Entrega de trabajos del tercer parcial

Barrera Pérez Carlos Tonatihu
Profesor: Genaro Juárez Martínez
Computing Selected Topics
Grupo: 3CM8

28 de noviembre de 2018

Índice

1. Autómata celular con matriz auxiliar

1.1. Introducción

Los autómatas celulares(AC) surgen en la década de 1940 con John Von Neumann, que intentaba modelar una máquina que fuera capaz de auto-replicarse, llegando así a un modelo matemático de dicha maquina con reglas complicadas sobre una red rectangular. Inicialmente fueron interpretados como conjunto de células que crecían, se reproducían y morían a medida que pasaba el tiempo. A esta similitud con el crecimiento de las células se le debe su nombre.[1]

Un autómata celular se caracteriza por contar con los siguientes elementos:

1. Arreglo regular. Ya sea un plano de dos dimensiones o un espacio n-dimensional, este es el espacio de evoluciones, y cada división homogénea del arreglo es llamada célula.

2. Conjunto de estados. Es finito y cada elemento o célula del arreglo toma un valor de este conjunto de estados. También se denomina alfabeto. Puede ser expresado en valores o colores.

3. configuración inicial. Consiste en asignar un estado a cada una de las células del espacio de evolución inicial del sistema.

4. Vecindades. Define el conjunto contiguo de células y posición relativa respecto a cada una de ellas. A cada vecindad diferente corresponde un elemento del conjunto de estados.

5. Función local. Es la regla de evolución que determina el comportamiento del A. C. Se conforma de una célula central y sus vecindades. Define como debe cambiar de estado cada célula dependiendo de los estados anteriores de sus vecindades. Puede ser una expresión algebraica o un grupo de ecuaciones.

1.2. Práctica a realizar

Este programa implementa la simulación de un autómata celular. Los puntos importantes a señalar son que cuenta con una interfaz gráfica para el usuario en la cual aparecen los unos (célula viva) y ceros (célula muerta) que son el principal elemento en este autómata celular y que son representados como pequeños cuadros que cambian su tamaño de acuerdo a la población que se tenga. Las características de este simulador son las siguientes:

- Permitir seleccionar el tamaño de la población de la matriz de unos y ceros.
- Permitir seleccionar la regla que se utilizara en cada iteración de la simulación.
- Se podrá elegir la distribución de unos que habrá en la matriz.

- Se podrá cambiar los colores de los ceros y los unos de la simulación.
- Se mostrara el cambio de unos que hay a lo largo de cada iteración.

El objetivo que se tiene es mostrar una matriz de hasta 1000 por 1000 para poder observar un comportamiento que nos proporcione información.

Finalmente, se cuenta con la opción para utilizar una matriz auxiliar junto con una función Φ para realizar el calculo de las matrices a través de las generaciones futuras. Esta opción consiste en lo siguiente.

1. Se elige una función Φ la cual puede ser de paridad, máximo o mínimo. También, se asigna un valor de τ el cual debe de ir de 3 a 8.
2. Después de τ iteraciones se utiliza la función Φ para obtener una matriz auxiliar, la función Φ utiliza como parámetros los valores de las τ matrices anteriores.
3. A la matriz auxiliar se le debe de aplicar la regla del autómata celular que se ha estado utilizado para calcular las iteraciones por lo que ahora tendremos una nueva matriz.
4. Esto se repite las veces que se desee.

1.3. Desarrollo

Este programa fue desarrollado utilizando JavaScript junto a otras herramientas para el desarrollo web por lo que para poder ver ejecutarlo se necesita un navegador web moderno y una conexión a internet para poder cargar la interfaz en su totalidad.

Archivo: index.html

Este archivo contiene la interfaz web que se le muestra al usuario y en donde se apreciara todo el funcionamiento del autómata, se utilizo la librería *Bootstrap* para mostrar una interfaz amigable.

```
<!DOCTYPE html>
<html>
<head>
    <title>Proyecto final</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
    />
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
    bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/
    SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
    crossorigin="anonymous">
    <style type="text/css">
        body {
            margin:0px;
        }
```

```
12          .custom-scroll {
13              overflow-x: scroll;
14              overflow-y: scroll;
15          }
16      </style>
17  </head>
18  <body>
19      <div class="container-fluid">
20      <div class="row">
21          <div class="col-8 custom-scroll">
22              <canvas id="canvas" width="1000" height="1000">
23              </canvas>
24          </div>
25          <div class="col-4">
26              <div class="row">
27                  <p id="information" class="lead"></p>
28                  <canvas id="myChart"></canvas>
29              </div>
30              <div class="form-group">
31                  <div class="row">
32                      <div class="col">
33                          <label for="size">
34                          Tamanio de la matriz:
35                          </label>
36                      </div>
37                      <div class="col">
38                          <input type="number" placeholder="Tamanio" id="
    size" class="form-control" value="100"/>
39                      </div>
40                  </div>
41              </div>
42              <div class="form-group">
43                  <div class="row">
44                      <div class="col-2">
45                          <label for="b1">B1:</label>
46                      </div>
47                      <div class="col">
48                          <input type="number" placeholder="B1" id="b1"
    class="form-control" value="2"/>
49                      </div>
50                      <div class="col-2">
51                          <label for="b2">B2:</label>
52                      </div>
53                      <div class="col">
54                          <input type="number" placeholder="B2" id="b2"
    class="form-control" value="3"/>
55                      </div>
56                  </div>
57              </div>
58              <div class="form-group">
59                  <div class="row">
60                      <div class="col-2">
```

```
61                      <label for="s1">S1:</label>
62                  </div>
63                  <div class="col">
64                      <input type="number" placeholder="S1" id="s1"
    class="form-control" value="3" />
65                  </div>
66                  <div class="col-2">
67                      <label for="s2">S2:</label>
68                  </div>
69                  <div class="col">
70                      <input type="number" placeholder="S2" id="s2"
    class="form-control" value="3"/>
71                  </div>
72              </div>
73          </div>
74          <div class="form-group">
75              <label for="distribution">
76              Distribucion de unos:
77              <span id="distribution_value">50</span>
78              %</label>
79              <input type="range" min="0" max="100" value="50" id="
    distribution">
80          </div>
81          <div class="row">
82              <div class="col">
83                  <label for="input_function">
84                      Funcion a utilizar:
85                  </label>
86              </div>
87              <div class="col">
88                  <select name="type_function" id="input_function"
    class="form-control">
89                      <option value="1">Maximo</option>
90                      <option value="2">Minimo</option>
91                      <option value="3">Paridad</option>
92                  </select>
93              </div>
94                  <div class="col-1">
95                      <label for="tau">Tau:</label>
96                  </div>
97                  <div class="col">
98                      <input type="number" placeholder="tau" id="tau"
    class="form-control" value="3"/>
99                  </div>
100             </div>
101             <div class="form-group">
102                 <div class="row">
103                     <div class="col">
104                         <label for="color_ones">Color de unos:</label
    >
105                     </div>
106                     <div class="col">
```

```
107                             <input type="color" value="#000000"  id="
    color_ones">
108                         </div>
109                         <div class="col">
110                             <label for="color_zeros">Color de ceros:</
    label>
111                         </div>
112                         <div class="col">
113                             <input type="color" value="#ffffff"  id="
    color_zeros">
114                         </div>
115                     </div>
116                 </div>
117                 <div class="form-group">
118                     <div class="row">
119                         <div class="col">
120                             <button id="btn_start" class="btn btn-primary
     btn-sm">
121                                 Iniciar/Reiniciar
122                             </button>
123                         </div>
124                         <div class="col">
125                             <button id="btn_pause" class="btn btn-
    secondary btn-sm">
126                                 Reanudar/Pausa
127                             </button>
128                         </div>
129                         <div class="col">
130                             <button id="btn_next" class="btn btn-success
    btn-sm">
131                                 Siguiente Iteracion
132                             </button>
133                         </div>
134                     </div>
135                 </div>
136                 <div class="form-group">
137                     <div class="row">
138                         <div class="col">
139                             <button id="btn_show_matrix" class="btn btn-
    warning btn-sm">
140                                 Ver matriz auxiliar
141                             </button>
142                         </div>
143                         <div class="col">
144                             <label class="checkbox-inline">
145                                 <input type="checkbox" value="1" id="
    check-matrix">
146                                 Con matriz auxiliar
147                             </label>
148                         </div>
149                     </div>
150                 </div>
```

```
            </div>
        </div>
        <div class="row">
            <div class="col-8 custom-scroll">
                <canvas id="canvas-aux" width="1000" height="1000">
                </canvas>
            </div>
        </div>
    </div>
    <script type="text/javascript" src="./Chart.bundle.min.js">
    </script>
    <script type="text/javascript" src="./logica.js">
    </script>
</body>
</html>
```

Archivo: logica.js

Este archivo contiene toda la lógica para controlar la interfaz web y para poder llevar a cabo la simulación del autómata celular, el lenguaje de programación utilizado fue JavaScript y se utilizo la librería *Chart.js* para graficar la cantidad de unos.

```
let CANVAS_SIZE = 1000;
let TYPE_FUNCTION_DICT = {
    "MODA": 1,
    "MINIMO": 2,
    "PARIDAD": 3
};
let c = document.getElementById("canvas");
let c_aux = document.getElementById("canvas-aux");
let information_output = document.getElementById("information");
let slider_distribution = document.getElementById("distribution");
let slider_output = document.getElementById("distribution_value");
let context = document.getElementById("myChart");
let ctx = c.getContext("2d");
let ctx_aux = c_aux.getContext("2d");
let size = 100;
let total_population = size*size;
let dimension = 10;
let rule = [2, 3, 3, 3];
let counter = 0;
let my_time = 0;
let suma = 0;
let interval = null;
let is_runninng = false;
let is_active = true;
let original = [];
let auxiliar = [];
let ones_distribution = 0.5;
let tau = 3;
let colors = ["#FFFFFF", "#000000"];
let type_function = TYPE_FUNCTION_DICT["MODA"];
```

```
let myLineChart = create_chart(context);
let matrices;
let is_auxiliar_showed = false;

ctx.lineWidth = "0.1";
slider_output.innerHTML = slider_distribution.value;
slider_distribution.oninput = function(e) {
    e.preventDefault();
    slider_output.innerHTML = this.value;
}

function create_chart(_context) {
    return new Chart(_context, {
        type: 'line',
        data: {
            labels: [],
            datasets: [{
                label: "Cantidad de unos",
                data: [],
                borderColor: "#c45850",
                fill: false,
                lineTension: 0,
                defaultFontSize: 30
            }]
        },
        options: {
            title: {
                display: true,
                text: 'Historial de unos',
            },
            legends: {
                labels: {
                    defaultFontSize: 30
                }
            },
            defaultFontSize: 30,
        },
    });
}

function get_zero_or_one(dis) {
    if (Math.random() < dis)
        return 1;
    else
        return 0;
}

function draw_matrix(size, dimension, matrix) {
    for (let i = 0; i < size; i++)
        for (let j = 0; j < size; j++){
            if (matrix[i][j] == 1)
                ctx.fillStyle = colors[1];
```

```
            else
                ctx.fillStyle = colors[0];
            ctx.fillRect(dimension*j, dimension*i, dimension, dimension);
        }
    ctx.stroke();
}

function draw_aux_matrix(size, dimension, matrix) {
    for (let i = 0; i < size; i++)
        for (let j = 0; j < size; j++){
            if (matrix[i][j] == 1)
                ctx_aux.fillStyle = colors[1];
            else
                ctx_aux.fillStyle = colors[0];
            ctx_aux.fillRect(dimension*j, dimension*i, dimension,
   dimension);
        }
    ctx_aux.stroke();
}

function draw_grid(size, dimension) {
    for (let i = 0; i < size; i++)
        for (let j = 0; j < size; j++)
            ctx.rect(dimension*j, dimension*i, dimension, dimension);
    ctx.stroke()
}

function check_neighbours(i, j, matrix, m_size) {
    let row = i-1;
    let col = j-1;
    if (row < 0)
        row = m_size-1;
    if (col < 0)
        col = m_size-1;
    let neighbours = matrix[row][col];
    neighbours += matrix[row][j];
    neighbours += matrix[row][(j + 1) % m_size];
    neighbours += matrix[i][(j + 1) % m_size];
    neighbours += matrix[(i + 1) % m_size][(j + 1) % m_size];
    neighbours += matrix[(i + 1) % m_size][j];
    neighbours += matrix[(i + 1) % m_size][col];
    neighbours += matrix[i][col];

    return neighbours;
}

function next_population(size, matrix, aux) {
    let vecinos = 0;
    let aux_counter = 0;
    for (let i = 0; i < size; i++)
        for (let j = 0; j < size; j++) {
            vecinos = check_neighbours(i, j, aux, size);
```

```
            if (aux[i][j] == 1) {
                if (vecinos < rule[0] || vecinos > rule[1]){
                    matrix[i][j] = 0;
                    //aux_counter--;
                }
            }
            else if (rule[2] <= vecinos && vecinos <= rule[3]) {
                matrix[i][j] = 1;
                //aux_counter++;
            }
            if (matrix[i][j] == 1)
                aux_counter++;
        }

    return aux_counter;
}

function copy_matrix(origen, destino) {
    for (let i = 0; i < origen.length; i++)
        for (let j = 0; j < origen.length; j++)
            destino[i][j] = origen[i][j];
}

function init() {
    size = get_size_matrix();
    total_population = size*size;
    dimension = 1;
    rule = get_rule();
    counter = 0;
    my_time = 0;
    suma = 0;
    interval = null;
    is_runninng = false;
    original = [];
    auxiliar = [];
    other_matrix = [];
    tau = get_tau();
    ones_distribution = get_distribution();
    colors = get_colors();
    type_function = get_type_function();
    myLineChart = create_chart(context);
    is_active = get_check_matrix();
    matrices = {1:[], 2:[], 3:[], 4:[], 5:[], 6:[], 7:[], 8:[]};

    while (dimension*size < CANVAS_SIZE) dimension++;

    for(let i=0; i<size; i++){
        original[i] = [];
        auxiliar[i] = [];
        for (let k = 1; k <= tau; k++)
            matrices[k][i] = [];
        for(let j=0; j<size; j++){
```

```
            original[i][j] = get_zero_or_one(ones_distribution);
            auxiliar[i][j] = original[i][j];
            if (auxiliar[i][j] == 1)
                counter++;
            for (let k = 1; k <= tau; k++)
                matrices[k][i][j] = 0;
        }
    }
}

function plot() {
    suma += counter;
    let promedio = suma/(my_time+1);
    promedio = promedio.toFixed(3);
    let densidad = promedio /(total_population);
    densidad = densidad.toFixed(3);
    information_output.innerHTML = "Promedio: " + promedio + ". Densidad:
    " + densidad;
    myLineChart.data.labels.push(my_time);
    myLineChart.data.datasets[0].data.push(counter);
    myLineChart.update();
}

function iterate() {
    my_time++;
    counter = next_population(size, original, auxiliar);
    copy_matrix(original, auxiliar);
    draw_matrix(size, dimension, original);
    plot();
}

function get_size_matrix() {
    return parseInt(document.getElementById("size").value);
}

function get_rule() {
    let aux_rule = [2, 3, 3, 3];
    aux_rule[0] = parseInt(document.getElementById("b1").value);
    aux_rule[1] = parseInt(document.getElementById("b2").value);
    aux_rule[2] = parseInt(document.getElementById("s1").value);
    aux_rule[3] = parseInt(document.getElementById("s2").value);
    return aux_rule;
}

function get_type_function() {
    return parseInt(document.getElementById("input_function").value);
}

function get_tau() {
    return parseInt(document.getElementById("tau").value);
}

```

```
function get_colors() {
    let aux_colors = ["#FFFFFF", "#000000"]
    aux_colors[0] = document.getElementById("color_zeros").value;
    aux_colors[1] = document.getElementById("color_ones").value;
    return aux_colors;
}

function get_distribution() {
    let aux_dis = parseInt(slider_distribution.value)/100;
    aux_dis = aux_dis.toFixed(2);
    return parseFloat(aux_dis);
}

function apply_paridad(temporal) {
    let aux_paridad = 0;
    for (let i = 0; i < tau; i++)
        aux_paridad += temporal[i];
    if (aux_paridad %2 == 0) return 1
    else return 0
}

function apply_moda(temporal) {
    let aux_moda = - (tau/2);
    for (let i = 0; i < tau; i++)
        aux_moda += temporal[i];
    if (aux_moda > 0) return 1
    else return 0
}

function apply_minimo(temporal) {
    if (apply_moda(temporal) == 0)
        return 1;
    else
        return 0;
}

function get_check_matrix() {
    return document.getElementById("check-matrix").checked;
}

let contador = 1;
function another_iteration() {
     if (is_active) {
        if (contador == 1)
            copy_matrix(original, matrices[1]);
        else if (contador == 2)
            copy_matrix(original, matrices[2]);
        else if (contador == 3)
            copy_matrix(original, matrices[3]);
        else if (contador == 4)
            copy_matrix(original, matrices[4]);
        else if (contador == 5)
```

```
289             copy_matrix(original, matrices[5]);
290         else if (contador == 6)
291             copy_matrix(original, matrices[6]);
292         else if (contador == 7)
293             copy_matrix(original, matrices[7]);
294         else if (contador == 8)
295             copy_matrix(original, matrices[8]);
296 
297         contador++;
298         if (contador > tau)
299             contador = 1;
300     }
301     let aux_res = 0;
302     if (is_active) {
303         if (((my_time+1) % tau) == 0 && my_time > 0) {
304             console.log("APLICANDO");
305             for (let i = 0; i < size; i++){
306                 for (let j = 0; j < size; j++){
307                     let arreglo = [];
308                     for (let k = 1; k <= tau; k++)
309                         arreglo.push(matrices[k][i][j])
310                     if (type_function == TYPE_FUNCTION_DICT["MODA"])
311                         auxiliar[i][j] = apply_moda(arreglo);
312                     else if (type_function == TYPE_FUNCTION_DICT["PARIDAD
    "])
313                         auxiliar[i][j] = apply_paridad(arreglo);
314                     else
315                         auxiliar[i][j] = apply_minimo(arreglo);
316                 }
317             }
318             if (is_auxiliar_showed)
319                 draw_aux_matrix(size, dimension, auxiliar);
320         }
321     }
322     iterate();
323 }
324 
325 document.getElementById("btn_start").addEventListener("click", function
    click(e) {
326     e.preventDefault();
327     init();
328     plot();
329     draw_grid(size, dimension);
330     draw_matrix(size, dimension, original);
331 });
332 
333 document.getElementById("check-matrix").addEventListener("click",
    function click(e) {
334     is_active = get_check_matrix();
335 });
336 
337 document.getElementById("btn_show_matrix").addEventListener("click",
```

```
    function click(e) {
338     is_auxiliar_showed = !is_auxiliar_showed;
339     if (!is_auxiliar_showed)
340         ctx_aux.clearRect(0, 0, CANVAS_SIZE, CANVAS_SIZE);
341 });
342
343 document.getElementById("btn_next").addEventListener("click", function
    click(e) {
344     e.preventDefault();
345     another_iteration();
346 });
347
348 document.getElementById("btn_pause").addEventListener("click", function
    click(e) {
349     e.preventDefault();
350     console.log("PAUSE");
351     is_runninng = !is_runninng;
352     if (is_runninng)
353         interval = setInterval(another_iteration, 250);
354     else {
355         clearInterval(interval);
356         interval = null;
357     }
358 });
359
360 c.addEventListener("click", function click(e) {
361     let my_y = e.pageY;
362     let my_x = e.pageX - 15;
363     let i = Math.floor(my_y/dimension);
364     let j = Math.floor(my_x/dimension);
365     if (original[i][j] == 1){
366         original[i][j] = 0;
367         auxiliar[i][j] = 0;
368         ctx.fillStyle = colors[0];
369     } else{
370         original[i][j] = 1
371         auxiliar[i][j] = 1;
372         ctx.fillStyle = colors[1];
373     }
374     ctx.fillRect(dimension*j, dimension*i, dimension, dimension);
375 });
```

1.4. Pruebas

Para probar el funcionamiento del programa se utilizaron diferentes reglas con diferentes densidades de población para poder observar su comportamiento. Es importante señalar que todas las pruebas se hicieron con un $\tau = 4$ y con diferentes funciones Φ además de solo iterar al rededor de 100 veces.

1.5. Regla: 2 7 4 6. Densidad: 20 %

Como se puede observar en la figura 1 esta regla termina por llenar la mitad del espacio con una malla de unos y ceros.

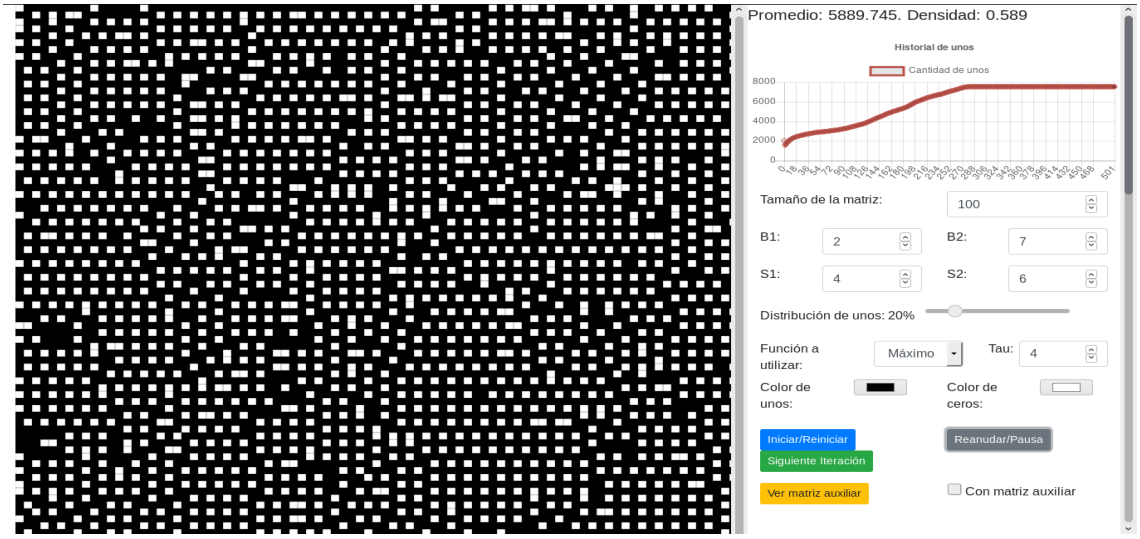

Figura 1: Resultado tras 100 iteraciones sin matriz auxiliar

Al aplicar la función de máximo no se llena todo el espacio y se estanca en una cantidad de unos.

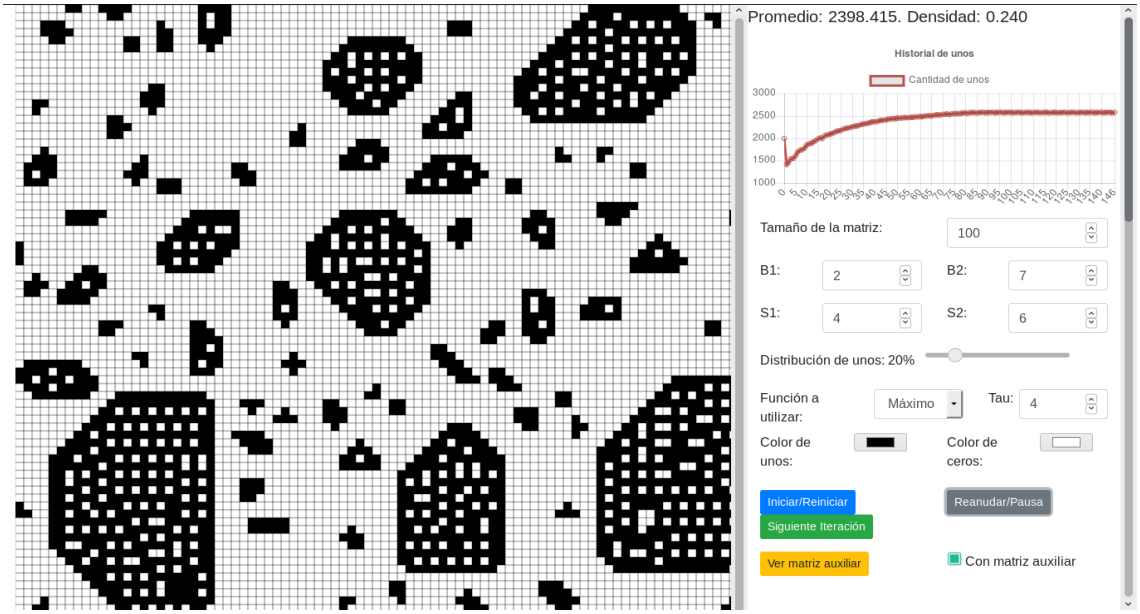

Figura 2: Utilizando la función de máximo

Es importante mencionar que el resultado obtenido al aplicar la función mínimo es muy similar al de máximo pero la matriz auxiliar se invierte.

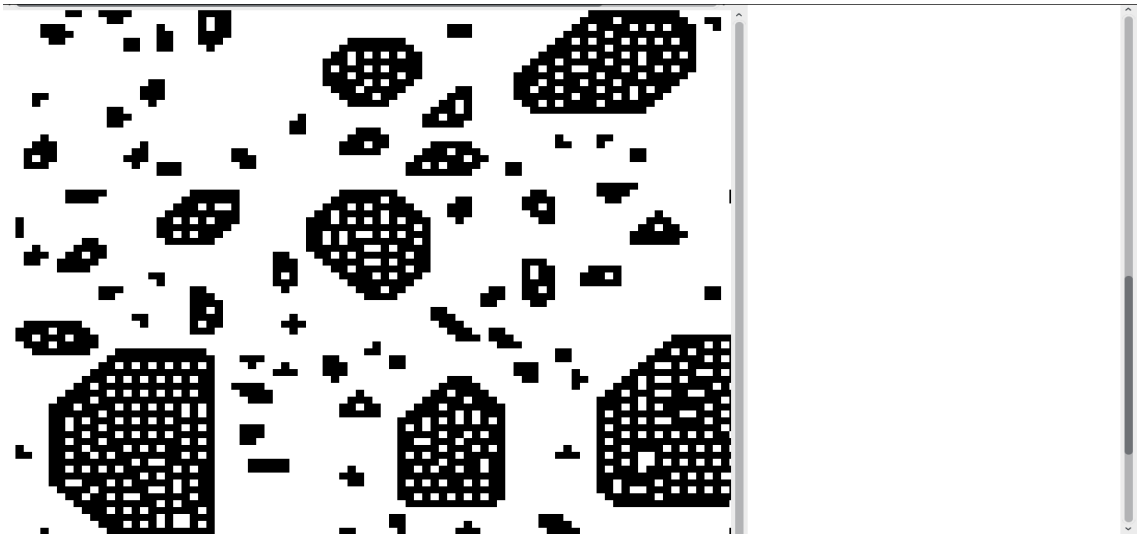

Figura 3: Matriz auxiliar del autómata anterior

La figura 4 es el resultado de utilizar la función de paridad en la cual la población decrece bastante rápido hasta quedarse en un solo patrón.

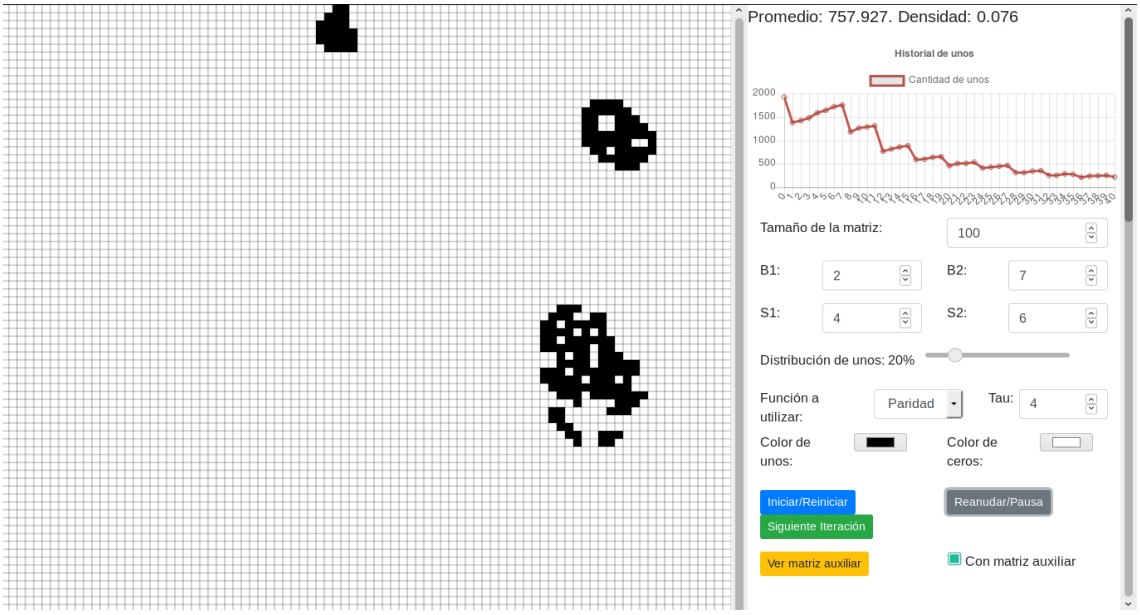

Figura 4: Utilizando la función de paridad

El comportamiento del automata anterior se puede ver reflejado en la matriz auxiliar que se muestra en la figura 5.

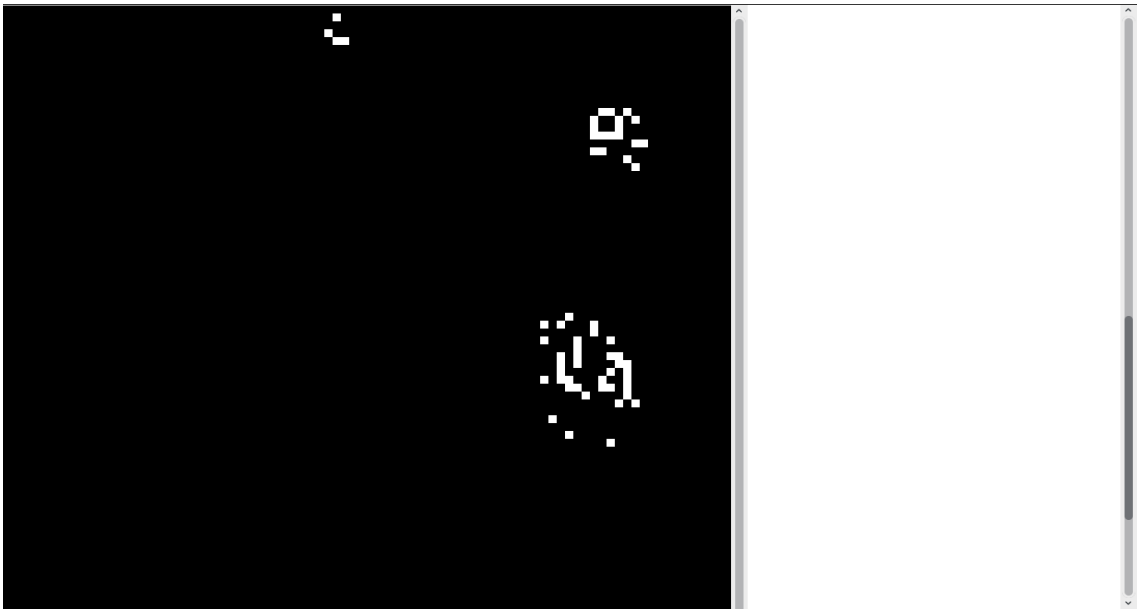

Figura 5: Matriz auxiliar del autómata anterior

1.6. Regla: 3 6 3 4. Densidad: 20 %

El comportamiento de esta regla es similar al anterior pero con una curva de crecimiento más suave.

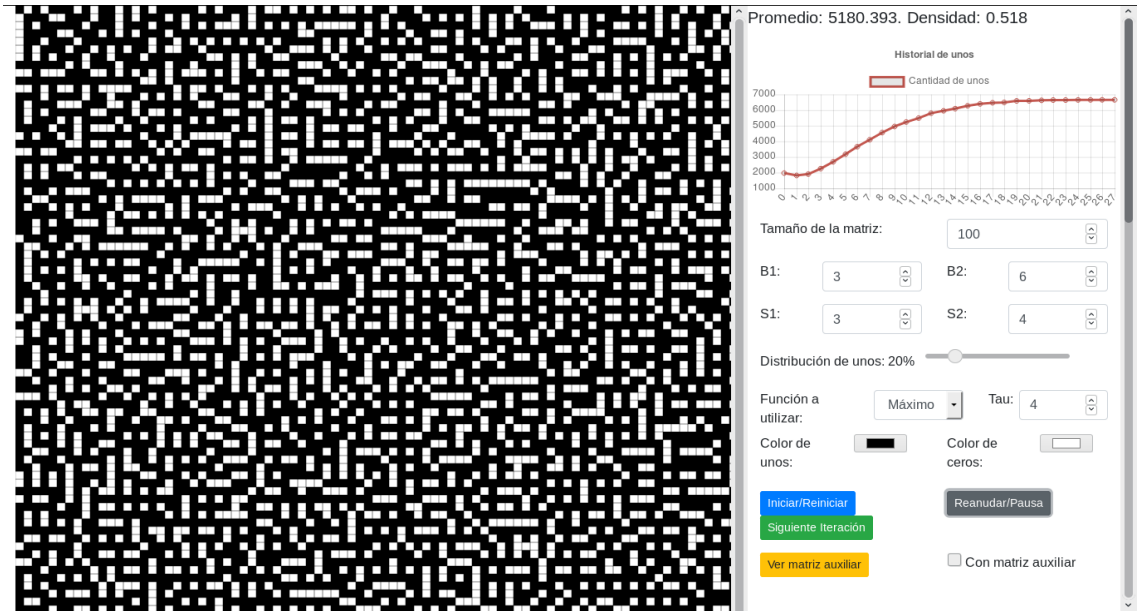

Figura 6: Resultado tras 100 iteraciones sin matriz auxiliar

Al utilizar la función de mínimo que se observa en la figura 7 se llega al mismo resultado pero con algunas perturbaciones al inicio de su crecimiento.

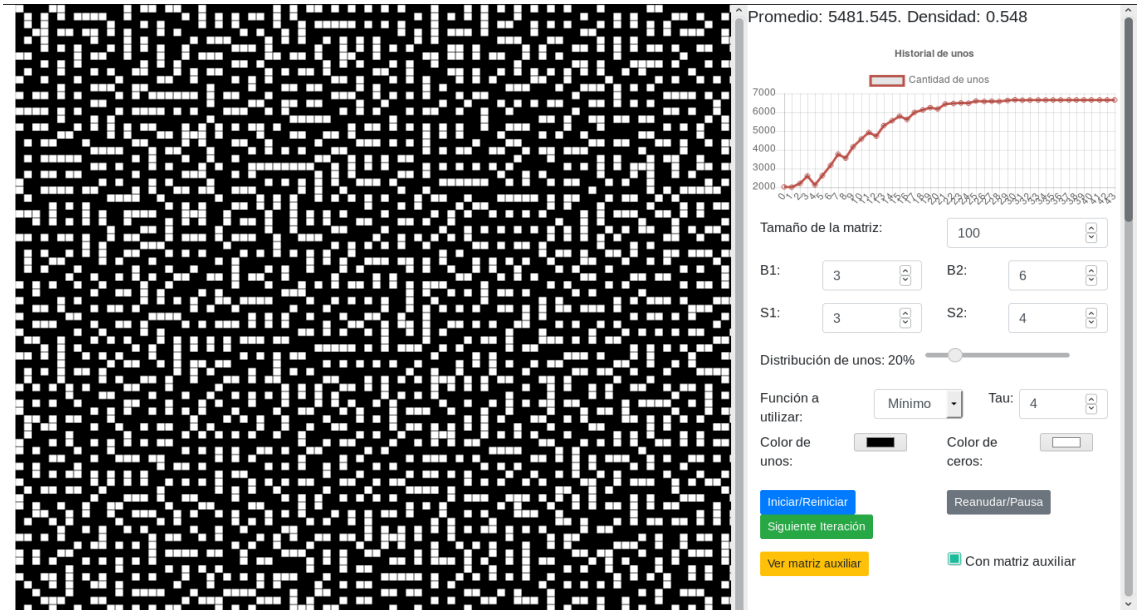

Figura 7: Utilizando la función de mínimo

Si se utiliza una función de máximo se obtiene un resultado similar pero invirtiendo

los colores de la matriz auxiliar que se observa en la figura 8.

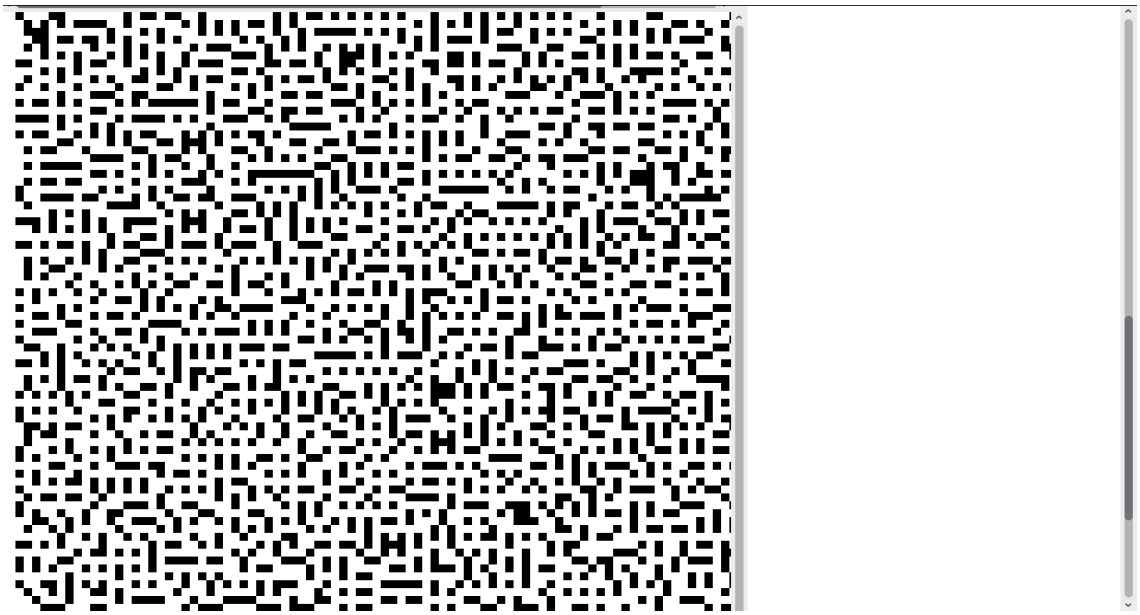

Figura 8: Matriz auxiliar del autómata anterior

La función de paridad es la que provoca el comportamiento más inestable ya que a pesar de que crece como en los ejemplos anteriores su población oscila a lo largo de un rango de valores

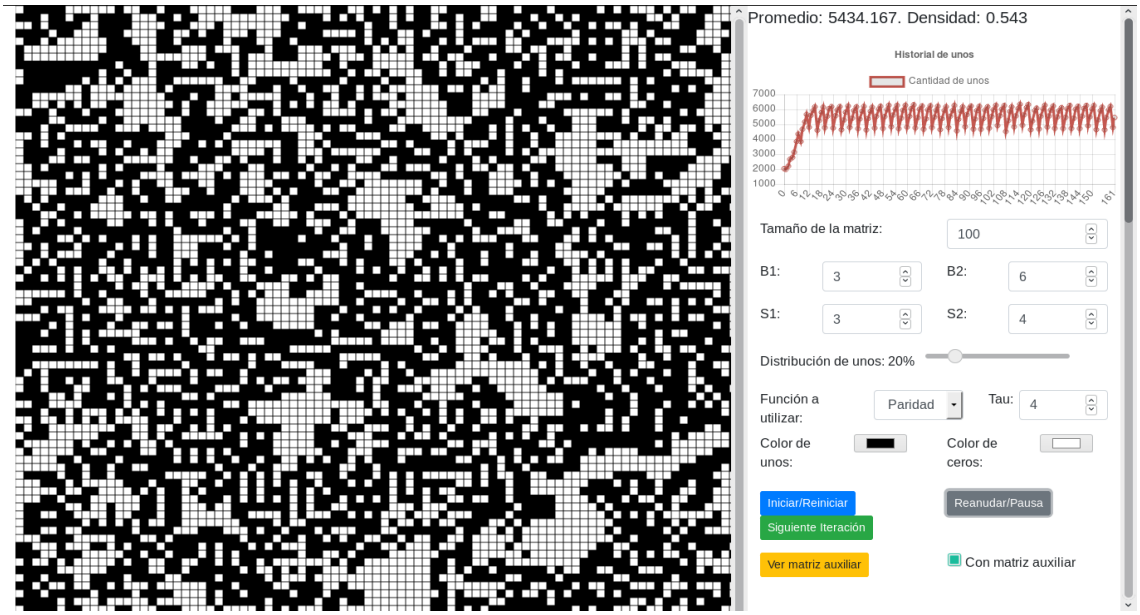

Figura 9: Utilizando la función de paridad

El comportamiento oscilatorio se observa en la matriz auxiliar ya que esta parece solo cambiar de colores una y otra vez.

Figura 10: Matriz auxiliar del autómata anterior

1.7. Regla: 1 6 1 6. Densidad: 10 %

El comportamiento de esta regla se tiende a estabilizar bastante rápido y formar una estructura que ya no cambia.

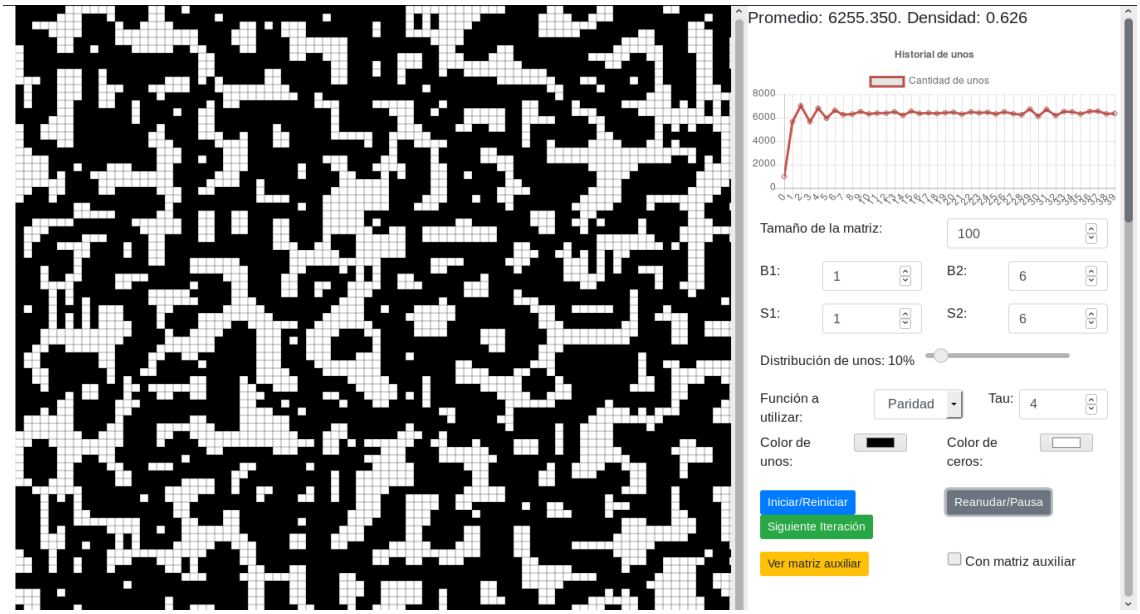

Figura 11: Resultado tras 100 iteraciones sin matriz auxiliar

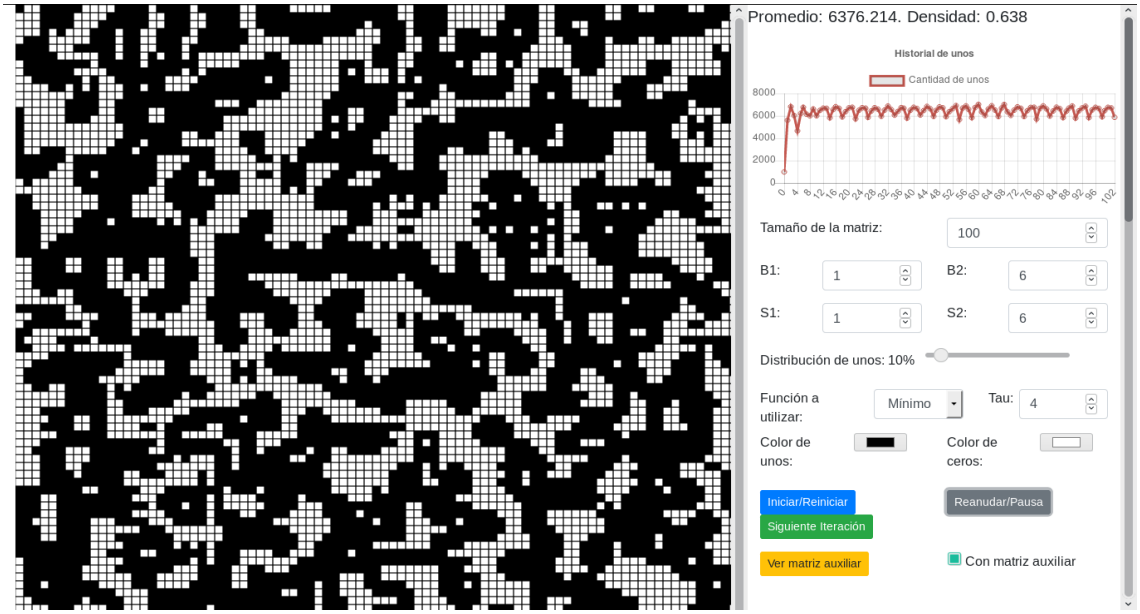

Figura 12: Utilizando la función de mínimo

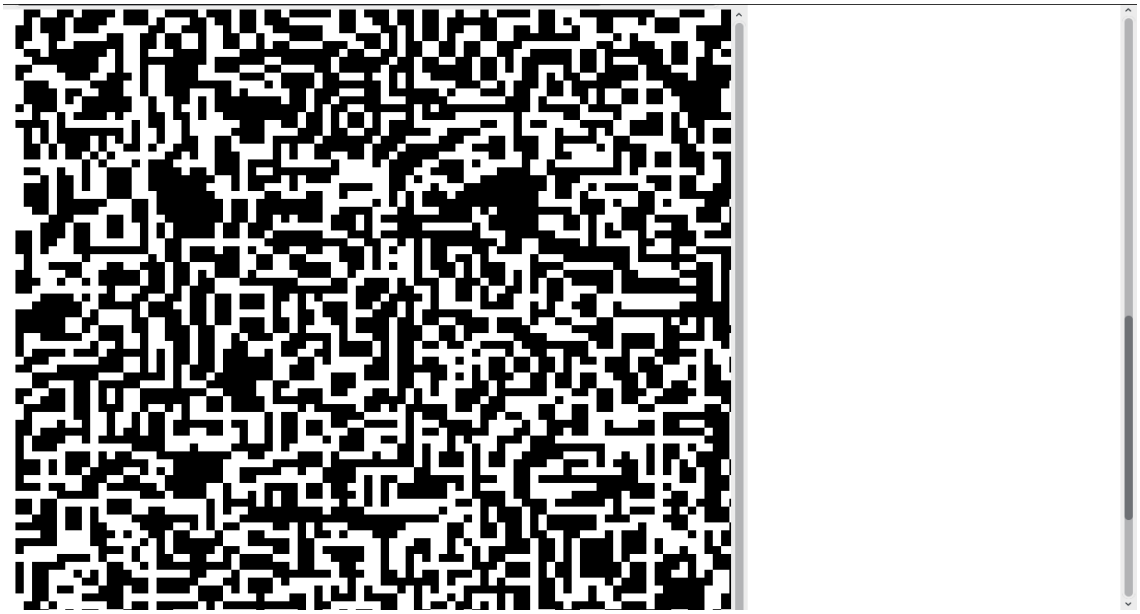

Figura 13: Matriz auxiliar del autómata anterior

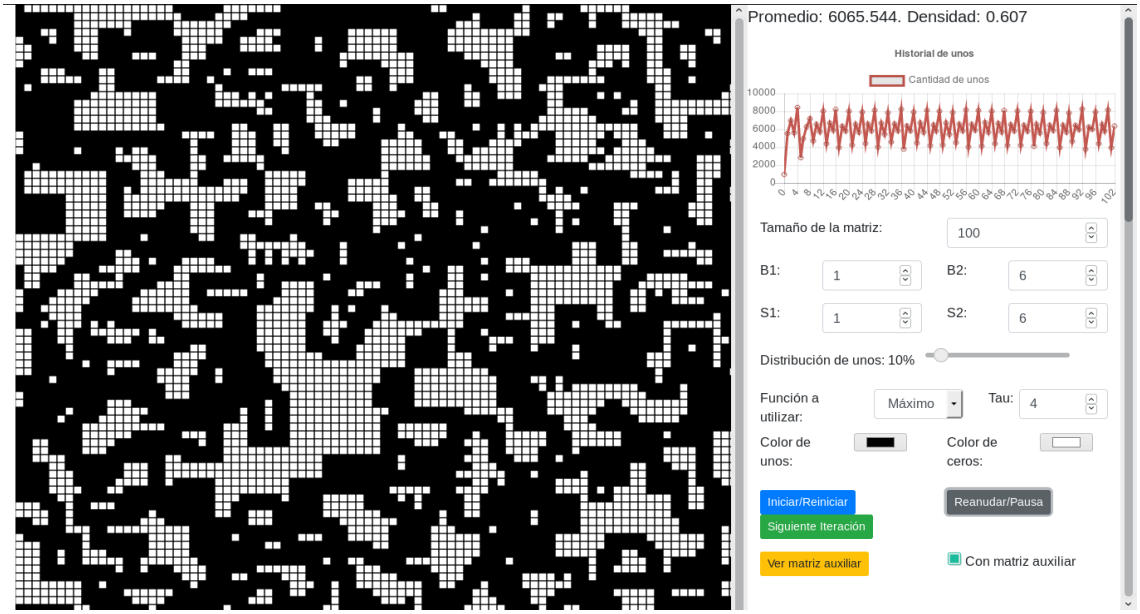

Figura 14: Utilizando la función de máximo

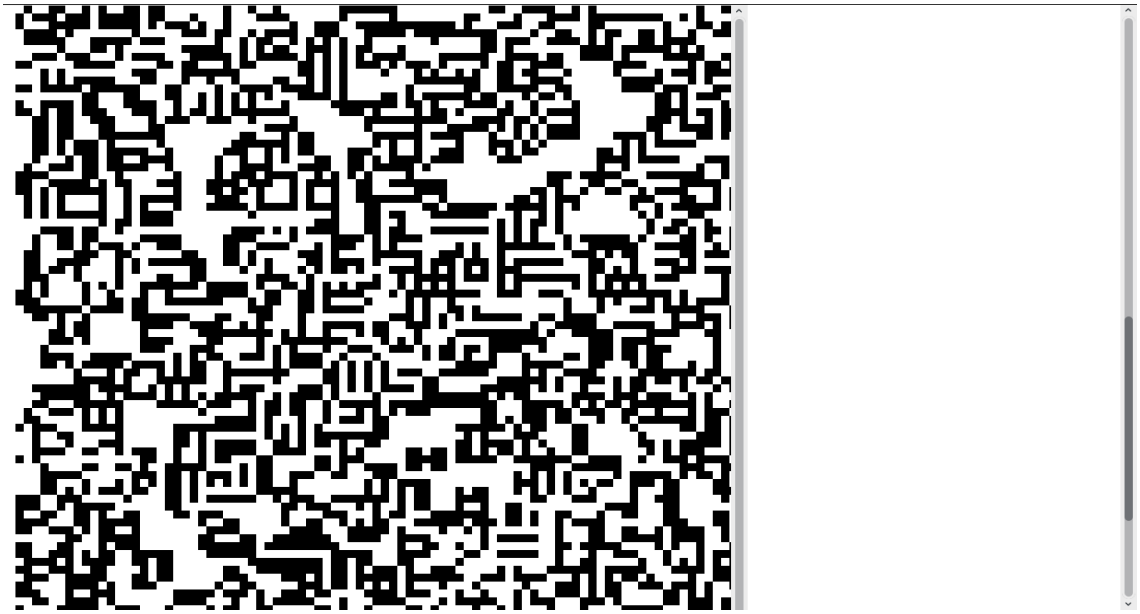

Figura 15: Matriz auxiliar del autómata anterior

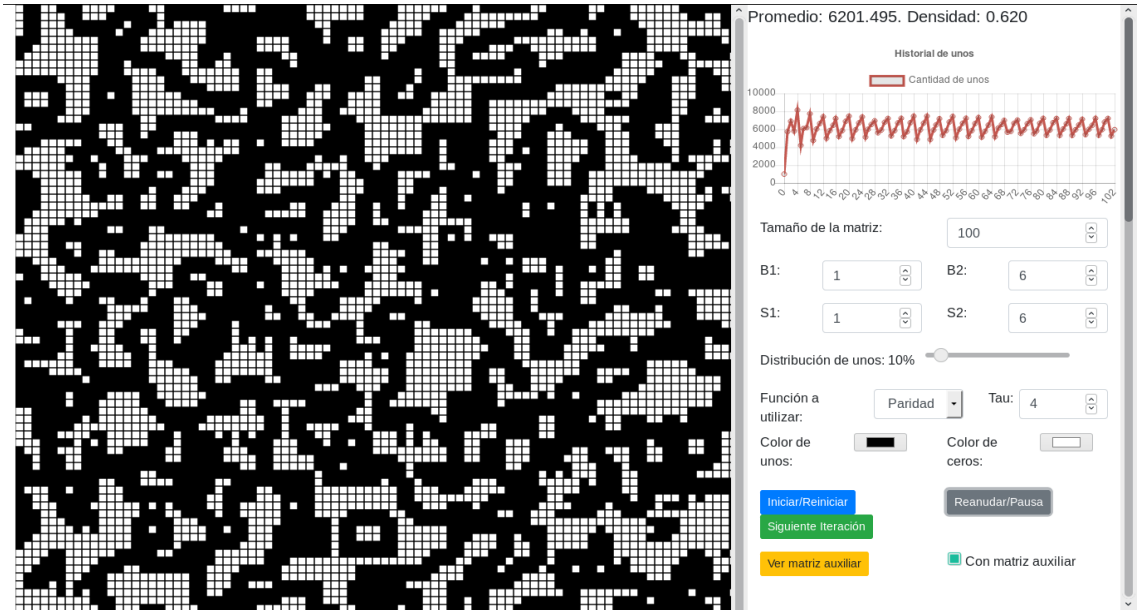

Figura 16: Utilizando la función de paridad

Figura 17: Matriz auxiliar del autómata anterior

Con esta regla y utilizando las tres funciones se obtiene el mismo resultado, sin embargo, el comportamiento de la cantidad de unos es diferente. Para la función de máximo se tiene el comportamiento más oscilatorio, después le sigue la función de paridad y por ultimo la función de mínimo.

1.8. Regla: 3 3 1 8. Densidad: 10 %

El comportamiento de esta regla oscila bastante y llega a un punto en el que parece que solo cambia los colores de las células.

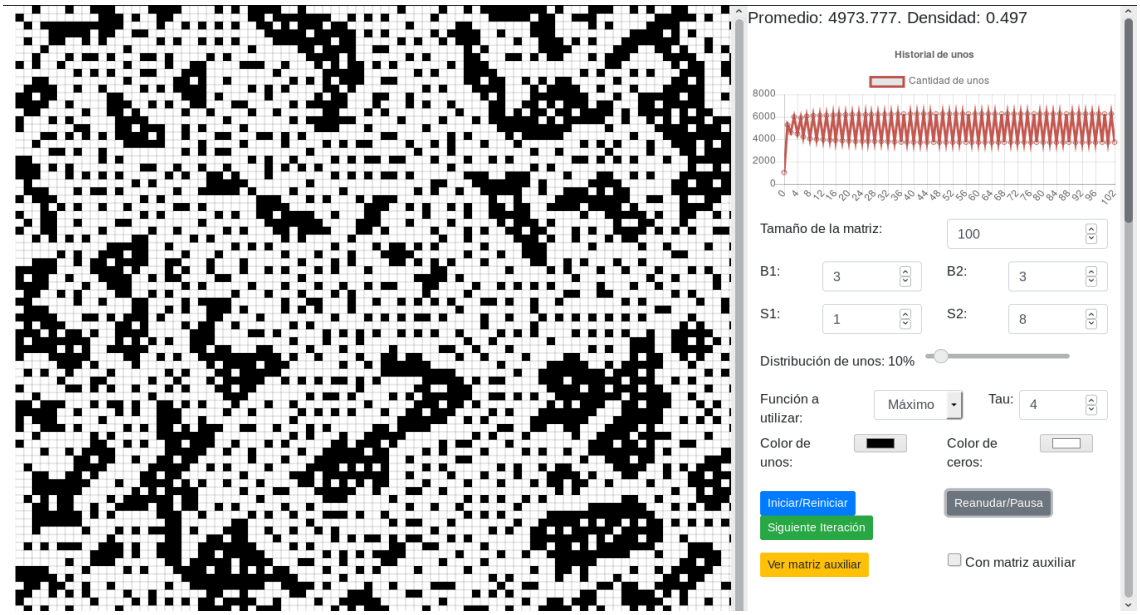

Figura 18: Resultado tras 100 iteraciones sin matriz auxiliar

Para la regla de mínimo se tiene un comportamiento extraño en el cual en lugar de aumentar la población y estabilizarse ocurre lo contrario hasta quedar con solo algunos patrones en la matriz.

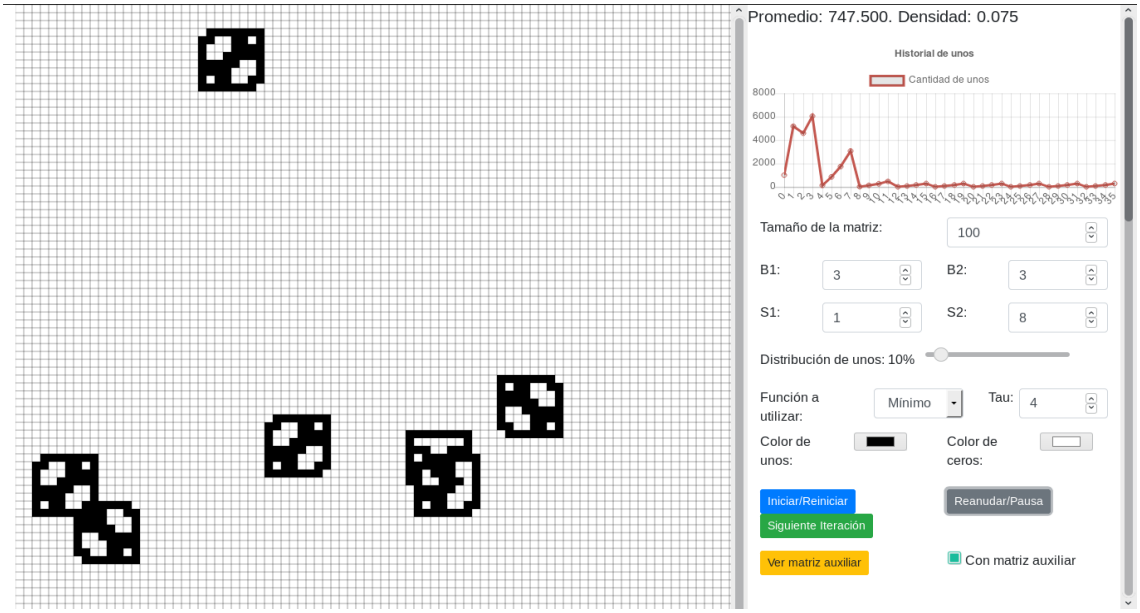

Figura 19: Utilizando la función de mínimo

Figura 20: Matriz auxiliar del autómata anterior

Al aplicar la función máximo se reduce la cantidad de oscilaciones que se tienen a lo largo de las iteraciones.

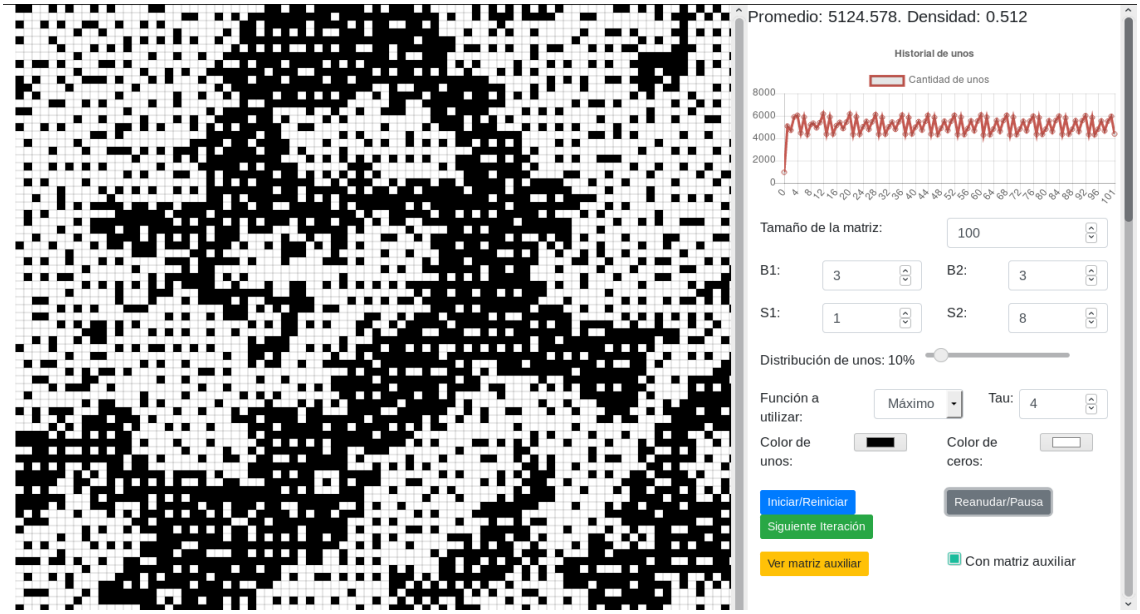

Figura 21: Utilizando la función de máximo

Figura 22: Matriz auxiliar del autómata anterior

En contraste con la función máximo se tiene lo contrario, en esta las oscilaciones son mayores.

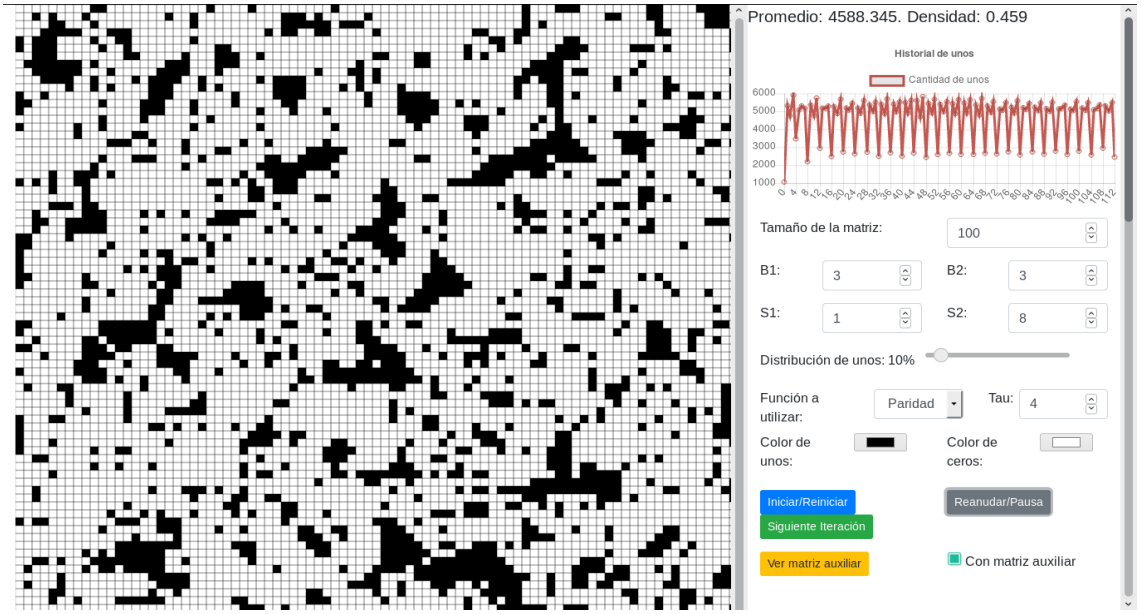

Figura 23: Utilizando la función de paridad

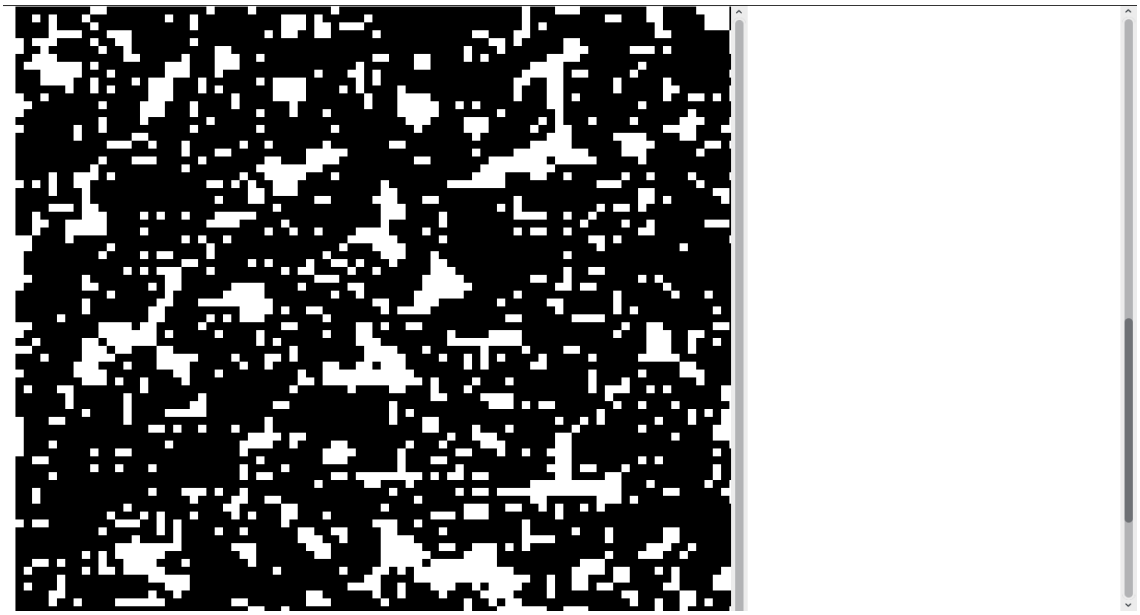

Figura 24: Matriz auxiliar del autómata anterior

1.9. Regla: 3 3 1 7. Densidad: 5 %

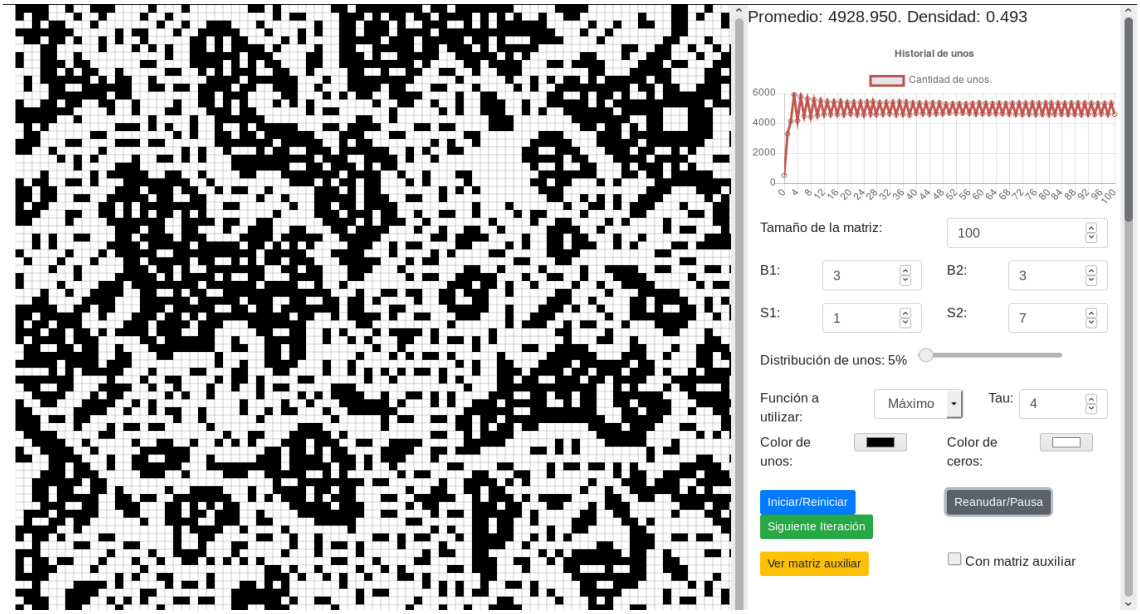

Figura 25: Resultado tras 100 iteraciones sin matriz auxiliar

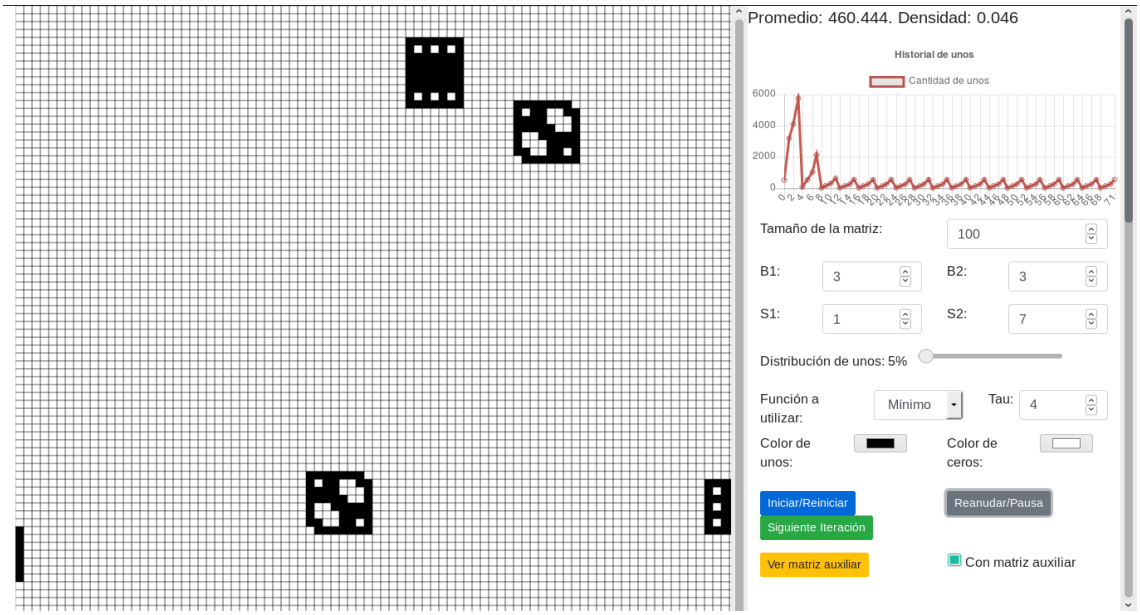

Figura 26: Utilizando la función de mínimo

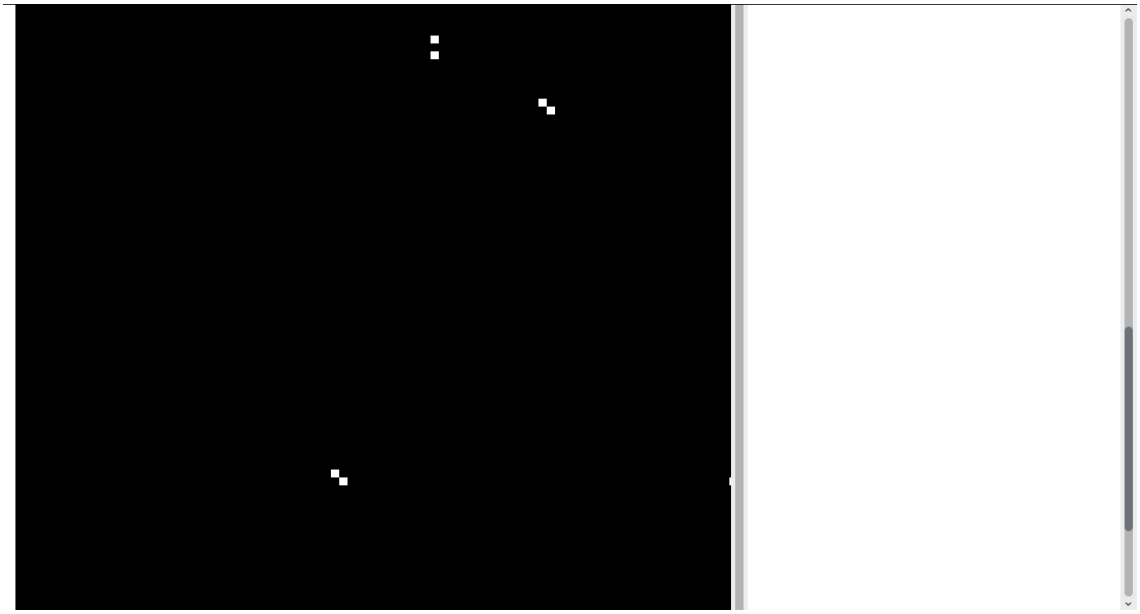

Figura 27: Matriz auxiliar del autómata anterior

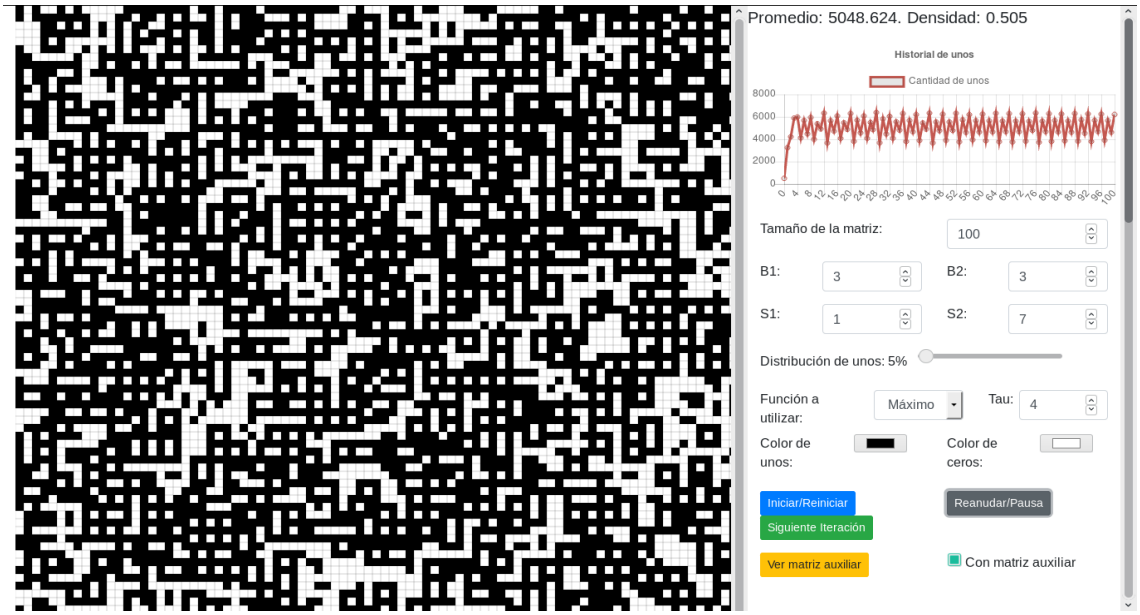

Figura 28: Utilizando la función de máximo

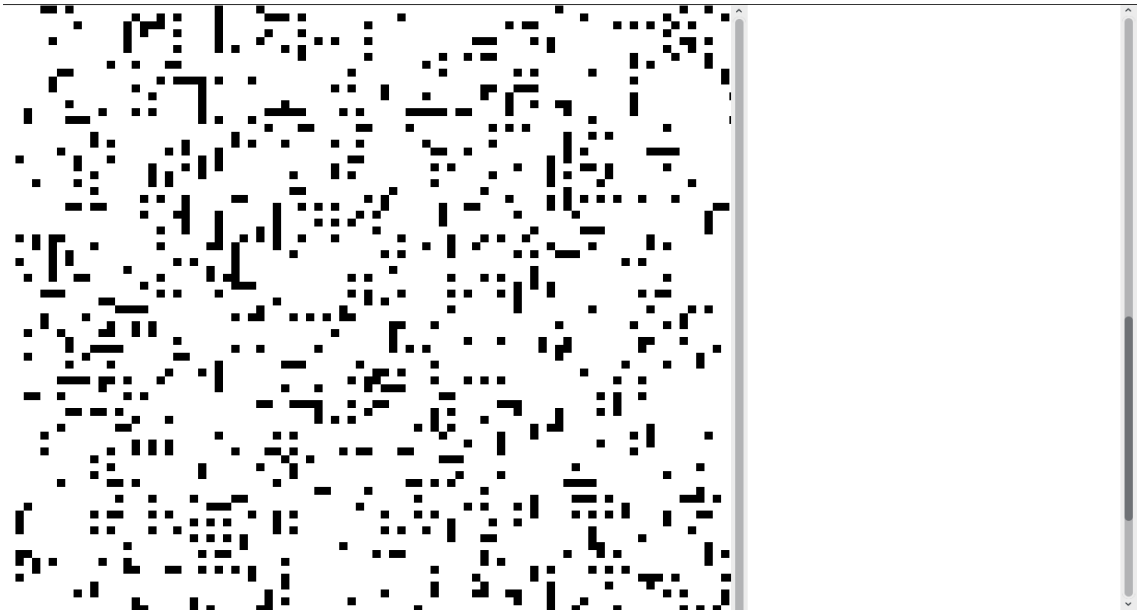

Figura 29: Matriz auxiliar del autómata anterior

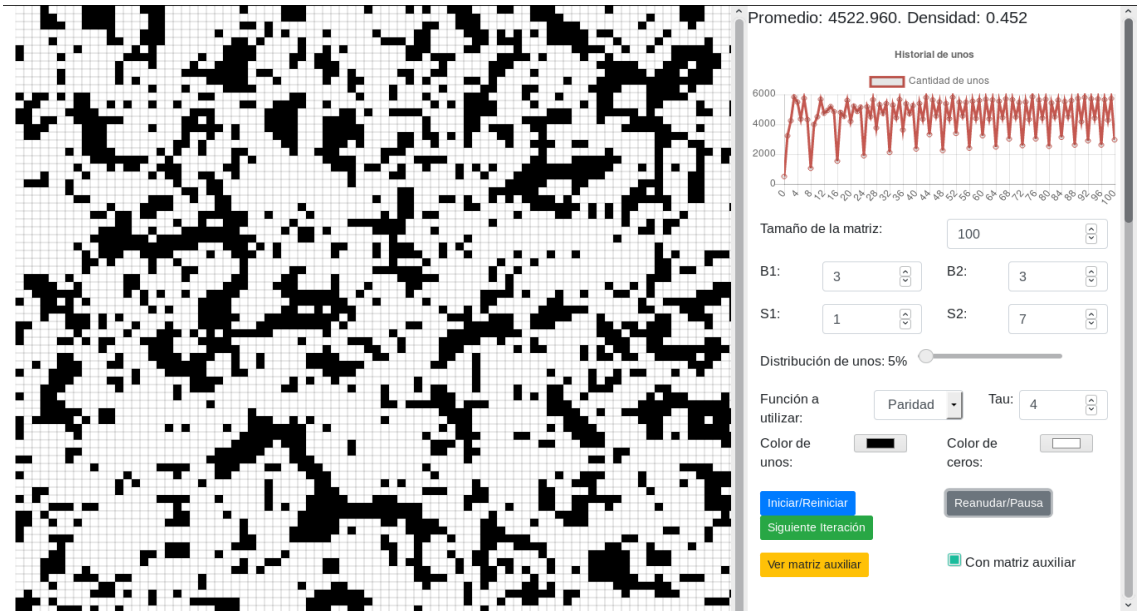

Figura 30: Utilizando la función de paridad

Figura 31: Matriz auxiliar del autómata anterior

Con esta regla se tiene el comportamiento más raro al aplicar la regla de mínimo ya que la población aumenta en un inicio pero al final termina disminuyendo demasiado y estancarse en unos patrones en especifico.

1.10. Regla: 2 3 3 6. Densidad: 10 %

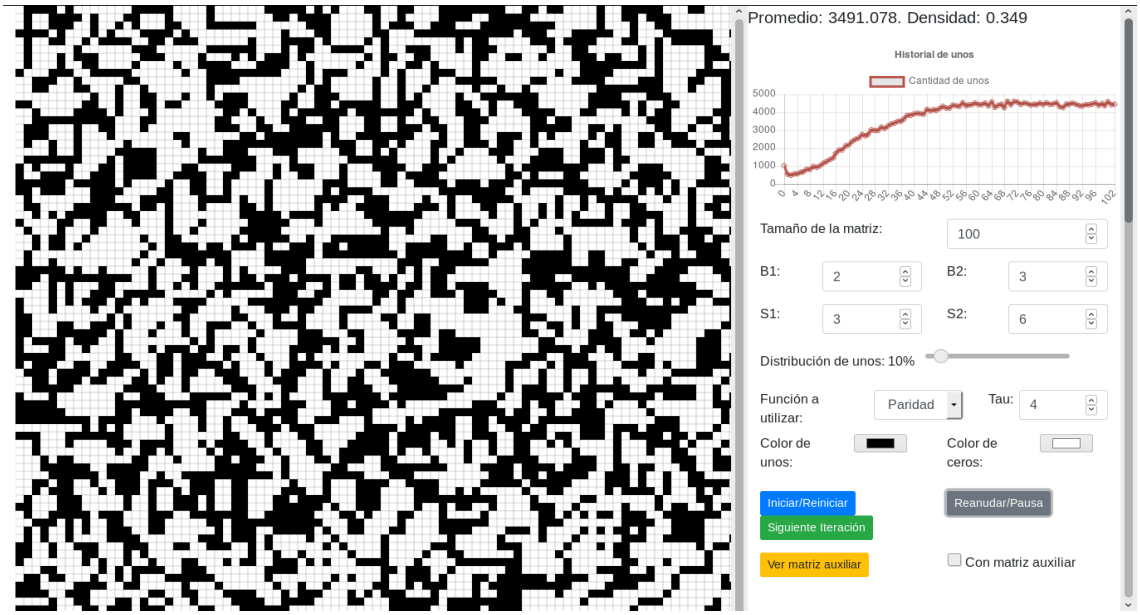

Figura 32: Resultado tras 100 iteraciones sin matriz auxiliar

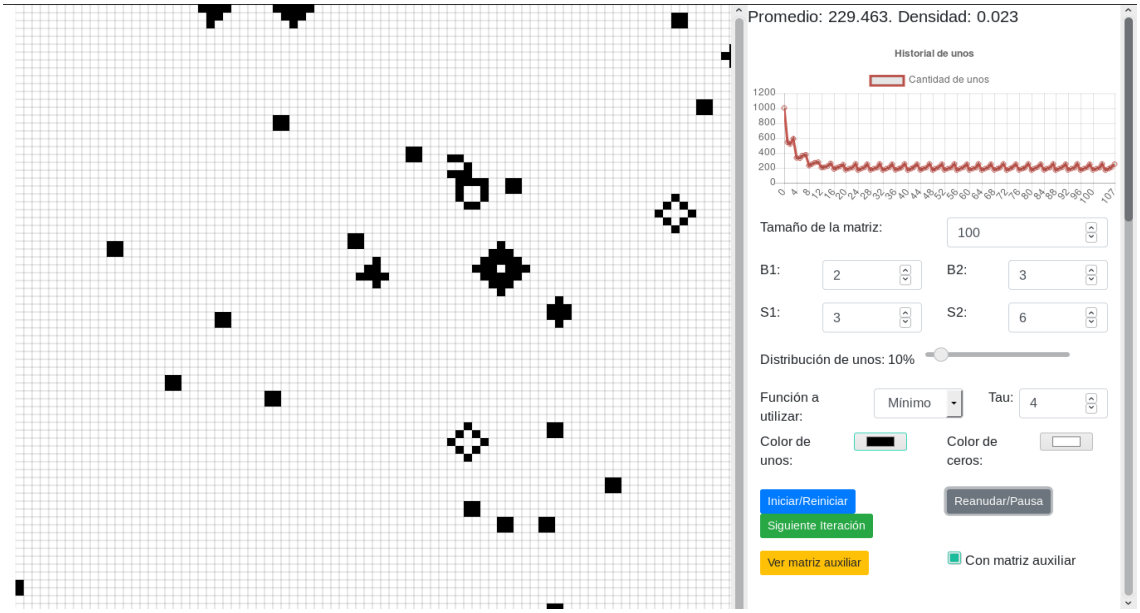

Figura 33: Utilizando la función de mínimo

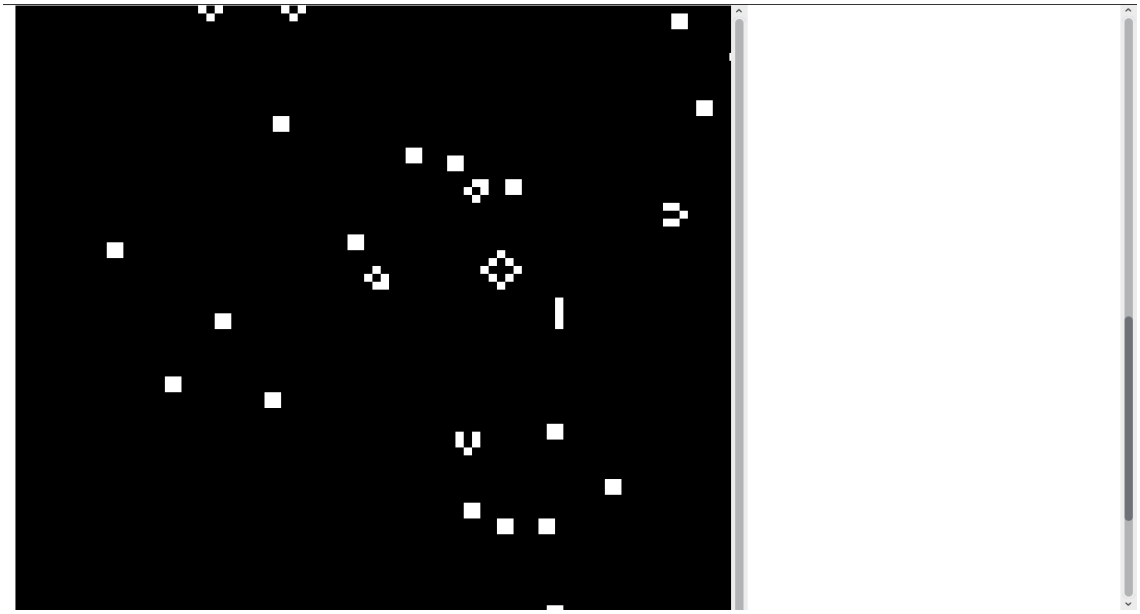

Figura 34: Matriz auxiliar del autómata anterior

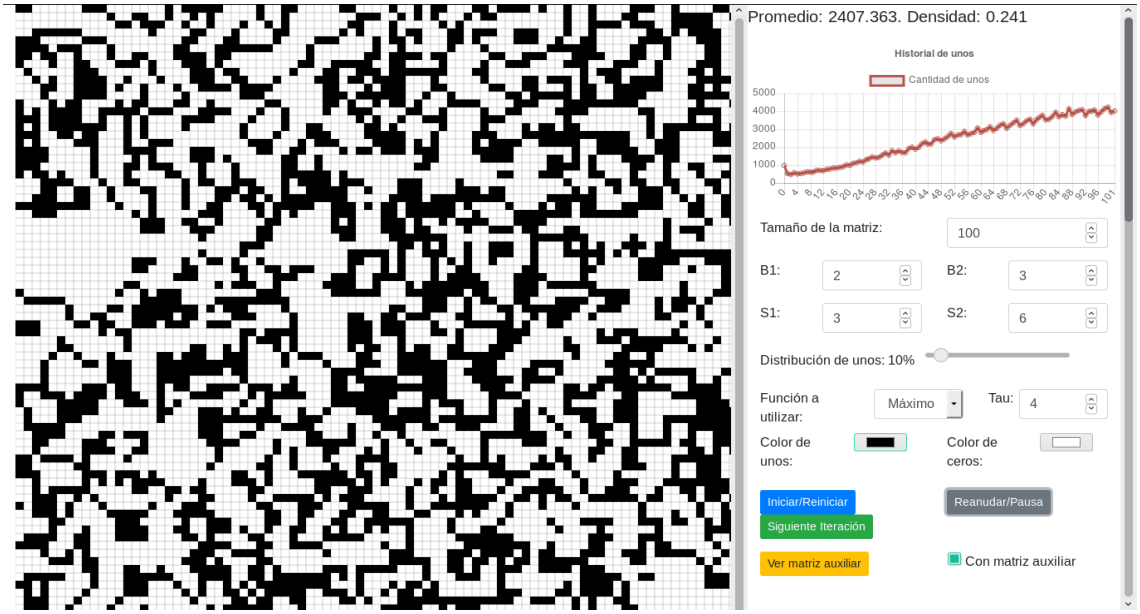

Figura 35: Utilizando la función de máximo

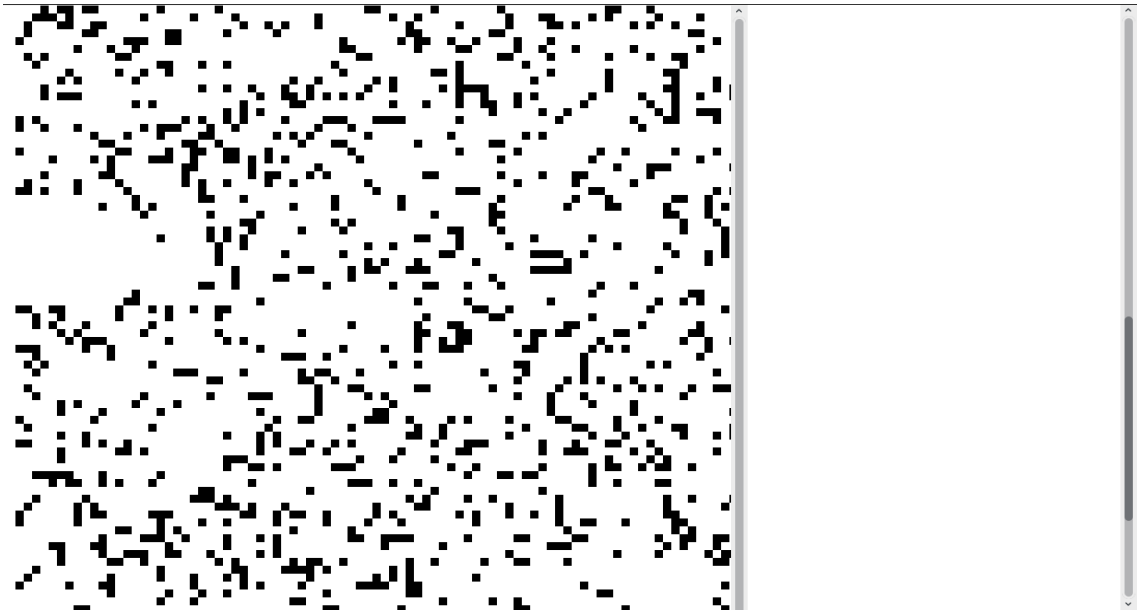

Figura 36: Matriz auxiliar del autómata anterior

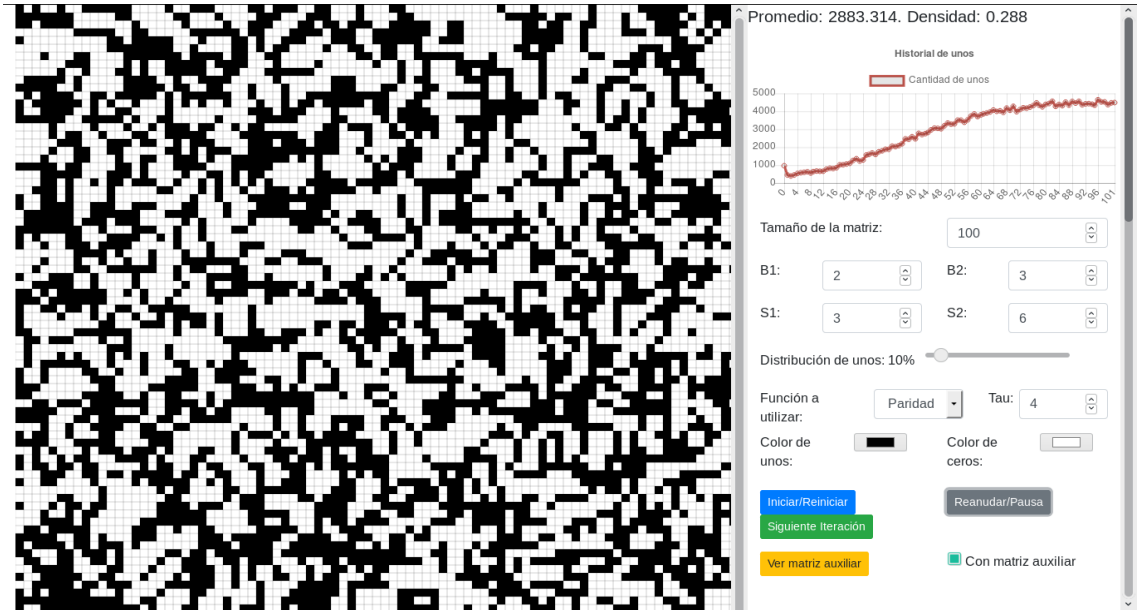

Figura 37: Utilizando la función de paridad

Figura 38: Matriz auxiliar del autómata anterior

Al igual que con la regla anterior, en esta regla se tiene el mismo comportamiento de disminución de la población al aplicar la regla de mínimo.

1.11. Conclusiones

El uso de una función y una matriz auxiliar para el calculo de las iteraciones en un autómata celular cambian el comportamiento de la función original, es decir, si se realiza la prueba y se compara la gráfica de unos de la regla del juego de la vida con y sin función auxiliar se puede apreciar que la que tiene la función auxiliar oscila con más frecuencia a diferencia de la que no utiliza una función extra, en esta la gráfica se estabiliza de una forma más rápida.

El uso de esta técnica al trabajar con autómatas celulares es bastante útil ya que permite el observar nuevos comportamientos con base a reglas ya conocidas, lo cual puede agilizar el estudio de estos modelos.

Referencias

[1] Reyes Gómez, D., *Descripción y Aplicaciones de los Autómatas Celulares*. cinvestav, 2011.