Instituto Politécnico Nacional

Escuela Superior De Computo
ESCOM

REPORTE PROGRAMA 2

David Baltazar Real

Grupo: 3CM19

Asignatura: Computing Selected Topics

Profesor: Juarez Martinez Genaro

Fecha de entrega: 15 de Enero del 2023

Índice general

Índice de figuras

Capítulo 1

Introduccion

Los atractores (o también conocidos como campos de atracción) son una forma de visualizar a un sistema complejo y sus posibles soluciones además de que ayudan a facilitar el estudio de estos, la noción de este modelo fue propuesto por C. C. Walker.
La topología de estos campos de atracción está definida por la función de transición del autómata celular al que pertenece y definen el comportamiento global de un sistema complejo, los estados globales están representados por nodos unidos por aristas, cada nodo puede tener cero o más aristas entrantes, pero como los sistemas son deterministas tienen solo una arista de salida.
En estos campos de atracción se pueden identificar los "ciclos atractores" que es como su nombre lo indica un conjunto de nodos que se ciclan por medio de las aristas dando a entender que son las soluciones del sistema. Otros elementos importantes de los campos de atracción son unos nodos denominados "Jardines del Edén" los cuales son aquellos que no poseen una arista de entrada, el conjunto de los caminos formados por todos los nodos "Jardines del Edén" que se dirigen a un nodo en común del ciclo atractor es denominado "Árbol de Trascendencia" de la cual se desprende la llamada "Rama de Trascendencia", y es el conjunto de todos los caminos que van de un nodo "Jardín del Edén" que van a un nodo dentro de un "Árbol de Trascendencia".
Los campos de atracción que se muestran en este reporte se realizan a partir del sistema complejo denominado Game of Life de John Horton Conway con espacios de 2x2, 3x3, 4x4 y 5x5, utilizando las reglas predeterminadas R(2,3,3,3) y otra regla R(2,2,7,7).

Capítulo 2

Capturas

Las capturas fueron hechas al utilizar el software Gephi para la visualizacion de los atractores

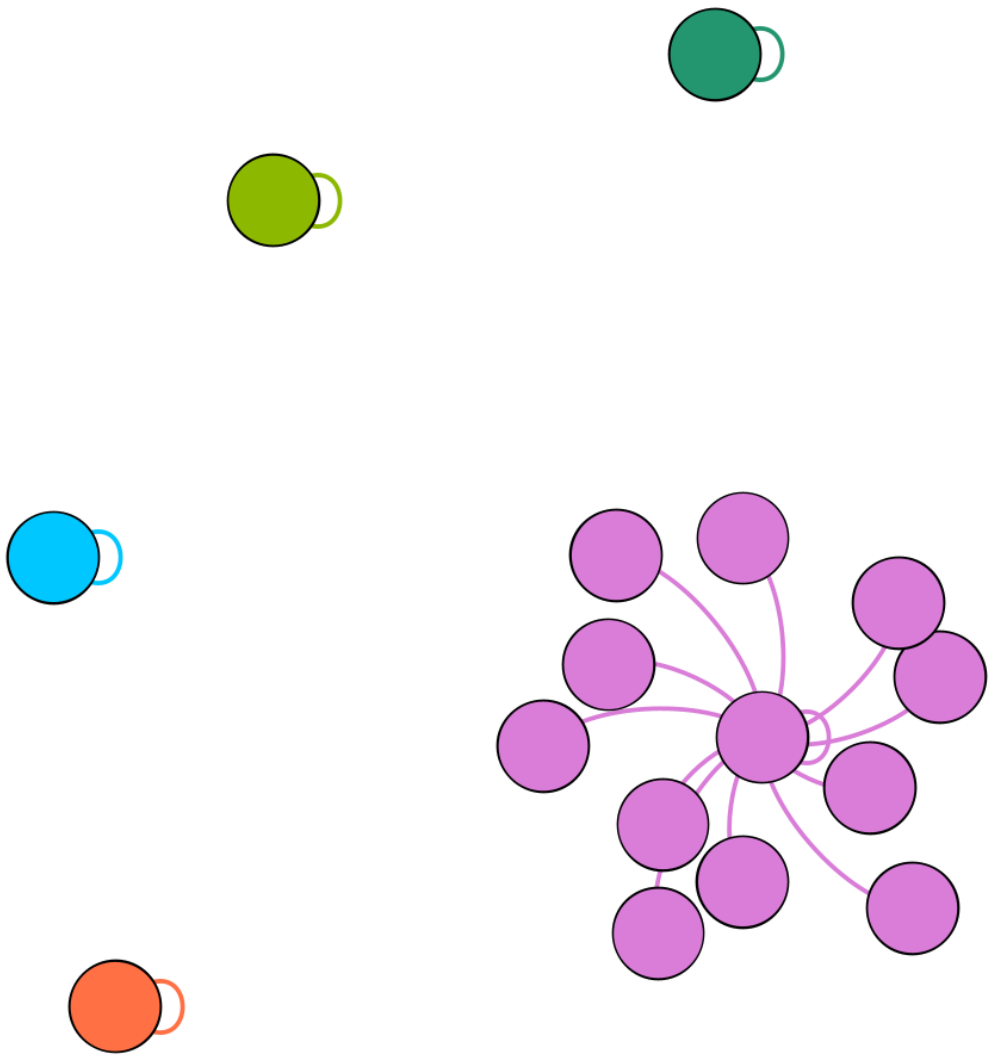

Figura 2.1: Prueba de atractores en un espacio 2x2 con R(2,3,3,3)

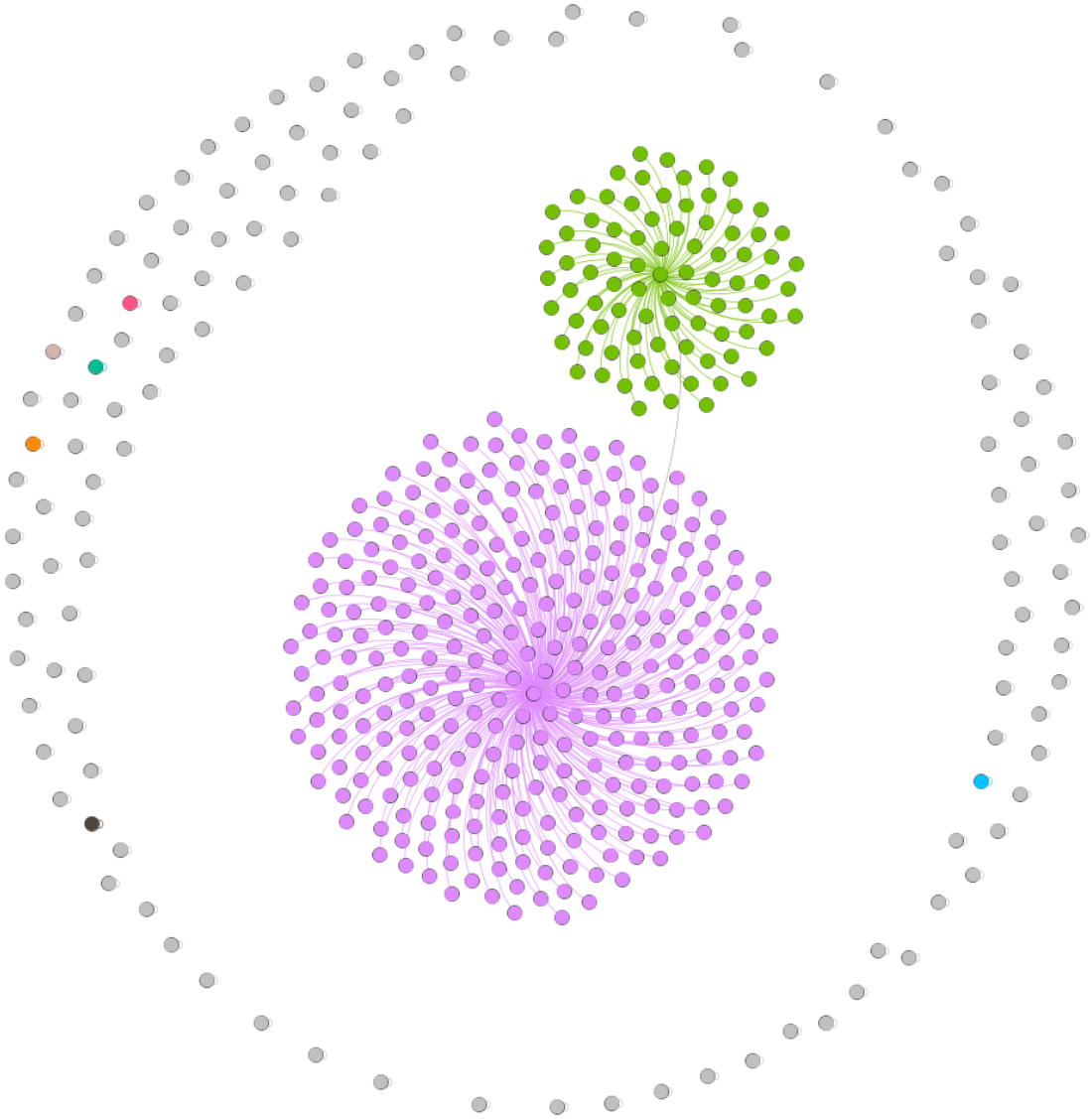

Figura 2.2: Prueba de atractores en un espacio 3x3 con R(2,3,3,3)

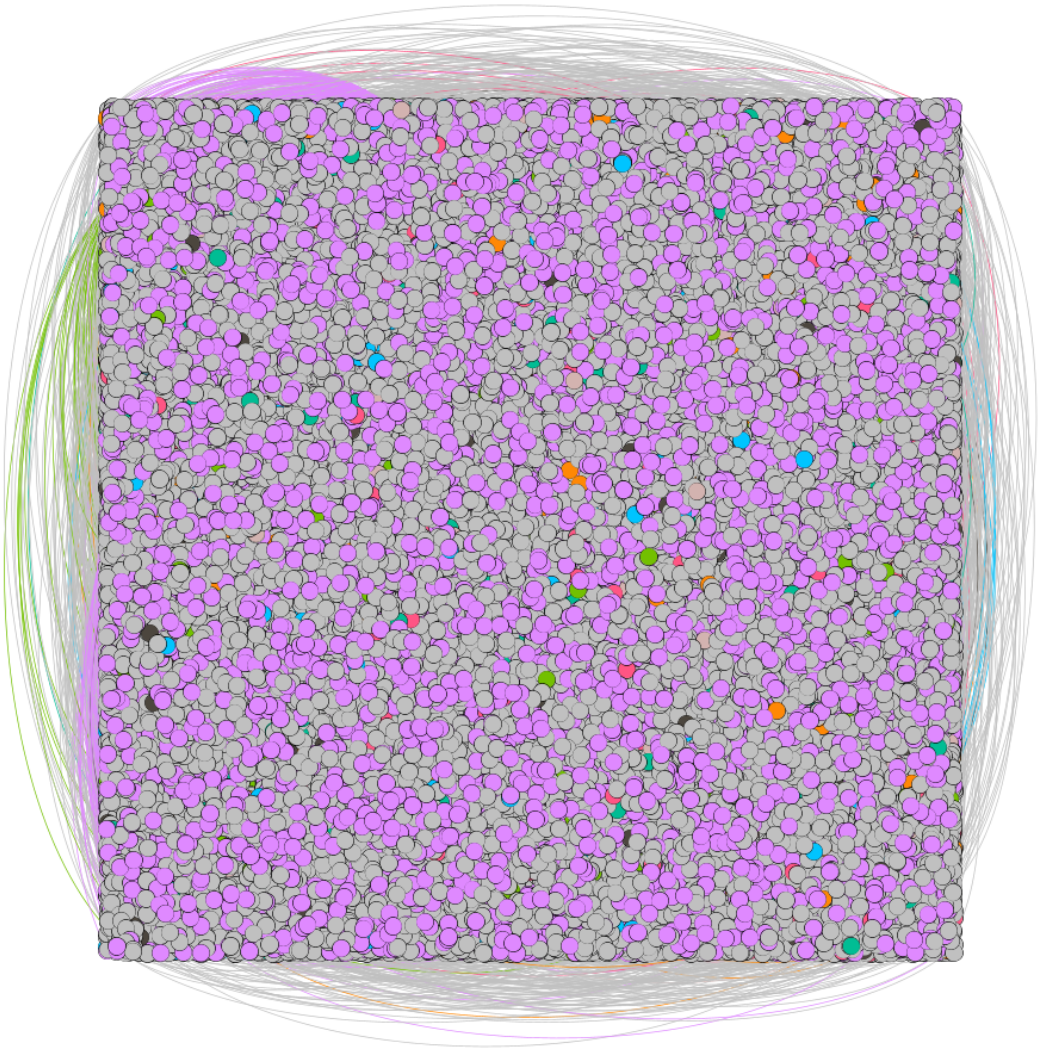

Figura 2.3: Prueba de atractores en un espacio 4x4 con R(2,3,3,3)

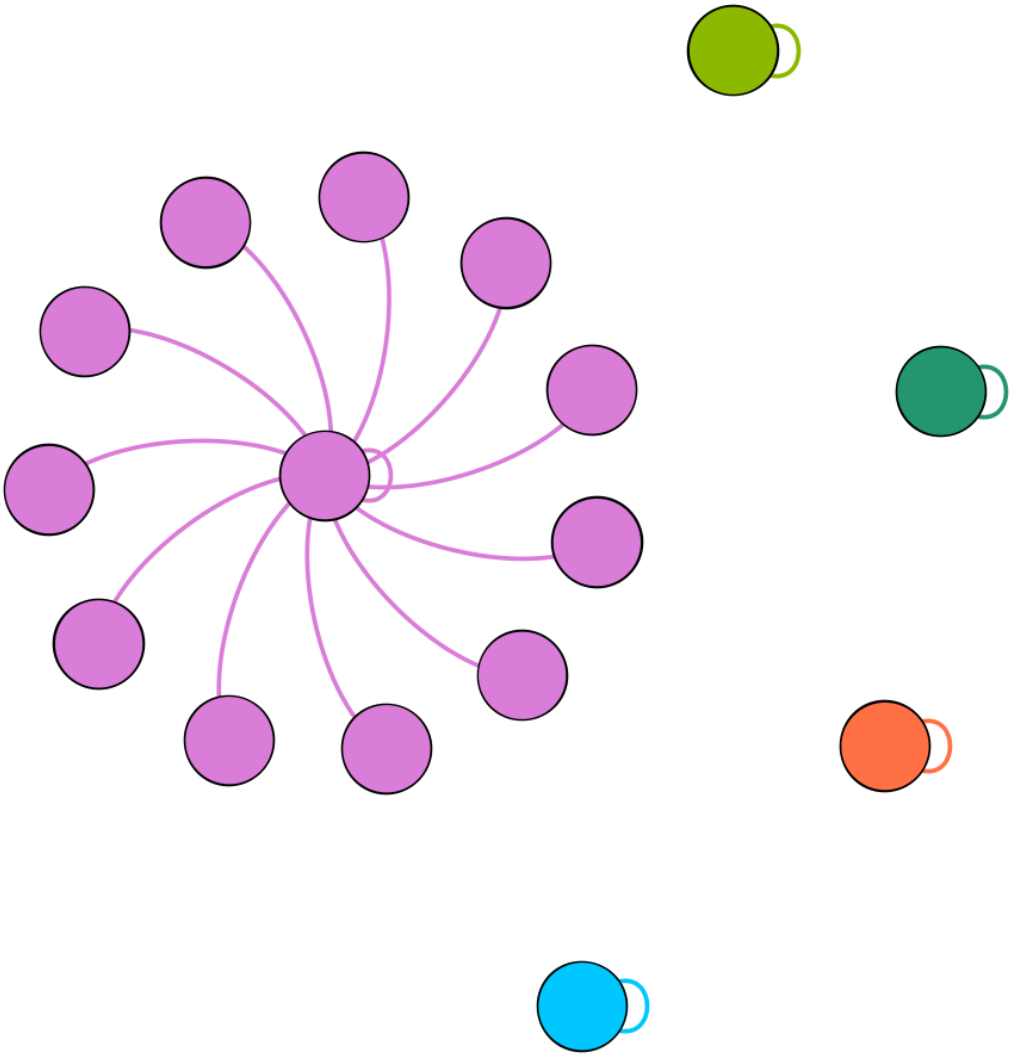

Figura 2.4: Prueba de atractores en un espacio 2x2 con R(2,2,7,7)

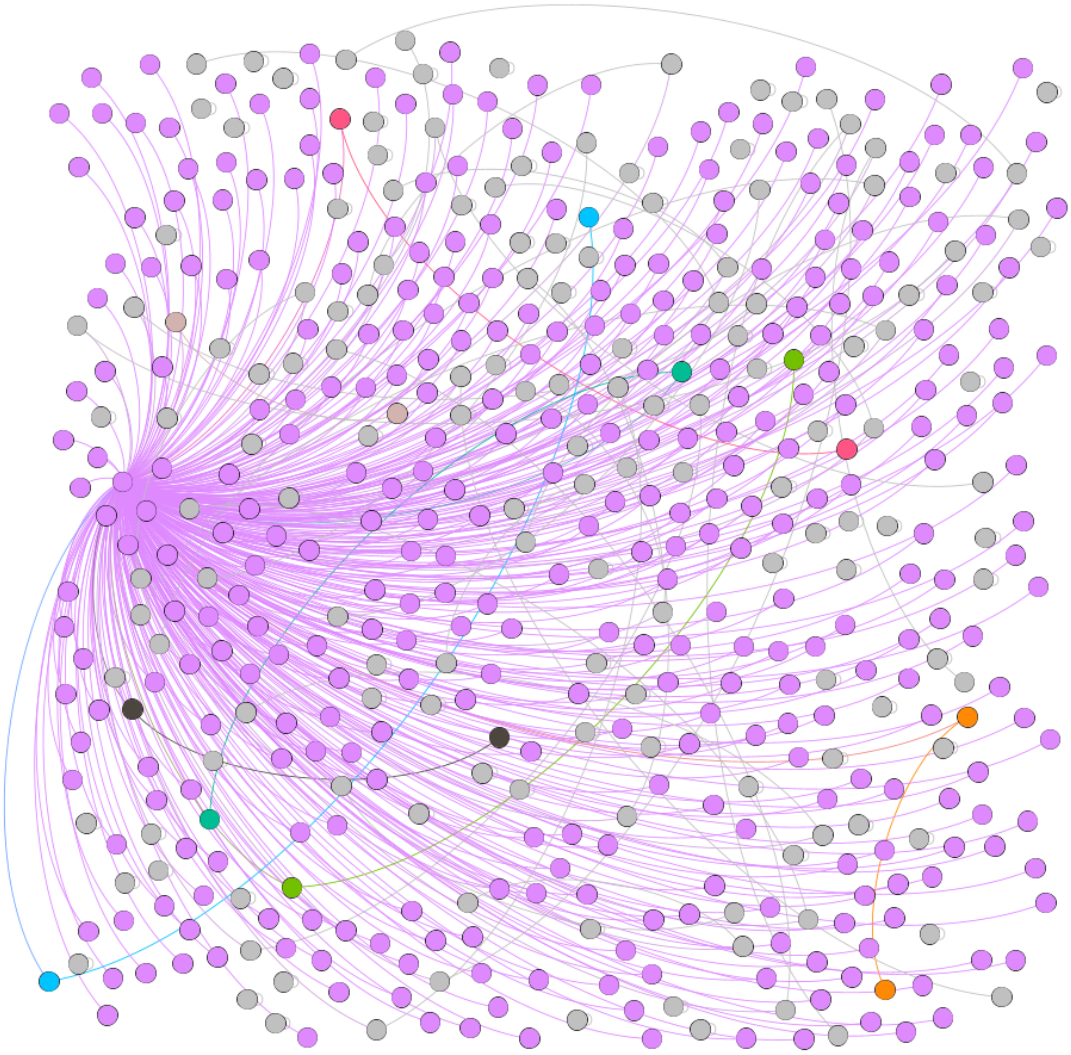

Figura 2.5: Prueba de atractores en un espacio 3x3 con R(2,2,7,7)

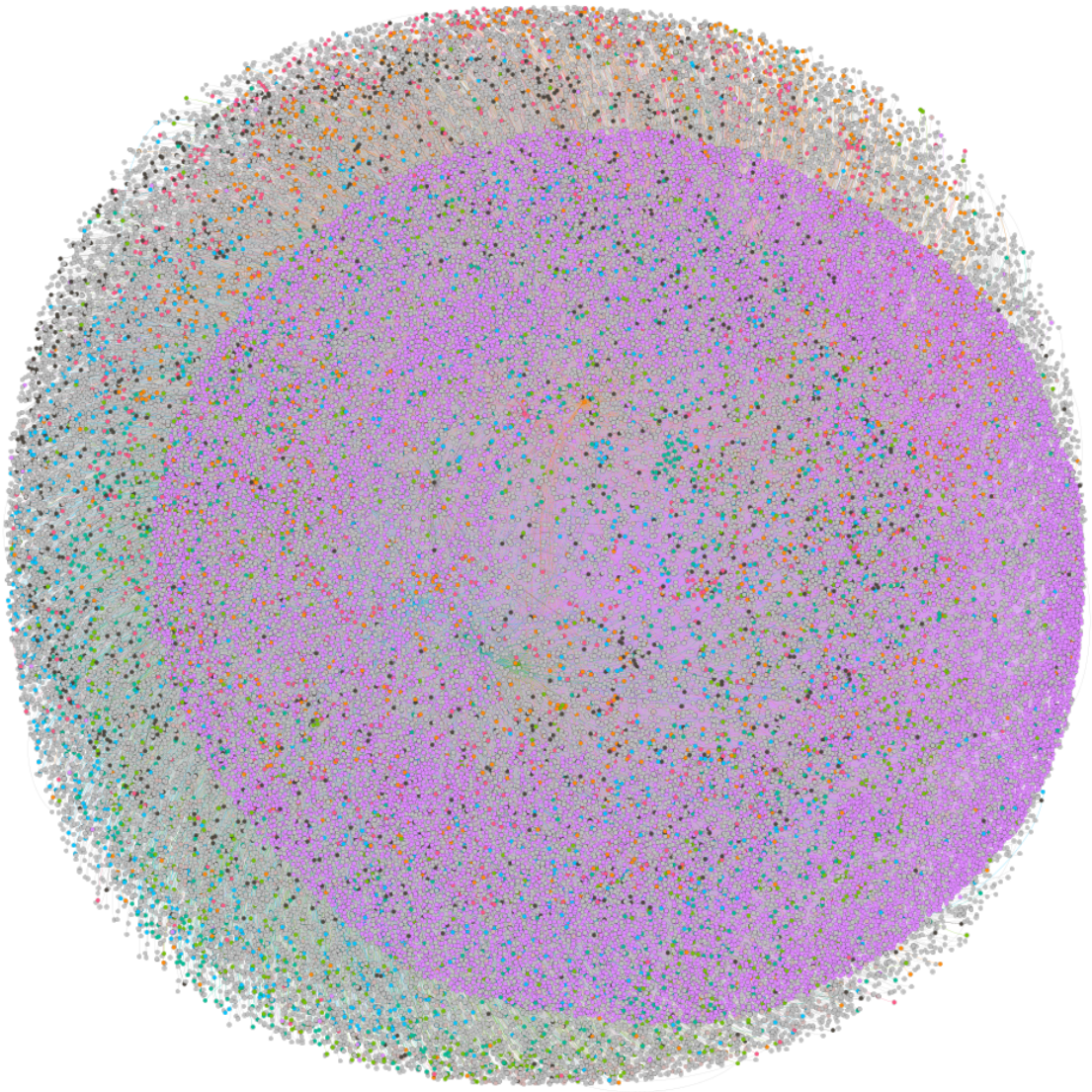

Figura 2.6: Prueba de atractores en un espacio 4x4 con R(2,2,4,4)

Capítulo 3

Codigos

3.1. atractores.py

```
1
2  from life import *
3  from progress.bar import ChargingBar
4  import sys
5
6  # Reglas de LIFE
7  sMin = 2
8  sMax = 3
9  bMin = 3
10 bMax = 3
11
12 def main():
13   if len(sys.argv) == 3:
14     nombre_archivo = sys.argv[1]
15     N = int(sys.argv[2])
16   else:
17     print("uso: python atractores.py <nombre_archivo> <N>")
18     return
19
20   f = open(nombre_archivo+"_nodos.csv", 'w')
21   f2 = open(nombre_archivo+"_aristas.csv", 'w')
22   f.write("ID,Label\n")
23   f2.write("Source,Target\n")
24
25   combinaciones = 2**(N*N)  # Total de combinaciones en el tablero
26   bar = ChargingBar('Calculando Atractores:', max=combinaciones)
27   # Calcula la funcion de transicion
28   for estado in range(0, combinaciones):
29     tablero = numToMatriz(estado, N)
30     nextGen(tablero, sMin, sMax, bMin, bMax)
31     estadoSig = matrizToNum(tablero, N)
32     f.write("{},{}\n".format(estado, estado))
33     f2.write("{},{}\n".format(estado, estadoSig))
34     bar.next()
35   f.close()
```

```
  f2.close()

  print("")

main()
```

3.2. life.py

```

from Cell import Cell


# Funciones

def printMatriz(matriz):
  for y in range(0, len(matriz)):
    for x in range(0, len(matriz)):
      print(matriz[y][x].estado, end=" ")
    print("")


def nextGen(matriz, sMin, sMax, bMin, bMax):
  for y in range(0, len(matriz)):
    for x in range(0, len(matriz)):
      # Calculamos el siguiente estado de las celulas
      suma = matriz[y][x].sumarVecinos()
      if (suma < sMin) or (suma > sMax):
        matriz[y][x].estadoSig = 0
      if (suma >= bMin) and (suma <= bMax):
        matriz[y][x].estadoSig = 1

  for y in range(0, len(matriz)):
    for x in range(0, len(matriz)):
      matriz[y][x].mutar()


def numToMatriz(num, n):
  matriz = []
  exp = 2**((n*n)-1)
  for y in range(n-1, -1, -1):
    aux = []
    for x in range(n-1, -1, -1):
      if ((num % exp) != num):
        aux.insert(0, Cell(x, y, 1))
        num = num - exp
      else:
        aux.insert(0, Cell(x, y, 0))
      exp = exp / 2
    matriz.insert(0, aux)

  for y in range(0, n):
    for x in range(0, n):
```

```
45        matriz[x][y].agregarVecinos(matriz)
46    return matriz
47
48
49  def matrizToNum(matriz, n):
50    exp = 1
51    num = 0
52    for y in range(0, n):
53      for x in range(0, n):
54        num = num + (matriz[y][x].estado * exp)
55        exp = exp * 2
56    return num
57
58
```

3.3. Cell.py

```
1
2   class Cell:
3     def __init__(self, x, y, estado):
4       self.x = x
5       self.y = y
6       self.estado = estado
7       self.estadoSig = self.estado
8
9       self.vecinos = []
10
11  def agregarVecinos(self, tablero):
12    N = len(tablero)
13    for i in range(-1, 2):
14      for j in range(-1,2):
15        xVecino = (self.x + j + N) % N
16        yVecino = (self.y + i + N) % N
17
18        if i != 0 or j != 0:
19          self.vecinos.append(tablero[yVecino][xVecino])
20
21  def sumarVecinos(self):
22    suma = 0
23    for vecino in self.vecinos:
24      suma = suma + vecino.estado
25    return suma
26
27  def mutar(self):
28    self.estado = self.estadoSig
```

Capítulo 4

Conclusiones

En esta práctica con ayuda de la visualización de los atractores podemos observar como el comportamiento caótico del autómata Celular "Game of Life" va creciendo al ir aumentando el tamaño del espacio ya que la cantidad de estados posibles aumenta de la manera $2\hat{}(n*n)$ y por lo mismo tampoco se puede apreciar bien el campo de atracción en los espacios mas grandes como 4x4 o 5x5(razón por la cual decidí no poner los diagramas del 5x5) pero algo que si se puede apreciar es la diferente forma de distribución que tiene el AC al aplicar dos reglas diferentes y viéndose de forma que al utilizar la configuración R(2,2,7,7) se observa como un sistema mas complejo que el R(2,3,3,3).

Bibliografía

[1] Wuensche, A., & Lesser, M. *(1992). The Global Dynamics of Cellular Automata: An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata (2nd edition).* Perseus Books.

[2] J. Martínez, G., Adamatzky, A., Chen, B., Chen, F., & Mora, J. (2010). *imple networks on complex cellular automata: From de Bruijn diagrams to jump-graphs.* (1.a ed.) [Pdf]. Springer.