Uso del Sistema de la Colonia de Hormigas para Optimizar Circuitos Lógicos Combinatorios

Benito Mendoza García

MAESTRÍA EN INTELIGENCIA ARTIFICIAL
UNIVERSIDAD VERACRUZANA

Dr. Carlos A. Coello Coello

CINVESTAV-IPN
ASESOR

Dr. Manuel Martínez Morales

MAESTRÍA EN INTELIGENCIA ARTIFICIAL - UV
REVISOR

M.C. Carlos Eduardo Mariano Romero

INSTITUTO MEXICANO DE TECNOLOGÍA DEL AGUA
REVISOR

Mayo, 2001

Agradecimientos

Mi principal agradecimiento es para Laura, mi esposa, ya que gracias a su apoyo, comprensión y cariño he podido completar este proyecto.

Agradezco a mis padres a mis hermanos y a mis suegros por el apoyo que siempre nos han brindado.

Gracias a mi asesor el Dr. Carlos Coello Coello por su orientación, ayuda y por compartir conmigo su experiencia.

Agradezco infinitamente a Oliva por la ayuda que me brindó para el análisis estadístico y al Dr. Manuel Martínez Morales, por guiarme en el diseño experimental.

Agradezco a toda la gente de la MIA, administrativos, profesores y compañeros, por haberme brindado su apoyo y amistad, durante todo este tiempo.

Agradezco el apoyo recibido del CONACyT a través de una beca para cursar la Maestría en Inteligencia Artificial del LANIA y la Universidad Veracruzana.

Agradezco también al Laboratorio Nacional de Informática Avanzada (LANIA), por haberme facilitado el uso de sus instalaciones, para hacer algunas corridas y pruebas.

Asimismo, agradezco la beca terminal otorgada por el CONACyT para concluir mi tesis de maestría, a través del proyecto NSF-CONACyT titulado "Desarrollo de métricas para optimización multiobjetivo con aplicación en problemas de saneamiento ambiental" (Ref. 33000-A), cuyo responsable es el M.C. Carlos Eduardo Mariano Romero.

Índice

Introducción

Las técnicas inspiradas en la evolución natural como son los *Algoritmos Genéticos, la Programación Genética y la Programación Evolutiva*, han sido utilizadas ampliamente, para la optimización de tareas. De estas técnicas, las más conocida quizá sea el Algoritmo Genético(AG), siendo ésta una herramienta muy poderosa en ciertos dominios.

Existen algunas investigaciones relacionadas con el diseño de circuitos lógicos combinatorios utilizando AG's, así como programación genética. A esta área de estudio se le denomina "Hardware Evolutivo".

Una técnica heurística más reciente, es la de la *Colonia de Hormigas (ACO: Ant Colony Optimization)*, basada en el comportamiento de forrajeo de las hormigas. El primer sistema desarrollado en el área *ACO* es el algoritmo llamado *Ant System (AS)*, propuesto por Marco Dorigo. Las hormigas son capaces de encontrar el camino más corto desde el hormiguero a una fuente de comida y viceversa sin usar pistas visuales. Asimismo, son capaces de adaptarse a cambios en el ambiente. El medio por el que las hormigas logran esto es por rastreo de la feromona que ellas mismas depositan mientras caminan. El AS ha sido utilizado con gran éxito en problemas con características similares a las del problema del viajero. Sin embargo, hasta donde se pudo indagar, este trabajo constituye la primera propuesta de uso del AS para diseñar circuitos eléctricos combinatorios.

Este trabajo se deriva de la investigación previa realizada por el asesor de tesis en la cuál se ha efectuado optimización de circuitos lógicos combinatorios, utilizando AG's. Dicho trabajo está enfocado no sólo a la producción de circuitos funcionales, sino también optimizarlos de acuerdo a ciertas medidas, entre ellas el número de compuertas.

En el primer capítulo de este trabajo se mencionan algunos trabajos previos que involucran técnicas evolutivas para el diseño de circuitos lógicos, además se presenta una visión muy general de lo que es el Hardware Evolutivo y una taxonomía de acuerdo a algunas propiedades. Por otro lado se define el problema del diseño de circuitos lógicos y sus implicaciones.

El segundo capítulo habla de las características del AS, algunas de sus aplicaciones y de algunas de las últimas extensiones que se han hecho de este algoritmo. Para aplicar el AS al diseño de circuitos lógicos combinatorios, se tuvo que ajustar el modelo del problema de diseño de circuitos lógicos, como si fuese del tipo de problemas para los que el algoritmo AS fue diseñado y en los que ha logrado buenos resultados. En este, capítulo también se describe el tratamiento del problema para lograr el mapeo entre un problema y otro.

En el tercer capítulo, se explica paso a paso el proceso que lleva a cabo nuestra aplicación, para lograr encontrar una solución para un circuito dado. Es decir, como se eligen las compuertas, como se decide combinarlas y como se almacenan los rastros, para lograr obtener un circuito funcional.

En el cuarto capítulo, se muestran resultados obtenidos para algunos circuitos. Además se hace una comparación de dichos resultados con los obtenidos tanto por técnicas utilizadas por diseñadores humanos, como por otras técnicas basadas en la evolución.

En el quinto capítulo se presenta el diseño y análisis de un modelo experimental factorial, realizado con el objetivo de entender el efecto que tienen los valores de los parámetros requeridos por nuestro algoritmo, sobre su desempeño, tomando como referencia el porcentaje de convergencia y el tiempo necesario para concluir una corrida.

1 Diseño de Circuitos Lógicos Combinatorios

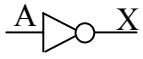
Hoy en día, la tecnología digital se aplica en un extenso rango de áreas: los sistemas telefónicos, sistemas de radar, sistemas de guía y navegación, sistemas militares, instrumentación médica, control de procesos industriales y por supuesto, en las computadoras. En éstas, la información la representan y procesan las redes de componentes electrónicos digitales, llamadas *circuitos lógicos.*

El amplio uso de dispositivos de tipo digital hace que el estudio de los circuitos digitales sea una de las ramas más importantes de la electrónica.

Un circuito lógico ejecuta una función booleana deseada, la cual está especificada por una tabla de verdad. Por lo tanto se puede decir que un circuito lógico es una implementación en hardware de una función booleana.

Los circuitos lógicos son construidos a partir de elementos primitivos llamados compuertas, combinándolos de formas innumerables. Las compuertas son dispositivos electrónicos que pueden ejecutar funciones que reciben y producen sólo valores booleanos (0 y 1). Las cinco compuertas básicas, principales bloques constructores de los circuitos, y sus tablas de verdad correspondientes se muestran en la figura 1.1.

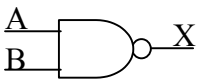
NOT



A	X
0	1
1	0

(a)

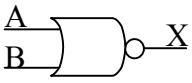
NAND



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

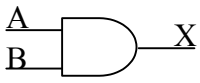
NOR



A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

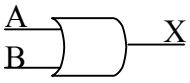
AND



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

OR



A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

Figura 1.1 Compuertas Lógicas Básicas

Muchas aplicaciones de lógica digital requieren un circuito con múltiples entradas y múltiples salidas en el cual las salidas son determinadas únicamente por las entradas actuales. Tales circuitos son llamados *Circuitos Lógicos Combinatorios*[32].

1.1 Antecedentes

Existen varios métodos tradicionales para diseñar circuitos lógicos, entre los que destacan los Mapas de Karnaugh[1], el Método de Quine-McCluskey[2], el proceso de simplificación algebraica[3] y las leyes De Morgan[4]. El primero es un método estándar ampliamente usado en las escuelas, en el que se utilizan gráficas para representar las funciones booleanas. Este método es manejable hasta con cinco o seis variables, pero en cuanto el número de variables aumenta, el uso de éste se vuelve una tarea muy compleja. Del método de Quine-McCluskey, se han tomado modelos para construir herramientas computacionales. Este método es útil para funciones de cualquier número de variables, pero los requerimientos de cómputo crecen exponencialmente con el número de entradas. El método de simplificación algebraica depende completamente de la familiaridad con los postulados, teoremas del álgebra y de la habilidad del diseñador para reconocer su aplicación, al igual que las leyes De Morgan. Sin embargo, el trabajo que se presenta aquí no se deriva de algún método tradicional, sino que utiliza técnicas de *Inteligencia Artificial*, específicamente, de computación evolutiva.

En general, las técnicas inspiradas en la evolución natural como son el *Algoritmo Genético (AG), la Programación Genética (PG) y la Programación Evolutiva (PE)* (éstas se describen en la siguiente sección), han sido utilizadas para la optimización de tareas. De estas técnicas, la más conocida quizá sea el Algoritmo Genético, siendo ésta una herramienta muy poderosa en ciertos dominios como la ingeniería, la investigación de operaciones, el reconocimiento de patrones, el procesamiento de imágenes, la biología, la física y ciencias sociales, entre otros[33]. Una técnica más reciente, es la *Colonia de Hormigas (ACO: Ant Colony Optimization)*[6], basada en el comportamiento de forrajeo de las hormigas. El primer sistema desarrollado en el área *ACO* es el algoritmo llamado *Ant System (AS)*, propuesto por Marco Dorigo[7].

Existen algunas investigaciones relacionadas con el diseño de circuitos lógicos utilizando AG's, pero no sabemos de investigaciones en esa misma rama que utilicen el AS. En este trabajo se presenta una herramienta basada en el AS, para diseñar circuitos lógicos optimizando el número de compuertas. Este trabajo es una extensión de la investigación realizada por Coello[8], en la cual se utiliza un AG para optimizar el diseño de circuitos lógicos. Más adelante se hablará de este trabajo en mayor detalle.

Coello[9] habla de trabajos previos en el área de diseño de circuitos que involucran técnicas evolutivas y menciona la tesis de Friedman[10], que data de mediados de los años 1950's, como el primer intento de utilizar procesos análogos a la selección natural. Atmar[11] fue uno de los primeros investigadores en utilizar una técnica basada en la evolución, incorporando directamente en el genotipo una cadena de bits que representa un circuito programable.

Uno de los pioneros en el uso de AG's para diseñar circuitos lógicos combinatorios es Louis[12], quien combina sistemas basados en el conocimiento y algoritmos genéticos, haciendo uso de un operador genético llamado *cruza enmascarada (masked crossover)*, la cual se adapta a la codificación, siendo capaz de explotar información normalmente no usada por los operadores clásicos de cruza. Aunque no resolvía por completo el problema del diseño de circuitos lógicos combinatorios, la idea de incorporar conocimiento acerca del dominio en el operador genético, constituye un gran paso hacia el incremento del poder del AG como una herramienta de diseño. Cabe mencionar, sin embargo, que el énfasis de Louis fue crear circuitos lógicos combinatorios funcionales, sin buscar su optimización.

Thompson[14] fue uno de los primeros en codificar compuertas lógicas básicas (AND, OR, NOT) y sus posibles interconexiones en hardware. Thompson y sus colaboradores se enfocaron al uso y configuración de *Arreglos de Compuertas con Campos Programables (Field Programable Gate Array FPGA)* usando algoritmos genéticos. Sus investigaciones fueron la base para trabajos posteriores de otros investigadores en hardware evolutivo a nivel de compuertas.

Miller[15], desarrolló una representación compacta en la que las entradas y las compuertas son elementos completamente separados en la cadena cromosómica, usando un simple gen para codificar una expresión booleana completa.

Los otros trabajos que reportados en la literatura que involucran hardware evolutivo a nivel de compuertas son el de Higuchi[16] y el de Iba[17]. En ambos es presentado un algoritmo genético con longitud variable cuya representación utiliza un arreglo. Sin embargo, el enfoque de esos trabajos es el aprendizaje y no la optimización.

El trabajo de Coello[8, 9] está enfocado no sólo a la producción de circuitos funcionales, sino también minimiza el número de compuertas utilizadas. Esta es una tarea muy complicada para el AG, ya que diseñar un circuito totalmente funcional a partir de un conjunto aleatorio de circuitos inválidos es un problema suficientemente difícil como para consumir el mayor tiempo de búsqueda de un AG convencional. En otras palabras, tratar de encontrar la región factible en ese espacio de búsqueda altamente restringido y luego intentar localizar el óptimo en tal región es una tarea difícil.

Otra área de la computación evolutiva, la programación genética, ha sido usada para el diseño de circuitos lógicos combinatorios; Koza[13] ha diseñado, por ejemplo, un sumador de dos bits, usando un conjunto pequeño de compuertas (AND, OR, NOT), pero su énfasis también ha sido el generar circuitos funcionales más que optimizarlos. En sus trabajos más recientes, Koza[13] se ha enfocado al diseño de circuitos analógicos en los que la meta es producir su propia topología y tamaño. Aunque la programación genética es una herramienta muy poderosa para algunas tareas como la programación automática, dada su representación, ésta produce circuitos altamente redundantes y difíciles de simplificar automáticamente. Además, los recursos normalmente requeridos para producir tales circuitos son muy altos en términos de memoria y tiempo de procesamiento o CPU.

1.2 Hardware Evolutivo

En la literatura contemporánea, el uso de técnicas basadas en la evolución para el diseño de circuitos ha sido llamado *Hardware Evolutivo (Evolvable Hardware Systems EHW)*.

Zebulum[5] propone una taxonomía del hardware evolutivo de acuerdo a cuatro propiedades, que son:

- Proceso de evaluación de hardware
- Enfoque de computación evolutiva
- Área de aplicación
- Plataforma evolutiva

Para visualizar mejor la taxonomía propuesta, la figura 1.2 presenta su estructura completa. En las siguientes secciones se explica cada una de las cuatro propiedades mencionadas.

Proceso de evaluación de hardware

Una posible forma de definir el Hardware Evolutivo es como una integración de dispositivos reconfigurables con aprendizaje genético. La propiedad del *proceso de evaluación de hardware* trata con el tipo de ambientes donde se evalúan cromosomas durante el aprendizaje genético. Se puede clasificar en dos ramas:

 EHW intrínseco

 EHW extrínseco

En *el enfoque EHW intrínseco*, la evolución se lleva a cabo en el dispositivo reconfigurable. Cromosomas o genotipos son introducidos en el dispositivo reconfigurable y evaluados usualmente a través de mecanismos de hardware. Hay dos ventajas en la utilización de este enfoque [18, 19, 20]: la primera es la velocidad del proceso de evolución y la segunda y más importante, el hecho de que no se necesita simulación para evaluar el circuito. Por lo tanto, en este caso, todas las soluciones se comportarán exactamente como lo harían en una aplicación en tiempo real [18]. Su posible desventaja sería el costo de los dispositivos reconfigurables, necesarios para la experimentación.

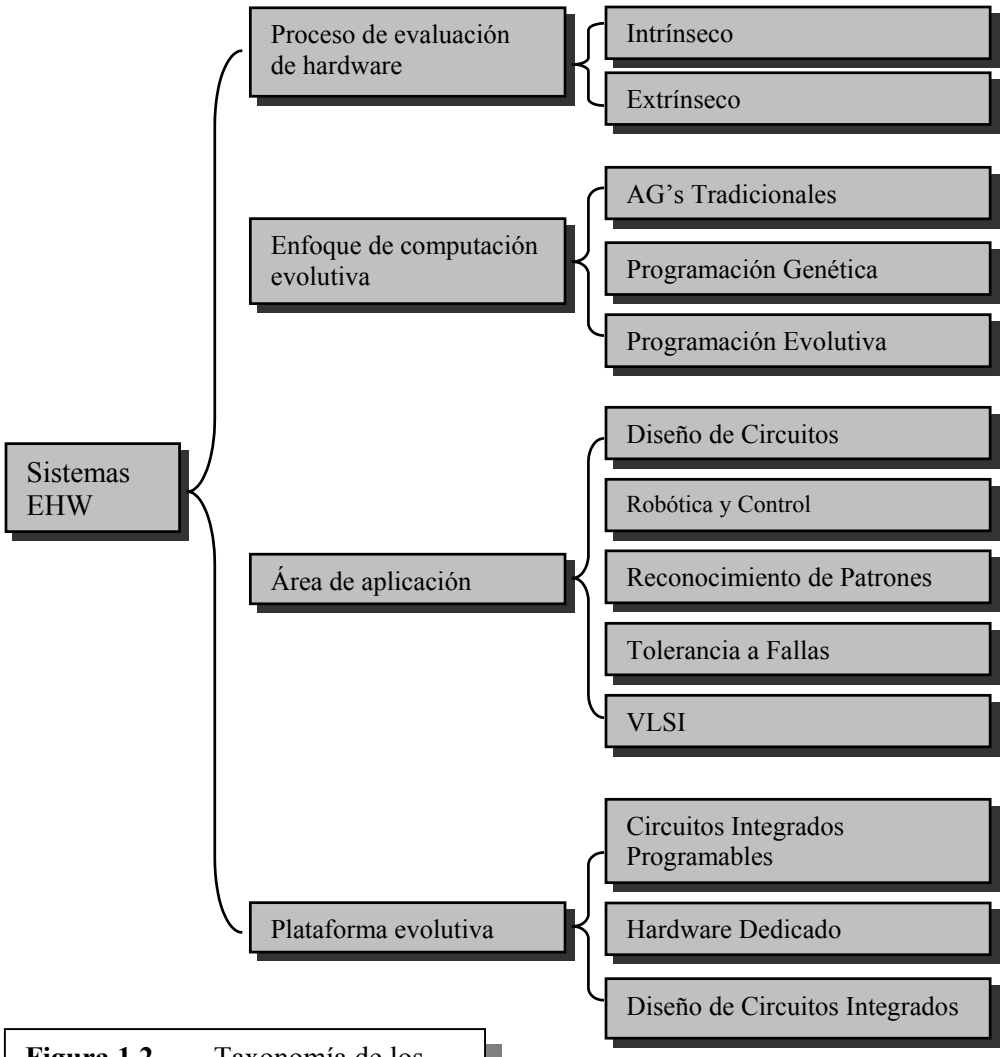


Figura 1.2 Taxonomía de los Sistemas EHW

El enfoque EHW extrínseco consiste en el uso de software para simular y evaluar los circuitos. Al final del proceso evolutivo, la solución encontrada (el mejor individuo) es introducida en el dispositivo reconfigurable. El enfoque extrínseco es más simple y en muchos casos, la única forma práctica de evaluar la aptitud de un número grande de individuos. Adicionalmente, este enfoque facilita la experimentación con nuevas teorías de Hardware Evolutivo.

Enfoque de computación evolutiva

El Hardware Evolutivo utiliza evolución artificial para diseñar circuitos, pero la forma en la que los investigadores implementan sus algoritmos evolutivos puede variar. Hay básicamente tres diferentes enfoques:

 Algoritmos genéticos tradicionales

 Programación genética

 Programación evolutiva

Los Algoritmos genéticos tradicionales (AGs) como fueron propuestos originalmente por John Holland [17] tienen las siguientes características básicas:

 Codificación binaria para formar cromosomas

 Cruza de un punto y mutación como operadores genéticos

 Reemplazamiento generacional de la población total con elitismo (seleccionando solo los mejores individuos)

La principal ventaja de la representación binaria es que ésta provee el número máximo de esquemas (bloques constructores) por unidad de información. Sin embargo, la representación binaria puede implicar longitudes muy largas de cromosomas en algunas aplicaciones.

En el método de *Programación genética (PG)*[21], los cromosomas son representados como árboles con ramas ordenadas, en las cuales los nodos internos son funciones y las hojas son símbolos terminales (variables y operandos). En este caso, la cruza consiste en intercambiar sub-árboles entre cromosomas [21].

El enfoque de la *Programación evolutiva (PE)* difiere de los Algoritmos Genéticos principalmente en la forma en que los cromosomas son codificados y en los operadores genéticos utilizados. En vez de usar representación binaria, en el enfoque PE se usan números enteros o reales para formar cromosomas. Adicionalmente, la PE utiliza operadores genéticos especiales, los cuales incorporan conocimiento específico del problema [22]. Este enfoque puede implicar longitudes más cortas de cromosomas y mejor desempeño de los sistemas evolutivos debido a los operadores genéticos especiales.

Además de los tres métodos básicos, hay otros enfoques alternativos de computación evolutiva, no tan comúnmente usados, como lo son los VGAs (Algoritmos genéticos con cromosomas de longitud variable) [16] y los PGAs (Algoritmos genéticos de producción) [23].

Área de aplicación

Hardware Evolutivo comprende una variedad de aplicaciones que pueden ser clasificadas de acuerdo a las siguientes categorías:

 Diseño de circuitos (digitales y analógicos)

 Control y robótica

 Reconocimiento de patrones

 Tolerancia a fallas

 VLSI

El diseño de circuitos electrónicos, tanto digitales como analógicos es la principal aplicación de la electrónica evolutiva. La disponibilidad de un gran rango de simuladores de circuitos para experimentos extrínsecos, así como dispositivos programables, como los FPGAs usados en sistemas EHW intrínsecos, ha permitido a los investigadores estudiar los sistemas de hardware evolutivo. Este es posiblemente, uno de los campos más prometedores en el área de EHW si consideramos el número de subsistemas en aplicaciones de *control* que pueden ser evolucionados, tal como las Máquinas de Estado Finito [24] y otros modelos básicos [25]. En particular, el área de *Robótica Evolutiva* [24], en la cual se usan técnicas evolutivas para desarrollar sistemas de control de robots [15], está también siendo ampliamente estudiada.

Las Redes Neuronales Artificiales han sido usadas cada vez más para aplicaciones *de reconocimiento de patrones* [26, 27]. Higuchi y colaboradores [25] proponen el uso de EHW en esta área, indicando las ventajas básicas de EHW en relación con Redes Neuronales. La tabla 1.1 resume la comparación entre EHW y Redes Neuronales hecha por Higuchi.

	EHW	**RNA**
Velocidad	Rápido	Lento
Adaptabilidad	En línea	Generalmente Fuera de Línea
Conversión del Hardware	Directa	Más Compleja

Tabla 1.1 EHW *vs.* Redes Neuronales Artificiales

Como se dice en el trabajo de Higuchi, las soluciones del Hardware Evolutivo son potencialmente más rápidas que las de Redes Neuronales porque la adaptación debida a cambios en el ambiente o fallas de hardware, puede realizarse en línea y el resultado de la adaptación es el nuevo hardware en sí [25]. La adaptación o entrenamiento de las Redes Neuronales es hasta ahora llevada a cabo principalmente en software.

Higuchi resalta en su trabajo que, dado que el resultado del proceso de aprendizaje en EHW es visible en términos de funciones Booleanas, éste puede ser fácilmente convertido a estructura de hardware. Por lo tanto, el mantenimiento de sistemas EHW sería más fácil que el de los sistemas de Redes Neuronales que son más difíciles de entender. Finalmente, Higuchi también dice en su trabajo que, tal como en las Redes Neuronales, EHW normalmente requiere mucho menos conocimiento de los procesos particulares que en el caso de sistemas convencionales.

Idealmente, las máquinas adaptativas pueden cambiar su estructura de hardware en respuesta a cambios ambientales, como mal funcionamiento. Por ejemplo, una computadora adaptativa ideal debería tener la propiedad de cambiar, si es necesario, su arquitectura en cada transición del reloj [28].

La forma más común de construir *sistemas de hardware tolerantes a fallas* es a través de la duplicación de componentes críticos, la cual es una solución poco económica y en algunos casos, no factible. Se puede pensar en un enfoque con menos ventajas para el problema de tolerancia a fallas usando el concepto de máquinas adaptativas [25].

EHW es un enfoque apropiado para sistemas de tolerancia a fallas, ya que los dispositivos reconfigurables pueden cambiarse a sí mismos a través de aprendizaje genético. En este caso, el proceso de aprendizaje genético consiste en buscar la mejor configuración de hardware dada una condición ambiental particular. Como menciona Higuchi [25], el resultado aprendido puede ser directamente traducido en un sistema de hardware nuevo, haciéndolo apropiado para aplicaciones reales.

Aunque hasta el momento hay pocos trabajos [29] de la aplicación de EHW en diseño *VLSI*[29], ésta es un área promisoria de aplicación futura de los sistemas evolutivos. En el diseño VLSI, los siguientes problemas son candidatos potenciales para aplicar el enfoque EHW:

 Diseño de circuitos (digitales y analógicos)

 Colocación ó emplazamiento (Placement)

 Ruteo

En el diseño de circuitos, las estructuras evolutivas pueden ser transistores, compuertas de transmisión o capas.

El emplazamiento y el ruteo son básicamente problemas de optimización combinatoria que pueden beneficiarse ampliamente de los enfoques evolutivos.

Plataforma evolutiva

Esto se refiere a la plataforma de hardware sobre la cual se lleva a cabo la evolución del circuito. Se pueden concebir tres clases principales de Plataformas Evolutivas:

 Circuitos integrados programables

 Hardware dedicado

 Diseño de circuitos integrados

Los circuitos integrados programables son la plataforma estándar utilizada en EHW. Estos pueden ser divididos en tres categorías [30]: *Memorias, Microprocesadores y Circuitos Lógicos.* EHW se enfoca principalmente en Circuitos Lógicos Programables. A su vez, los

circuitos lógicos pueden ser divididos en tres sub-categorías [30]: PLD (Programmable Logic Device), CPLD (Complex Programmable Logic Device) y FPGA (Field Programmable Gate Array). La figura 1.3 muestra esta jerarquía.

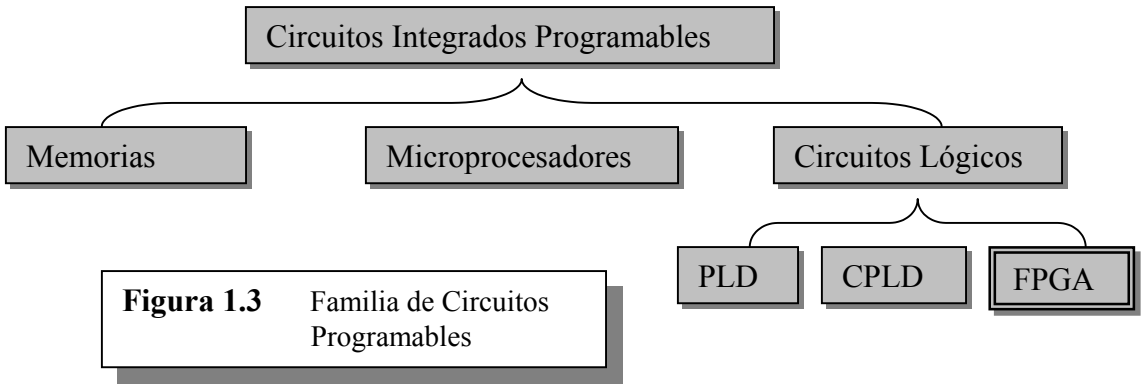


Figura 1.3 Familia de Circuitos Programables

Los circuitos PLD consisten en un arreglo de compuertas AND, el cual genera términos para las entradas del sistema y un arreglo de compuertas OR que genera la salida del sistema. De acuerdo a su grado de programabilidad, los circuitos PLD pueden ser clasificados en PROM (Programmable Read-Only Memory), PAL (Programmable Array Logic) y PLA (Programmable Logic Array) [30].

Un CPLD puede verse como una combinación de celdas programables que consiste normalmente de multiplexores o memorias y una red de interconexión que selecciona las entradas de las celdas programables.

Los FPGAs son los dispositivos reconfigurables más usados en EHW [18, 30]. Consisten en un arreglo de celdas lógicas y celdas I/O. Cada celda lógica consiste en una función universal [30] (multiplexor, demultiplexor y memoria) que puede ser programada para realizar una cierta función.

Los FPGAs de la familia XILINX [30] han sido ampliamente usados por los investigadores del área de EHW [18, 30]. Cada celda lógica o bloque funcional de un FPGA XILINX consiste en una tabla de búsqueda RAM en la cual se puede programar la tabla de verdad del circuito lógico combinatorio. Las interconexiones son controladas por los contenidos de un dispositivo RAM, lo que significa que esta familia de FPGAs puede ser reprogramada.

En vez de usar FPGAs, existe la posibilidad de usar una memoria como plataforma evolutiva. En este caso, los contenidos de la memoria serán programados genéticamente [19].

El proceso evolutivo puede ser dirigido para lograr una arquitectura dedicada. Por ejemplo, una arquitectura dedicada de Redes Neuronales [27] puede ser evolucionada usando neuronas artificiales como un elemento básico del proceso evolutivo. Los *Chips difusos (Fuzzy chips)* [31] son otra clase de *hardware dedicado* que es candidato al enfoque de diseño "Ingeniería Evolutiva". Tanto el enfoque intrínseco como el extrínseco pueden ser usados para este tipo de diseño.

En método de *diseño de circuitos integrados*, tanto capas de circuitos integrados (como metal, óxido y silicón) como dispositivos completos (como transistores o compuertas de transmisión) son manejados por el proceso evolutivo para crear un diseño de circuito completo. El diseño VLSI es un ejemplo de este enfoque. En este caso, el enfoque extrínseco es el único esquema viable para evaluar un número grande de circuitos.

1.3 Definición del problema

El diseño es en general una tarea no trivial cuya automatización puede considerarse muy compleja. Una definición formal de diseño es la siguiente[8]:

> "Diseño es el proceso de derivar a partir de una conducta dada (como entrada o como salida), una estructura que es funcional dentro de un cierto conjunto de restricciones dadas".

Aplicando esta definición al diseño de circuitos, la conducta corresponde a una función booleana (la que se desea ejecute el circuito) y la estructura a una cierta combinación de compuertas lógicas. Así mismo, un circuito se acepta como funcional si produce todas las salidas deseadas para todas las entradas dadas. Una restricción posible del problema puede ser el conjunto de compuertas lógicas disponibles. En función de lo anterior, insertando estas características en la definición tenemos que:

> *El diseño de circuitos es el proceso de derivar a partir de una función booleana (normalmente dada en una tabla de verdad), una combinación de compuertas lógicas que produce todas las salidas deseadas para todas las entradas dadas, limitando las compuertas lógicas disponibles a un conjunto predefinido.*

Pero además de buscar una estructura funcional, en el diseño de circuitos a menudo se busca una estructura óptima pues, dado que los circuitos digitales pueden ser construidos a partir de un número pequeño de elementos primitivos (compuertas) combinándolos de

formas innumerables, es posible generar circuitos de estructura diferente que implementan la misma función.

Una estructura óptima es aquella que reduce costos en comparación con las demás. Existen varios criterios para definir costos mínimos en diseño de circuitos. Puede que se desee minimizar, por ejemplo, el número total de literales, el número de operaciones binarias o bien, el número de compuertas.

Los diseñadores de circuitos a menudo tratan de reducir el número de compuertas en sus productos para reducir costos de producción (al reducir el número de componentes, el tamaño necesario del tablero, consumo de energía, etc.).

La complejidad de un circuito lógico es una función del número de compuertas del circuito. La complejidad de una compuerta es una función del número de entradas a ella. Para reducir la complejidad de un circuito, el diseñador debe encontrar otro circuito que implemente la misma función que el original, pero con menos compuertas o con compuertas más simples (por ejemplo, compuertas de una entrada en vez de compuertas de dos entradas).

Finalmente, añadiendo la característica de optimalidad, la definición del problema queda de la siguiente forma:

> *El diseño de circuitos es el proceso de derivar a partir de una función booleana (normalmente dada en una tabla de verdad), una combinación de compuertas lógicas que produce todas las salidas deseadas para todas las entradas dadas y optimiza ciertas características estructurales, limitando las compuertas lógicas disponibles a un conjunto predefinido.*

Resumen

El diseño de circuitos lógicos combinatorios ha sido en los últimos años una de la áreas de mayor interés dentro de la electrónica, debido a lo complejo que resulta su diseño y especialmente su simplificación. Durante mucho tiempo se ha utilizado el álgebra booleana, los mapas de Karnaugh y el método de Quine-McCluskey. En años recientes distintas técnicas de búsqueda, optimización y aprendizaje, cuya inspiración radica en las teorías de la evolución y la selección natural, han sido aplicadas para resolver este tipo de problemas. A estas técnicas se les conoce como Computación Evolutiva.

Las técnicas evolutivas han comenzado a incursionar en el diseño de circuitos debido principalmente a su poder exploratorio, ya que permiten evaluar diversas regiones de un espacio de diseño y encontrar varias soluciones a problemas complejos con relativa eficiencia. Actualmente a esta área se le conoce Hardware Evolutivo.

El Ant System es una técnica reciente basada en el comportamiento de forrajeo de las hormigas y con la cuál se construyó la aplicación presentada aquí. En el próximo capítulo se describirán los detalles del Ant System, su inspiración biológica y se hablará acerca de la forma en que se trató para su implementación en el problema de diseñar circuitos lógicos combinatorios, minimizando el número de compuertas.

Referencias Bibliográficas

[1] Karnaugh M.. **A map method for synthesis of combinational logic circuits**. *Transactions of the AIEE, Communications and Electronics*, 72 (I):593--599, November 1953.

[2] McCluskey E. J.. **Minimization of boolean functions**. *Bell Systems Technical Journal*, 35 (5):1417--1444, November 1956.

[3] Frausto Solis Juan, **Lógica Proposicional: *Una Presentación De Bolsillo***, http://w3.mor.itesm.mx/~logica/log9808/artlp2.html, jfrausto@campus.mor.itesm.mx

[4] Stallings William. **Computer Organization and Architecture**. *Prentice Hall, Fourth Edition*, ISBN 0-13-359985-X

[5] Zebulum Ricardo, Pacheco Marco y Vellasco Marley, **Evolvable Systems in Hardware Design: *Taxonomy, Survey and Applications***, *Proceedings of The First International Conference on Evolvable Systems: From Biology to Hardware (ICES'96)*, Lecture Notes in Computer Science 1259, pp. 344-358, Tsukuba, Japan, October 7-8 1996.

[6] Dorigo Marco, **Ant Colony Optimization**, http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html, IRIDIA, Université Libre de Bruxelles, Belgium.

[7] Dorigo M., Maniezzo V. y Colorni A, **The Ant System: *Optimization by a Colony of Cooperating Agents***. *IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):29-41*, 1996

[8] Coello Coello, Carlos A.; Christiansen, Alan D. and Hernández Aguirre, Arturo, **Use of Evolutionary Techniques to Automate the Design of Combinational Circuits**, *International Journal of Smart Engineering System Design, Elsevier Science*, Vol. 2, No. 4, pp. 299-314, June 2000

[9] Coello Coello, Carlos A.; Alan D. Christiansen & Arturo Hernández Aguirre, **Towards Automated Evolutionary Design of Combinational Circuits**, Lania-RI-99-03.d, Laboratorio Nacional de Informática Avanzada, 1999.

[10] Friedman George J.. **Selective Feedback Computers for Engineering Synthesis and Nervous System Analogy**. *Master's thesis, University of California at Los Angeles*, February 1956.

[11] Atmar J. Wirt. **Speculation on the Evolution of Intelligence and Its Possible Realization in Machine Form**. *PhD thesis, New Mexico State University*, Las Cruces, New Mexico, 1976.

[12] Sushil J. Louis and Gregory J. Rawlins. **Using Genetic Algorithms to Design Structures**. *Technical Report 326, Computer Science Department, Indiana University*, Bloomington, Indiana, February 1991.

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

[14] Thompson Adrian, I. Harvey, and Husbands Philip**. Unconstrained evolution and hard consequences**. *In E. Sanchez and M. Tomassini, editors, Toward Evolvable Hardware: The Evolutionary Engineering Approach (Lecture Notes in Computer Science, Vol. 1062)*, pages 136--165, Heidelberg, Germany, 1996. Springer-Verlag.

[15] Miller J. F., Thomson P., and *Fogarty* T.. **Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: *A Case Study***. *In D. Quagliarella, J. P'eriaux, C. Poloni, and G. Winter, editors, Genetic Algorithms and Evolution Strategy in Engineering and Computer Science*, pages 105--131. Morgan Kaufmann, Chichester, England, 1998.

[16] Higuchi Tetsuya, Iwata Masaya, Kaijitani Isamu, Murakawa Masahiro, Yoshizawa Shuji, and Furuya Tatsumi. **Hardware evolution at gate and function level**. *In Proceedings of the International Conference on Biologically Inspired Autonomous Systems: Computation, Cognition and Action*, Durham, North Carolina, March 1996.

[17] Hitoshi Iba, Masaya Iwata, and Tetsuya Higuchi. **Gate-Level Evolvable Hardware: *Empirical Study and Application***. *In Dipankar Dasgupta and Zbigniew Michalewicz, editors*, volutionary Algorithms in Engineering Applications, pages 260--275. Springer-Verlag, Berlin, 1997.

[18] Thompson Adrian, **Silicon Evolution**, *Proceedings of Genetic Programing*, MIT press, 1996.

[19] Thompson Adrian, **Evolving Electronic Robot Controllers That Exploit Hardware Resources***, Proceedings of the 3rd. European Conference on Artificial Life*, Springer Verlag, 1995.

[20] Thompson Adrian, Harvey Inman, Husbands Philip, **Unconstrained Evolution and Hard Consequences***, Lecture Notes in Computer Science-Toward Evolvable Hardware*, Vol. 1062, pp 136-165 Springer-Verlag, 1996.

[21] Tomasini M., **Evolutionary Algorithms***, Lecture Notes in Computer Science-Toward Evolvable Hardware*, Vol. 1062, pp 19-47 Springer-Verlag, 1996.

[22] Michalewicz Z., **Genetic Algorithms + Data structures = Evolution Programs***, Springer-Verlag*, 1994.

[23] Mizoguchi Jun'ichi, Hemmi Hitoshi and Shimohara Katsunori, **Production Genetic Algorithms for Automated Hardware Design through an Evolutionary Process***, IEEE Conference on Evolutionary Computation*, 1994.

[24] Harvey I., Husbands P. and Cliff D., **Genetic Convergence in as Species Evolved Robot Control Architectures***, S. Forrest, editor, Proc. Of 5th Int. Conf. On Genetic Algorithms*; p636, Morgan Kaufmann, 1993.

[25] Higuchi T., Iwata M., Kajitani I., Iba H., Hirao Y., Furuya T., Manderick B., **Evolvable Hardware and its Applications to Pattern Recognition and Fault-Tolerant Systems***, Lecture Notes in Computer Science-Toward Evolvable Hardware*, Vol. 1062, pp 118-135 Springer-Verlag, 1996.

[26] Perelmuter G., Carrera E., Vellasco M., Pacheco M**.,** ***Recognition of Industrial Parts Using Artificial Neural Networks****, Proceedings of EPMESC V-Education, Practice and Promotion of Computational Methods in Engineering Using Small Computers*, pp. 481-486, Macao, August 1995.

[27] Treleaven P., Pacehco M. A., Vellasco M., **VLSI architectures for neural networks***, IEEE micro*, v. 9, n. 6, p. 8-27, 1989.

[28] De Garis Hugo, **Evolvable Hardware Worshop Resort***, Technical Report, ART Human Information Processing Research Laboratories*, Evolutionary Systems Department, Kyoto, Japan, 1996.

[29] Blickle Tobias, Teich Jurgen, Thiele Lothar, **Systems-Level Synthesis Using Evolutionary Algorithms**, *Computer Engineering and Communication Networks Lab (TIK)*, Swiss Federal Institute of Technology (ETH), TIK-Report, Nr. 16, April, 1996.

[30] Sanchez E., **Field Programmable Gate Array (FPGA) Circuits***, Lecture Notes in Computer Science-Toward Evolvable Hardware*, Vol. 1062, pp 1-18 Springer-Verlag, 1996.

[31] Ungering A. P., Bauer H., Goser K., **Architecture of a Fuzzy-Processor based on an 8-bit Microprocessor***, IEEE*, pp. 297-301, 0-7803-1869-X/94, 1994.

[32] Tanenbaum Andrew S.. **Structured Computer Organization**. *Fourth Edition, Prentice Hall*, ISBN 0-13-095990-1.

[33] Goldberg David E., **Genetic Algorithms in Search, Optimization, and Machine Learning**, Addison-Wesly Publishing Company, inc. ISBN 0-201-15767-5.

2 El Ant System

Ant System (AS) es el primer algoritmo desarrollado en el área *ACO: Ant Colony Optimization.* En ésta se estudian sistemas artificiales que simulan colonias de hormigas reales, de donde toman su inspiración. Estos sistemas son utilizados para resolver problemas de optimización combinatoria, los cuales pueden ser descritos como problemas cuyo objetivo es encontrar la secuencia óptima de sus elementos componentes. Por ejemplo: el problema del vendedor viajero, planeación de horarios y ruteo en redes. El algoritmo *Ant System*, desarrollado por Marco Dorigo[1][2] es utilizado para resolver este tipo de problemas. Su principal aplicación y en la que además ha logrado muy buenos resultados es en el primero de estos problemas, por lo que a continuación se describe en detalle:

> *En el problema del vendedor viajero, éste tiene la tarea de visitar un número n de clientes que viven en diferentes ciudades. El problema es encontrar el orden en que las ciudades deben ser visitadas de modo que se minimice la distancia recorrida al final del viaje, tomando en cuenta que sólo se puede pasar una vez por cada ciudad[3].*

La figura 2.1 representa mediante un grafo la idea del problema del vendedor viajero, también conocido como *TSP* por sus siglas en inglés (*Traveling Salesman Problem*). En dicho grafo los círculos (vértices) representan las ciudades y las líneas que los unen (ejes) representan las vías de comunicación.

En este capítulo, después de introducir la idea básica del comportamiento real de las hormigas, se describirá cómo funciona el algoritmo AS y finalmente se explicará cómo se aplicó al diseño de circuitos lógicos, que aunque es un problema de optimización combinatoria, no es un problema de tipo *vendedor viajero,* razón por la cual no se había aplicado el algoritmo AS en él.

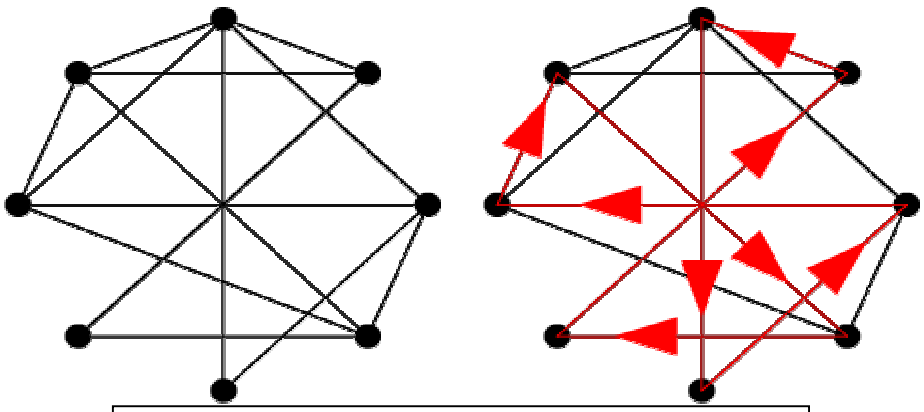


Figura 2.1 Representación Gráfica del TSP[4]

2.1 Inspiración Biológica

Las hormigas son capaces de encontrar el camino más corto desde el hormiguero a una fuente de comida y viceversa sin usar pistas visuales. Asimismo, son capaces de adaptarse a cambios en el ambiente. Por ejemplo, que un obstáculo sea colocado en la ruta que están utilizando como la más corta, como se ilustra en los cuadros 3, 4 y 5 de la figura 2.2. El medio por el que las hormigas logran esto es por rastreo de la feromona que ellas mismas depositan mientras caminan.

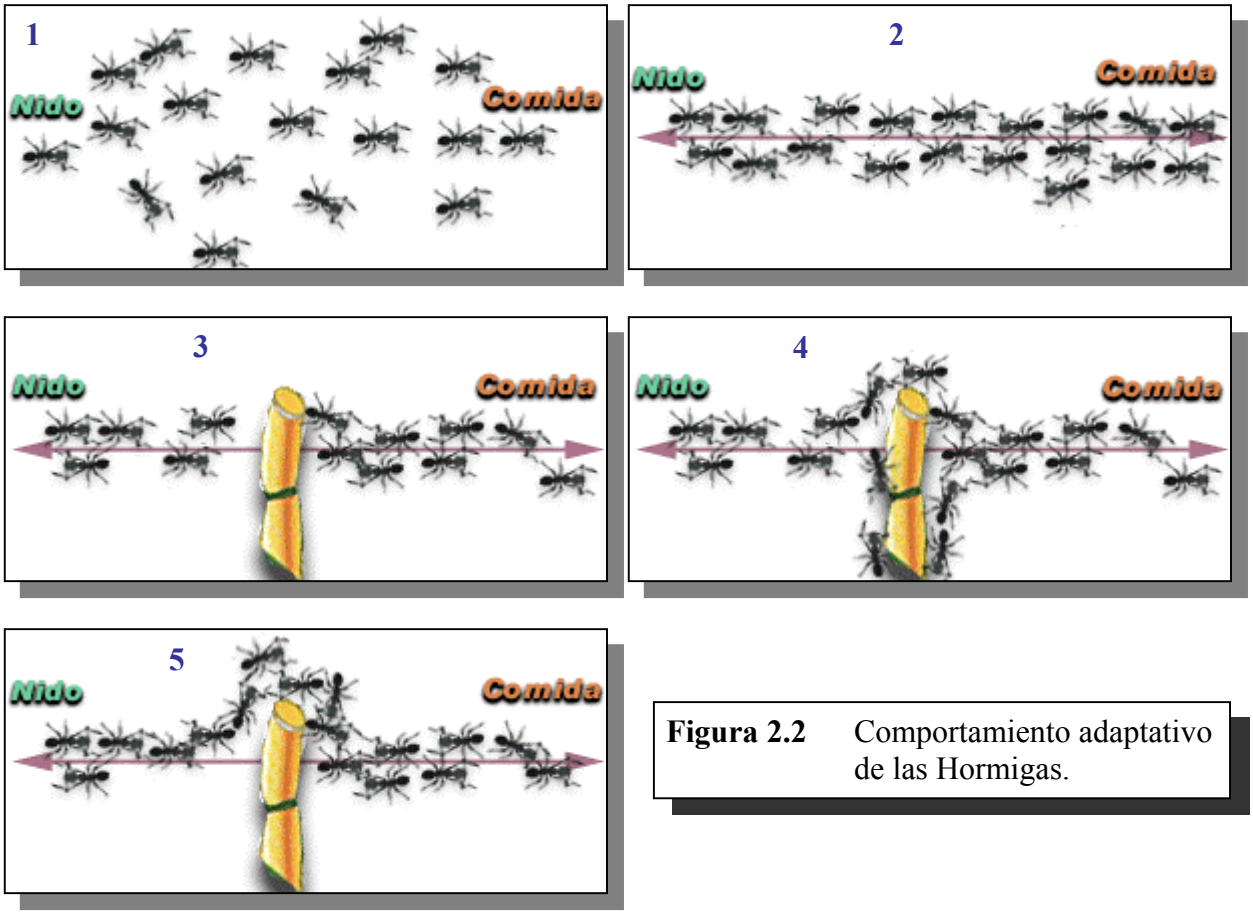


Figura 2.2 Comportamiento adaptativo de las Hormigas.

Todas las hormigas depositan una cierta cantidad de una substancia llamada feromona mientras caminan y a su vez, cada hormiga prefiere caminar en una dirección rica en feromona. Esta simple conducta de las hormigas explica porqué son capaces de ajustarse a cambios en el ambiente. Cuando un obstáculo inesperado es colocado en el camino que las hormigas están utilizando (cuadro 3 de la figura 2.2), las hormigas que están justo enfrente del obstáculo no pueden continuar siguiendo el rastro de feromona y por lo tanto, deben elegir sobre irse hacia la izquierda o hacia la derecha. La elección es aleatoria, es decir, cada hormiga decide al azar hacia donde irse, pero se esperar que aproximadamente la

mitad de las hormigas intente evadir el obstáculo por un lado y la otra mitad lo haga por el otro (cuadro 4 de la figura 2.2). De esta forma, las hormigas que eligieron (aleatoriamente) el camino corto, crearán en un cierto tiempo un depósito de feromona más fuerte que el de las hormigas que eligieron el camino largo. De tal forma, pasarán más hormigas por el camino corto (debido a que llegan al otro lado más rápido que las otras), quedando depositada, por lo tanto, más feromona en esa ruta, lo que origina que las hormigas que vienen atrás, prefieran caminar por ella, reestableciéndose así, el camino más corto (cuadro 5 de la figura 2.2) [5].

Resulta obvio que encontrar la ruta más corta es una conducta que parece ser emergente de la interacción entre el obstáculo y la conducta distribuida de las hormigas, aún cuando todas las hormigas caminan aproximadamente a la misma velocidad y depositan también, aproximadamente, la misma cantidad de feromona.

2.2 El Algoritmo

El algoritmo *Ant System* es un nuevo algoritmo de búsqueda cooperativo inspirado por la conducta de las hormigas reales. La conducta de las colonias de hormigas es imitada por el algoritmo AS usando agentes sencillos, llamados *hormigas* (*ants*), que se comunican indirectamente por medio de un mecanismo inspirado en el rastro de feromona. Los rastros de feromona artificial, son un tipo de información numérica distribuida que es modificada por las hormigas y refleja su experiencia en la solución de un problema en particular.

Hay tres principales ideas que el algoritmo de la colonia de hormigas ha adoptado de las colonias reales de hormigas:


Se utiliza comunicación indirecta a través de la feromona.


Las rutas más cortas tienden a tener una razón más alta de crecimiento del valor de la feromona.


Las hormigas tienen preferencia probabilística por las rutas con valores altos de feromona.

Además de estas características, se les ha dado a los agentes (hormigas) capacidades que no tienen las hormigas reales, pero que ayudan a resolver los problemas. Por ejemplo:


Cada hormiga es capaz de determinar qué tan lejos está de un estado.

Poseen información acerca de su ambiente y la utilizan al tomar decisiones. Así, su comportamiento no sólo es adaptativo sino también *"avaricioso" (greedy[1]).*

Tienen memoria, la cual es necesaria para asegurar que se generen sólo soluciones factibles (por ejemplo, en el TSP, necesitan saber las ciudades que han visitado pues una ciudad no se puede visitar más de una vez).

Descripción

Dado un problema *TSP* de n ciudades y distancias d_{ij} entre cada par de ciudades i y j:

1. Las hormigas artificiales se distribuyen inicialmente en las n ciudades de acuerdo a algún criterio (aleatoriamente, por ejemplo).

2. Posteriormente, cada hormiga decide la ciudad a visitar en cada paso de un ciclo que se repite hasta que todas las ciudades son visitadas exactamente una vez por cada hormiga. La decisión de qué ciudad visitar, se toma con base en la siguiente expresión, donde p_{ij} denota la probabilidad de que una ciudad j sea seleccionada para visitarse después de la ciudad i:

$$p_{ij} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}\,[\eta_{ij}]^{\beta}}{\sum_{h \in \Omega}[\tau_{ih}]^{\alpha}\,[\eta_{ih}]^{\beta}} & \text{si} \quad j \in \Omega \\ 0 & \text{en otro caso} \end{cases}$$

Siendo:

$$\eta_{ih} = \frac{1}{d_{ij}}$$

donde

 τ_{ij} cantidad de feromona entre las ciudades i y j

 α parámetro para regular la influencia de τ_{ij}

[1] El Greedy es un algoritmo aprendizaje por reforzamiento muy conocido, en el cual siempre se escoge la acción cuya recompensa estimada sea la más alta[21].

 η_{ij} visibilidad de la ciudad `j` desde la ciudad `i`

 β parámetro para regular la influencia de η_{ij}

 Ω conjunto de ciudades que aún no han sido visitadas

 d_{ij} distancia entre las ciudades `i` y `j`

3. Cuando se termina el ciclo (cada hormiga ha concluido su recorrido) se calcula la longitud del recorrido generado por cada hormiga y se actualiza el mejor recorrido encontrado hasta el momento.

4. Finalmente, se actualizan las cantidades de feromona. Esto se hace con la siguiente fórmula:

$$\tau_{ij} = p\,\tau_{ij} + \Delta\tau_{ij}$$

y

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{si la hormiga } k \text{ viajó por el eje } (i,j) \\ 0 & \text{en otro caso} \end{cases}$$

siendo:

 $p \in [0, 1]$ es un parámetro para regular la reducción de τ_{ij}

 $\Delta\tau_{ij}$ es el incremento en la cantidad de feromona en el eje `(i, j)`

m es el número de hormigas

$\Delta\tau^k_{ij}$ es el incremento en la cantidad de feromona en el eje `(i, j)` realizado por la hormiga `k`

Q es un valor constante que representa la cantidad de feromona depositada por una hormiga en cada recorrido

L_k es la longitud del recorrido de la hormiga `k`

En cada recorrido, cada hormiga deja una cantidad de feromona dada por Q/L_k, donde `Q` es una constante y L_k la longitud de su recorrido. Por lo tanto, en recorridos más cortos se deposita más feromona. En el algoritmo se simula la evaporación de feromona que sucede en la realidad. Las cantidades de feromona se reducen con un factor `(1-p)` antes de que se deposite nueva feromona. Esto se hace para evitar convergencia prematura.

Los pasos anteriores se repiten *t* veces, donde *t* es el número de iteraciones[9].

Pseudocódigo

El pseudocódigo del algoritmo AS original es el siguiente[9]:

```
Inicio
For t = 1 to Max_Iter do // Max_Iter es el número de iteraciones
   For k = 1 to m do //m es número de hormigas (agentes)
      Repetir hasta que la hormiga k complete su recorrido
          Seleccionar la siguiente ciudad que se va a visitar
      Calcular la longitud del recorrido de la hormiga k
   Actualizar los niveles de feromona
Fin
```

2.3 Algunas Aplicaciones

Las aplicaciones que hay hasta el momento del AS son en problemas que pueden verse de una manera muy similar al del vendedor viajero, pues el algoritmo fue creado precisamente para solucionar problemas de ese tipo. De hecho, Marco Dorigo, comenta en su página

Web[6]: "Para hacer un buen uso de la metodología de la colonia de hormigas, debemos poder reformular nuestro problema como un tipo de buscador de rutas en un grafo, e identificar una forma de definir las distancias entre nodos". Obviamente esto limita sus áreas de aplicación. Algunos ejemplos de trabajos realizados con AS son:

MOAQ (Mariano & Morales)[7], un algoritmo para la optimización de redes de irrigación, visto como un problema multiobjetivo, donde las ciudades se traducen como tomas de agua y la distancia entre ciudades es la longitud de la tubería necesaria para conectar las tomas.

Un enfoque ACO para el problema del flujo de procesos o FSP: Flow Shop Problem (Thomas Stützle)[8], donde el problema consiste en encontrar un orden para ejecutar *n* procesos en *m* máquinas, de manera que se minimice el tiempo de ejecución de los mismos. De esta manera el FSP es como el problema del vendedor viajero pues es un problema, en el cual se busca una permutación de enteros {1,...,n} que minimice alguna función objetivo.

Ruteo de vehículos (Bullnheimer et al.)[9]: donde un número x de vehículos repartidores tienen que salir de un almacén y distribuir la mercancía a n clientes y regresar al almacén. Cada cliente se debe visitar una sola vez por un solo vehículo. El objetivo es minimizar los costos o distancias de las rutas y lograr un reparto eficiente de la mercancía. Este problema y el TSP están muy relacionados.

Desde el trabajo de AS de Marco Dorigo, se han propuesto varias extensiones al algoritmo básico, entre las que destacan:

El algoritmo Ant-Q (Gambardella & Dorigo)[10]: es un híbrido del algoritmo AS con aprendizaje-Q (*Q-learning*) [11][12], un conocido algoritmo de aprendizaje por reforzamiento.

Ant Colony System: ACS (Dorigo and Gambardella, 1997) [13], el cual es una extensión del Ant-Q.

Nuevas versiones de híbridos de Ant Colony Optimization con búsquedas locales (Stützle and Dorigo 1998) [14].

MAX-MIN Ant System (Stützle and Hoos, 1998) [15]: sólo se permite actualizar los rastros de feromona a la mejor hormiga en cada ciclo y se establecen valores máximos y mínimos de acumulación de feromona.

Rank-based version of Ant System (Bullnheimer et al., 1997) [16] varía de la versión original del AS en la forma de calcular el incremento en las cantidades de feromona depositadas por las hormigas entre las ciudades: para actualizar la cantidad de feromona entre un par de ciudades *i* y *j* se suma la cantidad de feromona depositada entre dichas ciudades por las *n* mejores hormigas (si es que pasaron por ahí) y no la de todas las hormigas que pasaron por ahí. Además, se realiza un incremento adicional si la mejor hormiga viajó por esa ruta.

ASGA un híbrido entre AS y un GA (Tony White, Bernard Pagurek y Franz Oppacher) [17] en el cual se utiliza el AG para adaptar los parámetros del AS (número de iteraciones, número de agentes, sensibilidad a la feromona, sensibilidad al costo de la ruta entre un estado y otro), de tal forma que la búsqueda se va adaptando de acuerdo a los resultados.

Aquí en particular, se implementó el algoritmo AS básico aplicado a la optimización de circuitos lógicos.

2.4 Aplicación del AS al Diseño de Circuitos Lógicos Combinatorios

Para aplicar este algoritmo al diseño de circuitos lógicos, se tuvo que modelar el diseño de circuitos lógicos como el tipo de problemas para los que el algoritmo AS fue diseñado y en los que ha logrado buenos resultados. Estos problemas son todos aquellos que pueden modelarse directamente como el problema del vendedor viajero, como ya se mencionó.

En el caso de los circuitos lógicos, el tratamiento del problema no parece ser tan inmediato. No está claramente definido el mapeo entre un problema y otro. No hay una forma inmediata de identificar lo que correspondería a un estado (o ciudad) y cómo se mediría la distancia entre estados (ciudades).

Representación de un circuito

Para representar un circuito utilizamos una matriz bidimensional (figura 2.3). Cada elemento de la matriz es un triplete del tipo [Entrada 1, Entrada 2, Tipo de Compuerta]. Podemos utilizar cinco tipos de compuertas: AND, OR, NOT, XOR y WIRE, aunque este último no es una compuerta, sino más bien es una conexión (un alambre) que une un elemento de cierta columna con otro de la columna anterior. Cada elemento de la matriz recibe sus entradas únicamente de las salidas de la columna anterior. La primera columna

recibe sus entradas directamente de la tabla de verdad del circuito dado. La última columna es la que proporciona las salidas del circuito. Las primeras *N* filas corresponden a las *N* salidas del circuito. Esta forma de representar un circuito ha sido utilizada con mucho éxito.

Las primeras aplicaciones que se hicieron de este algoritmo, sólo se trabajaron con circuitos de una sola salida[20][23].

Como se dijo en el capítulo 1, solo se utilizan compuertas de dos entradas. En el caso del NOT y del WIRE, para fines prácticos asumimos que solo se utiliza la entrada 1.

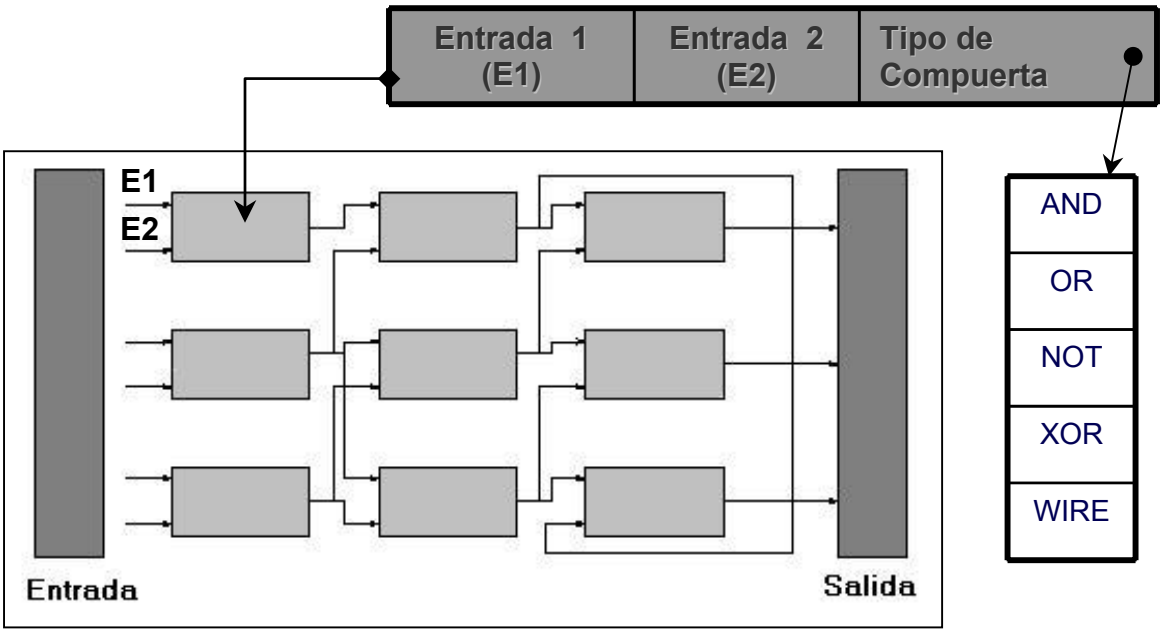


Figura 2.3 Matriz utilizada para la representación de un circuito.

La implementación

Como ya se mencionó, se trató modelar nuestro problema como si fuese del tipo *TSP*. Sin embargo varios elementos del *TSP* no fueron encontrados y tratados de la misma forma. Algunas correspondencias que se lograron son las siguientes:

La ruta de una hormiga o agente será un circuito completo. Es decir, cada hormiga recorre una ruta, y al hacerlo construye un circuito. En el *TSP* las hormigas encuentran la ruta más económica en términos de distancia, aquí lo hacen en términos del número de compuertas.

Un *estado o ciudad* es una columna, el cuál está compuesto por varios elementos a los que se les llama subestados, siendo estos cada una de las compuertas de una columna y el número de combinaciones de posibles entradas de cada compuerta de dicha columna. Los primeros *N* subestados (*N* es el número de salidas del circuito) se eligen con un factor *P* de selección (del cual se hablará más adelante) y las demás se eligen aleatoriamente.

La *distancia* (entre ciudades o estados) está medida como el aumento (o disminución) de los aciertos a las salidas del circuito al cambiar de un nivel a otro.

A diferencia del problema del viajero, en una misma ruta (circuito), *no tienen que visitarse todos los estados*.

Las feromonas se guardan en un arreglo llamado *Trails*. La longitud de este arreglo corresponde al número de salidas del circuito. Cada elemento de *Trails* es a su vez un arreglo tridimensional. A continuación se explica lo que representan cada una de las dimensiones del elemento.

La primera dimensión de este arreglo corresponde a la combinación de *posibles entradas* a la compuerta y va de 0 a 6. Las posibles combinaciones de entradas, independientemente del número de entradas de la tabla de verdad, serán siempre las que se presentan en la tabla 2.1.

No. de combinación	Posibles entradas a la compuerta
0	La compuerta recibe siempre 0 en sus dos entradas
1	La compuerta recibe 0 y 1, o bien, 1 y 0 como entradas
2	La compuerta recibe siempre 1 en sus dos entradas
3	Combinaciones 0 y 1
4	Combinaciones 0 y 2
5	Combinaciones 1 y 2
6	Combinaciones 0, 1 y 2 (Puede recibir cualquier cosa)

Tabla 2.1 Combinaciones de posibles entradas a una compuerta.

La segunda dimensión corresponde al número de compuerta, es decir, va de 0 al número de compuertas menos uno (NumGates-1).

La tercera dimensión corresponde al número de aciertos que se llevan hasta el nivel (columna) anterior y va de 0 al número de renglones en la tabla de verdad, pues el número de aciertos que se pueden tener en cualquier nivel está entre 0 y el número de renglones de la tabla de verdad.

La construcción de una solución (ruta)

Como ya se mencionó antes, un estado es una columna de la matriz, cada elemento de la columna es una compuerta con sus respectivas entradas y su salida. Por lo anterior, la elección de un estado es un proceso que se hace por partes (compuerta por compuerta), por lo que llamaremos a cada compuerta (elemento de la columna) un *subestado*. Un subestado es una combinación de tres elementos (compuerta, entrada1, entrada2). Para elegir un subestado de cualquiera de las primeras *N* filas, se le asigna a cada una de las combinaciones posibles un valor, llamado *factor P de selección*, con el que competirá por quedarse en esa posición. Dicho valor está dado por la siguiente fórmula:

$$p(k_{i,j,l}(t)) = f_{i,j,l}(t) * h_{i,j,l}$$

donde:

k es la hormiga o agente el cuál está construyendo la ruta o circuito.

$f_{i,l}$ es la cantidad de feromona que hay en el subestado ***i***, de la fila ***l***.

$h_{i,j,l}$ es la distancia entre los subestados ***i*** y ***j*** en la fila ***l***. La distancia es un valor heurístico y está dada por el número de aciertos que la porción del circuito construido hasta el momento produzca con respecto a la salida ***l*** de la tabla de verdad dada. Esto es análogo a la distancia en el TSP.

Una vez ha asignado un factor de selección a todas las combinaciones, se elige cual de ellas se queda en la posición en juego. Para esto se hace una selección por ruleta[2][22]. Esto se

[2] El método de selección por ruleta es una técnica de selección proporcional y como su nombre lo indica, la probabilidad de que un individuo sea elegido está dada por la proporción de su aptitud con respecto a la suma de la de todos los individuos [22].

repite con todos los subestados que pertenezcan a una de las filas que representen una salida del circuito. Los demás subestados, se eligen aleatoriamente. Esto se repite hasta llegar al último estado del circuito o columna de la matriz.

Cuando todas las hormigas terminan su recorrido, se actualizan los rastros de feromona. Esto se hace en dos pasos:

1. Primero se debe actualizar la cantidad feromona en los rastros, simulando la evaporación de los rastros dejados por las hormigas reales al paso del tiempo, esto se hace con la siguiente fórmula:

$$f_{i,j,l}(t+1) = (1-\alpha) \times f_{i,j,l}(t) + \sum_{k=1}^{m} f_{i,j,l}(t)$$

donde:

Donde α es un valor entre 0 y 1, llamado factor de evaporación.

m se refiere al número de agentes.

$\sum_{k=1}^{m} f_{i,j,l}(t)$ Corresponde a la cantidad total de feromona depositada por todas las hormigas que pasaron por el subestado *(i,j,l)*.

2. Después, los rastros se deben actualizar o incrementar de acuerdo a las rutas construidas por cada hormiga. Esto se hace de la siguiente forma:

Si el circuito resultado de la ruta no es válido (que no produzca todas las salidas).

$$f_{i,j,l}^{k} = f_{i,j,l}^{k} + (Aptitud\ (k))$$

Si el circuito es factible (que produzca todas las salidas)

$$f^k_{i,j,l} = f^k_{i,j,l} + (Aptitud(k) \times 2)$$

Si es el circuito con mayor aptitud (la mejor ruta encontrada)

$$f^k_{i,j,l} = f^k_{i,j,l} + (Aptitud\,(k) \times 3)$$

La *Aptitud* de la hormiga *k* se obtiene de la siguiente forma:

Si la solución obtenida por la hormiga *k* es válida

Aptitud(k) **= Núm_Aciertos + (Casillas_Matriz – Compuertas_Sol)**

En caso contrario

Aptitud(k) **= Núm_Aciertos**

En donde:

Núm_Aciertos es el número de aciertos o emparejamientos producidos entre la salida generada por el circuito construido por el agente y la tabla de verdad dada como entrada por el usuario.

Casillas_Matriz es el número de casillas de la matriz, en otras palabras el número de filas por el número de columnas.

Compuertas_Sol es el número de compuertas que se necesitan para construir la solución dada por el agente.

Pseudocódigo

El algoritmo AS para el diseño de circuitos lógicos en pseudocódigo es el siguiente:

```
Program AS-CircuitosLogicos
   Abrir archivos de entrada y salida
   Inicializar semilla de aleatorios
   Leer datos de entrada
   For i = 1 to Max_Cic
      //Hasta el núm. de ciclos o iteraciones
      For j = 1 to popsize
      //Para cada hormiga o agente
           Construir_Solucion(j)
           Evaluar_individuo(j)
      End
      Actualizar_Rastros
      Imprimir_reporte
   End
   Imprimir_mejor_global
   Cerrar archivos
End
```

Resumen

El área *ACO: Ant Colony Optimization* estudia sistemas artificiales que simulan colonias de hormigas reales, de donde toman su inspiración. Estos sistemas son utilizados para resolver problemas de optimización combinatoria principalmente. El algoritmo *Ant System*, desarrollado por Marco Dorigo fue el primero dentro esta área.

Las hormigas son capaces de encontrar el camino más corto desde el hormiguero a una fuente de comida y viceversa sin usar pistas visuales. Asimismo, son capaces de adaptarse a cambios en el ambiente. El medio por el que las hormigas logran esto es por rastreo de la feromona que ellas mismas depositan mientras caminan.

En el AS las hormigas artificiales tienen comunicación indirecta a través de la feromona y las hormigas tienen preferencia probabilística por las rutas con valores altos de feromona, además, las rutas más cortas tienden a tener una razón más alta de crecimiento del valor de la feromona. A dichas hormigas se les ha dado capacidades que no tienen las hormigas reales, como que sean capaces de determinar qué tan lejos está de un estado, tener información acerca de su ambiente que utilizan al tomar decisiones, además, tienen memoria, la cual es necesaria para asegurar que se generen sólo soluciones factibles.

Para implementar el AS para el problema del diseño de circuitos lógicos combinatorios, se tuvo que modelar este como si fuese un problema tipo el TSP, implementado un circuito en una matriz donde cada columna es un estado o ciudad y la distancia entre cada estado es el incremento o decremento en los aciertos a las salidas del circuito dadas por la tabla de verdad y que produce el cambio de estado.

En el próximo capítulo se describe paso a paso el funcionamiento de nuestra implementación, utilizando un circuito sumador de dos bits como ejemplo.

Referencias Bibliográficas

[1] Dorigo M., Maniezzo V. & Colorni A., **The Ant System: Optimization by a Colony of Cooperating Agents**. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1):29-41, 1996

[2] Dorigo M. **Optimization, Learning and Natural Algorithms.** Ph.D.Thesis, Politecnico di Milano, Italy. 1992.

[3] Dorigo M. and Gambardella L. M., **Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem.** *IEEE Transactions on Evolutionary Computation,* 1(1):53-66, 1997

[4] **Travelling Salesman Problem (TSP)**, http://mathworld.wolfram.com/T/TravelingSalesmanProblem.html.

[5] Dorigo Marco, **Behavior of Real Ants**, http://iridia.ulb.ac.be/~mdorigo/ACO/RealAnts.html, IRIDIA, Université Libre de Bruxelles, Belgium.

[6] Dorigo Marco, *Ant Colony Optimization,* http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html, IRIDIA, Université Libre de Bruxelles, Belgium.

[7] Mariano C.E, Morales E., **MOAQ: An Ant-Q Algorithm for Multiple Objective Optimization Problems**. En *Genetic and Evolutionary Computation Conference (GECCO-99)*, pp. 894-901, 1999.

[8] Stützle Thomas, **An Ant Approach to the Flow Shop Problem**, *Proceedings of EUFIT'98*, Aachen, pages 1560-1564, 1998**.**

[9] Bullnheimer B., R.F. Hartl and C. Strauss, **An Improved Ant system Algorithm for the Vehicle Routing Problem.** Sixth Viennese workshop on Optimal Control, Dynamic Games, Nonlinear Dynamics and Adaptive Systems, Vienna (Austria), May 21-23, 1997, también en: *Annals of Operations Research* (Dawid, Feichtinger and Hartl (eds.): Nonlinear Economic Dynamics and Control, 1999.

[10] Gambardella L.M. & Dorigo M., **Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem.** *Proceedings of ML-95, Twelfth International Conference on Machine Learning,* Tahoe City, CA, A. Prieditis and S. Russell (Eds.), Morgan Kaufmann, 252-260, 1995.

[11] Watkins, C. J. C. H., ***Learning from delayed rewards***. Doctoral thesis, Cambridge University, Cambridge, England, 1989.

[12] Harmon Mance E., **Reinforcement Learning: A Tutorial**. http://www-anw.cs.umass.edu/~mharmon/rltutorial/frames.html

[13] Dorigo M. & Gambardella L.M., **Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem.** *IEEE Transactions on Evolutionary Computation,* 1(1):53-66. (También disponible como reporte técnico TR/IRIDIA/1996-5, IRIDIA, Université Libre de Bruxelles.), 1997

[14] Stützle T. and M. Dorigo, **ACO Algorithms for the Traveling Salesman Problem.** In K. Miettinen, M. Makela, P. Neittaanmaki, J. Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science,* Wiley, 1999. También disponible como reporte técnico: IRIDIA/99-3, Université Libre de Bruxelles, Belgium.

[15] Stütztle T. and Hoos H., **Improvements on the Ant System: Introducing MAX-MIN Ant System.** *Artificial Neural Networks and Genetic Algorithms, pp. 245-249,* 1998.

[16] Bullnheimer B., Hartl R. F. And Strauss C., **A New Rank Based Version of the Ant System – A Computational Study.** *Technical report. University of Viena, Institute of Management Science*, 1997.

[17] White T., Pagurek B. and Oppacher F., **ASGA: Improving the Ant System by Integration with Genetic Algorithms**. *In Proceedings of the 3rd Conference on Genetic Programming (GP/SGA '98)*, July, 1998.

[18] Coello, Carlos A.; Christiansen, Alan D. & Hernández Aguirre, Arturo, **Automated Design of Combinational Logic Circuits Using Genetic Algorithms** *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, ICANNGA'97. University of East Anglia, Norwich, England. Edited by D. G. Smith, N. C. Steele and R. F. Albrecht, Springer-Verlag, pp. 335-338. 2-4 April 1997

[19] Coello Coello, Carlos A.; Christiansen, Alan D. and Hernández Aguirre, Arturo, **Use of Evolutionary Techniques to Automate the Design of Combinational Circuits**, *International Journal of Smart Engineering System Design, Elsevier Science*, Vol. 2, No. 4, pp. 299-314, June 2000

[20] Coello Coello, Carlos A.; Zavala G. Rosa Laura; Mendoza G., Benito and Hernández Aguirre, Arturo, **Ant Colony System for the Design of Combinational Logic Circuits**, in Julian Miller, Adrian Thompson, Peter Thomson and Terence C. Fogarty (Eds.), *Evolvable Systems: From Biology to Hardware*, Edinburgh, Scotland, Springer-Verlag, pp. 21--30, April 2000.

[21] Kaelbling, Leslie Pack and Littman, Michael L. **Reinforcement Learning: A Survey.** *Journal of Artificial Intelligence 4 (1996), pp. 237-285.*

[22] Goldberg David E., **Genetic Algorithms in Search, Optimization, and Machine Learning**, Addison-Wesly Publishing Company, inc. ISBN 0-201-15767-5.

[23] Coello Coello, Carlos A.; Zavala G. Rosa Laura; Mendoza G., Benito & Hernández Aguirre, Arturo, **On the Use of the Ant System to Design Electrical Circuits**, *X Congreso Latino-Iberoamericano de Investigación de Operaciones y Sistemas*, México, D.F., 4 al 8 de septiembre del 2000.

3 El Proceso de Construcción de las Rutas de los

Hasta ahora se explicó el Ant System y su aplicación para el diseño de circuitos lógicos combinatorios. A continuación se trabajará con un ejemplo, para ilustrar paso a paso el proceso de construcción de un circuito, es decir, la ruta de un agente, mediante la utilización de esta aplicación. Para ello, se ha tomado un ejemplo que cuenta con cuatro entradas y tres salidas; se trata de un sumador de dos bits. El cual se explicará con mayor detalle en el capítulo 4. La tabla 3.1 muestra su tabla de verdad.

Entradas				Salidas		
A	B	C	D	1	2	3
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Tabla 3.1 Tabla de verdad del sumador de dos bits

La tabla de verdad indica el comportamiento que el circuito debe tener. Ésta contiene todas las posibles combinaciones de valores que pueden entrar al circuito, dado el número de entradas, mismas que se encuentran representadas en las columnas etiquetadas de esa manera. También se indican los respectivos valores de salida deseados para cada combinación de entradas, representados en las columnas etiquetadas de esa manera. Por ejemplo, cuando las entradas A y B tienen cero y en las entradas C y D tienen uno, el resultado para la salida uno debe ser cero y para las salidas dos y tres debe ser uno.

Nuestra aplicación recibe esta matriz desde un archivo de entrada y produce el resultado en un archivo de salida. La figura 3.1 muestra la interfaz de nuestra aplicación hecha para experimentar en Windows (en modo de 32 bits).

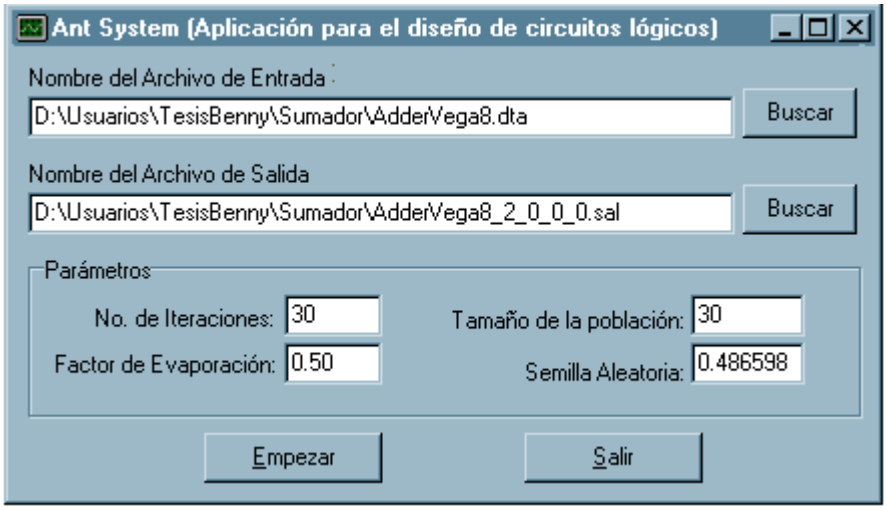


Figura 3.1 Interfaz de la aplicación en versión

3.1 Datos de entrada

Para poder trabajar, el programa requiere de los siguientes datos de entrada:

- Nombre del Archivo de Entrada
- Nombre del Archivo de Salida
- Número de Iteraciones
- Tamaño de la población
- Factor de Evaporación
- Semilla Aleatoria

El archivo de entrada

El archivo de entrada básicamente contiene la tabla de verdad que representa al circuito y algunos datos asociados a ésta. El formato es el mismo que ocupa el BGA de Coello[1][2], eliminando la información innecesaria, adaptándolo a las necesidades del AS. La figura 3.2 muestra la estructura del archivo de entrada para el sumador, utilizando una matriz de nueve filas por cinco columnas.

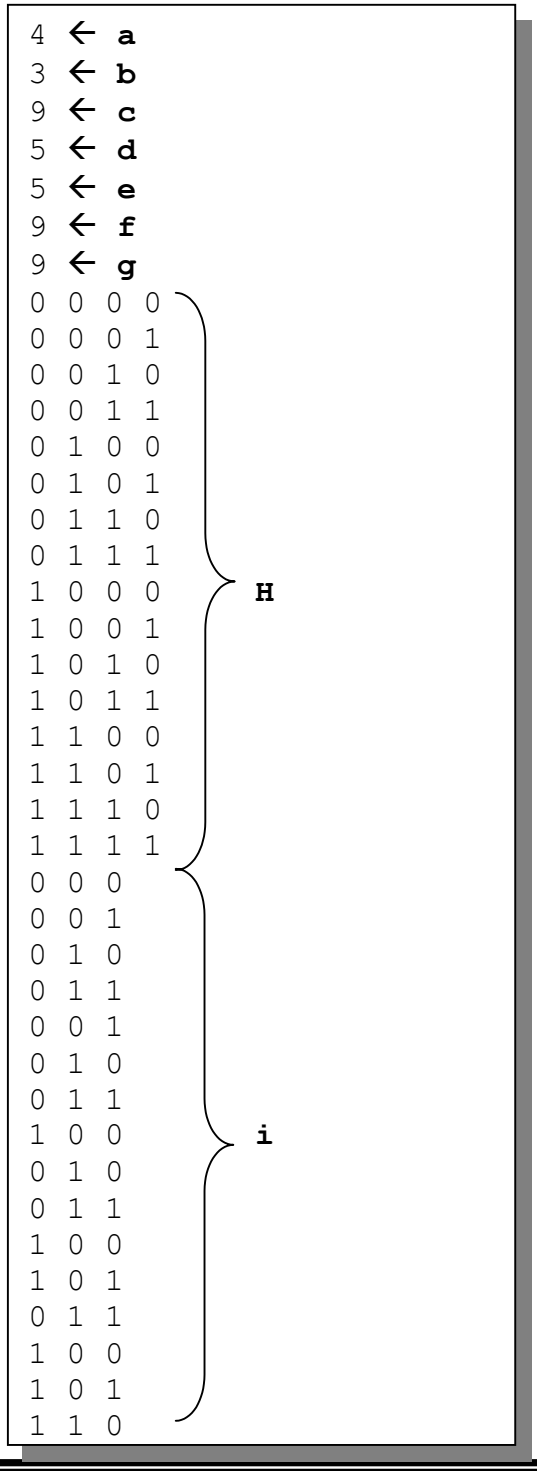


Figura 3.2 Esquema del archivo de entrada para un sumador de dos bits.

Donde:

a) Representa el número de entradas posibles (número de entradas de la tabla de verdad).

b) Número de salidas de la tabla de verdad.

c) Número de renglones (casillas en cada nivel o estado del circuito).

d) Número de columnas (número de niveles o estados en el circuito).

e) Número de compuertas (cuántos tipos de compuertas diferentes se pueden utilizar).

f) Número de posibles valores que puede tomar la entrada uno (el máximo entre número de renglones y el número de entradas de la tabla de verdad).

g) Número de posibles valores que puede tomar la entrada dos (el máximo entre el número de renglones y el número de entradas de la tabla de verdad).

h) Las entradas de la tabla de verdad.

i) Las salidas de la tabla de verdad para cada combinación de entradas.

El archivo de salida

En este archivo el algoritmo va guardando la matriz o ruta resultante de la mejor hormiga en cada iteración, y al final pone también la mejor matriz obtenida en toda la corrida, es decir en todas las iteraciones. En la figura 3.3 se muestra un fragmento de la matriz resultante para el circuito que se está tomando como ejemplo.

Número de iteraciones

El número de iteraciones se refiere al número de veces que cada uno de los agentes intentará crear una solución. Cabe mencionar que en la primera iteración no hay rastro de feromona alguno y que mientras aumenta el número de iteraciones, los agentes utilizarán los rastros generados en la iteraciones anteriores.

```
Fecha y tiempo de inicio: Mon May 01 23:16:44 2000

Semilla:0.111111
Iteraciones: 15

Mejor Individuo Global:

Encontrado en la iteración No.: 12
Aptitud     ---> 69.000000
Violaciones ---> 0
Aciertos    ---> 48
Wires---> 10
Num. de compuertas---> 8
Bonus ---> 21

CIRCUITO:
WIRE1<3, 8> AND<1, 6>   OR<1, 4>    WIRE1<1, 1> WIRE1<1, 1>
XOR<7, 1>   OR<5, 2>    XOR<4, 2>   WIRE1<2, 2> WIRE1<2, 2>
XOR<2, 4>   WIRE1<3, 3> WIRE1<3, 3> WIRE1<3, 3> WIRE1<3, 3>
OR<3, 4>    AND<5, 2>   OR<6, 3>    AND<3, 4>   NOT1<2, 5>
AND<8, 2>   WIRE1<8, 2> XOR<8, 4>   OR<7, 9>    OR<5, 3>
WIRE1<1, 1> XOR<4, 4>   OR<7, 8>    OR<7, 5>    AND<1, 3>
WIRE1<7, 4> NOT1<6, 8>  WIRE1<5, 9> XOR<3, 3>   AND<7, 3>
WIRE1<9, 2> WIRE1<6, 7> OR<7, 6>    NOT1<9, 3>  AND<4, 4>
WIRE1<8, 8> XOR<4, 8>   OR<7, 5>    AND<2, 3>   NOT1<9, 2>

Fecha y tiempo de Terminación: Mon May 01 23:18:45 2000
```

Figura 3.3 Fragmento de un archivo de salida.

Tamaño de la población

El tamaño de la población se refiere al número de hormigas de la colonia, que son los agentes que se encargarán de buscar rutas o circuitos válidos.

Factor de evaporación

En el capítulo anterior se mencionaba que el factor de evaporación es un valor que nos indica qué porcentaje de los rastros de feromona existentes se eliminan después de cada iteración. Esto es para simular la evaporación de la feromona depositada por las hormigas reales.

Semilla de aleatorios

El algoritmo genera números aleatorios en varias partes; este valor inicializa el sistema generador de números aleatorios.

Cabe recalcar que sólo el tamaño de la población, el número de iteraciones y el factor de evaporación son parámetros clásicos del AS; los demás son propios de esta aplicación.

3.2 La construcción de una ruta

Cada agente debe construir una ruta en cada iteración, es decir, llenar una matriz con compuertas y sus respectivas conexiones, tratando de construir un circuito válido. Llamamos válido al circuito que cumple con la tabla de verdad que recibe como entrada. Asumiremos para nuestro ejemplo que la matriz que utilizamos, dado el archivo de entrada de la figura 3.2, es de nueve filas por cinco columnas.

En el capítulo anterior se menciona que a cada columna de la matriz se le llama estado y cada elemento, fila o casilla de la columna es un subestado. Como en este ejemplo el circuito cuenta con tres salidas, los tres primeros subestados de cada columna se escogen mediante el factor P de selección explicado en el apartado 2.4 del capítulo anterior, mientras que los demás subestados se eligen aleatoriamente. La matriz se va construyendo columna por columna siguiendo los pasos que se explican a continuación.

PASO 1: Seleccionar las compuertas para las tres primeras casillas de la columna.

Obviamente este proceso se hace casilla por casilla. A continuación se muestra cómo se elige la compuerta que irá en la primera casilla.

????????				

Tabla 3.2 Estado de la matriz al empezar la construcción de una ruta.

La selección se hace asignando un valor, llamado factor P de selección, a cada combinación de: {Tipo de Compuerta, Entrada1 y Entrada2} que puede ponerse en esa casilla y entonces, mediante una selección por ruleta, se decide cuál de todas se queda en esa posición.

El pseudocódigo para la selección es el siguiente:

```
Selección de compuerta y entradas
/*Para todas las compuertas*/
Para i1 = 0 a NumGates-1
   /*Para todos los posibles valores que pueden entrar por la
     entrada uno */
   Para i2 = 1 a NumInputs1
      /*Para todos los posibles valores que pueden entrar por la
        entrada dos */
      Para i3 = 1 a NumInputs2
         Probar_Compuerta i1 con entradas (i2, i3)
         Asignar_Factor_P a la compuerta i1 con entradas (i2, i3)
      Fin Para i3
   Fin Para i2
Fin Para i1
Ruleta()
Fin de Selección
```

Como puede verse, en este ciclo se recorre cada tipo posible de compuerta con sus posibles entradas:

```
AND(1,1); AND(1,2); AND(1,3); ...AND(2,1); AND(2,2); ...AND(8,8)
OR(1, 1); OR(1, 2); OR(1, 3); ...OR(2, 1); OR(2, 2); ...OR(5, 5)
.
.
.
```

y se le asigna a cada posible combinación el factor P de selección, para posteriormente hacer una selección mediante ruleta.
Como se menciona en el capítulo anterior, el valor del factor P se calcula con una fórmula que toma la distancia y la feromona y nos da un valor que podría ser visto como la probabilidad de cambiar a ese nuevo subestado.

La feromona se toma del arreglo TRAILS, mencionado en el apartado 2.4.

Para calcular la distancia, se debe probar la compuerta con sus entradas, después, se debe obtener el incremento o decremento en el número de aciertos (comparándolos con los aciertos del subcircuito construido hasta la columna o paso anterior).

Para probar una compuerta lo que se hace es colocar en la casilla que se está tratando de asignar, la compuerta con sus entradas y rellenar las demás casillas con WIRE1(1, 1), para que la salida del circuito sea la salida de dicha compuerta. Entonces, se comparan las salidas producidas por esa parte del circuito con la salida 1 de la tabla de verdad dada de

entrada, para obtener el número de aciertos, ya que se trata de la primera casilla de la columna, es decir forma parte de la primera fila. Por ejemplo, si estamos tratando de probar la compuerta OR con entradas 2 y 3, para la situación descrita anteriormente:

OR(2, 3)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)
WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)
WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)
WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)
WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)
WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)
WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)
WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)
WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)	WIRE1(1, 1)

Tabla 3.3 Estado de la matriz, para probar OR(2, 3) en la primera casilla de la primera fila.

La ruta de esta parte del circuito está remarcada. Ésta es construida desde la última columna de derecha a izquierda, siguiendo las conexiones que sus componentes indiquen.

Cuando se está trabajando con la primera columna los aciertos hasta el nivel anterior son cero.

Una vez obtenida la distancia, se toma del arreglo TRAILS la feromona para esa combinación de compuerta y entradas, y se calcula el factor P de selección.

Obviamente, después de esta operación, la matriz se regresa a su estado anterior, como se muestra en la Tabla 3.2.

Una vez asignada la probabilidad de todas las combinaciones de {Tipo de Compuerta, Entrada1 y Entrada2}, se hace la selección por ruleta y entonces se coloca la combinación elegida. Por ejemplo, supongamos que una selección por ruleta nos dio como resultado WIRE1(3, 8), tal y como se muestra en la tabla 3.4.

Para elegir qué poner en las siguientes casillas se ocupa el mismo procedimiento, sólo que ahora la matriz se rellena de WIRE1(2,2) o WIRE1(3,3), según corresponda al subestado y debe acertar a las salidas 2 y 3 de la tabla de verdad respectivamente.

Supongamos que ahora vamos a elegir qué poner en la segunda casilla.

WIRE1(3, 8)				
????????				

Tabla 3.4 Estado de la matriz, al haber elegido WIRE(3, 8) en la primera casilla de la primera fila. Lista para elegir el elemento de la segunda casilla de la misma columna.

De la misma manera que se hizo para la primera casilla se prueban todas las combinaciones de compuertas y entradas y se les asigna su factor P de selección, para después hacer la selección por ruleta y elegir aquella que ocupará ese lugar. La Tabla 3.5, muestra cómo se rellena la matriz cuando se está probando la combinación AND(1, 3).

WIRE1(3,8)				
AND(1,3)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)
WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)
WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)
WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)
WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)
WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)
WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)
WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)	WIRE1(2,2)

La tabla 3.6 muestra las salidas y aciertos producidos por la combinación AND(1, 3), tomando en cuenta que 1 es la columna A de la tabla de verdad y 3 la columna C (tabla 3.1).

Entradas		Salidas		Aciertos
A	**C**	**AND(A,C)**	**Salida 2**	
0	0	0	0	☑
0	0	0	0	☑
0	1	0	1	☒
0	1	0	1	☒
0	0	0	0	☑
0	0	0	1	☒
0	1	0	1	☒
0	1	0	0	☑
1	0	0	1	☒
1	0	0	1	☒
1	1	1	0	☒
1	1	1	0	☒
1	0	0	1	☒
1	0	0	0	☑
1	1	0	0	☑
1	1	1	1	☑

Tabla 3.6 Salidas y aciertos producidos por la combinación AND(1, 3), tomando en cuenta que 1 es la columna A de la tabla de verdad y 3 la columna C (tabla 3.1).

Como puede observarse en la Tabla 3.6, la compuerta AND(1, 3) en esa posición produce 7 aciertos (los aciertos están marcados con ☑). Como se está en la primera columna la feromona depositada para ésta y todas las combinaciones es de cero. Debido a esto, el valor asignado resultante del factor P de selección es de 7 de acuerdo a la fórmula explicada en el apartado 2.4. Dicha fórmula indica que cuando la feromona o la distancia sean cero no se tomarán en cuenta para asignar el factor P. Al igual que como se hizo para la primera casilla de la columna, se les asigna a todas las combinaciones de {Tipo de Compuerta, Entrada1 y Entrada2} su factor P y posteriormente se hace la selección. Supongamos que la que ganó fue la combinación XOR(7,1).

Esto mismo se hace para la casilla 3, rellenando el resto de la matriz con WIRE1(3,3). Supongamos que la combinación elegida es XOR(2,4).

PASO 2: Elegir aleatoriamente las compuertas de las demás casillas de la columna.

Después de que se asignaron las compuertas a las tres primeras casillas, que para este ejemplo son las que representan las tres salidas de la tabla de verdad, lo que sigue es llenar las casillas que faltan. La tabla 3.7 muestra el estado de la matriz antes de realizar esta operación

WIRE1(3, 8)				
XOR(7,1)				
XOR(2,4)				
?????????				
?????????				
?????????				
?????????				
?????????				
?????????				

Tabla 3.7 Estado de la matriz después de haber elegido los elementos de las tres primeras casillas de la primera columna.

Se genera, para cada casilla, lo siguiente:


Un número aleatorio entre `0` y `NumGates-1` (número de compuertas permitidas)


Un número aleatorio entre `1` y `NumInputs1` (Posibles valores de la entrada uno de la compuerta)


Un número aleatorio entre `1` y `NumInputs2` (Posibles valores de la entrada dos de la compuerta)

Los resultados se colocan en la casilla correspondiente, supongamos los que se muestran en la tabla 5.8.

WIRE1(3, 8)				
XOR(7,1)				
XOR(2,4)				
OR(3,4)				
AND(8,2)				
WIRE1(1,1)				
WIRE1(7,4)				
WIRE1(9,2)				
WIRE1(8, 8)				

Tabla 3.8 Estado de la matriz después de haber elegido todos los elementos de las casillas de la primera columna.

PASO 3.- Repetir los pasos 1 y 2 para todas las columnas.

En las tablas 3.9, 3.10 y 3.11 se muestra los que pasa con la columna 2.

WIRE1(3, 8)	?????????			
XOR(7,1)	?????????			
XOR(2,4)	?????????			
OR(3,4)				
AND(8,2)				
WIRE1(1,1)				
WIRE1(7,4)				
WIRE1(9,2)				
WIRE1(8, 8)				

Tabla 3.9 Estado de la matriz, al empezar el paso 1 para la segunda columna.

Para cada una de las tres primeras casillas, se calcula el factor P de todas las combinaciones posibles de compuertas y entradas y se hace la selección (mediante ruleta), de la combinación que irá en cada casilla. Supongamos los valores que se muestran en la tabla 3.10.

WIRE1(3, 8)	AND(1, 6)			
XOR(7,1)	OR(5, 2)			
XOR(2,4)	WIRE1(3, 3)			
OR(3,4)	?????????			
AND(8,2)	?????????			
WIRE1(1,1)	?????????			
WIRE1(7,4)	?????????			
WIRE1(9,2)	?????????			
WIRE1(8, 8)	?????????			

Tabla 3.10 Estado de la matriz, después de haber elegido las tres primeras casillas de la segunda columna.

Después se eligen las demás compuertas de la columna aleatoriamente. Supongamos los valores que se muestran en la tabla 3.11.

WIRE1(3, 8)	AND(1, 6)			
XOR(7,1)	OR(5, 2)			
XOR(2,4)	WIRE1(3, 3)			
OR(3,4)	AND(5, 2)			
AND(8,2)	WIRE1(8, 2)			
WIRE1(1,1)	XOR(4, 4)			
WIRE1(7,4)	NOT1(6, 8)			
WIRE1(9,2)	WIRE1(6, 7)			
WIRE1(8, 8)	XOR(4, 8)			

Tabla 3.11 Estado de la matriz, después de haber elegido todos los elementos de todas las casillas de la segunda columna.

Todo esto se repite para las demás columnas de la matriz. Supongamos haber llenado las demás columnas con los valores que se muestra en la tabla 3.12.

WIRE1(3,8)	AND(1, 6)	OR(1, 4)	WIRE1(1, 1)	WIRE1(1, 1)
XOR(7,1)	OR(5, 2)	XOR(4, 2)	WIRE1(2, 2)	WIRE1(2, 2)
XOR(2,4)	WIRE1(3, 3)	WIRE1(3,3)	WIRE1(3,3)	WIRE1(3, 3)
OR(3,4)	AND(5, 2)	OR(6, 3)	AND(3, 4)	NOT1(2,
AND(8,2)	WIRE1(8, 2)	XOR(8, 4)	OR(7, 9)	OR(5, 3)
WIRE1(1,1)	XOR(4, 4)	OR(7, 8)	OR(7, 5)	AND(1, 3)
WIRE1(7,4)	NOT1(6, 8)	WIRE1(5,	9) XOR(3, 3)	AND(7, 3)
WIRE1(9,2)	WIRE1(6, 7)	OR(7, 6)	NOT1(9, 3)	AND(4, 4)
WIRE1(8, 8)	XOR(4, 8)	OR(7, 5)	AND(2, 3)	NOT1(9, 2)

Tabla 3.12 Estado de la matriz, después de haberse llenado todas las columnas.

En la tabla 3.12 se puede observar en las dos últimas columnas que las tres primeras casillas tienen WIRE1(1, 1), WIRE1(2, 2), WIRE1(3, 3), respectivamente. Esto quiere decir que después de la columna 3 ya no se encontró ninguna combinación que superara las encontradas en esa columna. De hecho, la tercera fila tiene WIRE1(3, 3) desde la segunda columna, es decir que no hubo nada que superara al XOR(3,4) de la primera columna, ya que la combinación de dicha compuerta y esas entradas producen todos los valores correspondientes a la salida 3 de la tabla de verdad del circuito.

PASO 4: Contar las compuertas, aciertos y calcular la aptitud.

No todas las casillas de la matriz son utilizadas por el circuito. En la tabla 3.13 se encuentran marcadas las casillas de la matriz que son parte de la solución obtenida por esta hormiga.

WIRE1(3,8)	AND(1, 6)	OR(1, 4)	WIRE1(1, 1)	WIRE1(1, 1)
XOR(7,1)	OR(5, 2)	XOR(4, 2)	WIRE1(2, 2)	WIRE1(2, 2)
XOR(2,4)	WIRE1(3, 3)	WIRE1(3,3)	WIRE1(3,3)	WIRE1(3, 3)
OR(3,4)	AND(5, 2)	OR(6, 3)	AND(3, 4)	NOT1(2, 5)
AND(8,2)	WIRE1(8, 2)	XOR(8, 4)	OR(7, 9)	OR(5, 3)
WIRE1(1,1)	XOR(4, 4)	OR(7, 8)	OR(7, 5)	AND(1, 3)
WIRE1(7,4)	NOT1(6, 8)	WIRE1(5,	9) XOR(3, 3)	AND(7, 3)
WIRE1(9,2)	WIRE1(6, 7)	OR(7, 6)	NOT1(9, 3)	AND(4, 4)
WIRE1(8, 8)	XOR(4, 8)	OR(7, 5)	AND(2, 3)	NOT1(9, 2)

Tabla 3.13 Casillas de la matriz utilizadas en la solución (ruta o tour) propuesta por la hormiga

Cabe recordar que las conexiones se hacen columna por columna de derecha a izquierda y que de la última columna sólo se toman las casillas que pertenecen a las filas que representan una salida (las tres primeras para este ejemplo). Otra cosa importante que se debe recordar, es que los WIREs no son compuertas, sino una conexión, por lo que no se cuentan como tal y sólo la entrada uno es la que se utiliza, es decir, solo tienen una entrada (la segunda no se toma en cuenta). En este ejemplo, de la primera columna (de izquierda a derecha) se utilizan 3 compuertas, de la segunda 3 y de la tercera 2. Las dos últimas columnas no tienen ninguna compuerta utilizada (los elementos utilizados son WIRE1). Con esto tenemos 3 + 3 + 2 = 8 compuertas. Los aciertos en total se obtienen sumando los aciertos a cada salida.

PASO 5: Repetir los pasos del 1 al 4 para el número de hormigas indicadas por `popsize`.

Lo anterior se hace para todas las hormigas, es decir se debe crear la ruta para cada una de las hormigas de la población y obtener sus respectivos resultados (número de compuertas, aciertos y aptitud).

PASO 6: Actualizar la feromona de la ruta seguida por cada hormiga.

La actualización de las feromonas consiste de dos pasos:

1. *Evaporación de Feromona*

   Supongamos que el factor de evaporación que utilizamos para este ejemplo sea 0.5. Esto quiere decir que el 50% de los rastros de feromona se perderán después de que todas las hormigas hayan construido una solución y antes de que se deposite la feromona de estas nuevas rutas. De manera que todos los rastros son disminuidos a la mitad aquí.

2. *Incremento en la feromona de las rutas visitadas por las hormigas*

   En este paso, cada hormiga actualiza la feromona de acuerdo a la ruta o solución que construyó. La cantidad de feromona que una hormiga deposita en cada una de las compuertas de las tres primeras filas de cada columna está dada de acuerdo a los siguientes criterios:

   Si el circuito viola restricciones (no acierta en todas las salidas)

   $$\textit{Feromona} = \textit{Feromona} + \textit{Aptitud}$$

   Si el circuito no viola restricciones (acierta en todas las salidas)

   $$\textit{Feromona} = \textit{Feromona} + (\textit{Aptitud} \times 2)$$

Para la ruta del mejor individuo global

$$Feromona = Feromona + (Aptitud \times 3)$$

La *Aptitud* se calcula como se muestra al final de la sección 2.4

PASO 7: Repetir todos los pasos hasta terminar el número de iteraciones dadas.

Se deben repetir todos los pasos tantas veces como lo indique el número de iteraciones dadas como entrada al algoritmo. En cada iteración se debe actualizar el mejor individuo global, el cual representará al final de todas las iteraciones, la mejor solución encontrada en la corrida.

Resumen

En este capítulo se explicó paso a paso como es que opera nuestra aplicación del AS, los requerimientos de entrada como el archivo de entrada, el archivo de salida. Además se explicaron que son cada uno de los parámetros que esta aplicación como el número de iteraciones, el tamaño de la población, el factor de evaporación y las semilla aleatoria.

En el próximo capítulo se mostrarán los mejores resultados obtenidos con algunos circuitos, comparándolos con los resultados obtenidos con otras técnicas tanto como tradicionales como evolutivas.

Referencias Bibliográficas

[1] Coello, Carlos A.; Christiansen, Alan D. & Hernández Aguirre, Arturo, **Automated Design of Combinational Logic Circuits Using Genetic Algorithms** *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, ICANNGA'97. University of East Anglia, Norwich, England. Edited by D. G. Smith, N. C. Steele and R. F. Albrecht, Springer-Verlag, pp. 335-338. 2-4 April 1997

[2] Coello Coello, Carlos A.; Christiansen, Alan D. and Hernández Aguirre, Arturo, **Use of Evolutionary Techniques to Automate the Design of Combinational Circuits**, *International Journal of Smart Engineering System Design, Elsevier Science*, Vol. 2, No. 4, pp. 299-314, June 2000

4 Resultados de la Aplicación del AS al diseño

Para evaluar el desempeño de nuestra implementación del Ant System, se utilizaron algunos circuitos lógicos obtenidos de la literatura (principalmente los utilizados por Coello[1] en la prueba de algunas técnicas evolutivas para diseñar circuitos combinatorios), todos con diferentes características y complejidad. Los resultados fueron comparados con aquellos obtenidos por diseñadores humanos y con otras técnicas evolutivas como son el Algoritmo Genético Multiobjetivo(MGA)[2] y el Algoritmo Genético Binario(BGA)[3].

En este capítulo se muestran dichos resultados y comparaciones, de los cuales algunos fueron publicados en [4] y [5].

Como el tiempo es dependiente también de las características de la computadora donde se corra el programa, se tomó como referencia los tiempos obtenidos en una máquina con las características físicas y de software de la tabla 4.1.

Procesador	Intel Pentium III a 550 Mhz.
Memoria RAM	128 Mbytes.
Disco Duro	13 Gbytes.
Memoria de Video	8 Mbytes.
Sistema Operativo	Microsoft Windows 98
Compilador	Borland C++Builder 4

Tabla 4.1 Características de la computadora donde se hicieron los experimentos.

4.1 Ejemplo 1

El primer ejemplo consiste en un circuito de 3 entradas y 1 salida, con la tabla de verdad que se muestra en la tabla 4.2[4]. La representación gráfica del circuito dado como mejor solución por el AS se muestra en la figura 4.2[4].

La comparación del resultado producido por el AS, un BGA y dos diseñadores humanos se muestran en la tabla 4.3. En este caso el diseñador humano 1 utilizó el método gráfico de Mapas de Karnaugh[6] y álgebra Booleana, para simplificar el circuito, mientras que el segundo utilizó el método tabular de Quine-McCluskey[8]. Se puede observar que la

solución encontrada por el AS tiene el mismo número de compuertas que el BGA y mejora las soluciones de los diseñadores humanos.

Entradas			Salida
X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Tabla 4.2 Tabla de verdad para el circuito del ejemplo 1.

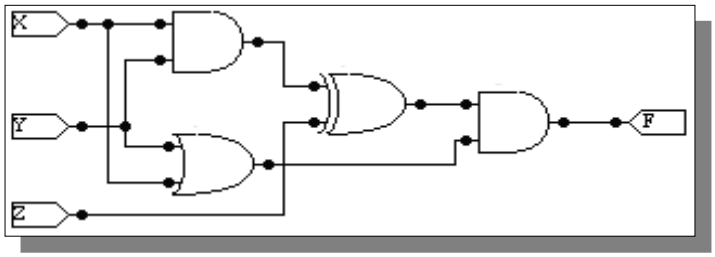


Figura 4.1 Circuito producido por el AS para el primer ejemplo.

Diseñador Humano 1	$F=Z(X \oplus Y) + Y(X \oplus Z)$	5 Compuertas (2 ANDs, 1 OR, 2 XORs)
Diseñador Humano 2	$F= X'YZ + X(Y \oplus Z)$	6 Compuertas (3 ANDs, 1 OR, 1 XORs, 1 NOT)
BGA	$F = Z(X + Y) \oplus (XY)$	4 Compuertas (2 ANDs, 1 OR, 1XOR)
AS	$F=(Z \oplus XY)\ (X + Z)$	4 Compuertas (2 ANDs, 1 OR, 1 XORs)

Tabla 4.3 Comparación de resultados entre el AS, un BGA y dos diseñadores humanos.

Este es un ejemplo muy pequeño. El tiempo que se tardó el AS en realizar la corrida en la que se obtuvo la mejor solución para este circuito con los parámetros de la tabla 4.4, es de 2 segundos.

Parámetro	Valor
Tamaño de la Matriz	5 x 5
Factor de Evaporación	0.75
Tamaño de la Población	10
Número de Iteraciones	30
Semilla aleatoria	0.740887

Tabla 4.4 Parámetros utilizados para obtener la solución de la figura 4.1.

4.2 Ejemplo 2 (circuito de Sasao)

El segundo ejemplo consiste en un circuito de 4 entradas y 1 salida, con la tabla de verdad que se muestra en la tabla 4.5. Este circuito ha sido usado por Sasao[9] para ilustrar su técnica de simplificación de circuitos, la cual se basa en un proceso de diseño con uso exclusivo de compuertas ANDs y XORs.

Entradas				Salida
W	X	Y	Z	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Tabla 4.5 Tabla de verdad del circuito de Sasao.

La representación gráfica del circuito dado como mejor solución por el AS se muestra en la figura 4.2 y la comparación de los resultados producidos por el AS, un BGA, un diseñador humano (usando Mapas de Karnaugh) y la solución obtenida por Sasao con su método, se muestran en la tabla 4.6. La solución presentada aquí supera a la publicada en [4] y [5] (9 compuertas) y consta de únicamente de 7 compuertas, mejorando los resultados producidos por los diseñadores humanos, el método de Sasao e incluso el BGA.

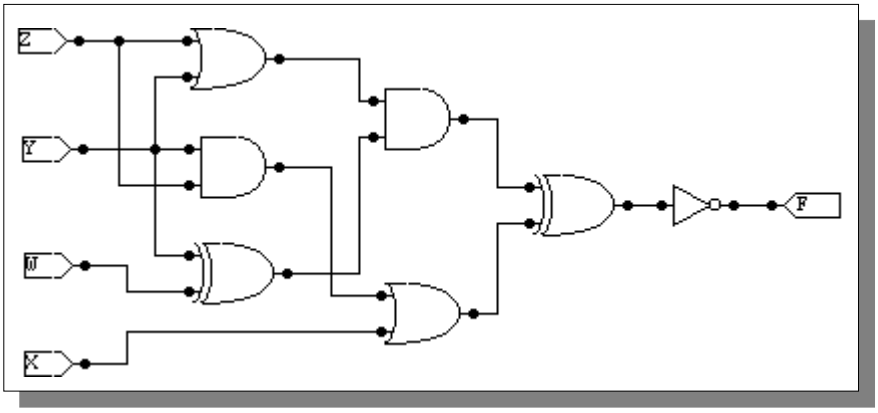


Figura 4.2 Circuito producido por el AS para el ejemplo 2.

Diseñador Humano 1	$F=((Z'X) \oplus (Y'W')) + ((X'Y)(Z \oplus W'))$	11 Compuertas (5 ANDs, 1 OR, 2 XORs, 4 NOTs)
Sasao	$F= X' \oplus Y'W' \oplus XY'Z' \oplus X'Y'W$	12 Compuertas (5 ANDs, 3 XORs, 4 NOTs)
BGA	$F = (WYX' \oplus ((W+Y) \oplus Z \oplus (X+Y+Z)))'$	10 Compuertas (2 ANDs, 3 ORs, 3XOR, 2 NOTs)
AS	$F=(((Z+ Y) (Y \oplus W)) \oplus ((YZ) + X)))'$	7 Compuertas (2 ANDs, 2 OR, 2 XORs, 1 NOT)

Tabla 4.6 Comparación de resultados entre el AS, un BGA y dos diseñadores humanos para el ejemplo 2.

Para obtener esta solución, se utilizaron los parámetros que se muestran en la tabla 4.7. El tiempo que tardó el AS en realizar esa corrida fue 228 segundos, o sea, 3.8 minutos. Aunque este circuito es de tan solo una salida, tiene cierta complejidad (al menos para el

AS) ya que las corridas tardan más que con otros circuitos con las mismas salidas e incluso que algunos otros de varias salidas y de los cuales se habla más adelante .

Parámetro	Valor
Tamaño de la Matriz	8 x 8
Factor de Evaporación	0.75
Tamaño de la Población	30
Número de Iteraciones	50
Semilla aleatoria	0.784660

Tabla 4.7 Parámetros utilizados para obtener la solución de la Figura 4.2.

4.3 Ejemplo 3 (Circuito de Katz de una salida)

El tercer ejemplo, igual que el anterior, consiste en un circuito de 4 entradas y 1 salida, su tabla de verdad se muestra en la tabla 4.8. A este ejemplo le llamamos de Katz por la referencia bibliográfica de donde se tomó[10].

Entradas				Salida
W	X	Y	Z	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Tabla 4.8 Tabla de verdad del circuito de Katz de una salida.

La gráfica del circuito dado como mejor solución por el AS se muestra en la figura 4.3 y la comparación del resultado producido por el AS, un BGA y las soluciones encontradas por dos diseñadores humanos (el primero usando Mapas de Karnaugh y el segundo el procedimiento de Quine-McCluskey) se muestran en la tabla 4.9. Esta solución supera a la publicada en [4], la cual constaba de 8 compuertas, mientras que esta consta únicamente de 7 compuertas.

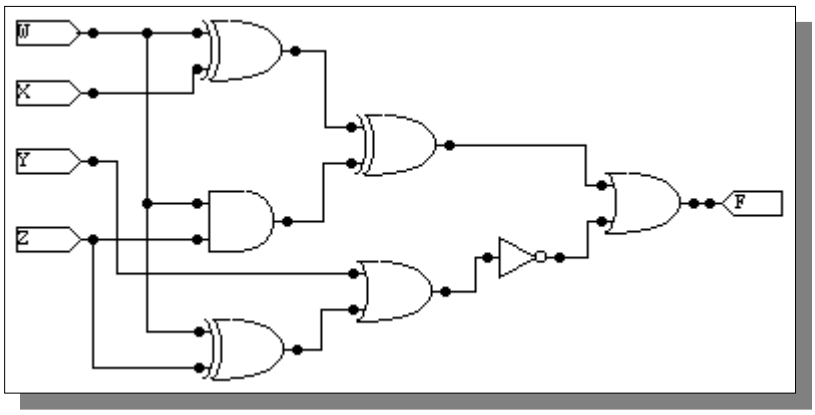


Figura 4.3 Circuito producido por el AS para el ejemplo 3.

Diseñador Humano 1	$F=((W \oplus X) \oplus ((WZ)(B+C))) +((A+C)+D)'$	9 Compuertas (2 ANDs, 4 OR, 2 XORs, 1 NOT)
Diseñador Humano 2	$F= W' X + W(X'Z'+Y'Z)$	10 Compuertas (4 ANDs, 2 ORs, 4 NOTs)
BGA	$F=((W \oplus X) \oplus WZ) + (Y+(W \oplus Z))'$	7 Compuertas (1 ANDs, 2 ORs, 3XOR, 1 NOT)
AS	$F= (W \oplus X) \oplus (WZ) + (Y+(W \oplus Z))'$	7 Compuertas (1 ANDs, 2 OR, 3 XORs, 1 NOT)

Tabla 4.9 Comparación de resultados entre el AS un BGA y dos diseñadores humanos para el ejemplo 3.

La solución dada por el AS es igual a la dada por el BGA (en cuanto a compuertas) y superan las de los métodos utilizados por los diseñadores humanos. Para obtener la solución del AS, se utilizaron los parámetros que se muestran en la tabla 4.10. El tiempo

que tardó el AS en realizar esa corrida fue 96 segundos, o sea, 1.6 minutos. Como se mencionó anteriormente, aunque este circuito tiene las misma cantidad de entradas y salidas que el de Sasao, requiere menos tiempo para sus corridas, además de que se encuentran soluciones factibles de manera más consistente que con el circuito de Sasao.

Parámetro	Valor
Tamaño de la Matriz	8 x 8
Factor de Evaporación	0.50
Tamaño de la Población	10
Número de Iteraciones	50
Semilla aleatoria	0.622876

Tabla 4.10 Parámetros utilizados para obtener la solución de la figura 4.2.

4.4 Ejemplo 4 (Sumador de dos bits)

Entradas				Salida		
A	B	C	D	F1	F2	F3
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Tabla 4.11 Tabla de verdad del Sumador de 2 bits

El proceso de diseñar circuitos de múltiples salidas resulta más complicado para los diseñadores humanos, ya que requieren hacer el diseño por separado para cada una de las salidas y luego se busca (mediante inspección visual) llegar a obtener una reutilización de las compuertas, para formar un solo circuito.

El cuarto ejemplo es el primer circuito de múltiples salidas con el que se probó el AS y consiste en un sumador de 2 bits el cual consta de 4 entradas y 3 salidas. La tabla de verdad que se muestra en la tabla 4.11, corresponde a este circuito.

La representación gráfica del circuito dado como mejor solución por el AS se muestra en la figura 4.4 y la comparación del resultado producido por el AS, un BGA y un diseñador humano (usando Mapas de Karnaugh), se muestran en la Tabla 4.12. Las soluciones del BGA y el AS son iguales en cuanto al número de compuertas y utilizan 5 compuertas menos que la del diseñador humano.

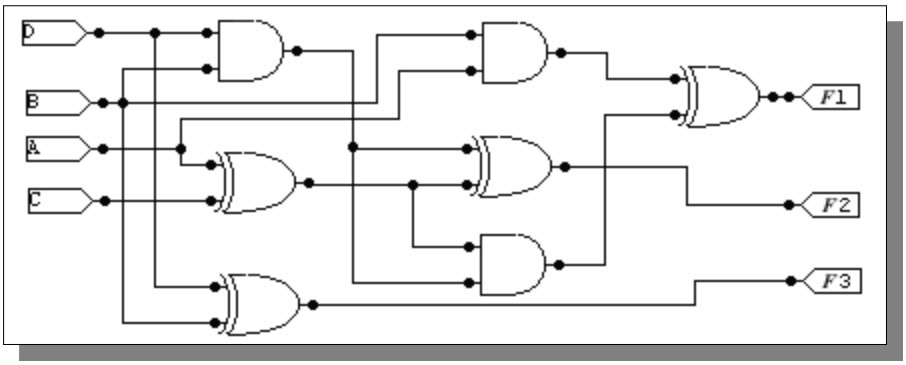


Figura 4.4 Circuito producido por el AS para el sumador de 2 bits.

Diseñador Humano	$F1=AC+BD(A+C)$ $F2=(A \oplus C)D' + ((A \oplus C) \oplus B)D$ $F3= B \oplus D$	12 Compuertas (5 ANDs, 3 ORs, 3 XORs, 1 NOT)
BGA	$F1=AC+BD(A \oplus C)$ $F2=(A \oplus C) \oplus BD$ $F3= B \oplus D$	7 Compuertas (3 ANDs, 1 OR, 3XORs)
AS	$F1= (DB\ (A \oplus C)) \oplus AB$ $F2 =DB\ \oplus (A \oplus C)$ $F3= D \oplus B$	7 Compuertas (3 ANDs, 4 XORs)

Tabla 4.12 Comparación de resultados entre el AS, un BGA y un diseñador humano para el ejemplo 4.

La solución dada por el diseñador humano, ocupa casi el doble de compuertas que las soluciones dadas por el AS y el BGA.

Para obtener la solución del AS presentada aquí, fueron utilizados los parámetros que se muestran en la tabla 4.13. El tiempo que tardó el AS en realizar esa corrida fue 231 segundos, o sea, 3.85 minutos.

Parámetro	Valor
Tamaño de la Matriz	8 x 8
Factor de Evaporación	0.75
Tamaño de la Población	50
Número de Iteraciones	10
Semilla aleatoria	0.075103

Tabla 4.13 Parámetros utilizados para obtener la solución de la Figura 4.4.

4.5 Ejemplo 5 (Multiplicador de dos bits)

Entradas				Salida			
A	B	C	D	F1	F2	F3	F4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Tabla 4.14 Tabla de verdad del Multiplicador de 2 bits

El quinto ejemplo consiste en un multiplicador de 2 bits y consta de 4 entradas y 4 salidas. La tabla de verdad correspondiente a este circuito se muestra en la tabla 4.14.

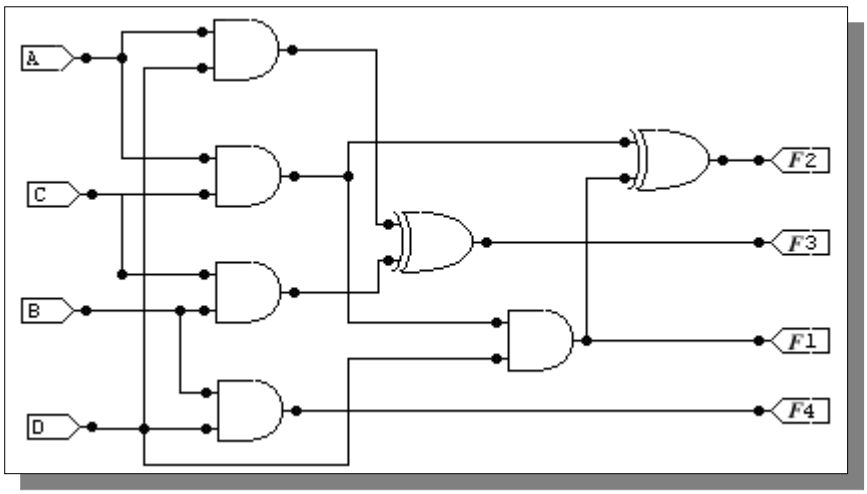


Figura 4.5 Circuito producido por el AS para el multiplicador de 2 bits.

Diseñador Humano	$F1=BDAC$ $F2=AC(BD)'$ $F3= BC \oplus AD$ $F4= BD$	8 Compuertas (6 ANDs, 1 XORs, 1 NOT)
Miller et al.	$F1=(AD \oplus BC)'(AD)$ $F2= (BD)'AC$ $F3= AD \oplus BC$ $F4= BD$	9 Compuertas (6 ANDs, 1 XORs, 2 NOTs)
MGA	$F1=BDAC$ $F2= BC \oplus (BDAC)$ $F3= BC \oplus AD$ $F4= BD$	7 Compuertas (5 ANDs, 2XORs)
AS	$F1=ACD$ $F2=((AC)D) \oplus (AC)$ $F3= AD \oplus CB$ $F4= BD$	7 Compuertas (5 ANDs, 2 XORs)

Tabla 4.15 Comparación de resultados entre el AS, un MGA y un diseñador humano para el ejemplo 4.

La comparación del resultado producido por el AS, un MGA, un diseñador humano (usando Mapas de Karnaugh) y una solución de Miller et al.[7] (obtenida mediante un algoritmo genético con representación cartesiana), se muestran en la tabla 4.15.

La figura 4.5 representa la gráfica del circuito dado como mejor solución por el AS, el cual consta de 7 compuertas. Esta solución consta de dos compuertas menos que la solución encontrada por Miller y una menos que la del diseñador humano, pero es igual que la mejor solución encontrada por el MGA.

Para obtener esta solución, se utilizaron los parámetros que se muestran en la tabla 4.16. El tiempo que tardó el AS en realizar esa corrida fue 3.03 minutos, o sea,182 segundos.

Parámetro	Valor
Tamaño de la Matriz	8 x 8
Factor de Evaporación	0.75
Tamaño de la Población	30
Número de Iteraciones	10
Semilla aleatoria	0.837228

Tabla 4.16 Parámetros utilizados para obtener la solución de la Figura 4.5.

4.6 Ejemplo 6 (Circuito de Katz de múltiples salidas)

El sexto ejemplo es un circuito que consta de 4 entradas y 3 salidas. La tabla de verdad que se muestra en la tabla 4.17, corresponde a este circuito.

La comparación del resultado producido por el AS, un BGA y dos diseñadores humanos (usando, el primero Mapas de Karnaugh y álgebra Booleana y el segundo el método Quine-McCluskey), se muestran en la tabla 4.18.

La representación gráfica del circuito dado como mejor solución por el AS se muestra en la figura 4.6. Dicha solución no es mejor (en términos del número de compuertas) que la producida por el MGA para este ejemplo. Sin embargo, sí resulta mejor que las producidas por ambos diseñadores humanos. La solución del MGA es de sólo 9 compuertas, mientras que las de los diseñadores humanos son de 19 y 13 compuertas respectivamente. La mejor solución encontrada por el AS consta de 11 compuertas.

Para obtener esa solución, se utilizaron los parámetros que se muestran en la tabla 4.19. El tiempo que tardó el AS en realizar esa corrida fue 8666 segundos, o sea, 2.41 horas. Como puede observarse, con este circuito el AS necesitó mucho más tiempo y de todas las corridas que se hicieron con este circuito y que se mencionarán en el siguiente capítulo, sólo en dos se encontraron soluciones con 11 compuertas.

Entradas				Salida		
A	B	C	D	F1	F2	F3
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	1	0
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	0	1

Tabla 4.17 Tabla de verdad del sexto ejemplo

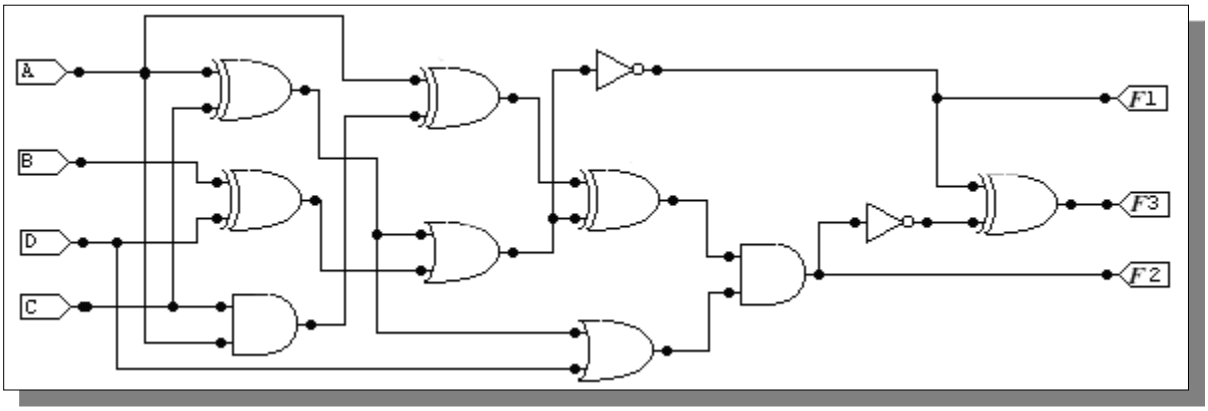


Figura 4.6 Circuito producido por el AS para el ejemplo 6.

Diseñador Humano 1	$F1 = (A \oplus C)'\ (B \oplus D)'$ $F2 = B'D\ (A' + C) + A'C$ $F3 = BD'\ (A + C') + AC'$	19 Compuertas (7 ANDs, 4 ORs, 2 XORs, 6 NOTs)
Diseñador Humano 2	$F1 = (A \oplus C)'\ (B \oplus D)'$ $F2 = A'C + (A \oplus C)'\ (B'D)$ $F3 = (F1 + F2)$	13 Compuertas (4 ANDs, 2 ORs, 2 XORs, 5 NOTs)
MGA	$F1 = ((B \oplus D)+(A \oplus C))'$ $F2 = F2 \oplus ((B \oplus D) + (A \oplus C))$ $F3 = ((B \oplus D)+(A \oplus C))(((A \oplus C)+(A \oplus B))\oplus C)$	9 Compuertas (2 ANDs, 3 ORs, 3 XORs, 2 NOTs)
AS	$F1 = ((A \oplus C) + (B \oplus D))'$ $F2 = ((CA) \oplus A) \oplus ((A \oplus C)+(B \oplus D))((A \oplus C)+D)$ $F3 = F1 \oplus F2'$	11 Compuertas (2 ANDs, 2 ORs, 5 XORs, 2 NOTs)

Tabla 4.18 Comparación de resultados entre el AS, un MGA y dos diseñadores humanos para el ejemplo 6.

Parámetro	Valor
Tamaño de la Matriz	12 x 12
Factor de Evaporación	0.50
Tamaño de la Población	50
Número de Iteraciones	50
Semilla aleatoria	0.026878

Tabla 4.19 Parámetros utilizados para obtener la solución de la figura 4.6.

Resumen

En este capítulo se mostraron los resultados obtenidos con nuestra aplicación al optimizar algunos circuitos. Los resultados fueron comparados con aquellos obtenidos por diseñadores humanos y con otras técnicas evolutivas como son el Algoritmo Genético Multiobjetivo y el Algoritmo Genético Binario.

Las soluciones obtenidas con nuestra aplicación superan a las soluciones encontradas con métodos tradicionales y son muy semejantes a las obtenidas con el MGA y el BGA.

En el próximo capítulo se presenta una análisis de la influencia de los parámetros del AS en su desempeño.

Referencias Bibliográficas

[1] Coello Coello, Carlos A.; Christiansen, Alan D. and Hernández Aguirre, Arturo, **Use of Evolutionary Techniques to Automate the Design of Combinational Circuits**, *International Journal of Smart Engineering System Design, Elsevier Science*, Vol. 2, No. 4, pp. 299-314, June 2000

[2] Coello Coello, Carlos A. and Hernández Aguirre, Arturo, **Evolutionary Multiobjective Design of Combinational Logic Circuits**, in Jason Lohn, Adrian Stoica, Didier Keymeulen & Silvano Colombano (editors), *Proceedings of the Second NASA/DoD Workshop on Evolvable Hardware*, pp. 161--170, IEEE Computer Society, Los Alamitos, California, July 2000.

[3] Coello Coello, Carlos A.; Alan D. Christiansen & Arturo Hernández Aguirre, **Towards Automated Evolutionary Design of Combinational Circuits**, Lania-RI-99-03.d, Laboratorio Nacional de Informática Avanzada, 1999.

[4] Coello Coello, Carlos A.; Zavala G. Rosa Laura; Mendoza G., Benito and Hernández Aguirre, Arturo, **Ant Colony System for the Design of Combinational Logic Circuits**, in Julian Miller, Adrian Thompson, Peter Thomson and Terence C. Fogarty (Eds.), *Evolvable Systems: From Biology to Hardware*, Edinburgh, Scotland, Springer-Verlag, pp. 21--30, April 2000.

[5] Coello Coello, Carlos A.; Zavala G. Rosa Laura; Mendoza G., Benito & Hernández Aguirre, Arturo, **On the Use of the Ant System to Design Electrical Circuits**, *X Congreso Latino-Iberoamericano de Investigación de Operaciones y Sistemas*, México, D.F., 4 al 8 de septiembre del 2000.

[6] Karnaugh M.. **A Map Method for Synthesis of Combinational Logic Circuits**. *Transactions of the AIEE, Communications and Electronics*, 72 (I):593--599, November 1953.

[7] Miller, J. F., Thomson, P., and Fogarty, T., **Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study**, In Quagliarella, D., Périaux, J., Poloni, C., and Winter, G. (eds), *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science*, pp. 105--131. Morgan Kaufmann, Chichester, England, 1998.

[8] McCluskey E. J. **Minimization of Boolean Functions**. *Bell Systems Technical Journal*, 35 (5):1417--1444, November 1956.

[9] Sasao Tsutomu, editor. **Logic Synthesis and Optimization.** *Kluwer Academic Press,* 1993.

[10] Katz, Randy H., **Contemporary logic design**. *Benjamin/Cummings Pub. Co*., Redwood City, Calif., 1994.

5 Análisis de la Influencia de los Valores de los Parámetros en el Desempeño del Algoritmo

Ya se ha hablado antes de los parámetros que utiliza esta aplicación. Idealmente, cada vez que quisiéramos resolver algún circuito, quisiéramos saber de antemano qué valores utilizar para dichos parámetros, de tal forma que con esos valores se pudiera llegar a la región factible (la región del problema donde hay soluciones válidas) para dicho circuito y de preferencia en el menor tiempo posible. En otras palabras, se desea obtener la mayor cantidad de soluciones válidas y en el menor tiempo posible.

El objetivo de este capítulo es entender el efecto que tienen los valores de los parámetros en el desempeño del algoritmo, para lo cual diseñamos un modelo experimental factorial, mediante el cual se pudiera analizar dicho efecto.

La diferencia entre un experimento y un experimento diseñado según Montgomery[1], es que un experimento es sólo una *prueba* o *ensayo*, mientras que un experimento diseñado, es una prueba o serie de pruebas en las cuales se inducen cambios en las variables de entrada de un proceso o sistema, de manera que sea posible observar e identificar las causas de los cambios en la respuesta de salida.

El proceso o sistema bajo estudio puede representarse por medio del modelo de la Fig. 5.1. Suele ser posible visualizar el proceso como una combinación de máquinas, métodos, personas y otros recursos que transforman alguna entrada (a menudo un material) en una salida que tiene una o más respuestas observables. Algunas de las variables del proceso x_1, x_2,..., x_p son controlables, mientras que otras z_1, z_2,..., z_q son incontrolables (aunque pueden ser controlables para los fines de una prueba). Entre los objetivos del experimento pueden incluirse:


Determinar cuáles variables tienen mayor influencia en la respuesta, *y*.


Determinar el mejor valor de las *x* que influye en *y*, de modo que *y* tenga casi siempre un valor cercano al valor nominal deseado.


Determinar el mejor valor de las *x* que influye en *y*, de modo que la variabilidad de *y* sea pequeña.


Determinar el mejor valor de las *x* que influye en *y*, de modo que se minimicen los efectos de las variables incontrolables z_1, z_2,..., z_q.

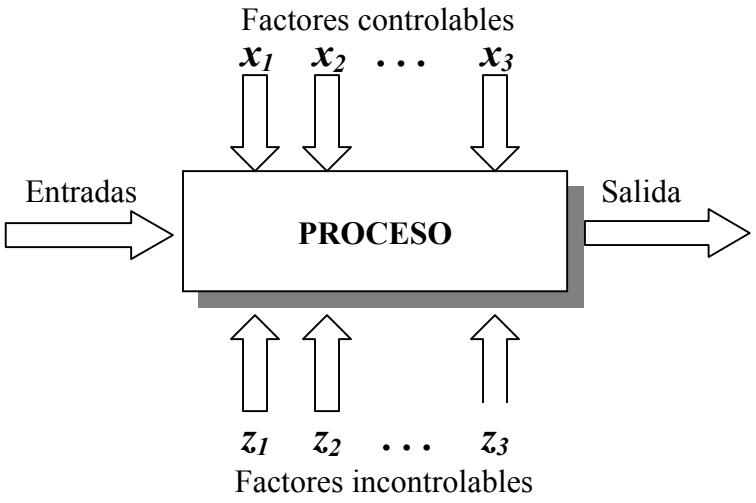


Figura 5.1 Modelo general de un proceso o sistema[1].

Por diseño factorial se entiende aquél en el que se investigan todas las posibles combinaciones de los niveles (valores) de los factores en cada ensayo completo o réplica del experimento. Y normalmente son los utilizados en problemas como el nuestro, en los que se desea estudiar el efecto producido por dos o más factores.

Para diseñar nuestro modelo experimental nos basamos en las directrices propuestas por Montgomery, para el diseño de experimentos, las cuales abarcan lo que se presenta en los siguientes apartados.

5.1 Comprensión y planteamiento del problema

Nuestra aplicación funciona mediante la combinación de subcircuitos (compuertas o pedazos de circuitos), que realizan las hormigas o agentes, guiadas en gran medida por los rastros que van dejando. Por lo anterior se sabe de antemano que entre más hormigas se tengan en *la población*, habrá mayor cantidad de oportunidades de combinar subcircuitos, en cada iteración, de tal forma que se espera que haya mayor porcentaje de convergencia. Pero como cada hormiga requiere de cierto tiempo para construir su ruta o circuito (elegir las combinaciones de compuertas o subcircuitos), también es claro que al aumentar *el tamaño de la población*, también aumentará el tiempo necesario para terminar una corrida.

Lo mismo pasa con *el número de ciclos o iteraciones*, el cual representa el número de veces que cada hormiga de la población intenta construir una ruta. Cuando este parámetro se aumenta, al igual que lo que sucede con el tamaño de la población, aumenta la cantidad de oportunidades de combinar subcircuitos, esperando tener un mayor porcentaje de

convergencia. Pero también se espera un incremento en el tiempo necesario para concluir una corrida.

La matriz representa el espacio donde se van colocando los subcircuitos que se van eligiendo, de tal forma que si *el tamaño de la matriz* aumenta, se tendrá más espacio para poner subcircuitos, y al tener más subcircuitos se tendrá también un mayor número de combinaciones posibles, con lo cual se espera tener mayor porcentaje de convergencia ya que cada hormiga tendrá que probar un mayor número de combinaciones (buscar más). Pero esto hará que una hormiga tarde más tiempo en construir su ruta o solución, lo que significa un incremento en el tiempo necesario para concluir una corrida.

De antemano podemos decir que *el factor de evaporación* no tiene ningún efecto sobre el tiempo, ya que sólo determina qué porcentaje del rastro de feromona se debe evaporar en cada iteración, lo cual implica la misma operación, para cualquier valor. Sin embargo, no se sabe de antemano, cuál es el efecto que pudiera tener sobre el porcentaje de convergencia, ya que de manera intuitiva sólo se puede pensar que si se evaporan mucho los rastros, la comunicación entre hormigas disminuye y si se evaporan muy poco, pudiera afectar el hecho de que perdure información que no sea relevante.

Dados los comentarios anteriores, las hipótesis que formulamos y que trataremos de sustentar son las que se presentan en la tabla 5.1.

Número	Descripción
1	El tamaño de la matriz, influye sobre el porcentaje de convergencia.
2	El tamaño de la matriz, influye sobre el tiempo necesario para terminar una corrida
3	El tamaño de la población, influye sobre el porcentaje de convergencia
4	El tamaño de la población, influye sobre el tiempo necesario para terminar una corrida.
5	El número de ciclos o iteraciones, influye sobre el porcentaje de convergencia
6	El número de ciclos o iteraciones, influye sobre el tiempo necesario para terminar una corrida.
7	El factor de evaporación, influye sobre el porcentaje de convergencia
8	El factor de evaporación NO influye sobre el tiempo necesario para terminar una corrida.

Tabla 5.1 Hipótesis de la influencia de los parámetros sobre el porcentaje de convergencia y tiempo.

5.2 Elección de factores y niveles

Los factores son las variables del experimento. En nuestro caso, se tenía claro que los factores son los parámetros de nuestra aplicación. Decidimos utilizar tres valores diferentes, los cuales representarán la variabilidad que cada parámetro pudiera tener, la tabla 5.2 muestra dichos valores.

Parámetro	Alias	Valor 1	Valor 2	Valor 3
Tamaño de la Matriz	Matriz	8 x 8	10 x 10	12 x 12
Factor de Evaporación	Evap	0.10	0.5	0.75
Tamaño de la Población	PopSize	10	30	50
Núm. de Ciclos o Iteraciones	Max_Cic	10	30	50

Tabla 5.2 Valores utilizados en los parámetros para los experimentos.

Con respecto al *tamaño de la matriz*, decidimos que el primer valor de la matriz fuese 8x8, ya que durante las primeras pruebas de nuestra aplicación fue uno de los tamaños de matriz más chicos con los que se lograron soluciones válidas casi con todos los circuitos que se usaron. Mientras que 12x12 fue un límite superior con el cual se consideró que el tiempo que se pudiera llevar una corrida de cualquier circuito, pudiera ser adecuado o viable como para repetir los experimentos. La idea era tener una referencia de lo que pasa cuando se aumenta el tamaño de la matriz, por lo que se decidió utilizar 10x10 como un valor intermedio para el tamaño de la matriz.

Con respecto al *factor de evaporación*, el valor 0.10, representa la evaporación casi nula de los rastros, mientras que 0.75, representa casi la evaporación total. Obviamente el 0.5, representa un valor intermedio.

Al igual que con el tamaño de la matriz, para decidir qué valores poner tanto para *el tamaño de la población como para el número de iteraciones*, se utilizaron los valores con los que se obtuvieron resultados válidos durante las primeras pruebas y que no representaran un costo excesivo de tiempo necesario para concluir una corrida. Por lo anterior se eligió 10 como mínimo y 50 como máximo; con el valor intermedio de 30.

5.3 Selección de la variable de respuesta

En nuestro caso lo que nos interesaba era saber qué tanto convergía nuestra aplicación con una cierta combinación de parámetros y el tiempo que se requería para terminar una corrida con dicha combinación.

Además, buscamos tratar de determinar cuál combinación era mejor en cuanto a porcentaje de convergencia y costo computacional. Por los que nuestras variables de respuesta son el *porcentaje de convergencia* y el *tiempo promedio necesario* para concluir una corrida. Cabe mencionar que el tiempo que se requiere para una corrida con cierta combinación de parámetros es constante, es decir, que no varía de una corrida a otra. Por ello, el tiempo promedio es el mismo que se requiere para cualquier corrida.

5.4 Elección del diseño experimental

Una vez decidido todo lo anterior, nuestro siguiente paso fue el diseño experimental que consistió en elegir con qué circuitos experimentar. Se decidió utilizar cinco circuitos los cuales, han sido utilizados por otros autores para probar sus propuestas y que además representan diferentes características en cuanto a entradas, salidas y complejidad. Dichos circuitos son:


El circuito de Sasao


El circuito de Katz de una salida


El sumador de dos bits


El multiplicador de dos bits


El Circuito de Katz de tres salidas

De todos estos circuitos ya se habló en el capítulo anterior, donde se muestran los mejores resultados obtenidos para cada uno de ellos.

Para cada circuito se decidieron realizar cuarenta corridas con cada combinación de valores en sus parámetros. Como son cuatro parámetros y tres valores para cada uno de ellos, tenemos 3^4 combinaciones, es decir 81 combinaciones de valores. Por lo tanto, tenemos

3240 corridas para cada circuito (81 combinaciones por 40 corridas). Como son 5 circuitos y 3240 corridas por cada uno, se tiene un total de 16200 corridas o experimentos.

El desarrollo de cada corrida o experimento, depende de la semilla aleatoria, que como se mencionó en el capítulo 2, inicializa un sistema generador de números aleatorios. Por lo que con el objetivo de que la semilla para cada corrida fuera elegida aleatoriamente, se utilizó una tabla de números aleatorios[2], la cual proporciona una serie de números elegidos previamente de manera aleatoria.

De cada corrida, se recuperó la mejor solución o ruta encontrada, además de lo siguiente:

 Aptitud

 Número de compuertas si es una solución válida o –1 si no lo es

 Iteración o ciclo en el que fue encontrada

 El número de violaciones si es que es una solución inválida o cero de lo contrario

 El tiempo necesario para terminar la corrida

Como ya se mencionó, el tiempo es el mismo para todas las corridas con la misma combinación de parámetros. Y como el tiempo es dependiente también de las características de la computadora donde se corra el programa, se tomó como referencia los tiempos obtenidos en una máquina con las siguientes características físicas y de software.

Procesador	Intel Pentium III a 550 Mhz.
Memoria RAM	128 Mbytes.
Disco Duro	13 Gbytes.
Memoria de Video	8 Mbytes.
Sistema Operativo	Microsoft Windows 98
Compilador	Borland C++ Builder 4

Tabla 5.3 Características de la computadora donde se hicieron los experimentos.

5.5 Realización del experimento

Una vez concluidas todas las corridas de cada circuito, se calculó lo siguiente, del conjunto de corridas obtenido por cada combinación de valores en los parámetros:


Tiempo promedio necesario para terminar una corrida


Porcentaje de convergencia. Es decir, el porcentaje de corridas que obtuvieron una solución válida.


El número de compuertas de la mejor solución obtenida de las 40 corridas

5.6 Resultados preliminares descriptivos

Con los resultados obtenidos de todas las corridas, se hizo un primer *análisis empírico*, para el cual se construyó un programa con el que se pueden graficar las cuarenta corridas de cada combinación de valores en los parámetros. De manera general, los resultados observados en las gráficas obtenidas, sugieren una tendencia similar del impacto de los valores de los parámetros con respecto al porcentaje de convergencia y tiempo, como la esperada dadas las hipótesis de la tabla 5.1, de lo cual se habla a continuación.

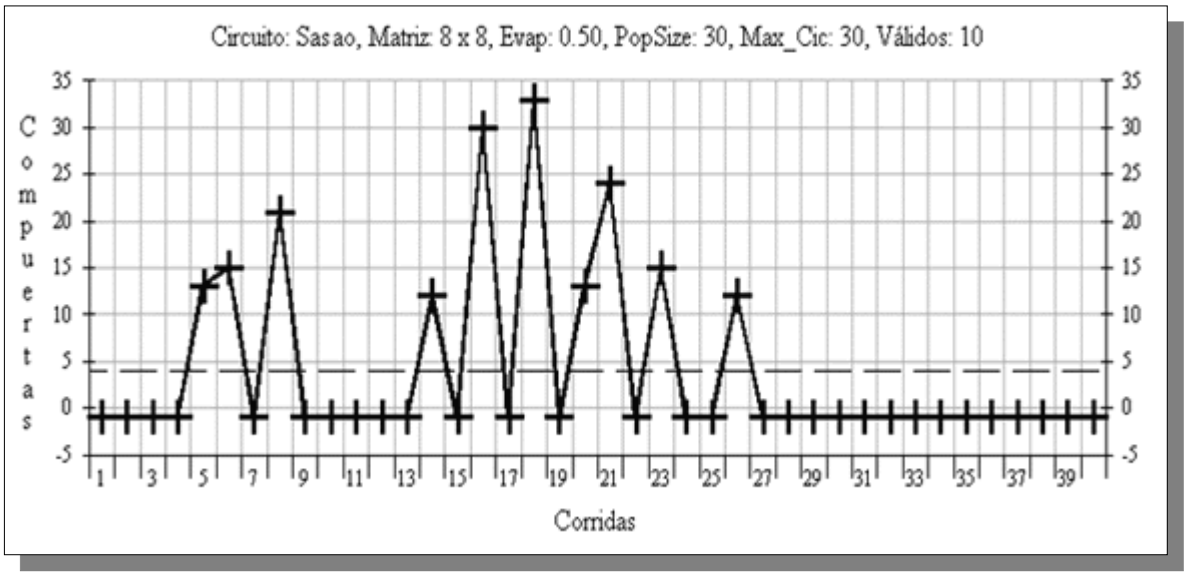
> *En las gráficas sólo son resultados válidos aquellos en que el número de compuertas es mayor que cero. Es decir aquellas corridas en las que el algoritmo no logro converger o encontrar una solución válida aparecerá abajo de cero compuertas.*

Tamaño de la matriz

A continuación se muestra el impacto observado del tamaño de la matriz sobre el porcentaje de convergencia y el tiempo necesario para concluir una corrida.

En la gráfica 5.1, se ilustra lo que sucedió con las cuarenta corridas del circuito de Sasao en las que se ocupó una matriz de 8 x 8, consiguiéndose únicamente 10 resultados válidos,

es decir, un porcentaje convergencia del 25% y un tiempo de 137 segundos para cada corrida (como se muestra en la tabla 5.4).

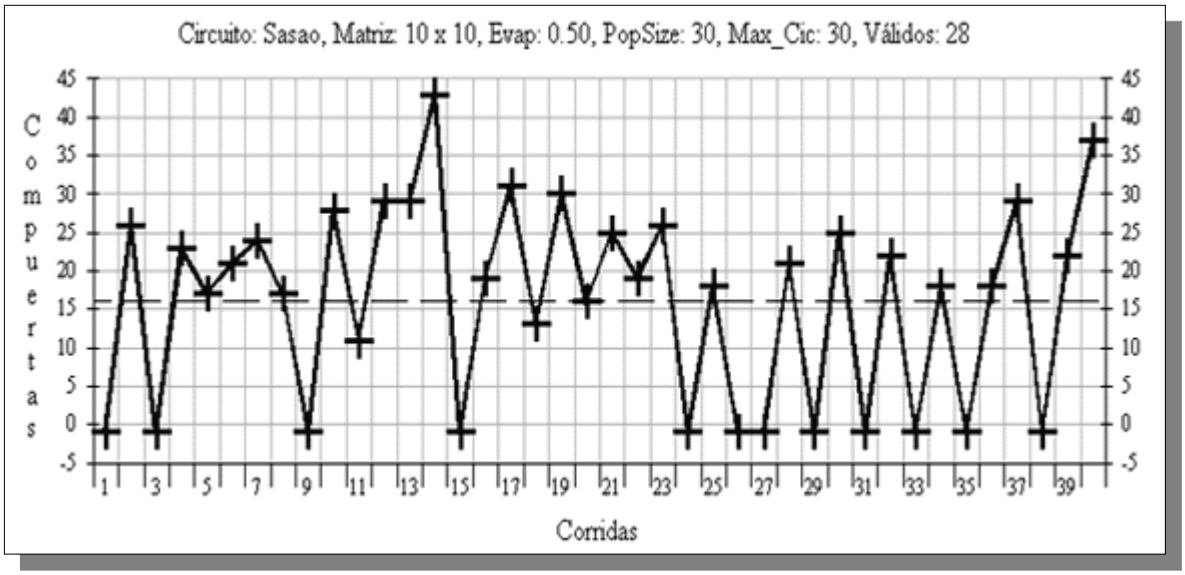


Gráfica 5.1 Resultados de 40 corridas con una Matriz de 8 x 8 con el circuito de Sasao .

Matriz	8 x 8
Evap	.50
PopSize	30
Max_Cic	30
Convergencia	25 %
Tiempo	137 seg.
Mejor Resultado(Compuertas)	12
Encontrado(Corrida)	26

Tabla 5.4 Parámetros y resultados de las corridas mostradas en la gráfica 5.1.

Ahora, en la gráfica 5.2 se muestran cuarenta corridas del mismo circuito en las que el único parámetro que se aumentó fue el tamaño de la matriz a 10 x 10. De esas corridas, 28 son válidas, como se puede apreciar en la tabla 5.5. El porcentaje de convergencia obtenido es del 70%. Es decir, aumentó con respecto a las corridas de la figura 5.1, pero también el tiempo aumentó de 137 a 411 segundos.

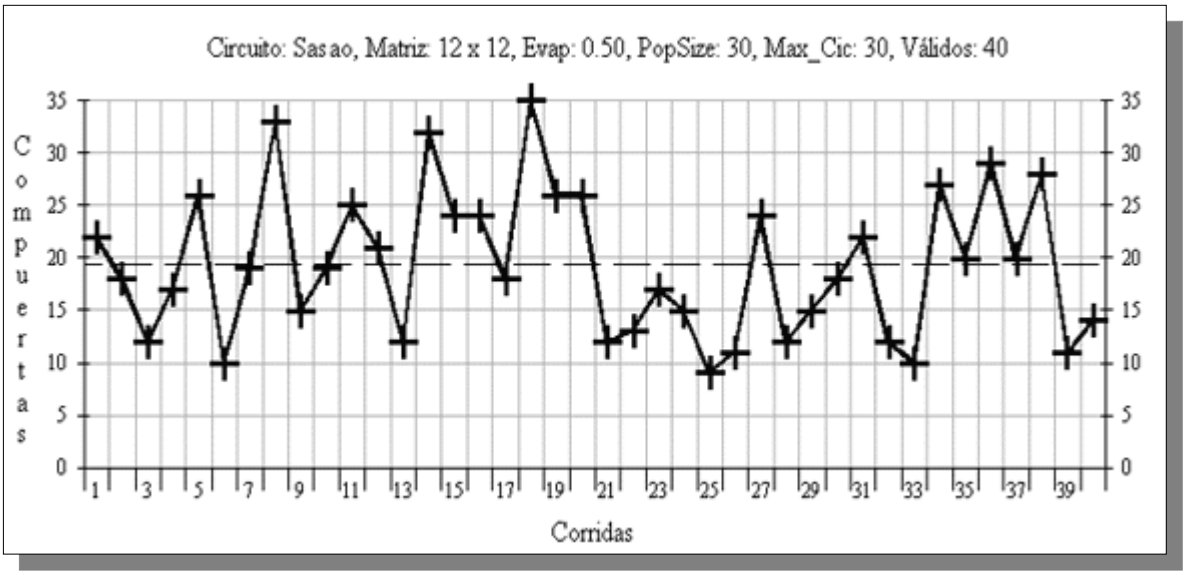


Gráfica 5.2 Resultados de 40 corridas con una Matriz de 10 x 10 con el circuito de Sasao .

Matriz	10 x 10
Evap	.50
PopSize	30
Max_Cic	30
%Convergencia	70
Tiempo	411
Mejor Resultado(Compuertas)	11
Encontrado(Corrida)	10

Tabla 5.5 Parámetros y resultados de las corridas mostradas en la gráfica 5.2.

En la gráfica 5.3, se muestran las corridas en las que se utilizó una matriz más grande que las utilizadas en las corridas de las gráficas anteriores. Al utilizar esta matriz de 12 x 12, el porcentaje de convergencia que se obtuvo fue del 100%. En otras palabras, todas las corridas encontraron una solución válida. Pero al igual que sucedió al aumentar el tamaño del matriz de 8 x 8 a 10 x 10, el tiempo necesario para terminar una corrida aumentó. Ahora el tiempo es de 1012 segundos.



Gráfica 5.3 Resultados de 40 corridas con una Matriz de 12 x 12 con el circuito de Sasao .

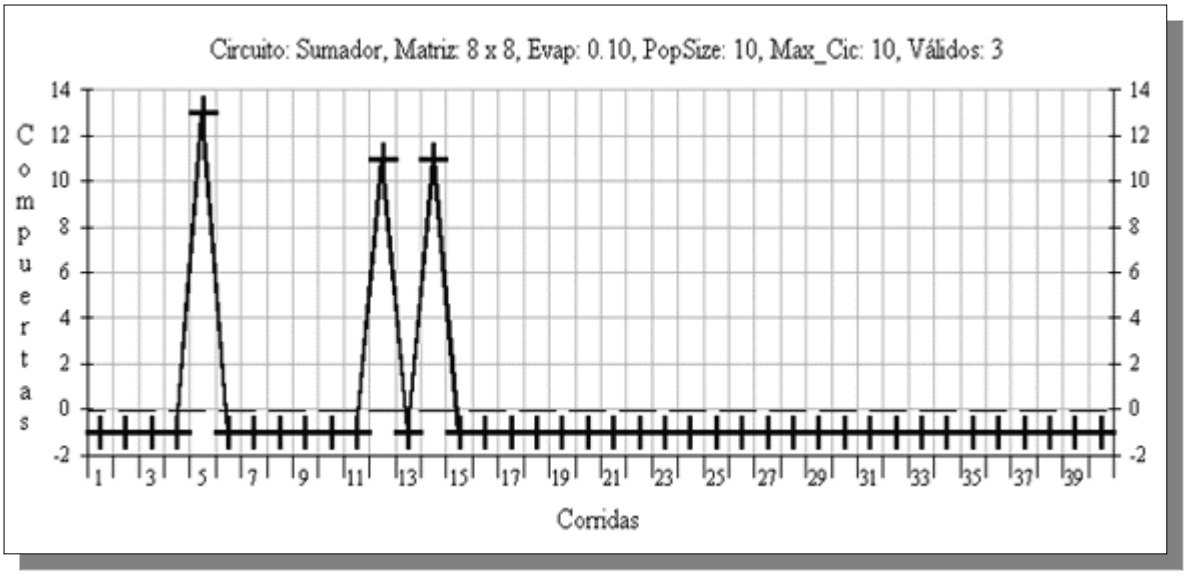
Matriz	12 x 12
Evap	.50
PopSize	30
Max_Cic	30
%Convergencia	100
Tiempo	1012
Mejor Resultado(Compuertas)	9
Encontrado(Corrida)	24

Tabla 5.6 Parámetros y resultados de las corridas mostradas en la gráfica 5.3.

Con los ejemplos anteriores se podría suponer que el tamaño de la matriz tiene un fuerte impacto en el porcentaje de convergencia. Es decir, que al aumentar el tamaño de la matriz se obtienen más soluciones válidas, pero como consecuencia, el tiempo necesario para concluir una corrida también aumenta considerablemente. Esto es lo que se esperaba dadas las hipótesis 1 y 2 de la tabla 5.1.

Factor de Evaporación

Como se explicó en el capítulo 3, el factor de evaporación o Evap es el porcentaje con que se evaporan los rastros dejados por las hormigas o agentes. A continuación se muestra el impacto de este parámetro en cuanto al porcentaje de convergencia y el tiempo necesario par concluir una corrida, sugerido por las gráficas.



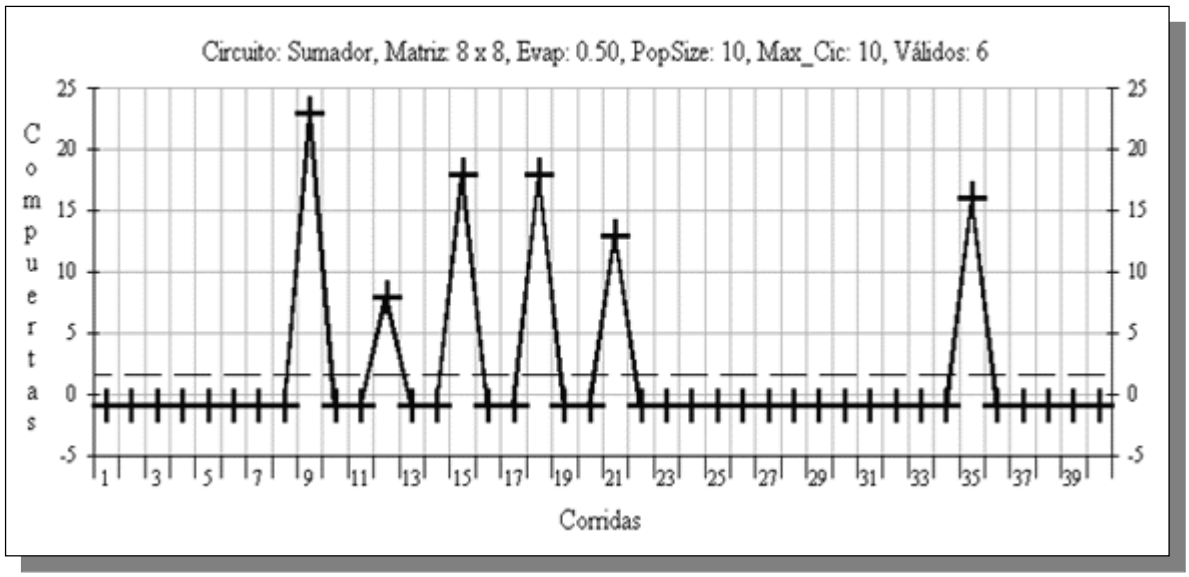
Gráfica 5.4 Resultados de 40 corridas con un *factor de evaporación* de 0.10 con el sumador de dos bits.

Matriz	8 x 8
Evap	0.10
PopSize	10
Max_Cic	10
%Convergencia	7.5
Tiempo	47
Mejor Resultado(Compuertas)	11
Encontrado(Corrida)	12

Tabla 5.7 Parámetros y resultados de las corridas mostradas en la gráfica 5.4.

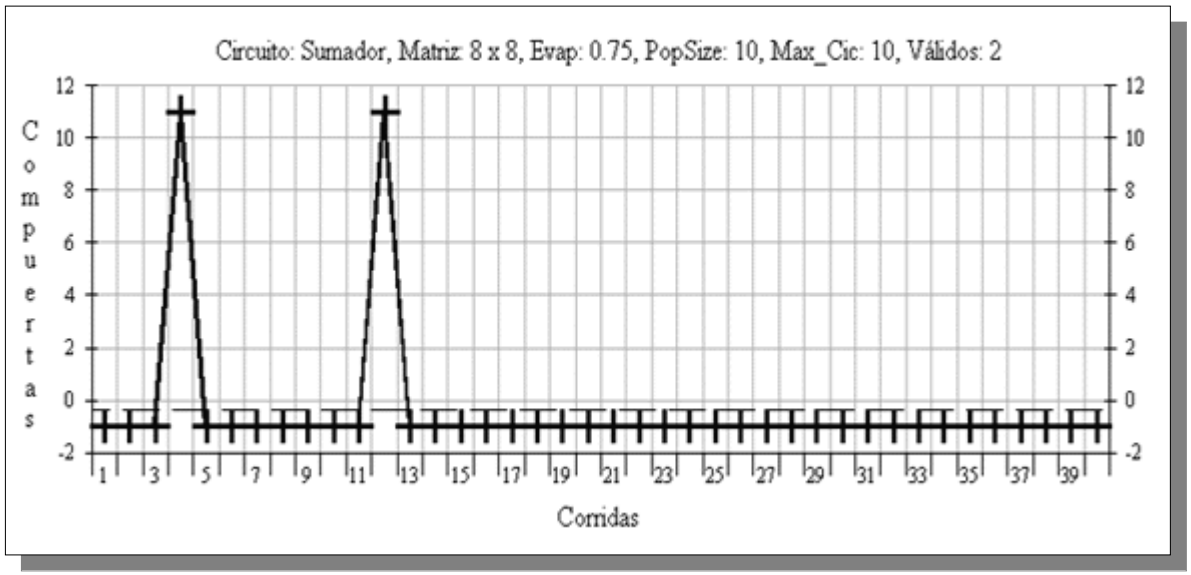
En la gráfica 5.4, se observan 40 corridas del sumador de dos bits, en las cuales se utilizaron los valores más pequeños utilizados en todos los experimentos. Estos valores son mostrados en la tabla 5.6. Con 0.10 en el Factor de Evaporación sólo el 10% de los valores de feromonas se evaporan. Con este valor, se logró obtener un porcentaje de convergencia del 7.5%, encontrando sólo tres corridas con soluciones válidas y para cada corrida fueron necesarios 47 segundos. La mejor solución obtenida fue de 11 compuertas.

En la gráfica 5.5, se muestran lo que sucedió, al aumentar únicamente el *factor de evaporación* de 0.10 a 0.50. Ahora se tiene un porcentaje de convergencia del 15%, es decir, sólo 6 corridas de 40 encontraron soluciones válidas, siendo el tiempo necesario para cada corrida el mismo que en los casos anteriores, 47 segundos, como muestran en las tablas 5.7 y 5.8. En estos experimentos el mejor resultado obtenido fue un circuito con 8 compuertas.



Gráfica 5.5 Resultados de 40 corridas con un *factor de evaporación* de 0.50 con el sumador de dos bits.

Matriz	8 x 8
Evap	0.50
PopSize	10
Max_Cic	10
%Convergencia	15
Tiempo	47
Mejor Resultado(Compuertas)	8
Encontrado(Corrida)	12



Gráfica 5.6 Resultados de 40 corridas con un *factor de evaporación* de 0.75 con el sumador de dos bits.

Matriz	8 x 8
Evap	0.75
PopSize	10
Max_Cic	10
%Convergencia	5
Tiempo	47
Mejor Resultado(Compuertas)	11
Encontrado(Corrida)	12

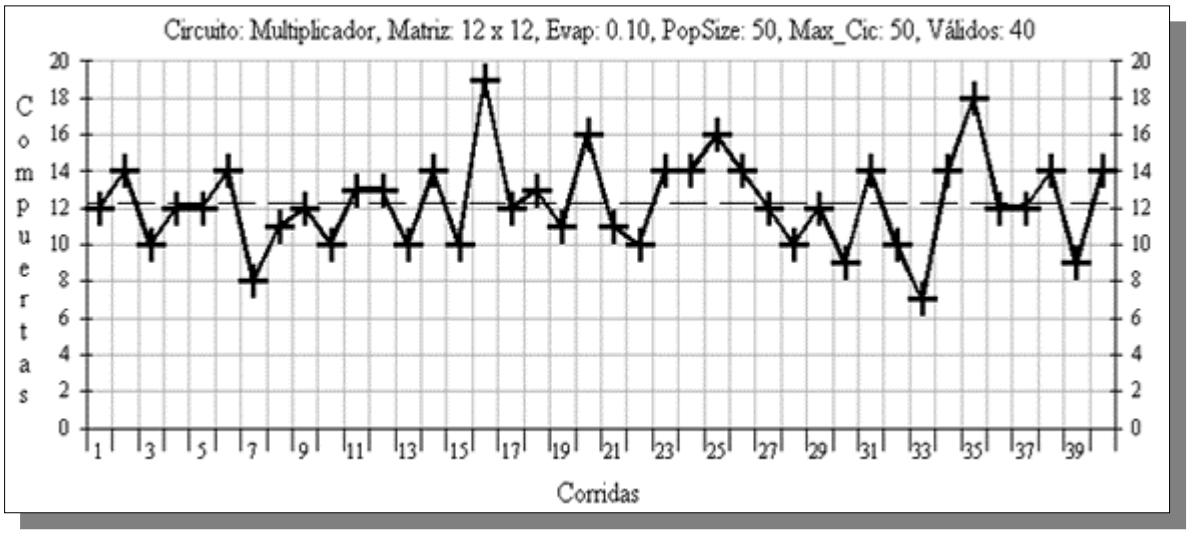
Tabla 5.9 Parámetros y resultados de las corridas mostradas en la gráfica 5.6.

En la gráfica 5.6 se muestra lo que sucedió al aumentar el Factor de Evaporación a 0.75. El porcentaje de convergencia bajó al 5%. El tiempo necesario para cada corrida no se movió, es decir, se quedó en 47. Las dos corridas con resultados válidos, obtuvieron una solución con 11 compuertas.

De manera general el comportamiento del factor de evaporación en todos los circuitos y combinaciones de valores fue el mismo. Los resultados sugieren que el tiempo, aparentemente, no se ve afectado por incrementos de este parámetro. En cuanto al porcentaje de convergencia, de manera general en todos los experimentos y con todos los circuitos, el incremento del valor de este parámetro no lo afecta notoriamente, como es el caso del tamaño de la matriz. Podríamos decir que la hipótesis 8 de la tabla 5.1 se cumple, mientras que la 7 no.

Sin embargo en la mayoría de los experimentos, se consiguieron más soluciones con 0.50. Esto tiene cierta relación con lo que recomiendan Stützle y Dorigo[3] para este parámetro, ya fue con el que encontraron mejores resultados en sus experimentos.

Por otro lado, aunque el Factor de Evaporación, dadas las gráficas anteriores, no se considere que tenga un impacto muy fuerte en el porcentaje de convergencia, se puede apreciar también, que sí parece tener cierto efecto en la calidad de las soluciones. Para ilustrar esto se muestran algunas corridas del multiplicador de dos bits.



Gráfica 5.7 Resultados de 40 corridas con un *factor de evaporación* de 0.10 con el multiplicador de dos bits.

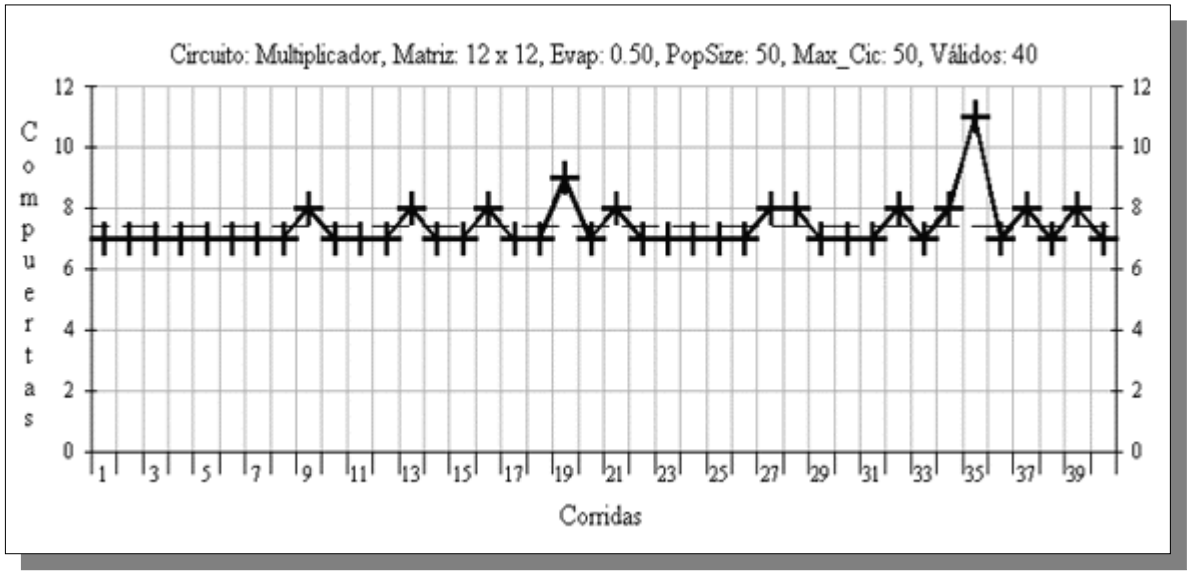
Matriz	12 x 12
Evap	.10
PopSize	30
Max_Cic	30
%Convergencia	100
Tiempo	11296
Mejor Resultado(Compuertas)	7
Encontrado(Corrida)	32

Tabla 5.10 Parámetros y resultados de las corridas mostradas en la gráfica 5.7.

En la figura 5.7 se muestran los resultados del sumador de dos bits con un *factor de evaporación* de 0.10. Todas las corridas obtuvieron soluciones válidas, o sea que el porcentaje de convergencia fue del 100%, debido a que se utilizó una matriz más grande que para las corridas mostradas anteriormente y otros valores para los demás parámetros (tabla 5.10). La mejor solución encontrada tiene 7 compuertas, pero sólo una con estas

características fue encontrada. La media de las soluciones en cuanto a compuertas es de 12.3, como se muestra en la línea punteada de la gráfica 5.7.

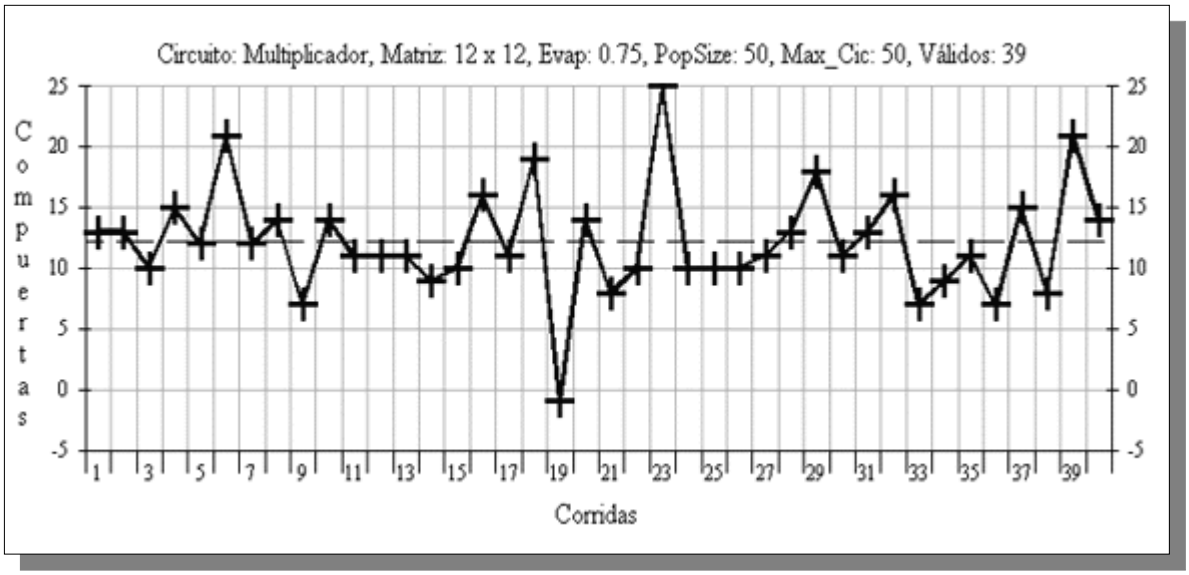
Ahora en la gráfica 5.8, se muestran las corridas en las cuales se ocuparon los mismos valores en los parámetros a excepción del *factor de evaporación*, esta vez se ocupó 0.50, como se muestra en las tablas 5.10 y 5.11. Al igual que en la gráfica 5.7, las 40 corridas obtuvieron soluciones válidas (100% de porcentaje de convergencia), pero la diferencia es que en la gráfica 5.8 hay 28 soluciones con 7 compuertas y la media de las soluciones en cuanto a compuertas es de 7.4. En otras palabras las soluciones encontradas con los parámetros de la tabla 5.11 son mejores, ya que proponen circuitos en su mayoría con menos compuertas que las soluciones con los parámetros de la tabla 5.10.



Gráfica 5.8 Resultados de 40 corridas con un *factor de evaporación* de 0.50 con el multiplicador de dos bits.

Matriz	12 x 12
Evap	0.50
PopSize	30
Max_Cic	30
%Convergencia	100
Tiempo	11296
Mejor Resultado(Compuertas)	7
Encontrado(Corrida)	1

Tabla 5.11 Parámetros y resultados de las corridas mostradas en la gráfica 5.8.



Gráfica 5.9 Resultados de 40 corridas con un *factor de evaporación* de 0.75 con el multiplicador de dos bits.

Matriz	12 x 12
Evap	0.75
PopSize	30
Max_Cic	30
%Convergencia	97.5
Tiempo	11296
Mejor Resultado(Compuertas)	7
Encontrado(Corrida)	9

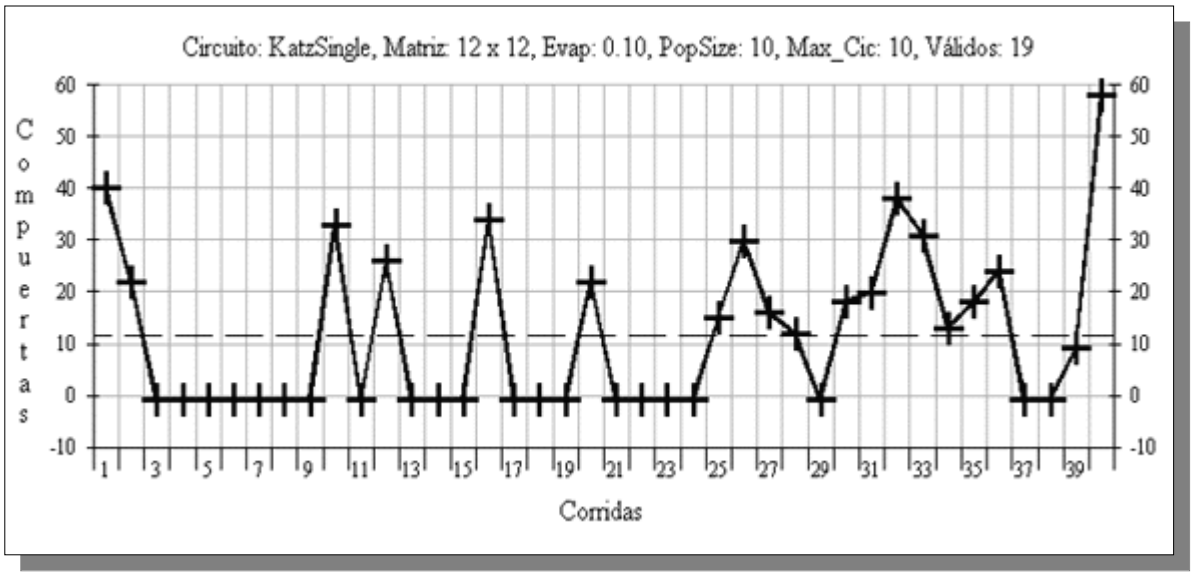
Tabla 5.12 Parámetros y resultados de las corridas mostradas en la gráfica 5.9.

Los resultados obtenidos al incrementar el factor de evaporación a 0.75, que se muestran en la gráfica 5.9, denotan una baja en el porcentaje de convergencia al igual que la calidad de las soluciones. Ahora se tiene un porcentaje de convergencia del 97.5%, ya que una de las cuarenta corridas no encontró una solución válida. Además, sólo tres de las soluciones son de 7 compuertas que fue la mejor solución obtenida (tabla 5.12), más que las corridas de la gráfica 5.7, pero mucho menos que las corridas de la gráfica 5.8.

En resumen, las observaciones derivadas de las gráficas sugieren, que el aumento o disminución del factor de evaporación no afecta al tiempo necesario para una corrida y tampoco afecta significativamente el porcentaje de convergencia, pero es claro también que los mejores resultados obtenidos fueron en su mayoría con 0.50 y por tanto, que con este valor generalmente se obtiene mejor calidad en las soluciones.

Tamaño de la Población

Como ya se mencionó en el capítulo 3, el tamaño de población o PopSize se refiere al número de hormigas o agentes que intervienen en la búsqueda de soluciones. A continuación se muestra el impacto de este parámetro en cuanto al porcentaje de convergencia y el tiempo necesario par concluir una corrida, observado através de las gráficas.

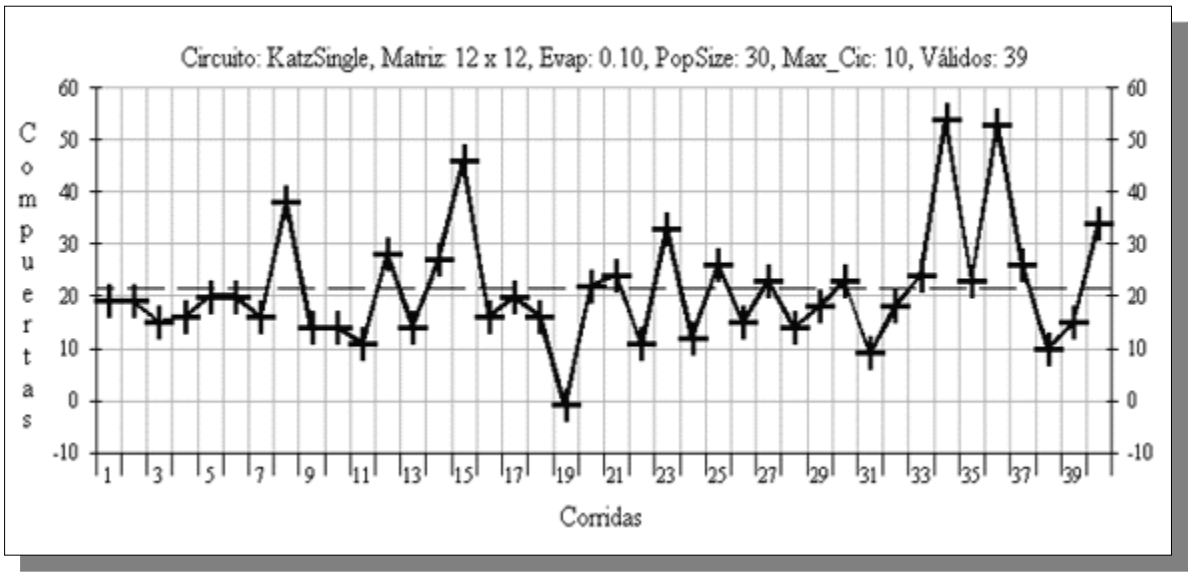


Gráfica 5.10 Resultados de 40 corridas con 10 como tamaño de población con el circuito de Katz de una salida.

Matriz	12 x 12
Evap	0.10
PopSize	10
Max_Cic	10
%Convergencia	47.5
Tiempo	143
Mejor Resultado(Compuertas)	9
Encontrado(Corrida)	39

Tabla 5.13 Parámetros y resultados de las corridas mostradas en la gráfica 5.10.

El 47.5% de los resultados de las corridas mostradas en la gráfica 5.10 son válidos, esto quiere decir que sólo 19 corridas de 40 convergieron. El tiempo necesario para terminar una corrida con los valores en los parámetros de la tablas 5.13 es de 143 segundos.

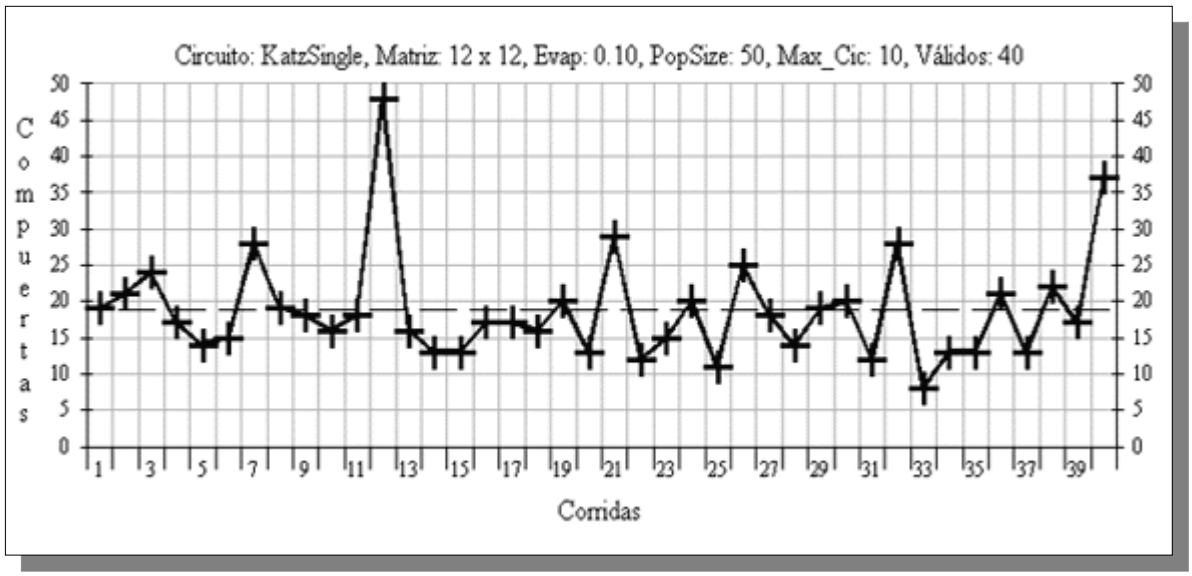


Gráfica 5.11 Resultados de 40 corridas con 30 como tamaño de población con el circuito de Katz de una salida.

Matriz	12 x 12
Evap	0.10
PopSize	10
Max_Cic	10
%Convergencia	97.5
Tiempo	425
Mejor Resultado(Compuertas)	9
Encontrado(Corrida)	31

Tabla 5.14 Parámetros y resultados de las corridas mostradas en la gráfica 5.11.

En cuanto se aumentó el tamaño de la población de 10 a 30 hormigas, el porcentaje de convergencia aumentó de 47.5 a 97.5, con los mismos valores en los demás parámetros. El tiempo también aumentó casi al triple (de 143 a 425 segundos). Sólo una corrida de 40 no encontró una solución válida.



Gráfica 5.12 Resultados de 40 corridas con 50 como tamaño de población con el circuito de Katz de una salida.

Matriz	12 x 12
Evap	0.10
PopSize	10
Max_Cic	10
%Convergencia	100
Tiempo	712
Mejor Resultado(Compuertas)	8
Encontrado(Corrida)	33

Tabla 5.15 Parámetros y resultados de las corridas mostradas en la gráfica 5.12.

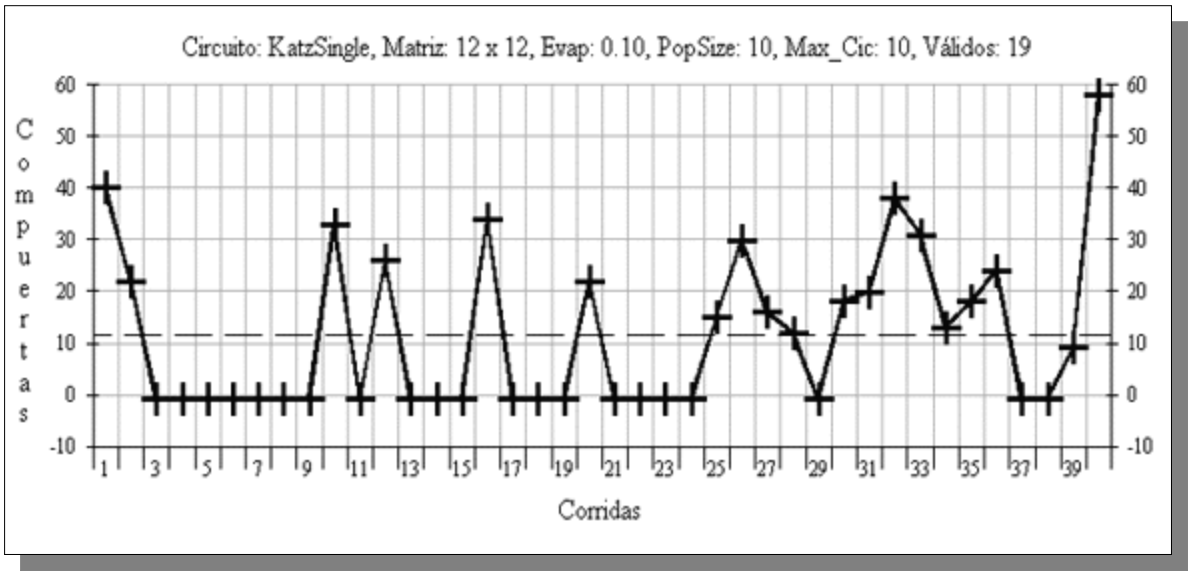
Todas las corridas encontraron una solución válida en cuanto se incrementó a 50 el número de hormigas de la población. Pero ahora el tiempo fue de 712 segundos para cada corrida.

Se puede apreciar que el tamaño de la población, aparentemente, tiene un fuerte impacto con respecto al porcentaje de convergencia, pero como es de imaginarse, entre más hormigas se tengan más tiempo tardará en terminarse una corrida, tal y como lo sugieren las hipótesis 3 y 4 de la tabla 5.1.

Algo que se puede apreciar con respecto al tiempo es que aumenta equitativamente con respecto al número de hormigas que se incremente. Como se puede ver en estos ejemplos cuando se incremento de 10 a 30 hormigas aumentó casi al triple y cuando se aumentó a 50, el tiempo fue casi 5 veces el tiempo que se tardaba con 10 hormigas.

Número de Iteraciones

El número de iteraciones o Max_Cic, es el número de veces que una hormiga trata de encontrar una solución, como ya se explicó en el capítulo 3.

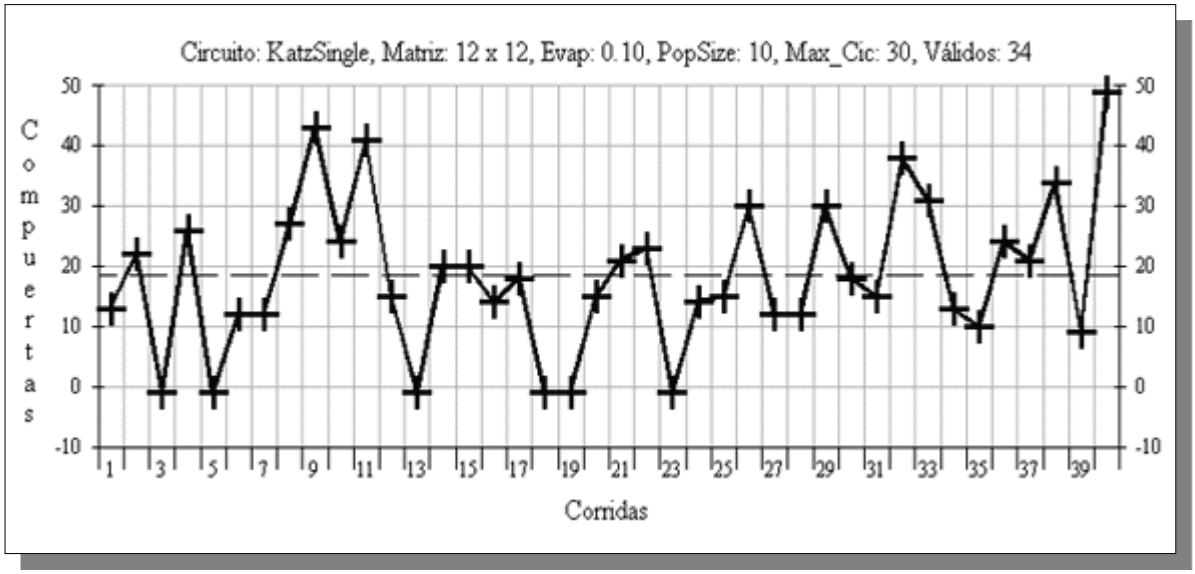


Gráfica 5.13 Resultados de 40 corridas con un valor con Max_Cic 10 para el circuito de Katz de una salida.

Matriz	12 x 12
Evap	0.10
PopSize	10
Max_Cic	10
%Convergencia	47.5
Tiempo	143
Mejor Resultado(Compuertas)	9
Encontrado(Corrida)	39

Tabla 5.16 Parámetros y resultados de las corridas mostradas en la gráfica 5.2.

Las corridas y obviamente los resultados son los mismos que los de la gráfica 5.10 y la tabla 5.13. El 47.5% las corridas mostradas en la gráfica 5.13 son válidos, esto quiere decir que sólo 19 corridas de 40 convergieron. El tiempo necesario para terminar una corrida con los valores en los parámetros de la tablas 5.16 es de 143 segundos.



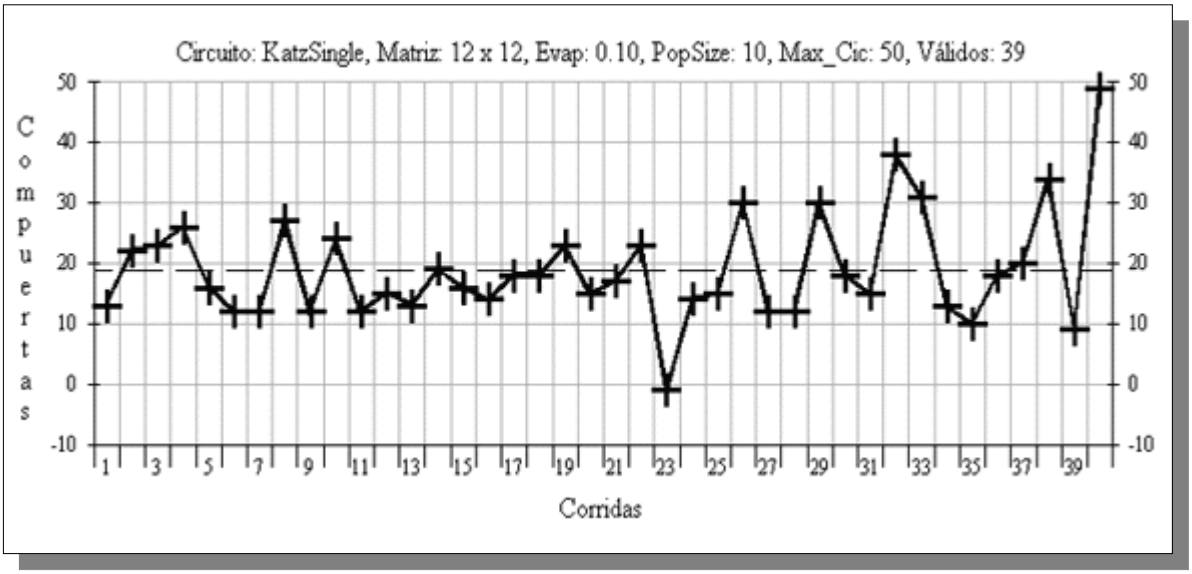
Gráfica 5.14 Resultados de 40 corridas con un Max_Cic 30 para el circuito de Katz de una salida.

Matriz	12 x 12
Evap	0.10
PopSize	10
Max_Cic	10
%Convergencia	85
Tiempo	425
Mejor Resultado(Compuertas)	9
Encontrado(Corrida)	39

Tabla 5.17 Parámetros y resultados de las corridas mostradas en la gráfica 5.14.

Al incrementar el número de iteraciones de 10 (gráfica 5.13) a 30 (gráfica 5.14) se obtuvieron 34 soluciones válidas. Es decir que el porcentaje de convergencia aumentó de 47.5% a 85% y el tiempo aumentó casi al triple de 143 a 425 segundos. Este es un comportamiento similar al del PopSize.

Esto mismo sucede en las corridas de la gráfica 5.14, en las que se aumentó el número de iteraciones a 50, como se muestra en la tabla 5.18. Con este cambio se obtuvieron 39 soluciones válidas, dándonos un porcentaje de convergencia del 97.5% y aumentando el tiempo casi a cinco veces de lo que se requería con Max_Cic igual a 10.



Gráfica 5.15 Resultados de 40 corridas con un Max_Cic 10 para el circuito de Katz de una salida.

Matriz	12 x 12
Evap	0.10
PopSize	10
Max_Cic	10
%Convergencia	97.5
Tiempo	714
Mejor Resultado(Compuertas)	9
Encontrado(Corrida)	39

Tabla 5.18 Parámetros y resultados de las corridas mostradas en la gráfica 5.15.

En general, el número de iteraciones se comporta casi igual que el tamaño de la población, como lo sugieren las hipótesis 5 y 6 de la tabla 5.1. Esto tiene sentido, ya que al aumentar al doble el número de iteraciones se está dando una doble oportunidad a cada hormiga de construir un circuito, lo cual es equivalente a que en lugar de aumentar el número de iteraciones se aumente el número de hormigas, ya que se producirían el mismo número de intentos por construir una ruta o solución. La única diferencia es que con más hormigas, la cantidad de feronoma depositada en los rastros aumenta, lo cual produce esa pequeña diferencia en el porcentaje de convergencia.

5.1 Análisis estadístico de los resultados en cuanto al tiempo

Con el fin de estudiar el efecto de los valores de los parámetros con respecto al porcentaje de convergencia, se implementó un análisis estadístico conocido como "Análisis de Varianza (ANOVA) del diseño factorial"[4].

En general, el propósito del análisis ANOVA es probar diferencias significativas entre medias para grupos de variables. Es decir, si las variables o factores involucrados en el experimento así como la interacción de éstos, afectan la respuesta de la variable dependiente. En otras palabras, el método de análisis de varianza, trata de analizar la variación de una respuesta (variable dependiente, en nuestro caso: tiempo) y de asignar porciones o componentes de esta variación a cada una de las variables independientes. El razonamiento se basa en que las variables de respuesta o dependientes se modifican por la variación de las variables independientes (en este caso: Matriz, Evap, PopSize y Max_Cic) y su posible interacción o combinación.

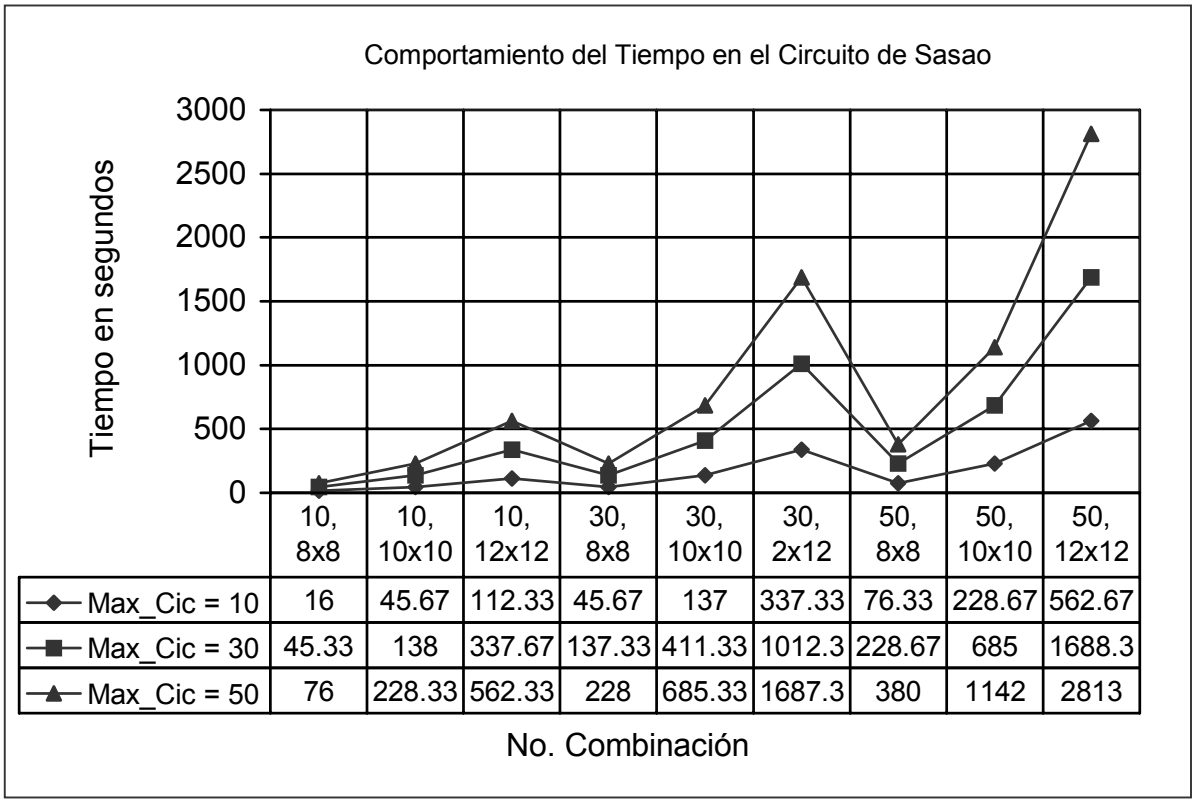
Para el análisis de los datos se hizo empleo de los paquetes computacionales estadísticos *Systat* y *Statistica* y las tablas de resultados generadas, para cada circuito se pueden consultar en el Apéndice A.

	Parámetros		***Tiempo***		
Núm. de Combinación	**PopSize**	**Matriz**	**Max_Cic 10**	**Max_Cic 30**	**Max_Cic 50**
1	**10**	**8x8**	16.00	45.33	76.00
2	**10**	**10x10**	45.67	138.00	228.33
3	**10**	**12x12**	112.33	337.67	562.33
4	**30**	**8x8**	45.67	137.33	228.00
5	**30**	**10x10**	137.00	411.33	685.33
6	**30**	**12x12**	337.33	1012.33	1687.33
7	**50**	**8x8**	76.33	228.67	380.00
8	**50**	**10x10**	228.67	685.00	1142.00
9	**50**	**12x12**	562.67	1688.33	2813.00
Medias por Max_Cic			**173.52**	**520.44**	**866.93**

Tabla 5.19 Medias de Tiempo (en segundos) para el circuito de Sasao, en las combinaciones de Matriz y PopSize, para cada Max_Cic.

Una vez recuperados todos los datos, producidos por nuestro modelo de experimentación, y nuestro análisis estadístico, procedimos a interpretar el efecto de los parámetros sobre el *tiempo*, que como esperábamos, se veía afectado conforme aumentaba Max_Cic, Matriz y PopSize, mientras que el valor de Evap, no reflejaba influencia sobre él. Por lo cual calculamos las medias de tiempo por cada valor de Max_Cic, para cada combinación de PopSize y Matriz. El comportamiento observado fue el mismo de manera general y a continuación se comenta utilizando como ejemplo el circuito de Sasao.

En general, a medida que aumentan los ciclos, el tiempo invertido en promedio tiende a aumentar (ver la última fila de la tabla 5.19). Lo mismo podemos observar dentro de cada ciclo, a medida que aumenta el tamaño de la matriz o el de PopSize, el tiempo promedio invertido en cada corrida es más alto.



	10, 8x8	10, 10x10	10, 12x12	30, 8x8	30, 10x10	30, 2x12	50, 8x8	50, 10x10	50, 12x12
Max_Cic = 10	16	45.67	112.33	45.67	137	337.33	76.33	228.67	562.67
Max_Cic = 30	45.33	138	337.67	137.33	411.33	1012.3	228.67	685	1688.3
Max_Cic = 50	76	228.33	562.33	228	685.33	1687.3	380	1142	2813

Gráfica 5.16 Medias de Tiempo (en segundos) para el circuito de Sasao, correspondientes a las combinaciones de Matriz y PopSize, para cada Max_Cic de la Tabla 5.19.

También se pudo apreciar que el tiempo aumentó de manera constante conforme aumentó Max_Cic. Es decir, por cada ciclo aumentó una cantidad de tiempo fijo (el tiempo que tarda un ciclo), como se muestra en la última fila (Medias por Max_Cic) de la tabla 5.19. Con Max_Cic = 10, el promedio fue 173.52, al aumentar Max_Cic a 30 (al triple), el tiempo promedio también se triplicó. Para Max_Cic = 50 (5 veces Max_Cic = 10), el tiempo promedio fue de 866.93, es decir 5 veces el tiempo promedio para Max_Cic = 10.

La misma proporción de aumento en el tiempo, sucede al aumentar el tamaño de población. Esto es debido, a que tanto Max_Cic y PopSize aumentan el número de intentos de construir una ruta o solución. Esto se puede apreciar tanto en la gráfica 5.16, como en la tabla 5.19.

5.2 Análisis estadístico de los resultados en cuanto al porcentaje de convergencia

Para analizar el efecto de los parámetros sobre el porcentaje de convergencia al igual que para el tiempo, se realizaron análisis de varianza para cada nivel o valor de la variable Max_Cic (10, 30, 50) para la variable dependiente *Porcentaje de Convergencia* con las variables independientes Matriz, PopSize y Evap. De modo que se obtuvieron 3 análisis por circuito (uno por cada valor de Max_Cic), con el fin de:

Probar si realmente los valores o niveles de los parámetros involucrados, y la combinación de éstos, influyen en la media del porcentaje de convergencia.

Y si del punto anterior se encontraban resultados estadísticamente significativos, identificar la combinación de los niveles de los parámetros que proporcione, la media más alta del porcentaje de convergencia en el menor tiempo, es decir, la combinación que llegue a la región factible del problema, con el menor costo en cuanto a tiempo.

De modo que los juegos de hipótesis a probar por circuito, ciclo, variable dependiente y par de variables independientes en nuestro estudio quedaron de la siguiente manera:

a) Ho: No existe diferencia significativa entre las medias de los niveles del parámetro *i*.

   Ha: Al menos un par de los niveles del parámetro *i* es significativamente diferente entre sí.

b) Ho: No existe diferencia significativa entre las medias de los niveles del parámetro *j*.

Ha: Al menos un par de los niveles del parámetro *j* es significativamente diferente entre sí.

c) Ho: No existe diferencia significativa entre las medias de las combinaciones de los niveles de los parámetros *ij*.

Ha: Al menos un par de las combinaciones es significativamente diferente entre sí.

Donde *i* y *j* son diferentes y pueden ser: PopSize, Matriz ó Evap.

Para todas las hipótesis se fijó un nivel de significancia (denominado α normalmente en éste tipo de análisis)[1] de 0.01. Para cada juego de hipótesis, se rechaza la hipótesis nula Ho, si P calculado (derivado del cuadro de ANOVA del diseño factorial que proporcionan las herramientas de cómputo), es menor que 0.01. En caso contrario no se rechaza. Los cuadros obtenidos del análisis ANOVA del diseño factorial de cada circuito se pueden consultar en el apéndice B.

Si la hipótesis nula (Ho) se rechaza, se procede a la aplicación del método conocido como Comparación de Medias con el fin de detectar cual de éstas es significativamente diferente al resto. Los resultados de este método para las hipótesis que se rechazaron en el ANOVA del diseño factorial pueden consultarse en el Apéndice B.

Resultados para el circuito de Sasao.

Para el circuito de Sasao, las variables *Matriz* y *PopSize* influyen en el *porcentaje de convergencia* en todos los niveles de Max_Cic, mientras que la variable independiente *Evap* mostró tener influencia sólo en el caso donde Max_Cic = 30. Las tablas del análisis de varianza generadas por ciclo pueden ser consultadas en el Apéndice A.

Dado que el objetivo principal de la aplicación del análisis de varianza era identificar aquella o aquellas combinaciones por ciclo que proporcionen la media más alta en el porcentaje de convergencia , se presentan las tablas de medias así como las gráficas para las combinaciones de los niveles de los parámetros tamaño de la Matriz y PopSize por Max_Cic, y Evaporación para Max_Cic=30, que son las combinaciones que presentan un fuerte impacto sobre el porcentaje de convergencia.

[1] El valor de α, lo que realmente determina es la confianza con la que se va a trabajar. Es decir que cuando se fija un valor de 0.01, lo que realmente está implicando es que se desea trabajar con un nivel de confianza del 99%. Es decir, que con un 99% de confianza, se rechaza o no una hipótesis. Cabe mencionar que 0.01 es el valor que comúnmente se usa para este tipo de análisis.

	% Convergencia		
Parámetro / **Nivel**	**0.10**	**0.50**	**0.75**
Evap	**70.81**(1580.78)	**74.44**(1581.33)	64.17

Tabla 5.20 Medias del porcentaje de convergencia del circuito de Sasao, en cada nivel de Evap, para Max_Cic = 30.

En la tabla 5.20 se puede observar que el porcentaje de convergencia más alto ocurre con 0.50, aunque éste no tenga diferencia significativa con la media de 0.10, siendo la más alta menor al 75%. En la misma tabla, aparece entre paréntesis la cantidad de tiempo promedio necesaria para terminar una corrida (en segundos). Algo importante que se puede notar en esta tabla, es que cuando los rastros se evaporan mucho, como es el caso 0.75, el porcentaje de convergencia, disminuye.

En la tabla 5.21 se muestran las medias de la combinación de los niveles tamaño de Matriz y PopSize.

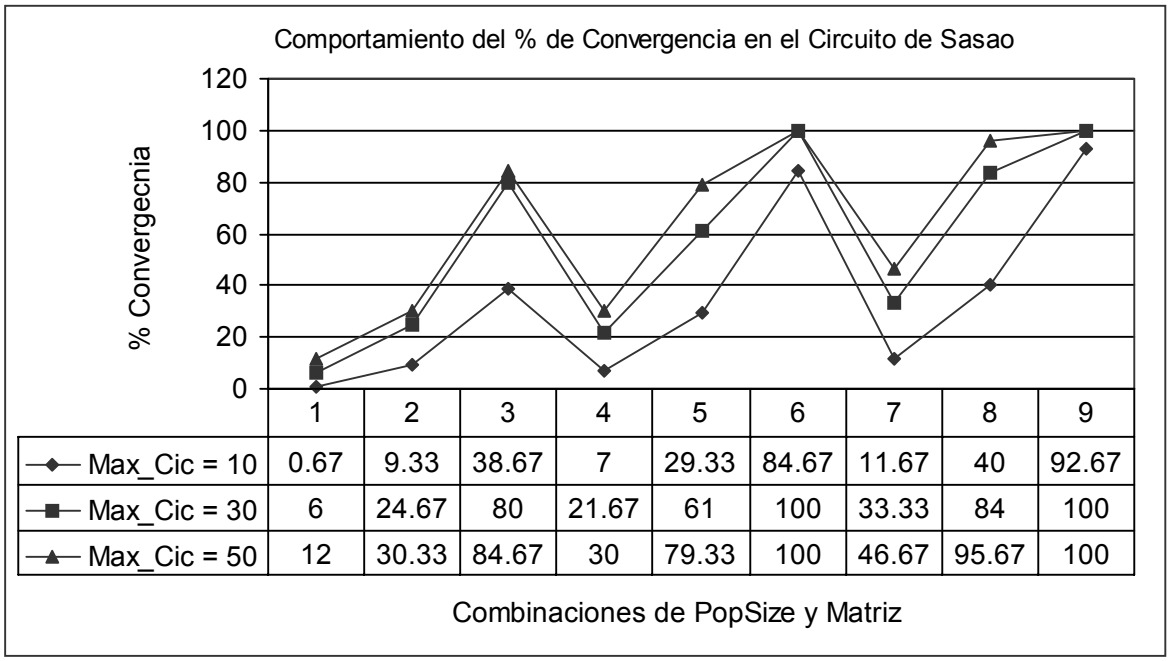
No. Combi.	Parámetros		%Convergencia		
	Matriz	PopSize	Max_Cic 10	Max_Cic 30	Max_Cic 50
1	**8x8**	**10**	0.67	6.00	12.00
2	**10x10**	**10**	9.33	24.67	30.33
3	**12x12**	**10**	38.67	80.00	**84.67**(562.33)
4	**8x8**	**30**	7.00	21.67	30.00
5	**10x10**	**30**	29.33	61.00	**79.33**(685.33)
6	**12x12**	**30**	**84.67**(337.33)	**100.00**(1012.33)	**100.00**(1687.33)
7	**8x8**	**50**	11.67	33.33	46.67
8	**10x10**	**50**	40.00	84.00	**95.67**(1142.00)
9	**12x12**	**50**	**92.67**(562.67)	**100.00**(1688.33)	**100.00**(2813.00)

Tabla 5.21 Medias del porcentaje de convergencia del circuito de Sasao, en las combinaciones de Matriz y PopSize, para cada Max_Cic.

En general, a medida que aumenta Max_Cic, los promedios de los porcentajes de convergencia tienden a aumentar, como podemos observar dentro de cada valor de

Max_Cic, en la tabla 5.21. Lo mismo sucede a medida que aumenta el tamaño de la Matriz para un PopSize fijo: la media del porcentaje de convergencia aumenta. En la misma tabla aparece entre paréntesis el tiempo en segundos de las combinaciones que logran mayor porcentaje de convergencia en promedio.

En la gráfica de 5.17 podemos observar como a medida que Max_Cic, Matriz y PopSize aumentan, el porcentaje de convergencia de las diferentes combinaciones de valores, también aumenta, así como el número de combinaciones que obtienen un porcentaje de convergencia cerca del 100%. De tal manera que las combinaciones 6 y 9 , que representan las combinaciones (PopSize=30, Matriz=12x12) y (PopSize=50, Matriz 12x12), respectivamente, logran obtener una media del 100%, tanto en Max_Cic = 30 como en Max_Cic=50.



	1	2	3	4	5	6	7	8	9
Max_Cic = 10	0.67	9.33	38.67	7	29.33	84.67	11.67	40	92.67
Max_Cic = 30	6	24.67	80	21.67	61	100	33.33	84	100
Max_Cic = 50	12	30.33	84.67	30	79.33	100	46.67	95.67	100

Gráfica 5.17 Medias del porcentaje de convergencia del circuito de Sasao, con las combinaciones de Matriz y PopSize, de la Tabla 5.21, para cada Max_Cic.

Si bien, *Evap* tuvo influencia en el porcentaje de convergencia, las medias de sus factores no llegan al 75% de convergencia. En cambio, ciertas combinaciones de Matriz y PopSize exceden este valor.
Como ya se mencionó y se puede observar tanto en la gráfica 5.17 y en la tabla 5.21, hay varias combinaciones de Matriz y PopSize que logran alcanzar un porcentaje de convergencia del 100% en promedio. De todas estas combinaciones, la que requiere una

menor cantidad de tiempo para concluir una corrida (1012.33 segundos) con este circuito, es la combinación de Matriz = 12x12 y PopSize = 30 en Max_Cic = 30. Con respecto al factor de evaporación, podríamos decir que en este circuito, se podría utilizar 0.10 ó 0 .50, ya que fueron con los valores que se obtuvo mayor porcentaje de convergencia y además son estadísticamente equivalentes.

De tal forma que podemos decir que para el circuito de Sasao, los valores mínimos necesarios de los parámetros, para que nuestra aplicación logre un buen porcentaje de convergencia, son los implicados en la combinación antes mencionada y que además, con esa combinación logramos obtener el mínimo de tiempo con un buen porcentaje de convergencia en promedio.

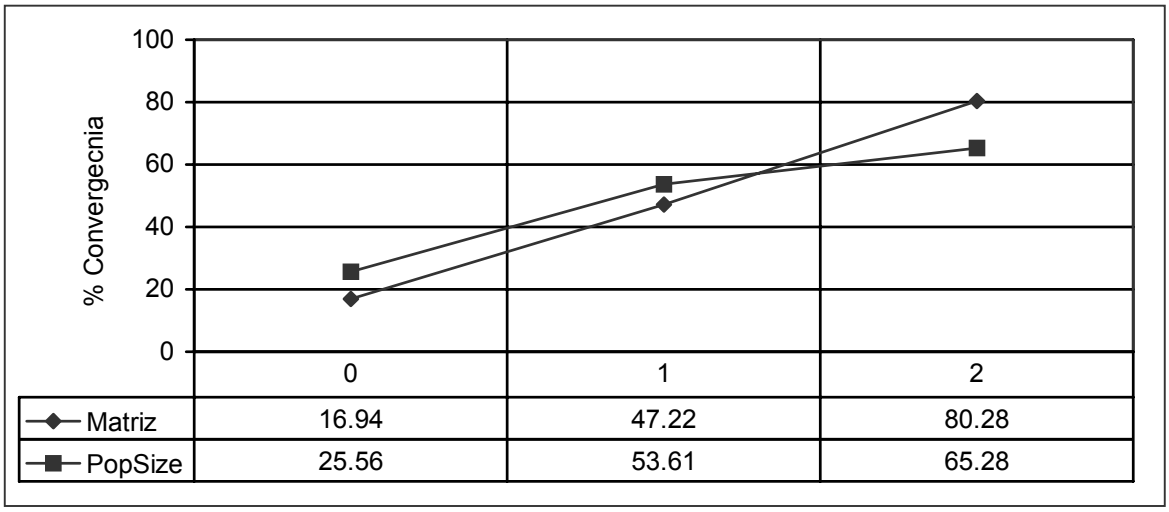
Resultados para el circuito de Katz de una salida.

Con el circuito de Katz de una salida, a diferencia del circuito anterior, Evap no mostró tener influencia en el porcentaje de convergencia en ningún valor de Max_Cic. Además en este circuito, con Max_Cic = 10, Matriz y PopSize, de manera individual, indicaron tener influencia en el porcentaje de convergencia, mientras que su combinación no manifestó dicha influencia. En los demás valores de Max_Cic, la combinación de Matriz y PopSize, tuvo influencia sobre el porcentaje de convergencia.

	% Convergencia		
Parámetro / Niveles	**0**	**1**	**2**
Matriz	16.94	47.22	**80.28**(426.44)
Popsize	25.56	**53.61**(217.67)	**65.28**(364.78)

Tabla 5.22 Medias del porcentaje de convergencia del circuito de Katz de una salida, para Matriz y PopSize, individualmente, con Max_Cic = 10.

La tabla 5.22, muestra las medias del porcentaje de convergencia de Matriz y PopSize para Max_Cic = 10, donde 0, 1 y 2, representan los niveles 8x8, 10x10 y 12x12, para Matriz y 10, 30 y 50 para PopSize, respectivamente. Como se puede apreciar en dicha tabla, a medida que se aumentan los valores tanto de Matriz como PopSize, el porcentaje de convergencia también aumenta.



	0	1	2
Matriz	16.94	47.22	80.28
PopSize	25.56	53.61	65.28

Gráfica 5.18 Medias del porcentaje de convergencia del del circuito de Katz de una salida, con Matriz y PopSize de manera individual, correspondientes a los datos de la Tabla 5.22, con Max_Cic = 10.

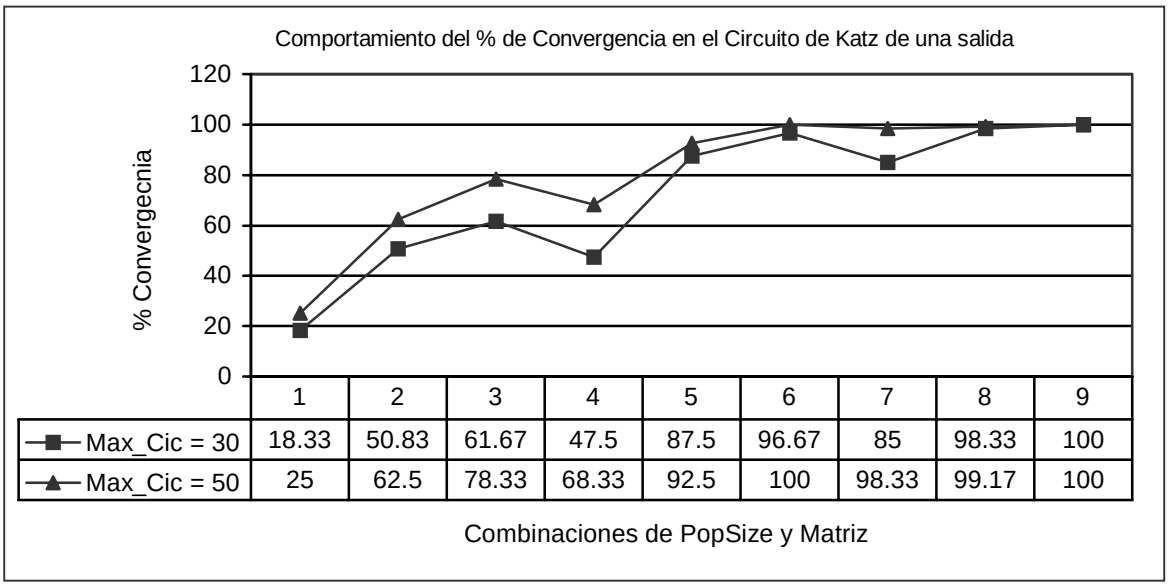
El valor más alto obtenido por Matriz de manera individual fue del 80.28% en el nivel 2 (12x12), mientras que para PopSize fue 65.28% en 50 (nivel 2), que de acuerdo a los datos del análisis, no tienen diferencia significativa con el obtenido en el nivel 1, que es del 53.61%. Algo interesante y que se puede observar en la gráfica 5.18, es que en los niveles 0 y1, PopSize obtiene mejores resultados que Matriz, pero en el nivel 2, Matriz lo supera 15%.

En la tabla 5.23, podemos observar, que a medida que aumenta Max_Cic, los promedios de los porcentajes de convergencia tienden a aumentar. Lo mismo sucede con el tamaño de la población y el tamaño de la Matriz.

En la gráfica 5.18 podemos observar cómo a medida que Max_Cic, Matriz y PopSize aumentan, el porcentaje de convergencia de las diferentes combinaciones de valores, también aumenta, así como el número de combinaciones que obtienen un porcentaje de convergencia en promedio cerca del 100%.

No. Combi.	Parámetros		% Convergencia	
	Matriz	PopSize	Max_Cic 30	Max_Cic 50
1	**8x8**	**10**	18.33	25.00
2	**8x8**	**30**	50.83	62.50
3	**8x8**	**50**	61.67	**78.33(**479.67**)**
4	**10x10**	**10**	47.50	68.33
5	**10x10**	**30**	**87.50(**520.33**)**	**92.50(**984.00**)**
6	**10x10**	**50**	**96.67(**863.67**)**	**100.00(**1440.00**)**
7	**12x12**	**10**	**85.00(**426.33**)**	**98.33(**711.33**)**
8	**12x12**	**30**	**98.33(**1279.33**)**	**99.17(**2133.67**)**
9	**12x12**	**50**	**100.00(**2121.67**)**	**100.00(**3558.00**)**

Tabla 5.23 Medias del porcentaje de convergencia del circuito de Katz de una salida, en las combinaciones de Matriz y PopSize, para Max_Cic 30 y 50.



Gráfica 5.19 Medias del porcentaje de convergencia del circuito de Katz de una salida, con las combinaciones de Matriz y PopSize, de la Tabla 5.22, para Max_Cic 30 y 50.

Las combinaciones que proporciona las medias de convergencia más altas se muestran en la tabla 5.23, en negritas. A diferencia del circuito de Sasao, con el circuito de Katz de una salida, hay más combinaciones de Matriz, PopSize, en Max_Cic 30 y 50 que logran un porcentaje de convergencia promedio cercano al 100%, por lo que consideramos que este circuito, es menos complicado, desde el punto de vista de la convergencia, aunque el tiempo requerido es mayor. Por ejemplo en la combinación Matriz 12x12, PopSize 50, tanto en Max_Cic 30 como en Max_Cic 50, requieren más tiempo en el de Sasao que en el de Katz de un salida.

Como puede observarse tanto en la gráfica 5.19 como en la tabla 5.23, hay tres combinaciones que logran en promedio el 100%, pero de esas tres, la combinación que requiere menos tiempo(1440 segundos) para concluir una corrida es en Max_Cic = 50, Matriz = 10x10 y PopSize = 50. En el caso del *factor de evaporación*, de acuerdo a los resultados del análisis de varianza factorial, se podría utilizar cualquiera de los valores, ya que no influye significativamente en el porcentaje de convergencia, en este nivel de Max_Cic.

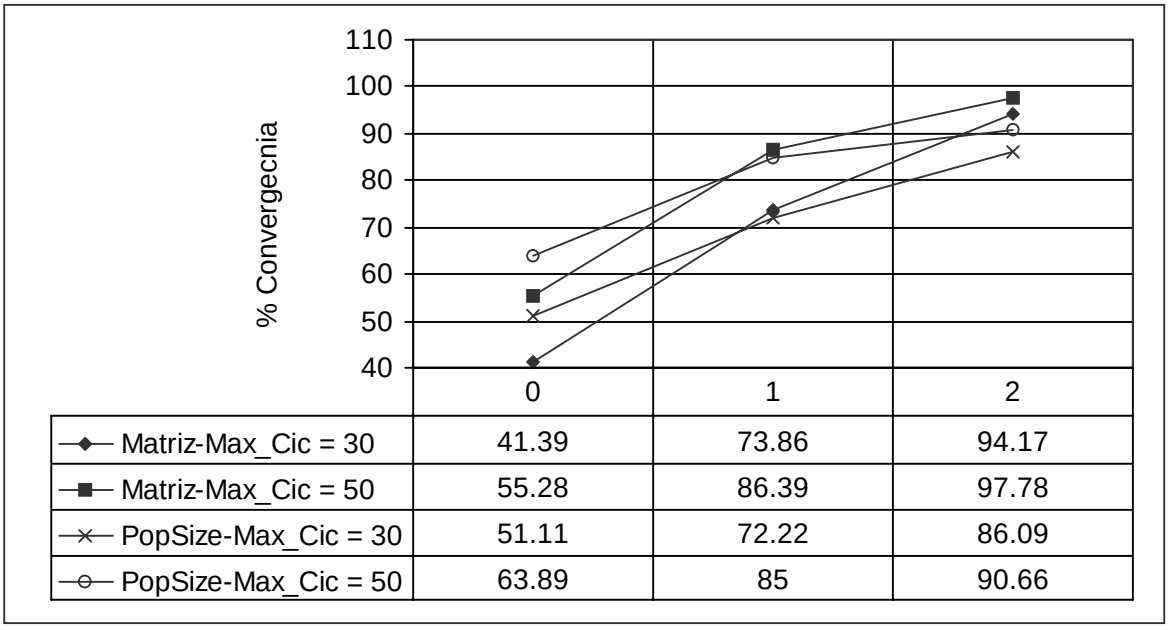
Resultados para el Sumador de dos bits.

Los resultados obtenidos del análisis de varianza con el Sumador de dos bits, al igual que en los circuitos anteriores, se detectó que sólo las variables *Matriz* y *PopSize* influyen en el *porcentaje de convergencia* y que la *evaporización* mostró no tener ningún impacto en dicha variable dependiente. Sólo con Max_Cic = 10 la interacción de Matriz y PopSize demostró tener influencia, mientras que con Max_Cic = 30 y Max_Cic = 50, Matriz y PopSize, tuvieron influencia, pero sólo de manera independiente, es decir no su interacción, tal y como se muestra en la tabla 5.24.

Parámetro/ Niveles	**% Convergencia**	
Matriz	**Max_Cic=30**	**Max_Cic=50**
8x8	41.39	55.28
10x10	73.86	**86.39**(2080.22)
12x12	**94.17**(3079.11)	**97.78**(5171.88)
PopSize		
10	51.11	63.89
30	**72.22**(1581.33)	**85.00**(5171.89)
50	**86.09**(2634.56)	**90.66**(4383.56)

Tabla 5.24 Medias del porcentaje de convergencia del sumador de dos bits, para Matriz y PopSize, de manera individual, con Max_Cic 30 y 50.

Como se puede observar en la tabla 5.24 y en la gráfica 5.20, con respecto a Matriz, los mejores resultados encontrados son con 12x12, que incluso son mejores que los de PopSize PopSize 50, que también son los más altos para ese parámetro. Podríamos decir que con Max_Cic 30 y 50, al aumentar el tamaño de la matriz o bien el PopSize se obtendrán resultados satisfactorios. Aunque el promedio de las medias no llegan al 100%, se acercan mucho.



	0	1	2
Matriz-Max_Cic = 30	41.39	73.86	94.17
Matriz-Max_Cic = 50	55.28	86.39	97.78
PopSize-Max_Cic = 30	51.11	72.22	86.09
PopSize-Max_Cic = 50	63.89	85	90.66

Gráfica 5.20 Medias del porcentaje de convergencia del sumador de dos bits, para Matriz y PopSize, de manera individual, con Max_Cic 30 y 50.

Parámetros		% Convergencia
Matriz	**Popsize**	**Ciclo 10**
8x8	**10**	9.17
8x8	**30**	10.83
8x8	**50**	20.00
10x10	**10**	13.33
10x10	**30**	47.50
10x10	**50**	58.33
12x12	**10**	44.17
12x12	**30**	**79.17**(1026.67)
12x12	**50**	**95.83**(1712.33)

Tabla 5.25 Medias del porcentaje de convergencia del sumador de dos bits, para las combinaciones de Matriz y PopSize, con Max_Cic = 10.

De acuerdo con el análisis, y como se puede apreciar en la tabla 5.25, podríamos decir que con el sumador de dos bits, un Max_Cic de 10, necesitaríamos aumentar ambas cosas (Matriz y PopSize) para obtener resultados cercanos al 100% en promedio.

Con lo comentado anteriormente, podríamos decir que si alguien quisiera hacer algunas pruebas de nuestra aplicación con el sumador de dos bits, se le recomendaría utilizar una matriz de 12x12, un Max_Cic de 30 y un PopSize de 30 y cualquier valor en Evap, ya que de acuerdo al análisis ANOVA, este parámetro no influye de manera significativa en la convergencia de nuestra aplicación con dicho circuito.

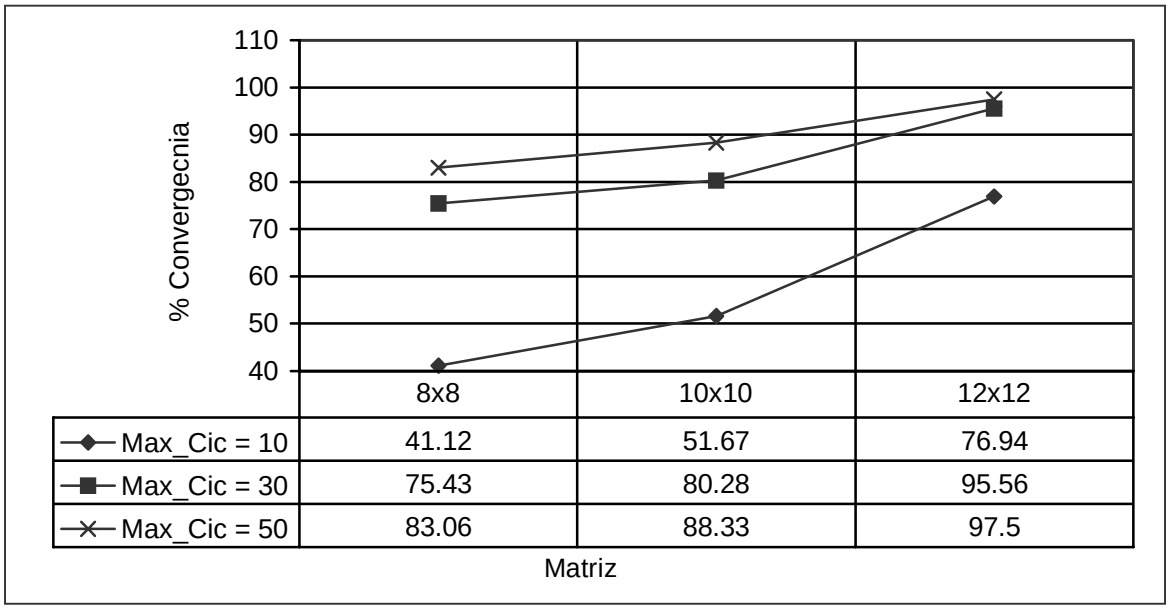
Resultados para el Multiplicador de dos bits.

Los resultados obtenidos del análisis de varianza factorial, de los experimentos con el Multiplicador de 2 bits, fueron muy diferentes a los de los circuitos anteriores, ya que con este circuito, todos los parámetros de manera individual, reportaron tener influencia sobre el porcentaje de convergencia, incluyendo al factor de evaporación, el cual en la mayoría de los circuitos, no reflejaba influencia sobre el porcentaje de convergencia en todos los niveles de Max_Cic. Además, ninguna combinación o interacción de los parámetros, mostró influencia significativa sobre esta variable de respuesta. Todo esto se puede apreciar en la tabla 5.26.

Parámetro/ Niveles	% Convergencia		
Matriz	**Max_Cic=10**	**Max_Cic=30**	**Max_Cic=50**
8x8	41.12	75.43	83.06
10x10	51.67	**80.28**(1654.67)	**88.33**(2757.67)
12x12	**76.94**(1365.89)	**95.56**(4101.33)	**97.50**(6939.78)
Popsize			
10	32.23	70.71	79.72
30	60.83	**86.67**(2130.78)	**93.06**(3627.33)
50	**76.67**(1161.78)	**93.89**(3478.89)	**96.11**(5803.89)
Evap			
0.10	52.78	**86.39**(2086.56)	**94.72**(3530.00)
0.50	63.90	**92.66**(2138.89)	**95.83**(3563.33)
0.75	53.05	72.22	78.33

Tabla 5.26 Medias del porcentaje de convergencia del multiplicador de dos bits, para Matriz, PopSize y Evaporación, de manera indiependiente, con todos los valores de Max_Cic.

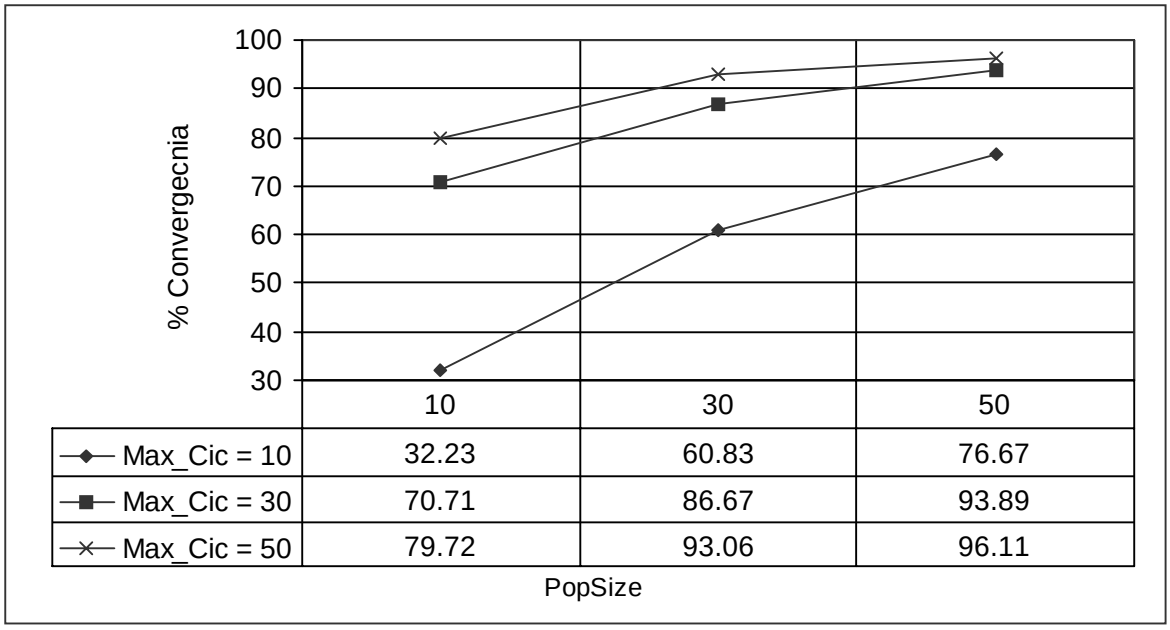
Como se puede apreciar en la tabla 5.26, con este circuito, los datos sugieren que con una matriz de 12x12 y con un mínimo de 30 ciclos o iteraciones, logramos más del 90% en promedio de porcentaje de convergencia. Lo mismo sucede con PopSize = 50.



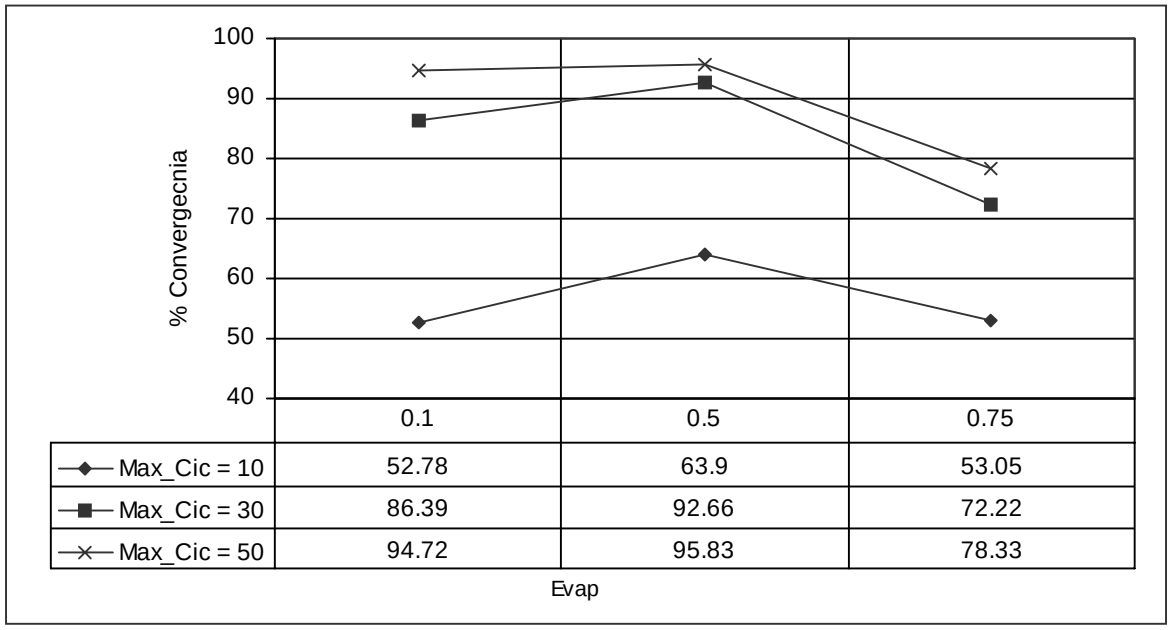
Gráfica 5.21 Medias del porcentaje de convergencia del multiplicador de dos bits, para Matriz de manera independiente, con todos los valores de Max_Cic.

En la gráfica 5.21 podemos observar como en cada nivel de Matriz, al aumentar de 10 a 30 el nivel de Max_Cic, se logra un gran aumento del porcentaje de convergencia, pero cuando se aumenta de 30 a 50 el porcentaje de convergencia no aumenta en la misma proporción. Por otro lado, como ya se mencionó, conforme se aumenta el tamaño de la matriz, el porcentaje de convergencia también aumenta, pero en menor grado que cuando se aumenta el número de ciclos. Además de el costo en cuanto a tiempo se incrementa mucho más que cuando se aumentan los ciclo, como ya se mencionó en el análisis del tiempo. Algo que también, se observa es que si quisiéramos aumentar el porcentaje de convergencia, es conveniente primero aumentar el número de ciclos y después la Matriz.

En la gráfica 5.22, podemos observar el efecto del tamaño de población en cada nivel de Max_Cic. Dicha gráfica sugiere que es más conveniente aumentar el número de ciclos que el tamaño de la población, para elevar el porcentaje de convergencia, aunque las diferencias entre Max_Cic 30 y 50, para PopSize 30 y 50, no son significativas. Esa es la razón por la que en la tabla 5.22, aparecen en negritas.



Gráfica 5.22 Medias del porcentaje de convergencia del multiplicador de dos bits, para PopSize de manera independiente, con todos los valores de Max_Cic.



Gráfica 5.23 Medias del porcentaje de convergencia del multiplicador de dos bits, para Evap de manera independiente, con todos los valores de Max_Cic.

En la gráfica 5.23 se muestra el efecto de la evaporación. Como se pueda apreciar, cuando los rastros de feromona se evaporan mucho, es decir en el caso de 0.75 ó 75%, el porcentaje de convergencia disminuye, con respecto a los resultados con Evap = 0.10 y Evap = 0.50. También es claro que los mejores resultados se logran con Evap = 0.50.

Con lo anterior, podríamos concluir que para el Multiplicador de 2 bits, los valores mínimos con los que se logran buenos resultados son una Matriz de 12x12, un PopSize de 30, con Evap 0.50 y un Max_Cic mayor o igual que 30.

Resultados para el circuito de Katz de 3 salidas.

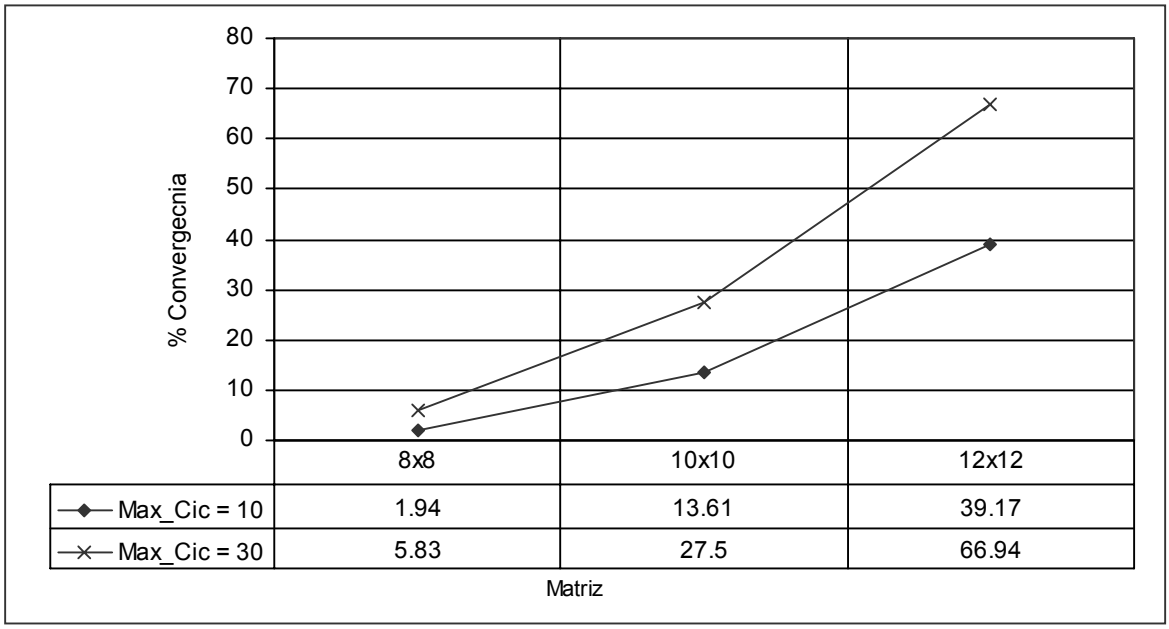
Con el circuito de Katz de tres salidas, el análisis de varianza arrojó los siguientes resultados. Los datos de los experimentos indican que en los niveles 10 y 30 de Max_Cic, la *evaporación* nuevamente no aparece como factor influyente en cuanto al porcentaje de convergencia, además de que Matriz y PopSize si muestran tener influencia, pero por separado y no en sus interacciones (tabla 5.27). Sin embargo, en el nivel 50 de Max_Cic la interacción de Matriz-PopSize y Matriz-Evap, son las que indican tener influencia en el porcentaje de convergencia (tabla 5.28).

Parámetro/ Niveles	**% Convergencia**	
Matriz	**Max_Cic=10**	**Max_Cic=30**
8x8	1.94	5.83
10x10	13.61	27.50
12x12	**39.17(1029.44)**	**66.94(3140.78)**
Popsize		
10	8.06	19.17
30	15.00	**27.22**
50	**31.67(874.00)**	**53.89(2681.11)**

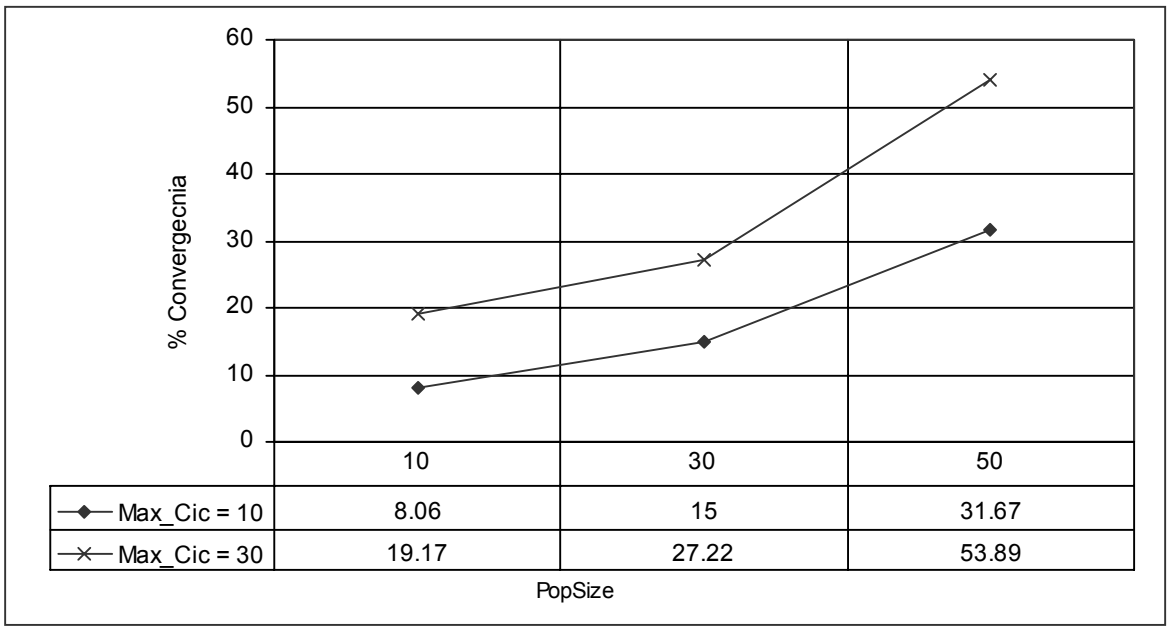
Tabla 5.27 Medias del porcentaje de convergencia del circuito de Katz de 3 salidas, para Matriz y PopSize, de manera independiente, con los niveles 10 y 30 para Max_Cic.

En la gráfica 5.24, podemos observar que a diferencia del circuito anterior, se logra mayor porcentaje de convergencia, al aumentar Matriz que Max_Cic.

La gráfica 5.25, sugiere que para este circuito, es más conveniente aumentar el número de ciclos, que el tamaño de la población.



Gráfica 5.24 Medias del porcentaje de convergencia del Katz de tres salidas, para Matriz de manera independiente, para Max_Cic 10 y 30.

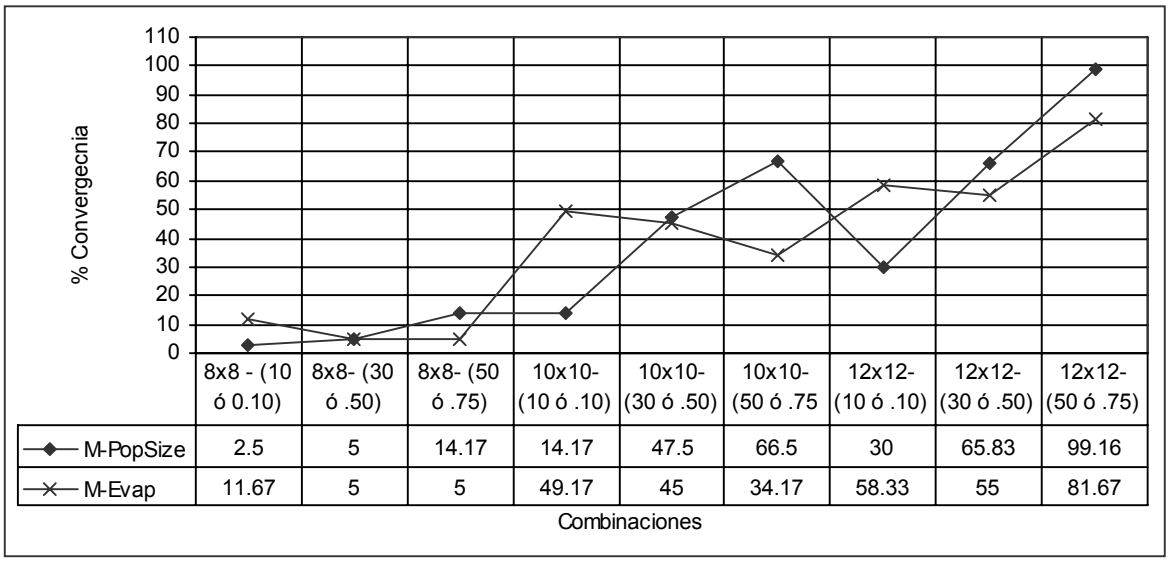


Gráfica 5.25 Medias del porcentaje de convergencia del Katz de tres salidas, para *tamaño de población* de manera independiente, para Max_Cic 10 y 30.

Parámetros		Convergencia	Parámetros		Convergencia
Matriz	**Popsize**	**Max_Cic 50**	**Matriz**	**Evap**	**Max_Cic 50**
8x8	**10**	2.50	**8x8**	**0.10**	11.67
8x8	**30**	5.00	**8x8**	**0.50**	5.00
8x8	**50**	14.17	**8x8**	**0.75**	5.00
10x10	**10**	14.17	**10x10**	**0.10**	**49.17(2039.67)**
10x10	**30**	47.50	**10x10**	**0.50**	**45.00(2039.67)**
10x10	**50**	**66.67(3421.67)**	**10x10**	**0.75**	34.17
12x12	**10**	30.00	**12x12**	**0.10**	**58.33(5105.67)**
12x12	**30**	**65.83(5130.33)**	**12x12**	**0.50**	**55.00(5152.67)**
12x12	**50**	**99.16(8572.00)**	**12x12**	**0.75**	**81.67(5165.00)**

Tabla 5.28 Medias del porcentaje de convergencia del circuito de Katz de 3 salidas, en las combinaciones de Matriz-PopSize y Matriz-Evap, con Max_Cic=50.

Como se puede apreciar en la tabla 5.28, para este circuito en Max_Cic, las interacciones o combinaciones de Matriz-PopSize y Matriz-Evap, tienen influencia significativa sobre el porcentaje de convergencia.



Gráfica 5.26 Medias del porcentaje de convergencia del Katz de tres salidas, para las cobinaciones de Matriz-PopSize y Matriz-Evap, para Max_Cic 50.

En la gráfica 5.26, podemos observar el comportamiento tanto de Matriz-PopSize como el de Matriz-Evap. Este último se comporta un poco diferentes que en los demás circuitos, ya que en los niveles 8x8 y 10x10 de Matriz, al aumentar el nivel de evaporación, la convergencia disminuye, aunque la diferencia no es significativa entre 0.10 y 0.50. Sólo en el último nivel de la gráfica (12x12, 0.75), tenemos que esta combinación aumenta con respecto a la combinación anterior(12x12, 0.50), aunque la diferencia, según el análisis, no es significativa.

Con respecto a este circuito, podemos decir, que fue más complicado para nuestra aplicación, desde el punto de vista del porcentaje de convergencia.

Creemos que para que este circuito logre llegar a porcentaje de convergencia del 100%, en promedio, es necesaria una matriz de mayor dimensión que 12x12. El mejor promedio de convergencia obtenido, fue mediante la combinación de Matriz 12x12, Max_Cic 50 y PopSize 50. En el caso de la *evaporación* los mejores resultados fueron encontrados con la combinación Matriz 12x12 y Evap 0.75, aunque posiblemente, eso sea mas atribuible al tamaño de la Matriz que a PopSize. Además, de acuerdo al análisis, no es hay diferencia significativa con los resultados obtenidos con los demás valores de Evap.

5.1 Conclusiones y comentarios

Las conclusiones de este análisis, en cuanto al *tiempo* están muy claras.

El *factor de evaporación*, NO influye sobre el tiempo necesario para terminar una corrida, tal y como lo esperábamos de acuerdo a la hipótesis 8 de la tabla 5.1.

El incremento del *tamaño de la población* incrementa el tiempo, casi en la misma proporción que lo incrementa el aumento del *número de iteraciones*. El incremento en el tiempo por cada uno de ellos, lo esperábamos de acuerdo a las hipótesis 6 y 4 respectivamente.

El tamaño de la matriz, influye sobre el tiempo necesario para terminar una corrida, tal y como lo esperábamos de acuerdo a la hipótesis 2.

Además de lo anterior nos percatamos de que el tiempo aumenta uniformemente de acuerdo al número de hormigas de la población, al igual que aumenta de acuerdo al número de iteraciones y el aumento por cada uno de ellos es similar. Por otro lado, el incremento del tamaño de la matriz de 8x8 a 10x10, provoca un incremento en el tiempo casi igual que el que provocan PopSize y Max_Cic al incrementarse de 10 a 30. Pero cuando se incrementa la Matriz de 10x10 a 12x12, provoca un incremento en el tiempo superior al que provocan el incremento de PopSize o Max_Cic de 30 a 50. Por lo que algunas veces, resulta más caro el incremento del tamaño de la Matriz (en cuanto a tiempo).

Con respecto a la convergencia, las conclusiones son las siguientes:


El tamaño de la matriz, influye sobre el porcentaje de convergencia, tal y como se esperaba de acuerdo a la hipótesis 1 de la tabla 5.1, ya que en todos los circuitos, éste aparecía como un factor significante en el porcentaje de convergencia, ya sea combinada con otro factor o por separado.


El tamaño de la población influye sobre el porcentaje de convergencia. Dentro del análisis, este factor indicaba que el aumento en sus niveles, significaba un aumento en el porcentaje de convergencia. Esto lo esperábamos de acuerdo a la hipótesis 3, presentada al principio de este capítulo.


Como lo esperábamos de acuerdo a la hipótesis 5, el número de ciclos o iteraciones, influye sobre el porcentaje de convergencia, ya que como veíamos en el análisis de cada uno de los circuitos, un incremento en el número de iteraciones, se reflejaba en un aumento en el porcentaje de convergencia.


Aunque no en todos los circuitos, el factor de evaporación, influye sobre el porcentaje de convergencia, como lo planteábamos en la hipótesis 7, ya que en algunos circuitos como el de Sasao y el Multiplicador, de manera independiente y en algunos niveles de Max_Cic, tuvo un impacto significativo sobre la convergencia, mientras que en el circuito de Katz de tres salidas, su impacto o contribución se manifestó mediante su combinación con Matriz. En los dos primeros circuitos los mejores resultados fueron obtenidos con 0.50, seguidos con los de 0.10, que de hecho no tenían un diferencia significativa. En el caso del último circuito, la combinación Matriz-Evap, obtenía mejores resultados con Evap = 0.10, seguidos por los 0.50 y finalmente 0.75. Sin embargo en la última combinación (Matriz=12x12, Evap=0.75), se logró el mayor porcentaje de convergencia, pero según el análisis, sin diferencia significativa entre los resultados de los demás valores de Evap y su combinación con Matriz.

El *tamaño de la matriz* es algo que juega un papel muy importante, tanto en el porcentaje de convergencia como en tiempo. De hecho su incremento, en la mayoría de los casos provoca un mayor aumento del porcentaje de convergencia que el de los demás factores, pero como ya se mencionó arriba, algunas veces el costo en cuanto a tiempo es mayor. Sin embargo, creemos que si no se tiene el tamaño de la Matriz adecuado para el circuito a diseñar, se tendría que aumentar seriamente el tamaño de los demás parámetros, principalmente PopSize y Max_Cic, para poder obtener buenos porcentajes de convergencia.

La *evaporación*, en demasía, provoca de manera general una baja en la convergencia. Además de lo que apreciamos en el análisis empírico presentado antes del estadístico, pareciera ser que el valor 0.50 de Evap, fomenta la aparición de soluciones de mayor

calidad que con los demás valores, aunque esto no lo podemos asegurar, ya que como mencionamos antes, fue un análisis empírico y no fue motivo de estudio en el análisis estadístico, ya que para eso hubiésemos necesitado una cantidad de experimentos mayor a la que pudimos realizar.

Con lo anterior, podemos decir que la estrategia para buscar los parámetros adecuados para la solución de un circuito sería la siguiente:

Utilizar inicialmente un tamaño de matriz con el cual creamos pueda funcionar nuestro circuito de acuerdo a sus características (entradas, salidas), tomando como referencia los utilizados en este experimento. Con circuitos con características similares a los de aquí presentados, recomendaríamos una Matriz de 10x10.

Al igual, que con Matriz, elegir un tamaño de población adecuado. Para circuitos como los aquí presentados podríamos sugerir 30 individuos.

Para Max_Cic, lo mismo que el tamaño de la población.

Para Evap, de acuerdo a lo observado, el valor que recomendaríamos sería utilizar 0.50, ya que en la mayoría de los casos fue con la que obtuvimos mejores resultados.

Con los valores anteriores, hacer algunos experimentos y si no logramos resultados satisfactorios, incrementar el número de iteraciones, para darles más tiempo a las hormigas de construir más rastros.

Si con lo anterior no se lograra resultados como los deseados, incrementar el tamaño de la población (40 o 50 por ejemplo).

Los dos puntos anteriores los podemos repetir en el mismo orden, si no se logran los resultados que deseamos. Pero una vez repetidos, lo que conviene más es incrementar el tamaño de la matriz, y regresar a los valores iniciales.

Repetir los últimos tres puntos, con los cuál exploraríamos, todas las combinaciones, comenzando con las que requieren menos tiempo. De tal forma que cuando logremos obtener los resultados deseados, podremos asegurar que fue con el menor tiempo de experimentación.

Resumen

En este capítulo se mostró el impacto que tiene cada uno de los parámetros del AS en el desempeño del algoritmo tanto en tiempo como en el porcentaje de convergencia. Para ello se diseño un modelo factorial de experimentación, en el cual se asignaron tres valores diferentes a cada parámetro y se hicieron 40 corridas para cada combinación de estos, utilizando cinco circuitos diferentes.

El análisis de los datos obtenidos de los experimentos, mostró que tanto el tamaño de la población como el número de iteraciones, tiene un impacto parecido tanto en la convergencia como en el tiempo, además mostró que tamaño de la matriz tiene un fuerte impacto tanto en el porcentaje de convergencia como en el tiempo, mientras que el factor de evaporación en cuanto al tiempo no mostró tener ninguna influencia, mientras que con respecto a la convergencia mostró tener cierta influencia, aunque no tan marcada como la de los demás parámetros. Por otro lado, empíricamente nos percatamos que la evaporación con 0.5 fue la que mejores resultados obtenía en cuanto a la calidad de las soluciones, cuando los demás parámetros contribuían obtener soluciones válidas.

Referencias Bibliográficas

[1] Montgomery, Douglas C.. **Diseño y Análisis de Experimentos**. *Grupo Editorial Iberoamericano.* Traducido por Jaime Delgado Saldivar. México D.F.1991.

[2] **Tabla de números aleatorios**, http://www.mrs.umn.edu/~sungurea/introstat/public/instruction/ranbox/randomnumbers.html

[3] Stützle Thomas and Dorigo Marco. **ACO Algorithms for the Traveling Salesman Problem**, *Iridia, Université Libre de Bruxelles, Belgium*, 1999.

[4] Infante Gil Said, Zárate de Lara Guillermo P. **Métodos Estadísticos *un enfoque interdisciplinario***. *Editorial Trillas*, 2ª Edición, México, D.F. ,1986.

Conclusiones

El ant system(AS), es una metaheurística basada en el comportamiento de forrajeo de las hormigas. El AS ha sido utilizado para resolver problemas de optimización combinatoria, los cuales pueden ser descritos como problemas cuyo objetivo es encontrar la secuencia óptima de elementos. Por ejemplo: el problema del vendedor viajero, planeación de horarios y ruteo en redes.

En este trabajo, hemos presentado una aplicación del ant system para optimizar circuitos lógicos combinatorios, donde además de crear circuitos válidos (totalmente funcionales), se trata de minimizar el uso de compuertas lógicas en las soluciones. Es importante aclarar que el diseño de circuitos lógicos es un problema con características diferentes a las de los problemas con los que se ha utilizado hasta hora el AS.

Dicha aplicación funciona mediante la combinación de subcircuitos (compuertas o pedazos de circuitos), que sería el equivalente a las ciudades del problema del vendedor viajero. Estas combinaciones son buscadas por las hormigas o agentes, guiadas en gran medida por los rastros que van dejando.

El enfoque propuesto fue descrito a detalle y presentamos varios resultados con diferentes circuitos con los cuales fue probado. Dichos resultados fueron comparados con otros métodos para el diseño de circuitos, donde demostramos que nuestra aplicación obtiene mejores resultados que los logrados con técnicas usadas por diseñadores humanos, tales como los Mapas de Karnaugh y las reglas Booleanas para la simplificación y con respecto a otras técnicas evolutivas como un BGA (Algoritmo Genético con representación Binaria) y el MGA (un algoritmo Genético Multiobjetivo), en todos los casos, los resultados fueron muy cercanos o mejores a los de las otras técnicas evaluadas.

En parte final de esta tesis, propusimos una estrategia para poder encontrar los mejores parámetros necesarios para la aplicación y lograr así, un porcentaje de convergencia más alto, usando un menor tiempo de procesamiento computacional.

Trabajos Futuros

Algunas de las futuras rutas de investigación que estamos interesados en explorar son la paralización del algoritmo para mejorar su desempeño, con lo cual, cada agente podría operar independientemente de los demás, hasta que todos terminen su ruta, para después reunirse y poder actualizar los rastros de feromona.

Por otro lado, estamos interesados en la exploración de otras alternativas más "poderosas" para la representación de una expresión Booleana, intentando así superar las inherentes limitaciones de la representación matricial actual. Esto nos permitiría resolver circuitos del mundo real en una cantidad de tiempo razonable y sin la necesidad excesiva de poder computacional. La alternativa que estamos considerando explorar en primer lugar es el uso de una representación mediante árboles, tal y como lo hace la programación genética.

Estamos también interesados en implementar el uso de subpoblaciones que se especialicen en optimizar una salida del circuito exclusivamente, dejando sus respectivos rastros, para que otra subpoblación encargada de optimizar el circuito total utilice ocupe los rastros de todas las poblaciones, con lo que se esperaría que los circuitos se resuelvan más rápido y con mayor calidad.

Finalmente, tenemos gran interés también en la hibridización de nuestro algoritmo con alguna otra técnica (un algoritmo genético por ejemplo). La idea es tener otro algoritmo como un sistema de búsqueda local, para mejorar las soluciones logradas por el AS, en cada iteración.

Apéndice A

Cuadros de Análisis de Varianza para cada circuito por variables dependientes, Max_Cic y par de variables independientes.

Circuito de Sasao

Max_Cic=10

Variable dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	400123.630	2	200061.815	697079.490	0.000000
Popsize	240587.185	2	120293.593	419141.440	0.000000
Evaporización	0.963	2	0.481	1.670	0.247690
Matriz*Popsize	119307.259	4	29826.815	103926.190	0.000000
Matriz*Evapor	2.148	4	0.537	1.870	0.209200
Popsize*Evapor	1.259	4	0.315	1.100	0.419153
Error	2.297	8	0.287		

Max_cic=30

Variable dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	3610958.000	2	1805479.000	2954957.450	0.000000
Popsize	2165280.667	2	1082640.333	1771915.440	0.000000
Evaporización	1.556	2	0.778	1.270	0.331893
Matriz*Popsize	1070401.333	4	267600.333	437971.090	0.000000
Matriz*Evapor	3.778	4	0.944	1.540	0.279174
Popsize*Evapor	0.444	4	0.111	0.180	0.942401
Error	4.889	8	0.611		

Max_Cic=50

Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	10,032,100.00	2	5016040.037	14666783.730	0.000000
Popsize	6014668.074	2	3007334.037	8793374.380	0.000000
Evaporización	0.963	2	0.481	1.410	0.298848
Matriz*Popsize	2974387.037	4	743596.759	2174259.530	0.000000
Matriz*Evapor	2.815	4	0.704	2.060	0.178400
Popsize*Evapor	2.148	4	0.537	1.570	0.271779
Error	2.740	8	0.342		

Circuito de Katz de una salida

Max_Cic=10

Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	645730.074	2	322865.037	69463.218	0.000000
Popsize	383986.741	2	191993.37	41306.663	0.000000
Evaporización	22.296	2	11.148	2.400	0.152588
Matriz*Popsize	190126.148	4	47531.537	10226.23	0.000000
Matriz*Evapor	43.259	4	10.815	2.330	0.143487
Popsize*Evapor	32.593	4	8.148	1.750	0.231938
Error	37.185	8	4.648		

Max_Cic=30

Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	5676730.296	2	2838365.148	4643.690	0.000000
Popsize	3528298.296	2	1764149.148	2886.230	0.000000
Evaporización	540.519	2	270.259	0.442	0.657545
Matriz*Popsize	1611319.030	4	402829.759	659.048	0.000000
Matriz*Evapor	538.148	4	134.537	0.220	0.919850
Popsize*Evapor	3049.481	4	762.370	1.247	0.365067
Error	4889.86	8	611.23		

Max_Cic=50

Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	16,036,900.00	2	8018458.815	1755.100	0.000000
Popsize	9640242.074	2	4820121.037	1055.040	0.000000
Evaporización	12243.852	2	6121.926	1.340	0.314829
Matriz*Popsize	4764756.370	4	1191189.093	260.730	0.000000
Matriz*Evapor	18025.926	4	4506.481	0.990	0.465317
Popsize*Evapor	22952.815	4	5738.204	1.260	0.360670
Error	36,549.40	8	4568.675		

Circuito Sumador de dos bits

Max_Cic=10
Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	3655733.407	2	1827866.704	6173.870	0.000000
Popsize	2287849.407	2	1143924.704	3863.762	0.000000
Evaporización	492.074	2	246.037	0.831	0.469993
Matriz*Popsize	1059428.370	4	264857.093	984.591	0.000000
Matriz*Evapor	966.370	4	241.593	0.816	0.549364
Popsize*Evapor	1116.370	4	279.093	0.943	0.486648
Error	2368.519	8	296.06		

Max_Cic=30
Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	33,419,000.00	2	16,709,500.00	3431108.830	0.000000
Popsize	19,990,200.00	2	9995083.370	2052378.515	0.000000
Evaporización	2.30	2	1.148	0.236	0.795090
Matriz*Popsize	9,907,437.48	4	2476859.370	508595.353	0.000000
Matriz*Evapor	28.59	4	7.148	1.468	0.297901
Popsize*Evapor	28.82	4	7.204	1.480	0.294681
Error	38.96	8	4.87		

Max_Cic=50
Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	93,401,300.00	2	46,700,700.00	3649.940	0.000000
Popsize	55,350,200.00	2	27,675,100.00	2162.970	0.000000
Evaporización	13,916.74	2	6958.370	0.544	0.600464
Matriz*Popsize	27,311,500.00	4	6827869.870	533.640	0.000000
Matriz*Evapor	56,402.59	4	14100.648	1.100	0.419153
Popsize*Evapor	18,845.93	4	4711.481	0.370	0.823833
Error	102,359,.41	8	12794.93		

Circuito Multiplicador de dos bits

Max_Cic=10

Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	6503164.519	2	3251582.259	18667.380	0.000000
Popsize	3867610.963	2	1933805.481	11107.950	0.000000
Evaporización	2473.852	2	1236.926	7.100	0.016864
Matriz*Popsize	1941332.148	4	485333.037	2787.800	0.000000
Matriz*Evapor	1332.593	4	333.148	1.910	0.202217
Popsize*Evapor	3312.815	4	828.204	4.760	0.029242
Error	1392.74	8	174.092		

Max_Cic=30

Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	59,182,700.00	2	29,591,400.00	10626.300	0.000000
Popsize	34,606,100.00	2	17,303,100.00	6213.560	0.000000
Evaporización	15,332.07	2	7666.037	2.750	0.123318
Matriz*Popsize	17,464,600.00	4	4366155.926	1567.890	0.000000
Matriz*Evapor	8,982.15	4	2245.537	0.810	0.552515
Popsize*Evapor	15,791.93	4	3947.981	1.420	0.311194
Error	22,277.85	8	2784.731		

Max_Cic=50

Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	170,959,000.00	2	85,479,300.00	94289.730	0.000000
Popsize	95,738,100.00	2	47,869,100.00	52803.010	0.000000
Evaporización	6,350.00	2	3,175.00	3.500	0.080909
Matriz*Popsize	48,111,200.00	4	12,027,800.00	13267.520	0.000000
Matriz*Evapor	2,779.78	4	694.94	0.770	0.573954
Popsize*Evapor	12,286.44	4	3,071.61	30.390	0.066648
Error	7252.45	8	906.56		

Circuito de Katz de tres salidas

Max_Cic=10
Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	3773656.889	2	1886828.444	60378.510	0.000000
Popsize	2198016.222	2	1099008.111	35168.260	0.000000
Evaporización	330.889	2	165.444	5.290	0.034370
Matriz*Popsize	1106188.889	4	276547.222	8849.510	0.000000
Matriz*Evapor	334.222	4	83.556	2.670	0.110572
Popsize*Evapor	304.889	4	76.222	2.440	0.131672
Error	250.000	8	31.25		

Max_Cic=30
Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	35,053,900.00	2	17,526,900.00	985.370	0.000000
Popsize	20,901,900.00	2	10,450,900.00	587.560	0.000000
Evaporización	32,366.52	2	16,183.26	0.909	0.440827
Matriz*Popsize	11,112,000.00	4	2,777,998.20	156.180	0.000000
Matriz*Evapor	89,148.82	4	22,287.20	1.250	0.364047
Popsize*Evapor	100,981.93	4	25,245.48	1.420	0.311194
Error	142,296.74	8	17787.09		

Max_Cic=50
Variable Dependiente: *T i e m p o*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	94,066,500.00	2	47,033,200.00	27899.140	0.000000
Popsize	55,058,600.00	2	27,529,300.00	16329.820	0.000000
Evaporización	4,873.56	2	2436.778	1.440	0.292310
Matriz*Popsize	27,824,300.00	4	6956074.222	4126.200	0.000000
Matriz*Evapor	4,208.89	4	1052.222	0.620	0.660973
Popsize*Evapor	6,272.89	4	1568.222	0.930	0.492724
Error	13,486.67	8	1685.83		

Apéndice B

Cuadros de Análisis de Varianza para cada circuito por variables dependientes, Max_Cic y par de variables independientes.

Circuito de Sasao

Max_Cic=10

Variable dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	20352.889	2	10176.444	256.547	0.000000
Popsize	4976.222	2	2488.111	62.725	0.000000
Evaporización	76.222	2	38.111	0.961	0.422633
Matriz*Popsize	1756.889	4	439.222	11.073	0.002404
Matriz*Evapor	42.889	4	10.722	0.270	0.889274
Popsize*Evapor	16.222	4	4.056	0.102	0.978656
Error	317.334	8	39.667		

Max_cic=30

Variable dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	23980.963	2	11990.482	1242.792	0.000000
Popsize	5921.185	2	2960.593	306.861	0.000000
Evaporización	229.407	2	114.704	11.889	0.004017
Matriz*Popsize	1377.037	4	344.259	35.682	0.000038
Matriz*Evapor	103.481	4	25.870	2.681	0.109670
Popsize*Evapor	41.926	4	10.482	1.086	0.424746
Error	77.186	8	9.648		

Max_Cic=50

Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	19440.296	2	9720.148	294.220	0.000000
Popsize	7056.519	2	3528.260	106.797	0.000002
Evaporización	102.741	2	51.371	1.555	0.268846
Matriz*Popsize	2153.481	4	538.370	16.296	0.000652
Matriz*Evapor	835.259	4	208.815	6.321	0.013479
Popsize*Evapor	177.037	4	44.259	1.340	0.334894
Error	264.296	8	33.037		

Circuito de Katz de una salida

Max_Cic=10

Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	18061.574	2	9030.787	114.744	0.000001
Popsize	7503.241	2	3751.620	47.667	0.000036
Evaporización	0.463	2	0.231	0.003	0.997006
Matriz*Popsize	975.926	4	243.981	3.100	0.081153
Matriz*Evapor	45.370	4	11.343	0.144	0.960650
Popsize*Evapor	103.704	4	25.926	0.329	0.851079
Error	629.629	8	78.704		

Max_Cic=30

Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	12031.019	2	6015.509	238.417	0.000000
Popsize	6464.352	2	3232.176	128.103	0.000001
Evaporización	6.019	2	3.009	0.119	0.889350
Matriz*Popsize	1093.981	4	273.495	10.840	0.002577
Matriz*Evapor	64.815	4	16.204	0.642	0.647580
Popsize*Evapor	73.148	4	18.287	0.725	0.598977
Error	201.851	8	25.231		

Max_Cic=50

Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	9235.185	2	4617.593	288.059	0.000000
Popsize	4000.463	2	2000.231	124.780	0.000001
Evaporización	14.352	2	7.176	0.448	0.654005
Matriz*Popsize	2148.148	4	537.037	33.502	0.000048
Matriz*Evapor	55.093	4	13.773	0.859	0.527284
Popsize*Evapor	18.981	4	4.745	0.296	0.872649
Error	128.241	8	16.030		

Circuito Sumador de dos bits

Max_Cic=10

Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	16122.68	2	8061.343	253.270	0.000000
Popsize	5972.68	2	2986.34	93.824	0.000003
Evaporización	56.019	2	28.009	0.880	0.451399
Matriz*Popsize	1713.426	4	428.356	13.458	0.001256
Matriz*Evapor	30.093	4	7.523	0.236	0.910307
Popsize*Evapor	338.426	4	84.606	2.658	0.111566
Error	254.629	8	31.829		

Max_Cic=30

Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	12756.712	2	6378.356	87.738	0.000004
Popsize	5582.389	2	2791.194	38.394	0.000079
Evaporización	488.834	2	244.417	3.362	0.087148
Matriz*Popsize	1106.693	4	276.673	3.806	0.051000
Matriz*Evapor	107.939	4	26.985	0.371	0.823164
Popsize*Evapor	89.635	4	22.409	0.308	0.864857
Error	581.585	8	72.698		

Max_Cic=50
Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	8711.574	2	4355.787	59.831	0.000015
Popsize	3562.963	2	1781.481	24.471	0.000390
Evaporización	1135.185	2	567.593	7.796	0.013222
Matriz*Popsize	1062.037	4	265.509	3.647	0.056381
Matriz*Evapor	368.981	4	92.245	1.267	0.358327
Popsize*Evapor	25.926	4	6.481	0.089	0.983347
Error	582.408	8	72.801		

Circuito Multiplicador de dos bits

Max_Cic=10
Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	6100.149	2	3050.074	79.788	0.000005
Popsize	9128.927	2	4564.463	119.404	0.000001
Evaporización	724.149	2	362.074	9.472	0.007772
Matriz*Popsize	139.631	4	34.908	0.913	0.500786
Matriz*Evapor	192.076	4	48.019	1.256	0.362016
Popsize*Evapor	353.298	4	88.324	2.311	0.145656
Error	305.817	8	38.227		

Max_Cic=30
Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	1985.349	2	992.674	13.627	0.002652
Popsize	2531.849	2	1265.924	17.378	0.001226
Evaporización	1972.460	2	986.230	13.538	0.002706
Matriz*Popsize	267.476	4	66.869	0.918	0.498401
Matriz*Evapor	633.031	4	158.258	2.172	0.162794
Popsize*Evapor	450.698	4	112.674	1.547	0.277428
Error	582.784	8	72.848		

Max_Cic=50

Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	961.574	2	480.787	9.672	0.007327
Popsize	1367.130	2	683.565	13.751	0.002578
Evaporización	1728.241	2	864.120	17.383	0.001225
Matriz*Popsize	334.259	4	83.565	1.681	0.246339
Matriz*Evapor	393.981	4	98.495	1.981	0.190497
Popsize*Evapor	100.926	4	25.231	0.508	0.732046
Error	397.685	8	49.711		

Circuito de Katz de tres salidas

Max_Cic=10

Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	6524.074	2	3262.037	38.555	0.000078
Popsize	2650.463	2	1325.231	15.663	0.001713
Evaporización	235.185	2	117.593	1.390	0.303309
Matriz*Popsize	1693.981	2	423.495	5.005	0.025636
Matriz*Evapor	317.593	4	79.398	0.938	0.488976
Popsize*Evapor	549.537	4	137.384	1.624	0.259036
Error	676.852	8	84.607		

Max_Cic=30

Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	17279.630	2	8639.815	45.023	0.000044
Popsize	5944.907	2	2972.454	15.490	0.001774
Evaporización	406.019	2	203.009	1.058	0.391133
Matriz*Popsize	1481.481	4	370.370	1.930	0.198831
Matriz*Evapor	420.370	4	105.093	0.548	0.706157
Popsize*Evapor	384.259	4	96.065	0.501	0.736625
Error	1535.185	8	191.898		

Max_Cic=50

Variable Dependiente: *Porcentaje de Convergencia*

Tabla I.1					
Fuente	**S.M.**	**G.L.**	**C.M**	**F**	**p**
Matriz	15288.889	2	7644.444	143.895	0.000001
Popsize	8905.556	2	4452.778	83.817	0.000004
Evaporización	151.389	2	75.694	1.425	0.295557
Matriz*Popsize	2734.722	4	683.681	12.869	0.001461
Matriz*Evapor	1563.889	4	390.972	7.359	0.008645
Popsize*Evapor	293.056	4	73.264	1.379	0.323091
Error	424.999	8	53.125		

Apéndice C

Cuadros de Comparaciones de Medias por Circuito de los niveles y/o combinaciones o las variables independientes que resultaron significativas por Max_-cic.

La prueba elegida para el contraste de medias fue la de *Scheffé*, con un nivel de significancia alfa de 0.01.

La hipótesis nula de no diferencia significativa de medias entre dos niveles y/o combinaciones se rechaza si:

$$\left|\overline{Y}_i - \overline{Y}_j\right| \rangle \quad V.S.$$

donde

$$V.S. = \left[\frac{2(k-1)}{n} F_{k-1,CME}\right]^{1/2}$$

n: número de observaciones dentro de cada nivel o combinación.

k: número de niveles o combinaciones.

Valor *F* con *k-1* grados de libertad en el numerador y con los grados de libertad del error para el denominador y con un nivel de significancia de 0.01.

En nuestro estudio estuvimos *exclusivamente* interesados en contrastar el nivel o combinación de niveles que obtuvo el mayor porcentaje de convergencia con el resto y detectar con cuales otros niveles o combinaciones no existe diferencia significativa en las medias.

Los resultados se presentan por Circuito y Max_Cic en las siguientes tablas. El nivel o combinación que obtuvo la media de convergencia más alta se presenta con color y con y con negritas la(s) media(s) que no fueron significativamente diferentes a ésta.

Circuito de Sasao

Max_Cic=10

Tabla II.1 Comparación de las Medias de la combinación de los valores o niveles de las variables Tamaño de la Matriz y Popsize para % de Convergencia, con V.S.=35.18

Parámetros			
Matriz	**Popsize**	**Medias**	$\lvert 92.67 - \overline{X_i} \rvert$
8x8	**10**	0.67	92.00
8x8	**30**	7.00	85.67
8x8	**50**	11.67	81.00
10x10	**10**	9.33	83.34
10x10	**30**	29.33	63.34
10x10	**50**	40.00	52.67
12x12	**10**	38.67	54.00
12x12	**30**	**84.67**	8.00
12x12	**50**	**92.67**	**-------**

Max_Cic=30

Tabla II.2 Comparación de las Medias de los niveles de la variable Evaporación para % de Convergencia, con V.S.=6.09

Evaporación	**Medias**	$\lvert 74.44 - \overline{X_i} \rvert$
0.10	**70.81**	3.64
0.50	**74.44**	------
0.75	64.17	10.28

Tabla II.3 Comparación de las Medias de la combinación de los valores o niveles de las variables Tamaño de la Matriz y Popsize para % de Convergencia, con V.S.=17.615

Parámetros			
Matriz	**Popsize**	**Medias**	$\lvert 100.00 - \overline{X_i} \rvert$
8x8	**10**	6.00	94.00
8x8	**30**	21.67	78.33
8x8	**50**	33.33	66.67
10x10	**10**	24.67	75.33
10x10	**30**	61.00	39.00
10x10	**50**	**84.00**	16.00
12x12	**10**	80.00	20.00
12x12	**30**	**100.00**	------
12x12	**50**	**100.00**	**------**

Max_Cic=50

Tabla II.4 Comparación de las Medias de la combinación de los valores o niveles de las variables Tamaño de la Matriz y Popsize para % de Convergencia, con V.S.=32.596

Parámetros			
Matriz	**Popsize**	**Medias**	$\lvert 100.00 - \overline{X_i} \rvert$
8x8	**10**	12.00	88.00
8x8	**30**	30.00	70.00
8x8	**50**	46.67	53.33
10x10	**10**	30.33	69.67
10x10	**30**	**79.33**	20.67
10x10	**50**	**95.67**	4.33
12x12	**10**	**84.67**	15.33
12x12	**30**	**100.00**	------
12x12	**50**	**100.00**	------

Circuito Katz de una salida

Max_Cic=10

Tabla II.5 Comparación de las Medias de los niveles de la variable tamaño de la Matriz y Popsize para % de Convergencia, con V.S.=17.40

Matriz	**Medias**	$\lvert 80.28 - \overline{X_i} \rvert$	***Popsize***	**Medias**	$\lvert 65.28 - \overline{X_i} \rvert$
8x8	16.94	63.34	**10**	25.56	39.72
10x10	47.22	33.06	**30**	**53.61**	11.67
12x12	**80.28**	------	**50**	**65.28**	------

Max_Cic=30

Tabla II.6 Comparación de las Medias de la combinación de los valores o niveles de las variables Tamaño de la Matriz y Popsize para % de Convergencia, con V.S.=28.49

Parámetros			
Matriz	**Popsize**	**Medias**	$\lvert 100.00 - \overline{X_i} \rvert$
8x8	**10**	18.33	81.67
8x8	**30**	50.83	49.17
8x8	**50**	61.67	38.33
10x10	**10**	47.50	52.50
10x10	**30**	**87.50**	12.50
10x10	**50**	**96.67**	3.33
12x12	**10**	**85.00**	15.00
12x12	**30**	**98.33**	1.67
12x12	**50**	**100.00**	-----

Max_Cic=50

Tabla II.7 Comparación de las Medias de la combinación de los valores o niveles

de las variables Tamaño de la Matriz y Popsize para % de Convergencia, con V.S.=22.71

Parámetros			
Matriz	**Popsize**	**Medias**	$\lvert 100.00 - \overline{X_i} \rvert$
8x8	**10**	25.00	75.00
8x8	**30**	62.50	37.50
8x8	**50**	**78.33**	21.67
10x10	**10**	68.33	31.67
10x10	**30**	**92.50**	7.50
10x10	**50**	**100.00**	-----
12x12	**10**	**98.33**	1.67
12x12	**30**	**99.17**	0.83
12x12	**50**	**100.00**	-----

Circuito Sumador de dos bits

Max_Cic=10

Tabla II.8 Comparación de las Medias de la combinación de los valores o niveles de las variables Tamaño de la Matriz y Popsize para % de Convergencia, con V.S.=31.99

Parámetros			
Matriz	**Popsize**	**Medias**	$\lvert 95.83 - \overline{X_i} \rvert$
8x8	**10**	9.17	86.66
8x8	**30**	10.83	85.00
8x8	**50**	20.00	75.83
10x10	**10**	13.33	82.50
10x10	**30**	47.50	48.33
10x10	**50**	58.33	37.50
12x12	**10**	44.17	51.66
12x12	**30**	**79.17**	16.66
12x12	**50**	**95.83**	-----

Max_Cic=30

Tabla II.9 Comparación de las Medias de los niveles de la variable tamaño de la Matriz y Popsize para % de Convergencia, con V.S.=16.72

Matriz	**Medias**	$\lvert 94.17 - \overline{X}_i \rvert$	***Popsize***	**Medias**	$\lvert 86.09 - \overline{X}_i \rvert$
8x8	41.39	52.78	**10**	51.11	34.98
10x10	73.86	20.31	**30**	**72.22**	13.87
12x12	**94.17**	------	**50**	**86.09**	------

Max_Cic=50

Tabla II.10 Comparación de las Medias de los niveles de la variable tamaño de la Matriz y Popsize para % de Convergencia, con V.S.=16.73

Matriz	**Medias**	$\lvert 97.78 - \overline{X}_i \rvert$	***Popsize***	**Medias**	$\lvert 90.66 - \overline{X}_i \rvert$
8x8	55.28	42.50	**10**	63.89	26.67
10x10	**86.39**	11.39	**30**	**85.00**	5.56
12x12	**97.78**	------	**50**	**90.66**	------

Circuito Multiplicador de dos bits

Max_Cic=10

Tabla II.11 Comparación de las Medias de los niveles de la variable tamaño de la Matriz, Popsize y Evaporación para % de Convergencia, con V.S.=12.12

Matriz	**Medias**	$\lvert 76.94 - \overline{X}_i \rvert$	***Popsize***	**Medias**	$\lvert 76.67 - \overline{X}_i \rvert$	***Evaporación***	**Medias**	$\lvert 63.90 - \overline{X}_i \rvert$
8x8	41.12	35.82	**10**	32.23	44.44	**0.10**	52.78**	11.12
10x10	51.67	25.27	**30**	60.83	15.84	**0.50**	63.90**	------
12x12	**76.94**	------	**50**	**76.67**	------	**0.75**	53.05**	10.85

Max_Cic=30

Tabla II.12 Comparación de las Medias de los niveles de la variable tamaño de la Matriz, Popsize y Evaporación para % de Convergencia, con V.S.=16.73

Matriz	**Medias**	$\lvert 95.56 - \overline{X_i} \rvert$	***Popsize***	**Medias**	$\lvert 93.89 - \overline{X_i} \rvert$	***Evaporación***	**Medias**	$\lvert 92.66 - \overline{X_i} \rvert$
8x8	75.43	20.13	**10**	70.71	22.96	**0.10**	**86.39**	6.27
10x10	**80.28**	15.28	**30**	**86.67**	7.22	**0.50**	**92.66**	------
12x12	**95.56**	------	**50**	**93.89**	------	**0.75**	72.22	20.44

Max_Cic=50

Tabla II.13 Comparación de las Medias de los niveles de la variable tamaño de la Matriz, Popsize y Evaporación para % de Convergencia, con V.S.=13.82

Matriz	**Medias**	$\lvert 97.50 - \overline{X_i} \rvert$	**Popsize**	**Medias**	$\lvert 96.11 - \overline{X_i} \rvert$	***Evaporación***	**Medias**	$\lvert 95.83 - \overline{X_i} \rvert$
8x8	83.06	14.44	**10**	79.72	79.72	**0.10**	**94.72**	1.11
10x10	**88.33**	9.17	**30**	**93.06**	93.06	**0.50**	**95.83**	------
12x12	**97.50**	------	**50**	**96.11**	96.11	**0.75**	78.33	17.50

Circuito Katz de tres salidas

Max_Cic=10

Tabla II.14 Comparación de las Medias de los niveles de la variable tamaño de la Matriz y Popsize para % de Convergencia, con V.S.=18.04

Matriz	**Medias**	$\lvert 39.17 - \overline{X_i} \rvert$	***Popsize***	**Medias**	$\lvert 31.67 - \overline{X_i} \rvert$
8x8	1.94	37.23	**10**	8.06	23.61
10x10	13.61	25.56	**30**	**15.00**	16.67
12x12	**39.17**	------	**50**	**31.67**	------

Max_Cic=30

Tabla II.15 Comparación de las Medias de los niveles de la variable tamaño de la Matriz y Popsize para % de Convergencia, con V.S.=27.16

Matriz	**Medias**	$\lvert 66.94 - \overline{X_i} \rvert$	***Popsize***	**Medias**	$\lvert 53.89 - \overline{X_i} \rvert$
8x8	5.83	61.11	**10**	19.17	34.72
10x10	27.50	39.44	**30**	**27.22**	26.67
12x12	**66.94**	------	**50**	**53.89**	

Max_Cic=50

Tabla II.16 Comparación de las Medias de la combinación de los valores o niveles de las variables Tamaño de la Matriz-Popsize y Tamaño de la Matriz-Evaporación para % de Convergencia, con V.S.=41.33

Matriz	**Popsize**	**Medias**	$\lvert 99.16 - \overline{X_i} \rvert$	**Matriz**	**Evaporación**	**Medias**	$\lvert 81.67 - \overline{X_i} \rvert$
8x8	**10**	2.50	96.66	**8x8**	**10**	11.67	70.00
8x8	**30**	5.00	94.16	**8x8**	**30**	5.00	76.67
8x8	**50**	14.17	84.99	**8x8**	**50**	5.00	76.67
10x10	**10**	14.17	84.99	**10x10**	**10**	**49.17**	32.50
10x10	**30**	47.50	51.66	**10x10**	**30**	**45.00**	36.67
10x10	**50**	**66.67**	32.49	**10x10**	**50**	34.17	47.50
12x12	**10**	30.00	69.16	**12x12**	**10**	**58.33**	23.34
12x12	**30**	**65.83**	33.33	**12x12**	**30**	**55.00**	26.67
12x12	**50**	**99.16**	------	**12x12**	**50**	**81.67**	------