Trabajo Práctico de Investigación: Evolución de reglas de transición locales en autómatas celulares, utilizando Algoritmos genéticos.

Martin José Abente Lahaye[1] and José Luis Prous Rivas[1]

[1] Universidad Católica "Nuestra Señora de la Asunción", Asunción Paraguay
{tincho_02, joseprous}@hotmail.com

Abstract:

Aplicación de algoritmos genéticos en la evolución de reglas locales de transición para autómatas celares.

Keywords:

Algoritmo genético, Autómata Celular.

1. Introducción

Como parte del curso de informática II, se nos encomendó como trabajo de campo la investigación, experimentación y aplicación de algoritmos genéticos para evolucionar reglas de transición locales en autómatas celulares, aplicadas a tres problemas distintos los cuales se mencionan más adelante.

Se nos proporcionó distintos textos de referencia los cuales fueron de gran ayuda al momento de la conceptualización. [1] ,[2], [3], [5] La intensión de este texto consiste en mostrar los resultados de nuestra experiencia durante el proceso de aprendizaje, aplicación y experimentación, acompañado de conceptos teóricos básicos sobre los temas involucrados.

Hemos de presentar ejemplos, producto de nuestras experiencias en los distintos problemas presentados, "majority problem", "checkerboard problem" y "evolving bitmaps".

2. Autómatas Celulares

2.1. Breve reseña histórica

Los autómatas celulares o (CA) Cellular Automaton por su siglas en inglés, son producto de las investigaciones realizadas en los años 1940 por parte de dos investigadores del "Los Álamos nacional laboratory", Stanislaw Ulam y John von Neumann.

En años posteriores, 1970 para ser preciso un CA bidimensional llamado "Juego de la vida" también conocido como "Game of life" por su nombre original en la lengua inglesa se volvió ampliamente popular, el cual fue inventado por John Conway.

En 1983 Stephen Wolfram publicó una serie de textos en los cuales trataba de una poca conocida clase de CA, el cual llamó "Elementary Cellular Automata" o en español, Autómata celular elemental, el cual se caracterizaba por su comportamiento complejo en base a reglas simples. [6] ,[8]

2.2. Definición de Autómata celular

Un Autómata Celular, o CA por sus siglas en inglés Cellular Automata, es un modelo discreto estudiado en la teoría de la computación, matemática, biología teórica y modelado de micro estructuras.

Elementos:

- Un plano bidimensional o un espacio n-dimensional dividido en un número de subespacios homogéneos, conocidos como celdas. A todo esto se le denomina Teselación Homogénea.
- Cada celda puede estar en uno de un conjunto finito o numerable S de estados.
- Una Configuración C, la que consiste en asignarle un estado a cada celda del autómata.
- Una Vecindad definida para cada celda, la que consiste en un conjunto contíguo de celdas, indicando sus posiciones relativas respecto a la celda misma.
- Una Regla de Evolución, la cual define cómo debe cada celda cambiar de estado, dependiendo del estado inmediatamente anterior de su vecindad.
- Un Reloj Virtual de Cómputo conectado a cada celda del autómata, el cual generará "tics" o pulsos simultáneos a todas las celdas indicando que debe aplicarse la regla de evolución y de esta forma cada celda cambiará de estado. [7]

El tamaño de la vecindad depende del radio, el cual es un valor numérico finito mayor a cero, el tipo del vecindario y de la dimensión del CA.

Dos tipos de vecindarios comúnmente utilizados son:

Para un CA bidimensional:

1. Von Neumann
2. Moore

En un CA con d dimensiones $e_1, e_2, \ldots, e_d$ las celdas $a(i_1, i_2, \ldots, i_d)$ y $b(j_1, j_2, \ldots, j_d)$ son adyacentes en una vecindad von Neumann si

$\sum_{k=1}^{d} \min(|i_k - j_k|, e_k - |i_k - j_k|) = 1$

En una vecindad Moore son adyacentes si

$\min(|i_k - j_k|, e_k - |i_k - j_k|) \leq 1$ for $1 \leq k \leq d$

3. Problemas tratados

3.1. Majority Problem

Descripción informal Majority Problem, también conocido como Tarea de clasificación de densidad, fue resuelto para varias configuraciones por Gacs, Kurdyumov y Levin.

Usando reglas locales de transición, las células no pueden conocer el total de "unos" en el sistema. Por tanto se requiere que el sistema envié mensaje a través de un distancia lineal del tamaño del sistema. Este problema se caracteriza por mostrar el poder de los autómatas celulares. [8]

Enunciado del problema Dada una configuración inicial de un autómata celular de dos estados con i + j células en total, i de las cuales se encuentran en el estado "cero" y j de las cuales se encuentran en el estado "uno".

Sea p una solución correcta al problema de la votación, esta debe eventualmente asignar el estado "cero" a todas las demás células si $j/(i+j) < p$, y debe eventualmente asignar todas las células al estado "uno" si $j/(j+i) > p$. No se especifica para el caso donde $i/(i+j) = p$. [2]

Imposibilidad de un clasificador perfecto En 1994, Land y Belew, demostraron que ningún autómata celular bidimensional de radio r con densidad p puede resolver el problema de la votación para todas las configuraciones para un número suficientemente grande de grande de células, mayor a (4*r/p). [8]

3.2. Checkerboard problem

Definición informal Este problema demuestra como un CA puede generar un patrón global simple utilizando sólo reglas locales.

Enunciado del problema Encuentre una regla de transición la cual, dado un estado inicial del CA, itere sobre este hasta llegar al "patrón checkerboard", en un número de I iteraciones. [2]

Un "patrón checkerboard" puede ser definido como un estado el cual cada célula adyacente tiene un valor distinto.

3.3. Evolving bitmaps problem

Enunciado del problema Dado un estado inicial y un estado final definido, encontrar una regla que itere desde el estado inicial al estado final en menos de I iteraciones. [2]

4. Algoritmos genéticos

4.1. Breve reseña historia

La idea de la computación evolucionarla fue introducida en los años 1960 por I. Rechenberg en su trabajo “Estrategias de evolución” (Evolutionsstrategie). Su idea fue entonces desarrollada por otros investigadores.

Los Algoritmos genéticos o (GA) por sus siglas en inglés Genetic Algorithm, fueron inventados por John Holland y desarrollados por el mismo, sus estudiantes y colegas.

Lo cual lo lleva a escribir un libro “Adaptación en sistemas naturales y artificiales”, publicado en 1975. [4]

4.2. Definición de algoritmo genético

Se define “Espacio de búsqueda” como el espacio en el cual cada punto representa una solución factible a un problema dado.

El algoritmo inicia con un conjunto de soluciones, representado por cromosomas, el cual se conoce como población.

Las soluciones de una población son tomadas y se utilizan para formar una nueva población.

Esto se realiza con esperanza de que la nueva población sea "mejor" que la anterior, estas soluciones se miden mediante una función de peso o “fitness”, la cual determina cuales de las soluciones son mejores. [4]

Los algoritmos genéticos poseen distintos operadores.

1. Fitness: Mide la calidad del individuo.
2. Selección: El cual determina cuales individuos son aptos para el cruce.
3. Cruza: Genera un nuevo individuo a partir de uno o mas individuos de la población.
4. Mutación: Produce cambios esporádicos en el individuo.

5. Descripción del algoritmo genético implementado

5.1. Codificación del cromosoma

Cada cromosoma representa una función de transición total. Esto se realiza mediante una enumeración sobre el conjunto de valores posibles para el dominio de la función, codificados en una cadena de bits.

Sea:

- N el tamaño de la vecindad.
- P un función biyectiva tal que, P: $\{0,1\}^N$ ->ENTERO, el cual es el valor entero representado en la codificación binaria.
- B una cadena que pertenece a $\{0,1\}^{2\hat{}N}$
- Para una cadena cualquiera S que pertenece a $\{0,1\}^N$

- Se entiende la delta como, $B_{p(S)}$

Para ejemplificar, para una vecindad de tipo Moore, de radio igual a 1:

Sea N (Neighborhood Size) el tamaño de la vecindad, donde N = 9.

Por lo cual para presentar la función de transición total, se necesitan 2^N = 512 bits, la cual es B.

Ahora sea $0_1 1_2 0_3 1_4 ... 1_{511}$, ("cero" a las posiciones con índice par y "uno" a las posiciones con índices impares).

$B_{p(000000000)}$ ->0

$B_{p(000000001)}$ ->1

$B_{p(000000010)}$ ->0

$B_{p(000000011)}$ ->1

...

$B_{p(111111111)}$ ->0

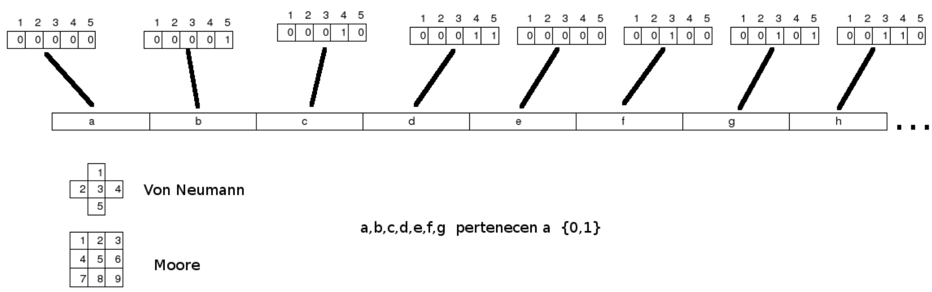

Figura 1. Un ejemplo gráfico de la codificación utilizada.

5.2. Fitness

En términos generales lo que se pretende es medir la distancia de la configuración actual con respecto a la configuración final.

Esto se calcula midiendo la distancia entre la configuración actual del CA y la final, mediante la comparación uno a uno de los estados entre las células.

5.3. Selección

Pese a que en las especificaciones del trabajo se definió que el método de selección se realizaría mediante "torneo", quisimos extender el mismo para poder implementar una forma simple elitismo, inspirada en la jerarquía de algunas especies animales, como los leones, lobos entre otros, la idea se basa en el hecho

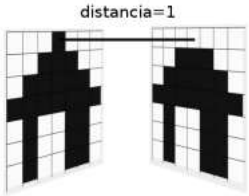

Figura 2. Ejemplo gráfico del proceso de comparación entre configuraciones

de que en toda población existe un macho alpha al cual se considera el "mas apto" de la población, por lo cual este tiene mayor probabilidad de reproducirse (cruzarse) y pasar a la siguiente generación.

Por lo cual, implementamos una versión híbrida entre selección por torneo y selección por el método el macho alpha (alpha-male selection), la cual consiste en:

1. Se elige el macho alpha, el cual pasara directamente a la siguiente generación.
2. Los demás individuos compiten en mediante el torneo.
3. Al momento del cruce, los ganadores del torneo tienen una cierta probabilidad de cruzarse con el macho alpha, en caso contrario se cruzan con otro ganador del torneo.

Observación: Si la probabilidad de que el macho alpha se cruce tiende a 0, este método de selección tiende a ser más parecido al método de selección por torneo convencional, con excepción de que el "mas apto" siempre pasa a la siguiente generación.

5.4. Cruce

El cruce se realiza mediante el método de cruza simple en un punto.

Por Ej., C1 {1,1,|1,0} y C2 {0,0,|1,1} resulta en C3 {1,1,1,1} o en C4 {0,0,1,0}

La elección del punto de cruce y del hijo se realizan en forma aleatoria.

5.5. Mutación

Se define la probabilidad de que mute un bit, luego de cada cruce, un número esperado de bits se alteran aleatoriamente.

Por ej, Sea C1 {0,0,0,1} un cromosoma cualquiera, con una probabilidad de mutación por bit igual a $\frac{1}{4}$, se espera que en promedio un bit sea alterado, por lo cual C1 podría resultar C1 {1,0,0,1}.

5.6. Comentarios finales sobre la implementación

La implementación esta divida en cuatro módulos.

Parametrizador Es básicamente un "parser", el cual carga desde un archivo los parámetros, los cuales son:

- N "Número de filas."
- M "Número de columnas."
- R "Radio de la vecindad."
- NHT "Tipo de vecindario."
- $CAITERATIONS$ "Número máximo de transiciones para el CA."
- $PSIZE$ "El tamaño de la población del GA."
- $GAITERATIONS$ "Número de generaciones del GA."
- $MUTEPROB$ "Probabilidad de que mute un bit."
- $ISIZE$ "Número de estados iniciales con el cual se medirá el fitness."
- $CROSSPROB$ "Probabilidad con la cual los ganadores del torneo se cruzan con el macho alpha."
- T "Es parámetro se utiliza para generalizar el algoritmo del fitness, con T = 1, el fitness se calcula por cada transición, en caso contrario solo se calcula en la ultima transición."
- Q "Es el tamaño del torneo."

Este modulo es utilizado por el algoritmo y por el Generador de datos de entrada.

El algoritmo genético De propósito general recibe un conjunto de pares {configuración inicial, configuración final} con el cual calcula el fitness, Al terminar escribe la mejor función de transición en un archivo.

El generador de datos de entrada para el problema especifico Para mantener el algoritmo genético lo más general posible, se utiliza un conjunto módulos intermedios los cuales generan los datos de entradas para cada problema específico.

Por Ej., para el majority problem, genera pares de configuraciones iniciales al azar, con su correspondiente configuración final, según el número de unos en la configuración inicial.

La interfaz gráfica Es la interfaz que interactúa con el usuario, todos los demás módulos funcionan de manera transparente.

Guía de uso resumida (para iniciar una búsqueda de una solución):

- Majority problem: Simplemente configurar parámetros, y presionar RUN.
- Checkerboard problem: Configurar los parámetros, y presionar RUN.
- Evolving Bitmaps: Configurar los parámetros, presionar RESET para limpiar la grilla, haciendo clicks con el botón izquierdo del mouse dibujar sobre la grilla, y presionar RUN.

Una vez terminado el proceso realizado por los distintos módulos, se devuelve el control a la interfaz gráfica.

Para salvar una regla especifica, en el menú seleccionar FILE y SAVE DELTA.

Para ver paso a paso, en el caso de majority y checkerboard inicialmente presionar RANDOM para generar una configuración al azar.

En el caso de bitmap, presionar RESET, dibujar el punto en el medio, el cual es el estado inicial. Para realizar una transición presionar STEP.

6. Resultados de la experimentación

Se realizaron distintas pruebas para cada problema, y se obtuvieron resultados altamente satisfactorios. A continuación mostraremos los parámetros utilizados, las funciones transiciones para los mismos.

6.1. Majority problem

Parámetros:

- Filas: 13
- Columnas: 13
- Radio: 1
- Tipo de vecindario: Von Neumann
- Numero de transiciones: 320
- Numero de generaciones: 100
- Tamaño de la población: 100
- Coeficiente de mutación por bit: 10, la probabilidad de mutación por cada bit es 1/10.
- Número de pares de configuraciones (inicial, final): 10
- Coeficiente de cruce para el macho alpha: 2, la probabilidad de que un ganador se se cruce con el macho alpha es de $\frac{1}{2}$.
- Tamaño del torneo: 2

Observación: Para poder probar las deltas, se debe primero, configurar los parámetros, luego, seleccionando en el menú de la interfaz, FILE ->LOAD DELTA y después para realizar una transición presionar STEP.

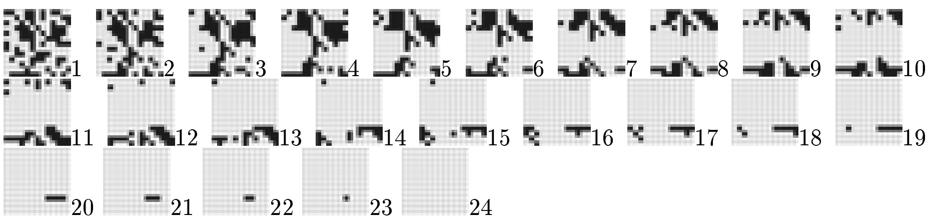

Figura 3. Se muestra paso a paso los cambios en el CA utilizando la regla que se encuentra en el archivo majority.txt

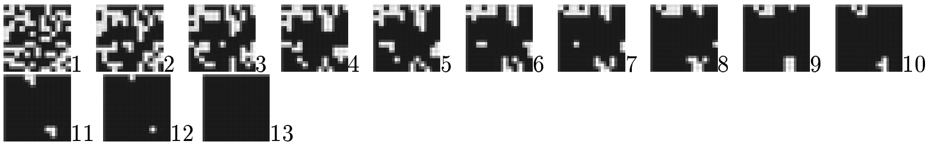

Figura 4. Otro ejemplo utilizando la misma delta

6.2. Checkerboard problem

Parámetros:

- Filas: 8
- Columnas: 8
- Radio: 1
- Tipo de vecindario: Von Neumann,
- Numero de transiciones: 100
- Numero de generaciones: 100
- Tamaño de la población: 100
- Coeficiente de mutación por bit: 10, la probabilidad de mutación por cada bit es 1/10.
- Numero de pares de configuraciones (inicial, final): 10
- Coeficiente de cruce para el macho alpha: 2, la probabilidad de que un ganador se se cruce con el macho alpha es de $\frac{1}{2}$.
- Tamaño del torneo: 2

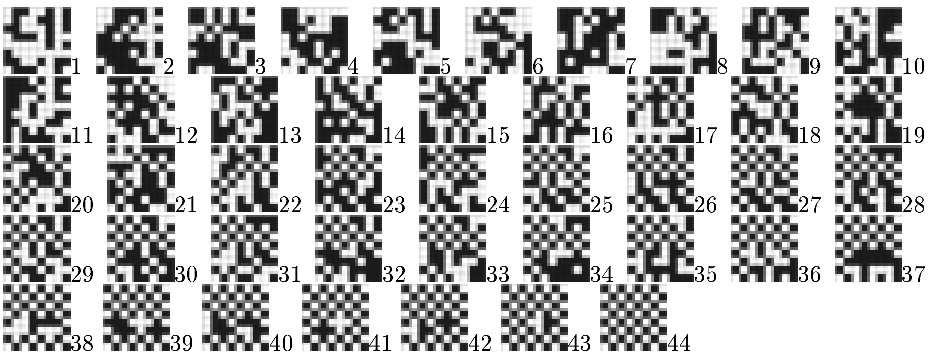

Figura 5. Ejemplo del comportamiento utilizando la delta desde el archivo checkerboard.txt

6.3. Evolving Bitmap

Parámetros:

- Filas: 7
- Columnas: 7
- Radio: 1
- Tipo de vecindario: Moore,
- Numero de transiciones: 10
- Numero de generaciones: 100
- Tamaño de la población: 10000
- Coeficiente de mutación por bit: 10, la probabilidad de mutación por cada bit es 1/10.
- Número de pares de configuraciones (inicial, final): 10
- Coeficiente de cruce para el macho alpha: 2, la probabilidad de que un ganador se se cruce con el macho alpha es de $\frac{1}{2}$.
- Tamaño del torneo: 2

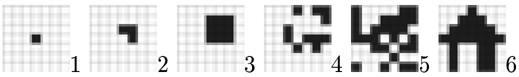

Figura 6. Utilizando la delta desde el archivo casita2.txt

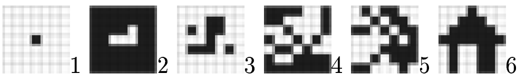

Figura 7. Utilizando la delta desde el archivo casita1.txt

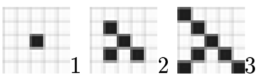

Figura 8. Utilizando la delta desde el archivo lambda.txt, modificando los parámetros, rows = 5, columns = 5

7. Conclusiones

Como resultado del proceso de estudio, implementación y experimentación, pudimos comprobar que existe una utilidad real en la aplicación de algoritmos genéticos para evolucionar reglas de transición en autómatas celulares. Los resultados experimentales dieron cierta perspectiva del campo real de aplicación

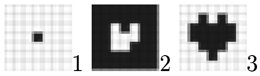

Figura 9. Utilizando la delta desde el archivo corazon.txt

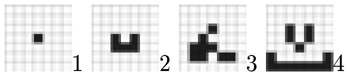

Figura 10. Utilizando la delta desde el archivo feliz.txt

de este enfoque para la resolución de distintos problemas. Por otro lado, pudimos observar ciertos aspectos puntuales sobre los factores que influyen de mayor manera a la evolución de mejores soluciones.

Por Ej., para cualquiera de los problemas, el tamaño de la población es el parámetro que mayormente determina la calidad de la solución, en promedio fue marcando esta tendencia con respecto a la calidad de la solución, cuya explicación se comenta en [2], al referirse a la distribución de las soluciones en la población inicial. Existen otros aspectos que caben resaltar para cada problema especifico, para el majority problem, encontramos reglas suficientemente buenas sin la necesidad de utilizar un conjunto de pares {estado inicial, estado final} muy grande, lo mismo se aplica para el checkerboard problem, en cual tampoco se necesito un número grande transiciones del CA, promediando 50.

Por otro lado, en el evolving bitmap problem, el número de transiciones, para un CA de dimensiones 7x7, no se necesitó más de 10 transiciones para llegar a resultados aceptables.

Con respecto al uso de los distintos tipos de vecindades sugiere el hecho de que para problemas como majority y checkerboard el uso de vencidades de tipo Von Neumann resultan en mejores soluciones, por el contrario en evolving bitmaps, la vecindad tipo Moore resultó en mejores soluciones en la gran mayoría de los casos. Lastimosamente no se pudo realizar suficientes pruebas como para llegar a resultados concluyentes sobre la implementación del método de selección híbrido torneo-machoalpha, en el cual mantuvimos la probabilidad de cruce con el macho alpha en 50 % en casi todas las pruebas, pero se pudo observar que por el simple hecho de haber implementando una forma simple de elitismo hubo una mejora significativa en la calidad de los resultados.

Tampoco consideramos de mucha relevancia los demás parámetros como el coeficiente de mutación, el cual se mantuvo alrededor del 10, donde la probabilidad de que un bit al azar también fue de 1/10, así tampoco con respecto al tamaño del torneo, el cual se mantuvo en 2 durante las distintas pruebas. Dichos aspectos inconclusos podrían analizarse con mayor detenimiento en un futuro periodo de pruebas, pero por el momento estos superan las intensiones propuestas para este trabajo.

Como comentario final, queremos nuevamente resaltar lo satisfactorio de los resultados los cuales superaron nuestras expectativas iniciales.

Referencias

1. D. Andre, F. Bennet, y J. Koza. Discovery by Genetic Programming of a Cellular Automata Rule that is Better than any known Rule for the Majority Classification Problem.
2. R. Breukelaar, y T. Bäck. Using a Genetic Algorithm to Evolve Behavior in Multi Dimensional Cellular Automata. En: Proceedings of GECCO'05. Junio 25-29, 2005.
3. M. Dorigo, V. Maniezzo, y A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. IEEE
4. Introduction to Genetic Algorithms, Online manual, http://www.obitko.com/tutorials/genetic-algorithms/index.php Transactions on Systems, Man, and Cybernetics 26(1), 1-13 (1996).
5. J. Kennedy, y R. Eberhart. Particle Swarm Optimization. En: Proceedings of the IEEE Conference on Neural Networks, volume 4, páginas 1942-1948. 1995.
6. Moshe Sipper, Moshe Sipper's web site, http://www.moshesipper.com
7. Una Introducción a los Autómatas Celulares, http://yupana.autonoma.edu.co/publicaciones/yupana/005/autocelular/Automatas.html
8. Wikipedia, La enciclopedia libre.http://www.wikipedia.org