

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/264496425>

GENERADOR DE AUTOMATAS CELULARES

Conference Paper · March 2011

CITATIONS

0

READS

580

2 authors, including:



[Cesar Hernán Rodríguez Garavito](#)

Universidad de La Salle

17 PUBLICATIONS 58 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



LSI IVVI 2.0 [View project](#)



Estabilizador de imágenes activo LQR-FTS [View project](#)

GENERADOR DE AUTOMATAS CELULARES

Cesar H. Rodriguez G.

Facultad de Ingeniería, programa de Ingeniería en Automatización. Universidad de La Salle.
Bogotá. D.C. Colombia.

y

José A. Tumialan B.

Facultad de Ingeniería, programa de Ingeniería en Automatización. Universidad de La Salle.
Bogotá. D.C. Colombia.

RESUMEN

Este artículo presenta el diseño, la implementación y los resultados obtenidos, de un generador de autómatas celulares (GAC), implementado a través de un algoritmo genético (AG). Se busca crear una herramienta que permita encontrar Autómatas Celulares (AC) con características definidas según la aplicación para la cual se requieran, en este caso, AC clase III, dinámicos, cíclicos y con el mayor periodo de vida posible. Dentro de los resultados más relevantes se destaca la metodología de sintonización de la búsqueda, basada en AG, y el hecho de que recurrentemente fue hallado uno de los Autómatas celulares con mayor periodo reportado, el R-pentonomio con un periodo de vida de 1103 generaciones.

Palabras clave: Autómatas Celulares, Algoritmos Genéticos, computación Universal.

1. INTRODUCCION

En palabras de Stephen Wolfram, uno de los científicos más prominentes en el estudio de los autómatas celulares, éstos “pueden ser vistos como unidades de cómputo, donde los datos, representados por las condiciones iniciales del entorno, son procesados a través de la evolución temporal del ambiente”. Los autómatas celulares han sido objeto de un estudio riguroso, ya que son una forma discreta de representar un mundo constituido por sistemas dinámicos, cuyas interacciones crecen en orden exponencial y por lo tanto son intratables, aun cuando sea considerado solo un conjunto reducido de variables. En este contexto el primer AC generalizado fue el juego de la vida, Game Life, propuesto por el matemático británico John Conway en 1960. Consiste en un mundo virtual donde elementos unicelulares nacen y mueren. Este no es un juego propiamente dicho donde dos competidores buscan un objetivo común, y la victoria de un jugador implica la derrota del adversario. El juego de la vida es una red dinámica donde los elementos discretos que existen en cada posición pueden cambiar de estado en función del contenido de las celdas vecinas, existen diversas definiciones de vecindario de un elemento en la cuadrícula, pero el adoptado como modelo estándar es el de Moore [1], ver figura 1. Así, el nuevo estado del mundo cambia bajo la aplicación a cada posición de la red de las leyes de la vida o leyes de conway [2].

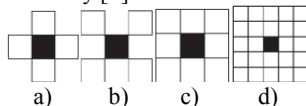


Figura 1 a),b) Vecindad Von Neumann y su variante. c) Vecindad de Moore. d) Vecindad extendida de Moore.

Las reglas del juego de la vida obedecen a una máquina de estados tipo Mealy como se muestra en el diagrama de estado de la figura 2, allí, un pixel del mundo se enciende o vive, si el número de vecinos que tiene a su alrededor es igual a 3 (nacimiento); un pixel del mundo que está vivo en un instante dado, sobrevivirá si su número de vecinos es igual a 2 ó a 3 (supervivencia), en otro caso éste se apaga o muere (muerte).

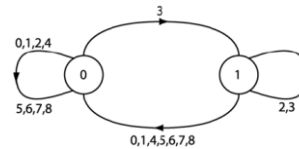


Figura 2. Diagrama de estados del juego de la vida.

En general los AC se clasifican según su dimensión y sincronía, sin embargo desde los 1980's se empezaron a clasificar por Wolfram, en Clase I, II, III y IV [6]. En la clase I se encuentran aquellos AC que tienden a un estado estático, la clase II se caracterizan por exhibir comportamientos cíclicos, los de clase III, evolucionan de forma tal que describen formas aleatorias, finalmente, los de clase IV reflejan como la vida surge, presentan la ocurrencia de formas definidas que se mueven e interactúan en el entorno de evolución.

Los autómatas celulares son de gran interés porque presentan un nuevo paradigma de programación, tal como se ha visto a lo largo de la historia con el paso de la programación en texto, a la programación orientada a eventos y posteriormente a la programación orientada a objetos. Así, el siguiente paso del desarrollo conceptual de la programación podría encontrarse en la computación espacio temporal.

En [3] se ha demostrado como se es posible establecer un símil entre los elementos estructurales del juego de la vida y la máquina de Turing, se comprueba mediante la implementación de un algoritmo para la duplicación de cadenas de unos y ceros, que el juego de la vida es una máquina de Turing, esto quiere decir que cualquier algoritmo es susceptible de ser implementado en esta arquitectura. En [4] por su parte, se presenta los fundamentos de la implementación de las compuertas universales, AND, OR y NOT, sobre las cuales es posible implementar cualquier función booleana.

Existen trabajos sobre otras estructuras computables que pueden ser implementadas sobre el juego de la vida [5], allí se demuestra la potencialidad de las redes neuronales celulares RNC, como una máquina de Turing, ya que las RCN son implementables en el juego de la vida. Otro enfoque en el estudio de AC, son los estudios sobre como generar estrategias de cómputo que emergen de la generación de autómatas a través de AG, este trabajo abordado por Melanie [9], pretende entender las reglas para la aparición de inteligencia en sistemas complejos.

Por otro lado, trabajos como el reportado en [1], muestran como la arquitectura de los AC puede ser explotada en cuanto a su facilidad para ser embebida en plataformas paralelas, se presentan tres nuevos modelos AC sujetos a conjuntos de reglas diferentes al juego de la vida: Espacio cíclico, Máquina de Hodepodge y Greenburg-Hasting.

Los autómatas que permitieron concebir la implementación de una máquina de Turing en un ambiente espacial de agentes discretos, Autómatas Celulares clase IV, fueron en primer lugar, los llamados gliders o planeadores, los que mostraron la noción de transferencia de información en el espacio, ya que se asimilan a naves que viajan diagonalmente con un periodo de cambio de cuatro formas y avanzan una celda cada dos iteraciones. Más adelante con la aparición de autómatas

generadores de planeadores, gliders gun, se abrió el horizonte a la implementación de funciones lógicas y estructuras de computación de instrucciones, maquinas de estados finitos.

Hay alrededor de 700 esquemas de AC reportados en [8]. Gracias a los comportamientos descritos en este compendio, la implementación de una maquina de Turing es posible, ya que si sus elementos fundamentales están presentes en el comportamiento de algún autómatas o en alguna interrelación de estos, la implementación se hace factible.

Dado que los elementos constitutivos de una máquina de Turing son AC clase III y Clase IV, se propone desarrollar una herramienta capaz de generar AC con estas características, ya que no existe definiciones matemáticas para ello y se debe realizar una búsqueda exhaustiva de todas las posibles configuraciones a lo largo de un espacio discontinuo, para efectos computacionales infinito, discreto y dinámico.

El desarrollo del Generador de AC, GAC, parte de una implementación desarrollada en lenguaje gráfico, sección II, junto con el algoritmo de búsqueda basado en AG, sección III, y finalmente, la validación de la herramienta a través de una serie de experimentos para sintonización del AG y corrida de la búsqueda sobre AC de 6 por 6 células.

2. IMPLEMENTACIÓN DEL GENERADOR DE AUTÓMATAS CELULARES (AC)

Como se ha visto, el descubrimiento de nuevos autómatas celulares que muestren comportamientos no reportados, puede inducir nuevas aplicaciones aun no exploradas. La búsqueda de estas configuraciones es una tarea compleja, por cuanto no es posible determinar heurísticas que reduzcan el espacio de búsqueda. El problema a resolver entonces, es la búsqueda de soluciones en un ambiente discreto, dinámico y de dimensión 2^n , donde n es el número de celdas que conforman el autómatas celular. Por ejemplo, si se decide buscar formas que estén contenidas en una retícula de 5 X 5 celdas, tendremos 33'554.432 posibles candidatos, solución al problema.

Por este motivo, se implementa un algoritmo genético para efectuar la exploración del espacio mediante búsqueda en paralelo. El flujograma de la implementación se observa en la figura 3.

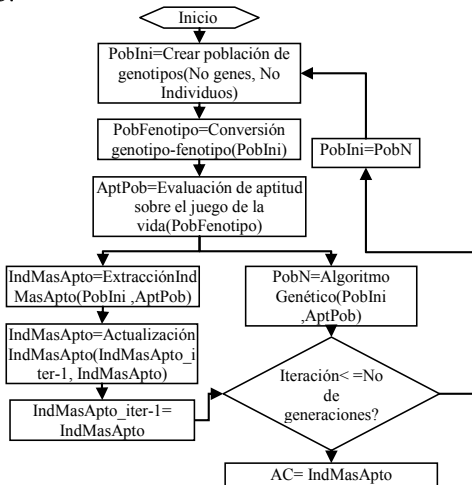


Figura 3. Diagrama de flujo de la aplicación: Generador de Autómatas Celulares.

Este algoritmo implementado en lenguaje G, permite visualizar el funcionamiento de la aplicación como sigue. Inicialmente se genera una población aleatoria de individuos o genotipos, candidatos a ser posibles autómatas celulares de interés, AC con comportamiento cíclico y perdurable a través de la aplicación subsecuente de las leyes de Conway. A continuación se permite evolucionar la población, un número preestablecido de generaciones, para poder cubrir el mayor porcentaje posible del

espacio de búsqueda, sin explorarlo completamente, garantizando eficiencia computacional.

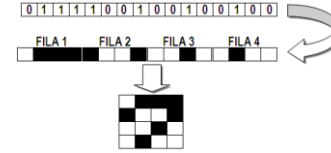


Figura 4. Conversión de una cadena binaria (Fenotipo) a su representación del Autómata celular (Genotipo).

La evolución de una generación consiste en La conversión de cada AC codificado en un genotipo o cadena binaria, a su correspondiente forma o fenotipo, como se muestra en la figura 4. A continuación el AC es colocado en el mundo virtual del juego de la vida, ver figura 5.

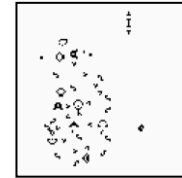


Figura 5. Mundo Virtual del juego de la vida. Implementación de una celda de memoria.

Una vez posicionado, se deja evolucionar para evaluar su comportamiento. El flujograma que describe la evaluación de aptitud sobre el juego de la vida se muestra en la figura 6. Así mismo, el índice de medición utilizado o función de aptitud para la valoración viene dado por la siguiente expresión.

$$\text{Aptitud} = \sum_{t=1}^P \sum_{i=1}^N \sum_{j=1}^M AC(i, j)^t + \sum_{t=1}^P \sum_{i=1}^N \sum_{j=1}^M \{AC(t) \oplus$$

$$AC(t-1)\} (i, j) \quad (1)$$

La primera parte de la ecuación expresa la componente estática del Autómata, es la sumatoria de todos los pixeles encendidos del autómatas a lo largo de toda su evolución, esta sumatoria crecerá si el AC sobrevive a la evolución. La segunda parte de la función de aptitud, representa el aspecto dinámico del autómatas, es la sumatoria de los pixeles que cambian de una iteración a otra consecutiva durante su evolución, si el AC es variante en el tiempo, y cambia de forma, este valor se incrementa, de no ser así, dicho valor se acota.

Finalmente, con la aptitud de cada individuo medida con el índice descrito en la ecuación 1, se opera el algoritmo genético para obtener nuevos Autómatas que sean posiblemente mejores que sus antecesores.

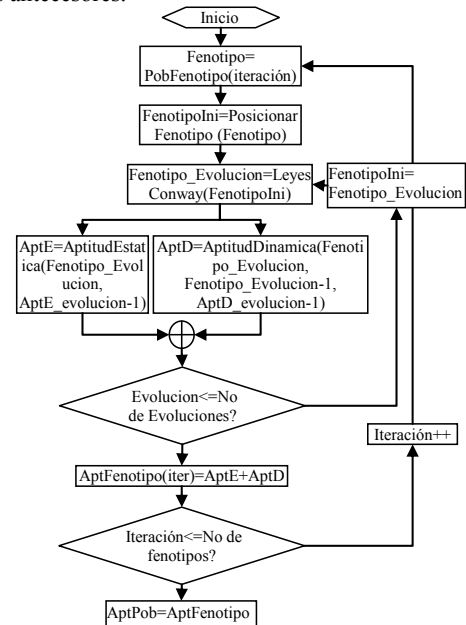


Figura 6. Algoritmo que implementa la evaluación de aptitud sobre el juego de la vida.

3. IMPLEMENTACIÓN DEL AG

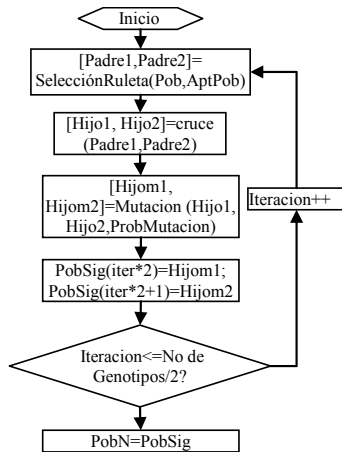


Figura 7. Implementación del algoritmo genético con selección por ruleta, cruce por máscara y mutación por umbral.

El esquema de la figura 7, muestra los procesos que sufren los genotipos que han sido cuantificados en el juego de la vida según su índice de aptitud. En primer lugar, de la población entrante al algoritmo, se seleccionan los mejores individuos según la técnica de selección llamada ruleta [10], la cual consiste en asignar números de una ruleta a los genotipos en proporción a su aptitud, relativa a la población donde existe el individuo, a mayor índice, mas números en la ruleta, a continuación se genera un número aleatorio (lanzamiento de la bolilla) y el individuo que posea el número ganador, será el escogido en el proceso. Hay que aclarar que siempre se eligen dos padres, ya que son necesarios en pares para efectuar el proceso de cruce. La reproducción de la nueva generación se realiza a través de una máscara de cruce, cadena del mismo tamaño del genotipo de un individuo, que determina que gen del padre 1 o 2 es transmitido a su descendiente. Generados una cantidad de hijos igual al tamaño de la población, termina el algoritmo con la posibilidad de cambiar el genotipo de cada nuevo individuo en uno de sus genes, en dependencia de si un número aleatorio asociado a cada individuo, es menor que un umbral pequeño o probabilidad de mutación. Razón por la cual, solo una pequeña porción de la nueva generación es afectada.

4. PRUEBAS Y RESULTADOS

A continuación se presentan los resultados de tres experimentos: primero, variación paramétrica para autómatas de 4 X 4 células, con mutación en una posición, segundo, igual que el caso anterior pero con mutación en múltiples posiciones aleatorias, y tercero, búsqueda de autómatas de 6 X 6 células.

Los dos primeros experimentos son corridas donde se prueba la incidencia de las variaciones en 4 parámetros del AG sobre los resultados de la búsqueda. Se varía en su orden: la probabilidad de mutación, el número de poblaciones, el número de cromosomas y el número de iteraciones de prueba, para el cálculo de la aptitud de los individuos. La probabilidad de mutación varía de $1/N$ a 1, en N pasos de magnitud $1/N$. El número de poblaciones, número de cromosomas y número de iteraciones de prueba varía de $100/N$ a 100, en N pasos de magnitud $100/N$. Los valores por defecto cuando no se realiza variación sobre los parámetros son: número de cromosomas 25, número de genes 16, número de poblaciones 50, probabilidad de mutación 0.3, número de iteraciones de prueba 50. Para estos experimentos el número de variaciones correspondiente a N es 50.

El tercer experimento es una corrida donde se busca

autómatas celulares de dimensión 6 X 6 células en un espacio de búsqueda de $2^{36}=68.719'476.736$ posibles autómatas, probando 1'000.000 de probables configuraciones que exhiban comportamiento cíclico en el tiempo.

Experimento Variación Paramétrica con Mutación en un Pixel.

En la figura 8, se muestra el valor de aptitud medido sobre los autómatas que registraron la aptitud más alta en la evolución de varias poblaciones para cada variación paramétrica: probabilidad de mutación, número de poblaciones, número de cromosomas y número de iteraciones de prueba.

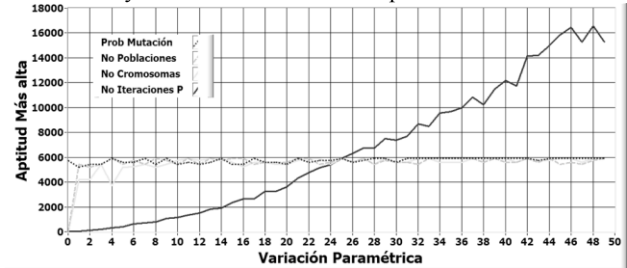


Figura 8. Gráfica de la aptitud más alta de un autómata para cada variación sobre cada parámetro del experimento.

De la figura 8, se puede observar que el valor de aptitud más alta registrado por algún autómata celular en todo el experimento es de 6000 unidades. La gráfica de número de iteraciones de prueba contra aptitud más alta muestra un comportamiento creciente como es de esperarse, ya que a mayor número de iteraciones en la vida de un autómata cíclico, mayor será su índice de aptitud según se explicó en la ecuación (1).

A continuación en la figura 9 se muestra las gráficas que consolidan los resultados de variación paramétrica en: probabilidad de mutación, número de poblaciones, número de cromosomas y número de iteraciones de prueba.

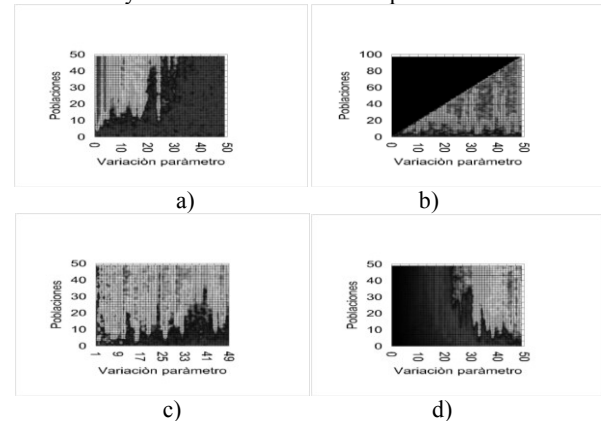


Figura 9. Gráficas de aptitud promedio vrs parámetro: a) probabilidad de mutación. b) número de poblaciones. c) número de cromosomas. y d) número de iteraciones de prueba.

Del análisis de las gráficas anteriores, se puede obtener los valores adecuados para los parámetros de configuración del algoritmo genético, de tal forma que se alcance una mayor eficiencia en la exploración del espacio de búsqueda y mejores resultados, es decir, autómatas celulares cíclicos de alto periodo. El rango de valores para cada parámetro se determina según sea alcanzado el valor máximo de aptitud promedio en el experimento donde se varía el parámetro en estudio, es decir, por ejemplo para el caso de variación en la probabilidad de mutación, figura 9a, en el rango de 0 a 0.1 se observa que toda corrida del AG después del 20% de las iteraciones, alcanza valores máximos de aptitud promedio.

Los valores así obtenidos para todos los parámetros variados son: probabilidad de mutación hasta 0.1 con fenotipos excelentes y hasta 0.5 con fenotipos buenos; número de

poblaciones mayor a 40.

Para el óptimo del parámetro número de cromosomas se observa la tendencia de la curva correspondiente en la figura 8, y se toma el valor mínimo para el cual se ha alcanzado estado estacionario, es decir, el valor de parámetro mínimo donde algún fenotipo en la evolución del AG alcanza el valor nominal de 6000 unidades, para los parámetros por defecto del experimento 1. Este comportamiento se halla en 20 cromosomas.

Por último, el numero de iteraciones de prueba adecuado para una corrida eficiente del AG es 50 iteraciones, ya que como se observa en la figura 8 y 9d, el costo computacional es aproximadamente lineal con el numero de iteraciones de prueba y por tanto si se toma como parámetro un valor bajo, se corre el riesgo que dos fenotipos, uno con ciclo mucho mayor a otro, por no poder evolucionar por encima de un número de iteraciones bajo, presenten igual valor de aptitud, o de otro lado, si el valor del parámetro se fija muy alto, el tiempo de corrida se incrementa inoficiosamente.

Los resultados de este experimento se pueden considerar como un punto de partida para la adecuada escogencia de los parámetros de configuración de motor de búsqueda basado en AG, y por tanto constituye una metodología de sintonización.

Los resultados de la búsqueda en el primer experimento se presentan a continuación: En general se observaron tres comportamientos terminales representativos asociados a múltiples condiciones iniciales.

El primer comportamiento final mostrado en la figura 10, se desencadena a partir del Autómata Celular 1, mostrado en la figura 11.

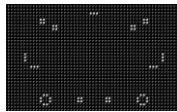


Figura 10. Estado final del autómata celular 1. Ciclo transiente de 170 formas.

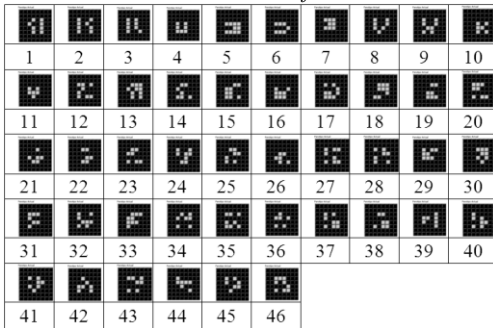


Figura 11. Estados iniciales del autómata 1, experimento 1.

Los estados 40-41 son similares a **B-heptomino**, según el compendio reportado en [8], el estado 7 es similar a **bullet heptomino**, los estados 11-26 son similares a **E-heptomino**, el estado 6 es similar a **hat** y el estado 4 es igual a **pi-heptomino**.

| No autómata | Estado inicial | Estado final | Observación |
|-------------|----------------|--------------|--|
| 2 | | | |
| 3 | | | Estado generador de un glider, se produce como resultado de la corrida del AG para un número de iteraciones muy bajo, 6 periodos de evolución. |

Figura 12. Autómatas experimento 1.

Experimento Variación paramétrica con mutación por mascara

Se repite el esquema del experimento anterior utilizando un operador de mutación diferente, mutación por mascara de cruce. Este cambio se realiza con el objetivo de lograr un mayor cubrimiento en el espacio de búsqueda en búsqueda de máximos mejores, autómatas cíclicos de periodos mayores.

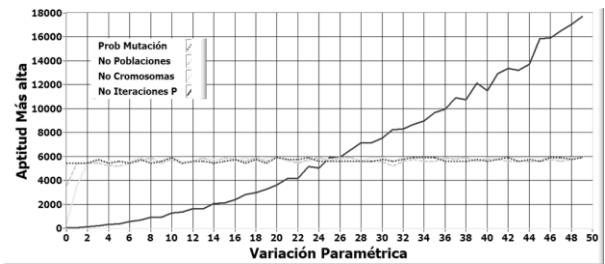


Figura 13. Gráfica de la aptitud más alta de un autómata para cada variación sobre cada parámetro del experimento.

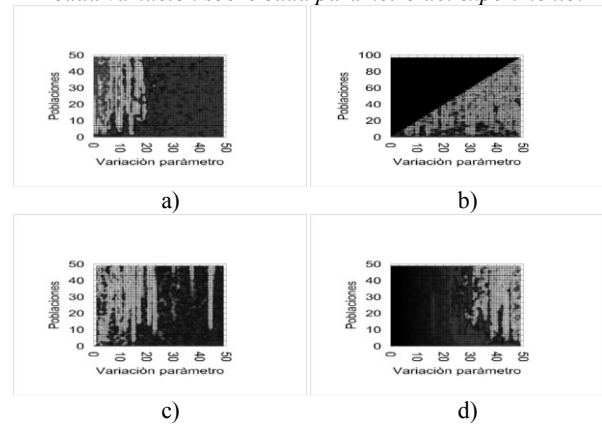


Figura 14. Gráficas de aptitud promedio vrs parámetro: a) probabilidad de mutación. b) número de poblaciones. c) número de cromosomas. y d) número de iteraciones de prueba.

El primer autómata generado en el segundo experimento ya había sido encontrado en el primero, sin embargo no se hallaron tantos estados iniciales como en el caso anterior, los estados iniciales encontrados haciendo referencia a la figura 11 fueron: 1, 2, 3, 4, 5, 6, 7, 11, 12, 13, 15, 16, 22, 23, 25, 27, 29, 30, 31, 36, 38, 39, 40, 41, 42, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59 y 60.

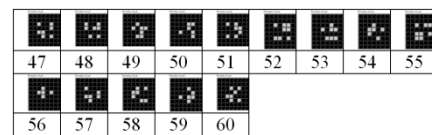


Figura 15. Otros estados iniciales asociados al autómata 1, producto del experimento 2.

Los autómatas 2 y 3 también fueron encontrados, más otras configuraciones adicionales se muestran en la figura 16.

| No autómata | Estados iniciales | Estado final | Ciclo | No iteraciones prueba |
|-------------|-------------------|--------------|--------|-----------------------|
| 4 | | | 67 | 50 |
| 5 | | | 7 | 50 |
| 6 | | | 309 | 50 |
| 7 | | | 10 | 4 |
| 8 | | | Slider | 6 |



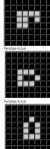







| | | | | |
|----|---|---|-----|-------|
| 9 |  |  | 111 | 8 |
| 10 |  |  | 16 | 10,12 |
| 11 |  |  | 81 | 16,20 |
| 12 |  |  | 62 | 18 |
| 13 |  |  | 51 | 22,26 |

Figura 16. *Autómatas celulares obtenidos del experimento 2.*

El número de iteraciones de prueba influncia enormemente el comportamiento de los autómatas hallados como se aprecia en la figura 16. En general a mayor sea el número de iteraciones de prueba, mas cíclicos y con mayor periodo de vida se desarrollarán los AC encontrados.

Experimento búsqueda de autómatas de 6 X 6 con mutación por máscara.

La relación de exploración del espacio de búsqueda es de aproximadamente 0,001%. La configuración de los parámetros para el experimento se determina a partir de los resultados de los experimentos de variación paramétrica. El número de cromosomas es 100, el número de genes es 36, el número de poblaciones es 10000, la probabilidad de mutación 0.3, el número de iteraciones de prueba 50.

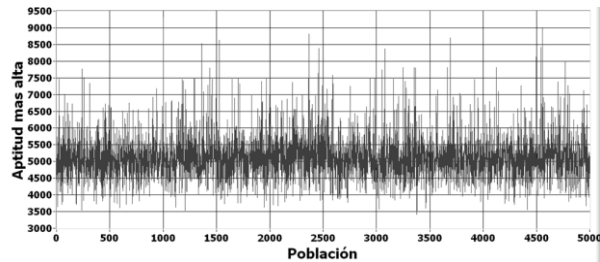


Figura 17. *Gráficas de aptitud más alta y promedio a lo largo de toda una corrida de 5000 generaciones en busca de autómatas de 6X6 células, para 50 iteraciones de prueba.*

El experimento encontró el estado final del autómata celular de los experimentos 1 y 2, dado que su espacio de búsqueda está incluido en el actual experimento. Los estados iniciales se presentan en la Figura 18.















































































































































































































| | | | | | | | | | |
|--|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  |  |  |  |  |  |  |  |  |  |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|  |  |  |  |  |  |  |  |  |  |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
|  |  |  |  |  |  |  |  |  |  |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
|  |  |  |  |  |  |  |  |  |  |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
|  |  |  |  |  |  |  |  |  |  |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
|  |  |  |  |  |  |  |  |  |  |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
|  |  |  |  |  |  |  |  |  |  |
| 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
|  |  |  |  |  |  |  |  |  |  |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
|  |  |  |  |  |  |  |  |  |  |
| 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |
|  |  |  |  |  |  |  |  |  |  |
| 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 |
|  |  |  |  |  |  |  |  |  |  |
| 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 |
|  |  |  |  |  |  |  |  |  |  |
| 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 |
|  |  |  |  |  |  |  |  |  |  |
| 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 |
|  |  |  |  |  |  |  |  |  |  |
| 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 |
|  |  |  |  |  |  |  |  |  |  |
| 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 |
|  |  |  |  |  |  |  |  |  |  |
| 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 |
|  |  |  |  |  |  |  |  |  |  |
| 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 180 |
|  |  |  |  |  |  |  |  |  |  |
| 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 |
|  |  |  |  |  |  |  |  |  |  |
| 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 |
|  |  |  |  |  |  | | | | |
| 201 | 202 | 203 | 204 | 205 | 206 | | | | |

Figura 18. *Estados iniciales del autómata celular 1, experimento 3.*

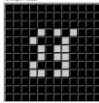
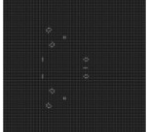
| No Autómata | Estado Inicial | Estado Final | Aptitud Dinámico |
|-------------|---|---|------------------|
| 2 |  |  | 27878 |

Figura 19. *Autómata celular 2, igual al encontrado en el experimento 1 y 2.*

Otras configuraciones cíclicas y simétricas encontradas en el experimento 3 se muestran en la Figura 20.

| No | Estado Inicial | Estado Final | Aptitud Dinámica | No | Estado Inicial | Estado Final | Aptitud Dinámica |
|----|----------------|--------------|------------------|----|----------------|--------------|------------------|
| 1 | | | 18872 | 13 | | | 4495 |
| 2 | | | 10040 | 14 | | | 14015 |
| 3 | | | 4435 | 15 | | | 56865 |
| 4 | | | NP | 16 | | | 4795 |
| 5 | | | 6611 | 17 | | | 29901 |
| 6 | | | 10047 | 18 | | | 67818 |
| 7 | | | 8321 | 19 | | | 21007 |
| 8 | | | 119272 | 20 | | | 3065 |
| 9 | | | 16346 | 21 | | | 4398 |
| 10 | | | 54191 | 22 | | | 15492 |
| 11 | | | 54618 | 23 | | | 17866 |
| 12 | | | 31359 | | | | |

Figura 20. Autómatas celulares cíclico-dinámicos producto de la corrida del experimento 3.

5. CONCLUSIONES

Encontrar Autómatas celulares clase III y IV, ha sido una tarea dispendiosa por tratarse de una búsqueda en un espacio computacionalmente ilimitado, donde la dinámica de evolución no se determina a través de ecuaciones físicas, sino por interacción local según una base de reglas.

El Generador de Autómatas Celulares GAC arrojó los resultados esperados, se encontraron 23 configuraciones de AC 2D que mostraron un comportamiento dinámico, simétrico y con periodos de vida significativos.

El proceso de sintonización de los parámetros del AG se abordó a través de un experimento de variación paramétrica, donde se evaluó la incidencia en el comportamiento de los AC obtenidos ante cambios en la probabilidad de mutación, el número de poblaciones, el número de cromosomas y el número de iteraciones de prueba.

El autómata celular 1 apareció recurrentemente como resultado de todas las corridas del GAC, por lo que se le puede considerar como un atractor fuerte en el juego de la vida.

Las aplicaciones de los AG clase III y IV van desde la

criptografía, tratada por wolfram, hasta el modelamiento de eventos naturales como el crecimiento de bacterias [7], pero tal vez su mayor interés se encuentra en la posibilidad de poder ejecutar algoritmos, si se implementa una máquina de estados a partir de los AC, clase IV, con los que se pueden construir las funciones lógicas básicas, y con las cuales posteriormente se puede escalar a registros, unidades de control y caminos de datos, que en suma constituyen la arquitectura de los computadores actuales.

Se propone como trabajo futuro explorar la posibilidad de buscar la mejor configuración de reglas por búsqueda a través de AG para obtener versiones alternativas al juego de la vida. Otro camino de exploración se encuentra en la implementación de reglas que rijan el comportamiento de los AC como relaciones difusas, las cuales se podrían hacer evolucionar para generar modelos más cercanos a los contextos donde se quieran utilizar los AC y metodologías de modelamiento de manera intuitiva.

REFERENCIAS

- [1] Saldaña, Winfer C. Tabares, William Emmanuel S. Yu. Parallel Implementations of Cellular Automata Algorithms on the AGILA High Performance Computing System. Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN.02). 2002 IEEE
- [2] Gardner, Martin. Mathematical Games. The fantastic combinations of John Conway's new solitaire game "life". Scientific American 223 (October 1970): 120-123.
- [3] Rendell, Paul. A Turing Machine In Conway's Game Life. Enero 2005.
- [4] Renard, Jean Philippe. Implementation of logical functions in the Game of Life.
- [5] Kenneth R. Crounse and Leon O. Chua. The CNN Universal Machine is as Universal as a Turing Machine. IEEE Transactions on Circuits and Systems-I Fundamental Theory and Applications, vol. 43, no. 4, April 1996.
- [6] Schiff, Joel L. Cellular Automata, A Discrete view of the World. A John Wiley & Sons, Inc., Publication. 2008.
- [7] J. Ermentrout, G. B. and Edelstein-Keshet, L., Theor. Cellular Automata approaches to biological modeling. Biol. 160 (1993), 97-133.
- [8] Silver, Stephen. Life Lexicon. http://www.argentum.freemove.co.uk/lex_home.htm. 2006.
- [9] Melanie M., Crutchfield J., Hrabar P. Physica D: Nonlinear Phenomena, Volume 75, Issues 1-3, 1 August 1994.
- [10] A. Lourdes, C. Cervigón. Algoritmos Evolutivos, un enfoque práctico. Alfaomega, 2009.