Uso del Sistema de la Colonia de Hormigas para el Diseño de Circuitos Lógicos Combinatorios

Benito Mendoza García * y Carlos A. Coello Coello **

***Resumen*—Este artículo presenta una propuesta sobre el uso de la heurística denominada "colonia de hormigas" para el diseño de circuitos lógicos combinatorios al nivel de compuertas. Además de describir los detalles principales de nuestra implementación, ésta es validada a través de varios circuitos de prueba, mostrando un buen desempeño con respecto a otras heurísticas similares. El artículo concluye con los resultados del análisis de sensibilidad efectuado a la técnica para evaluar el efecto que sus parámetros tienen en los resultados.**

I. 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 entre estas técnicas, quizás la más conocida sea el Algoritmo Genético (AG) [1]. 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 [2]. El primer sistema desarrollado en el área ACO es el algoritmo llamado *Ant System* (AS) [3].

Este trabajo se deriva de la investigación previa realizada por Coello [5], en la cual 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 a optimizarlos de acuerdo a una cierta métrica (el número de compuertas). El AS ha sido utilizado con gran éxito en problemas de optimización combinatoria como el problema del viajero (TSP), planeación de horarios y ruteo en redes. Sin embargo, su aplicación en el diseño de circuitos combinatorios requiere un replanteamiento cuidadoso del problema, como veremos en este artículo.

II. EL DISEÑO DE CIRCUITOS LÓGICOS

El diseño de circuitos es el proceso de derivar a partir de una función booleana (normalmente proporcionada a través de una tabla de verdad), una combinación de compuertas lógicas que produzca todas las salidas deseadas para todas las entradas dadas, limitando las compuertas lógicas disponibles a un conjunto predefinido [5].

Pero además de buscar una estructura funcional, en el diseño de circuitos a menudo se busca optimizar algún aspecto de su diseño. Existen varios criterios para definir costes 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.). De acuerdo a lo anterior, 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 dos entradas en vez de compuertas de tres entradas).

III. LA COLONIA DE HORMIGAS

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, los cuadros 3, 4 y 5 de la figura 1 ilustran la manera en que un grupo de hormigas cambia su camino al toparse con un obstáculo en su ruta original.

Todas las hormigas depositan una cierta cantidad de una sustancia llamada feromona mientras caminan. A su vez, cada hormiga prefiere caminar en una dirección rica en feromona. Esta simple conducta explica porqué son capaces de ajustarse a cambios en el ambiente. Cuando un obstáculo inesperado es colocado en el camino que las hormigas están siguiendo (cuadro 3 de la figura 1), las hormigas que están justo enfrente del obstáculo no pueden continuar siguiendo el rastro de feromona y por lo tanto, deben elegir entre avanzar hacia la izquierda o hacia la derecha. La elección es aleatoria. Es decir, cada hormiga decide al azar hacia donde avanzar, pero se espera 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 1). De esta forma, las hormigas que eligieron (aleatoriamente) el camino corto, crearán en un cierto tiempo un depósito de feromona más intenso que el de las hormigas que eligieron el camino largo. Esto se debe a que llegan al otro lado más rápido que las otras, quedando depositada,

* Maestría en Inteligencia Artificial Universidad Veracruzana, bmendoza@mia.uv.mx.
** Departamento. de Ingeniería Eléctrica, Sección de Computación CINVESTAV-IPN, ccoello@cs.cinvestav.mx.

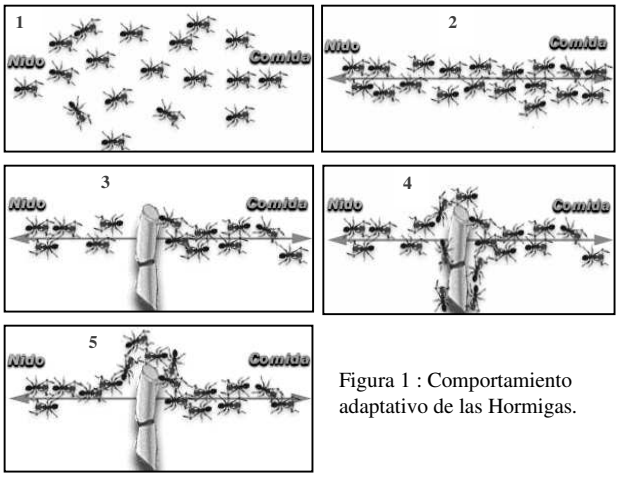


Figura 1 : Comportamiento adaptativo de las Hormigas.

por lo tanto, más feromona en esa ruta, lo que origina que las hormigas que vienen atrás, prefieran caminar por esta ruta, reestableciéndose así, el camino más corto (cuadro 5 de la figura 1) [7].

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.

A. *El algoritmo Ant System*

El Ant System es un algoritmo de búsqueda cooperativo en el que se imita la conducta de las colonias de hormigas usando agentes sencillos. Por razones obvias, a estos agentes se les llama hormigas (*ants*), y se comunican entre sí 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:

1. Se utiliza comunicación indirecta a través de la feromona.
2. Las rutas más cortas tienden a tener una razón más alta de crecimiento del valor de la feromona.
3. 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 problemas. Por ejemplo:

1. Cada hormiga es capaz de determinar la distancia a la que se encuentra de un estado determinado.
2. Poseen información acerca de su ambiente y la utilizan al tomar decisiones. Así, su comportamiento no sólo es adaptativo sino también "avaricioso" (greedy[1]).
3. 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).

IV. El Ant System en el diseño de circuitos

Para aplicar este algoritmo al diseño de circuitos lógicos, se tuvo que replantear el diseño de circuitos lógicos como una variante del problema del viajero. Nótese, sin embargo, que el tratamiento del problema no es simple. El principal problema es que no hay forma inmediata de identificar lo que correspondería a un estado (o ciudad) y no resulta obvio cómo medir la distancia entre estados (ciudades).

A. *Representación de un circuito*

Para representar un circuito utilizamos una matriz bidimensional (figura 2). Cada elemento de la matriz es una tercia del tipo [*Entrada1*, *Entrada2*, *Tipo de Compuerta*]. Para nuestros diseños, *Tipo de Compuerta* puede tomar cualquiera de los siguientes valores: AND, OR, NOT, XOR y WIRE, cada uno de los cuales representa un tipo de compuerta lógica cuyas entradas serán los valores de salida producidos por los elementos situados en las filas *Entrada1* y *Entrada2* de la columna anterior (para el caso del NOT y del WIRE sólo se toma en cuenta el valor de *Entrada1*). Los elementos de la primera columna reciben sus entradas directamente de las columnas de entrada de la tabla de verdad del circuito dado. Los elementos situados en las primeras *N* filas de la última columna, proporcionan los valores de las *N* salidas del circuito (cada uno de dichos elementos deberá producir los valores de una columna de salida de la tabla de verdad del circuito). Esta forma de representar un circuito ha sido utilizada antes con éxito con los AG's [5].

En este artículo extendemos trabajo en el cual se resolvieron únicamente circuitos de una sola salida [9]. Es decir, que nuestro sistema es ahora capaz de resolver circuitos de cualquier número de salidas.

B. *La implementación*

Como ya se mencionó, se trató de 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:

1. 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.

[1] Un método "avaricioso" es un algoritmo de aprendizaje por refuerzo muy conocido, en el cual siempre se escoge la acción cuya recompensa estimada sea la más alta [8].

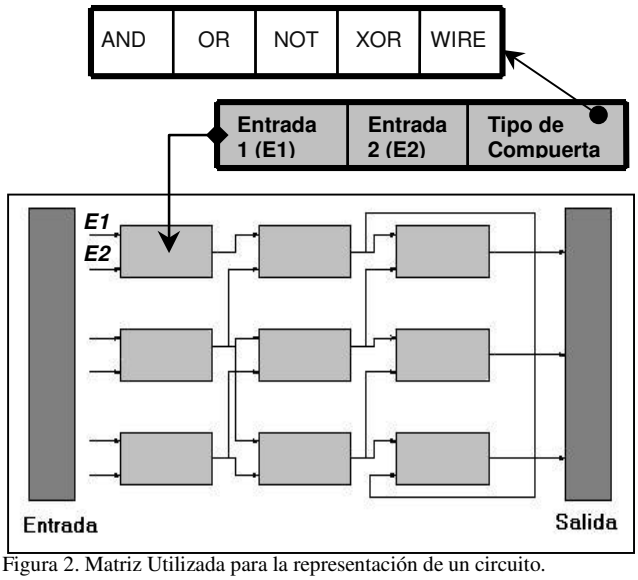


Figura 2. Matriz Utilizada para la representación de un circuito.

2. Un estado o ciudad es una columna, el cual está compuesto por varios elementos a los que se llama subestados, siendo éstos cada una de las compuertas de una columna. Los primeros *N* subestados (siendo *N* 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.
3. La distancia se mide en este caso entre subestados. Es un valor heurístico que corresponde a la diferencia en el número de aciertos (producidos con respecto a la salidas de la tabla de verdad) entre dos subestados.
4. A diferencia del problema del viajero, en una misma ruta (circuito), no tienen que visitarse todos los estados.
5. Las feromonas se guardan en una registro llamado *Trails*. La longitud de este registro corresponde al número de salidas del circuito. Cada elemento de *Trails* es a su vez una matriz tridimensional. Las dimensiones corresponden a *tipo de compuerta*, *combinaciones de entrada* y *aciertos anteriores*, respectivamente.

C. La construcción de una solución (ruta)

La elección de un estado es un proceso que se hace por partes (subestado por subestado). 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:

$$p(k_{i,j,l}) = f_{i,j,l} \cdot h_{i,j,l}$$

donde:

- k es la hormiga o agente el cual está construyendo la ruta o circuito.
- $f_{i,j,l}$ es la cantidad de feromona que hay entre los estados *i,j* en el subestado o fila *l*.
- $h_{i,j,l}$ es la distancia entre los estados *i* y *j* en el subestado o fila *l*.

Una vez que se ha asignado un factor de selección a todas las combinaciones de compuerta y entradas posibles, se elige cuál de ellas se queda en la posición en juego. Para esto se hace una selección por ruleta[2] [1]. Esto se repite con todos los subestados que pertenezcan a una de las filas que represente una salida del circuito. Los demás subestados, se eligen aleatoriamente.

Lo anterior se repite sucesivamente 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 de feromona en los rastros, simulando la evaporación de los rastros dejados por las hormigas reales al paso del tiempo. Para ello se usa la siguiente fórmula:

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

donde:

- α es un valor entre 0 y 1, llamado factor de evaporación.
- m se refiere al número de agentes.
- $\sum_{k=1}^{m} f_{i,j,l}$ 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 (o sea, no acierta a todas las salidas):

$$f^{k}_{i,j,l} = f^{k}_{i,j,l} + (Aptitud(k))$$

   - Si el circuito es factible (o sea que acierta a todas las salidas) :

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

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

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

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:

[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 de la población [1].

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, se trata del 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. Los WIRE no son contados como compuertas.

V. Resultados con algunos circuitos

Como se indicó antes, con las primeras versiones de nuestro sistema se podían resolver únicamente circuitos de una sola salida [9], mientras que ahora podemos resolver circuitos de cualquier número de salidas. 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. Las heurísticas como el algoritmo genético y el Ant System tienden a buscar ciertos patrones de diseño que hacen posible reutilizar ciertas porciones del circuito a fin de reducir su número total de compuertas [10]. Con la finalidad de validar nuestra técnica se utilizaron varios circuitos de prueba que han sido resueltos antes por diseñadores humanos y por diferentes tipos de algoritmos genéticos. Cada ejemplo fue probado con diferentes valores para los parámetros (tamaño de la matriz, factor de evaporación, tamaño de la población e iteraciones). Con cada ejemplo se mencionan los valores con los cuales se obtuvo la mejor solución.

A. Sumador de dos bits (Ejemplo 1)

El primer circuito de múltiples salidas con que se probó el AS fue un sumador de 2 bits el cual consta de 4 entradas y 3 salidas. La tabla de verdad que se muestra en la tabla 1, corresponde a este circuito.

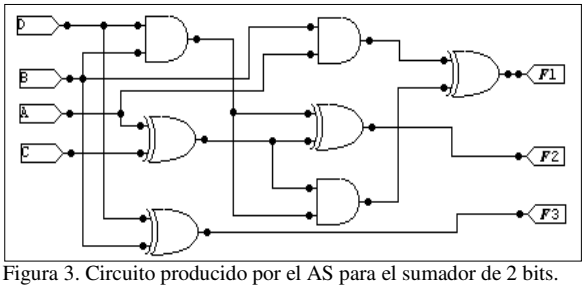


Figura 3. Circuito producido por el AS para el sumador de 2 bits.

La representación gráfica del circuito dado como mejor solución por el AS se muestra en la figura 3 y la comparación del resultado producido por el AS, un AG tradicional con representación binaria (BGA) y un diseñador humano (usando Mapas de Karnaugh), se muestran en la tabla 2. 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. La solución dada por el diseñador humano, emplea casi el doble de compuertas que las soluciones dadas por el AS y el BGA.

Tabla 1. Tabla de verdad del Sumador de 2 bits

Entradas				Salidas		
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

Para obtener la solución del AS presentada aquí, se utilizó una Matriz de 8×8, un factor de evaporación de 0.75, una población de 50 hormigas y 10 iteraciones. El tiempo promedio que tardó el AS en resolver este circuito fue de 231 segundos (3.85 minutos).

Tabla 2 Comparación de resultados entre el AS, un BGA y un diseñador humano para el ejemplo 1

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	**AS**
$F1=AC+ BD(A \oplus C)$ $F2=(A \oplus C) \oplus BD$ $F3= B \oplus D$	$F1= (DB\ (A \oplus C)) \oplus AB$ $F2 =DB\ \oplus (A \oplus C)$ $F3= D \oplus B$
7 Compuertas (3 ANDs, 1 OR, 3XORs)	7 Compuertas (3 ANDs, 4 XORs)

B. Circuito de Katz de múltiples salidas (ejemplo 2)

El segundo ejemplo es un circuito que consta de 4 entradas y 3 salidas (tabla 3).

El resultado producido por el AS es comparado con un MGA y dos diseñadores humanos (usando, el primero Mapas de Karnaugh y álgebra Booleana y el segundo el método Quine-McCluskey) en la tabla 4.

Tabla 3. Tabla de verdad del segundo ejemplo

Entradas				Salidas		
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. COMPARACIÓN DE RESULTADOS ENTRE EL AS, UN MGA Y DOS DISEÑADORES HUMANOS PARA EL EJEMPLO 2.

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

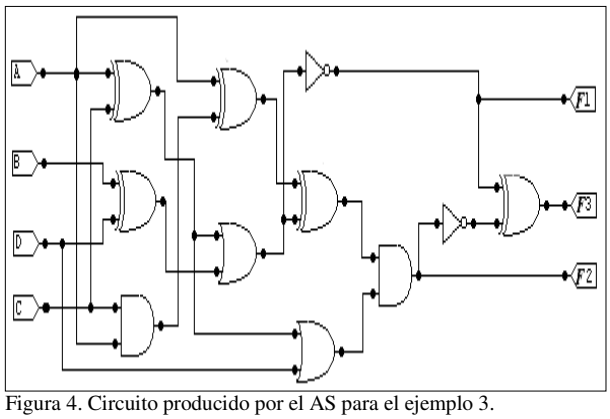


Figura 4. Circuito producido por el AS para el ejemplo 3.

La representación gráfica del circuito dado como mejor solución por el AS se muestra en la figura 4. 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.

VI. ANÁLISIS DE LA INFLUENCIA DE LOS VALORES DE LOS PARÁMETROS EN EL DESEMPEÑO DEL ALGORITMO

Nuestra aplicación funciona mediante la combinación de subcircuitos (compuertas o pedazos de circuitos), que construyen las hormigas o agentes, guiadas en gran medida por los rastros que van dejando. Por lo anterior suponemos de antemano que entre más hormigas se tengan en *la población*, más veces se repita la construcción de una ruta por cada hormiga (*mayor* número de *ciclos* o *iteraciones*) y/o mayor sea el *tamaño de la matriz*, habrá mayor cantidad de oportunidades de combinar subcircuitos, por lo que se espera que haya mayor porcentaje de convergencia. Sin embargo, también se espera un incremento en el tiempo necesario para concluir un experimento.

De antemano suponemos también 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, pero la operación es la misma 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.

Formalmente, las hipótesis que formulamos y que trataremos de sustentar son las siguientes:

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 un experimento.
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 un experimento.
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 un experimento.
7. El factor de evaporación influye sobre el porcentaje de convergencia.
8. El factor de evaporación NO influye sobre el tiempo necesario para terminar un experimento.

A. *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 para cada factor, los cuales representarán la variabilidad que cada uno de dichos factores puede tener; la tabla 5 muestra dichos valores.

La selección de los valores de dichos factores fue con base en la experiencia que se había obtenido en los experimentos realizados inicialmente para probar el AS.

TABLA 5. VALORES UTILIZADOS EN LOS PARÁMETROS

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

B. Selección de la variable de respuesta

En nuestro caso, lo que nos interesaba era determinar la mejor combinación de parámetros en función del porcentaje de convergencia y costo computacional (en cuestiones de tiempo solamente). Por lo tanto, nuestras variables de respuesta son el *porcentaje de convergencia* y el *tiempo promedio necesario* para concluir un experimento.

C. Elección del diseño experimental

Nuestro siguiente paso fue el diseño experimental que consistió en elegir con qué circuitos experimentar. Se decidió utilizar cinco circuitos (incluyendo los dos mostrados en el apartado anterior) 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.

Para cada circuito se decidieron realizar cuarenta experimentos con cada combinación de valores en sus parámetros y variando la semilla de números aleatorios[3]. Como son cuatro parámetros y tres valores para cada uno de ellos, tenemos 3^4 combinaciones, es decir 81 combinaciones de valores de los parámetros. Por lo tanto, tenemos 3240 experimentos para cada circuito (81 combinaciones por 40 experimentos). Así, con los 5 circuitos se tiene un total de 16200 experimentos.

De cada experimento, se recuperó la mejor solución o ruta encontrada y los siguientes datos de dicha solución:

- Aptitud.
- Número de compuertas si la solución es válida ó –1 en caso contrario.
- Iteración o ciclo en el que fue encontrada.
- Número de violaciones si la solución es inválida ó 0 en caso contrario.
- El tiempo necesario para terminar el experimento.

D. Realización del experimento

Una vez concluidos todas los experimentos de cada circuito, se usó el conjunto de experimentos obtenido por cada combinación de valores en los parámetros para calcular:

- Tiempo promedio necesario para terminar un experimento.
- Porcentaje de convergencia. Es decir, el porcentaje de experimentos que obtuvieron una solución válida.
- El número de compuertas de la mejor solución obtenida de los 40 experimentos.

E. Análisis estadístico de los resultados en cuanto al tiempo

Con el fin de estudiar el efecto de los valores de los parámetros con respecto nuestras variables de respuesta, se implementó un análisis de varianza (ANOVA) [4]. 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.

TABLA 6. MEDIAS DE TIEMPO (EN SEGUNDOS) PARA DE LOS CIRCUITOS, EN LAS COMBINACIONES DE MATRIZ Y POPSIZE, PARA CADA MAX_CIC.

	Parámetros		**Tiempo**		
No. De Combin.	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**

El comportamiento de la variable tiempo, se veía afectado conforme aumentaba Max_Cic, Matriz y PopSize, mientras que el valor de Evap, no reflejaba influencia sobre él (esto sucedió en general para todos los circuitos). Por ende, 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 uno de los circuitos como ejemplo.

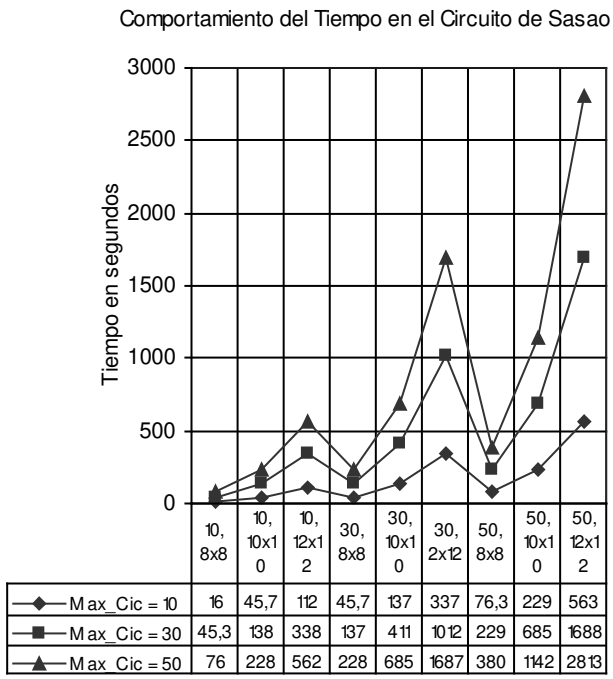
En general, a medida que aumentan los ciclos, el tiempo invertido en promedio tiende a aumentar (ver la última fila de la tabla 6). Lo mismo podemos observar dentro de cada ciclo: a medida que aumenta el tamaño de la matriz o el de PopSize, el tiempo promedio invertido en cada experimento es más alto.

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 6. 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

[3] La semilla de números aleatorios es un número fraccionario que se usa para inicializar el sistema de generación de números aleatorios.

debido a que tanto Max_Cic como PopSize aumentan el número de intentos de construir una ruta o solución. Esto se puede apreciar tanto en la figura 5, como en la tabla 6.



	10, 8x8	10, 10x10	10, 12x12	30, 8x8	30, 10x10	30, 2x12	50, 8x8	50, 10x10	50, 12x12
Max_Cic = 10	16	45,7	112	45,7	137	337	76,3	229	563
Max_Cic = 30	45,3	138	338	137	411	1012	229	685	1688
Max_Cic = 50	76	228	562	228	685	1687	380	1142	2813

Figura 5. Gráfica de los tiempos de un circuito de ejemplo

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

- El *factor de evaporación*, NO influye sobre el tiempo necesario para terminar un experimento, tal y como lo esperábamos de acuerdo a la hipótesis inicial 8.
- 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 un experimento, 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 8×8 a 10×10, 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 10×10 a 12×12, se provoca un incremento en el tiempo superior al que provocan el incremento de PopSize o Max_Cic de 30 a 50. De este modo algunas veces, resulta más caro el incremento del tamaño de la Matriz (en cuanto a tiempo).

F. 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* respecto a 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, se procedía a identificar la combinación de los niveles de los parámetros que proporcionaran la media más alta del porcentaje de convergencia en el menor tiempo; es decir, la combinación que llegase a la región factible del problema con el menor costo computacional (medido en 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, ya que en todos los circuitos éste aparecía como un factor significativo en el porcentaje de convergencia, ya sea combinado 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, antes indicada.
- 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, de manera independiente y en algunos niveles de Max_Cic, tuvo un impacto significativo sobre la convergencia, mientras que en otros 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 una diferencia significativa. En el caso del circuito del ejemplo 3 (de la sección V), 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=12×12, Evap=0.75), se logró el mayor porcentaje de convergencia, pero sin diferencia significativa entre los resultados de los

demás valores de Evap y su combinación con Matriz.

VII. CONCLUSIONES Y TRABAJO FUTURO

El Ant System es un algoritmo adecuado para la solución de circuitos combinatorios al nivel de compuertas. El buen desempeño del algoritmo, sin embargo, está supeditado a una elección apropiada de parámetros, tal y como se indica a continuación.

El *tamaño de la matriz* juega un papel importante tanto en el porcentaje de convergencia como en el 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 tuviera el tamaño de la Matriz adecuado para el circuito a diseñar, se tendría que aumentar significativamente 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 disminución de la convergencia.

Podemos ver también que las soluciones producidas por nuestro algoritmo son razonablemente buenas con respecto a las de los diseñadores humanos, aunque inferiores (en general) a las producidas por un algoritmo genético. Esto indica que todavía se requiere mejorar la implementación a fin de aumentar su eficiencia. En ese sentido, nos interesa explorar el uso del Ant System con una representación de árbol que sea más flexible que la que usamos en este trabajo. También se explora el posible desarrollo de un híbrido con otra heurística o la incorporación de conocimiento del dominio dentro del algoritmo del Ant System [10].

VIII. RECONOCIMIENTOS

Agradecemos los comentarios y sugerencias de los dos revisores anónimos, los cuales contribuyeron a mejorar significativamente el contenido de este artículo.

El segundo autor agradece a CONACyT el apoyo proporcionado a través del proyecto NSF-CONACyT 32999-A.

IX. REFERENCIAS

[1] Goldberg, David E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesly Publishing Company, 1989.

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

[3] 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.

[4] Zebulum, Ricardo; Pacheco Marco and 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.

[5] 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.

[6] 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.

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

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

[9] 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 et al. (Eds.), *Evolvable Systems: From Biology to Hardware, Proceedings of the Third International Conference*, Edinburgh, Scotland, Springer-Verlag, pp. 21--30, April 2000.

[10] Islas Pérez, E.; Coello Coello, Carlos A. & Hernández Aguirre, A. "Extraction of Design Patterns from Evolutionary Algorithms Using Case-Based Reasoning", in Yong Liu, Kiyoshi Tanaka, Masaya Iwata, Tetsuya Higuchi & Moritoshi Yasunaga (Eds.), *Evolvable Systems: From Biology to Hardware, Proceedings of the Fourth International Conference*, Tokyo, Japan, Springer-Verlag, pp. 244—255, October 2001.

[11] 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.

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