Proyecto Autómatas Celulares

Mauricio Castro
Andrés Lostaunau

June 2020

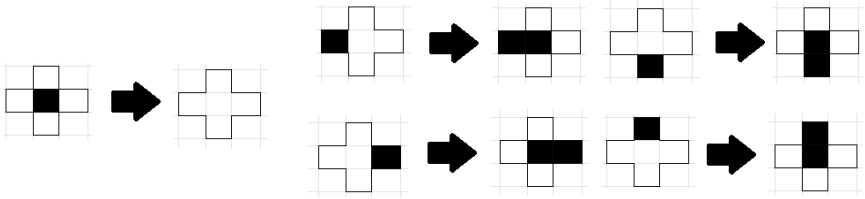
1 Introducción

1.1 Autómatas celulares

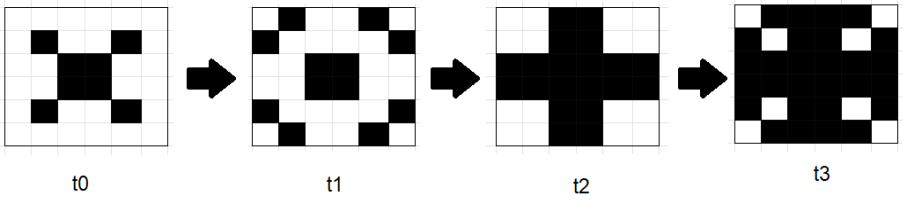
Los autómatas celulares se pueden definir como un 3-tupla (V, v0, f), donde:

- V = Es el grupo de los estados celulares posibles.
- v0 = Es un estado perteneciente a V, llamado "estado inactivo"
- f = La función de transición de una n-tupla de elementos de V a V.

Es muy conveniente presentar el espacio celular como un plano cartesiano de células (no obligatorio), donde las células que influencian en una transición a otra se pueden denominar como células vecinas [2]. Si tenemos una célula v0 que puede a cambiar de estado 0 a 1 dependiendo de las células v1, v2, v3; podemos denominar a las últimas células como vecinas de v0. Las transiciones en un autómata celular siempre se van a dar en simultaneo, lo que significa que la iteración t+1 va a usar los valores de la iteración t. Ejemplo:



Tenemos aquí los siguientes estados de transición. Podemos decribirlos como si tenemos una célula en 1 y el resto de valores adyancentes son 0, la célula cambiará a 0; y si tenemos una célula en 0 y tiene solo un valor adyacente 1, entonces este cambiará a 1. En el resto de casos la célula se mantendrá. Si tenemos a t0 como estado inicial el resultado será el siguiente:



El autómata iniciará en t0, donde ocho células van a ser inicializadas con 1 de manera arbitraria. Al pasar al estado t1, notaremos que las cuatro células "exteriores" van a cambiar otras dos células cada una, mientra que las centrales se mantienen. Una vez pasando al estado t2 cada una de las celulas previas se apagarán generando una nueva célula, y las células centrales se expandirán, hasta llegar al estado t3 el cual sería nuestro estado final.

1.2 Problema

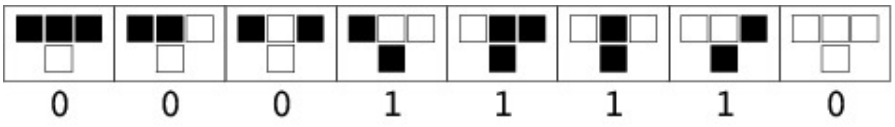
En este proyecto, buscamos plantear métodos y aplicaciones para las autómatas celulares (ACC). En primer lugar, existen maneras bastante sencillas de entender y aplicar ACC. Explicaremos a detalle las reglas necesarias para la creación de estas. En segundo lugar, mostraremos aplicaciones de ACC en la actualidad. En este caso nos enfocaremos en la aplicacion de ACC en simulaciones de incendios de bosques de mediana y grande amplitud.

1.3 Estado del Arte

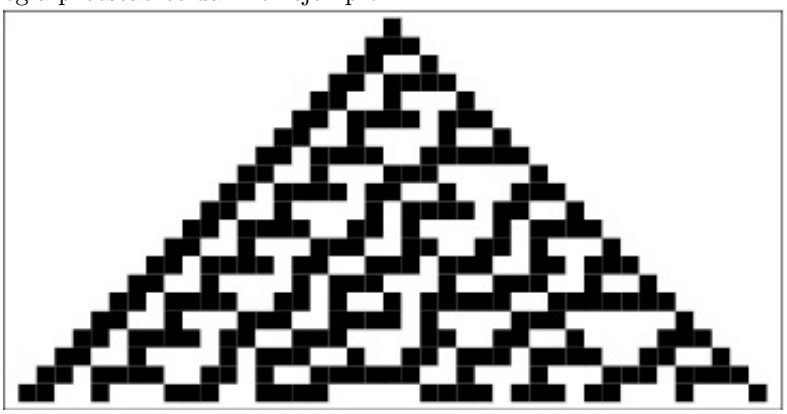
En las lecturas investigada se pueden obtener las siguientes conclusiones acerca de las autómatas celulares. En primer lugar, en todos los modelos de ACC, por más diferentes que luzcan, siguen ciertas pautas para ser consideradas ACC's. En segundo lugar, por más sencillas que parezcan, las ACC's pueden llegar a un nivel de complejidad bastante alto. En tercer lugar, en todos los modelos, los vecinos de cada célula son quienes definen el valor que tomará esta en las siguientes generaciones.

2 Pseudocódigo

Debido a que un autómata celular no es un algoritmo no tiene un pseudocodigo preciso, pero si se puede modelar uno a partir de un autómata celular específico. En este caso vamos a describir uno de los autómatas celulares más representados, el *Elementary Cellular Automaton*. Este autómata es caracterizado por sus numerosas reglas a las que se somete un mismo estado inicial. Son exactamente $2^8 = 256$ reglas las cuales representan el output para las 8 combinaciones posibles dentro de un grupo de 3 bits. Por ejemplo, la regla 30 vendría a tener la siguiente forma:



El estado inicial siempre va a ser una cadena de 0s con un 1 en el elemento central. Entonces el siguiente estado va a depender de la cadena previa siguiendo la regla preestablecida. Por ejemplo:



Este sería la cuadrícula generada por el autómata con la regla 30.Viendo este comportamiento podríamos diseñar un algoritmo que haga lo que hace este autómata de la siguiente manera:

Algorithm 1 Elementary Cellular Automaton

```
1:  procedure REGLA N
2:      bool matriz[d][w]
3:      d ← profundidad de la cuadrícula
4:      w ← ancho de la cuadrícula
5:      i ← estado actual = 0
6:      NextState:
7:      if i < d − 1 then
8:          n ← 1
9:          HorizontalIt:
10:         if n + 1! = w then
11:             matriz[i+1][n] ← rule-n(matriz[i][n-1],matriz[i][n],matriz[i][n+1])
12:             n ← n + 1
13:             goto HorizontalIt
14:         i ← i + 1
15:         goto NextState
```

3 Problema: Modelado de incendios forestales usando autómatas celulares

Los incendios forestales tiende a exparsirse dependiendo de la cercanía de otros árboles, la velocidad del viento y su dirección. La propagación de estos se comporta de manera constante dependiendo del valor de las variables mencionadas. Este sistema se puede modelar como un autómata celular de múltiples reglas posibles [1].

Analizaremos un número determinado de casos usando un programa que nos ayude a recopilar y comparar cada matriz dependientemente de la cantidad de arboles por matriz. Usaremos matrizes de 20x20 y variables como posición inicial del fuego, dirección de del viento (A favor o en contra del fuego) y la fuerza del viento (0, 1 o 2).

El proceso por el cual se generan los árboles de la matriz sigue los siguientes pasos. Primero, áboles serán dispersados en la matriz con un algoritmo. Este se encarga de elegir posiciones aleatorias en la misma. Si tiene un álbol, no hace nada, si está vacío, planta un árbol y así sicesivamente hasta plantarlos a todos. Segundo, se genera fuego en una posición aleatoria x, el cual a partir del la primera iteración se exparsirá en la dirección predefinida. Tercero, A partir de la fuerza del viento se le permitirá al fuego analizar más vecinos hacia la dirección del viento. Finalmente se procederá a generar fuego hasta que el rango de vecinos en la dirección determinada ya no posea más árboles a quemar.

3.1 Simulaciones

3.1.1 Velocidad: 6km/h; dirección: 315 grados

Para esta situación la regla tendría esta forma:

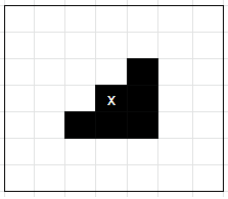


Figure 1: Regla

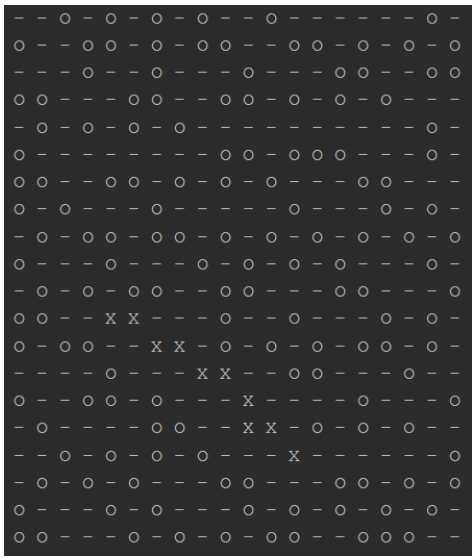


Figure 2: Porcentaje de destrucción: 9 por ciento
Densidad poblacional = 40 por ciento
Fuego inicial: X = 4, Y = 11
Iteraciones = 9

Figure 3: Porcentaje de destrucción: 19 por ciento
Densidad poblacional = 75 por ciento
Fuego inicial: X = 12, Y = 6
Iteraciones = 19

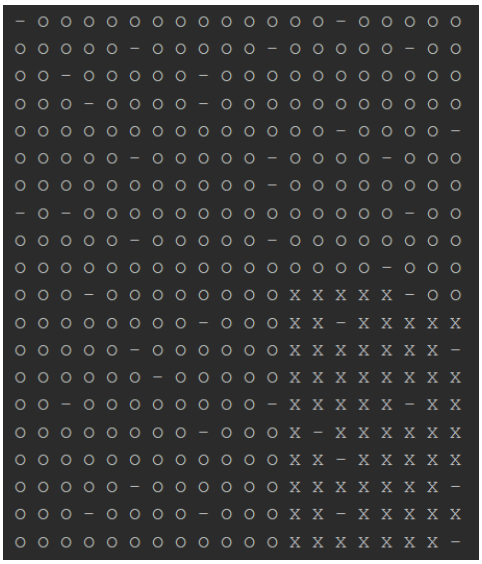


Figure 4: Porcentaje de destrucción: 19 por ciento
Densidad poblacional = 90 por ciento
Fuego inicial: X = 12, Y = 10
Iteraciones = 10

Como se puede apreciar, a medida que la densidad poblacional de árboles sea mayor la chances de que el porcentaje de destrucción sea mayor y además de que el proceso sea más rapido.

3.1.2 Velocidad: 12 km/h; dirección:180 grados

Para esta situación la regla tendría esta forma:

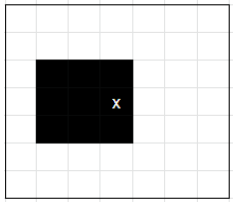


Figure 5: Regla

Figure 6: Porcentaje de destrucción: 18 por ciento
Densidad poblacional = 40 por ciento
Fuego inicial: X = 9, Y = 8
Iteraciones = 7

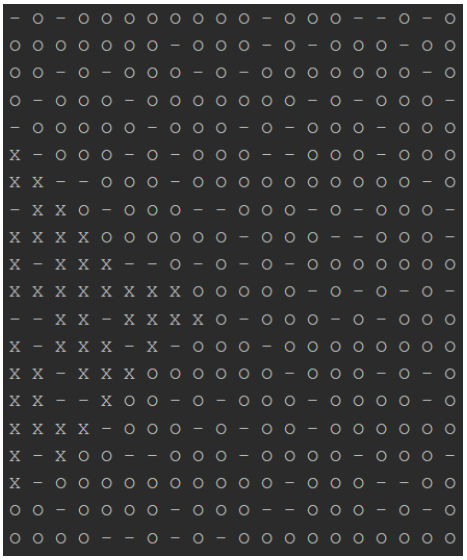


Figure 7: Porcentaje de destrucción: 15 por ciento
Densidad poblacional = 75 por ciento
Fuego inicial: X = 8, Y = 11
Iteraciones = 7

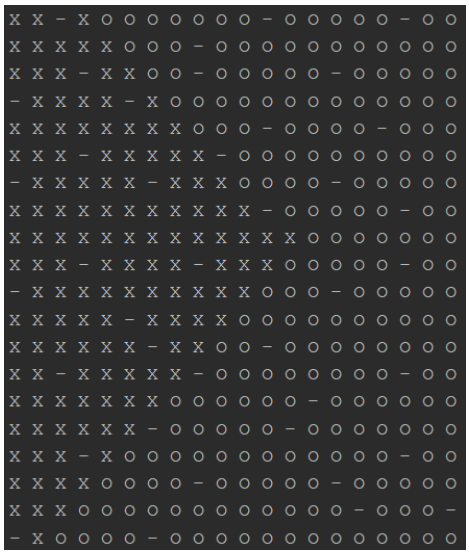


Figure 8: Porcentaje de destrucción: 37 por ciento
Densidad poblacional = 90 por ciento
Fuego inicial: X = 12, Y = 8
Iteraciones = 12

4 Recolección de datos

A continuación, se presentarán casos con diferente densidad poblacional. Esto con el objetivo de presentar comparaciones acerca del promedio de porcentaje de destrucción con 10, 30, 50, 70 y 90 porciento de densidad,

4.1 10% de densidad poblacional

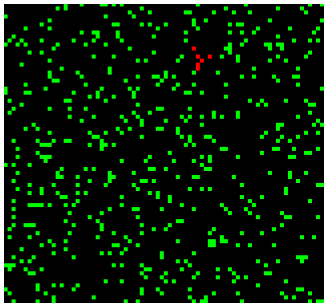


Figure 9: Resultado más alto para un 10% de densidad poblacional.
Iteraciones: 5, % Destrucción: 1%

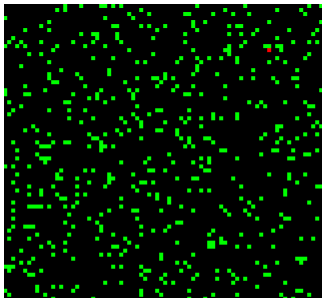


Figure 10: Resultado más bajo para un 10% de densidad poblacional.
Iteraciones: 1, % Destrucción: 0%

En este caso el porcendaje de destrucción no sobrepasa, en su mayoría, el 1%. Tambien el número de iteraciones realziadas son menores a 3. Esto se debe a la gran separación entre los árboles del bosque. El viento muchas veces no es suficientemente fuerte para llevar el fuego más allá de 2 bloques.

4.2 30% de densidad poblacional

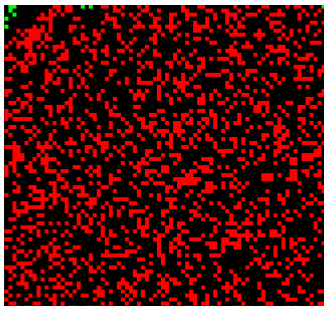


Figure 11: Resultado más alto para un 30% de densidad poblacional.
Iteraciones: 87, % Destrucción: 99%

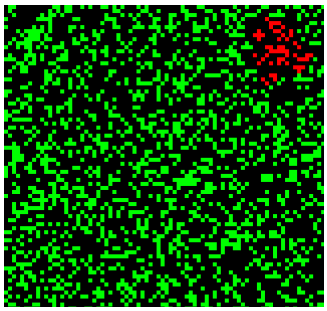


Figure 12: Resultado más bajo para un 30% de densidad poblacional.
Iteraciones: 11, % Destrucción: 2%

Ahora, con un DP del 30%, podemos observar que hay una mayor posibilidad de que el fuego se propague por todo el bosque. Pero el fuego aún puede ser detenido si el viento se coloca justo en su contra, tal y como vemos en la figura 11. Se a notado en con este DP la dependencia sobre la fuerza y dirección del viento llega a ser completamente aleatoria. No podemos asegurar qué porcentaje de destrucción se obtiene con más posibilidad.

El número promedio de iteraciones y porcentaje de destrucción, en 10 casos, fue de 22 y 14.4%, respectivamente.

4.3 50% de densidad poblacional

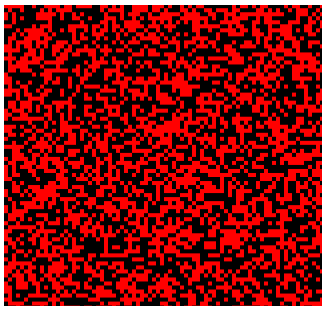


Figure 13: Resultado más alto para un 50% de densidad poblacional.
Iteraciones: 99, % Destrucción: 100%

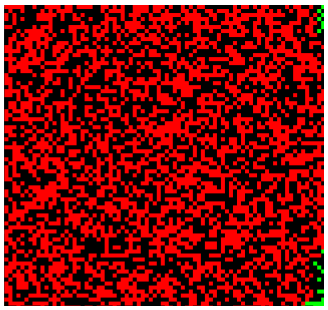


Figure 14: Resultado más bajo para un 50% de densidad poblacional.
Iteraciones: 59, % Destrucción: 98%

En este caso, podemos ver que,con un 50% de DP, es casi seguro por lo menos obtener un 98% de destrucción. Pero, además, se realizan una cantidad enorme de iteraciones. Esto es por el aumento de posibilidades de direcciones a las que el fuego puede propagarse y la actidud aleatoria del viento.

El número promedio de iteraciones y porcentaje de destrucción, en 10 casos, fue de 77 y 98.8%, respectivamente.

4.4 70% de densidad poblacional

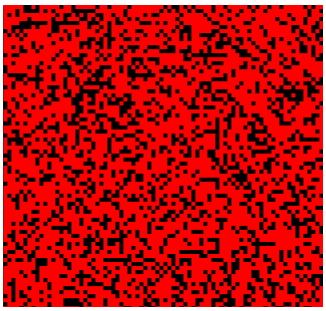


Figure 15: Resultado más alto para un 70% de densidad poblacional.
Iteraciones: 50, % Destrucción: 100%

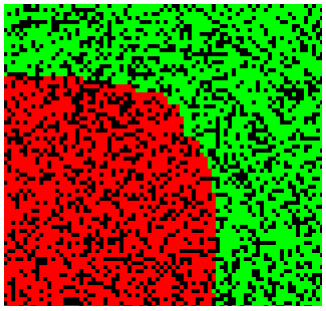


Figure 16: Resultado más bajo para un 70% de densidad poblacional.
Iteraciones: 52, % Destrucción: 47%

Analizando las simulaciones con 70% DP, el número de iteraciones requeridas para alcanzar el 100% de destrucción se reduce drásticamente. Esto se debe a la fácil propagación delfuego por el abundante número de árboles cercanos. Pero el fuego se vuelve un poco más controlable por el viento.

El número promedio de iteraciones y porcentaje de destrucción, en 10 casos, fue de 59 y 89.3%, respectivamente.

4.5 90% de densidad poblacional

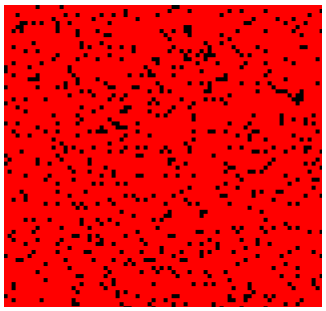


Figure 17: Resultado más alto para un 90% de densidad poblacional.
Iteraciones: 65, % Destrucción: 100%

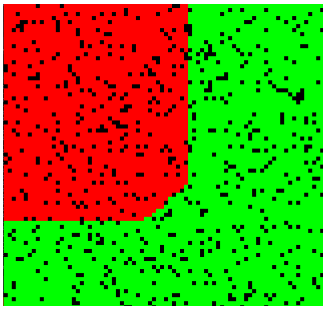


Figure 18: Resultado más bajo para un 90% de densidad poblacional.
Iteraciones: 35, % Destrucción: 37%

Este caso no es muy diferente al anterior, pero se puede observar que el número de iteraciones aumento para alcanzar un 100% y se redujo para alcanzar el mínimo porcentaje de destrucción. Ahora, el abundante número de árboles hace enteramente dependiente al movimiento del fuego con la dirección del viento. Por lo tanto, a mayor número de árboles cercanos, más fácil es el control de incendios solo mediante el viento.

El número promedio de iteraciones y porcentaje de destrucción, en 10 casos, fue de 53 y 70%, respectivamente.

References

[1] M Peredo and R Ramallo. Aplicación de autómatas celulares a simulación básica de incendios forestales (tesis de maestría).

[2] Stephen Wolfram. *Cellular automata and complexity: collected papers*. CRC Press, 2018.