



TRABAJO FIN DE GRADO

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS

Juegos de vida: simulación y caracterización

Autor

Daniel Jiménez López

Tutores

Antonio Miguel Lallena Rojo
Juan Antonio López Villanueva



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, 17 de junio de 2019

Juegos de vida: simulación y caracterización

Daniel Jiménez López

Palabras clave: Juego de vida, Juego de vida α -asíncrono, simulación Monte Carlo, autómatas celulares, autómatas celulares m -asíncrono.

Resumen

El juego de vida de Conway ha sido uno de los casos más estudiados desde que von Neumann y Ulam introdujeran el concepto de autómatas celulares. El conjunto de reglas que determinan la evolución de éste autómatas celulares sobre una red son muy simples, sin embargo dan lugar a comportamientos complejos y difíciles de predecir, que han atraído la atención de científicos de diversos campos. Desde entonces, a parte de descubrir y catalogar dichos comportamientos, también ha surgido interés en el estudio de cómo afectan las perturbaciones en la red y en la evolución. Además se han propuesto nuevos tipos de autómatas celulares similares al juego de vida pero que presentan comportamientos totalmente diferentes, mostrando que pequeñas alteraciones en las reglas pueden llevar a grandes cambios.

En este trabajo hemos explorado el juego de vida de Conway, mostrando especial interés en la clasificación de los comportamientos complejos que se dan en el mismo y cuál es su evolución cuando se exponen a perturbaciones. En particular nos hemos preguntado qué ocurre si las perturbaciones se introducen a nivel de aplicación de las reglas, es decir, qué es lo que ocurre cuando las reglas no se aplican sobre todos los individuos simultáneamente, si no en función de cierta probabilidad α . Esta modificación define los denominados autómatas α -asíncronos. Con el fin de definir un contexto que agrupe tanto esta idea como la del juego de vida original hemos tomado la definición de autómatas celulares m -asíncrono.

Dado el carácter probabilístico del juego de vida α -asíncrono, la herramienta escogida para estudiar el desarrollo de las configuraciones ha sido la técnica de simulación Monte Carlo. Éstas se apoyan principalmente en la generación de números pseudo-aleatorios de una determinada distribución y en dos teoremas fundamentales de la teoría de la probabilidad: el teorema central del límite y la ley de los grandes números. Por tanto, en este trabajo enunciamos y demostramos las versiones *clásicas* de estos teoremas.

En este trabajo hemos concluido que en situación de evolución α -asíncrona, las configuraciones escogidas muestran un esquema de auto-organización, independientemente del número de iteraciones. Adicionalmente, concluimos que la α -asíncronicidad ralentiza en general la evolución de las configuraciones y que cuando α se aproxima a la unidad, los promedios de las variables observadas se aproximan de manera continua a los valores que se obtienen en el juego de vida síncrono.

Games of Life: simulation and characterization

Daniel Jiménez López

Keywords: Game of life, α -asynchronous Game of life, Monte Carlo simulation, cellular automata, m -asynchronous cellular automata.

Abstract

The cellular automata theory was introduced by von Neumann and Ulam in 1950 in order to study artificial self reproducing machines. The design of this kind of machines included the environment in which they evolve, a rectangular lattice. It is filled with cells, so each node is in one of the following states: filled (1) or empty (0). They evolve simultaneously obeying a fixed set of rules called transition rules. For each node, they take into account the number of filled nodes surrounding the node what is also known as the neighborhood of the node.

Conway's game of life has been one of the most popular cellular automata since its invention in 1970. In this particular case the neighborhood of one node is made of all surrounding nodes in all directions at a distance of one node. Conway's game of life implements the following set of transition rules:

- Any filled node with two or three filled nodes in its neighborhood remains filled, otherwise it empties.
- Any empty node with exactly three filled nodes in its neighborhood is filled, otherwise it keeps its current state.

As unpredictable behaviours emerged from the transition rules, scientists from many fields became interested in the game of life and new sets of transition rules were developed. In this work we study the behaviour of a reduced set of configurations when transition rules are applied with a probability α so that in each iteration only a subset of nodes are truly updated and the remaining set of nodes keep their state. This scheme of updating defines a new entire class of cellular automata called α -asynchronous cellular automata whereby we define the α -asynchronous game of life.

The lattice in which the game of life evolves is infinite but the memory of the computer is finite, as we think that there is no need to store a big finite lattice in memory, in our implementation of both the game of life and the α -asynchronous game of life only the filled nodes are stored in memory so that

larger patterns can fit in it. The major drawback of our implementation is that in order to apply the transition rules in each iteration we must compute the set of nodes which are going to be updated.

Even though there had been a huge interest in the game of life and many patterns had been found and classified, there is not a baseline classification which is globally accepted. In this work, we considered the three following classes of patterns of interest:

- Still lives, which are static patterns.
- Oscillators, patterns that repeat themselves after a fixed number of iterations. They can be seen as a subset of still lives.
- Spaceships, patterns that translate themselves across the board. The number of iterations before it repeats but in a different position is named the period of the spaceship.

Those classes of patterns had been collected in the biggest online database of objects in Conway's Game of Life and similar cellular automata, *Catagolue*. Note that as still lives are not affected by the α -asynchronous update scheme, so we did not study them. In this work we choose the two most common oscillators of period 2, 3 and 4 and the four most common spaceships. The behaviour of those patterns are studied in the α -asynchronous game of life. The chosen patterns are stored and read in the run length encoded format (RLE format) which allow us to store huge patterns in small files.

Given the randomness of the α -asynchronous game of life, we employ the Monte Carlo methods to study the change in the behaviour of the chosen patterns. In order to measure it we study in each iteration the mean values of:

- The number of filled nodes.
- The area of the smallest rectangle which contains all the filled nodes.
- The number of nodes which change their state also known as heat.
- The number of clusters, which are the smallest set of nodes with no node filled shared in their neighborhood.
- The number of still lives, that is, those clusters which are static patterns.

A section of our work is devoted to the mathematical foundations of the α -asynchronous game of life by means of the m -asynchronous cellular automata. A wider class of cellular automata which contains the game of life and

the α -asynchronous game of life cellular automatas. In addition we prove the two main theorems which support the Monte Carlo methods. Those are the (strong) law of big numbers and the central limit theorem.

We obtain the measures of the mean values of the variables of interest in a process with three stages, each intermediate stage is feed with the output of the stage before. The input of the first stage is a pattern in RLE format and a set of α values.

1. Fixed an α value, generate a JSON file with a set of fields filled with the data extracted from multiple simulations of each iteration of a chosen pattern.
2. The JSON file is loaded, then the mean, the standard deviation and a p-value is computed for each variable of interest in each iteration. Those values are outputted to a CSV file in which each column store the values described previously.
3. For each CSV file generated in the stage before, three kind of plots are made. Two which show the evolution of the mean values of one variable of interest for all values of α given and one showing only the mean value of the one variable of interest for a fixed value of α .

As a result of our study we conclude that the α -asynchronous scheme induces stability in the behaviour of the nodes. The observed mean values of the variables of interest converge continuously to the values obtained in the synchronous game of life when α values converges to 1. On the other hand, when α values converges to 0 the mean values of the variables of interest tend to diverge from the values obtained in the synchronous game of life. We have observed that spaceships can be characterized by the number of still lives in their α -asynchronous update, which is zero for all iterations.

Índice general

Índice general	6
1 Introducción	7
2 Aproximación conceptual	11
2.1. Juego de vida de Conway	11
2.2. Juego de vida de Conway α -asíncrono	15
2.3. Configuraciones iniciales del juego de vida de Conway	16
2.4. Simulaciones	22
3 Metodología	26
3.1. Teoría de la computación	26
3.2. Teoría de la probabilidad	30
3.3. Fundamentos de las simulaciones Monte Carlo	39
4 Análisis	44
5 Conclusiones	69
Bibliografía	71

1

Introducción

El autómata celular fue inventado por von Neumann y Ulam en 1950 para estudiar el problema de construir máquinas artificiales que se reproduzcan a sí mismas [1]. Con el fin de imitar el comportamiento de los seres vivos, el diseño de dichas máquinas incluye el espacio en el que se desarrollan, representado por una malla rectangular en la que en los nodos se sitúan células y éstas evolucionan simultáneamente de acuerdo a un conjunto de reglas simples que dirigen la *física* de su pequeño universo abstracto.

El juego de vida es un autómata celular propuesto por Conway en 1970 y popularizado por Gardner en el mismo año [2]. Consiste en la evolución de una disposición inicial de nodos ocupados con células en una malla rectangular infinita. Dicha evolución viene regida por un conjunto de reglas que se aplican simultáneamente a todas las células en cada iteración.

Uno de los motivos que atrajo la atención de científicos de diversos campos hacia el juego de vida fue observar cómo patrones complejos surgen de la aplicación de un conjunto muy simple y reducido de reglas. De esta manera comenzaron a analizarse configuraciones iniciales que daban lugar a comportamientos interesantes, tales como *glider* (Figura 1.1a) que se desplaza sobre la malla, *blinker* (Figura 1.1c) que retorna a su configuración inicial tras un número finito de iteraciones o *block* (Figura 1.1b) que no ve alterado su forma en cada iteración.

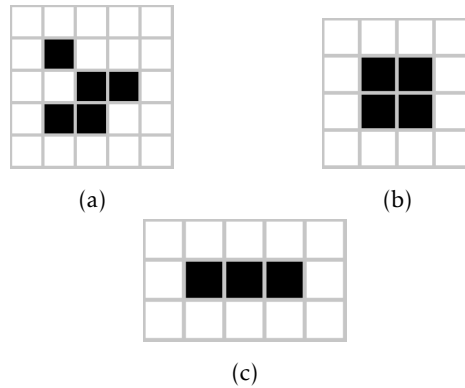


Figura 1.1: Algunas configuraciones iniciales básicas.

Al intentar realizar simulaciones de juegos de vida, los ordenadores se erigieron como la herramienta principal para llevarlas a cabo, siendo necesario afrontar el problema de representar una malla rectangular infinita en un ordenador con memoria finita. Una inteligente solución es alterar las características topológicas de la malla rectangular, identificando los bordes opuestos para obtener superficies topológicamente equivalentes a las de una botella de Klein, una esfera o un toro. En particular, esta última resultó atraer gran interés, pues se encontraron evidencias de que reduce los efectos asociados a la finitud de la malla [3, 4]. Cabe también descartar el estudio de la alteración de las cualidades geométricas de la malla, tales como el uso de figuras geométricas como el triángulo, el hexágono [5], teselaciones de Penrose [6] o el empleo del espacio geométrico hiperbólico [7]. Finalmente, existen implementaciones en las cuales no se almacena la malla en la memoria, si no que se almacena la posición de cada nodo ocupado respecto a un origen de coordenadas [8].

El juego de vida de Conway también muestra interesantes características en el campo de la teoría de la computación, ya que pertenece a la clase IV de Wolfram, esto es, su evolución lleva a estructuras aisladas que muestran un comportamiento complejo [9, 10]. Se ha demostrado que tiene la capacidad de cómputo de una máquina de Turing universal. Por tanto existe una disposición inicial de nodos ocupados que simula una máquina de Turing, la cual fue extendida a una máquina universal de Turing [11, 12]. Como muestra de dicha capacidad de computación, en un esfuerzo colectivo se ha implementado sobre el juego de vida un ordenador con sus propios lenguajes de bajo y alto nivel, y se ha programado el conocido juego Tetris [13, 14].

Si los autómatas celulares son un modelo que representa organismos vivos se podría pensar que la hipótesis de actualización simultánea es cuestionable ya que en la naturaleza no se propaga la información de manera instantánea.

nea y mucho menos de manera perfecta. Es ese aspecto, que está ligado a la robustez del modelo frente a perturbaciones en su evolución, el que pretendemos analizar en este trabajo. Un modelo será robusto si pequeños cambios en su evolución se traducen en pequeñas perturbaciones del comportamiento global del sistema, mientras que si esta pequeña modificación produce un cambio cualitativo en la dinámica, el sistema será poco robusto o, simplemente, se tratará de un sistema muy sensible a las variaciones que puedan ocurrir en sus elementos. En algunos contextos, los cambios cualitativos a los que nos referimos aquí se conocen como transición de fase. En la bibliografía la introducción de estas perturbaciones se lleva a cabo a través de la asincronicidad en la aplicación de las reglas de evolución [15]. Se pueden considerar las siguientes opciones:

- **Evolución totalmente asíncrona.** En cada iteración, las reglas de evolución se aplican solamente a un individuo escogido del conjunto de nodos, el criterio de elección puede ser o no ser completamente aleatorio.
- **Evolución α -asíncrona.** En cada iteración, cada nodo tiene probabilidad α de que se le apliquen las reglas de evolución y probabilidad $1 - \alpha$ de mantener su estado.

Estos esquemas de evolución también se conocen como **evolución guiada por pasos** y **evolución guiada por tiempo**, respectivamente [16]. Los primeros en estudiar los efectos de la evolución asíncrona frente a la evolución síncrona en el juego de vida fueron *Blok y Bergersen* [17]: aplicando un esquema de evolución α -asíncrona demostraron la existencia de una transición de fase de un comportamiento *estático*, donde el sistema termina alcanzando alguna situación completamente estable, a un comportamiento *vívido* y, por tanto, inestable. Posteriormente se estudió cómo afectaban las variaciones en la topología de la malla a la transición de fase, concluyendo que su aparición depende fuertemente de la regularidad de la malla rectangular [18]. Debido a que no existe una definición globalmente aceptada de autómatas celulares que agrupe los dos esquemas de actualización asíncrona anteriores, asumimos la definición de autómatas celulares m -asíncronos dada en [19]. Este autómata implementa un tipo de *oráculo* que selecciona los nodos de la malla que evolucionan en cada iteración.

Hasta donde podemos saber solo se ha estudiado el comportamiento en situaciones de α -asincronicidad de configuraciones iniciales aleatorias, también conocidas como *sopas*, con una densidad prefijada, obteniendo resultados que se comparan con las características conocidas del juego de vida síncrono. En

este trabajo queremos caracterizar la manera en la que configuraciones iniciales bien conocidas y estudiadas, alteran su comportamiento en situaciones de α -asincronicidad. Utilizaremos las técnicas Monte Calo para medir diferentes variables tales como el crecimiento de la población, el número de cúmulos de nodos ocupados y la actividad de cambio.

El capítulo 2 está dedicado a una descripción de los aspectos esenciales de nuestro trabajo. A continuación el capítulo 3 se proporciona una visión más formal de los conceptos expuestos en el capítulo anterior. En el capítulo 4 se documentan los resultados obtenidos y se discuten los aspectos más relevantes de los mismos. Por último en el capítulo 5 se establecen las conclusiones de nuestro trabajo.

Aproximación conceptual

2

En este capítulo daremos una descripción conceptual del universo del juego de vida de Conway junto con la del juego de vida de Conway α -asíncrono. En este contexto se describe en qué han consistido las simulaciones y sobre qué configuraciones iniciales se han realizado.

2.1. Juego de vida de Conway

Una configuración del juego de vida de Conway es la disposición sobre la malla de un conjunto de nodos ocupados. Por tanto una configuración inicial hace referencia a la disposición inicial de nodos ocupados sobre las que aún no se han aplicado las reglas de evolución. De esta manera cada nodo de la malla tiene dos estados posibles: vacío (0) o ocupado (1).

Las reglas de evolución que se aplican en cada iteración simultáneamente a todos los nodos de la malla son las siguientes:

- Un nodo ocupado se mantiene así si en su vecindario tiene solamente dos o tres nodos ocupados, en otro caso el nodo se vacía.
- Un nodo vacío se ocupa cuando en su vecindario hay exactamente tres nodos, en otro caso mantiene su estado vacío.

Usualmente se consideran como pertenecientes al vecindario los nodos adyacentes en las direcciones horizontal, vertical y diagonales.

En lo que hemos interpretado como un sentido *biológico*, las reglas se pueden describir también interpretando los nodos ocupados como células vivas y la configuración inicial como una población de células:

- Una célula puede *morir de soledad*, es decir, tiene solamente una célula en su vecindario o por *superpoblación*, esto es, tiene cuatro o más células en su vecindario.
- En un nodo vacío *nace* una célula si en su vecindario hay exactamente tres células vivas.

La elección de las reglas de evolución parecería *a priori* aleatoria, sin embargo, Conway las escogió aplicando las siguientes pautas [20]:

- No debe existir una disposición inicial de nodos ocupados para la cual haya una *prueba simple* de que la población crezca sin límite. Esto es, no debe de ser posible predecir fácilmente la evolución de una configuración inicial.
- Debe haber disposiciones iniciales de nodos ocupados que aparentemente crezcan sin límite.
- Debe haber disposiciones iniciales de nodos ocupados *sencillas* que crezcan y cambien durante un periodo relativamente largo, llegando a tres posibles finales: desaparecer completamente ya sea debido a superpoblación o a dispersión, estabilizarse en una configuración que se mantenga constante o entrar en un ciclo sin fin de oscilación.

Sin embargo las reglas que proporcionó Conway no son las únicas que muestran evoluciones interesantes. La siguiente notación nos es útil para expresar distintas reglas de evolución abreviadamente. Las reglas de evolución vienen dadas en la forma Bx/Sy donde y define el número de nodos ocupados en el vecindario para que un nodo ocupado se mantenga y x el número de nodos ocupados en el vecindario para que un nodo vacío se ocupe. Por ejemplo, para el juego de vida es B3/S23. Destacamos la regla B1357/S1357 conocida como *Edward Fredkin's replicating automaton*, en la cual cada configuración es eventualmente reemplazada por múltiples copias de sí misma, la regla B3/S12345 conocida como *Maze*, genera configuraciones similares a laberintos, una variación muy curiosa es que si añadimos el número 7 a la parte B es posible observar como un nodo ocupado recorre el laberinto y la regla B3/S012345678 conocida como *Life without Death*, en la cual los nodos ocupados nunca se vacían, se caracteriza por un crecimiento caótico y la aparición de configuraciones similares a escaleras que pueden ser usadas para simular circuitos booleanos [21].

Representación interna y actualización

Como se comentaba en la introducción, plantearse la simulación del juego de vida implica afrontar el problema de representar una malla infinita de dos dimensiones en la memoria finita de un ordenador. Aunque la cantidad de memoria y velocidad de acceso a la misma ha mejorado significativamente con el paso del tiempo, perseguimos una representación que cumpla las siguientes dos características:

- Una simulación de una configuración inicial del juego de vida tiene que finalizar en un tiempo razonable, pues la clave de los métodos Monte Carlo son la repetición de las mismas y como se comenta posteriormente en 3.3, al aumentar el número de simulaciones disminuye la varianza, permitiendo mayor precisión.
- El comportamiento de las configuraciones iniciales es difícil de predecir, por lo que aquellas que crezcan sin límite podrían agotar los recursos de memoria disponibles haciendo que la ejecución sea imposible. En particular, una situación con alto consumo de memoria dificulta la ejecución de múltiples simulaciones independientes en paralelo.

Este último punto es, en nuestra opinión, el más restrictivo. Un planteamiento inicial nos podría sugerir que limitar el tamaño de la malla dos dimensional, sin embargo se perdería información en aquellas configuraciones iniciales que excedieran el tamaño fijado de la malla. Para reducir el impacto de la finitud de la malla se ha estudiado la identificación de los bordes opuestos simulando un espacio *infinito* que imita la superficie de un toro, obteniendo resultados favorables [3, 4]. Pero no es necesario lidiar con los errores derivados de este planteamiento. Una implementación en nuestra opinión más *literal* de la descripción formal del juego de vida, nos permite romper con el paradigma de la limitación de la malla. En lugar de almacenar en memoria la malla completa independientemente de su utilización, se almacenan los nodos ocupados, dados por coordenadas sobre la malla rectangular identificada con el plano cartesiano [8]. La contrapartida de esta representación es que los nodos ocupados no son los únicos sobre los que se aplican las reglas, existen nodos vacíos sobre los que también se aplican. Así la implementación del proceso de evolución se puede desglosar en dos etapas:

- Calcular todos los nodos sobre los que se van a aplicar las reglas de evolución.
- Aplicar las reglas de evolución sobre cada nodo.

El algoritmo de la primera etapa es el siguiente (Algoritmo 1). Para cada nodo ocupado se genera su vecindario y en una tabla común a todos los nodos se apunta el número de veces que aparece cada nodo del vecindario. De esta manera en la tabla resultante tenemos enumerados cada nodo que va a evolucionar y el número de nodos ocupados en su vecindario. El vecindario de un nodo ocupado con coordenadas (x, y) viene dado por el conjunto de posiciones que difieren del nodo en cuestión en una unidad como máximo:

$$\begin{array}{ccccc}
(x-1, y+1) & (x, y+1) & (x+1, y+1) \\
(x-1, y) & (x, y) & (x+1, y) \\
(x-1, y-1) & (x, y-1) & (x+1, y-1)
\end{array}$$

Algorithm 1 Cálculo de los nodos sobre los que se van a aplicar las reglas de evolución

Require: *occupiedNodes*

$n \leftarrow [(-1, 1), (0, 1), (1, 1), (-1, 0), (1, 0), (-1, -1), (0, -1), (1, -1)]$

for all *node* \in *occupiedNodes* **do**

for *i* = 0 **to** 8 **do**

$neighborhoodNode = node + n[i]$

if $neighborhoodNode \notin table$ **then**

$table[neighborhoodNode] = 1$

else

$table[neighborhoodNode] += 1$

end if

end for

end for

return *table*

El proceso de evolución de cada nodo almacenado en la variable *table* que devuelve el algoritmo de la etapa anterior se describe en el algoritmo 2.

Algorithm 2 Evolución síncrona

Require: *table*

Require: *occupiedNodes*

$nextOccupiedNodes \leftarrow []$

for all *node* \in *table* **do**

if ($table[node] = 3$ **or** $table[node] = 2$) **and** $node \in occupiedNodes$ **then**

$nextOccupiedNodes.append(node)$

else if $table[node] = 3$ **and** $node \notin occupiedNodes$ **then**

$nextOccupiedNodes.append(node)$

end if

end for

return $nextOccupiedNodes$

Notar que existen algoritmos notablemente más complejos que el descrito en esta sección, como el algoritmo *QuickLife*, el cual se utiliza en una implementación de software libre de un simulador de autómatas celulares llamado

Golly [22] o el algoritmo *HashLife*. La idea principal de este último algoritmo se basa en el reconocimiento de configuraciones más pequeñas repetidas dentro de la configuración a la que se le están aplicando las reglas de evolución [23]. Nuestra elección viene motivada por la facilidad con la que se puede implementar, describir y modificar para más tarde representar la evolución α -asíncrona.

2.2. Juego de vida de Conway α -asíncrono

Antes de nada, es importante señalar que la modificación que el juego α -asíncrono introduce en las iteraciones de actualización de los estados de los nodos de la malla, sólo afecta a éstos y no a las características de la propia malla.

En un juego de vida α -asíncrono cada nodo tiene probabilidad α de ser actualizado y probabilidad $1 - \alpha$ de mantener su estado actual. Para ello se ha de generar, para cada nodo, un número pseudo-aleatorio de acuerdo a una distribución uniforme estándar. Si el número obtenido es superior a α las reglas se aplican tal y como están establecidas y, en caso contrario, no se hace, manteniendo el nodo su estado actual. Éste proceso se puede observar en detalle en el algoritmo 3. Notar que si $\alpha = 1$ el algoritmo de actualización α -asíncrono coincide con el síncrono.

Algorithm 3 Evolución α -síncrona

Require: *table*

Require: *occupiedNodes*

Require: *getRandomNumber()*

Ensure: $0 < \alpha \leq 1$

Ensure: $0 < \text{getRandomNumber}() < 1$

nextOccupiedNodes $\leftarrow []$

for all *node* \in *table* **do**

if *getRandomNumber()* $< \alpha$ **then**

if (*table*[*node*] = 3 **or** *table*[*node*] = 2) **and** *node* \in *occupiedNodes* **then**

nextOccupiedNodes.append(*node*)

else if *table*[*node*] = 3 **and** *node* \notin *occupiedNodes* **then**

nextOccupiedNodes.append(*node*)

end if

end if

end for

return *nextOccupiedNodes*

2.3. Configuraciones iniciales del juego de vida de Conway

Como ya se ha indicado anteriormente, en este trabajo tratamos de caracterizar el comportamiento de configuraciones iniciales del juego de vida de Conway bajo la hipótesis de actualización α -asíncrona. Nuestra elección de configuraciones iniciales está motivada por la simplicidad de los mismos, lo que nos permite visualizar de manera sencilla el impacto del α -asincronismo.

Dado que las configuraciones iniciales son muy diversas, han existido algunos esfuerzos por realizar una taxonomía de patrones, pero no existe un consenso global. A pesar de ello, hemos considerado la existencia de tres categorías principales que, a continuación, pasamos a describir.

Vidas inmóviles

Probablemente las *vidas inmóviles* sean las configuraciones con el comportamiento más simple y fácil de observar. Esta sección está extraída de las siguientes fuentes: [24], [25] y [26].

Una *vida inmóvil* es una configuración inicial que permanece inalterada en su evolución. A continuación mostramos ejemplos de estos tipos de vidas inmóviles en la Figura 2.1. La Figura 2.1a es una *vida inmóvil* en la que todos los nodos ocupados dependen entre sí los unos de los otros, si alguno se queda libre entonces la configuración deja de serlo. Análogamente, en la Figura 2.1b la mitad horizontal derecha conserva su estabilidad gracias a su homólogo reflejado de la mitad horizontal izquierda y al alterar el estado de cualquier nodo ocupado dicha estabilidad se desvanece. Por último, la Figura 2.1c es una *vida inmóvil* y a su vez contiene a otra *vida inmóvil*.

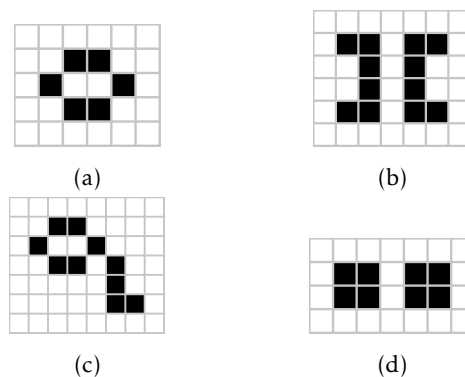


Figura 2.1: Algunas vidas inmóviles del juego de vida de Conway.

Notar que pueden estar formadas a su vez por varias *vida inmóvil* y que o

bien dependan entre sí para mantener su estabilidad como en la [Figura 2.1c](#), o bien sean independientes y al eliminar algunas de ellas la estabilidad se mantenga como en la [Figura 2.1d](#). También existen *vida inmóvil* que pueden ser separadas en vidas inmóviles independientes y que además existan nodos vacíos que tanto en la configuración inicial como en las *vidas inmóviles* independientes se mantengan así. Un ejemplo de esta situación se pueden observar en [Figura 2.2](#).

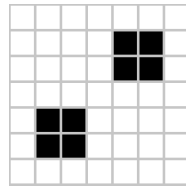


Figura 2.2: Configuración inicial que en el centro hay un nodo vacío que se mantiene tanto en las *vidas inmóviles* independientes como en el total.

Cabría preguntarse el problema de dado un número finito de nodos ocupados, ¿cuántas *vidas inmóviles* diferentes existen? Dicho problema ha sido resuelto para *vidas inmóviles* de hasta 32 nodos ocupados, como se puede consultar en [\[27\]](#).

Por último, hemos observado experimentalmente que las *vidas inmóviles* se mantienen inalteradas también en el juego de vida α -asíncrono. Ya que si se actualiza alguno de sus nodos, éste permanece en el mismo estado y si no se actualiza, también. Por tanto podemos concluir que la categoría de *vidas inmóviles* no se ve alterada por la introducción de la evolución α -asíncrona. Notar que esto no quiere decir cuando una vida inmóvil colisione con otros tipos de configuraciones se mantenga inalterada. Sin embargo, existen *vidas inmóviles* que cuando colisionan con cierto tipo de naves espaciales generan la destrucción de las mismas y tras algunas iteraciones la configuración vuelve a ser inmóvil. Este tipo de vidas inmóviles se les conoce como *eaters* y existen algunos que mientras que se da la interacción con una configuración, la cual más tarde desaparece, se mantienen en todo el proceso de desaparición inalterados [\[28\]](#).

Osciladores

Un *oscilador* es una configuración inicial que tras un número fijo de iteraciones se repite en la misma posición, al número de iteraciones se le conoce como periodo del *oscilador*. En particular, las vidas inmóviles pueden ser interpretadas como *osciladores* de periodo una iteración.

En la [Figura 2.3](#) mostramos dos configuraciones iniciales de periodo dos, durante tres iteraciones. Por un lado la [Figura 2.3a](#) muestra tres iteraciones de la configuración inicial nombrada *blinker* y por otro la [Figura 2.3b](#) son muestra iteraciones de la configuración inicial *toad*.

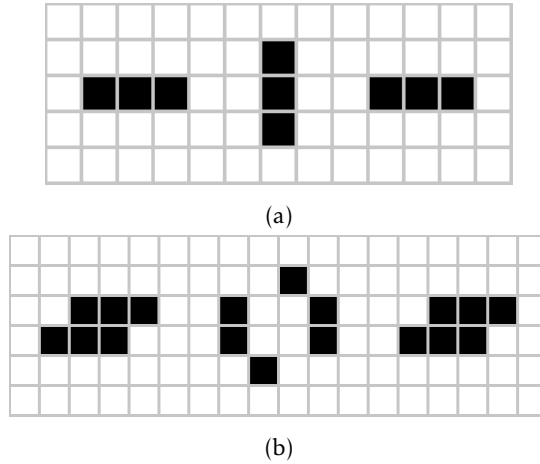


Figura 2.3: Primeras 3 iteraciones de 2 *osciladores* de periodo 2 del juego de vida de Conway.

Naves espaciales

Una *nave espacial* es una configuración inicial que tras un número fijo de iteraciones se repite pero en una posición desplazada. Particularmente, pueden ser vistas como osciladores que en el periodo de oscilación se desplazan. Dado que este tipo de configuraciones iniciales se desplaza sobre la malla rectangular es interesante considerar la velocidad con la que lo hacen. Si una configuración se desplaza (dx, dy) unidades cada periodo de longitud n , la velocidad de desplazamiento de la nave espacial es:

$$v = \frac{\max\{|dx|, |dy|\}}{k}$$

y su pendiente es x/y . Curiosamente se ha probado que para cada pendiente existe una *nave espacial* con dicha pendiente [29]. Notar que c es la velocidad máxima teórica, esto es, un desplazamiento por iteración.

En la [Figura 2.4](#) encontramos dos *naves espaciales* que se desplazan en distintas direcciones. La [Figura 2.4a](#) es la *nave espacial* más pequeña conocida de velocidad $c/4$ y su desplazamiento es diagonal y la [Figura 2.4b](#) es la más pequeña conocida de velocidad $c/2$ y su desplazamiento es horizontal.



Figura 2.4: Dos *naves espaciales* del juego de vida de Conway.

Almacenamiento de las configuraciones iniciales

Al igual que no existe un consenso global sobre las categorías de configuraciones del juego de vida, tampoco existe una única representación globalmente aceptada para almacenar una configuración en disco. Este problema es similar al de representar una malla rectangular infinita en una memoria finita. Existen soluciones que almacenan el rectángulo más pequeño que contiene a todos los nodos ocupados como sigue: cada fila de este rectángulo está representada por una cadena de caracteres en la cual los nodos ocupados se identifican por un carácter y los nodos vacíos por otro. Un ejemplo de este tipo de formato es *Life*1.05 que utiliza el carácter `*` para los nodos ocupados y el `.` para los nodos vacíos.

Sin embargo, este tipo de representación puede ocupar mucho espacio en configuraciones de gran tamaño, por lo que en este trabajo hemos utilizado una versión simplificada del formato *rle* [30], el cual es más compacto que el anterior. La primera línea de este formato tiene la forma $x = m, y = n$ donde m y n son las dimensiones del rectángulo de menor tamaño que contiene a todos los nodos ocupados. La siguiente línea codifica la configuración en una secuencia de elementos de la forma `número de repeticionesetiqueta`, donde `número de repeticiones` es el número de ocurrencias de una etiqueta que puede ser alguno de los siguientes caracteres: `b` que representa un nodo vacío, `o` que representa un nodo ocupado y el carácter `$` se emplea para indicar el final de la columna del rectángulo de menor tamaño que contiene a todos los nodos ocupados. El último elemento va seguido del carácter `!` que indica el final de la configuración. El número de nodos vacíos al final de la última fila de la configuración no tiene que estar necesariamente codificado y tampoco el final de la última fila tiene que incluir al carácter `$`. Las líneas iniciales de este formato que comiencen por el carácter `#` se interpretan como comentarios y por defecto se omiten. Por ejemplo las configuraciones de la [Figura 2.1](#) tendrían la siguiente codificación *rle*:

- # Configuración [Figura 2.1a](#)
 $x = 4, y = 3$

b2o\$02bo\$b2o!

- # Configuración [Figura 2.1b](#)

x = 5, y = 4

2ob2o\$bobo\$bobo\$2ob2o!

- # Configuración [Figura 2.1c](#)

x = 6, y = 5

b2o\$02bo\$b2obo\$4bo\$4b2o!

- # Configuración [Figura 2.1d](#)

x = 5, y = 2

2ob2o\$2ob2o!

Selección de configuraciones iniciales

Los primeros censos de configuraciones iniciales fueron *The Online Life-Life CA Soup Search* y *Achim Flammenkamp's census*, en los que se contabilizaron 174.631.866.050 y 50.158.095.316 configuraciones del juego de vida, respectivamente. El primero de ellos consistió en la evolución de 6.412.048.029 configuraciones iniciales aleatorias que cubren un cuadrado de lado 20 con densidad inicial de 0.5 sobre una malla rectangular infinita [31]. El segundo exploró la evolución de 1.829.196 configuraciones iniciales aleatorias sobre una malla cuadrada de lado 2048 con los bordes opuestos identificados y con una densidad inicial de 0.375 [32]. De ambos censos se puede extraer la conclusión de que las configuraciones que aparecen más a menudo son las *vidas inmóviles*, seguidas por los *osciladores* y por último las *naves espaciales*.

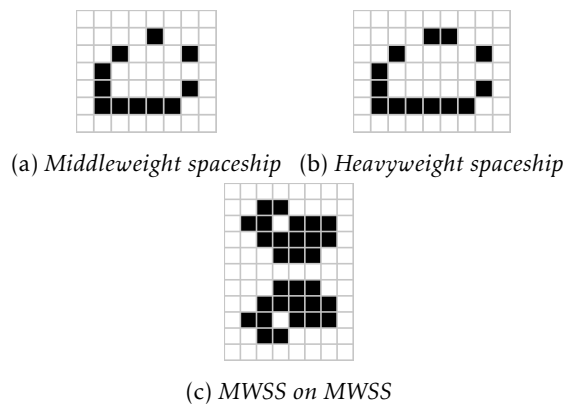


Figura 2.5: Tres últimas *naves espaciales* de las 5 más frecuentes en el censo Catalogue.

Para nuestro trabajo tomaremos de referencia el censo más actual *Catalogue* que recoge las ejecuciones de 19.640.649.096.999 configuraciones iniciales aleatorias cuadradas de lado 16 del juego de vida de Conway. Se han obtenido un total de 429.049.899.985.558 patrones de los cuales se encontraron 161.861 tipos diferentes [33]. En la [Tabla 2.1](#) se muestran las primeras 5 *naves espaciales* más frecuentes, de las cuales hemos seleccionado 4 para realizar nuestro experimento. Destacar que todas son de periodo 4 y que no tomaremos más dado que el número de ocurrencias disminuye drásticamente de la cuarta a la quinta posición. En la [Figura 2.5](#) podemos observar la forma de estas configuraciones iniciales.

Nombre	Periodo	Ocurrencias
<i>Glider</i>	4	37.699.263.597.381
<i>Lightweight spaceship</i>	4	55.075.316.989
<i>Middleweight spaceship (MWSS)</i>	4	14.511.262.233
<i>Heavyweight spaceship</i>	4	2.521.819.486
<i>MWSS on MWSS</i>	4	7.077

Cuadro 2.1: *Naves espaciales* más frecuentes en el censo *Catalogue*.

Como representantes de la categoría de *osciladores* vamos a estudiar los dos *osciladores* más frecuentes de periodo 2, 3 y 4, cuyas frecuencias se muestran en la [Tabla 2.2](#). Los *osciladores* más comunes de periodo 2 son *blinker* y *toad* introducidos anteriormente en la [Figura 2.3](#). Los de periodo 3 y 4 son respectivamente *pulsar* y *jam*, y *mold* y *mazing* ([Figura 2.6](#)).

Nombre	Periodo	Ocurrencias
<i>Blinker</i>	2	124.127.579.342.223
<i>Toad</i>	2	947.040.964.637
<i>Pulsar</i>	3	30.512.370.641
<i>Jam</i>	3	676.267
<i>Mold</i>	4	6.731.991
<i>Mazing</i>	4	1.281.808

Cuadro 2.2: Osciladores de periodos 2, 3 y 4 más frecuentes

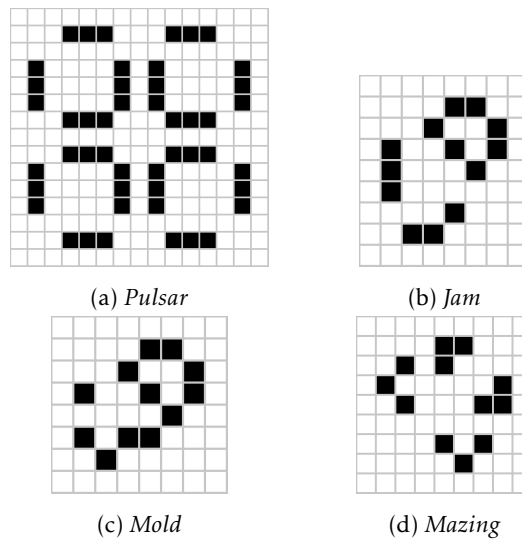


Figura 2.6: Osciladores de periodo 3 y 4 más frecuentes en el censo Catalogue.

2.4. Simulaciones

Dado el carácter aleatorio del juego de vida α -asíncrono emplearemos los fundamentos de Monte Carlo expuestos en la sección 3.3 para medir los parámetros de interés que a continuación exponemos.

- Crecimiento de la configuración inicial: dispondremos de tres herramientas para medir el cambio de crecimiento. Estudiaremos la evolución del número de nodos ocupados en cada etapa, la evolución del área del rectángulo de menor tamaño que contenga a todas los nodos ocupados de cada iteración y su densidad, esto es, el cociente del número de no-

dos ocupados por el área anterior. Cuyas unidades son respectivamente *nodos*, $nodos^2$ y $nodos^{-1}$.

- Tasa de cambio de la configuración inicial: emplearemos el concepto de calor, el número de nodos que cambian de estado por iteración. Cuya unidad es *nodos*.
- Distribución de los nodos ocupados en cúmulos: contabilizaremos el número de cúmulos por iteración, entendiendo por cúmulo al mayor conjunto de nodos ocupados cuyo vecindario no es disjunto, es decir, en un cúmulo cada nodo ocupado está contenido en el vecindario de otro nodo ocupado del cúmulo.
- Número de vida inmóviles simples, esto es, en cada iteración comprobamos cuales de los clústeres calculados es una vida inmóvil. Notar que como se comentaba anteriormente existen vidas inmóviles compuestas a su vez por otras vidas inmóviles, este tipo de vidas inmóviles no serán contabilizadas.

Estas variables serán medidas para distintos valores de α con el fin de estudiar el efecto de la aleatoriedad en las configuraciones iniciales.

Obtención de los datos

Dado que realizar múltiples simulaciones de una configuración inicial toma una cantidad considerable de tiempo, hemos considerado la separación del proceso de obtención de datos de las simulaciones en tres etapas bien diferenciadas:

1. Obtención de las variables de interés para cada conjunto de simulaciones de una configuración inicial.
2. Cálculo de los promedios y sus intervalos de confianza para cada variable en cada iteración.
3. Representación gráfica de los múltiples datos obtenidos para visualizar el impacto de la α -asincronicidad en la evolución.

La primera etapa recibe como entrada:

- Número de simulaciones
- Número de iteraciones que va a realizar cada simulación
- Distintos valores de α para los que se van a realizar las simulaciones.

- Una lista de configuraciones iniciales.

Con estos datos realiza la ejecución de múltiples simulaciones con un número dado de iteraciones de una o múltiples configuraciones iniciales. En cada iteración se calcula el número de nodos ocupados, el área del menor rectángulo que contiene a todos ellos, el calor y el número de clústeres. Adicionalmente se calcula un valor que represente unívocamente a una configuración, de manera que en caso de que en la misma iteración de múltiples simulaciones se repita un valor, éste no se almacene por duplicado. Esta etapa genera un archivo en formato *JSON* [34] con las siguientes entradas:

- *samplerSeed*: dado que cada simulación individual tiene una semilla propia, hemos decidido que dicha semilla sea generada a su vez por otro generador de números pseudo-aleatorios para asegurar una correcta distribución de las mismas. Este campo almacena la semilla que se ha empleado para el generador de semillas, asegurando la reproducibilidad de cada conjunto de simulaciones.
- *pattern*: este campo almacena el nombre de la configuración inicial sobre la que se han realizado las simulaciones.
- *alpha*: representa el valor de α -asincronismo en la evolución que se ha aplicado las simulaciones, éste valor tiene que estar entre 0 y 1 para ser un valor correcto. En particular, es el valor de α con el que se aplica el algoritmo 3 en cada iteración.
- *numberOfSteps*: este campo representa el número de iteraciones que se han realizado en cada simulación.
- *numberOfRuns*: este campo almacena el número de simulaciones que se han realizado.
- *runs*: es una lista en la cual cada posición corresponde con una iteración, a su vez cada iteración contiene una lista que representa las simulaciones realizadas. Cada elemento de esta última lista tiene las siguientes entradas:
 - *ncells*: este valor es el número de nodos ocupados.
 - *nclusters*: este campo es el número de clústeres calculados.
 - *area*: este valor es el área del menor rectángulo que contiene a todos los nodos ocupados.
 - *nstillLives*: este campo contabiliza el número de vidas inmóviles que se han podido localizar.

Notar que dado que las simulaciones no comparten información entre sí, pueden ser realizadas en procesos independientes para más tarde agregar todos los resultados en una sola salida.

La siguiente etapa acepta como entrada el archivo en formato JSON de la etapa anterior y calcula el valor medio y de la desviación típica de cada variable, generando un archivo en formato CSV [35] para cada variable medida en el cual cada fila tiene cuatro columnas en el siguiente orden: número de iteración, media, el radio del intervalo de confianza y el valor de p -value de un test de normalidad (sección 4). El nombre que recibe cada archivo está compuesto por la variable medida, el valor de α , el número de simulaciones y el número de iteraciones. En la etapa final utilizaremos el nombre de los ficheros generados para interpretar su contenido adecuadamente y nombrar a su vez los gráficos de salida.

Finalmente la última etapa recibe como entrada los archivos CSV generados en la etapa anterior y realiza dos tipos de gráficos. En primer lugar realiza un gráfico para cada archivo de entrada, donde el eje X viene dado por el número de la iteración, el eje Y por la variable observada, esta última se pinta en conjunción con su intervalo de confianza y el color que representa si supera o no el test de normalidad. A continuación, para cada variable medida realiza un nuevo gráfico, esta vez en lugar de representar la evolución de una variable para un solo valor de α , representa la evolución de la variable para todos los valores de α .

3

Metodología

Los formalismos nos permitirán articular la intuición de asincronismo en el esquema de actualización del autómata celular, cuyo posible homónimo biológico sería el procesamiento imperfecto de información entre individuos a causa de perturbaciones derivadas del medio o de la interacción con otros individuos. En este trabajo nos restringimos a un caso simple de asincronismo en la actualización: examinaremos que ocurre si todas las transiciones ocurren al mismo tiempo pero los individuos reciben la información del estado de sus vecinos de forma imperfecta.

En primer lugar introducimos el autómata celular m -asíncrono que nos permite articular el juego de vida de Conway α -asíncrono, a continuación exponemos los conceptos básicos de teoría de conjuntos, probabilidad y generación de números aleatorios sobre los que desarrollaremos las estimaciones Monte Carlo. Las claves de este desarrollo son el teorema central del límite y la ley de los grandes números que esencialmente justifican la efectividad del método Monte Carlo.

3.1. Teoría de la computación

Introducción

Dado que el comportamiento completamente síncrono de un autómata celular como herramienta de modelado es una rareza, se han realizado numerosas investigaciones empíricas del autómata celular asíncrono. Sin embargo, los pocos análisis formales realizados o bien se refieren a ejemplos o bien a casos particulares de asincronicidad. Tomaremos el concepto de autómata celular m -asíncrono [19], cuya idea principal es tener algún tipo de oráculo el cual en cada iteración escoge los nodos que tienen que ser actualizados. Dicho oráculo se implementa a través de una medida de probabilidad μ sobre subconjuntos de enteros d -dimensionales, \mathbb{Z}^d . Notar que la definición con la que trataremos es la extensión a espacios multidimensionales de la dada en [19].

Autómata celular determinista

Un autómata celular determinista es un sistema dinámico discreto consistente en un array d -dimensional de autómatas finitos, llamados nodos. Cada nodo está conectado uniformemente a un vecindario formado por un número finito de nodos, tiene un estado de un conjunto finito de estados y actualiza su estado de acuerdo a una función de transición local, la cual determina el siguiente estado de un nodo considerando su propio estado y el de su vecindario.

Definición 3.1.1. Formalmente, la tupla $A = (\mathbb{Z}^d, N, Q, f)$ es un autómata celular determinista, de ahora en adelante autómata celular, donde:

- \mathbb{Z}^d es un espacio d -dimensional.
- Q el conjunto de estados posibles para cada nodo.
- $N \in (\mathbb{Z}^d)^k$ es el vecindario genérico de un autómata celular, esto es, para $N = (n_1, \dots, n_k)$, $a \in \mathbb{Z}^d$ nodo, cada nodo ocupado en $\{(a + n_1, \dots, a + n_k)\}$ es un nodo ocupado vecino de a .
- $f : Q^{k+1} \rightarrow Q$ es la función de transición local que define la transición de estado de cada nodo como función de su propio estado y del estado de cada nodo ocupado en su vecindario.

Definición 3.1.2. Una configuración es una función $g : \mathbb{Z}^d \rightarrow Q$, la cual a cada punto del espacio \mathbb{Z}^d le asigna un estado del conjunto de estados Q , al conjunto de las configuraciones lo notaremos $Q^{\mathbb{Z}^d}$. Entenderemos por configuración inicial, a aquella a la que aún no se le ha aplicado la función de transición global.

Definición 3.1.3. La función de transición local induce una función de transición global $F : Q^{\mathbb{Z}^d} \rightarrow Q^{\mathbb{Z}^d}$ definida como sigue:

$$\forall x \in Q^{\mathbb{Z}^d}, \quad \forall i \in \mathbb{Z}^d, \quad F(x)(i) = f(x(i), x(i + n_1), \dots, x(i + n_k)).$$

Nos gustaría poder plasmar la intuición de que un nodo tenga la misma probabilidad de ser actualizada independientemente o no del resto de nodos actualizados en cada iteración. El concepto de ultrafiltro nos permitirá establecer una clase de autómata celular lo suficientemente general como para abarcar los nuevos tipos de autómatas celulares que introduciremos en posteriores secciones.

Dado un conjunto X , $\mathcal{P}(X)$ denota el conjunto de todos los subconjuntos de X . Dado $A \in \mathcal{P}(X)$, notaremos su complementario A^c .

Definición 3.1.4. $U \in \mathcal{P}(X)$ es un ultrafiltro de X si:

1. $\emptyset \in U$.
2. Sean $A, B \in \mathcal{P}(X)$ tales que $A \subset B$ y $A \in U$, entonces $B \in U$.
3. Si $A, B \in U$, entonces $A \cap B \in U$.
4. Si $A \in \mathcal{P}(X)$ entonces o bien $A \in U$, o bien $A^c \in U$.

Además dado $p \in X$, el ultrafiltro U_p diremos que es principal si es el más pequeño que contiene a p , esto es, la colección de todos los conjuntos que contienen a p .

Definición 3.1.5. Un autómata celular m -asíncrono C es una tupla (A, μ) donde:

- A es un autómata celular.
- μ es una medida de probabilidad sobre la σ -álgebra de Borel en $\mathcal{P}(\mathbb{Z}^d)$.

Definición 3.1.6. Para cada función de transición local f y cada conjunto $\tau \in \mathcal{P}(\mathbb{Z}^d)$, definimos la función de transición global $F : Q^{\mathbb{Z}^d} \rightarrow Q^{\mathbb{Z}^d}$ como sigue:

$$\forall x \in Q^{\mathbb{Z}^d}, \quad \forall i \in \mathbb{Z}^d, \quad F_\tau(x)(i) = \begin{cases} f(x(i), x(i+n_1), \dots, x(i+n_k)) & \text{si } i \in \tau, \\ x(i) & \text{si } i \notin \tau. \end{cases}$$

Es decir, F_τ aplica la función de transición local solo sobre los elementos de $\tau \subset \mathbb{Z}^d$.

Notar que cada nodo $i \in \mathbb{Z}^d$ es actualizado con probabilidad $\mu(U_i)$.

Esta nueva definición de autómata celular m -asíncrono, incluye la de autómata celular síncrono. Fijada una σ -álgebra \mathbb{B} sobre \mathbb{Z}^d y sea $C_0 = (A, \mu_0)$ un autómata celular m -asíncrono donde $\mu_0 : \mathbb{B} \rightarrow [0, 1]$ viene dada por:

$$\forall A \in \mathcal{P}(\mathbb{Z}^d), \quad \mu_0(A) = \begin{cases} 1 & \text{si } \mathbb{Z}^d \in A, \\ 0 & \text{si } \mathbb{Z}^d \notin A. \end{cases}$$

De esta manera, $\mu_0(\{\mathbb{Z}^d\}) = 1$ y por lo tanto, en cada instante de tiempo se aplicará la función de transición local sobre \mathbb{Z}^d .

Por otro lado, también contiene el concepto de evolución totalmente asíncrona comentado en la introducción, esto es, en cada instante se aplica la función de transición local a un solo nodo.

Definición 3.1.7. Consideramos ahora el autómata celular m -asíncrono $C = (A, \mu_1)$ donde $\mu_1 : \mathbb{B} \rightarrow [0, 1]$ verifica lo siguiente:

1. $\mu_1(U_i) > 0, \quad \forall i \in \mathbb{Z}^d.$
2. $\mu_1(U_i \cap U_j) = 0, \quad \forall i \neq j, \quad i, j \in \mathbb{Z}^d.$

Así solo los ultrafiltros de la forma $\{k\}$ ($k \in \mathbb{Z}^d$) se les aplica la función de transición local.

Por último, contiene el concepto de evolución α -asíncrona que nos interesa.

Definición 3.1.8. Dado $C = (A, \mu_2)$ un autómata celular m -asíncrono y sea $\alpha \in (0, 1)$ la probabilidad con la que se actualizan los nodos, donde $\mu_2 : \mathbb{B} \rightarrow [0, 1]$ satisface:

1. $\mu_2(U_i) = \alpha, \quad \forall i \in \mathbb{Z}^d.$
2. $\forall A \subseteq \mathbb{Z}^d \quad \text{finito}, \quad \mu_2(\bigcap_{a \in A} U_a) = \prod_{a \in A} \mu_2(U_a).$

y lo notaremos $C(\alpha)$.

Juego de vida de Conway

Definición 3.1.9. El juego de vida de Conway es un autómata celular síncrono:

$$\mathbb{A} = (\mathbb{Z}^2, N, Q, f)$$

donde

- $N = \{(-1, 1), (0, 1), (1, 1), (-1, 0), (1, 0), (-1, -1), (0, -1), (1, -1)\},$
- $Q = \{0, 1\},$
- $f : \{0, 1\}^9 \rightarrow \{0, 1\}$ dada por:

$$f(x) = \begin{cases} 1 & \text{si } x_0 = 0 \text{ y } \sum_{i=1}^8 x_i = 3 \\ 1 & \text{si } x_0 = 1 \text{ y } \sum_{i=1}^8 x_i \in \{2, 3\} \\ 0 & \text{si } \sum_{i=1}^8 x_i \notin \{2, 3\} \end{cases} \quad (3.1)$$

y $x = (x_0, x_1, \dots, x_8) = (c, c + n_1, \dots, c + n_8)$ con $c \in \mathbb{Z}^d$ nodo.

Configuraciones iniciales

Fijada una configuración inicial $z \in Q^{\mathbb{Z}^2}$, enteremos por ejecución o simulación del juego de vida de Conway de duración $n \in \mathbb{N} \cup \{\infty\}$ al resultado de aplicar n veces la función de transición global del juego de vida de Conway a la configuración inicial y lo notaremos $\mathbb{A}_n(z)$. Sea t tal que $1 \leq t \leq n$, diremos que es el instante t de la ejecución del juego de vida de Conway de duración t , el resultado de aplicar t veces la función de transición global a una configuración inicial z y lo notaremos $\mathbb{A}_n^t(z)$. Por último, los conjuntos de puntos $\mathbb{A}_n^{t'}(z')$, $\mathbb{A}_n^t(z)$ serán iguales, si los conjuntos de nodos ocupados de cada simulación lo son.

Definición 3.1.10. Una *vida inmóvil* es una configuración inicial que permanece en cada iteración, esto es, fijado $n \in \mathbb{N}$, se verifica que $z = \mathbb{A}_0^n(z) = \mathbb{A}_t^n(z)$ para todo $t \leq n$.

Definición 3.1.11. Un *oscilador* es una configuración inicial $z \in Q^{\mathbb{Z}^2}$ que se repite tras la aplicación de k veces de la función de transición global, esto es, fijado $n \in \mathbb{N}$, se verifica que existe $k > 1, \in \mathbb{N}$ tal que $\mathbb{A}_t^n(z) = \mathbb{A}_{t+k}^n(z)$ para todo $t \leq n - k$ y diremos que k es el periodo del *oscilador*.

Definición 3.1.12. Una *nave espacial* es una configuración inicial $z \in Q^{\mathbb{Z}^2}$ que se repite tras la aplicación de k veces de la función de transición global pero en una posición distinta, esto es, fijado $n \in \mathbb{N}$, se verifica que existen $k \in \mathbb{N}$ y una traslación distinta de la identidad, $\phi : \mathbb{Z}^2 \rightarrow \mathbb{Z}^2$, tal que $\mathbb{A}_t^n(z) = \phi(\mathbb{A}_{t+k}^n(z))$ para todo $t \leq n - k$ y diremos que k es el periodo de la *nave espacial*.

Juego de vida de Conway α -asíncrono

Definición 3.1.13. El juego de vida de Conway α -asíncrono es un autómata celular m -asíncrono $\mathbb{A}_\alpha = (\mathbb{A}, \mu)$ donde \mathbb{A} es el autómata celular síncrono del juego de vida de Conway, $\mu : \mathbb{B} \rightarrow [0, 1]$ una medida de probabilidad verificando las condiciones expuestas en 3.1.8 y $\alpha \in (0, 1)$.

3.2. Teoría de la probabilidad

El contenido de esta sección está extraído de los siguientes textos: [36–38].

Definición 3.2.1. Una σ -álgebra, \mathbb{F} , sobre un conjunto X , es una colección no vacía de subconjuntos de X cerrados para uniones numerables y para la operación de complementario, esto es:

- $\forall A \in \mathbb{F}$ se verifica que $A^c \in \mathbb{F}$.
- $\forall A_n \in \mathbb{F}, n \in \mathbb{N}$ se verifica que $\bigcup_{n \in \mathbb{N}} A_n \in \mathbb{F}$.

Definición 3.2.2. Sean un conjunto X con su σ -álgebra asociada, \mathbb{F} , el par (X, \mathbb{F}) es un espacio medible.

Definición 3.2.3. Una función medible es una función entre espacios medibles, $g : (X, \mathbb{F}) \rightarrow (X', \mathbb{F}')$ tal que: $g^{-1}(A) \in \mathbb{F} \quad \forall A \in \mathbb{F}'$.

Definición 3.2.4. La tupla (X, \mathbb{F}, P) es un espacio de probabilidad si:

- X es el espacio de muestreo, esto es, algún conjunto no vacío.
- \mathbb{F} es una σ -álgebra de sucesos.
- $P : \mathbb{F} \rightarrow \mathbb{R}$ es una medida de probabilidad, esto es, P satisface los siguientes axiomas de Kolmogorov:
 1. Para cada $A \in \mathbb{F}$, existe un número $P(A) \geq 0$, esto es, la probabilidad del suceso A ,
 2. $P(X) = 1$.
 3. Sean $A_n, n \geq 1$ disjuntos, entonces:

$$P\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} P(A_n).$$

Definición 3.2.5. Los sucesos $A_n, n \geq 1$ son independientes si y solo si

$$P\left(\bigcap_{n \geq 1} A_n\right) = \prod_{n \geq 1} P(A_n).$$

Definición 3.2.6. Un conjunto A es abierto si, para cada punto $x \in A$, existe una bola de centro el punto y radio $\epsilon > 0$, $B(x, \epsilon) = \{z : |z - x| < \epsilon\}$, tal que $B(x, \epsilon) \subset A$. Así la σ -álgebra de Borel, es aquella generada por los conjuntos abiertos de \mathbb{R} .

De ahora en adelante supondremos que se ha fijado el espacio medible dado por $X = \mathbb{R}$ y $\mathbb{F} = \mathbb{B}$ la σ -álgebra de Borel sobre \mathbb{R} .

Variables aleatorias

Definición 3.2.7. Una variable aleatoria definida sobre un espacio de probabilidad (X, \mathbb{B}, P) es una función medible $A : X \rightarrow \mathbb{R}$.

Cada valor de A se corresponde con un subconjunto de puntos de X que se aplica en dicho valor: $\{w \in X : A(w) = x\}$, que notaremos por simplicidad $\{X = x\}$. A parte de los anteriores conjuntos también nos resultarán de interés lo siguientes:

$$\{w \in X : A(w) \leq x\} = \{A \leq x\}$$

$$\{w \in X : A(w) < x\} = \{A < x\}$$

$$\{w \in X : A(w) > x\} = \{A > x\}$$

$$\{w \in X : A(w) \geq x\} = \{A \geq x\}$$

Proposición 3.2.1. Si $g : \mathbb{R} \rightarrow \mathbb{R}$ es medible y A es una variable aleatoria entonces $A' = g(A)$ es una variable aleatoria.

Definición 3.2.8. Las variables aleatorias A_1, A_2, \dots, A_n son independientes si y solo si, para arbitrarios conjuntos de la σ -álgebra de Borel B_1, B_2, \dots, B_n :

$$P\left(\bigcap_{k=1}^n \{A_k \in B_k\}\right) = \prod_{k=1}^n P(A_k \in B_k).$$

Definición 3.2.9. Dada una variable aleatoria A se define su función de distribución como $F : \mathbb{R} \rightarrow [0, 1]$ dada por:

$$x \mapsto F(x) = P(A \leq x).$$

Proposición 3.2.2. La función de distribución de la variable aleatoria A satisface:

- Es monótona no decreciente.
- Dada una sucesión decreciente de elementos de \mathbb{R} , $\{x_n\}_{n \in \mathbb{N}} \in \mathbb{R}$ convergente a $x \in \mathbb{R}$ se tiene $\lim_{x_n \rightarrow x} F(x_n) = F(x)$, es decir, es continua a la derecha.
- $\lim_{x \rightarrow +\infty} F(x) = 1$ y $\lim_{x \rightarrow -\infty} F(x) = 0$.

Definición 3.2.10. Sea F una función de distribución definimos la función de densidad como la función integrable, f , tal que:

$$F(b) - F(a) = \int_a^b f(x) dx, \quad \forall a < b.$$

Definición 3.2.11. Sea A una variable aleatoria, definimos su valor esperado o esperanza como sigue:

$$\mathbb{E}(A) = \int_X A(w) dP(w).$$

Adicionalmente si $\mathbb{E}|A| < \infty$, diremos que A es integrable.

Definición 3.2.12. Sea una variable aleatoria A , definimos:

- Los momentos de orden n de A : $\mathbb{E}(A^n) = \int_X A(w)^n dP(w)$, $n \in \mathbb{N}$.
- Los momentos centrados de orden n de A : $\mathbb{E}_c(A^n) = \mathbb{E}((A - \mathbb{E}(A))^n)$, $n \in \mathbb{N}$.

Notar que los momentos no existen necesariamente para todo $n \in \mathbb{N}$.

Definición 3.2.13. Definimos la varianza de la variable aleatoria A con esperanza $\mu < \infty$ y $\mathbb{E}_c(A^2) < \infty$ como:

$$\text{var}(A) \equiv \mathbb{E}_c(A^2) = \mathbb{E}(A^2) - \mu^2.$$

Definición 3.2.14. A la raíz cuadrada positiva de la varianza la notaremos $\sigma(A) = +\sqrt{\text{var}(A)}$ y diremos que es la desviación estándar de la variable aleatoria A .

Proposición 3.2.3. Sea A una variable aleatoria con esperanza $\mu < \infty$ y varianza $\sigma^2 < \infty$ y sea $A' = aA + b$, donde $a, b \in \mathbb{R}$, entonces:

$$\mathbb{E}(A') = a\mu + b \quad \text{y} \quad \text{var}(A') = a^2 \sigma^2.$$

Variables aleatorias discretas

Definición 3.2.15. Una variable aleatoria A diremos que es discreta si toma valores en un conjunto numerable, esto es, $\exists E = \{x_n\}_{n \in \mathbb{N}} \subset \mathbb{R}$ tal que $P(A \in E) = 1$.

Definición 3.2.16. La función de distribución de una variable aleatoria discreta A es la siguiente:

$$\forall x \in \mathbb{R}, \quad F(x) = P(A \leq x) = \sum_{x_n \in E, x_n \leq x} P(A = x_n).$$

Definición 3.2.17. Sea una variable aleatoria discreta A , definimos:

- Los momentos de orden n de A : $\mathbb{E}(A^n) = \sum_{x_m \in E} x_m^n P(A = x_m)$.
- Los momentos centrados de orden n de A : $\mathbb{E}_c(A^n) = \sum_{x_m \in E} (x_m - \mu)^n P(A = x_m)$.

El momento $n = 1$ se conoce como valor esperado o esperanza:

$$\mathbb{E}(A) \equiv \sum_{x_m \in E} x_m P(A = x_m).$$

Definición 3.2.18 (Tipos de convergencia: convergencia en probabilidad, convergencia casi segura y convergencia en distribución). Sean $A_n, n \in \mathbb{N}$ y A variables aleatorias, definimos:

- $A_n \rightarrow A$ en probabilidad, si para todo $\epsilon > 0$, $\lim_{n \rightarrow \infty} P(|A_n - A| > \epsilon) = 0$ y lo notaremos $A_n \xrightarrow{P} A$.
- $A_n \rightarrow A$ casi seguramente, si $P(\lim_{n \rightarrow \infty} A_n = A) = 1$ y lo notaremos $A_n \xrightarrow{c.s.} A$.
- $A_n \rightarrow A$ en distribución, si $\lim_{n \rightarrow \infty} P(A_n \leq x) = P(A \leq x)$, $\forall x \in \mathbb{R}$ donde $x \mapsto P(A \leq x)$ es una función continua y lo notaremos $A_n \xrightarrow{d} A$.

Definición 3.2.19. Sea i el número complejo $i = \sqrt{-1}$, la extensión de la función exponencial $\exp : \mathbb{R} \rightarrow \mathbb{R}^+$ al cuerpo de los números complejos es $\exp : \mathbb{C} \rightarrow \mathbb{C}$ dada por:

$$z \mapsto \exp(iz) = e^{iz} = \cos z + i \sin z.$$

Notar que el módulo de la exponencial compleja está acotado por la unidad :

$$|\exp(iz)| = |\cos z + i \sin z| = \sqrt{\cos^2 z + \sin^2 z} = 1.$$

Esta propiedad de la exponencial compleja nos asegura la existencia para toda variable aleatoria de la siguiente definición.

Definición 3.2.20. La función característica asociada a la variable aleatoria A es la función $\phi_A : \mathbb{R} \rightarrow \mathbb{C}$, dada por:

$$\phi_A(t) = \mathbb{E}(e^{itA}).$$

Proposición 3.2.4. Sea ϕ_A la función característica de la variable aleatoria A , entonces:

- $|\phi_A(t)| \leq \phi_A(0) = 1$.
- ϕ_A es una función uniformemente continua.
- $\phi_{cA+b}(t) = e^{itb} \phi_A(ct)$, $c, b \in \mathbb{R}$.
- Sea $A_1, A_2, \dots, A_n, n \in \mathbb{N}$ una sucesión finita de variables aleatorias independientes, entonces:

$$\phi_{\sum_{i=1}^n A_i}(t) = \prod_{i=1}^n \phi_{A_i}(t).$$

Además si A_1, A_2, \dots, A_n son idénticamente distribuidas:

$$\phi_{\sum_{i=1}^n A_i}(t) = \left(\phi_{A_1}(t)\right)^n.$$

- Si $\mathbb{E}(A^k) < \infty$ su derivada k -ésima evaluada en 0 es $\phi_A^{(k)}(0) = i^k \mathbb{E}(A^k)$.

Distribución normal

Definición 3.2.21. Diremos que una variable aleatoria A sigue una distribución normal con media $\mu < \infty$ y varianza $\sigma^2 < \infty$, $N(\mu, \sigma^2)$, si A tiene la siguiente función de densidad:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right), \quad \forall x \in \mathbb{R}.$$

Además su función característica es:

$$\phi_A(t) = e^{-\frac{t^2}{2}}.$$

Proposición 3.2.5. Sean A, A' variables aleatorias pertenecientes a distribuciones normales con medias $\mu, \mu' < \infty$ y varianzas $\sigma_0^2, \sigma_1^2 < \infty$, $N(\mu, \sigma_0^2)$ y $N(\mu', \sigma_1^2)$ respectivamente, entonces la variable aleatoria dada por la suma $A + A'$ es una variable aleatoria que pertenece a una distribución normal $N(\mu + \mu', \sigma_0^2 + \sigma_1^2)$.

Proposición 3.2.6. Sea $N(\mu, \sigma^2)$ una distribución normal con media $\mu < \infty$ y varianza $\sigma^2 < \infty$, entonces el 68.27% de los valores de la distribución normal se encuentran en el intervalo $[\mu + \sigma, \mu - \sigma]$, el 95.45% en el intervalo $[\mu + 2\sigma, \mu - 2\sigma]$ y el 99.7% en el intervalo $[\mu + 3\sigma, \mu - 3\sigma]$, estos intervalos se conocen también como intervalos de confianza de la distribución normal.

Teorema central del límite

Teorema 3.2.1. Sean A_1, A_2, \dots, A_n variables aleatorias independientes, con esperanza $\mu < \infty$, varianza $\sigma^2 < \infty$ e idénticamente distribuidas. Entonces:

$$\frac{1}{\sigma\sqrt{n}} \left(\frac{1}{n} \sum_{i=1}^n A_i - n\mu \right) \rightarrow^d N(0, 1).$$

Previa a la demostración del teorema central del límite, introducimos las herramientas matemáticas que nos harán posible su demostración.

Teorema 3.2.2 (Teorema de Taylor). Sea $k \in \mathbb{N}$ y $f : \mathbb{R} \rightarrow \mathbb{R}$ k -veces diferenciable en el punto $a \in \mathbb{R}$. Entonces existe una función $h_k : \mathbb{R} \rightarrow \mathbb{R}$ tal que:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots + \frac{f^{(k)}(a)}{k!}(x-a)^k + h_k(x)(x-a)^k$$

$$\text{y } \lim_{x \rightarrow a} h_k(x) = 0.$$

Teorema 3.2.3 (Teorema de continuidad). Sean $A_1, A_2, \dots, A_n, n \in \mathbb{N}$ variables aleatorias entonces:

$$\lim_{n \rightarrow \infty} \phi_{A_n} = \phi_A(t), \quad \forall t \in \mathbb{R},$$

si y solo si

$$A_n \rightarrow^d A.$$

Demostración (Teorema central del límite). Sean A_1, A_2, \dots, A_n variables aleatorias independientes idénticamente distribuidas con esperanza $\mu < \infty$ y varianza $\sigma^2 < \infty$. Sea ahora

$$Z_n = \frac{A_1 + A_2 + \dots + A_n - n\mu}{\sigma\sqrt{n}}.$$

Definimos una nueva variable aleatoria, Y_i , que es la versión normalizada de A_i :

$$Y_i = \frac{A_i - \mu}{\sigma}$$

Así definida, Y_i es idénticamente distribuida con esperanza y varianza:

$$E(Y_i) = 0, \text{Var}(Y_i) = 1.$$

Sea ahora $Z_n = \frac{Y_1 + Y_2 + \dots + Y_n}{\sqrt{n}}$, queremos ver que:

$$\lim_{n \rightarrow \infty} \phi_{Z_n}(t) = e^{-\frac{t^2}{2}}.$$

Procedemos a desarrollar el siguiente término:

$$\phi_{\frac{Y_1 + Y_2 + \dots + Y_n}{\sqrt{n}}}(t) = \prod_{i=1}^n \phi_{\frac{Y_i}{\sqrt{n}}}(t) = \left(\phi_{\frac{Y_1}{\sqrt{n}}}(t) \right)^n.$$

Aplicando el teorema de Taylor para obtener el desarrollo centrado en 0 para $k = 2$ de $\phi_{Y_1}(\frac{t}{\sqrt{n}})$:

$$\begin{aligned}
\phi_{\frac{Y_1}{\sqrt{n}}}(t) &= \phi_{Y_1}\left(\frac{t}{\sqrt{n}}\right) \\
&= \phi_{Y_1}(0) + \frac{t}{\sqrt{n}}\phi'_{Y_1}(0) + \frac{t^2}{2n}\phi''_{Y_1}(0) + \frac{t^2}{n}h_2(t) \\
&= 1 + i\frac{t}{\sqrt{n}}\mathbb{E}(Y_1) - \frac{t^2}{2n}\mathbb{E}(Y_1^2) + \frac{t^2}{n}h_2(t) \\
&= 1 + 0 - \frac{t^2}{2n} + \frac{t^2}{n}h_2\left(\frac{t}{\sqrt{n}}\right),
\end{aligned}$$

donde $\lim_{t \rightarrow 0} h_2\left(\frac{t}{\sqrt{n}}\right) = 0$.

Obtenemos que:

$$\left(\phi_{Y_1}\left(\frac{t}{\sqrt{n}}\right)\right)^n = \left(1 - \frac{t^2}{2n} + \frac{t^2}{n}h_2\left(\frac{t}{\sqrt{n}}\right)\right)^n \longrightarrow e^{-\frac{t^2}{2}}.$$

cuyo límite es la función característica de una variable aleatoria perteneciente a una distribución normal con media 0 y varianza 1, concluimos la demostración aplicando el teorema de continuidad 3.2.3. \square

Ley de los grandes números

Teorema 3.2.4. Sean A_n , $n \in \mathbb{N}$ variables aleatorias independientes, con esperanza $\mu < \infty$ e idénticamente distribuidas. Entonces el valor medio de A_n , $\bar{\mu}$, converge casi seguramente a μ :

$$\bar{\mu} = \frac{1}{n} \sum_{n \in \mathbb{N}} A_n \xrightarrow{c.s.} \mu,$$

esto es, $P(\lim_{n \rightarrow \infty} \bar{\mu} = \mu) = 1$.

Para obtener una demostración un tanto más breve de éste teorema añadiremos la hipótesis de existencia del momento de orden 4 de A_n . Una demostración completa del teorema sin la hipótesis adicional se puede consultar en [36].

Lema 3.2.1. Sean $A_n, n \geq 1$ variables aleatorias no negativas, entonces:

$$E\left(\sum_{n \geq 1} A_n\right) = \sum_{n \geq 1} E(A_n).$$

Lema 3.2.2. En las mismas condiciones de la ley de los grandes números, existe una constante $K < \infty$ tal que para todo $n \geq 0$:

$$\mathbb{E}((\bar{\mu} - n\mu)^4) \leq Kn^2.$$

Demostración. Sean

$$Z_k = A_k - \mu \quad y \quad T_n = Z_1 + Z_2 + \dots + Z_n = \sum_{i=1}^n A_i - n\mu.$$

Entonces:

$$\mathbb{E}(T_n^4) = \mathbb{E}\left(\left(\sum_{i=1}^n Z_i\right)^4\right) = n\mathbb{E}(Z_1^4) + 3n(n-1)\mathbb{E}(Z_1^2 Z_2^2) \leq Kn^2.$$

donde en la segunda igualdad se ha empleado el desarrollo multinomial:

$$(x_1 + x_2 + \dots + x_m)^n = \sum_{k_1 + k_2 + \dots + k_m = n} \binom{n}{k_1, k_2, \dots, k_m} \prod_{1 \leq t \leq m} x_t^{k_t},$$

con

$$\binom{n}{k_1, k_2, \dots, k_n} = \frac{n!}{k_1! k_2! \dots k_n!}.$$

Dado que $\mathbb{E}(Z_k) = 0 \quad \forall k$ y la independencia de las variables Z_k , se cancelan todos los sumandos de la forma:

$$\begin{aligned} \mathbb{E}(Z_i Z_j^3) &= \mathbb{E}(Z_i) \mathbb{E}(Z_j^3) = 0, \quad 1 \leq i, j \leq n, \quad i \neq j, \\ \mathbb{E}(Z_i Z_j Z_k Z_l) &= \mathbb{E}(Z_i) \mathbb{E}(Z_j) \mathbb{E}(Z_k) \mathbb{E}(Z_l) = 0, \quad 1 \leq i, j, k, l \leq n, \quad i \neq j \neq k \neq l. \end{aligned}$$

y siendo K adecuadamente elegida, por ejemplo $K = 4 \max\{\mathbb{E}(Z_1^4), \mathbb{E}(Z_1^2)^2\}$. □

Ya tenemos todos los rudimentos necesarios para proceder a demostrar el teorema de esta sección.

Demostración (Ley de los grandes números). Asumamos que $\mathbb{E}_c(A_n^4) < \infty \quad \forall n$, aplicando el lema anterior:

$$\mathbb{E}((\bar{\mu} - \mu)^4) \leq \frac{K}{n^2}.$$

Ahora sea $Y_n = (\bar{\mu} - \mu)^4, \forall n \in \mathbb{N}$ una variable aleatoria por la proposición 3.2.1 y en particular es no negativa, luego podemos aplicar el lema 3.2.1 en la siguiente cadena de igualdades:

$$\mathbb{E}\left(\sum_{n \geq 1} (\bar{\mu} - \mu)^4\right) = \sum_{n \geq 1} \mathbb{E}((\bar{\mu} - \mu)^4) \leq K \sum_{n \geq 1} \frac{1}{n^2} < \infty,$$

lo que implica:

$$\left| \sum_{n \geq 1} (\bar{\mu} - \mu)^4 \right| = \sum_{n \geq 1} (\bar{\mu} - \mu)^4 < \infty \quad c.s.$$

Pero si una serie es convergente, entonces la sucesión de su término general de la serie converge a cero, por tanto:

$$\bar{\mu} \rightarrow \mu \quad c.s.$$

□

3.3. Fundamentos de las simulaciones Monte Carlo

El nombre *Monte Carlo* fue acuñado por los científicos que trabajaban en el desarrollo de armas nucleares en Los Álamos en la década de los 40 para designar una clase de métodos numéricos basados en el uso de números aleatorios. La esencia de este método reside en la invención de juegos de azar cuyo comportamiento puede ser usado para estudiar algún fenómeno de interés. Se podría pensar que el hecho de que resultados obtenidos por estos métodos estén sujetos a las leyes del azar es un problema, sin embargo, es un problema menor puesto que se puede determinar como de exactos son sus resultados y si se deseara obtener resultados más precisos, bastaría con incrementar el número de experimentos realizados. Actualmente, los métodos de Monte Carlo juegan un papel fundamental en la resolución de problemas matemáticos complejos, en los cuales, o bien los métodos de resolución analíticos o bien los métodos numéricos existentes requieren de grandes periodos de tiempo cómputo.

Curiosamente, en las definiciones de los métodos de Monte Carlo no hay referencia explícita al empleo de la capacidades de cómputo de los ordenadores, sin embargo el gran desarrollo que han experimentado éstos desde el

último tercio de siglo XX hasta nuestros días, los ha convertido en herramientas indispensables en las simulaciones Monte Carlo. La generación de números aleatorios ha experimentado también un importante crecimiento en las últimas décadas.

Generadores de números pseudo-aleatorios

Los generadores de números pseudo-aleatorios producen secuencias indistinguibles de una realmente aleatoria, es decir, no generan valores de una distribución uniforme si no que dado una semilla o valor de inicialización generan siempre la misma sucesión de números.

Un tipo generador de números aleatorios muy usado venía dado por la siguiente ecuación expresión:

$$I_{j+1} = aI_j + c \quad \text{mód} (m) \quad (3.2)$$

donde a es un entero positivo llamado multiplicador y c es un número natural llamado incremento. Para $c \neq 0$, el generador de la ecuación 3.2 es conocido por el nombre de *generador lineal congruente*. Claramente, en $n < m$ pasos la ecuación comienza a generar valores duplicados en el mismo orden. Conocido ésto, se hacían elecciones particulares de a, c y m que proporcionaran el mayor periodo posible. En [39] podemos encontrar algunos resultados notables sobre la elección parámetros a, c y m . La elección del valor inicial I_0 no es relevante, pues se generarán todos los naturales posibles entre 0 y $m-1$ antes de la primera repetición. Sin embargo no es suficiente con generar una sucesión de números de un largo periodo, además deben superar rigurosas baterías de tests empíricos que aseguren una buena distribución de las secuencias, además de la ausencia de patrones en las mismas. En el caso de los generadores lineales congruentes existe un resultado que afirma que las sucesivas n -tuplas de valores generados residen en al menos $(n!m)^{\frac{1}{m}}$ hiperplanos paralelos [40], luego esta clase de generadores no son adecuados para la generación de números aleatorios.

Dado el carácter empírico de las baterías de test, superarlas con éxito no asegura un generador de números perfecto, sino que probablemente se trate de un buen generador, ya que eventualmente con el suficiente tiempo se podría encontrar un test que no fuera superado con éxito. Por tanto nos interesaremos en tests que demuestren el mal comportamiento de un generador en un tiempo razonable.

De entre las numerosas baterías de tests existentes destacamos dos [41, 42]: *Dieharder*, que está basada en los primeros test estadísticos propuestos en *Diehard battery of tests*, incluye también los test desarrollados por el NIST

(National Institute for Standards and Technology) y la batería de tests *TestU01* [41, 42].

Nuestra elección es una variante del generador *Mersenne Twister* [43] presente en la biblioteca estándar del lenguaje de programación Python en las versiones posteriores a la 2.3 [44], y en el lenguaje de programación estadística *R* [45].

Métodos Monte Carlo

Los métodos de Monte Carlo se apoyan fundamentalmente en dos grandes resultados de la teoría de la probabilidad: la ley de los grandes números y el teorema central de límite. Ambos nos permiten describir la distribución límite de la suma de las variables aleatorias independientes, proporcionando también una estimación del error.

Definición 3.3.1. Una muestra aleatoria simple S_n , es un conjunto de $n \in \mathbb{N}$ variables aleatorias, A_1, A_2, \dots, A_n , independientes e idénticamente distribuidas. En caso de que la media y la varianza de las variables aleatorias A_1, A_2, \dots, A_n sean finitas, las notaremos μ y σ^2 respectivamente.

La media de una muestra aleatoria simple S_n , $\mu_S = E(S_n) = \frac{1}{n} \sum_{i=1}^n A_i$ es una variable aleatoria gracias al resultado 3.2.1, si además la media μ y varianza σ de A_i son finitas, dicha variable aleatoria tiene la siguiente media y varianza:

$$E(\mu_S) = E\left(\frac{1}{n} \sum_{i=1}^n A_i\right) = \frac{1}{n} \sum_{i=1}^n E(A_i) = \mu,$$

$$var(\mu_S) = var\left(\frac{1}{n} \sum_{i=1}^n A_i\right) = \frac{1}{n^2} \sum_{i=1}^n var(A_i) = \frac{\sigma^2}{n}.$$

Luego su desviación estándar es $\sqrt{var(\mu_S)} = \frac{\sigma}{\sqrt{n}}$.

Definición 3.3.2. La variable aleatoria μ_S anteriormente definida diremos que es un *estimador* del valor esperado $E(A) = \mu$.

Definición 3.3.3. Sea μ_S un estimador de la media de una muestra aleatoria simple, S_n , de media $\mu < \infty$ y varianza $\sigma^2 < \infty$ y $\delta > 0$, la desigualdad de Chebychev es:

$$P\left(|\mu_S - \mu| \geq \sqrt{\frac{var(\mu_S)}{\delta}}\right) = P\left(|\mu_S - \mu| \geq \frac{\sigma}{\sqrt{n\delta}}\right) \leq \delta.$$

El gran resultado o *teorema fundamental de las simulaciones Monte Carlo* que podemos deducir de los anteriores, en particular es una consecuencia directa del teorema 3.2.4, el estimador G de la media de una muestra aleatoria simple, S_n , de media $\mu < \infty$ y varianza $\sigma^2 < \infty$ converge en probabilidad al valor esperado μ :

$$\forall \epsilon > 0, \quad \lim_{n \rightarrow \infty} P(|G - \mu| > \epsilon) = 0.$$

Es posible aplicar la desigualdad 3.3.3 para obtener la velocidad de convergencia respecto a n . Veamos un ejemplo de éste hecho, dado $\delta = \frac{1}{100}$:

$$P\left((G - \mu)^2 \geq \frac{100}{n} \sigma^2\right) \leq \frac{1}{100},$$

Haciendo n lo suficientemente grande, la varianza de G se hace tan pequeña como se quiera, esto es, disminuye considerablemente la probabilidad de obtener una gran desviación relativa a δ entre el valor esperado y el los valores obtenidos.

Es posible obtener un resultado más fuerte que el anterior como consecuencia del teorema 3.2.1. Existe una función de distribución de probabilidad que aproxima los valores del estimador G , esto es, cuando $n \rightarrow \infty$, el teorema central del límite afirma que asintóticamente los valores de G convergen a una distribución normal 3.2.21. Por tanto, es posible reescribir la función de distribución como sigue:

$$f(G) = \sqrt{\frac{n}{2\pi\sigma^2}} \exp\left(-\frac{n(G - \mu)^2}{2\sigma^2}\right).$$

Cuando $n \rightarrow \infty$, el valor de G se encuentra en intervalos cada vez más estrechos centrados en $\mathbb{E}(G)$ y es posible medir la desviación en unidades de σ , es decir, el valor de G está dentro del intervalo centrado en $\mathbb{E}(G)$ de un error estándar el 68.3% de las veces, de dos errores estándar el 95.4% de las veces y de tres errores estándar el 99.7% de las veces, los intervalos son $[\mathbb{E}(G) - \sqrt{\text{var}(G)}, \mathbb{E}(G) + \sqrt{\text{var}(G)}]$, $[\mathbb{E}(G) - 2\sqrt{\text{var}(G)}, \mathbb{E}(G) + 2\sqrt{\text{var}(G)}]$ y $[\mathbb{E}(G) - 3\sqrt{\text{var}(G)}, \mathbb{E}(G) + 3\sqrt{\text{var}(G)}]$, respectivamente (Proposición 3.2.6). Como comentábamos anteriormente la convergencia es asintótica por lo que inicialmente desconocemos como de grande debe de ser n para poder aplicar el teorema.

Cuando la varianza no es finita, es posible encontrar una distribución límite para G que llevará a un caso particular del teorema central del límite, la cual se puede consultar en [36], en estos casos la distribución límite no será en general la distribución normal. En dicho caso, un estimador de la varianza

de la media estimada viene dado por:

$$var(G_n) = \frac{1}{n-1} \left(\frac{1}{n} \sum_{i=1}^n A_i^2 - \left(\frac{1}{n} \sum_{i=1}^n A_i \right)^2 \right)$$

4

Análisis

En esta sección vamos a estudiar el impacto de la α -asíncronicidad sobre las configuraciones iniciales escogidas en la sección 2.3. Las ejecuciones son de 50 iteraciones, cada iteración simulada 5000 veces, con los valores de α : 0.15, 0.3, 0.45, 0.6, 0.75 y 0.9. En las gráficas de múltiples valores de α se muestra también el valor $\alpha = 1$ para poder comparar el comportamiento α -asíncrono con el síncrono. Como resultado hemos obtenido los valores medios de las variables expuestas en la sección 2.4 junto con sus correspondientes intervalos de confianza para cada iteración.

Hipótesis de normalidad

Para que las estimaciones Monte Carlo tengan los intervalos de confianza tengan el significado que se les da en la sección 3.3, es necesario comprobar que la distribución del estimador de la media sigue una distribución normal. Para ello aplicamos el test de normalidad de Shapiro-Wilk [46] con un nivel de significancia de 0.05. En nuestra situación la hipótesis nula, H_0 , es que el estimador de la media sigue una distribución normal y la hipótesis alternativa H_1 es que sigue una distribución distinta a la normal. Con este test no podremos afirmar que definitivamente la distribución sea normal por lo que nos reduciremos a afirmar que no podemos negar que la distribución sea normal, esto es, no rechazar la hipótesis nula. Una buena propiedad de este test, la cual ha motivado su elección, es que la probabilidad de rechazar la hipótesis nula cuando es cierta es aproximadamente del 5% independientemente del tamaño de la muestra [47].

En la Figura 4.1 es posible observar como para cada iteración tenemos representado el intervalo de confianza $[\mu - 3\sigma, \mu + 3\sigma]$, junto con el punto central del intervalo: μ , la media del estimador. En las situaciones en las cuales el p -value generado por el test de normalidad no supera el nivel de significación se ha pintado de color naranja, es decir, las situaciones en las que se rechaza H_0 y en otro caso de color azul. En caso de que la hipótesis nula sea rechazada tan solo podemos afirmar que la media del estimador converge en probabilidad como se comenta en la sección 3.3.

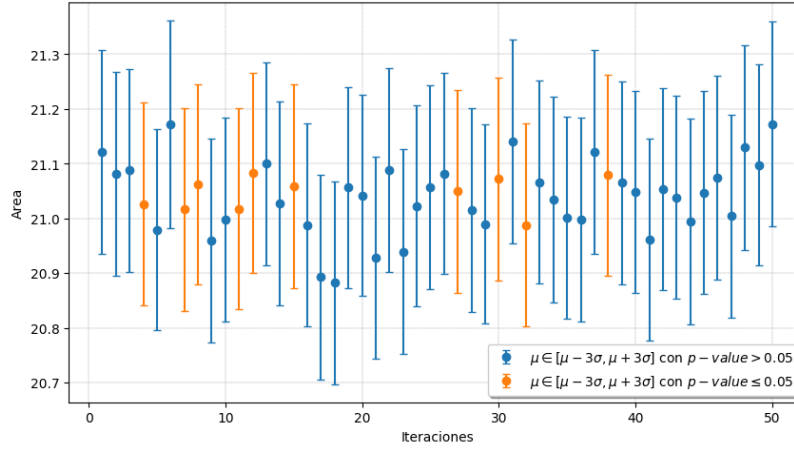


Figura 4.1: Promedio de densidad para $\alpha=0.3$ de la configuración *lightweight spaceship*.

Notar que para valores extremos de α hemos observado que se pueden encontrar situaciones en la cuales se rechaza más a menudo la hipótesis de normalidad, como las que se muestran en la Figura 4.2. Entendemos que esta situación se da porque si existe un número de simulaciones a partir del cual el estimador sigue una distribución normal, aún no se ha alcanzado. Otra situación posible en la que se rechaza la hipótesis de normalidad para casi todas las iteraciones es que los valores obtenidos sean constantes en todas las simulaciones y por tanto el estimador converge en probabilidad trivialmente como se muestra en la Figura 4.3 pero sin seguir una distribución normal.

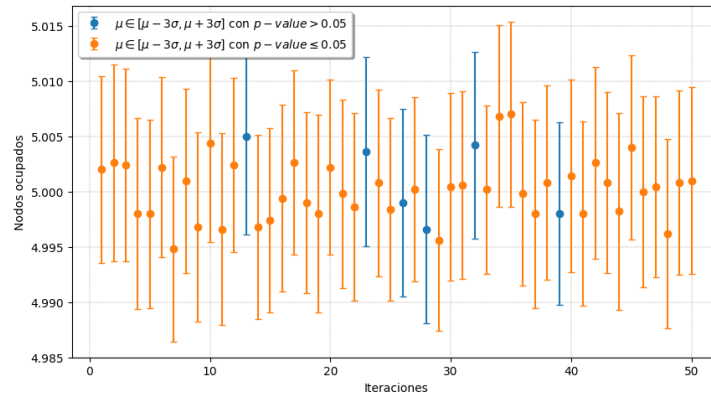


Figura 4.2: Promedio de nodos ocupados para $\alpha=0.99$ de la configuración *glider*.

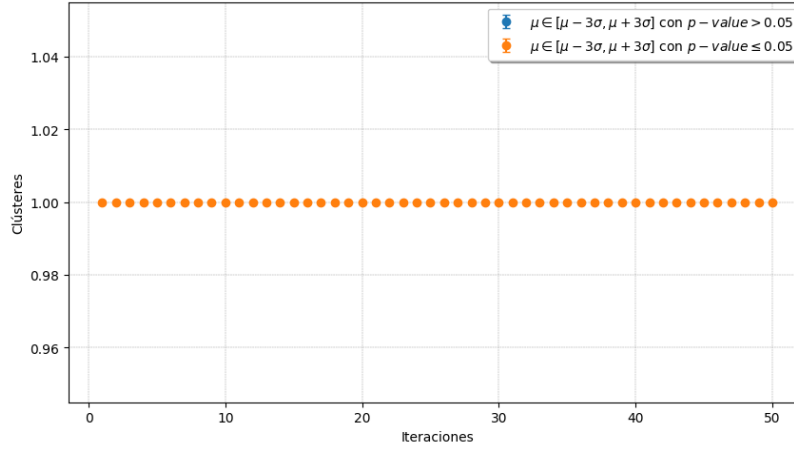


Figura 4.3: Promedio de clústeres para $\alpha=0.99$ de la configuración *lightweight spaceship*.

Osciladores de periodo 2

Comenzamos el análisis con la [Figura 2.3a](#). Cuando no existe perturbación en el intercambio de información entre nodos ocupados, el calor en cada iteración es de 4 *nodos*, ocupa un área de 3 *nodos*² y está conformada por un solo clúster de 3 nodos ocupados. En primer lugar destacamos que independientemente al valor que tome α el número medio de clústeres es constantemente la unidad.

Si observamos la [Figura 4.4](#) es posible ver que los valores medios permanecen aproximadamente constantes para cada valor de α . Además a medida que α se aproxima a la unidad, el promedio de calor se aproxima al que se obtiene en condiciones síncronas de evolución. También que puede observar notablemente que la variación de α produce que el calor medio varíe en más de 3 unidades, lo que nos hace pensar que el efecto de la α -asincronicidad sobre esta configuración ralentiza la evolución de la misma. En la [Figura 4.5](#) se observa con más claridad la evolución del calor medio junto con su desviación estándar frente al valor de α

Notar que el hecho de que los valores medios permanecen constantes independientemente al valor que tome α se da en todas las configuraciones, por lo que de ahora en adelante lo omitiremos.

Por otro lado, en la [Figura 4.6](#) se observa un comportamiento distinto al anterior. Se mantiene el hecho de que para valores de α cercanos a la unidad, el promedio se aproxima de forma continua al que se da bajo situación de

sincronicidad. En el rango de valores de $\alpha \in [0.15, 0.6]$ se observa que el área media crece hasta alcanzar su máximo valor cercano a 4 nodos^2 cuando α crece hasta 0.45, sin embargo a partir de este valor de α los promedios descienden hasta 3.5 nodos^2 en $\alpha = 0.75$.

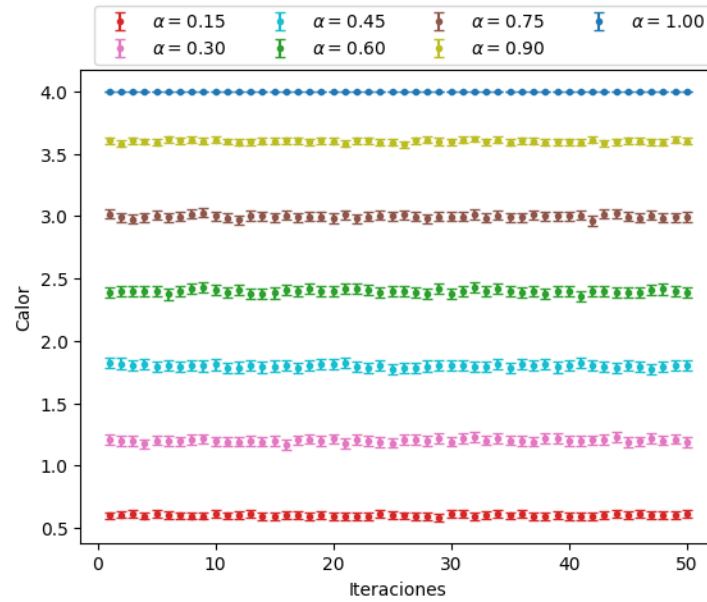


Figura 4.4: Evolución α -asíncrona del calor medio de la configuración *blinker*.

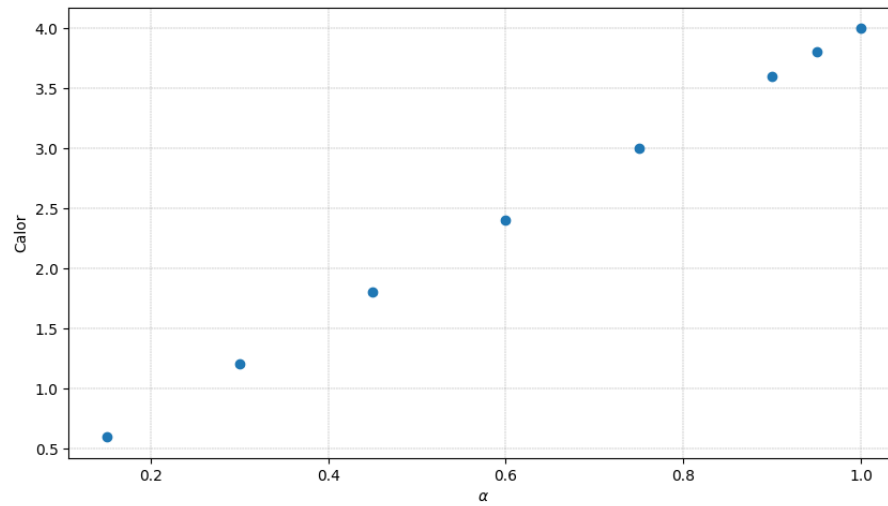


Figura 4.5: Evolución α -asíncrona del calor medio de la configuración *blinker*.

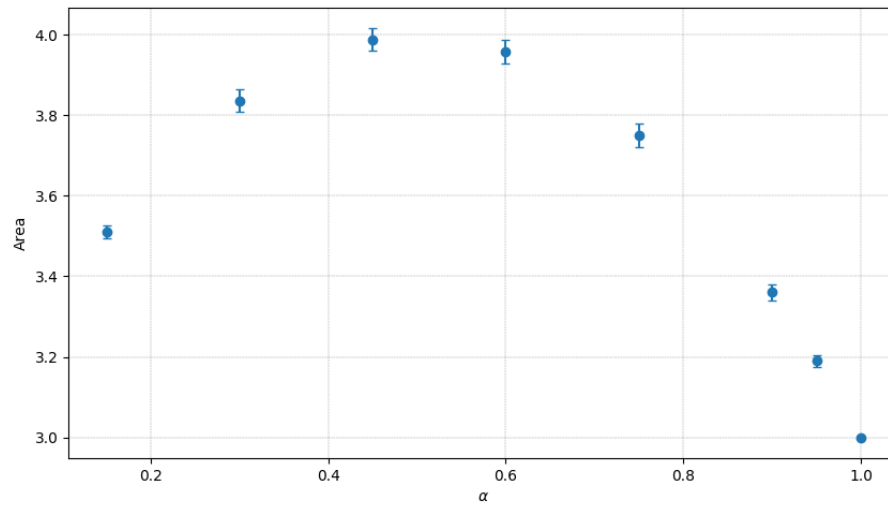


Figura 4.6: Evolución α -asíncrona del área media de la configuración *blinker*.

Tanto la densidad media como el promedio de vidas inmóviles y el promedio de células varían sutilmente, es decir, la longitud del intervalo en el que suceden los cambios es a lo sumo de 0.5 unidades, por tanto omitimos su análisis.

El otro oscilador de periodo dos que hemos estudiado, *toad*, (Figura 2.3b)

está formado por 6 nodos ocupados que ocupan un área de 8 nodos^2 en las iteraciones pares y 16 nodos^2 en las impares. A diferencia de la configuración anterior tanto el área como el calor medios experimentan un comportamiento muy similar en situación de α -asíncronicidad al descrito en la [Figura 4.4](#).

El número de clústeres medio deja de ser constante y varía en aproximadamente una unidad para los distintos valores de α ([Figura 4.7](#) y [Figura 4.8](#)). Por último la densidad media de esta configuración varía sutilmente y omitimos su análisis.

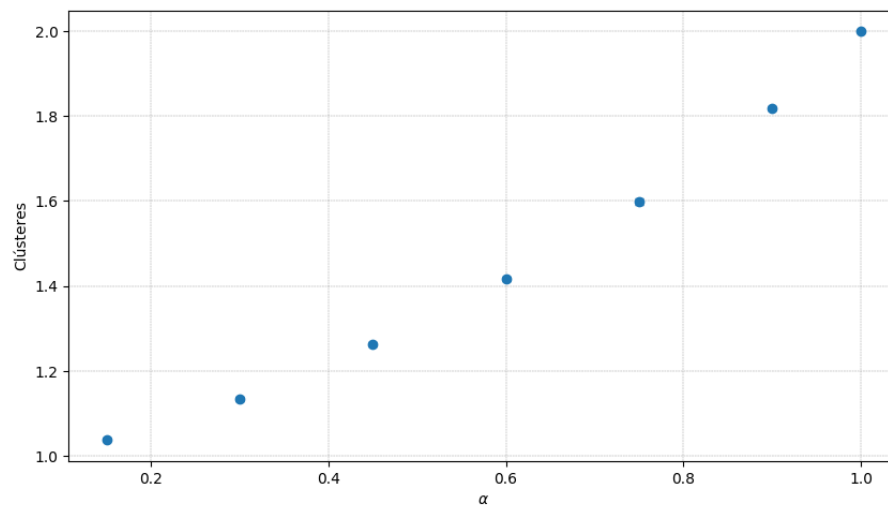


Figura 4.7: Evolución α -asíncrona del número de clústeres medio de la configuración *toad*.

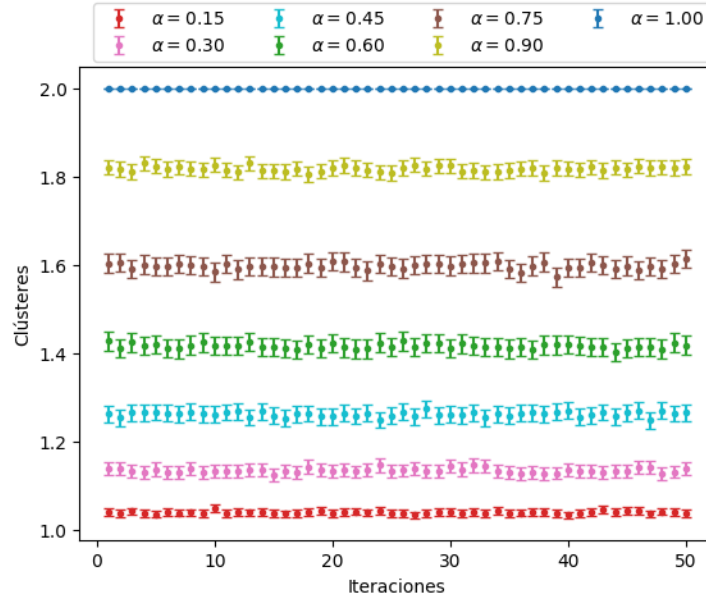


Figura 4.8: Evolución α -asíncrona del número de clústeres medio de la configuración *toad*.

Osciladores de periodo 3

En la Figura 4.9 podemos observar tres iteraciones de la configuración *jam*, la cual curiosamente contiene una vida inmóvil. En situación de actualización síncrona esta configuración inicial está formada por 13 nodos ocupados agrupados en tres clústeres y ocupa un área de 42 *nodos*². En su primera iteración se incrementa el número de nodos a 16, agrupados en un clúster y con un calor de 4 nodos, manteniendo el mismo área. La siguiente iteración incrementa su área a 49 *nodos*², disminuyendo el número de nodos ocupados a 14, agrupados en dos clústeres con un calor de 9 nodos. Al avanzar una iteración más, la configuración retorna al estado de la configuración inicial.

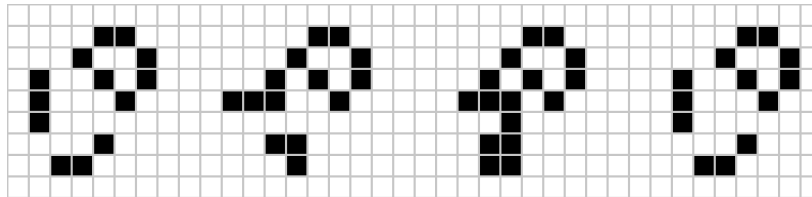


Figura 4.9: De izquierda a derecha, evolución síncrona de la configuración *jam*.

En esta configuración se observa el mismo comportamiento para los promedios de área y calor que para la configuración *toad*. Por el contrario, para el número medio de clústeres (Figura 4.10 y Figura 4.11) se tiene el comportamiento opuesto, esto es, el valor decrece con el crecimiento de α , siendo su máximo valor cercano a 2.7 clústeres para $\alpha = 0.15$ y su mínimo próximo 2 clústeres para $\alpha = 0.9$.

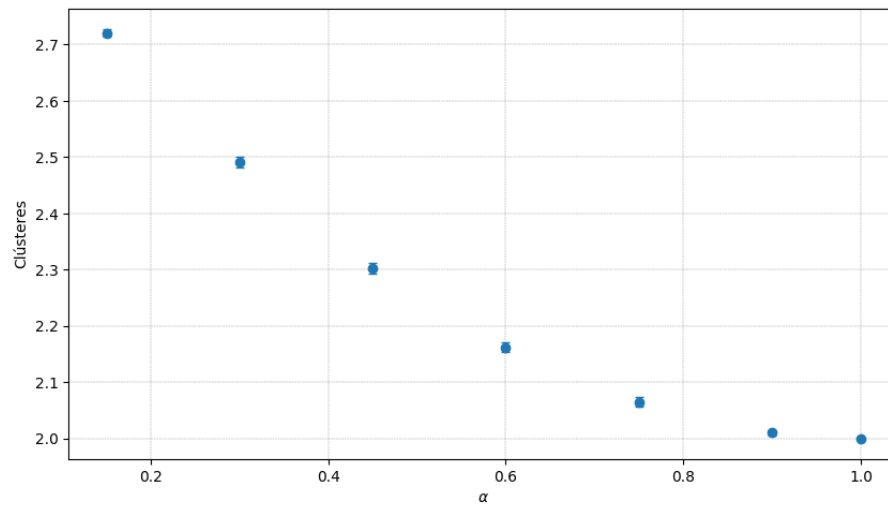


Figura 4.10: Evolución α -asíncrona del promedio de clústeres de la configuración *jam*.

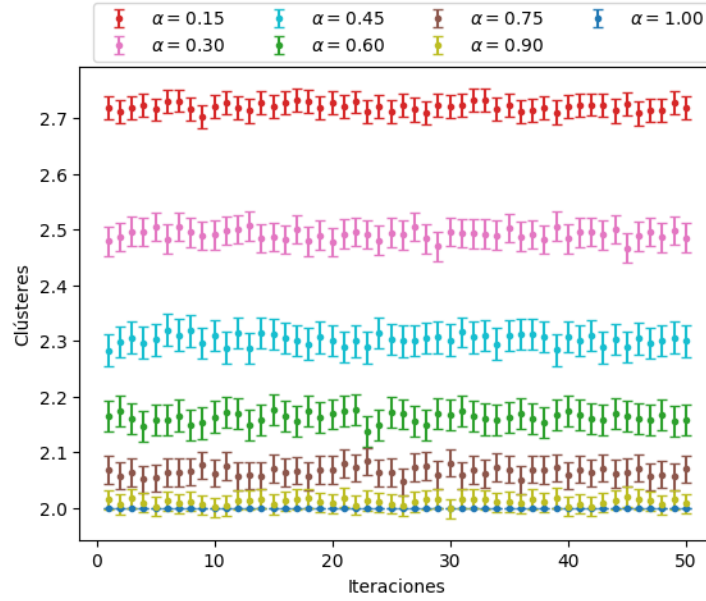


Figura 4.11: Evolución α -asíncrona del promedio de clústeres de la configuración *jam*.

A continuación, en la [Figura 4.12](#) podemos observar tres iteraciones de la configuración *pulsar*, la cual es notablemente la configuración de mayor tamaño que hemos estudiado. En situación de actualización síncrona esta configuración inicial está formada por 48 nodos ocupados agrupados en 12 clústeres y ocupa un área de 169 nodos^2 . En su primera iteración se incrementa el número de nodos a 72, agrupados en el mismo número de clústeres y con un calor de 32 nodos, incrementando el área a 225 nodos^2 . La siguiente iteración retorna su área a 169 nodos^2 , manteniendo el número de nodos ocupados, agrupados en 4 clústeres con un calor de 40 nodos. Al avanzar una iteración más, la configuración retorna al estado de la configuración inicial con un calor de 56 nodos.

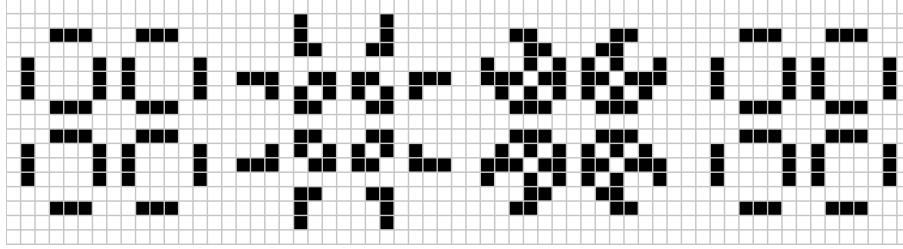


Figura 4.12: De izquierda a derecha, evolución síncrona de la configuración *pulsar*.

Esta configuración experimenta el mismo comportamiento en el calor medio que la configuración *jam*. Por otra parte la variación del área media (Figura 4.13) es similar, notando que la separación entre los valores medios crece respecto de α , alcanzando para $\alpha = 0.9$ un valor medio próximo a los 225 *nodos*² de la primera iteración síncrona de esta configuración.

El número medio de clústeres (Figura 4.14 y Figura 4.15) decrece de $\alpha = 0.15$ a $\alpha = 0.6$ una unidad, a continuación crece 1.25 clústeres de $\alpha = 0.6$ a $\alpha = 0.9$. Además en la (Figura 4.16 y Figura 4.17) el número de vidas inmóviles crece casi 1.5 unidades respecto de α .

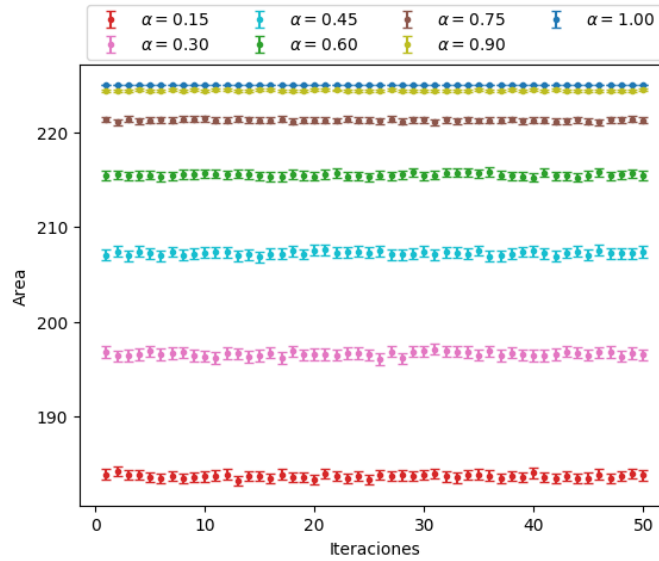


Figura 4.13: Evolución α -asíncrona del promedio de área de la configuración *pulsar*.

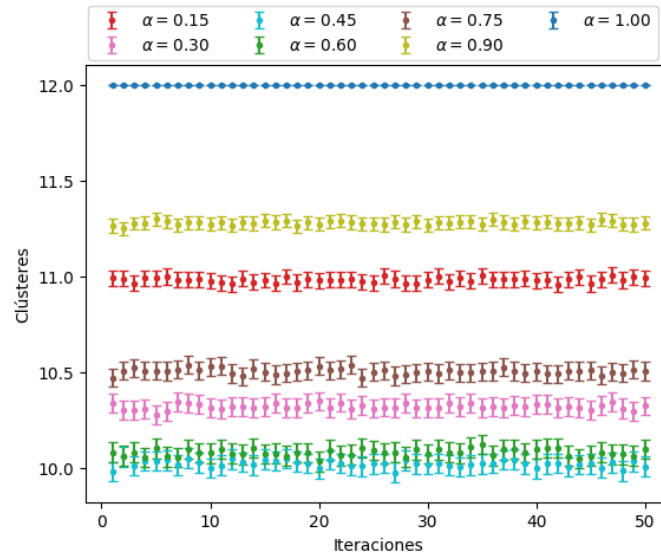


Figura 4.14: Evolución α -asíncrona del promedio de clústeres de la configuración *pulsar*.

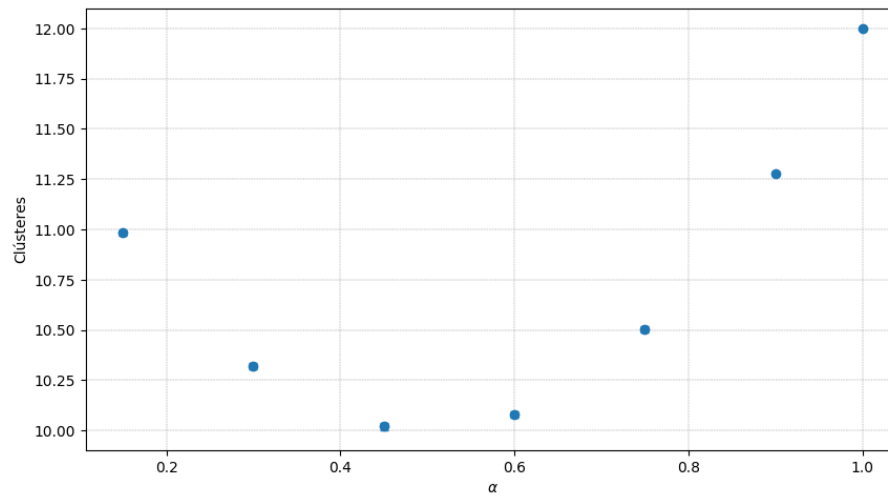


Figura 4.15: Evolución α -asíncrona del promedio de clústeres de la configuración *pulsar*.

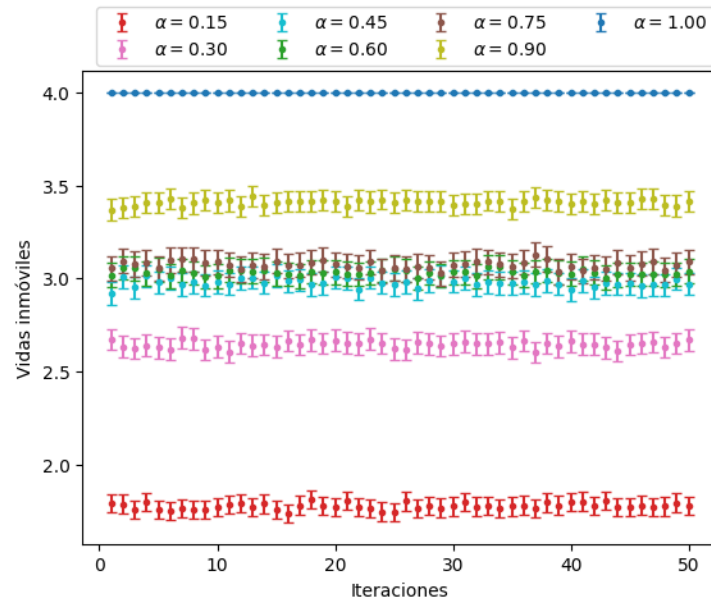


Figura 4.16: Evolución α -asíncrona del promedio de vidas inmóviles de la configuración *pulsar*.

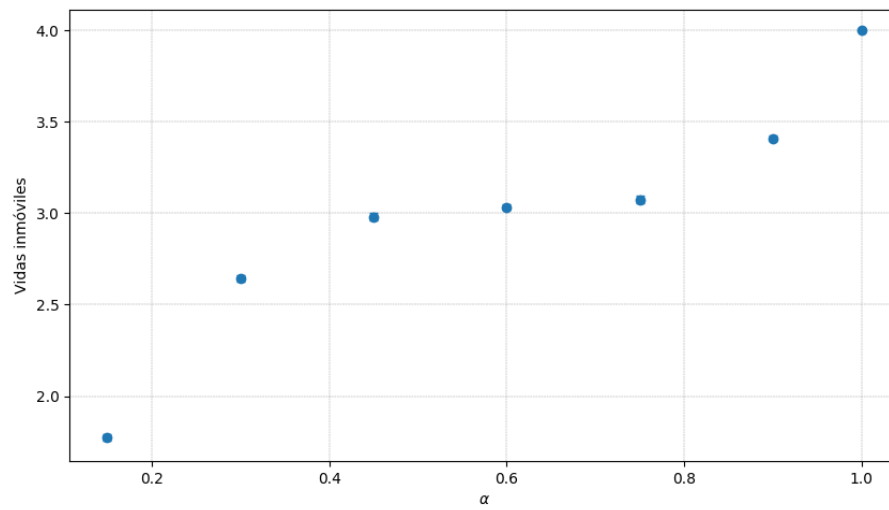


Figura 4.17: Evolución α -asíncrona del promedio de vidas inmóviles de la configuración *pulsar*.

Osciladores de periodo 4

En esta sección estudiaremos la variación α -asíncrona de las configuraciones *mold* y *mazing*. En la [Figura 4.18](#) es posible observar la 4 iteraciones de la evolución en condiciones de sincronicidad de la configuración *mold*. Esta configuración inicial cuenta con 12 nodos ocupados, agrupados en dos clústeres con un área de 36 nodos^2 . En su primera iteración genera un calor de 8 nodos, variando tanto el número de nodos ocupados como el área que ocupan, a 14 nodos ocupados y 30 nodos^2 , agrupados en un solo cluster. Esta iteración genera un calor de 6 nodos. La siguiente iteración la configuración retorna al mismo área, nodos ocupados y clústeres de la configuración inicial pero en diferente disposición, con un calor de 8. La tercera iteración ocupa un área de 30 nodos^2 , que se reparten 14 nodos ocupados en un único clúster con un calor de 6. En la cuarta iteración la configuración coincide con la configuración inicial con un calor de 8 nodos.

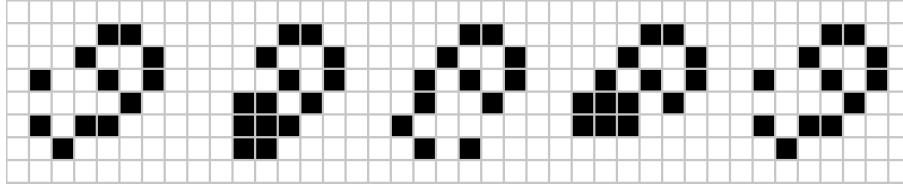


Figura 4.18: De izquierda a derecha, evolución síncrona de la configuración *mold*.

Esta configuración tiene en común con la configuración *blinker* que el promedio de clústeres permanece constante en la unidad. Además el calor medio se comporta de la misma manera que en la configuración *blinker*, solo que en lugar de crecer aproximadamente 0.5 nodos, crece 0.75 nodos respecto de α .

El área media varía de una forma muy curiosa ([Figura 4.19](#)). En lugar de crecer con el valor de α , decrece obteniendo el mayor valor, 36 nodos^2 , para $\alpha = 0.15$ y el menor, 30.5 nodos^2 , para $\alpha = 0.9$. Además el crecimiento respecto de α es constante, aproximadamente una unidad. Análogamente, el número medio de nodos ocupados crece respecto de α aproximadamente una unidad, alcanzando su máximo, 14 nodos ocupados, para $\alpha = 0.15$ y su mínimo, 11.5 nodos ocupados, para $\alpha = 0.9$ ([Figura 4.20](#)).

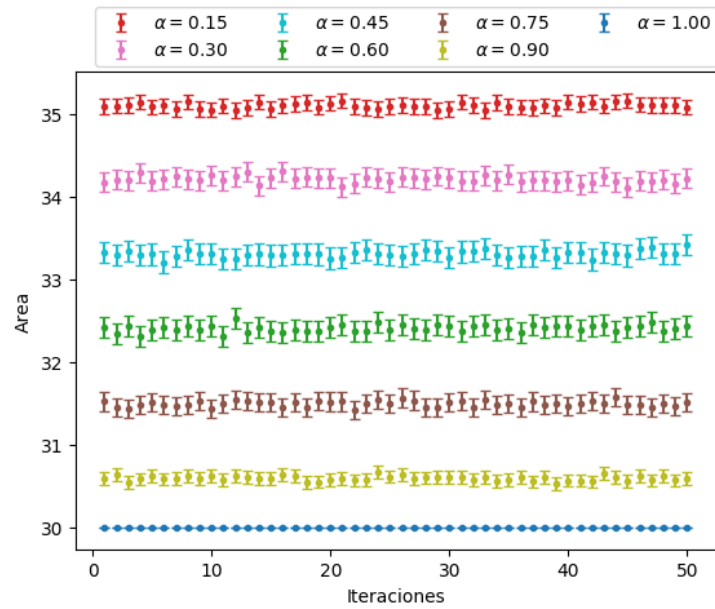


Figura 4.19: Evolución α -asíncrona del área media de la configuración *mold*.

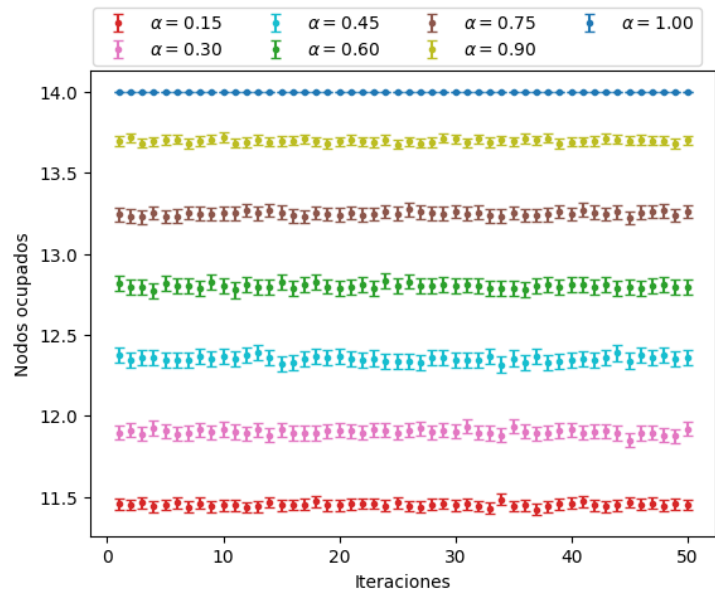


Figura 4.20: Evolución α -asíncrona del promedio de nodos ocupados de la configuración *mold*.

En la [Figura 4.21](#) es posible observar la 4 iteraciones de la evolución en condiciones de sincronización de la configuración *mazing*. Esta configuración inicial cuenta con 12 nodos ocupados, agrupados en 4 clústeres con un área de 49 *nodos*². En su primera iteración genera un calor de 14 nodos, variando tanto el número de nodos ocupados como el de clústeres a 18 nodos ocupados y un clúster, manteniendo el área anterior. Esta iteración genera un calor de 14 nodos. La siguiente iteración la configuración retorna al mismo área, nodos ocupados y clústeres de la configuración inicial pero en diferente disposición, con un calor de 14 nodos. La tercera iteración tiene el mismo área, nodos ocupados y clústeres de la primera iteración pero en diferente disposición con un calor de 14 nodos. En la cuarta iteración la configuración coincide con la configuración inicial.

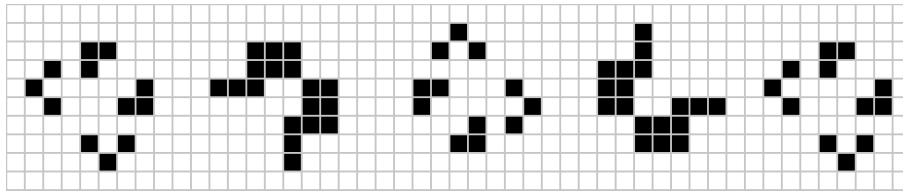


Figura 4.21: De izquierda a derecha, evolución síncrona de la configuración *mazing*.

En esta configuración destaca el hecho de que el área permanezca constante en 49 *nodos*² independientemente al valor de α . Por otra parte tanto el calor medio como el número de nodos ocupados tienen el mismo comportamiento, crecen respecto de α , la única diferencia es que en el primero el incremento respecto de α es de dos unidades y en el segundo de tan solo una unidad. Dicho comportamiento es común a la configuración anterior, *mold*.

El promedio de vidas inmóviles decrece respecto de α pero no lo hace en la misma medida que el calor medio, si no que para los valores de $\alpha \leq 0.6$ es prácticamente el mismo decrecimiento, siendo el valor máximo $\alpha = 0.15$ con algo más de 1.5 vidas inmóviles en media ([Figura 4.23](#) y [Figura 4.22](#)). Sin embargo a partir de este valor, el incremento disminuye rápidamente y el promedio de vidas inmóviles se aproxima a 0.

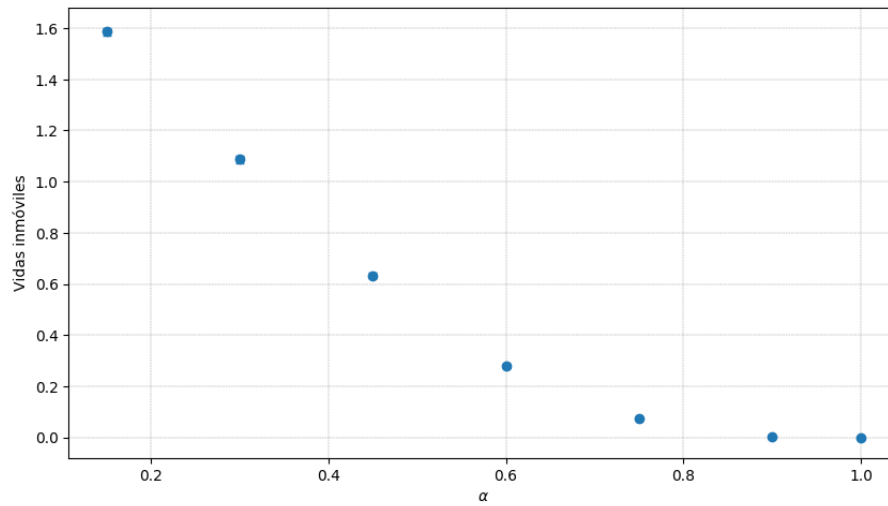


Figura 4.22: Evolución α -asíncrona del promedio de vidas inmóviles de la configuración *mazing*.

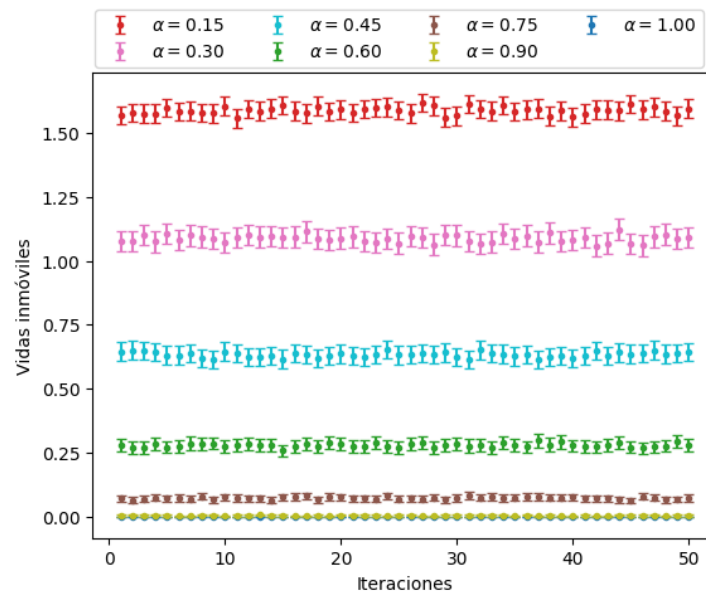


Figura 4.23: Evolución α -asíncrona del promedio de vidas inmóviles de la configuración *mazing*.

Por último, el número medio de clústeres decrece respecto de α (Figu-

ra 4.24) pero de nuevo no es en la misma medida que el calor medio, si no que de $\alpha = 0.15$ a $\alpha = 0.3$ y de $\alpha = 0.75$ a $\alpha = 0.9$ el incremento es aproximadamente el mismo, 0.5 clústeres y los decrementos para los valores restantes de α son aproximadamente 0.25 clústeres.

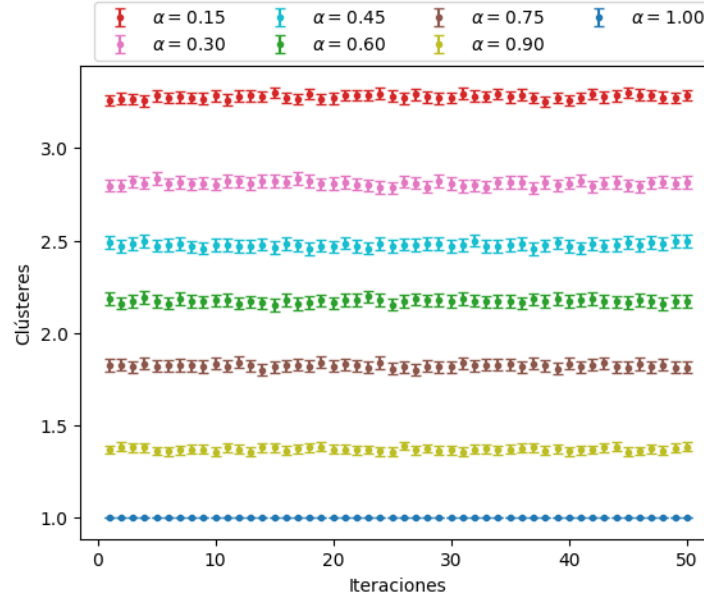


Figura 4.24: Evolución α -asíncrona del promedio de clústeres de la configuración *mazing*.

Naves espaciales

Hasta ahora solo hemos descrito el comportamiento de *osciladores*. Como se expuso en la sección 2.3, las *naves espaciales* pueden ser vistas como *osciladores* que se desplazan, luego es interesante explorar si los comportamientos que hemos observado en las configuraciones iniciales anteriores se reproducen en este tipo de configuraciones iniciales.

Comenzamos la sección estudiando la configuración inicial *lightweight spaceship* (Figura 2.4b). En la Figura 4.25 se muestran 4 iteraciones de esta configuración inicial. Se trata de una configuración de velocidad $c/2$ que se desplaza paralelamente al eje horizontal con un área constante de 20 *nodos*². En las iteraciones pares tiene 9 nodos ocupados agrupados en dos clústeres con un calor de 9 nodos y en las iteraciones impares tiene 12 nodos agrupados en un solo clúster con un calor de 13 nodos.

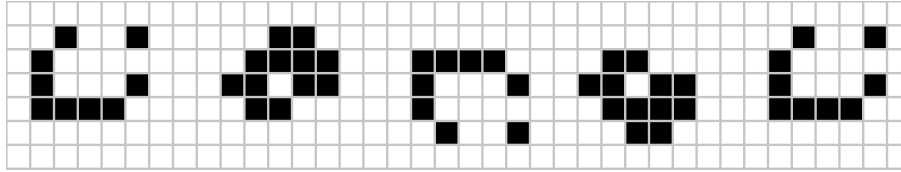


Figura 4.25: De derecha a izquierda, evolución síncrona de la configuración *lightweight spaceship*.

Uno de los característicos efectos que produce la introducción de α -asincronismo en esta configuración inicial es que el número medio de clústeres permanece constante en todas las ejecuciones como en la configuración *blinker*. Además el número de vidas inmóviles es constantemente 0 independientemente al valor de α . Si observamos el número medio de nodos ocupados (Figura 4.26) se observa que la separación vertical de los valores medios constantes es aproximadamente igual a 0.5 nodos ocupados entre valores consecutivos de α . De esta manera observamos que cuando el valor de α el promedio nodos ocupados se acerca a los 12 nodos ocupados en las iteraciones impares de esta configuración con la evolución síncrona.

Mientras que la densidad y el calor medios de esta configuración tienen un comportamiento similar al número medio de nodos ocupados se puede observar una variación diferente en el área media (Figura 4.27 y Figura 4.28). Cuando α se aproxima, tanto inferior como superiormente, a 0.5 los valores medios de área se estabilizan entorno al valor 21.2 nodos^2 . Por otro lado cuando α se aproxima a 0 ó a 1, los valores medios se aproximan a 20.5 nodos^2 , un valor muy cercano a los 20 nodos^2 de la configuración con evolución síncrona. Otra cuestión notable es que el valor de área media varía en el intervalo de una unidad de longitud cuando α varía. A diferencia de la longitud de los intervalos en los que se encuentran los promedios de nodos ocupados, densidad y calor, que es mayor.

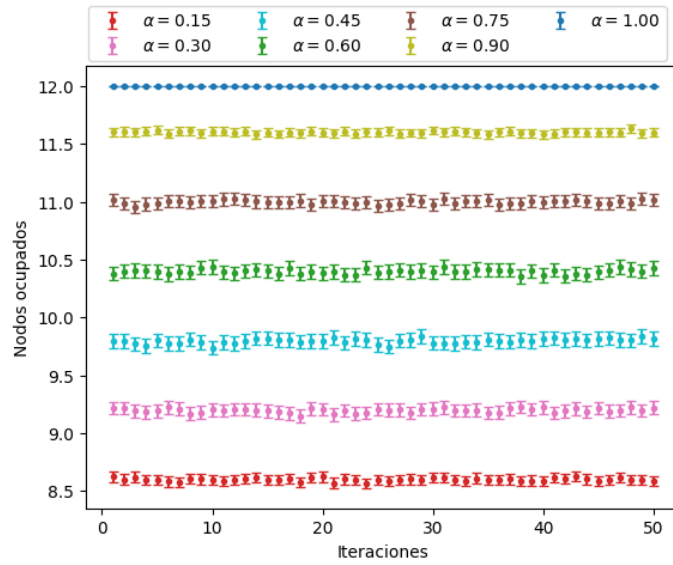


Figura 4.26: Evolución α -asíncrona del promedio de nodos ocupados de la configuración *lightweight spaceship*.

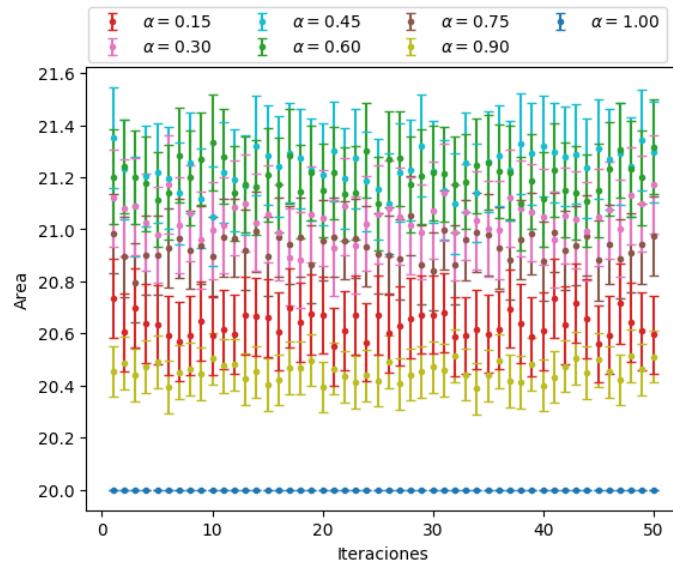


Figura 4.27: Evolución α -asíncrona del área media de la configuración *lightweight spaceship*.

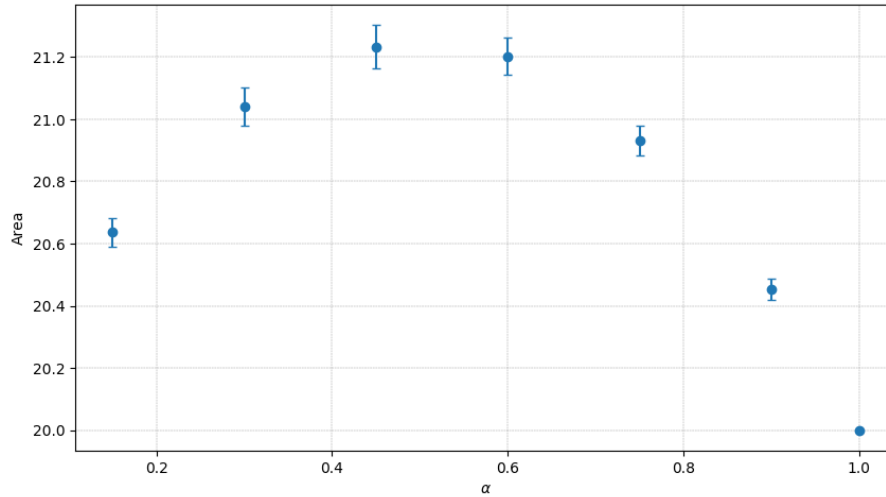


Figura 4.28: Evolución α -asíncrona del área media de la configuración *lightweight spaceship*.

La configuración inicial *middleweight spaceship* (Figura 2.5a) tiene un aspecto y una evolución síncrona similar a la configuración *lightweight spaceship*. En la Figura 4.29 se muestran 4 iteraciones de esta configuración inicial. Se trata de una configuración de velocidad $c/2$ que se desplaza paralelamente al eje horizontal con un área constante de 24 nodos^2 . En las iteraciones pares tiene 11 nodos ocupados agrupados en tres clústeres con un calor de 12 nodos y en las iteraciones impares tiene 15 nodos agrupados en un solo clúster con un calor de 18 nodos.

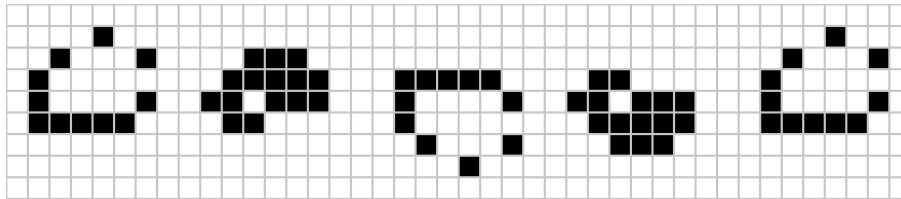


Figura 4.29: De derecha a izquierda, evolución síncrona de la configuración *middleweight spaceship*.

Al igual que en la configuración anterior, *lightweight spaceship*, el número de clústeres y el de vidas inmóviles permanece constante independientemente del valor de α . Y tanto el promedio de nodos ocupados como el de densidad tienen el mismo comportamiento. Sin embargo para el valor medio de área se

produce una variación del comportamiento que se puede observar en la [Figura 4.30](#) y en la [Figura 4.31](#). El área varía prácticamente de la misma manera para $\alpha = 0.15$, 0.75 y a excepción de $\alpha = 0.90$ el resto de valores de α muestran un comportamiento constante muy parecido entre sí.

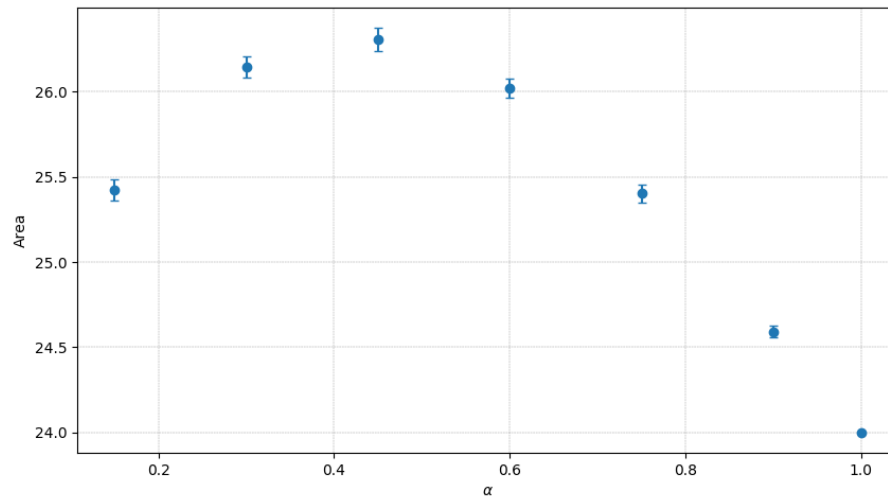


Figura 4.30: Evolución α -asíncrona del área media de la configuración *middleweight spaceship*.

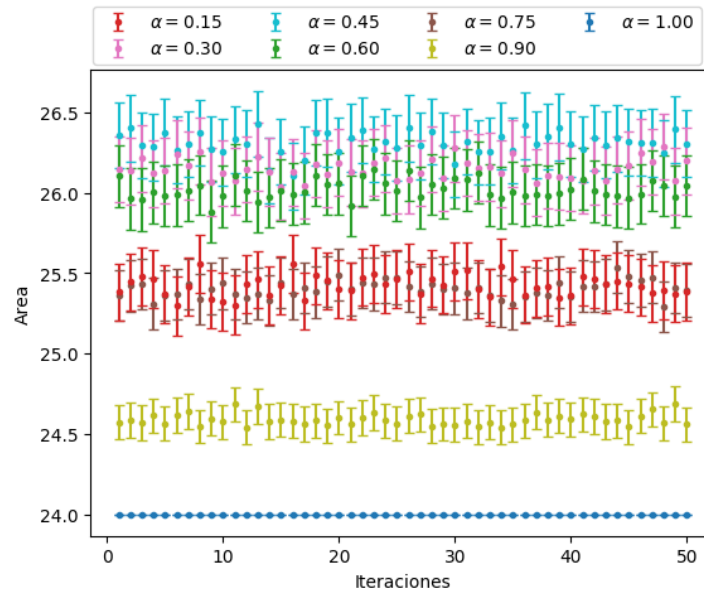


Figura 4.31: Evolución α -asíncrona del área media de la configuración *middleweight spaceship*.

Por otro lado, una diferencia notable con la configuración anterior es que la longitud del intervalo en el que varía el promedio crece aproximadamente una unidad más.

La siguiente configuración que vamos a estudiar, *heavyweight spaceship* muestra un aspecto similar a las dos anteriores tanto en forma como en su evolución síncrona. En la Figura 4.32 se muestran 4 iteraciones de esta configuración inicial. Se trata de una configuración de velocidad $c/2$ que se desplaza paralelamente al eje horizontal con un área constante de 35 nodos^2 . En las iteraciones pares tiene 13 nodos ocupados agrupados en tres clústeres con un calor de 15 nodos y en las iteraciones impares tiene 18 nodos agrupados en un solo clúster con un calor de 23 nodos.

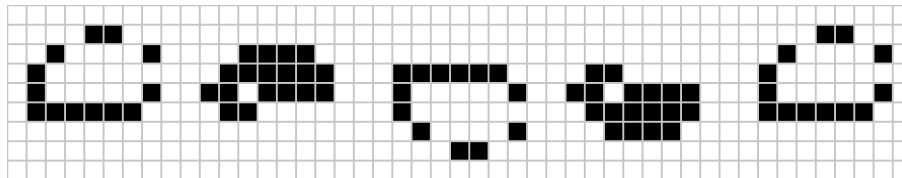


Figura 4.32: De derecha a izquierda, evolución síncrona de la configuración *heavyweight spaceship*.

A diferencia de las anteriores *naves espaciales*, en la configuración *heavy-weight spaceship* el número de clústeres no se mantiene constante independientemente del valor que α tome. Aunque el número de vidas inmóviles si continúa siendo constantemente 0. En la [Figura 4.33](#) es posible observar este cambio, además todos los valores medios se mantienen aproximadamente constantes a partir de la décima iteración. A medida que α incrementa se visualiza que la separación vertical de los valores del promedio de clústeres no es constante como en las *naves espaciales* anteriores ([Figura 4.26](#)) y dicha separación incrementa con el valor de α .

Para el valor medio de área, el intervalo en el que oscila para distintos valores de α crece hasta una longitud de 8 nodos^{-1} , siendo para $\alpha = 0.9$ el menor valor de área, 31 nodos^2 y para $\alpha = 0.30, 0.45$ los mayores valores medios, aproximadamente 39 nodos^2 .

Los promedios de calor y número de nodos ocupados muestran un comportamiento idéntico al que se puede observar en la [Figura 4.26](#).

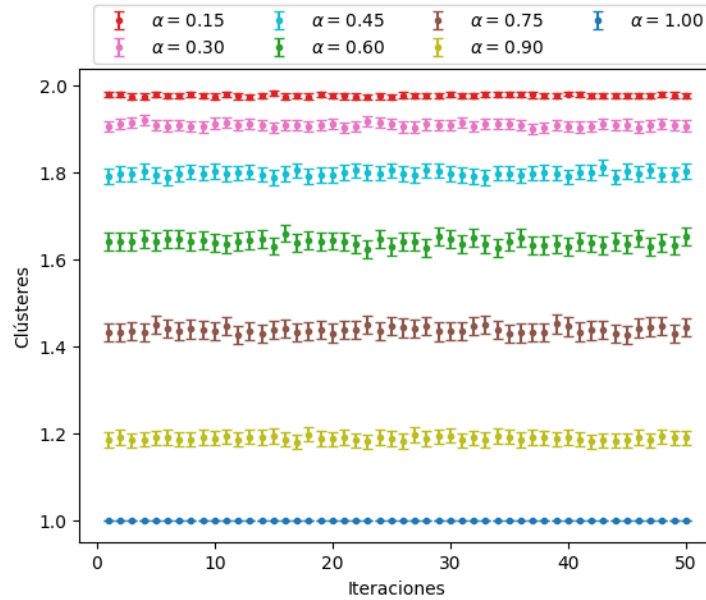


Figura 4.33: Variación del promedio de clústeres en la configuración *heavy-weight spaceship* para distintos valores de α .

Por último, la configuración inicial *glider* es una nave espacial de velocidad $c/4$ formada por 5 nodos ocupados agrupados en un único clúster, tiene un calor de 4 *nodos* y ocupa un área de 9 *nodos*² ([Figura 4.34](#)). De esta figura

destacamos que el calor medio se desarrolla de forma similar al que hemos observado para la configuración *blinker* al igual que lo hace el área media pero en intervalos de mayor longitud (Figura 4.35 y Figura 4.36). Mientras que los promedios restantes varían sutilmente en intervalos de muy reducida longitud.

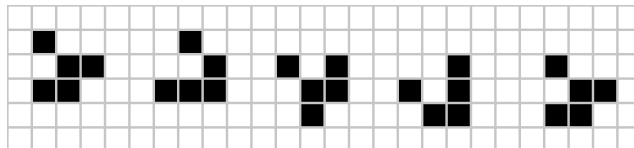


Figura 4.34: De izquierda a derecha, evolución síncrona de la configuración *glider*.

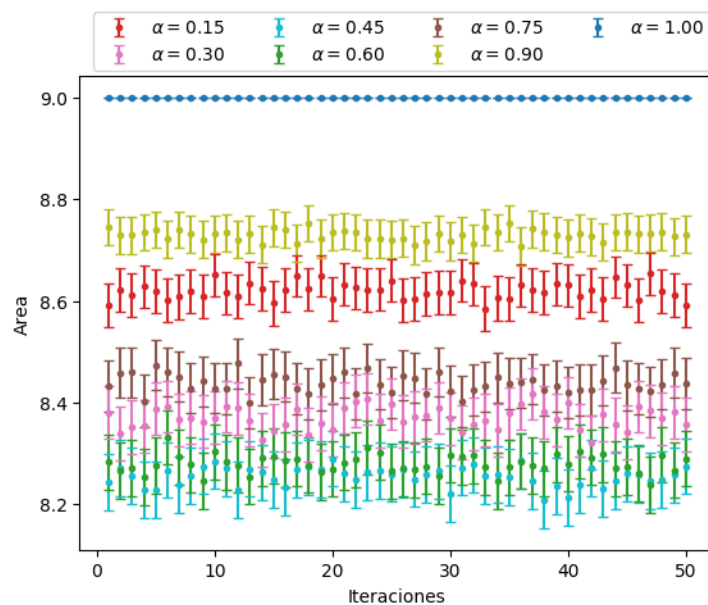


Figura 4.35: Evolución α -asíncrona del área media de la configuración *glider*.

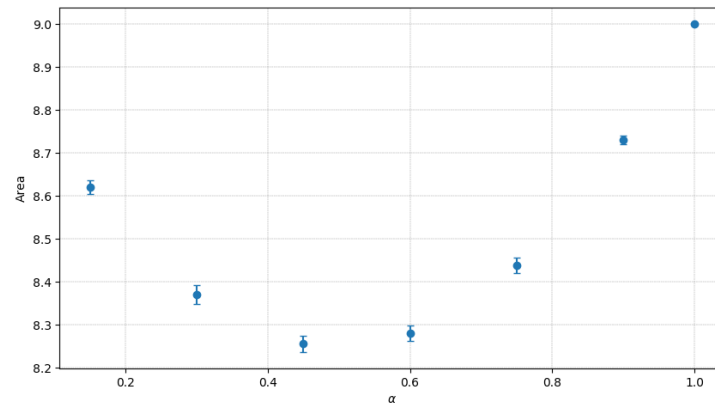


Figura 4.36: Evolución α -asíncrona del área media de la configuración *glider*.

5

Conclusiones

En este trabajo se ha estudiado la evolución α -asíncrona de tres tipos de configuraciones del juego de vida de Conway, mediante las técnicas de simulación Monte Carlo.

Una de las cuestiones más interesantes que se ha observado en todas las ejecuciones es que los promedios de las variables medidas alcanzan un valor aproximadamente constante para cada valor escogido de α , independientemente del número de iteraciones. A pesar de que podría introducir inestabilidad, el α -asíncronismo induce cierta *estabilidad* en las variables observadas, concluyendo que las configuraciones reproducen un esquema de auto-organización, hecho que ya se había puesto de manifiesto para configuraciones iniciales aleatorias.

A menudo se ha observado que cuando α se aproxima a la unidad es habitual que los valores medios estudiados se aproximen a los valores que se obtienen en un esquema de actualización síncrono, mientras que por lo contrario cuando α se aproxima a cero los comportamientos son diversos y en general se alejan del comportamiento síncrono.

Otra observación de interés es que cuando α decrece se tiende a ralentizar el número de cambios de estados en los nodos y que por el contrario cuando α se aproxima a la unidad la actividad crece. Este fenómeno se ve reflejado en el comportamiento global del calor medio sobre las configuraciones estudiadas, lo que confirma la intuición de que cuanto menor sea la probabilidad de actualizar los nodos, menor será el cambio de estado de los mismos.

En general se puede observar que de las categorías propuestas de configuraciones, solo se preservan las vidas inmóviles debido a su carácter estable. Por otra parte, hemos observado la evolución α -asíncrona de las *naves espaciales* estudiadas se puede caracterizar por la ausencia de vidas inmóviles puesto que eso no ocurre en los otros tipos de configuraciones iniciales estudiadas.

Como trabajo futuro, proponemos analizar más configuraciones de cada categoría, añadir categorías nuevas que permitan recoger comportamientos más complejos así como un tratamiento estadístico más completo. También

atrae nuestro interés la profundización del trabajo a través de la observación de más variables que sean computacionalmente más costosas. Por ejemplo, se puede realizar un censo de los clústeres de configuraciones más frecuentes, las cuales nos permitan realizar una extrapolación del censo utilizado en éste trabajo a la situación de actualización α -asíncrona.

Bibliografía

- [1] Edward F Moore. Machine models of self-reproduction. 14(1962):17–33, 1962.
- [2] Martin Gardner. Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, 223:120–123, 1970.
- [3] Jan Hemmingsson. Consistent results on 'life'. *Physica D: Nonlinear Phenomena*, 80(1-2):151–153, 1995.
- [4] Hendrik J Blok and Birger Bergersen. Effect of boundary conditions on scaling in the "game of life". *Physical Review E*, 55(5):6249–6252, 1997.
- [5] Carter Bays. Cellular automata in the triangular tessellation. *Complex Systems*, 8(2):127, 1994.
- [6] Nick Owens and Susan Stepney. Investigations of game of life cellular automata rules on penrose tilings: Lifetime, ash, and oscillator statistics. *J. Cellular Automata*, 5(3):207–225, 2010.
- [7] Clifford A Reiter. The game of life on a hyperbolic domain. *Computers & Graphics*, 21(5):673–683, 1997.
- [8] Conway's Game of Life - Boardless approach. https://web.archive.org/web/20190212184430/https://rosettacode.org/wiki/Conway's_Game_of_Life#Boardless_approach. [Online; accessed 14-February-2019].
- [9] Stephen Wolfram. *Celular Automata and Complexity*. Addison-Wesley, 1994.
- [10] Harold V McIntosh. Wolfram's class iv automata and a good life. *Physica D: Nonlinear Phenomena*, 45(1-3):105–121, 1990.
- [11] Paul Rendell. *Turing universality of the game of life*. Springer, 2002.
- [12] A Turing Machine in Conway's Game of Life, extendable to a Universal Turing Machine. <https://web.archive.org/web/20190118023640/>

- <http://rendell-attic.org/gol/tm.htm>. [Online; accessed 14-February-2019].
- [13] Build a working game of Tetris in Conway's Game of Life - StackExchange. <https://web.archive.org/web/20190120082452/https://codegolf.stackexchange.com/questions/11880/build-a-working-game-of-tetris-in-conways-game-of-life>. [Online; accessed 14-February-2019].
- [14] Jean-Philippe Rennard. *Implementation of logical functions in the Game of Life*. Springer, 2002.
- [15] Nazim A Fatès and Michel Morvan. An experimental study of robustness to asynchronism for elementary cellular automata. *arXiv preprint nlin/0402016*, 2004.
- [16] Birgitt Schönfisch and André de Roos. Synchronous and asynchronous updating in cellular automata. *BioSystems*, 51(3):123–143, 1999.
- [17] Birger Blok, Hendrik J.; Bergersen. Synchronous versus asynchronous updating in the “game of life”. *Physical Review E*, 59:3876–3879, 1999.
- [18] Nazim Fates. Critical phenomena in cellular automata: perturbing the update, transitions, the topology. *Acta Physica Polonica B*, 3:315–325, 2010.
- [19] Alberto Dennunzio, Enrico Formenti, Luca Manzoni, and Giancarlo Mauri. m-asynchronous cellular automata: from fairness to quasi-fairness. *Natural Computing*, 12(4):561–572, 2013.
- [20] Martin Gardner. *Wheels, life, and other mathematical amusements*. 86, 1983.
- [21] David Eppstein. *Growth and decay in life-like cellular automata*. Springer, 2010.
- [22] Quicklife algorithm implementation. <https://sourceforge.net/p/golly/code/ci/master/tree/gollybase/qlifealgo.cpp>. [Online; accessed 2-Jun-2019].
- [23] R Wm Gosper. Exploiting regularities in large cellular spaces. *Physica D: Nonlinear Phenomena*, 10(1-2):75–80, 1984.
- [24] Noam David Elkies. The still-life density problem and its generalizations. *Voronoi's Impact on Modern Science, Book I*, 1998.

- [25] Matthew Cook. Still life theory. *New Constructions in Cellular Automata*, 226, 2003.
- [26] Still Life - lifewiki. http://www.conwaylife.com/wiki/Still_life. [Online; accessed 21-May-2019].
- [27] Number of stable n-celled patterns still lifes in Conway's game of Life. <https://oeis.org/A019473>. [Online; accessed 21-May-2019].
- [28] Game of life lexicon. http://www.conwaylife.com/ref/lexicon/lex_e.htm#eater. [Online; accessed 2-Jun-2019].
- [29] ER Berlekamp, JH Conway, and RK Guy. What is life?, chapter 25. *Winning Ways for Your Mathematical Plays*, 2, 1982.
- [30] Run length encoded - LifeWiki. http://www.conwaylife.com/wiki/Run_Length_Encoded. [Online; accessed 2-Jun-2019].
- [31] The Online Life-Like CA Soup Search - lifewiki. http://www.conwaylife.com/wiki/Nathaniel%27s_census. [Online; accessed 21-May-2019].
- [32] Achim Flammenkamp's census - LifeWiki. http://www.conwaylife.com/wiki/Achim_Flammenkamp%27s_census. [Online; accessed 21-May-2019].
- [33] Catagolue database. <https://catagolue.appspot.com/home>. [Online; accessed 2-Jun-2019].
- [34] The JSON Data Interchange Syntax. <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>. [Online; accessed 2-Jun-2019].
- [35] Common Format and MIME Type for Comma-Separated Values (CSV) Files. <https://tools.ietf.org/html/rfc4180>. [Online; accessed 2-Jun-2019].
- [36] Allan Gut. *Probability: a graduate course*, volume 75. Springer Science & Business Media, 2013.
- [37] David Williams. *Probability with martingales*. Cambridge Mathematical Textbooks. Cambridge University Press, 1991.
- [38] Michel Loeve. *Probability theory I*. Springer, 4th edition, 1977.
- [39] Donald E. Knuth. *The art of computer programming II*. Addison-Wesley Professional, 3 edition, 1997.

- [40] George Marsaglia. Random numbers fall mainly in the planes. *Proceedings of the National Academy of Sciences of the United States of America*, 61(1):25, 1968.
- [41] Dieharder: A Random Number Test Suite. http://web.archive.org/web/20190402181134/http://webhome.phy.duke.edu/~rgb/General/rand_rate.php. [Online; accessed 02-April-2019].
- [42] Pierre L'Ecuyer and Richard Simard. Testu01: A library for empirical testing of random number generators. *ACM Transactions on Mathematical Software (TOMS)*, 33(4):22, 2007.
- [43] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.
- [44] Python 2.7.16 Documentation. <http://web.archive.org/web/20170907114211/https://docs.python.org/2.7/library/random.html>. [Online; accessed 29-April-2019].
- [45] R Documentation, random number generation. <https://web.archive.org/web/20190106064055/https://stat.ethz.ch/R-manual/R-devel/library/base/html/Random.html>. [Online; accessed 29-April-2019].
- [46] SS Shapiro and Wilk MB. An analysis of variance test for normality. *Biometrika*, 52, 1965.
- [47] Nornadiah Mohd Razali, Yap Bee Wah, et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33, 2011.