *CR* y *F*, son definidos por el usuario.

Algoritmo 1. Evolución Diferencial estándar, versión DE/rand/1/bin

Begin

$G = 0$

Crear aleatoriamente la población inicial $\vec{x}_G \; \forall i, i = 1, \ldots, NP$

Evaluar $f(\vec{x}_G) \; \forall i, i = 1, \ldots, NP$

For $G = 1$ to G_{max} **Do**

For $i = 1$ to NP **Do**

Seleccionar aleatoriamente $r_1 \neq r_2 \neq r_3$

$j_{rand} = \text{randint}(1,D)$

For $j = 1$ to D **Do**

If ($\text{rand}_j[0,1) < CR$ or $j = j_{rand}$) **then**

$u^i_{j,G+1} = x^{r_3}_{j,G} + F\left(x^{r_1}_{j,G} - x^{r_2}_{j,G}\right)$

Else

$u^i_{j,G+1} = u^i_{j,G}$

End if

End for

If $\left(f\left(\vec{u}^i_{G+1}\right) \leq f\left(\vec{x}^i_G\right)\right)$ **then**

$\vec{x}^i_{G+1} = \vec{u}^i_{G+1}$

Else

$\vec{x}^i_{G+1} = \vec{x}^i_G$

End if

End For

$G = G + 1$

End For

End

La versión con población reducida de este mismo algoritmo se observa modularmente en la Figura 2. En nuestros experimentos con ED utilizamos una población de 5 individuos, ya que se comprobó funcionalmente que con un menor número de individuos se obtiene una convergencia prematura y utilizar más de 5 individuos no mejora los resultados. Se determinó que el criterio para alcanzar la convergencia nominal fuera al cabo de 5 generaciones, refiriéndose al ciclo interno del algoritmo. En este rubro también se realizaron pruebas con 10 generaciones, sin observarse alguna mejoría en los resultados.

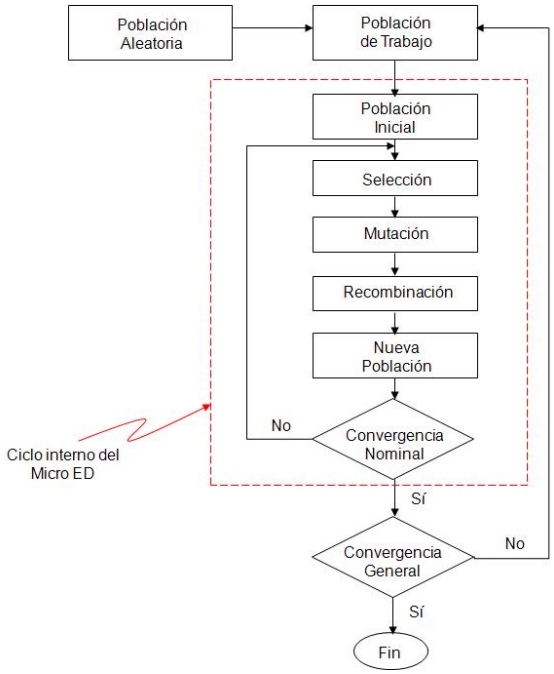


Figura 2. Diagrama a bloques del micro algoritmo de ED.

En la generación cero, utilizando una población aleatoria de 5 individuos, se procedió a copiar los mismos en la población de trabajo y en la población inicial. Nótese que para el ciclo interno en donde aplica la convergencia nominal, el algoritmo ED se implementa sin cambios en sus operadores y estrategias para evaluar la metodología de manera básica. Al alcanzarse la convergencia nominal es necesario determinar cuántos individuos se copiarán a la población de trabajo y aplicar el proceso de reinicialización. Después de una serie de experimentos, se decidió copiar los 4 mejores individuos y el restante se genera aleatoriamente. Los experimentos realizados contemplaron copiar 1, 2, 3 y 4 individuos con la mejor aptitud, sin embargo con 4 individuos se obtuvieron los mejores resultados. En este micro algoritmo la diversidad se mantiene gracias a las etapas de mutación y recombinación del propio algoritmo base de ED y al individuo aleatorio incorporado en la población de trabajo.

1. Experimentos y Resultados

Para los experimentos se utilizaron las siguientes funciones de alta dimensionalidad, referidas en [12]. ***f1*** y ***f2*** son funciones unimodales y separables. ***f5*** es una función multimodal y no separable.

f1 – Esfera de De Jong

$$f(x) = \sum_{i=1}^{30} (x_i)^2 \qquad (1)$$

$-100 \leq x_i \geq 100$
$min\ (f_1) = f_1\ (0, \ldots, 0) = 0$

f2 – Función techo

$$f(x) = \sum_{i=1}^{30} (\lfloor x_i + 0.5 \rfloor)^2 \qquad (2)$$

$-100 \leq x_i \geq 100$
$min\ (f_2) = f_2\ (0, \ldots, 0) = 0$

f5 – Función generalizada de Rosenbrock

$$f(x) = \sum_{i=1}^{29} \left| 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right| \qquad (3)$$

$-30 \leq x_i \geq 30$
$min\ (f_3) = f_3\ (1, \ldots, 1) = 0$

Las tres funciones fueron probadas en el micro algoritmos ED, realizando 20 ejecuciones para cada una y obteniendo los resultados que a continuación se discutirán. Se definieron los parámetros indicados en la Tabla 1, los cuales son sugeridos en [12]:

***Tabla 1**: Parámetros utilizados para el micro algoritmo de ED.*

Función	*CR*	*F*
f1	0.9	$\in [0.3,0.9]$
f2	0.0	$\in [0.3,0.9]$
f5	0.0	$\in [0.3,0.9]$

En la Tabla 2 se indican las generaciones utilizadas para cada función, así como los resultados experimentales alrededor del valor óptimo.

***Tabla 2**: 20 ejecuciones del micro algoritmo de ED.*

Función	*Ciclo Externo*	*Ciclo interno*	*Mejor*	*Peor*	*Media*
f1	300	5	0.0	0.0064	0.00010
f2	200	5	0.0	1.0	0.20
f5	300	5	0.0	0.0010	0.00012

En la Figura 3, se aprecia el comportamiento del algoritmo de ED estándar en su versión **DE/rand/1/bin** con respecto a la aptitud, para ***f*2**. Se utilizó una población de 60 individuos, 1000 generaciones y los mismos valores de *CR* y *F* listados en la Tabla 1.

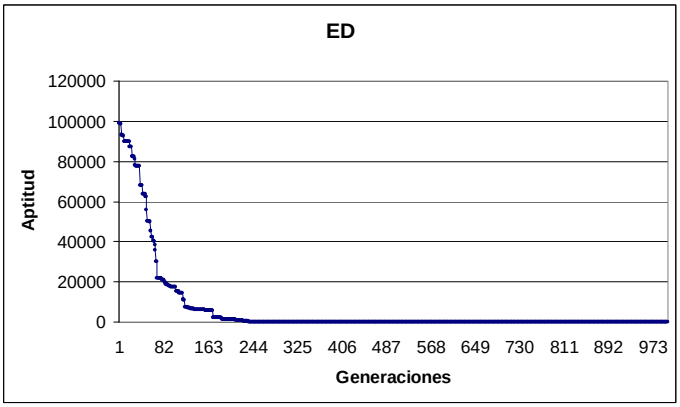


*Figura 3. Aptitud para **f*2** con ED estándar.*

Utilizando una población de 5 individuos y los parámetros indicados en la Tabla 2, se obtuvo la gráfica exhibida en la Figura 4, para ***f*2** probada en el micro algoritmo de ED.

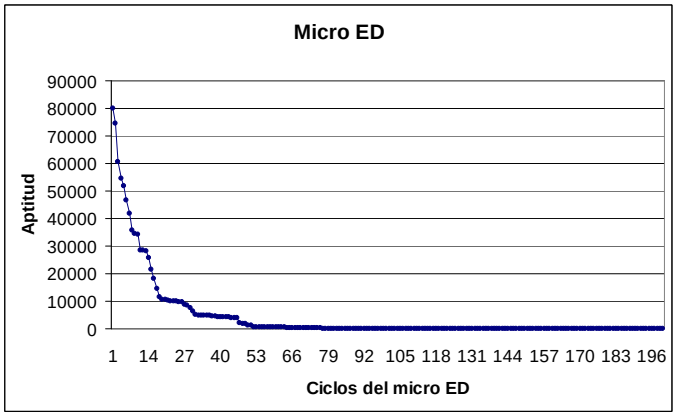


*Figura 4. Aptitud para **f*2** con micro algoritmo de ED.*

Es importante notar que para las funciones experimentales, el desempeño del algoritmo de población reducida es tan eficiente como el del algoritmo en su versión estándar. La cantidad de evaluaciones a la función objetivo es similar, considerando el ciclo interno del micro algoritmo.

2. Conclusiones

El micro algoritmo de ED (en su versión *DE/rand/1/bin*) generó buenos resultado con base en la metodología aplicada y con respecto a las funciones seleccionadas. En los resultados experimentales se obtuvo el óptimo para las tres funciones de prueba de

alta dimensionalidad, lo que denota resultados alentadores para continuar con la investigación. Es posible afirmar que el desempeño del micro algoritmo de ED es tan bueno como el de su contraparte estándar, aunque con menores requerimientos de espacio en memoria de datos, lo que permite enfocar el diseño hacia aplicaciones con un bajo costo computacional o a implementaciones en circuitos embebidos (p. e. dispositivos de lógica programable y microcontroladores convencionales) refiriéndose al denominado *hardware evolutivo*. El manejo de espacios restringidos y la optimización con múltiples objetivos se pueden incorporar al esquema básico planteado, seguramente éste será un trabajo a futuro, al igual que explorar la adaptación de otros algoritmos bioinspirados comunes en la resolución de problemas de optimización.

Referencias

1. D. Ashlock. Evolutionary Computation for Modeling and Optimization. Springer, first edition, 2005.
2. M. Munetomo, Y. Satake. Enhancing Model-building Efficiency in Extended Compact Genetic Algorithms. Systems, Man and Cybernetics, 2006. ICSMC '06. IEEE International Conference on Volume 3, 8-11 Oct. 2006 Page(s):2362 – 2367.3.
3. J. Torresen. Possibilities and limitations of applying evolvable hardware to real-world application. In Field-Programmable Logic and Applications: 10th International Conference on Field Programmable Logic and Applications (FPL-2000), volume 1896 of Lecture Notes in Computer Science, R. W. Hartenstein et al. (eds.), Springer-Verlag, 2000, pp. 230–239.
4. D. E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading, MA, 1989.
5. K. Krishnakumar. Micro-genetic algorithms for stationary and non-stationary function optimization. In SPIE Proceedings: Intelligent Control and Adaptive systems, pages 289–296, 1989.
6. G. Toscano, Carlos. A. Coello. A Micro-Genetic Algorithm for multiobjective optimization. First International Conference on Evolutionary Multi-criterion Optimization. Springer – Verlag, Lecture Notes in Computer Science number 1993, 2001, pp. 126 – 140.
7. J. C. Fuentes, C. A. Coello, Handling Constraints in Particle Swarm Optimization Using a Small Population Size, in Lecture Notes in Computer Science, Springer. MICAI 2007: Advances in Artificial Intelligence, Volume 4827/2007.
8. J. C. Fuentes. Un nuevo Algoritmo de optimización basado en optimización mediante cúmulos de partículas utilizando tamaños de población pequeños. Tesis de Maestría. CINVESTAV- IPN. México, 2008.
9. T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. Handbook of Evolutionary Computation. Institute of Physics Publishing and Oxford University Press, New York, 1997.
10. R. Storn, K. Price. Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, ICSI, March 1995, ftp.icsi.berkeley.edu. Accedido por última vez el 20 de junio de 2009.

Anexo D

Bio-inspired architecture design for a PWM module

Abstract

This paper presents the analysis and design of a Pulse Width Modulation embedded unit as an alternative to convert digital signals to analog signals extrapolating the biological principle of DNA hybridization. Biological hybridization process is a binding event between two complementary DNA single strands and that leads to the formation of a double-stranded helix. The proposed technique to detect the hybridization is based on direct comparison of data, allowing a high degree of parallelism in the Bio-inspired architecture specially designed to regulate the light intensity of LED lamps as a home automation application.

1. Introduction

In digital electronic design, a PWM (Pulse Width Modulation) unit is basically a digital – analog converter (DAC) which can be used in solutions that do not require a high frequency of sampling, it converts a binary data into a series of pulses, so that the pulse duration is directly proportional to the value of binary data. PWM is widely used in different areas of control systems, such as robotics, industrial process control, power control systems, and others [1]. With regard to home automation (domotics), PWM has been very useful in regulating the light intensity of lamps (LED, fluorescent and incandescent), as well as regulating valves and small engines that automate some process [2], [3].

When designing with programmable logic devices (FPGA, CPLD), embedded realization of such converters is common [4], so this paper proposes an alternative to design a PWM unit simulating the combination of DNA strands in order to regulate the light intensity of a LED lamp in an application for home automation and so contribute to energy savings. The artificial DNA hybridization suggests a simple mechanical that compare parallel binary strands stored in data registers looking for evidence that they are complementary to each other to validate a subsequent action as a response.

This approach is very similar to evaluation of inference rules in an expert system that tries to model the reasoning of a human operator [5].

2. Electronic detection of DNA hybridization

DNA (Deoxyribonucleic Acid) is a chemical substance that is found in the nucleus of cells, which stores the basic code of all life translated as biological instructions. In the molecular genetics theory about DNA hybridization, single strands of DNA from two different species are allowed to join together to form hybrid double helices, much like a twisted ladder. These hybrid segments of DNA are used to determine the evolutionary relatedness of organisms by examining how similar (or dissimilar) the DNA base pair sequences are; in other words, the degree of hybridization is proportional to the degree of similarity between the molecules of DNA from the two species.

The way to detect the hybridization of two single strands of DNA has been replicated in the field of electronics through DNA Array, also called DNA chip or Gene array [6]. This chip is a matrix structure on which are distributed DNA single strands that have been implanted into a silicon base. These sequences have a simple fixed value that when they are incubated with strands injected up to the chip, generating helices of DNA that can be detected through electronic tools. Engineering has shown that DNA chips can be used to store and evaluate in parallel, Boolean or fuzzy rules [5], [7].

In electronic design, hybridization can be implemented using a reference register that will be compared in parallel with others test registers, looking for evidence that they are complementary. In Figure 1 test register represents the simple sequence of DNA with a fixed value and reference register is the DNA strand that is injected or introduced to chip. To maintain the analogy with biological hybridization, the reference register is unique and will be compared with all test registers in parallel form.

Note that to make this comparison, both registers must have the same size and only if they are complementary, the flag output will be high logical value. To verify that the bits of both registers are mutually complementary, XOR logic gates are used for bitwise comparison.

3. Pulse width modulation (PWM)

In digital electronic circuits a PWM embedded unit is usually accomplished by connecting a binary counter and a comparator circuit together; this last one will determine when data applied to the entrance of the unit is less than the value of binary counter constantly

changing. At the output of the PWM unit, is necessary to connect a low-pass RC filter to determine the voltage of the analog signal equivalent to digital data entered [8]. The entire period of a PWM cycle is equal to the product of the period clock signal reference (system clock) with 2^n, where n is the number of bits of the counter circuit proposed.

In the particularity of this work were considered 4-bit data registers, so in [8] is demonstrated that a converter of more than 8 bits does not get more benefits, so 16 different levels (at a rate of 2^4) of luminous intensity are adequate.

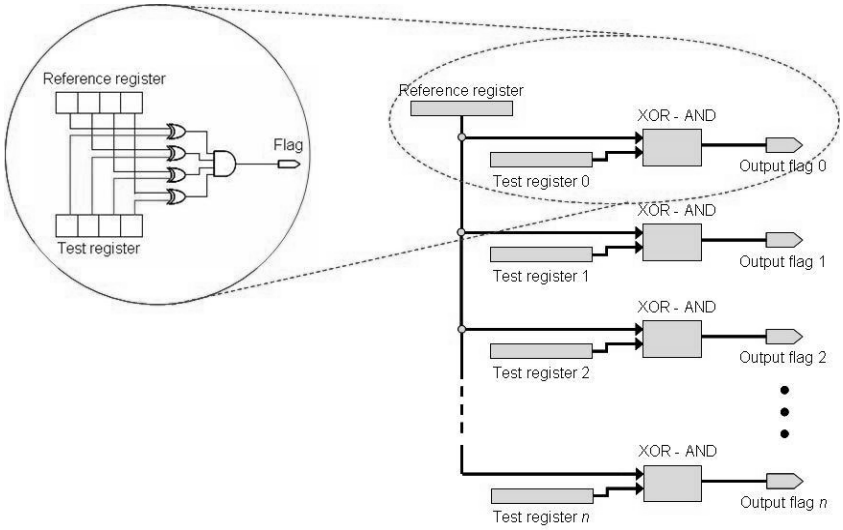


Figure 1. Standard model of artificial hybridization with digital electronic.

4. Bio-inspired architecture

We consider a modular design consisting of three parts. First part of this realization uses a reference register to be compared in parallel with every test registers, simultaneously.
According to the scheme established with 4-bit registers, 16 test registers were designed with fixed and different values, and a single reference register which establishes the digital data to convert to their analog equivalent in a range of 0 to 5 V, for example, a binary input equal to 0000 will get an analog voltage of 0V; in the same way, a binary input equal to 1111 will get the maximum analog value equal to 5V.

The fixed values of the test registers were assigned complementary in correspondence to the values that they can take at the rate of 2^n, in other words, for a binary entry 0000, stored in the reference register, the test register labeled as "0" will store a binary data 1111; for an

entry 0001, it was assigned 1110 in the test register labeled as "1". Just the only one output from the module in Figure 1 is a data flag that indicates a successful hybridization through a high logical level.

In the second part of our design each flag from the previous stage actives individually a tri-state data register which is connected to a common output bus. This time were considered 16-bit tri-state registers to improve the resolution of the converter. In this scheme, only a single tri-state register may be enabled at a time. Each tri-state register has a fixed value assigned according to the binary value that will be delivered as result of modulation.

The third and final stage of architecture, is the stage of control, consisting of a 16-bit multiplexer that with help of a 4-bit binary counter, performs a binary sweep of the only tri-state register enabled, starting with the most significant bit. The same binary counter determines that after 16 pulses clock, when the unit PWM has finished data conversion, the reference register can receive a new data to process.

Figure 2 shows the schematic diagram of the complete unit designed with the three parts. All modules were described in VHDL for logic synthesis.

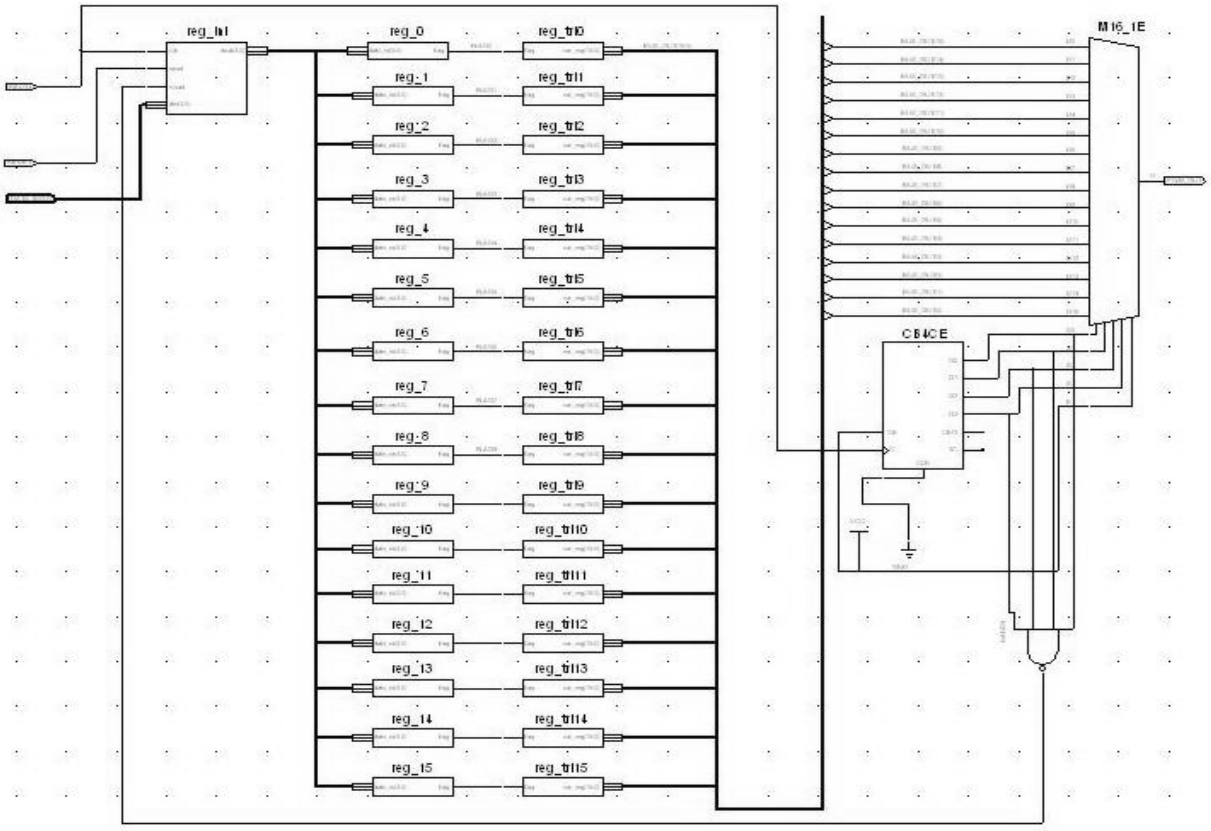


Figure 2. PWM parallel architecture using the principle of hybridization of DNA strands.

5. Results

The logic synthesis design was carried out using ISE Webpack v.8.1i, to configure a 3S200 Spartan 3 FPGA of vendor Xilinx. The PWM Bio-inspired unit implemented uses only 4% of the amount of internal resources of FPGA.

The PWM unit was tested at different frequencies, in each case calculating the values of RC filter. They were considering the following clock frequencies: 4MHz, 1MHz, 700Hz, 320Hz and 120Hz. Brief details of the conversion can be seen in Figure 3. It verifies the correct operation of PWM generator.

Tri-state registers	*Binary data*
reg_tri0	0000000000000000
reg_tri1	1000000000000000
reg_tri2	1100000000000000
reg_tri3	1110000000000000
reg_tri4	1111000000000000
reg_tri5	1111100000000000
reg_tri6	1111110000000000
reg_tri7	1111111000000000
reg_tri8	1111111100000000
reg_tri9	1111111110000000
reg_tri10	1111111111000000
reg_tri11	1111111111100000
reg_tri12	1111111111110000
reg_tri13	1111111111111000
reg_tri14	1111111111111100
reg_tri15	1111111111111110

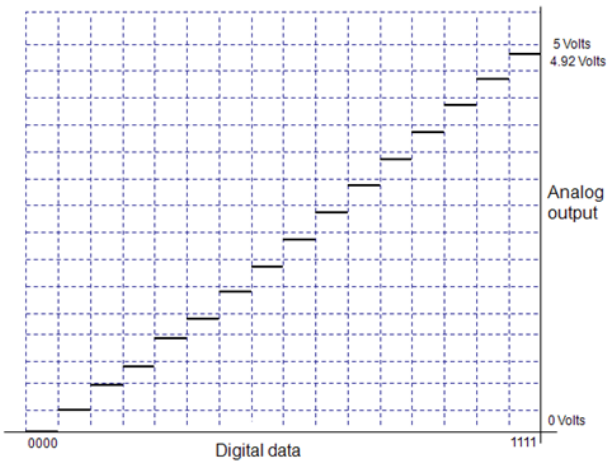


Figure 3. Details of the digital–analog conversion in practical experiments.

It was necessary to design an electronic power stage to attach the PWM output with the LED lamp used in laboratory. In Figure 4 we show the lamp used in our experiments. This lamp was designed with similar characteristics as commercial lamps with a panel of 18 x 18 high brightness LEDs.

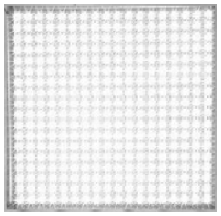


Figure 4. LED lamp used in laboratory.

6. Conclusions

The artificial hybridization of DNA single strands allows directing a design based on inference rules, into a natural parallelization that can be explored through implementation in programmable logic devices. This technique transferred to Bio-inspired hardware has allowed the completion of a functional PWM generator as an alternative to convert digital signals to analog signals.

Each tri-state register enabled individually, as a result of a successful hybridization, stores a value that can be updated in real-time increasing the scope of this proposal. This will allow attacking the quantification problem, which directly affects the converter resolution and that a conventional PWM unit could not stand it.

This modular design supports changing the content of the registers according to the needs of modulation, being possible to increase or decrease the resolution of the conversion.

It will be interesting to explore the capabilities of parallel evaluation of inference rules in expert systems more complex; in future work we will apply this same technique to create intelligent systems that allow the automatic regulation of the luminous intensity in conventional lamps, focusing more specialized home automation applications, encouraging a systematic energy saving.

References

[1] Zrilic, D. G.: Alternative Approach to Use of Pulse Width Modulation, Automation Congress, 2006. WAC '06. World IEEE, 24-26 July 2006, pp. 19–24.

[2] Alonso J. M.; Cardesin, J.; Calleja, A. J.; Rico-Secades, M.; García, J.: A fluorescent lamp electronic ballast for railway applications based on low cost microcontroller, Industry Applications Conference, 2003. 38th IAS Annual Meeting IEEE. Conference Record of the Volume 1, Issue , 12-16 Oct. 2003, pp. 523 – 530.

[3] Sandu, F.; Romanca, M.; Nedelcu, A.; Borza, P.; Dimova, R.: Remote and mobile control in domotics, Optimization of Electrical and Electronic Equipment, 2008. OPTIM 2008. 11th International Conference on 22-24 May 2008, pp. 225 – 228.

[4] Dan, D.; Su, C.; Joss, G.: FPGA implementation of PWM pattern generators [for PWM invertors], Canadian Conference on Electrical and Computer Engineering IEEE, 2001, vol.1, pp. 225-230.

[5] Delgado, A.: Rule base evaluation using DNA chips, Proceedings American Control Conference, pp. 3242 – 3245, Anchorage - Alaska, Mayo 8-10, 2002.

[6] Moeller, R.; Fritzsche, W.: Chip-based electrical detection of DNA. Nanobiotechnology, IEE Proceedings - Vol 152, Issue 1, 4 Feb. 2005, pp. 47 – 51.

[7] Delgado, A.: DNA chips as lookup tables for rule based systems. IEE Computing and Control Engineering Journal 2002, Vol. 13, No. 3, pp. 113-119.

[8] Yu, Z.; Mohammed, A.; Panahi, I.: A Review of three PWM Techniques, American Control Conference, 1997, Proceedings of the 1997. Vol. 1, 4-6 June 1997, pp. 257 – 261Khalil, Hassan K. Nonlinear Systems. Third Edition. Ed. Prentice Hall. 2002. ISBN 0-13-06-7389-7.