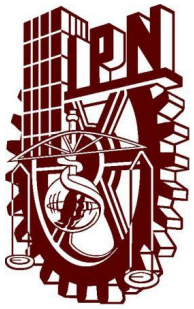

INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Laboratorio de Inteligencia Artificial
Laboratorio de Simulación y Modelado

Aprendizaje de fenómenos con representación de estados en 2D

TESIS

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A:

Ing. Héctor Daniel Moreno Leyva

Directores de tesis:
Dr. Salvador Godoy Calderón
M. C. Germán Téllez Castillo

Mexico, CDMX Octubre 2020

Resumen

Para la modelación de sistemas físicos, químicos y biológicos, los autómatas celulares han sido una herramienta importante por su capacidad de modelar sistemas dinámicos de manera discreta en el espacio y tiempo; y a su vez también ha sido una herramienta cuyo uso es limitado por la complejidad que existe en la generación de las reglas de evolución que modela el sistema completamente. Se ha demostrado que los algoritmos de aprendizaje simbólicos y sub-simbólicos han resultado de gran utilidad para generar modelos interpretables a partir de un conjunto de datos, es por esto que en el presente trabajo se busca estudiar cómo es posible facilitar el uso de estas herramientas de modelado realizando un propuesta de algoritmo de aprendizaje automático que nos permita generar reglas a partir de un conjunto de datos representativos obtenidos de cierto fenómeno, sin perder la semántica que se encuentra embebida en los datos de los que se está aprendiendo.

Abstract

The modelation of physical, biological and chemistry systems, the cellular automata models have been an important tool for their capacity to model dynamical systems in a discrete way in space and time; also it is a tool whose use is limited by the complexity that exists to select the right evolve transition rules to model the system completely. It has been demonstrated that symbolic and sub-symbolic algorithms of learning have been of great utility to generate interpretative models from a dataset, is because of this that in order to facilitate the use of this tools, we are proposing a model that let us generate a set of rules from a representative dataset of a certain phenomena, without loosing the semantics of what does the data represent.

Thanks

I deeply appreciate the support received from CIC-IPN ...

To my family ...

Índice general

Capítulo 1

Introducción

Desde hace ya varios años los seres humanos se han beneficiado de la modelación y simulación de sistemas dinámicos para entender la naturaleza. En la actualidad existe una amplia gama de métodos de simulación y modelado. Siendo una de ellas los autómatas celulares, los cuales son una herramienta de modelación discreta en el espacio y tiempo.

Los autómatas celulares se usan actualmente para modelar sistemas dinámicos de diversos tipos, para llevarlo a cabo los investigadores hacen la definición de las reglas de evolución. Esto lo realizan basándose en gran medida de su experiencia y empleando diversos técnicas de análisis para obtener información que ayude a generar estas reglas, teniendo asi que realizar múltiples experimentos hasta encontrar el conjunto de reglas que satisfacen las condiciones para las cuales el modelo fue creado.

Hoy en día este problema ya está siendo investigado desde distintos aspectos, sin embargo muchos de ellos se centran en crear métodos específicos para el área de estudio en el que se está realizando la modelación del fenómeno, por ejemplo, la búsqueda de reglas para modelar la conversión de un área rural a un área urbana, o la búsqueda de reglas para modelar patrones de bioconvección en algas unicelulares; y otros trabajos de propósito más general que emplean algoritmos evolutivos para la generación de reglas donde el comportamiento específico no es conocido, esto es, llegar de un estado inicial a un estado final sin tomar en cuenta lo que suceda en los estados intermedios. Este último trabajo es el más apegado a lo que se busca en esta tesis, sin embargo, un punto importante en el que se diferencian los dos trabajos es que a nosotros nos interesa tomar en cuenta la información del fenómeno de principio a fin, para crear un conjunto de reglas que repliquen este fenómeno

lo más preciso posible, esto es, pasando por cada uno los estados de los que tenemos información.

1.1. Definiciones

1.1.1. Autómata Celular

Los autómatas celulares son sistemas dinámicos y discretos en espacio, estado y tiempo. Propuestos por los matemáticos John von Neumann y Stanislaw Ulam como un sistema discreto para crear un modelo reduccionista de la auto-replicación, creando así en 1950 el primer autómata celular como método para calcular el movimiento de un líquido.
En 1970 el autómata celular "Game of Life" inventado por John Conway (8), que consiste de dos dimensiones y dos estados, se volvió ampliamente conocido, sobre todo en la comunidad computacional.
En 1981 Stephen Wolfram empezó a trabajar independientemente en autómatas celulares, en 1985 conjeturo que la regla 110 de un autómata celular elemental era equivalente a una máquina de Turing, cosa que demostró Matthew Cook en el 2004 (4).
Como podemos ver a través de los años con el incremento de poder de cómputo y el trabajo que se ha ido realizando alrededor de los autómatas celulares, ha ido incrementado el interés por utilizarlos para modelar sistemas naturales y artificiales.
Los autómatas celulares pueden ser empleados como una alternativa a las ecuaciones diferenciales para modelar sistemas físicos (21), y como un modelo de cómputo paralelo y distribuido (9).

El uso exitoso de los autómatas celulares en distintos campos, tales como:

- Simulación de tránsito (19)
- Dinámica de fluidos (13)
- Formación de patrones (20; 2)
- Conexiones con los lenguajes formales (15; 5)
- Modelación y simulación de diversos sistemas físicos (22; 12) y biológicos (7)

es una de las razones por las cuales este trabajo de tesis es de importancia.

La mayoría de los autómatas celulares poseen las siguientes cinco características(10):

- **Una lattice de células discretas**: El sistema consiste de una estructura llamada lattice la cual puede ser de dimensionalidad d, donde $d \epsilon \aleph \cup \{0\}$
- **Homogeneidad**: Todas las células son equivalentes.
- **Estados discretos**: Cada célula toma un estado de un conjunto finito de estados discretos.
- **Interacciones locales**: Cada célula interactúa solo con las células que están en su vecindad.
- **Sistemas dinámicos discretos**: En cada paso de tiempo discreto, cada célula actualiza su estado actual de acuerdo a una función o regla de evolución que toma como entrada los estados de las células vecinas y da como salida un estado del conjunto de estados.

Definición 1.1.1 Un autómata celular es una 5-tupla (L, D, S, H, f) donde:

- L es una matriz de dimensión d
- $D \epsilon \aleph \cup \{0\}$ y es la dimensión del autómata.
- S es un conjunto finito de elementos llamados estados y es denotado por:
  $S = \{s_k : k\epsilon\{0, \ldots, |S| - 1\}\}$, donde $|S|$ es la cardinalidad del conjunto de estados S
- H es un subconjunto finito de Z^d llamado vecindad y es denotado por $\{v_j : x_{1,j}, \ldots, x_{d,j} : j\epsilon 1, \ldots, |H|\}$, donde los elementos v_j son llamados vectores vecindad.
- f es una función de $S^{|H|}$ en S, llamada la función de evolución o regla.

Definición 1.1.2 En un autómata celular se dice que posee fronteras nulas, si el vecino de la izquierda (o de la derecha) de la célula del extremo izquierdo (o derecho) se considera siempre como cero.

Definición 1.1.3 En un autómata celular se dice que posee fronteras periódicas, si los extremos derecho e izquierdo son adyacente el uno del otro.

1.1.2. Aprendizaje Automático

El área de aprendizaje automático es un sub-area de la Inteligencia Artificial (AI), que se basa en distintos enfoques para mejorar en el desempeño de un programa computacional sobre una tarea utilizando la experiencia que se tiene sobre la tarea. El aprendizaje puede ser de distintos tipos, los principales siendo: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje reforzado.

Aprendizaje supervisado: Consiste en el uso de un conjunto de entrenamiento para mejorar el desempeño en el programa computacional utilizando como entrada pares (x, y) y la tarea es encontrar una funcion f que al ingresar x produzca y como salida.

Aprendizaje no supervisado: Consiste en el uso de un conjunto de entrenamiento donde se conocen valores de entrada x, pero estos datos no están etiquetados. La tarea entonces es encontrar una función f tal, que al ingresar x como entrada produzca como salida y de tal forma que y sea igual para todas las entradas x que compartan cierta medida de similitud.

Aprendizaje reforzado: En este tipo de aprendizaje no tenemos conocimiento previo sobre la tarea, si no que se va obteniendo experiencia sobre la tarea como se va realizando, y se va ajustando nuestro programa computacional con respecto a la métrica de desempeño obtenida sobre la tarea en cada iteración.

1.1.3. Algoritmos genéticos

Los algoritmos genéticos son algoritmos de búsqueda basados en las mecánicas de la selección natural. Combinan el supervivencia del mas apto con

el intercambio de información para formar un algoritmo de búsqueda con el novedoso instinto de búsqueda humana.

1.2. Planteamiento del problema

Dada una secuencia de estados bidimensionales, encontrar un conjunto de reglas que al ingresarlas a un autómata celular puedan reproducir estos estados.

1.3. Objetivos

1.3.1. Objetivo general

Diseñar un nuevo (simbólico o sub-simbólico) algoritmo de aprendizaje para aprender reglas que gobiernan un fenómeno para las cuales nosotros solo tenemos como entrada rejillas estados en 2 dimensiones.

1.3.2. Objetivos específicos

- Revisar y seleccionar algunos algoritmos de aprendizaje de reglas para el problema planteado.
- Proponer un nuevo algoritmo de aprendizaje.
- Diseñar procedimientos específicos para la simplificación de locales a globales de los conjuntos de reglas.
- Construir el modelo de autómata celular que reproduce el fenómeno cuando es alimentado con las reglas aprendidas.
- Usar errores de aprendizaje y generalización para evaluar los resultados de cada experimento.

1.4. Justificación

La búsqueda de reglas para que un autómata celular pueda reproducir un fenómeno es compleja debido a que el espacio de búsqueda crece exponencialmente tomando en cuenta la cardinalidad del vecindario y la cardinalidad del espacio de estados. Es por esto que se requiere una búsqueda que pueda mejorar el desempeño en este espacio de búsqueda tomando en consideración información adicional.

1.4.1. Beneficios esperados

Esta tesis pretende generar una propuesta de algoritmo genético multipoblacional distribuido el cual pueda solucionar el problema planteado.

1.4.2. Alcances y límites

En este trabajo se pretende encontrar un algoritmo que pueda aprender un fenómeno del cual solo se cuenta con un conjunto de estados 2-dimensional las reglas de evolución para que al ingresarlas a un autómata celular pueda replicar el fenómeno completo, para esto utilizamos los estados obtenidos de la evolución de autómatas celulares, ya que obtener estados de la forma requerida de fenómenos reales era una tarea ardua, la cual no se pudo realizar con los recursos de tiempo que se tenían disponibles.

Capítulo 2

Marco teórico

2.1. Algoritmos genéticos

Los algoritmos genéticos fueron desarrollados por John Holland, sus colegas y estudiantes en la universidad de Michigan. Los objetivos de esta investigación eran los siguientes:

1. Abstraer y explicar rigurosamente los procesos adaptativos de los sistemas naturales.

2. Diseñar sistemas artificiales de software que retuvieran los mecanismos importantes de los sistemas naturales.

El tema central de investigación en los algoritmos genéticos ha sido la robustez, el balance entre la eficiencia y la eficacia necesaria para sobrevivir en distintos ambientes. Los algoritmos genéticos han probado teórica y experimentalmente que proveen una búsqueda robusta en espacios complejos.

La implementación de un algoritmo genético consta de la definición de:

- La representación de las soluciones.
- Función de aptitud.
- Los operadores de selección, mutación y cruza.
- El tamaño de la población.
- Los valores de probabilidad con la que se aplican los operadores genéticos.

El siguiente pseudocódigo ilustra el funcionamiento de un algoritmo genético simple.

```
1 Inicializar la población con candidatos aleatorios de la solución;
2 Evaluar cada candidato;
3 mientras La condicion de paro no se satisfaga hacer
4 |   Paso 1: Seleccionar a los padres;
5 |   Paso 2: Recombinar los pares de padres;
6 |   Paso 3: Mutar a los hijos;
7 |   Paso 4: Evaluar a los nuevos candidatos;
8 |   Paso 5: Seleccionar a los individuos para la siguiente generación;
9 fin
```

Algoritmo 1: Pseudocódigo de un algoritmo genético simple.

2.1.1. Representación

Este paso consiste en hacer un vinculo entre el contexto original del problema y espacio de solución del problema donde la evolución se lleva acabo. Esto se lleva acabo decidiendo como las soluciones deben ser especificadas y guardadas para que puedan ser manipuladas por una computadora. Los objetos formando posibles soluciones en el contexto original son llamadas fenotipos, mientras que su codificación en el espacio de solución son llamados genotipos.

2.1.2. Función de aptitud

El rol de la función de aptitud es representar los requerimientos a los que se debe adaptar la población. Desde el punto de la perspectiva de resolución de problemas, representa la tarea a ser resuelta en el contexto evolutivo. Y técnicamente es una función o procedimiento que le asigna una medida cualitativa a los genotipos.

2.1.3. Población

La población forma la unidad básica de evolución, los individuos son objetos estáticos que no cambian o se adaptan, es la población la que lo

hace. La diversidad de la población que indica las diferentes soluciones que hay presentes.

2.1.4. Mecanismo de selección de padres

Este mecanismo tiene como rol distinguir de entre los individuos basándose en su calidad, para permitir que los mejores individuos sean los padres para la siguiente generación. Este mecanismo junto con el mecanismo de selección de sobrevivientes son los responsables por ingresar mejoras en la calidad de las soluciones.

2.1.5. Recombinación

Como el nombre indica este operador, es un operador de variacional que recombina la información de dos padres en uno o mas genotipos descendientes. la recombinación es una operación estocástica esto es, como se eligen los genes de los padres y como los combinas depende del azar. El principio detrás de la recombinación es sencillo - al aparear a dos individuos con diferentes pero deseables atributos, podemos obtener una descendencia que combine ambas características.

2.1.6. Mutación

Este operador variacional unario es aplicado a un genotipo, que resulta en mutante ligeramente modificado. El operador de mutación siempre es un operador aleatorio. Este operador es el causante de ingresar ”sangre fresca” a la población.

2.1.7. Mecanismo de selección de sobrevivientes

Este mecanismo toma la decisión de que individuos serán los elegidos para pasar a la siguiente población, esta decisión es frecuentemente basada en sus valores de aptitud. Al contrario del mecanismo de selección de padres que es aleatorio, este mecanismo es frecuentemente determinista.

2.2. GA-Nuggets

El GA-Nuggets fue diseñado para la modelación de dependencias en dos versiones, la primera una población centralizada de individuos donde diferentes individuos pueden representar reglas de predicción para diferentes atributos objetivo, La segunda versión mantiene una población distribuida de muchas sub-poblaciones donde cada una de ellas evoluciona de manera independiente pero con la posibilidad de que algunos individuos puedan migrar entre las sub-poblaciones, en esta versión cada población tiene asociado un atributo objetivo a predecir, es por esto que todos los individuos de cada sub-población representan reglas para predecir el mismo atributo objetivo.

2.2.1. Representación de los individuos

Cada individuo representa una regla de predicción candidata de la forma IF Ant THEN $Cons$, donde Ant es el antecedente de la regla y $Cons$ es el consecuente. El antecedente Ant consiste en una conjunción de condiciones, donde cada condición es un par atributo valor de la forma $A_i = V_{ij}$, donde A_i es el i-ésimo atributo y V_{ij} es el j-ésimo valor del atributo A_i. El algoritmo solo maneja valores categóricos, por lo que es necesario discretizar los valores. El consecuente $Cons$ consiste en un solo par atributo-valor de la forma $G_k = V_{kl}$, donde G_k es el k-ésimo atributo objetivo y V_{kl} es el l-ésimo valor del atributo G_k. El conjunto de atributos objetivos son seleccionados por el usuario entre todos los atributos del conjunto de datos y el resto de los atributos se utiliza como atributos predictivos.
Un individuo es codificado en una cadena de longitud fija conteniendo z genes, donde z es el numero de atributos considerando los atributos predictivos y objetivo. De los valores de los atributos codificados en el genoma solo un subconjunto de los valores de los atributos sera decodificado, para realizar esto se utiliza un valor que no se encuentre entre los posibles valores de los atributos por ejemplo -1, para indicar que ese gen no sera decodificado en un antecedente para la regla, gracias a esto a pesar de que la longitud de la cadena sea fija, podemos tener un antecedente de longitud variable.
Una vez que se genero el antecedente de la regla, el algoritmo selecciona el atributo objetivo que incremente la aptitud del individuo.

2.2.2. Función de aptitud

La función de aptitud consiste de dos partes, la primera mide el grado de interés de la regla, y la segunda mide su certeza de predicción. El grado de interés de la regla en dos términos, el primero se refiere al antecedente y el segundo al consecuente de la regla. El grado de interés de la regla se mide con una métrica de teoría de la información. Donde el grado de interés para el antecedente de la regla esta dada por:

$$AntInt = 1 - \left(\frac{\sum_{i=1}^{n} InfoGain(A_i)/n}{log_2(|dom(G_k)|)}, \right) \tag{2.1}$$

donde n es el número de atributos que ocurren en el antecedente de la regla y $|dom(G_k)|$ es la cardinalidad del dominio (es decir, el número de valores posibles) del atributo de destino G_k que se produce en el consecuente. El término log se incluye en la fórmula 2.1 para normalizar el valor de $AntInt$, de modo que esta medida tenga un valor entre 0 y 1. El $InfoGain$ viene dado por:

$$InfoGain(A_i) = Info(G_k) - Info(G_k|\, A_i), \tag{2.2}$$

donde,

$$Info(G_i) = -\sum_{i=1}^{mk} \left(Pr(V_{kl}) log_2(Pr(V_{kl})) \right), \tag{2.3}$$

y

$$Info(G_k|\, A_i) = \sum_{i=1}^{n_i} \left(Pr(V_{ij}) \left(-\sum_{j=1}^{mk} Pr\left(V_{kl}|V_{ij}\right) log_2\left(Pr\left(V_{kl}|V_{ij}\right)\right) \right) \right). \tag{2.4}$$

Donde m_k es el numero de posibles valores que puede tomar el atributo objetivo G_k, n_i es el numero de posibles valores para el atributo A_i, $Pr(X)$ denota la probabilidad de X y $Pr(X|Y)$ denota la probabilidad de X dado Y.

La métrica que proporciona AntInt puede ser justificada de la siguiente forma, en general dado un atributo predictor A_i cuya ganancia de información sea alta con respecto a G_k, nos lleva a considerar que A_i es un buen predictor de G_k cuando se considera individualmente, ignorando sus interacciones con otros atributos predictivos. Por esto, si se considera que el usuario ya conoce los atributos que son buenos predictores del atributo objetivo G_k por lo que estos atributos podrían no parecerle interesantes al usuario. Sin embargo, los atributos que tienen baja ganancia de información y aparecen en el antecedente de la regla, el usuario los podría considerar como irrelevantes por si solos, pero cuando interaccionan con los otros atributos podrían generar que estos se vuelvan relevantes y esto los haría interesantes para el usuario.
La computación del grado de interés del consecuente de la regla esta basada en la idea de que mientras mas raro sea el valor de un atributo objetivo con respecto a un valor común del atributo, mas interesante sera para el usuario. En otras palabras, mientras mas grande sea la frecuencia relativa en el conjunto de entrenamiento del valor predicho por el consecuente, menos interesante sera para el usuario.
Mas precisamente la regla para predecir el grado de interés del consecuente de la regla es:

$$ConsInt = (1 - Pr(G_{kl}))^{1/\beta} \tag{2.5}$$

Donde $Pr(G_{kl})$ es la frecuencia relativa del valor del atributo objetivo G_{kl}, β es un parámetro especificado por el usuario.
La segunda parte de la función de aptitud mide la exactitud predictiva de la regla, y esta dada por:

$$PredAcc = \frac{|A\&C| - 1/2}{|A|} \tag{2.6}$$

Donde $|A\&C|$ es el numero de ejemplos que satisfacen tanto el antecedente como el consecuente de la regla, y $|A|$ es el numero de ejemplos que satisfacen solo el antecedente. El termino $1/2$ es para penalizar las reglas que cubren pocos ejemplos de entrenamiento.
Finalmente, la función de aptitud seria:

$$Fitness = \frac{w_1(AntInt + ConsInt)/2 + w_2 PredAcc}{w_1 + w_2}, \tag{2.7}$$

Donde w_1 y w_2 son pesos definidos por el usuario.

2.2.3. Método de selección y operadores genéticos

GA-Nuggets utiliza el bien conocido selección por torneo con un tamaño de torneo de 2, y utiliza una cruza uniforme extendida con un procedimiento de reparación. En la cruza uniforme existe una probabilidad de aplicar cruza a dos individuos y otra probabilidad para intercambiar cada gen. Después de que se realizo la cruza el algoritmo verifica si se genero un individuo invalido. Si es así, se inicia un procedimiento de reparación para generar individuos validos. El operador de mutación transforma de forma aleatoria el valor de un atributo en otro dentro del mismo dominio del atributo.
Ademas de la cruza y la mutación, se agregan otros dos operadores: insertar-condición y eliminar-condición, que controlan el tamaño de las reglas que están siendo evolucionadas, de la siguiente forma: mientras mas mayor sea el numero de condiciones en el antecedente de la regla actual, menor sera la probabilidad de aplicar el operador insertar-condición y este operador no se aplicara si el antecedente ya tiene el máximo numero de condiciones especificado por el usuario. En cambio, la probabilidad de aplicar el operador eliminar-condición sera mayor mientras menor sea el numero de condiciones en el antecedente de la regla, y este operador no se aplicara si el antecedente solo tiene una condición.

2.3. Enfoque OCAT (One Clause At a Time)

El enfoque Üna Clausula A la Vezü OCAT por sus siglas en ingles (One Clause At a Time) es un enfoque que trata de resolver dos debilidades con las técnicas de aprendizaje bayesianas y conexionistas como lo son las redes neuronales, estas debilidades son las siguientes:

1. La forma en que estos métodos funcionan y producen recomendaciones pueden no ser atractivos por los expertos en el dominio.

2. El conjunto de entrenamiento puede no contener suficientes ejemplos como para que garanticen resultados estadisticamente significantes. Como resultado estos métodos pueden no ser fidedignos para aplicaciones reales.

Este enfoque esta basado en conceptos de lógica matemática y optimización discreta. El enfoque es de naturaleza avara en el sentido de que busca

aceptar a todos los ejemplos positivos y rechazar tantos ejemplos negativos como pueda, y así sucesivamente para cada una de las clausulas que genera, hasta que ya no tenga mas ejemplos negativos para rechazar.
El enfoque OCAT busca generar un conjunto de clausulas que pueden estar en forma normal conjuntiva (FNC) o forma normal disyuntiva (FND). Es conocido (16) que cualquier función Booleana puede ser transformada en forma FNC o FND.
Algunas asumpciones que este enfoque toma son las siguientes:

- Se tienen observaciones que describen el comportamiento del sistema a aprender.
- Que un conjunto de n atributos de estas observaciones describen totalmente el sistema.
- Cada una de las observaciones pertenece a una y solo una de las K clases.
- Que las observaciones están libres de ruido.
- También asume que la clase a la que pertenece la observación es la correcta.

El siguiente pseudocódigo ejemplifica el funcionamiento general del enfoque OCAT.

```
1 i = 0; C = ∅;{Inicializaciónes};
2 mientras E⁻ ≠ ∅ hacer
3 |   Paso 1: i ← i + 1;
4 |   Paso 2: Encontrar una clausula c_i que acepte todos los miembros
  |     de E⁺ mientras rechaza tantos miembros sea posible de E⁻;
5 |   Paso 3: SeaE⁻(c_i) el conjunto de miembros de E⁻ que son
  |     rechazados por c_i;
6 |   Paso 4: Sea C ← C ∧ c_i;
7 |   Paso 5: Sea E⁻ ← E⁻ − E⁻(c_i);
8 fin
```

Algoritmo 2: Pseudocódigo de enfoque OCAT para generar clausulas en forma normal conjuntiva.

2.3.1. Heuristica RA1

El enfoque heuristico, denominado RA1 por Randomized Algorithm 1, fue propuesto en (6). El cual es un algoritmo que como se muestra en el algoritmo 3 selecciona deforma aleatoria dentro los mejores atributos candidatos, lo cual evita quedarse estancado en un mínimo local.

mientras $E^- \neq \emptyset$ **hacer**
  $C = \emptyset$ (inicialización);
  mientras $E^+ \neq \emptyset$ **hacer**
    Paso 1: Clasificar en orden descendente todos los atributos $a_i \in a$ (donde a_i es A_i o $\neg A_i$) de acuerdo a su valor $POS(a_i)/NEG(a_i)$. Si $NEG(a_i) = 0$, entonces $POS(a_i)/NEG(_i) = 1000$ (i.e., un valor arbitrariamente alto);
    Paso 2: Formar una lista de candidatos de los atributos que tienen los l valores más altos $POS(a_i)/NEG(a_i)$;
    Paso 3: Elegir al azar un atributo a_k de la lista de candidatos;
    Paso 4: Sea $C \leftarrow C \vee a_k$ el conjunto de atributos en la cláusula actual;
    Paso 5: Sea $E^+(a_k)$ el conjunto de miembros de E^+ aceptados cuando a_k se incluye en la cláusula FNC actual;
    Paso 6: Sea $E+ \leftarrow E^+ - E^+(a_k)$;
    Paso 7: Sea $a \leftarrow a - a_k$;
    Paso 8: Calcular los nuevos valores $POS(a_i)$ para todos $a_i \in a$;
  fin
  Paso 9: Sea $E^-(C)$ el conjunto de miembros de E^- que son rechazados por C;
  Paso 10: Sea $E^- \leftarrow E^- - E^-(C)$;
  Paso 11: Reiniciar E^+;
fin
elegir el sistema Booleano final que tenga el menor número de clausulas, de los sistemas ITRS anteriores.

Algoritmo 3: La heurística RA1 (6)

2.4. Algoritmo Quine–McCluskey

Este algoritmo es método utilizado para la simplificación de funciones Booleanas y fue desarrollado por Willard V.Quine 1955 y extendido Edward J. McCluskey en 1956. Este método involucra dos pasos:

1. Encontrar los implicantes primos de la función.
2. Usar esos implicantes primos para encontrar los implicantes primos esenciales de la función, así como los otros implicantes primos necesarios para realizar la cobertura de la función.

El siguiente pseudocódigo ejemplifica este algoritmo.

```
1  inicio
2      Organizar los términos mínimos en orden ascendente;
3      Formar como máximo n + 1 grupos en función del número de
         unos presentes en sus representaciones binarias[1];
4      inicio
5          Obtener los implicantes primos:
6          mientras no se tengan todos los implicantes primos hacer
7              comparar los términos mínimos presentes en grupos
                 sucesivos
               si existe un cambio en la posición de un solo bit
               entonces
8                  Tomar ese par de dos términos mínimos;
9                  Colocar el símbolo "_" en la posición del bit de cambio;
10                 Mantener el resto de los bits tal como están;
11             fin
12         fin
13     fin
```

[1]Para n variables Booleanas en una función Booleana, o para n bits en su equivalente binario de términos mínimos.

```
15
16  inicio
17      Formular la tabla de primos implicantes, que consiste en un
         conjunto de filas y columnas:
         inicio
18           Los implicantes primos se colocan en sentido de las filas;
19           Los términos mínimos se colocan en sentido de las
              columnas;
20           Colocar un "1" en las celdas correspondientes a los
              términos mínimos que se cubren en cada implicante
              primo;
21       fin
22  fin
23  inicio
24      Encontrar los implicantes primos esenciales:
         para todo término mínimo de la función Booleana hacer
25           observar cada columna de la tabla de primos implicantes
              si el término mínimo está cubierto solo por un implicante
              primo entonces
26               este término es un implicante esencial y formará parte
                  de la función Booleana simplificada
27           fin
28           eliminar la fila de cada implicante primo esencial y las
              columnas correspondientes a los términos mínimos que
              están cubiertos en ese implicante primo esencial
29       fin
30  fin
31 fin
```

Algoritmo 4: Método Tabular Quine-McCluskey: para simplificar funciones Booleanas.

Capítulo 3

Estado del Arte

De los trabajos que existen en la actualidad, podemos mencionar tres que resultan de especial interés, cuyas metodologías se encuentran las siguientes:

- Búsqueda de reglas locales para un autómata celular mediante el análisis estadístico de las observaciones de un fenómeno.
- Configuración de reglas locales para un autómata celular mediante el empleo de algoritmos evolutivos.
- Búsqueda de reglas locales para un autómata celular mediante el empleo de algoritmos evolutivos para satisfacer un patrón especifico.

Estos son algunos de los enfoques para emplear los autómatas celulares como herramienta para modelar fenómenos, que si bien tienen relación con este trabajo de tesis, no se conectan directamente con el.

3.1. Cellular Automata Automatically Constructed from a Bioconvection Pattern

Esta investigación (11) se centra en realizar el descubrimiento de las reglas de evolución para un autómata celular unidimensional, empleando métodos de análisis estadísticos para determinar estas reglas a partir de los datos observados de los patrones de bioconvección generados como resultado de la respuesta a una estimulación luminosa de un tipo de alga unicelular llamada Euglena gracilis.

El experimento consistió en iluminar por la parte de abajo de un contenedor anular conteniendo una suspensión de E. gracilis con una fuerte intensidad de luz. Los individuos al ser alcanzados por la luz sufren de fototaxis negativa, lo cual hace que se acumulen cerca de la superficie. Como la densidad del E. gracilis es mas pesada que el agua, las regiones ricas en Euglena se precipitan al fondo para conducir el flujo local. Estas interacciones entre los individuos y el flujo eventualmente forma patrones de bioconvección.
El proceso lo realizan de la siguiente forma:

1. Recopilación de imágenes de los patrones de bioconvección.
2. Transformación de las imágenes recopiladas a escala de grises.
3. Uso de filtros para la eliminación del ruido.
4. Discretización de las imágenes en escala de grises.
5. Se predetermina el número de estados para el sitio k y se discretiza la información observada acorde a esto.
6. Basado en el número de vecinos establecido m se calcula la frecuencia con la que aparecen los estados por cada combinación de posibles estados vecinos.
7. Con base en esta frecuencia se determinan las reglas locales para el AC.

3.2. Discovery of Transition Rules for Cellular Automata Using Artificial Bee Colony and Particle Swarm Optimization Algorithms in Urban Growth Modeling

En este trabajo (14) se muestra un método para descubrir las reglas de transición de un autómata celular de lattice bidimensional que modele el crecimiento urbano empleando un algoritmo de optimización de colonia de abejas artificiales. Para ello utilizan como datos de entrada información multitemporal sensada remotamente de el área urbana de la ciudad de Urmia, Iran.
El autómata celular en este trabajo se define con la siguiente ecuación:

$$P_{ij}^t = S_{ij}^t \times \Omega_{ij}^t \times Con \times e_r \tag{3.1}$$

donde:

- P_{ij}^t es el potencial de desarrollo de una celda ij.
- S_{ij}^t es la idoneidad de la celda ij para cambiar basada en factores relevantes en el tiempo t.
- Ω_{ij}^t es el efecto que tiene la densidad de desarrollo del vecindario.
- Con es una condición funcional que se vuelve verdadero cuando se encuentra la idoneidad de la celda.
- e_r es un termino para la perturbación estocástica por errores.

El potencial de desarrollo se compara con un valor umbral para determinar si una célula no urbanizada va a poder ser transformada en una célula urbanizada en el tiempo $t + 1$. Entonces los valores a encontrar son los valores de umbral para los cuales transiciona una célula no urbanizada a una urbanizada.
Como requerimiento para la evaluación del algoritmo, se emplean otras dos técnicas además de la colonia de abejas artificiales,las cuales son:

- Particle Swarm Optimization (PSO)
- Logistic Regression

Los siguientes diagramas de continuación muestran el procedimiento que se siguió con cada uno de los algoritmos para la calibración de las reglas para el autómata celular.

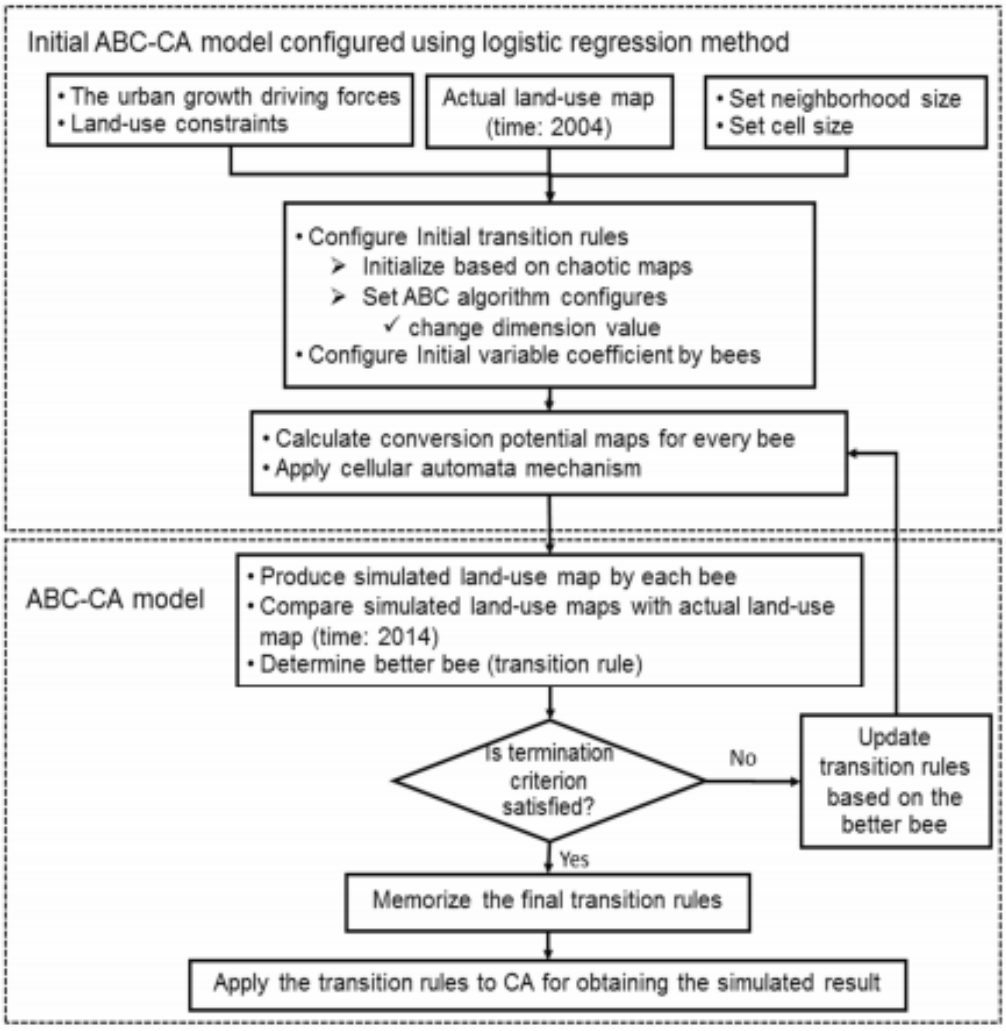

Figura 3.1: Modelo para la obtención de reglas del autómata celular empleando un algoritmo de optimización de colonia de abejas artificiales.

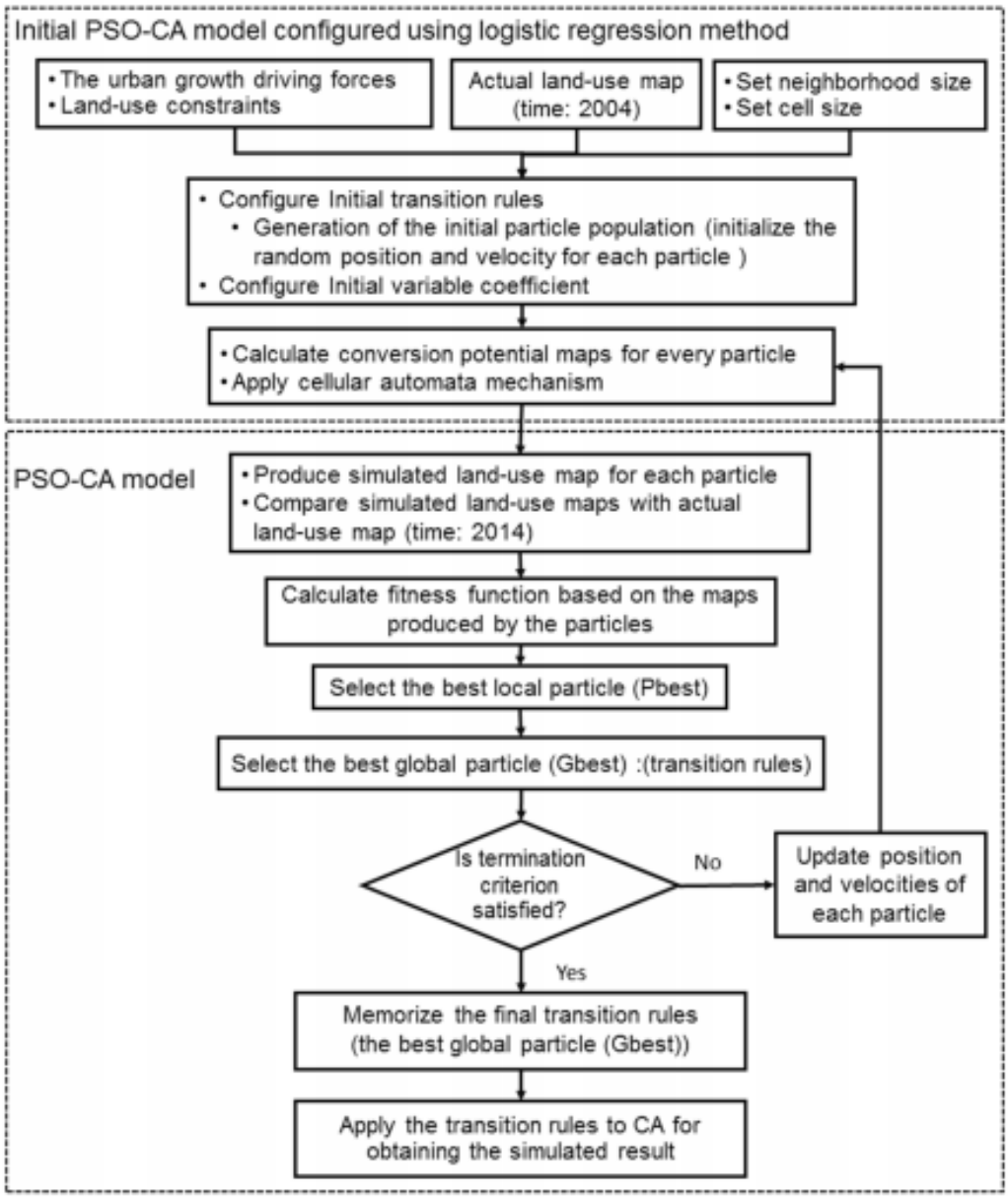

Figura 3.2: Modelo para la obtención de reglas del autómata celular empleando un algoritmo de optimización de enjambre de partículas.

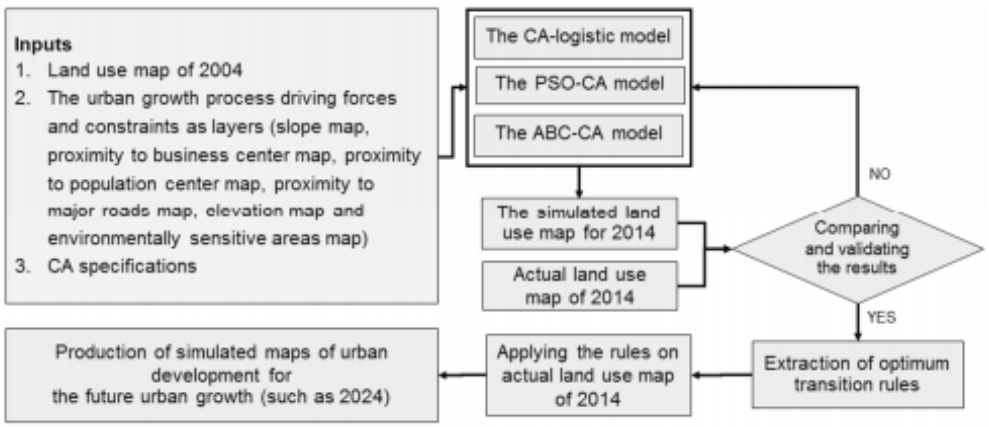

Figura 3.3: Modelo de evaluación de los algoritmos.

Validation Method	Prediction Approach		
	CA-Logistic	PSO-CA	ABC-CA
Overall accuracy (%)	82.8	87.5	89
Figure of merit (%)	30	32.6	35.7
False alarms (%)	15.1	7.7	6.2
Misses (%)	2.1	4.8	4.8
Allocation disagreement (%)	17.2	12.5	11
Correctly predicted unchanged cells (%)	75.4	81.4	82.9
Protection of agricultural areas from urbanization (%)	62.2	68.6	74.1
Areas of the simulated gain of the urban lands in 2004–2014 (the actual gain area of the city is 2500 hectares) (hectares)	4960	3155	2824
TOC (closeness to maximum boundary)	low	medium	high

Figura 3.4: Resultados de la evaluación de los algoritmos.

Como podemos ver en la tabla de resultados el algoritmo de colonia de abejas fue capaz de obtener un mejor rendimiento sobre los otros métodos, lo cual ayudara a obtener un mejor despeño para la estimación correcta del crecimiento urbano. Sin embargo este método aquí planteado sigue teniendo ciertos obstáculos como lo es encontrar la definición correcta de la ecuación que define al autómata celular.

3.3. On Routine Evolution of Complex Cellular Automata

El enfoque de este trabajo (1) es la creación de un método para la obtención de un conjunto de reglas que puedan replicar un comportamiento, sin embargo no toman en cuenta conocimiento previo del fenómeno que se quiere replicar. Lo que se hace es utilizar un algoritmo evolutivo cuya función de aptitud es dependiente del resultado al que se quiere llegar y la codificación de las reglas viene dada de la siguiente forma:

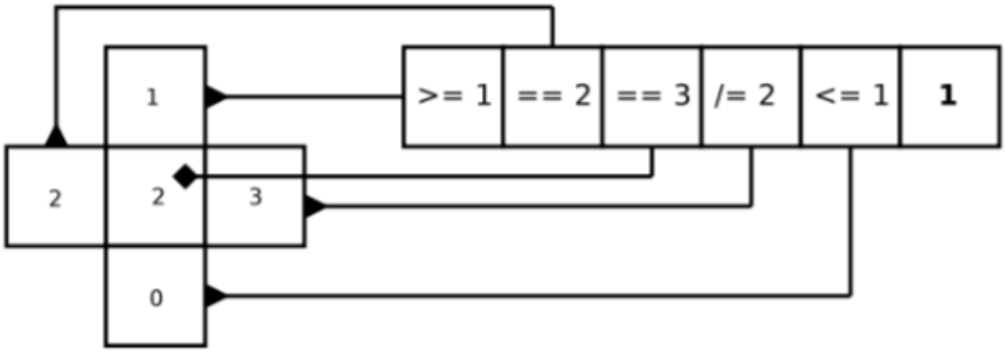

Figura 3.5: Ejemplo de la codificación de las reglas de evolución del autómata con un vecindario de tipo Moore.

Capítulo 4

Métodos y metodologías

En esta sección detallamos cual fue la metodología empleada para llevar acabo los experimentos, en el siguiente diagrama podemos ver el panorama general de la metodología.

4.1. Conjunto de datos

Para realizar nuestros experimentos se opto por adquirir un conjunto de datos que consiste de los estados de la evolución de 4 diferentes autómatas celulares bidimensionales, los cuales fueron obtenidos de (Rucker and Walker). Para cada uno de los autómatas se obtuvo 200 imágenes de 50x50 pixeles en escala rgb.

4.1.1. Brain

Es un autómata celular bidimensional, desarrollado por Brian Silverman, que consiste en lo siguiente:

- **Vecindario:** es de tipo Moore con 8 vecinos.
- **Espacio de estados:** 3 estados
  - 0 que representa el estado apagado
  - 1 que representa el estado encendido
  - 2 que representa el estado muriendo.

- **Función de evolución:**
  - 1 si se encuentra apagado y si dos o mas vecinos se encuentran encendidos.
  - 2 si en el estado anterior estaba encendido.
  - 0 si en el estado anterior estaba muriendo.

4.1.2. Byl

Es un autómata celular bidimensional, desarrollado por Jhon Byl (3), que consiste en lo siguiente:

- **Vecindario:** es de tipo Von Neumann con 5 vecinos.
- **Espacio de estados:** consiste de 6 estados del 0 al 6.
- **Función de evolución:**

CTRBL → I		CTRBL → I		CTRBL → I		CTRBL → I		CTRBL → I	
00003	1	10003	3	20215	5	31215	1	40242	4
00012	2	10004	0	20235	3	31223	1	40252	0
00013	1	10033	0	20252	5	31233	1	40325	5
00015	2	10043	1	2 - - - -	2	31235	5	4 - - - -	3
00025	5	10321	3	30001	0	31432	1	50022	5
00031	5	11253	1	30003	0	31452	5	50032	5
00032	3	12453	3	30011	0	3 - - - -	3	50212	4
00042	2	1 - - - -	4	30012	1			50222	0
0 - - - -	0	20000	0	30121	1	40003	0	50322	0
		20015	5	30123	1	40043	0	5 - - - -	2
10000	0	20022	0	31122	1	40212	0		
10001	0	20202	0	31123	1	40232	0		

Cuadro 4.1: Función de evolución de Byl CA (3).

$$\begin{array}{ccc} & \mathrm{T} & \\ \mathrm{L} & \mathrm{C} & \mathrm{R} \rightarrow \mathrm{I} \\ & \mathrm{B} & \end{array}$$

Cuadro 4.2: Vecindario.

4.1.3. Evoloops

Es un autómata bidimensional, desarrollado por Hiroki Sayama (18), que consiste en lo siguiente:

- **Vecindario:** es de tipo Von Neumann con 5 vecinos.
- **Espacio de estados:** consiste de 8 estados del 0 al 8.
- **Función de evolución:** se muestra en el cuadro 4.3

CTRBL → I		CTRBL → I		CTRBL → I		CTRBL → I		CTRBL → I		CTRBL → I	
00001	2	10202	1	11272	7	20172	2	21322	2	40125	0
00004	3	10211	1	11273	5	20202	2	21422	2	40162	0
00012	2	10212	1	11322	1	20203	2	21622	2	40212	0
00015	2	10213	1	11332	1	20205	2	21722	2	40215	0
00021	2	10221	1	11542	4	20206	5	22224	2	40222	1
00024	2	10224	4	11572	7	20207	3	22227	2	40232	1
00042	2	10227	7	11624	4	20212	2	22234	2	40262	6
00045	2	10232	4	11627	7	20215	2	22237	2	40312	0
00075	2	10241	4	12224	4	20221	2	22243	2	40322	1
00102	2	10242	4	12227	7	20222	2	22244	2	50002	5
00214	1	10243	4	12243	4	20223	2	22273	2	50012	5
00217	1	10251	1	12273	7	20232	3	22277	2	50021	5
00232	2	10252	7	12324	4	20242	2	22324	3	50023	2
01122	1	10254	3	12327	7	20245	2	22327	3	50024	5
01212	1	10257	7	12426	6	20252	5	30001	3	50027	5
01232	1	10271	7	12433	3	20262	0	30002	2	50042	5
01242	1	10272	7	12627	6	20265	0	30003	2	50072	5
01245	1	10273	5	20001	2	20272	2	30004	3	50202	2
01252	6	10512	1	20002	2	20275	2	30007	4	50205	2
01262	6	10542	4	20004	2	20312	2	30012	3	50212	5

01272	1	10572	7	20005	2	20322	2	30032	2	50215	2
01275	1	10621	1	20006	0	20342	2	30042	1	50242	5
01342	1	10624	4	20007	1	20345	2	30102	1	50272	5
01372	1	10627	7	20012	2	20372	2	30125	0	50312	0
01422	1	11112	1	20015	2	20412	2	30212	3	60202	2
01425	1	11122	1	20021	2	20422	2	30242	3	60212	2
01432	1	11124	4	20022	2	20442	2	30252	1	60222	0
01435	1	11125	1	20023	2	20512	2	30272	3	60242	2
01442	1	11127	7	20024	2	20542	5	30332	1	60272	2
01462	1	11162	1	20026	0	20572	5	31212	3	61222	0
01722	1	11212	1	20027	2	20612	5	31242	3	62224	0
01725	1	11213	1	20032	4	20621	2	31252	1	62227	0
01756	1	11215	1	20042	3	20642	5	31272	3	70102	0
01762	1	11222	1	20045	2	20672	5	32424	3	70112	0
01772	1	11224	4	20054	5	20712	2	32425	1	70122	0
10001	1	11227	7	20057	5	20722	2	32427	3	70125	0
10012	1	11232	1	20062	0	20772	2	32527	1	70162	0
10021	1	11242	4	20072	2	21122	2	32727	3	70212	0
10024	4	11243	4	20075	2	21222	2	40000	1	70215	0
10027	7	11252	7	20102	2	21223	2	40002	1	70222	1
10121	1	11254	3	20112	2	21224	2	40102	0	70232	0
10124	4	11257	7	20122	2	21227	2	40112	0	70262	6
10127	7	11262	6	20142	2	21232	3	40122	0	70312	0

Cuadro 4.3: Función de evolución de Evoloops CA (18) .

4.1.4. Mite

Es un autómata bidimensional, desarrollado por Dan Drake, que consiste en lo siguiente:

- **Vecindario:** es de tipo Von Neumann con 8 vecinos.
- **Espacio de estados:** consiste de 3 estados del 0 al 3.
- **Función de evolución:**
  - mover la posición del predador de manera aleatoria dentro de su vecindario local.

- si no hay presa en su posición el predador muere.
- si hay suficientes presas en su posición se incrementa el numero de predadores.
- si hay dos presas juntas se incrementa el numero de presas.

4.2. Preprocesamiento

Una vez adquiridas las secuencias de imágenes de cada autómata celular, se procede a realizar los siguientes preprocesamientos:

- **Discretización**.- pasar las imágenes de los canales rgb a valores discretos.
- **Binarización**.- este paso solo se realiza para los datos que se van a ingresar al algoritmo RA1.

4.2.1. Discretización

La implementación de este procedimiento se realizo con el lenguaje Python y se ejemplifica mejor con el siguiente pseudocódigo.

```
   entrada: La ruta de la carpeta donde se encuentran las imágenes
   salida  : Un archivo con formato .pkl que contiene las imágenes
             procesadas.
 1 diccionario = {} ; contador = 0; imágenes=[ ] ;
    /* inicializaciónes */
 2 para cada Imagen en carpeta hacer
 3 |   Paso 1:Imagen' = Transformar imagen a escala de grises;
   |   /* Se considera a la imagen como una matriz de pixeles
   |      */
 4 |   nuevaImagen = [ ];
 5 |   para cada fila en Imagen hacer
   |   |   /* una fila es un arreglo de pixeles              */
 6 |   |   nuevaFila = [ ] ;
 7 |   |   para cada pixel en fila hacer
   |   |   |   /* un pixel corresponde a un valor entre 0 y 255
   |   |   |      despues de la transformación               */
 8 |   |   |   si pixel no esta en diccionario entonces
 9 |   |   |   |   Paso 2: diccionarion[pixel] = contador;
10 |   |   |   |   Paso 3: contador = contador + 1;
11 |   |   |   fin
12 |   |   |   Paso 4: nuevaFila.agregar(diccionario[pixel]);
13 |   |   fin
14 |   |   Paso 5: nuevaImagen.agregar(nuevaFila);
15 |   fin
16 |   Paso 6: imagenes.agregar(nuevaImagen);
17 fin
18 Paso 7: Guardar imágenes en formato .pkl;
```

Algoritmo 5: Pseudocódigo para la discretización de las imágenes.

4.2.2. Binarización

Este proceso se realiza solamente para los datos de entrada que se van a ingresar al algoritmo RA1 ya que este algoritmo solo es capaz de aprender de datos categóricos. El siguiente pseudocódigo ejemplifica este proceso.

```
   entrada: MEstado: matriz de estado
   salida  : El estado binarizado
 1 nuevoEstado = [];
 2 para cada fila en MEstado hacer
 3 |   nuevaFila = [];
 4 |   para cada celda en fila hacer
 5 |   |   Paso 1: Encontrar el dominio de la celda;
 6 |   |   Paso 2: dominio = ordenarAscendente (dominio);
 7 |   |   para cada valor en dominio hacer
 8 |   |   |   si celda <valor entonces
 9 |   |   |   |   Paso 3a: nuevaFila.agregar(1);
10 |   |   |   en otro caso
11 |   |   |   |   Paso 3b: nuevaFila.agregar(0);
12 |   |   |   fin
13 |   |   fin
14 |   fin
15 |   Paso 4: nuevoEstado.agregar(nuevaFila);
16 fin
```

Algoritmo 6: Pseudocódigo para la discretización de las imágenes.

4.3. Algoritmos de aprendizaje

Los algoritmos de aprendizaje utilizados en este trabajo consisten en los siguientes:

- RA1
- GA-Nuggets
- LRDEA el algoritmo que estamos proponiendo.

Debido a que no se encontraron la implementación de los algoritmos RA1 y GA-Nuggets, se realizo la tarea de realizar dicha implementación en el lenguaje de programación Python.

4.3.1. LRDEA (Local Rule Discovery Evolutive Algorithm)

Nuestra propuesta de solución para este problema se basa en un algoritmo genético distribuido, cuyo funcionamiento combina el principio del algoritmo RA1 que es ir generando clausulas que cubran a los ejemplos positivos y rechace a los ejemplos negativos, y el principio de un algoritmo genético que es tener recombinación entre un conjunto de individuos para explorar el espacio de búsqueda. El algoritmo maneja una subpoblación por cada una de los valores en el espacio de estado del autómata celular.

Representación

Al igual que el algoritmo GA-Nuggets la codificación para cada individuo representa una regla de predicción candidata de la forma IF *Ant* THEN *Cons*, donde *Ant* es el antecedente de la regla y *Cons* es el consecuente. El antecedente *Ant* consiste en una conjunción de condiciones, donde cada condición es un par vecino valor de la forma $n_i = V_i$, donde n_i es el i-ésimo vecino y n_i es un conjunto de los posibles valores que puede tener n_i de la forma $\{v_0, \ldots, v_n\}$ donde v es un valor dentro del dominio del vecino n_i. El algoritmo solo maneja valores categóricos, por lo que es necesario discretizar los valores. El consecuente *Cons* consiste en un solo valor en el dominio que puede tomar la celda que se esta evaluando c. Se utiliza el valor -1 para indicar que el par vecino valor no va a formar parte de la regla.

$$\begin{array}{ccc} n_1 & n_2 & n_3 \\ n_4 & c & n_5 \\ n_6 & n_7 & n_8 \end{array} \rightarrow \text{Cons}$$

Cuadro 4.4: Representación del vecindario.

$n_1 = V_1$	$n_2 = V_2$	$n_3 = V_3$	$n_4 = V_4$	$c = V_c$	$n_5 = V_5$	$n_6 = V_6$	$n_7 = V_7$	$n_8 = V_8$	Cons

Cuadro 4.5: Representación del la codificación.

Función de aptitud

Nuestra función de aptitud esta definida por la siguiente ecuación:

$$Fitness(C, E^+) = \sum_{i=1}^{|E^+|} \left(\frac{1}{|C|} \sum_{j=1}^{|C|} Eq(E_{ij}^+, C_j) \right)$$

$$Eq(x, y) = \begin{cases} 1 & if\ x = y \\ 0 & otherwise \end{cases}$$

donde:

- C es el cromosoma.
- $|C|$ es la cardinalidad del cromosoma.
- E^+ es un subconjunto de los ejemplos que pertenece a la clase para la cual se esta evaluando la aptitud del cromosoma.
- $|E^+|$ es la cardinalidad de E^+.
- $E_{i,j}^+$ es un valor en la posición j del elemento en la posicion i de E^+.
- C_j es el valor del cromosoma en la posición j.

Mecanismo de selección de padres

Para el mecanismo de selección de padres se escogió la selección por torneo, que consiste en lo siguiente:

entrada: k (la cantidad de individuos participantes en el torneo)
1 **Paso 1:** Escoger k individuos de la población aleatoriamente;
2 **Paso 2:** Escoger el individuo más apto del torneo con probabilidad p;
3 **Paso 3:** Escoger el segundo individuo más apto con probabilidad p(1-p) ;
4 **Paso 4:** Escoger el k-esimo individuo más apto con probabilidad $p(1-p)^k$;

Algoritmo 7: Pseudocódigo de selección por torneo.

Cruza

Para el proceso de cruce se seleccionó el método de cruce en un punto, esto es de la siguiente manera.

entrada: padre1, padre2
1 **Paso 1:** Escoger de forma aleatoria el punto p donde p es menor a la cardinalidad de padre1 y padre2;
2 **Paso 2:** Partir el cromosoma del padre1 y el padre2 en el punto p;
3 **Paso 3:** Recombinar la primera mitad del padre1 con la segunda mitad del padre2 para generar al hijo1;
4 **Paso 4:** Recombinar la segunda mitad del padre1 con la primera mitad del padre2 para generar al hijo2;

Algoritmo 8: Pseudocódigo de cruce en un punto.

Mutación

El operador de mutación consiste en dos partes, la primera es una probabilidad de ser mutado, la segunda es la selección del gen que se va a mutar. El gen mutado puede adquirir un nuevo valor en el conjunto de valores o puede removerse uno de estos valores.

Mecanismo de selección de sobrevivientes

Para el mecanismo de selección de sobrevivientes se ordenan los individuos de cada subpoblación y se dividen en dos, se toman los primeros n elementos de cada una de las mitades, y se descarta el resto.

4.4. Simplificación de reglas

Para la simplificación de las reglas se utiliza el algoritmo desarrollado por Willard V.Quine 1955 y extendido Edward J. McCluskey en 1956. Este mecanismo solo se emplea para la simplificación de las reglas generadas por el algoritmo RA1.

4.5. Evaluación

Capítulo 5

Resultados y análisis

Capítulo 6

Conclusiones y trabajo futuro

En base a los resultados obtenidos en XX, YY y ZZ podemos concluir AA y BB pero no CC por tanto DD.

Bibliografía

[1] Bidlo, M. (2016). On routine evolution of complex cellular automata. *IEEE Transactions on Evolutionary Computation*, 20(5):742–754.

[2] Boerlijst, M. (1992). k hogeweg, p.(1992). self-structuring and selection: Spiral waves as a substrate for prebiotic evolution. *Artificial Life II. Addison Wesley, Reading, Mass*, page 255.

[3] Byl, J. (1989). Self-reproduction in small cellular automata. *Physica D: Nonlinear Phenomena*, 34(1):295 – 299.

[4] Cook, M. (2004). Universality in elementary cellular automata. *Complex systems*, 15(1):1–40.

[5] Culik II, K., Hurd, L. P., and Yu, S. (1990). Computation theoretic aspects of cellular automata. *Physica D: Nonlinear Phenomena*, 45(1-3):357–378.

[6] Deshpande, A. and Triantaphyllou, E. (1998). A greedy randomized adaptive search procedure (grasp) for inferring logical clauses from examples in polynomial time and some extensions. *Mathematical and Computer Modelling*, 27(1):75 – 99.

[7] Ermentrout, G. B. and Edelstein-Keshet, L. (1993). Cellular automata approaches to biological modeling. *Journal of theoretical Biology*, 160(1):97–133.

[8] Gardner, M. (1970). Mathematical games. *Scientific American*, 222(6):132–140.

[9] Hillis, W. D. (1984). The connection machine: A computer architecture based on cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):213–228.

[10] Ilachinski, A. (2001). *Cellular automata: a discrete universe*. World Scientific Publishing Company.

[11] Kawaharada, A., Shoji, E., Nishimori, H., Awazu, A., Izumi, S., and Iima, M. (2016). Cellular automata automatically constructed from a bio-convection pattern. In *Recent Advances in Natural Computing*, pages 15–25. Springer.

[12] Manneville, P., Boccara, N., Vichniac, G. Y., and Bidaux, R. (2012). *Cellular Automata and Modeling of Complex Physical Systems: Proceedings of the Winter School, Les Houches, France, February 21–28, 1989*, volume 46. Springer Science & Business Media.

[13] Margolus, N., Toffoli, T., and Vichniac, G. (1986). Cellular-automata supercomputers for fluid-dynamics modeling. *Physical Review Letters*, 56(16):1694.

[14] Naghibi, F. and Delavar, M. (2016). Discovery of transition rules for cellular automata using artificial bee colony and particle swarm optimization algorithms in urban growth modeling. *ISPRS International Journal of Geo-Information*, 5(12):241.

[15] Nordahl, M. G. (1989). Formal languages and finite cellular automata. *Complex Systems*, 3(1).

[16] Peysakh, J. (1987). *Fast Algorithm to Convert Boolean Expression Into a DNF*. IBM Corporation, Thomas J. Watson Research Center.

[Rucker and Walker] Rucker, R. and Walker, J.

[18] Sayama, H. (1998). Constructing evolutionary systems on a simple deterministic cellular automata space.

[19] Simon, P. and Nagel, K. (1998). Simplified cellular automaton model for city traffic. *Physical Review E*, 58(2):1286.

[20] Tamayo, P. and Hartman, H. (1987). Cellular automata, reaction-diffusion systems, and the origin of life. In *ALIFE*, pages 105–124.

[21] Toffoli, T. (1984). Cellular automata as an alternative to (rather than an approximation of) differential equations in modeling physics. *Physica D: Nonlinear Phenomena*, 10(1-2):117–127.

[22] Vichniac, G. Y. (1984). Simulating physics with cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):96–116.