Análisis de Máquinas de Estados Finitos Usando la Programación Evolutiva

Jose L. Aguilar C.

***Resumen*—El presente trabajo evalúa el uso de la Programación Evolutiva (PE) en el diseño de Máquinas de Estados Finitos (MEF). En este trabajo se hace un análisis de diferentes tipos de MEF: máquinas parcialmente especificadas, fuertemente conectadas, equivalencia entre máquinas, etc., y se explora el uso de la PE en el análisis de las mismas, proponiéndose un sistema computacional basado en ella para diseñar autómatas de estado finito.**

***Palabras claves*—Máquinas de Estado Finito, Programación Evolutiva.**

I. INTRODUCCIÓN

Una MEF es un transductor el cual opera sobre un conjunto finito de símbolos de entrada (alfabeto), posee un número finito de estados internos, y produce símbolos de salida desde un alfabeto finito [1-6]. Las MEF sirven, entre otras cosas, para modelar Sistemas Dinámicos a Eventos Discretos (SDED).

Algunas técnicas inteligentes como las Redes Neuronales, la Lógica Difusa y los Programas Evolutivos [7, 8] han sido utilizadas para crear modelos continuos, pero hasta el momento las técnicas inteligentes aplicadas al problema de modelado de MEF son muy limitadas. Así, el propósito de este trabajo es analizar el uso de la PE en el estudio de MEF.

La PE que tiene sus bases en la Computación Evolutiva, y no es más que un método de búsqueda y optimización. El planteamiento es hacer evolucionar un conjunto de individuos, cada uno representando una MEF, que a su vez es una posible solución del sistema que se desea identificar.

Este trabajo está organizado de la siguiente manera: Se hace una introducción al marco teórico que cubre este trabajo, después se presenta la propuesta. Seguidamente se plantea como diseñar MEF usando PE. Finalmente se presenta un estudio exhaustivo sobre la construcción de MEF usando la PE.

Artículo recibido el 18 de Diciembre de 2009.

JLAC está con el CEMISID, Departamento de Computación, Facultad de Ingeniería, Universidad de Los Andes, Núcleo La Hechicera, Mérida 5101,Venezuela, Tlf. +58-274-2402914, Fax: +58-274-2402872, E-mail: aguilar@ula.ve.

II. MARCO TEÓRICO

A. Máquinas de Estado Finito (MEF)

Una MEF es una lógica matemática. Es esencialmente un programa de computadora: representa una secuencia de instrucciones a ser ejecutadas, donde cada instrucción depende del estado actual de la máquina y del actual estimulo [1-6].

Las posibles entradas al sistema son una secuencia de símbolos seleccionadas desde un conjunto finito I de símbolos de entrada, y las salidas resultantes son secuencias de símbolos escogidas desde un conjunto finito Z de símbolos de salida. Cualquier "caja negra" que produce un símbolo de salida cuando un símbolo de entrada es aplicado y que satisface las propiedades anteriormente mencionadas es llamada Máquina Secuencial o Transductor Finito.

La selección de un conjunto de estados para representar una máquina dada no es un proceso único, esto no es una limitación muy seria, ya que el principal objetivo es describir el comportamiento general entrada-salida de la máquina en vez de su construcción.

Representación de Estado Finito

Formalmente, una MEF es una 5-tupla [3-6]:

$$M = (Q, I, Z, s, o)$$

Donde: Q es el conjunto de estados; I es el conjunto de símbolos de Entrada; Z el conjunto de símbolos de salida; s(QxI)->Q es la función de estado siguiente; o(QxI)->Z es la función de salida siguiente. El comportamiento de una MEF es descrito como una secuencia de eventos que ocurren en instantes discretos t=1, 2, 3, etc.

Cualquier 5-tupla de conjuntos y funciones que satisfaga la definición anterior será interpretada como una descripción matemática de la MEF, es decir, si es proporcionado un símbolo de entrada x estando en un estado q, el símbolo de salida estará dado por o(q, x), y la transición al siguiente estado estará dada por s(q, x). Sólo la información contenida en el estado actual describirá como actuará la máquina para un estimulo dado.

Así que una MEF no es más que un transductor que puede ser estimulado por un alfabeto finito de símbolos de entrada, que puede responder con un alfabeto finito de símbolos de salida y que posee un número finito de diferentes estados internos. El correspondiente par símbolo entrada-salida y

ISBN: 978-980-7185-1

transición al estado siguiente para cada símbolo de entrada, tomado sobre cada estado, específica el comportamiento de cualquier MEF, dado un estado inicial.

Existen tres técnicas comúnmente usadas para representar las propiedades analíticas de una máquina [3-6]: las tablas de transición, los diagramas de transición y las matrices de transición. En la Tabla 1 se muestra un ejemplo de una matriz de transición para una máquina con tres estados, cuyo alfabeto de símbolos de entrada es {0,1} y el alfabeto de símbolos de salida es el conjunto {α, β, δ}.

TABLA 1.
MATRIZ DE TRANSICIÓN

Edo. Actual/ Edo. Siguiente	A	B	C
A	1/β	0/β	
B		0/δ	1/α
C	1/δ	0/β	

B. Computación Evolutiva

Bajo el término de Computación Evolutiva se engloba a un amplio conjunto de técnicas de resolución de problemas complejos basados en la emulación de los procesos naturales de evolución [7, 9, 10].

El principal aporte de la Computación Evolutiva a la metodología de resolución de problemas consiste en el uso de mecanismos de selección de soluciones potenciales y de construcción de nuevos candidatos por recombinación de características de otros ya presentes, de modo parecido a como ocurre en la evolución de los organismos naturales.

Las implementaciones concretas de Computación Evolutiva se conocen como Algoritmos Evolutivos. El propósito de los Algoritmos Evolutivos es guiar una búsqueda estocástica haciendo evolucionar un conjunto de estructuras y seleccionando de modo iterativo las más aptas. Un Algoritmo Evolutivo se ejecuta sobre una población de individuos, que representan un conjunto de candidatos a soluciones de un problema. Dichos individuos son sometidos a una serie de transformaciones, con las que se actualiza la búsqueda, y después a un proceso de selección que favorece a los mejores individuos. Cada ciclo de transformación + selección constituye una generación. Se espera del Algoritmo Evolutivo que tras cierto número de generaciones (iteraciones), el mejor individuo esté razonablemente próximo a la solución buscada.

Los Operadores Evolutivos son los que realizan los cambios de la población durante la ejecución de un Algoritmo Evolutivo. Clásicamente, existen dos operadores genéticos: mutación (Es un operador unario que simula el proceso evolutivo que ocurre en los individuos cuando cambian su estructura genética.) y cruce (Es un operador normalmente binario, que permite representar el procesos de apareamiento natural usando operaciones sencillas. toman diversos componentes de distintos individuos para generar con ellos nuevos individuos, de tal manera que hereden características de sus padres). Sin embargo, han sido propuestos otros operadores que se acoplan a problemas particulares, por ejemplo: Dominación, Segregación, Translocación, Inversión, entre otros.

Los pasos de un Algoritmo Evolutivo son los siguientes:

- Generación de una población inicial, generalmente en forma aleatoria.
- Evaluación de los individuos
- Selección de algunos individuos de la población, a través de algún mecanismo.
- Modificación de los genes de los progenitores seleccionados usando los operadores genéticos.
- Generación de una nueva población mediante algún mecanismo de reemplazo.
- Verificación del criterio de parada del Algoritmo, y se regresa al paso 3, si es el caso.

Las principales técnicas de la Computación Evolutiva son [7]: los Algoritmos Genéticos propuestos por Holland, las Estrategias Evolutivas propuestas por Rochenberg y Schwefel, la Programación Genética propuesta por Koza, y la Programación Evolutiva propuesta por Fogel.

Programación Evolutiva (PE)

La Programación Evolutiva fue originalmente concebida por L. Fogel en 1960 [7, 9, 10, 11]. Es un mecanismo que hace evolucionar un conjunto de MEF. Esta técnica desarrolla un conjunto de soluciones, las cuales muestran un comportamiento óptimo con respecto a un ambiente y a una función deseada. A continuación se hace una explicación exhaustiva de como la Programación Evolutiva hace evolucionar las MEF.

- InitPopulation P(t): Comienza con una población de MEF que representan soluciones al problema en cuestión.
- evaluate P'(t): Cada máquina es evaluada por medio de una función específica, que calcula la capacidad del individuo para resolver el problema.
- Mecanismo de Selección: todos los individuos son seleccionados, ya que cada máquina genera un descendiente a través de un proceso de mutación que se aplica sobre ella.
- P'(t) = mutate P(t): cada miembro de la población es alterado a través de un proceso de mutación, el cruce no es utilizado. Clásicamente existen cinco tipos de mutaciones aleatorias:
  - Cambiar un símbolo de salida.
  - Cambiar la transición de un estado a otro.
  - Agregar un estado.
  - Eliminar un estado.
  - Cambiar el estado inicial.
- P(t+1) = survive P(t),P'(t): se escogen los individuos que van a sobrevivir para la siguiente generación. Normalmente se toman las k mejores máquinas de la población total. Generalmente el tamaño de la población permanece constante, pero no hay restricción en este caso.
- Stopping Criterion: El proceso termina cuando la solución alcanza una calidad predeterminada, cuando ha sido obtenido un número específico de iteraciones (generaciones), o algún otro criterio de parada.

En la PE no se utiliza el operador de cruce, ya que este operador produce MEF no validas (sin sentido). En la literatura se ha comprobado que con solo el operador de mutación es suficiente para hacer evolucionar una población de MEF [7, 9, 10, 11]

III. LA PROGRAMACION EVOLUTIVA PARA EL DISENO DE MÁQUINAS DE ESTADOS FINITOS

En esta sección se presenta como debe ser caracterizada la PE para poder ser usada en la construcción de MEF.

A. Diseño de la Estructura del Genotipo

El Genotipo en la PE debe representar una MEF, además debe ser capaz de generar una secuencia de salida debido a una de entrada (debe poseer una función de salida), dicho requisito es indispensable por el hecho de que se desea hacer un proceso de identificación. A continuación se presentan dos modelos de MEF: Mealy y Moore.

Modelo de Mealy (1955)

El modelo de Mealy no es más que un autómata generador de lenguaje representado por la siguiente 6-tupla [3- 6]:

$$M = (Q, I, Z, s, o, q(0))$$

Donde: Q es el conjunto de estados; I es el conjunto de símbolos de entrada; Z es el conjunto de símbolos de salida; s(QxI)->Q es la función de estado siguiente; o(QxI)->Z es la función de salida siguiente; q(0) es el estado inicial.

En esta estructura se debe notar que la salida depende de la entrada, esto quiere decir, que una máquina de Mealy sólo genera un símbolo de salida cuando es presentado un símbolo de entrada.

La semántica procedimental del modelo de Mealy es el siguiente, la máquina se encuentra en un estado q(0), posteriormente, cuando recibe un literal de entrada, entonces emite el símbolo de salida y transita al nuevo estado. El diagrama de transición y la tabla de transición de Mealy se observan en la Figura 1 y en la Tabla 2, respectivamente.

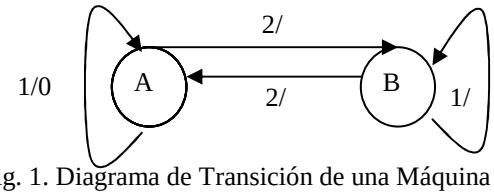

Fig. 1. Diagrama de Transición de una Máquina de Mealy

TABLA 2.
TABLA DE TRANSICIÓN DE UNA MÁQUINA DE MEALY

Estados / Entradas	1	2
A	A/0	B/1
B	B/1	A/0

Modelo de Moore (1956)

Una máquina de Moore es similar a una de Mealy, salvo en que la respuesta sólo depende del estado actual de la máquina y es independiente de la entrada. Una máquina de Moore es una estructura de la forma [3-]:

$$M = (Q, I, Z, s, o, q(0))$$

Donde: Q es el conjunto de estados; I es el conjunto de símbolos de entrada; Z es el conjunto de símbolos de salida; s(QxI)->Q es la función de estado siguiente; o(QxI)->Z es la función de salida siguiente; q(0) es el estado inicial.

El modelo de Moore opera de la siguiente manera, iniciando la máquina se encuentra en el estado q(0). Posteriormente, cuando recibe una literal de entrada, entonces transita al nuevo estado y después emite el símbolo de salida.

El diagrama de transición y la tabla de transición de Moore se observan en la Figura 2 y en la Tabla 3, respectivamente.

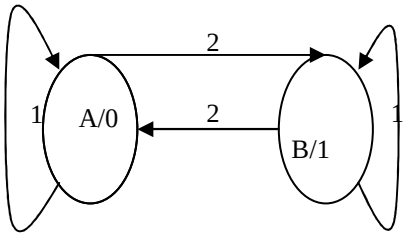

Fig. 2. Modelo de Moore

TABLA 3.
TABLA DE TRANSICIÓN DE UNA MÁQUINA DE MOORE

Estados / Entradas	1	2	Salida
A	A	B	0
B	B	A	1

Equivalencias entre los dos Modelos

Una MEF representada como un modelo de Mealy (Moore) tiene su equivalente representación en un modelo de Moore (Mealy), ya que ambas máquinas son completamente equivalentes (La comprobación de equivalencia entre las máquinas de Mealy y Moore se puede encontrar en [9, 11]). A causa de esta equivalencia, existen procedimientos sencillos de conversión entre un modelo y otro.

La razón fundamental de selección de qué tipo de estructura se utiliza para codificar el genotipo es por rendimiento computacional, una máquina de tipo Mealy generalmente requiere de menos estados que una de Moore para reconocer una señal de entrada y generar la salida correspondiente. Como se requiere trabajar con poblaciones relativamente grandes, entonces, no se desea disminuir la velocidad de búsqueda y optimización del algoritmo al introducir individuos muy complejos.

B. Inicialización de las Máquinas

En el sistema se implementaron dos formas de inicialización de la población de MEF:

- Completamente aleatoria, se asigna un estado inicial, después se escoge al azar un valor que será el número de estados que posee la máquina, que debe ser menor o igual al número máximo de posibles estados. Luego se comienza a llenar cada una de las celdas de la tabla de transiciones, la cantidad de columnas está previamente definida, ya que es el conjunto de símbolos de entrada. En cada intersección de estado con símbolo de transición se

encuentra la celda que indica hacía qué estado se transitará y cuál será el correspondiente símbolo de salida, estos dos elementos también son generados aleatoriamente. Por último, se crea una vector que contendrá las etiquetas (por llamarlo de alguna manera) respectivas para identificar los estados, este vector no puede contener elementos repetidos (estados repetidos), y se va actualizando junto con la tabla de transiciones al momento que se efectúe una mutación sobre el individuo (sólo si la mutación es agregar o eliminar un estado).

- La otra manera es inicializarla desde una población ya existente.

C. *Las Mutaciones*

Se usan las cinco mutaciones enunciadas por Fogel, aleatoriamente se escoge que tipo de mutación se va a aplicar sobre cada individuo.

D. *Función de Costo*

Posee dos atributos que son dos cadenas de caracteres, la secuencia de entrada y la secuencia de salida, que son compartidas para todos los individuos de la población. La capacidad de cada máquina se mide comparando la secuencia de salida que genera el individuo con la secuencia de salida de referencia, cuando es enfrentado a la secuencia de entrada. Este procedimiento se ejecuta de la siguiente manera: se aplica el primer símbolo de entrada al individuo que se encuentra en un estado inicial, si ese símbolo no es reconocido por el autómata se penaliza con un valor positivo, de lo contrario, si el símbolo es reconocido se compara el símbolo de salida que genera el autómata con el símbolo de salida de referencia, si los símbolos son diferentes el individuo es penalizado con una valor positivo, pero, sí los símbolos de salida coinciden la máquina es premiada con una valor negativo. Luego, se verifica hacia qué estado transita el autómata y se repite el proceso para el siguiente símbolo de entrada-salida correspondiente hasta que se hayan alcanzado el final de las secuencias.

Como se pudo constatar en el procedimiento anteriormente relatado, la función de costos se está minimizando, por lo cual aquellos individuos que posean funciones con valores más negativos serán los más aptos de la generación.

Además se incorporó en la función de costos la capacidad de recompensar a aquellos autómatas que poseen menos estados, esto es opcional y es con la finalidad de obtener máquinas de estado finito que no sólo transducen una secuencia de entrada, sino que también sean simples (desde el punto de vista del tamaño).

IV. ANALISIS DE MEF USANDO PE

Para el análisis que presentamos a continuación, se uso el sistema computacional propuesto en [12], el cual permite construir MEF usando la PE. La pantalla del sistema es mostrada en la Figura 3.

Se van a mostrar una serie de experimentos que se realizaron usando el sistema propuesto en [12], para verificar la eficacia de la PE para modelar MEF. Los resultados obtenidos (o modelos) son comparados directamente con el autómata real (obtenido de la literatura [3-6]). Por cada experimento se mostrara la MEF real y la MEF obtenida por la PE. Además, se dará la lista de algunos de los parámetros usados, particularmente las secuencias de entrada-salida. Todos las MEF estudiadas en esta parte están propuestas en [3-6].

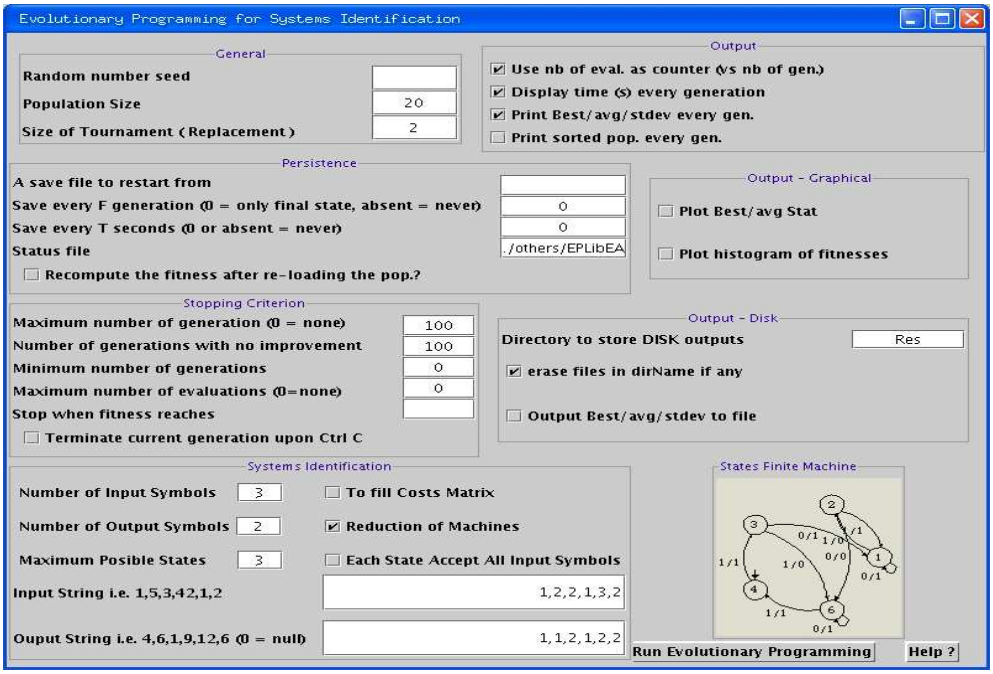

Fig 3. Interfaz Gráfica de Usuario

A. Máquinas Completamente Especificadas

Son aquellas donde cada estado de la máquina reconoce todos y cada uno de los eventos del conjunto de entrada, y además genera un símbolo de salida cuando se realiza una transición, ya sea a otro estado de la máquina o al mismo. En la Figura 4 se muestra un autómata que pertenece a esta categoría.

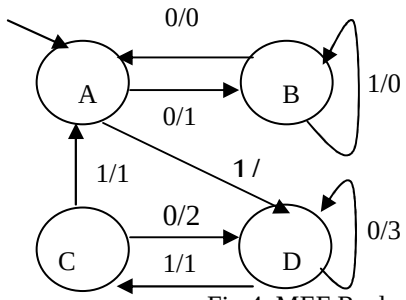

Fig 4. MEF Real

Los parámetros más importantes usados en este experimento son: Tamaño de la Población= 20 , Número de Generaciones= 695, Secuencia de Entrada=0,0,0,1,0,1,0,0,1,0,0,1,1,0; Secuencia de Salida= 1,0,1,0,0,0,3,3,1,2,3,1,1,1. La MEF identificada se muestra en la Figura 5.

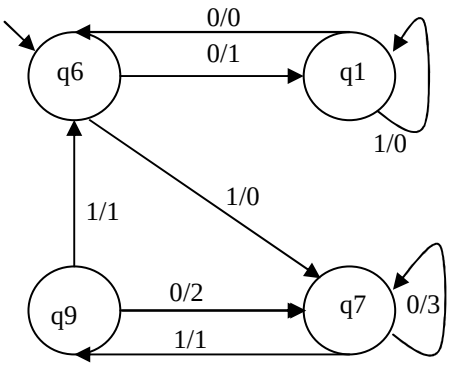

Fig. 5. MEF obtenida por la PE

Es evidente que es igual al modelo real, no es necesario comprobar la aptitud del modelo generado ante una secuencia de validación. En todas las experiencias de este tipo la máquina identificada es completamente igual a la MEF real (ver [12] para mas detalles).

B. Máquinas Parcialmente Especificadas

Una máquina parcialmente especificada es aquella en la cual una o más transiciones son indefinidas o no especificadas, o la máquina reconoce el símbolo de entrada pero no genera una salida. La Figura 6 es el primer ejemplo de máquinas parcialmente especificadas.

Algunos de los parámetros usados para su identificación fueron: Tamaño de la Población= 20; Número de Generaciones= 46, Secuencia de Entrada= u2, u1, u2, u1, u2; Secuencia de Salida=-,-, y4, y3,-.

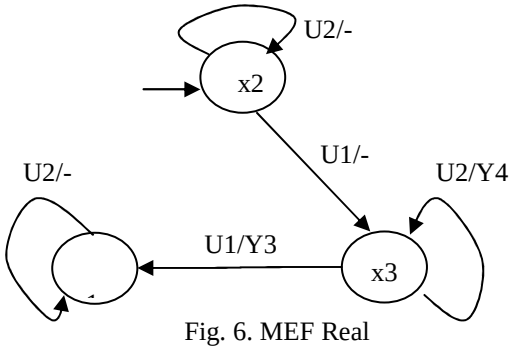

Fig. 6. MEF Real

A diferencia de la anterior prueba, la MEF resultante no es igual al modelo real (ver Figura 7), por esa razón es conveniente examinar el comportamiento del modelo resultante ante una secuencia entrada-salida de validación. Se selecciono la secuencia: Secuencia de Entrada= u1, u1, u2; Secuencia de Salida = -, y 3,-. La MEF identificada traduce de manera satisfactoria la secuencia de validación aplicada.

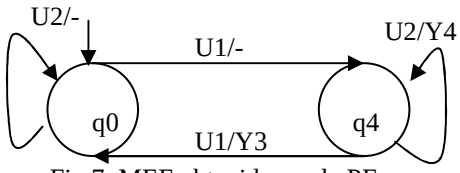

Fig 7. MEF obtenida por la PE

Este experimento muestra una característica muy especial, traduce la secuencia de entrada en una de salida previamente especificada, pero requiriendo menos estados que el modelo real. Se ha encontrando así otra utilidad de la PE, optimiza autómatas desde el punto de vista de estados.

Esta reducción o realización de MEF se comenta luego, por ahora, el experimento demuestra este hecho, pasando de un autómata de tres estados a uno con tan sólo dos, este último reconociendo la secuencia de validación aplicada. La MEF identificada en la experiencia de esta parte introduce además una nueva categoría de máquinas, estas son aquellas que pueden transitar de un estado a otro con distintos eventos.

C. Máquinas Fuertemente Conectadas

Una máquina con un conjunto Q de estados es fuertemente conectada sí para cualquier par de estados Qi y Qj $\in$ Q, existe al menos una secuencia de entrada tal que el estado Qj pueda ser alcanzado por Qi [3-6]. Las máquinas fuertemente conectadas son interesantes (teórica y prácticamente) porque tienen la propiedad de que cualquier estado de la máquina puede ser alcanzado desde cualquier otro estado. En la teoría de Control Automático el concepto de "Sistema Controlable" es análogo a la noción de máquina fuertemente conectada.

El experimento que se muestra a continuación introduce máquinas de un tipo muy especial (Figura 8), en estas la transición entre dos estados determinados puede no ser única, y además el símbolo de salida que genera el autómata cuando está pasando de un estado a otro también puede no ser único, y

va a depender de cual evento (de una lista de transiciones que reconoce) es el que está ocasionando el cambio de estado.

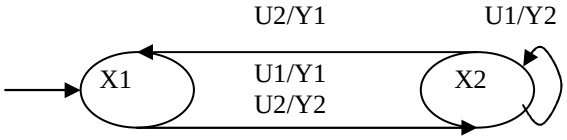

Fig 8. Máquinas de Estado Finito Real

La MEF identificada se muestra en la Figura 9. Los parámetros más importantes usados en esta parte son: Tamaño de la Población= 20; Número de Generaciones= 777; Secuencia de Entrada= u1, u1, u2, u2, u1, u1, u2, u1; Secuencia de Salida= y1, y2, y1, y2, y2, y2, y1, y1.

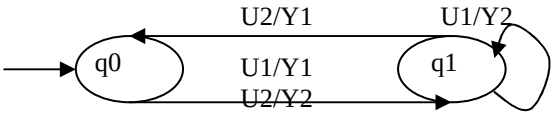

Fig 9. Máquinas de Estado Finito obtenida por la PE

D. Realización de Máquinas

Antes se había mencionado la posibilidad de que la herramienta diseñada usando la PE era capaz de reducir el número de estados que un autómata necesita para definir un comportamiento entrada-salida. Esto es viable, ya que el algoritmo puede encontrar máquinas equivalentes con menos estados que el autómata en estudio. Esta reducción no es fortuita, y generalmente se debe a la presencia de estados equivalentes en una MEF. Dos estados son equivalentes si tiene idénticas características entrada-salida, es decir, dos estado Qi y Qj son equivalentes sí y solo sí cada posible secuencia de entrada aplicada sobre la máquina produce la misma secuencia de salida cuando sus estados iníciales son Qi o Qj [3-6].

Para este experimento no se generó secuencia de validación, sin embargo, al observar cada uno de los autómatas reales y los identificados, se puede constatar que no son iguales, entonces (ver Figuras 10 y 11), ¿Por qué no es necesaria al menos una secuencia de validación?, la respuesta a esta pregunta es que la máquina real y la máquina identificada son máquinas equivalentes (eso lo logra descubrir la PE).

Los parámetros usados en este caso más importantes son: Tamaño de la Población= 20 ; Número de Generaciones= 139; Secuencia de Entrada= u1, u2, u1, u2, u1, u2, u1, u2, u1, u2, u1, u2, u2, u2, u2, u2, u2, u2; Secuencia de Salida= y1, y1, y1, y2, y1, y1, y1, y2, y1, y2, y1, y2, y1, y2.

Si se observa con atención el experimento, se podrán encontrar los estados equivalentes en cada MEF. Por ejemplo, el autómata real posee estados como X1, X3 y X5 que son equivalentes entre sí, además los estados X2, X4 y X6 también lo son. Analizando la MEF generada, se comprueba que con sólo dos estados es suficiente para describir el comportamiento del autómata, los estados X1, X3 y X5 del modelo real, son representados por su equivalente q2 en el modelo identificado, lo mismo sucede para los estados X2, X4 y X6, con su equivalente q4.

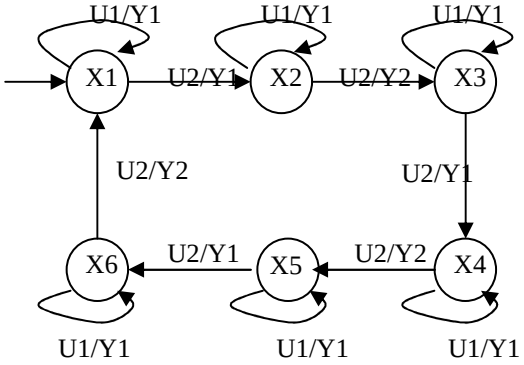

Fig 10. MEF Real

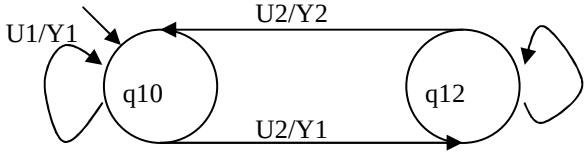

Fig 11. MEF obtenida por la PE

D. Modelado de sistemas dinámicos a eventos discretos

Con el último experimento no se introduce un tipo de máquina diferente a las ya estudiadas, sólo se desea conocer la respuesta de la herramienta al trabajar con modelos de sistemas dinámicos a eventos discretos mucho más complejos que los abordados antes, por esa razón se presenta la MEF de la Figura 12, la cual presenta una interconexión de estados algo más elaborada. El autómata resultante (ver Figura 13) para el experimento, fue validado según la secuencia siguiente: Secuencia de Entrada= i3, i1, i2, i2, i3, i3, i1, i1, i3, i1, i1, i1, i1, i2, i1, i3, i3, i1, i2, i1, i2, i1, i3; Secuencia de Salida = 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, , 0, 0, 0, 1, 1, 0, 0, 0, 0.

Algunos de los parámetros usados en este experimento fueron: Número de Generaciones= 4354 ; Secuencia de Entrada= i1, i1, i3, i3, i1, i2, i2, i3, i1, i2, i3, i3, i3, i1, i3, i3 i3, i1, i1, i1, i1, i1, i3, i1, i2, i2, i1, i1, i1, i3, i3, i3, i2, i1, i3, i1, i2, i3, i2; Secuencia de Salida= 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1,0.

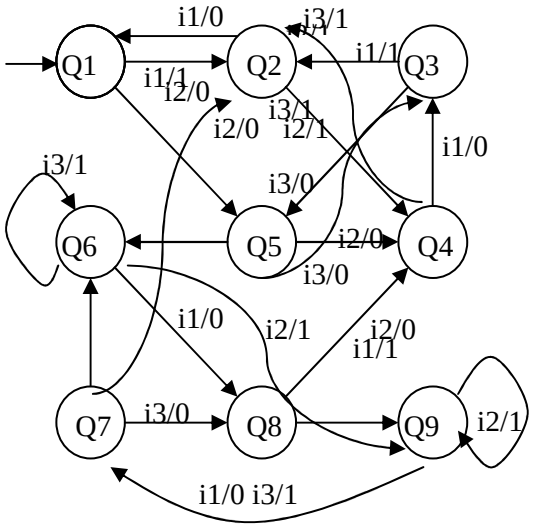

Fig 12. MEF Real

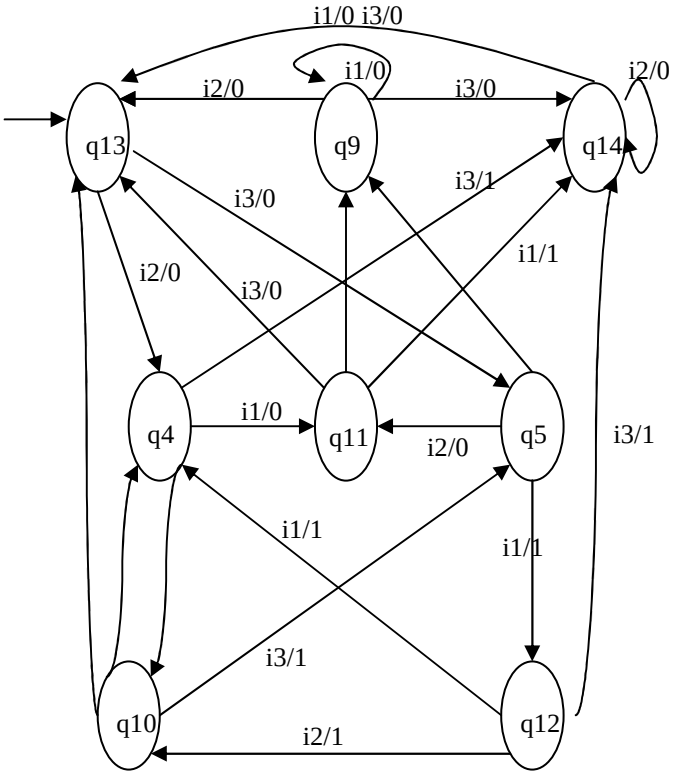

Fig 13. MEF obtenida por la PE

V. CONCLUSIONES

Se ha analizado a lo largo de esta investigación la incidencia de la PE para la resolución de problemas referentes a la construcción de MEF, obteniendo resultados satisfactorios, demostrando así la eficiencia que aporta un algoritmo evolutivo en esta tarea. Durante el desarrollo de este trabajo se fueron agregando características para evaluar el desempeño de la PE en tareas de identificación de MEF que modelen sistemas reales, por consiguiente, se hizo un amplio análisis de diferentes tipos de MEF.

Entre otras cosas, se logro comprobar que la PE logra generar MEF equivalentes a las existentes de MEF de sistemas reales, con menos estados que los que presentan las originales. Además, la técnica introducida en este trabajo muestra una combinación de simplicidad, facilidad, rapidez y eficacia, que le permite competir con cualquier otra manera de analizar MEF.

REFERENCIAS BIBLIOGRÁFICAS

[1] I. Aleksander, F. Hanna. "**Automata Theory: An Engineering Approach**". Computer Systems Engineering Series. Crane, Russak & Company, New York, 1975.
[2] M. Simon. "**Automata Theory**". World Scientific, 1999
[3] G. Berry, R. Sethi. "**From Regular Expressions to Deterministic automata**" Theoretical Computer Science. Vol. 48. pp. 117-126. 1987.
[4] R. Kain. "**Automata Theory: Machines and Languages**". McGraw-Hill, New York, 1972.
[5] Z. Kohavi. "**Switching and Finite Automata Theory**". McGraw-Hill, New York, 1970.
[6] A. Malcher, "**Minimizing Finite Automata is Computationally Hard**". Theoretical Computer Science. Vol. 327. n.º 3. pp. 375-390, 2004.
[7] B. Ramón. "**Autómatas y Lenguajes. Un Enfoque de Diseño"**. Tecnológico de Monterrey, México, 2003.
[8] J. Hopcroft, R. Motwani, J. Ullman "**Introduction to Automata Theory, Languages, and Computation"** (2nd Edition). Pearson, 2000.
[9] J. Aguilar, F. Rivas (Eds.). "**Introducción a las Técnicas de Computación Inteligente**", Mérida, Venezuela, MERITEC, 2001.
[10] M. Cerrada, J. Aguilar, "**Genetic Programming-Based Approach for System Identification**", in Advances in Fuzzy Systems and Evolutionary Computation, Artificial Intelligence (A Series of Reference Books and Textbooks), (Ed. N. Mastorakis), World Scientific and Engineering Society Press, pp. 329-334, 2001.
[11] D. Fogel. **Evolutionary Computation**. IEEE Press, 1995.
[12] D. Fogel. "**Handbook of Evolutionary Computation**", Oxford University, 1997.
[13] J. Heitkotter, D. Beasley. "**Hitch Hiker's Guide to Evolutionary Computation**", 2000.
[14] A. Piña, J. Aguilar, J. Cardillo, "**La Programación Evolutiva en Identificación de Sistemas**", Technical Report 21-2004, CEMISID, Universidad de los Andes, 2007.