



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



Escola Tècnica  
Superior d'Enginyeria  
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica  
Universitat Politècnica de València

# Modelado computacional de pandemias mediante autómatas celulares

Trabajo Fin de Grado

**Grado en Ingeniería Informática**

**Autor:** Sánchez Gómez, Pablo

**Tutor:** José María Sempere Luna

Curso 2020-2021

## Resumen

El objetivo de este TFG es desarrollar un modelo computacional que sea capaz de usar autómatas celulares para la predicción de la evolución de pandemias sanitarias y los factores que influyen en las mismas. Para este tipo de predicciones se suele usar modelos estadísticos para inferir a partir de datos recogidos en situaciones sanitarias parecidas previamente y poder así estimar la evolución de la situación actual, pero en este TFG usaremos Autómatas celulares para realizar una predicción de la evolución de una pandemia para una delimitada población dados los parámetros del virus.

**Palabras clave:** autómata celular, pandemia, predicción, evolución, modelo computacional.

---

## Abstract

The main aim of this project is to develop a computational model able to use cellular automata to predict the evolution of sanitary pandemic and the different factors involved in it. For this kind of predictions statistical models are usually used with data collected from previous similar situations in order to try to infer an estimation of the evolution of the current situation, but in this project we are going to be using cellular automata to make this prediction of the evolution of the pandemic for a determined population given certain parameters of a given virus.

**Key words:** cellular automata, pandemic, prediction, evolution, computational model.



# Estructura del TFG

---

## **1 Introducción**

- 1.1 Motivación
- 1.2 Problemas Planteados
- 1.3 Soluciones propuestas
- 1.4 Estructura de la memoria

## **2 Modelo epidemiológico: El modelo SIR**

## **3 Autómatas celulares**

- 3.1 Conceptos básicos
- 3.2 Autómatas bidimensionales
- 3.3 El juego de la vida

## **4 Descripción de la solución**

- 4.1 Lenguaje de programación
- 4.2 Arquitectura del proyecto
- 4.3 Diseño final

## **5 Presentación de la herramienta**

- 5.1 Reglas de infección
- 5.2 Reglas de movilidad

## **6 Ejemplos de escenarios: ejecuciones y gráficas**

## **7 Conclusiones**

- 7.1 Simulación covid-19
- 7.2 Posibles mejoras futuras
- 7.3 Conclusiones finales

## **Bibliografía**



---

# Capítulo 1

## Introducción

---

En este capítulo se expone la motivación del trabajo realizado, los problemas planteados antes de realizarlo, las soluciones a las que se ha llegado y, por último, la estructura que sigue la memoria.

### 1.1 Motivación

---

Si hace apenas unos años nos hubiesen dicho que en 2020 y 2021 nuestro mundo lleno de libertades y comodidades se vendría abajo, llegando incluso a pararse durante meses completos, yo al menos hubiera tildado al emisor de loco. Desgraciadamente, conceptos otrora tan ajenos a nosotros como llevar una mascarilla por la calle, la cancelación del ocio e incluso tener que salir de la calle bajo amenaza de multa por un impuesto toque de queda son y han sido durante este último año nuestro día a día.

A día de hoy todas nuestras vidas se han visto alteradas por la irrupción de una pandemia sanitaria que ha bloqueado al mundo, por eso al ver este TFG listado y la relevancia en la actualidad del mismo decidí escogerlo. Poder desarrollar un trabajo que pueda tener una utilidad real para poder predecir escenarios de pandemias en una determinada población es algo que hubiese podido ayudar en demasía a nuestros expertos y epidemiólogos a la hora de elaborar las medidas necesarias para evitar la propagación y daño de la pandemia del COVID-19. Por ello, realizar este TFG como prueba de concepto acerca de la validez de un modelo computacional de predicción de la evolución de pandemias mediante autómatas celulares se propuso como una idea irrechazable.

Aparte, dentro de la educación recibida en la rama de computación, una de los conceptos más interesantes y menos profundizados vistos en clases fueron los autómatas celulares, apenas si pudimos trabajar con ellos en una práctica y me resultaron muy interesantes siendo además particularmente aptos para un trabajo como este por motivos que explicaré en secciones posteriores de esta memoria del trabajo realizado.

### 1.2 Problemas planteados

---

Obviamente la tarea de predecir de manera exacta una pandemia es básicamente imposible y se ha de trabajar con modelos que pueden en la medida de lo posible realizar estimaciones medianamente precisas para poder tener unos datos estimados en torno a los

cuales tomar decisiones. Cómo computar todas y cada una de los escenarios y variables posibles que se puedan dar durante una pandemia es completamente imposible, el mayor problema a solventar con este trabajo es una discretización y simplificación realista de los escenarios posibles para poder trabajar sobre ellos en el proyecto, además teniendo en cuenta las tecnología a aplicarse.

Otro problema que se planteó es la inexistencia de bibliotecas de uso específico en python desarrolladas para trabajar con autómatas celulares, por ello el grueso de la parte técnica del proyecto irá encaminada a programar los autómatas celulares y todo lo tocante al uso de los mismos. Se decidió descartar el Wolfram Mathematica para el desarrollo del proyecto usando python en su lugar por las limitaciones del Mathematica para mostrar los resultados, buscándose así un lenguaje de propósito general con el que poder desarrollar la herramienta de una manera más personalizada y pudiendo así además adentrarnos en el desarrollo de los autómatas celulares con el propósito específico de poder predecir pandemias en vez de intentar adaptar algún modelo ya existente en Mathematica y tratar de personalizarlo para que se ajuste al conjunto de reglas y estados que se requieren en este proyecto.

## 1.3 Soluciones propuestas

---

Para los problemas anteriormente planteados, se ha decidido simplificar el modelo de pandemias para recoger algunos estados posibles creándose una celdilla en la que cada celda individual representa alguno de estos estados, siendo los estados posibles, celda vacía, individuo sano o individuo contagiado. Aparte de este grupo inicial de estados se ha enriquecido en proyecto con otros estados para dar algo más de realismo al trabajo, esto es que un individuo puede estar contagiado y ser asintomático además que un individuo contagiado puede ingresar al hospital, tanto en el hospital como en casa un individuo contagiado tendrá una determinada probabilidad de fallecer añadiendo así otro estado posible, estos estados así como las interacciones entre los mismos será fruto de estudio en la sección 5 de esta memoria donde veremos las reglas de infección, movilidad y distintas ocurrencias que se puedan llegar a dar en la rejilla.

En cuanto a las dificultades en el apartado técnico se ha decido usar python con la ayuda de diversas bibliotecas para el trabajo con matrices, así como la presentación de las mismas, por último también se ha hecho uso de otra biblioteca para poder pedir los datos mediante una sencilla interfaz en vez de por consola o directamente por código. Con esto, ha sido mía la tarea de generar y hacer funcionar de manera correcta los autómatas celulares, el desarrollo de este proyecto será explicado en profundidad en la sección 4 de esta memoria.

## 1.4 Estructura de la memoria

---

En esta memoria y una vez concluida esta introducción, presentaré primero el modelo SIR, un modelo sencillo de extenso uso técnico capaz de capturar muchas de las características típicas de los brotes epidémicos. Una vez visto este modelo, explicaré en la tercera sección de esta memoria la tecnología usada para llevar a cabo el proyecto, esto es una introducción a los autómatas celulares, sus características, funcionamiento y los específicos usados en este trabajo. Luego de introducida la tecnología de desarrollo, nos centraremos en el desarrollo en sí de la solución al problema planteado, en donde discutiremos la elección del lenguaje de programación así como sus bibliotecas de soporte usadas y presentaré la arquitectura del código desarrollado así como el diseño del trabajo realizado. A continuación veremos los específicos de la herramienta desarrollada, esto es las reglas de infección, de movilidad y las interacciones entre los distintos estados para luego hacer distintas ejecuciones y poder evaluar la validez o carencia de la misma del uso de autómatas celulares con las normas ya citadas para la correcta estimación en la evolución de una pandemia dados unos parámetros elegidos, mostrando también las gráficas que representan el desarrollo de los números de contagiados, individuos sanos, fallecidos, recuperados y demás.

Por último redactaré una conclusión que recoja los resultados obtenidos así como una evaluación de los mismos para poder contestar al motivo del proyecto.





---

## Capítulo 2

# Modelo epidemiológico: El modelo SIR

---

El modelo SIR es un modelo publicado en el artículo [0] de 1927 por William Oglivy Kermack y Anderson Gray McKendrick. Aún con ya casi cien años desde la publicación de este artículo, el modelo SIR sigue de estricta relevancia a día de hoy siendo el modelo usado por excelencia en este campo.

Las siglas del modelo SIR corresponden a las tres poblaciones que han de considerarse para el estudio del mismo: individuos Susceptibles a infectarse, individuos Infectados y individuos Recuperados entrando en este grupo curados y fallecidos. Siendo la población total  $N$  la suma de los tres grupos anteriores. El modelo usará estas compartimentaciones de la población para dividir a los sujetos bajo estudio y describir el flujo entre los grupos.

Un sujeto bajo estudio solo puede pertenecer a uno de los grupos previamente descritos, pero sí puede pasar de un grupo a otro siempre siguiendo el siguiente orden:

$$S \rightarrow I \rightarrow R$$

Es decir, un sujeto susceptible puede infectarse y un infectado puede recuperarse, pero cualquier otro cambio de estado no está contemplado. Este modelo obvia algunos factores existentes en una población realista como la muerte de sujetos dentro de la población bajo estudio por causas ajenas a la epidemia que se esté estudiando.

Para poder inferir sobre estos grupos y poder formular ecuaciones matemáticas para aplicar sobre poblaciones activas se ha de tener también en cuenta la variable tiempo  $t$ , para poder estudiar la población en distintos momentos de una línea temporal sacando las siguientes ecuaciones diferenciales:

$$S'(t) = \frac{dS}{dt} = -\beta SI$$

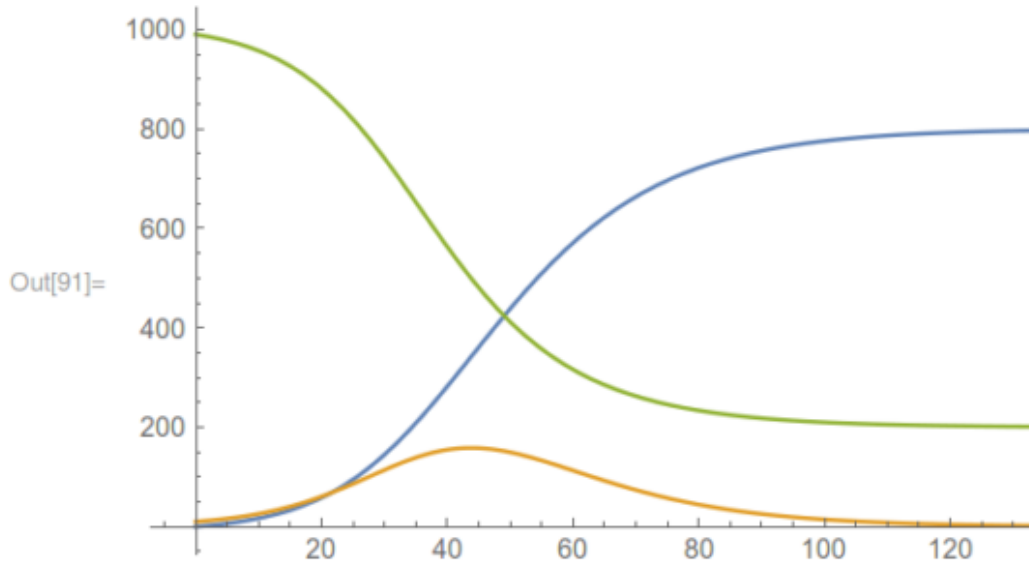
$$I'(t) = \frac{dI}{dt} = +\beta SI - \gamma I$$

$$R'(t) = \frac{dR}{dt} = \gamma I$$

Siendo así  $\beta$  la tasa de transmisión,  $\gamma$  la tasa de recuperación, así  $1/\gamma$  es el período medio de recuperación

Como es fácilmente deducible de las ecuaciones previamente expuestas, en plantear las ecuaciones del modelo no recae la dificultad del mismo, sino en usar unos

valores  $\beta$  y  $\gamma$  apropiados para la epidemia y población en estudio, un ejemplo de una gráfica en donde se muestra la evolución típica del número de pertenecientes a cada grupo durante el tiempo medido puede ser:



Siendo el eje y el número de personas, el x el tiempo en días y los colores siguen un diagrama tal que: **S** (susceptibles) = verde, **I** (infectados) = naranja y **R** (recuperados) = azul

De la gráfica anterior cabe destacar que en  $t = 0$ , tenemos nula incidencia del virus, entre  $t = 40$  y  $t = 50$  tenemos la peor fase de la epidemia y a partir de  $t = 80$  en adelante se denota la inmunidad de rebaño frente al virus (aunque dado que el modelo SRI incluye a los muertos en R, en función de la tasa de mortalidad la población puede haberse visto fuertemente mermada para ese entonces)

---

## Capítulo 3

# Autómatas celulares

---

En este capítulo introduciré la tecnología en la cual se basa el modelo computacional desarrollado, los autómatas celulares, se profundizará en el tipo utilizado para el proyecto y además se introducirá una de las implicaciones más relevantes de esta tecnología a nivel histórico.

### 3.1 Conceptos básicos

---

Para entender qué es un autómata celular y porqué se sugiere que pueda ser útil para simular la evolución de una pandemia, primero haré una breve introducción a la invención historia y propósitos iniciales de esta tecnología.

Esta tecnología data de los cuarenta, concretamente del laboratorio nacional de Los Álamos, Nuevo Méjico donde se atribuye a los investigadores Stanisław Marcin Ulam y John von Neumann la creación del concepto de autómatas celulares inventado al estar Ulman tratando de estudiar el crecimiento de los cristales y von Neumann tanto la creación de robots que crean copias de sí mismos (lo cual puede ser visto como un intento de reproducción del ADN) como el desarrollo conjunto a finales de los cincuenta de un método fiable para calcular el movimiento del líquido. A pesar de que con las investigaciones de von Neumann quizás sea menos obvio relacionar conceptos, tanto en el caso del crecimiento de los cristales como el movimiento del líquido, sí que se puede establecer una correlación en la que se ve aparente que esta tecnología ha sido desde un inicio desarrollada para explicar y simular complejos fenómenos o sistemas naturales, tanto es así que los siguientes desarrollos siguen esta tendencia intentado explicar la conducción de impulsos en sistemas cardíacos que sigue usándose a día de hoy para explicar las arritmias cardíacas [1]. Durante las siguientes décadas se suceden diferentes investigaciones con el uso de autómatas celulares como el conocido Juego de la vida del cual hablaré en el apartado 3.3 hasta que en 2002 con un paper conocido como “Un nuevo tipo de ciencia” [2] con el que Stephen Wolfram trata de argumentar que la recurrencia en los campos de las ciencias naturales tal que se puedan explicar distintos fenómenos mediante autómatas celulares no son hechos aislados sino prueba irrefutable que los autómatas celulares tiene importancia en todos los campos de la ciencia incluyendo ciencias de la computación, física, química, artes (concretamente se han hecho múltiples investigaciones con esta tecnología para la generación de música [3]), matemáticas y en específico el interés de este proyecto, ciencias naturales.

Una vez hecha esta breve introducción a la historia del descubrimiento de los autómatas celulares procederé a explicar los conceptos básicos de lo que es un autómata celular así como una descripción formal.

En su forma más elemental, un autómata celular no es sino una matriz o enrejado de celdas (en la literatura también llamadas células) individuales teniendo estas un determinado estado propio, este enrejado además tendrá una función de transición al siguiente paso del autómata basada en su vecindario, es decir en sus celdas colindantes.

Una definición formal por Alfons G. Hoekstra, Jirí Kroc, y Peter M.A. Sloot [7] para este concepto sería tal que:

- **Un espacio de estados celulares discreto  $\mathcal{L}$**  : también llamado enrejado tal que sea el conjunto finito de celdas (una celda puede también ser llamada célula) del autómata interconectados.
- **Un espacio de valores local  $\Sigma$** : que define todos los posibles valores para cada celda para un determinado autómata celular.
- **Un vecindario  $N$** : que será el conjunto de las  $N$  topológicamente colindantes celdas vecinas que tendrán influencia en la función de transición al siguiente estado de una celda determinada.
- **Condiciones del perímetro**: que puede ser periódica, fijo o reflectante sobre otros. Las periódicas son usadas para simular infinitud en un enrejado finito.

- **Una función de transición  $\phi$**  :  $\overbrace{\Sigma \times \Sigma \times \dots \times \Sigma}^N \rightarrow \Sigma$  que define el cambio del estado de una determinada celda del enrejado al siguiente basándose en su vecindario  $N$ .

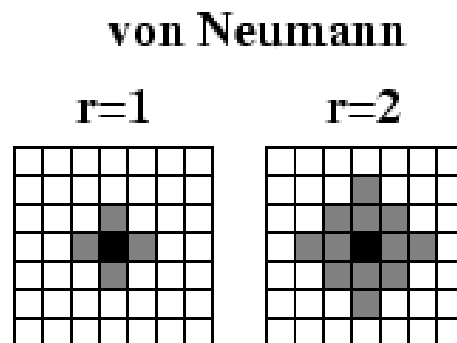
- Toda celda cambia su estado de forma síncrona en sucesivas iteraciones.

## 3.2 Autómatas bidimensionales

---

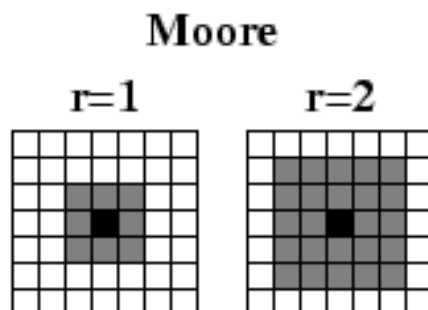
A continuación expondré el tipo de autómatas usado en el proyecto, como se puede discernir del título, se trata de un autómata celular bidimensional, en esta categoría, principalmente se distinguen dos tipos de vecindarios:

**Vecindario von Neumann:** también conocido como vecindario de cuatro es aquel autómata en cuyo enrejado, cada celda define su vecindario como las cuatro celdas en distancia de Manhattan  $r$ , siendo estas descritas como en la siguiente figura:



Como se puede ver en la anterior figura, el vecindario dependerá de  $r$ , por lo que la primera figura sería un vecindario de von Neumann de rango uno y la segunda de rango 2, el número de celdas en un vecindario de von Neumann bidimensional estará basado en el rango  $r$  tal que:  $r^2 + (r + 1)^2$

**Vecindario de Moore:** es un vecindario compuesto por una celda central y las celdas que la rodean, al igual que con von Neumann, existe un rango  $r$  esta vez definido por la distancia de Chebyshev tal y como se ve en la siguiente figura:



Como se puede ver en la anterior figura, el vecindario dependerá de  $r$ , por lo que la primera figura sería un vecindario de Moore de rango uno y la segunda de rango 2, el número de

celdas en un vecindario de Moore bidimensional estará basado en el rango  $r$  tal que:  $(2r + 1)^2$

Para este proyecto dado que al se han de tener en cuenta para cada paso de la simulación todas las celdas colindantes para obtener el nuevo estado de cada celda, se ha usado el vecindario de Moore, aparte se han propuesto condiciones de perímetro cerradas, esto es una celda de una esquina no tendrá un vecindario completo de 8 vecinos (junto consigo misma), sino de tres vecinos.

### 3.3 El juego de la vida

---

Como ya se dijo en la introducción al contexto histórico de los autómatas celulares, durante los setenta, concretamente en octubre de 1970 el matemático británico John Horton Conway publica el concepto teórico de este juego con connotaciones matemáticas, económicas, computacionales e incluso filosóficas.

Este sistema complejo formulado por un autómata celular es particularmente interesante ya que teóricamente es equivalente a una máquina universal de Turing ya que todo se lo que se puede computar algorítmicamente se puede computar en el juego de la vida. Exponiendo una simple y vital representación de comportamientos extremadamente complejos surgiendo de reglas extremadamente sencillas.

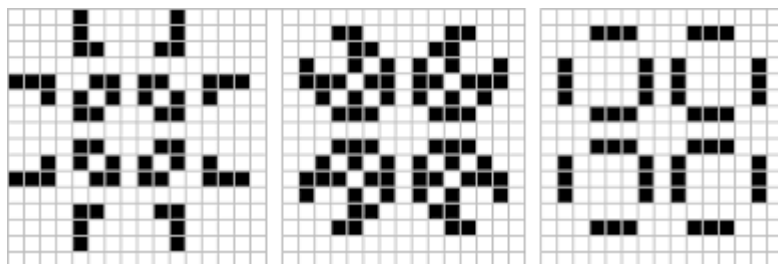
Se trata de una autómata celular con vecindario de Moore  $r = 1$  con unas reglas de evolución harto simples tales que se han de evaluar los siguientes escenarios para cada celda individualmente:

- 1 Soledad:** una celda viva muere si tiene menos de dos vecinos vivos
- 2 Sobrepoblación:** una celda viva con más de tres vecinos vivos muere
- 3 Continuación:** una celda viva con dos o tres vecinos vivos permanece viva
- 4 Renacimiento:** una celda muerta con tres vecinos vivos vuelve a la vida

Con tan solo estas reglas fijas, surge un juego de cero jugadores donde el único aporte por parte del jugador es la configuración inicial del enrejado, aunque se suele optar por un enrejado de generación aleatoria con unas dimensiones definidas.

Con estas reglas se ve la aparición de patrones comunes que serán citados a continuación:

**Osciladores:** son patrones que tras un número fijo de iteraciones vuelven a su estado inicial, estos osciladores tienen rotadores y estatores, siendo los rotadores las celdas que van cambiando su estado y los estatores aquellas que permanecen con un estado fijo. Se han descubierto osciladores de todos los periodos (el periodo de un oscilador es el número fijo de iteraciones hasta volver a su forma inicial) y algunos ejemplos son:

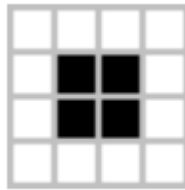


[4]

Estos son de periodo tres y al ser los más comunes se conocen como púlsares

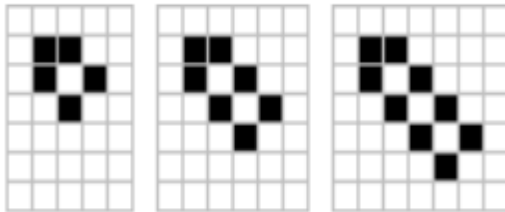


**Inmortales:** son patrones que permanecen inalterados por las iteraciones si no hay incidencia externa, que pueden a su vez ser considerados como osciladores de periodo uno, el ejemplo más simple conocido como bloque sería tal que:

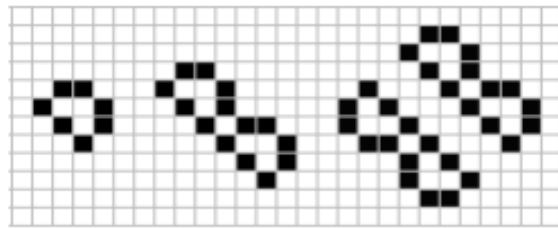


[4]

Existiendo de múltiples formas y tamaños como se puede ver a continuación:

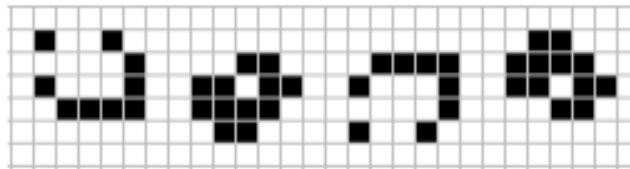


Este patrón es conocido como barco



Este otro como hogaza (o doble hogaza)

**Naves espaciales:** son patrones que aparentemente se desplazan en el espacio con el tiempo, esto es que van variando hasta reaparecer con la misma forma, pero desplazados espacialmente, la velocidad a la que este se desplace está definida por el número de celdas que se desplaza durante un periodo

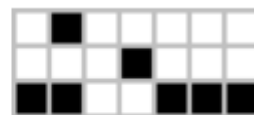


Esta es conocida como nave ligera y como se ve se desplaza de forma horizontal o vertical.

**Matusalenes:** son patrones capaces de evolucionar durante muchas iteraciones hasta terminar estabilizandose, los dos ejemplos expuestos a continuación, Diehard y Acorn tardan 130 y 5206 iteraciones respectivamente:



Diehard



Acorn

---

## Capítulo 4

### Descripción de la solución

---

En este capítulo explicaré los entresijos de la creación de la solución a los retos planteados, en primer lugar los motivos detrás de la elección de las tecnologías utilizadas, esto es el lenguaje de programación y las bibliotecas empleadas, luego revisaremos la estructura del proyecto, la arquitectura detrás de la solución planteada diseccionando sus diferentes partes y por último presentaré el diseño al cual se ha llegado con las tecnologías usadas.

#### 4.1 Lenguaje de programación

---

Para la realización de este proyecto hemos decidido utilizar Python3 como lenguaje de programación para el desarrollo de la solución al problema planteado creando con este el simulador de pandemias mediante autómatas celulares que se buscaba crear.

Para cualquiera que pueda tener un mínimo de conocimientos en la parte práctica de la programación y uso de autómatas celulares, esta elección puede llegar a parecer incoherente ya que la mayoría de proyectos que puedan utilizar esta tecnología en su desarrollo utilizarán Wolfram Mathematica (aunque normalmente se suele hacer referencia al mismo únicamente como Mathematica), un lenguaje de programación desarrollado por un científico ya mencionado de vital importancia para la actual concepción de los autómatas celulares, Stephen Wolfram y su equipo de desarrollo.

La ventaja de usar este software en vez de cualquier otro lenguaje de programación es la existencia de librerías propias (built-in) específicamente creadas y optimizadas para trabajar con computación simbólica, manejar matrices (de vital importancia para el desarrollo de autómatas celulares) y la graficación, aparte de una variedad de librerías desarrolladas únicamente para el trabajo con autómatas celulares que hacen que uno solo tenga que preocuparse por llevar la idea lista, la parte del desarrollo técnico ya está solventada y a su inmediata disposición.

Con esto, el desarrollo de un proyecto como aquel del que versa esta memoria se hubiese visto significativamente simplificado, pero al tratarse de librerías propias de código privado, en caso de querer hacer pequeñas modificaciones para tratar con el problema planteado en cuestión requerirían de dejar de usar estas librerías y tener que desarrollar código desde cero, aparte, las librerías enfocadas a graficación del Mathematica, aunque eficientes ya que desde luego cumplen su función. Pueden no siempre ser personalizables haciendo imposible la muestra de los resultados de la misma forma que en otros lenguajes más grandes con librerías de uso general podría ser posible. Por ello decidimos usar un lenguaje de propósito general para el desarrollo del proyecto, eligiendo aquel en el cual yo más cómodo me hallo, Python.

De entrada, con esta elección de lenguaje de programación se plantean ciertos retos, tras haber buscado librerías para la creación y uso de autómatas celulares me di cuenta de que las únicas librerías existentes están muy poco desarrolladas, no son personalizables y apenas si sirven para demostraciones puntuales de cosas como el juego de la vida o pruebas de concepto para proyectos con propósito específico. Por ello y aunque hubiese facilitado enormemente el desarrollo técnico del proyecto, hube de desarrollarlo desde cero tan solo contando con la ayuda de librerías para la parte matemática y de interfaces tal y como discutiré a continuación.

Se han utilizado las siguientes librerías:

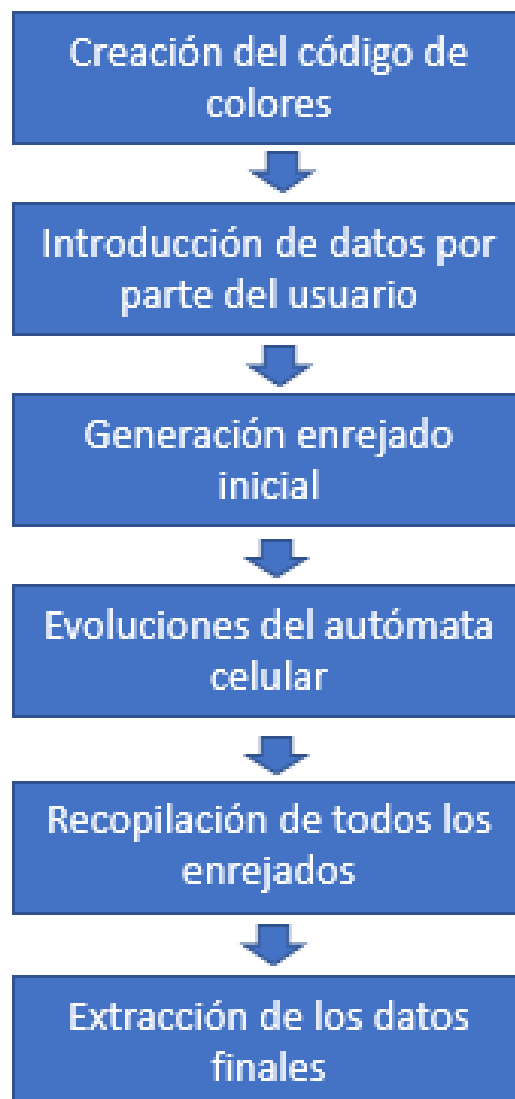
- **Matplotlib** una librería para crear estáticos, animados e interactivas visualizaciones en Python, en el proyecto se ha usado para crear la representación gráfica del enrejado correspondiente al autómata celular y para la graficación de las estadísticas correspondientes a los números indicativos de la evolución de la pandemia.
- **Numpy** para múltiples operaciones matemáticas comunes optimizadas para su posible uso en Python trabajando en segundo plano en C y Fortran.
- **Pathlib** para poder guardar las sucesivas representaciones del enrejado del autómata celular.
- **Openpyxl y tkinter** para la creación y funcionamiento de la pequeña interfaz de entrada para introducir los datos de la pandemia, su evolución y los datos fundamentales del autómata celular

## 4.2 Arquitectura del proyecto

---

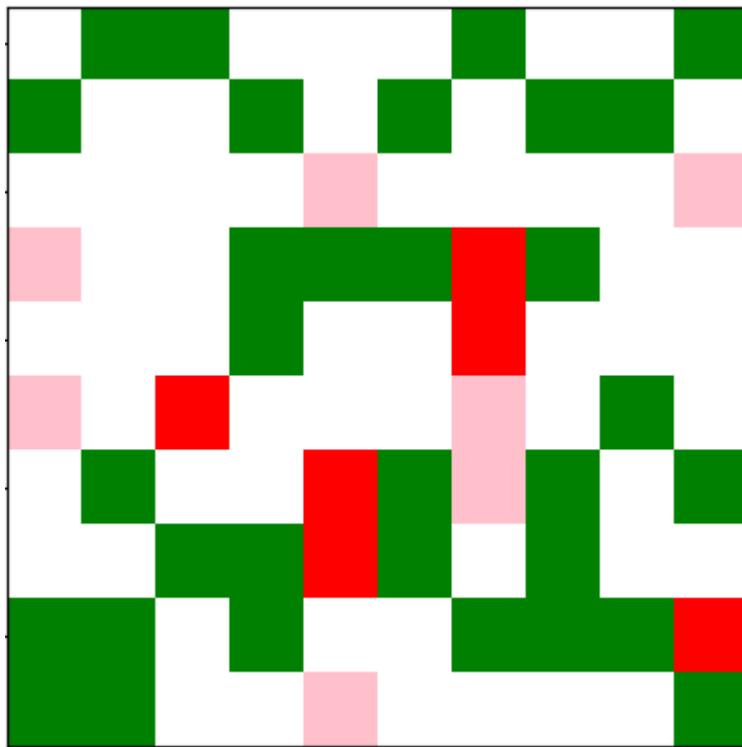
Como ya se mencionó en el anterior apartado, dado la inexistencia de librerías para el desarrollo de proyectos de autómatas celulares en Python, tuve que desarrollar todos los pasos necesarios sin poder apoyarme en biblioteca externa alguna. Aparte, no existe una manera fácil de representar una matriz de la manera concebida para el proyecto por lo que se tuvo que llegar a una solución creativa para que la coloración de la matriz fuese la deseada.

La arquitectura seguida en la solución sigue el siguiente esquema secuencial:



## 4.2.1 Creación del código de colores

En primer lugar y antes de comenzar con el código del resto del proyecto, se ha de codificar una pequeña argucia para solventar uno de los problemas de la representación deseada para el enrejado representativo del proyecto. Cómo se ha descrito ya en anteriores apartados, lo que se quiere conseguir con este autómata celular es un enrejado en el que cada celda representa un individuo (o la ausencia del mismo) y su estado, siendo el estado de cada individuo o celdilla representado por un código de colores, esto es, que cada estado del autómata celular se vea representado por una matriz donde cada celda está pintada de un color representativo del individuo al que representa como en la captura siguiente:



Para mi sorpresa, no existe una manera sencilla de establecer una representación similar para una matriz en python, siendo lo único aproximable una ListedColormap de la biblioteca matplotlib.

El reto aquí consiste en que cada celda en código interno ve representado su estado por un número entero y luego se ha de conseguir una especie de permutación o equivalencia fija número-color para conseguir una representación constante que nosotros hayamos fijado.

Para esto se ha en primer lugar que definir cuáles serán los estados posibles que toda celdilla podrá alcanzar, para este proyecto se han utilizado los siguientes posibles estados:

- **Celda vacía:** codificada con un cero y representada por una celda color blanco, esta celda podrá ser ocupada por individuos que se muevan y la cantidad inicial de celdas vacías es introducida por el usuario
- **Celda contagiada:** se asume que un individuo que enferma y no es asintomático pasará la enfermedad en catorce días si no se dan circunstancias agravantes que le fueren a una estancia indefinida en el hospital, al estar 14 días contagiado, la codificación numérica para un individuo contagiado irá desde el uno al 14 ambos inclusive siendo su codificación de color rojo. Un individuo contagiado puede moverse y tendrá una probabilidad por cada evolución del autómata de pasar al hospital y una mucho menor de fallecer en casa, todas estas posibilidades pueden ser cambiadas por el usuario.
- **Celda asintomática:** un individuo será asintomático si contrae la enfermedad pero no presenta síntomas que la hagan evidente, este estado durará los catorce días que tarde el individuo en pasar la enfermedad. Al durar este estado catorce días, su codificación numérica comprenderá entre 15 y 28 ambos inclusive siendo su codificación de color rosa claro. Un individuo asintomático al no presentar síntomas no tendrá una probabilidad por evolución del autómata de empeorar su condición, pero sí es capaz de contagiar la enfermedad y por supuesto tendrá plena capacidad de movimiento.
- **Celda inmune:** un individuo será inmune temporalmente si ha pasado la enfermedad y la ha superado. Su codificación numérica comprenderá entre 29 y 42 siendo su codificación de color azul. En este estado, esta celda ni contagia ni puede ser contagiada.
- **Celda hospital:** un individuo contagiado tiene una probabilidad por evolución del autómata celular de pasar a estar hospitalizado. La codificación numérica de un individuo hospitalizado es el 43, siendo el color correspondiente el amarillo. Una vez en el hospital, no existe una duración estimada fija sino que existe una probabilidad introducida por el usuario para que el individuo salga del hospital y otra para que el sujeto perezca en el mismo. Un individuo en el hospital no posee capacidad de movimiento.
- **Celda fallecida:** un individuo que fallece tarda un turno en fallecer, durante el cual su celda pasa a una codificación numérica de 44 y un color negro.
- **Celda sana:** esta celda con codificación numérica 45 y color verde representa a un individuo sano, este individuo se contagiará si en su vecindario (recordemos que se usa un vecindario de Moore de rango  $r = 1$ ) se encuentran tres contagiados. Los individuos sanos que se contagien tienen una probabilidad definida por el usuario de ser asintomáticos. Los individuos sanos tienen plenas capacidades de movimiento, siendo la probabilidad por evolución del autómata de desplazarse elegida también por el usuario, aunque como veremos en las ejecuciones y gráficas de la memoria, para obtener una simulación mínimamente realista, la probabilidad de movimiento ha de ser sustancial. Por el caso de estudio se ha decidido que no existan otras causas de muerte aparte de la enfermedad en representación.

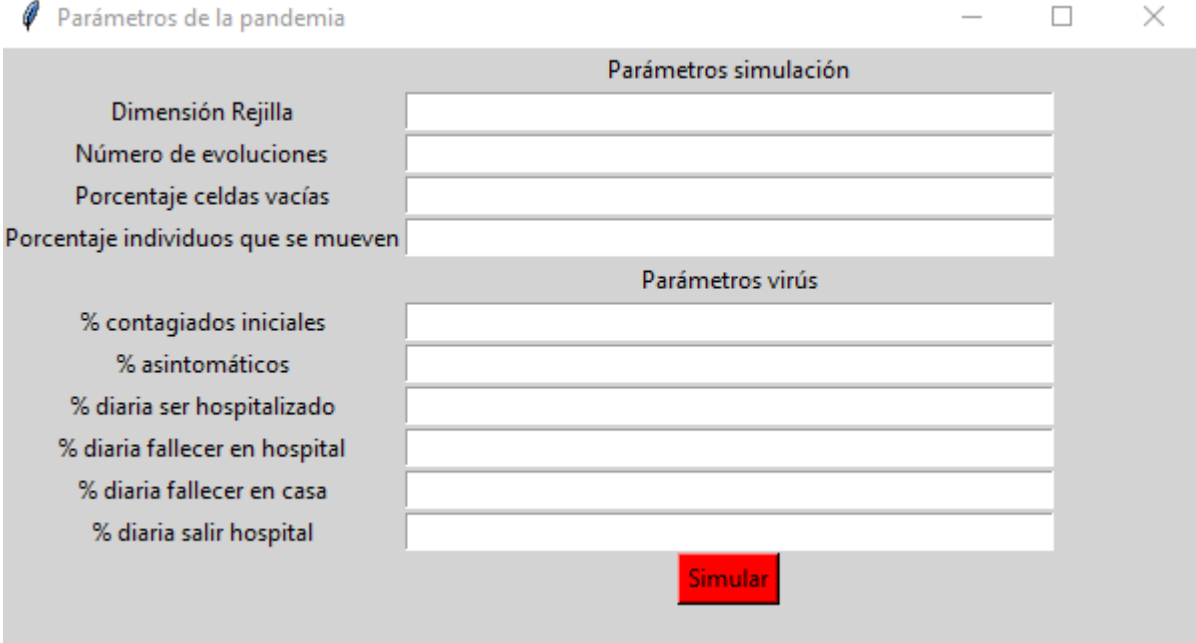
Con esta explicación de los colores y codificaciones numéricas de los distintos estados, para que el ListedColormap funcione como ha de funcionar, se habrá de crear un array de colores con 45 posiciones para las 45 distintas codificaciones numéricas existentes, permitiendo así crear una permutación entre la codificación numérica y este array de colores. La creación del array irá tal que así:

```
# La creación del array correspondiente a el código de colores
colores = ['white']
aux = ['red', 'pink', 'blue']
for i in range(3):
    for j in range(14):
        colores.append(aux[i])
colores.extend(['yellow', 'black', 'green'])

cmap = matplotlib.colors.ListedColormap(colores, name='colores', N=46)
```

## 4.2.2 Introducción de datos por parte del usuario

Lo siguiente a analizar es la introducción de datos por parte del usuario mediante la siguiente simple interfaz hecha con openpyxl y tkinter:



The image shows a Tkinter window titled "Parámetros de la pandemia" with standard window controls (minimize, maximize, close). The window content is organized into a grid. At the top, there is a section header "Parámetros simulación" followed by four text input fields labeled "Dimensión Rejilla", "Número de evoluciones", "Porcentaje celdas vacías", and "Porcentaje individuos que se mueven". Below this is another section header "Parámetros virus" followed by six text input fields labeled "% contagiados iniciales", "% asintomáticos", "% diaria ser hospitalizado", "% diaria fallecer en hospital", "% diaria fallecer en casa", and "% diaria salir hospital". At the bottom center of the window is a red button labeled "Simular".

Aquí el usuario ha de decidir en primer lugar los parámetros de la pandemia. Primero los parámetros de la simulación, tanto los del autómata celular como los necesarios para la creación y evolución de la celdilla.

Luego se han de especificar los tantos por ciento de los distintos parámetros correspondientes al virus. Aquí es donde el usuario podría llegar a simular la evolución de un virus distinto a aquel que inspiró el proyecto pudiendo la herramienta ser útil para predecir la incidencia de nuevas epidemias en poblaciones aisladas.

Esta interfaz se ha hecho mediante un sistema en grid donde cada texto, botón o textbox viene emplazado en la interfaz mediante una posición preasignada.

Dado que el código aún para una interfaz tan simple es bastante amplio y poco significativo para el objetivo del proyecto, he decidido no introducir ningún recorte del código en esta sección de la memoria sino un extracto del resultado de dicho código.



### 4.2.3 Generación del enrejado inicial

Una vez tenemos los datos de la pandemia proporcionados por el usuario, tendremos que seguir el primer paso en todo proyecto que involucre autómatas celulares, la generación inicial del enrejado.

En primer lugar se ha de generar la celdilla con las dimensiones proporcionadas por el usuario e inicialmente he optado por rellenar a ceros ya que como se explica en el primer punto de la arquitectura, los ceros corresponden a las celdas vacías, ahorrando así tener que estar luego rellenando estas celdas.

Una vez hecho esto he creado un método auxiliar que rellena en una matriz dada el número pedido de celdas con un número proporcionado.

```
CELDA_VACIA = 0
CONTAGIADO_1 = 1
CONTAGIADO_N = 14
ASINTOMATICO_1 = 15
ASINTOMATICO_N = 28
AISLADO_1 = 29
AISLADO_N = 42
HOSPITAL = 43
MUERTO = 44
SANO = 45
```

(Se han asignado estas variables a estos enteros para facilitar la codificación numérica)

```
def generacion_rejilla_inicial(dimension_inicial_rejilla, porcentaje_celdas_vacias, porcentaje_sanos, porcentaje_contagiados, porcentaje_asintomaticos):
    numero_inviduos = int((dimension_inicial_rejilla*dimension_inicial_rejilla)*(1-porcentaje_celdas_vacias))
    numero_contagiados = int(numero_inviduos*porcentaje_contagiados)
    numero_sanos = int(numero_inviduos- numero_contagiados)
    numero_asintomaticos = int(numero_contagiados*porcentaje_asintomaticos)
    numero_sintomaticos = int(numero_contagiados - numero_asintomaticos)
    # numero_celdas = int(dimension_inicial_rejilla*dimension_inicial_rejilla)

    # Iniciamos la matriz a celdas vacías es decir con un 45
    matrix = np.zeros(shape = (dimension_inicial_rejilla,dimension_inicial_rejilla), dtype = np.int8)

    # Rellenamos insertando los individuos existentes

    # Primero los sanos
    matrix = rellenar_celdilla(matrix,numero_sanos,dimension_inicial_rejilla,SANO)
    # Luego los asintomáticos
    matrix = rellenar_celdilla(matrix,numero_asintomaticos,dimension_inicial_rejilla,ASINTOMATICO_1)
    # Luego los sintomáticos
    matrix = rellenar_celdilla(matrix,numero_sintomaticos,dimension_inicial_rejilla,CONTAGIADO_1)
    #print(matrix)
    return matrix
```

Con esto, tras haber llamado tres veces al método auxiliar para colocar en posiciones aleatorias los contagiados, asintomáticos e individuos sanos poblando la matriz concluye el método de creación inicial del enrejado.

## 4.2.4 Evoluciones del autómata celular

Dado que no existe biblioteca para trabajar con autómatas celulares donde definir unas claras reglas de evolución que se apliquen a una rejilla inicial, se ha tenido que implementar un método propio donde se apliquen al enrejado creado anteriormente las reglas que se han decidido seguir y también aplicar una vez calculado los estados de la evolución posterior, las reglas de movimiento para tener la matriz evolucionada en un paso.

Para esto decidí crear un método que evoluciona la matriz en un único paso al cual llamaré tantas veces como evoluciones marque el usuario pertinentes. Este método recibe la matriz y los parámetros requeridos para calcular el futuro de cada celdilla.

En este método recorro la matriz con un sistema de bucles que nos permite llegar a cada celda, luego seguiremos las reglas de evolución para cada individuo:

- Un individuo **contagiado** puede seguir contagiado hasta que pasen los catorce días de convalecencia o en función de las probabilidades introducidas por el usuario puede fallecer en casa o requerir hospitalización. Si no se produce complicación alguna, tras catorce días el individuo pasa a estar sano
- Un individuo **asintomático** pasa los catorce días asintomático y luego vuelve a estar sano, pudiendo contagiar durante este periodo.
- Un individuo **hospitalizado** puede permanecer en el hospital, fallecer o salir del mismo una vez esté sano.
- Un individuo **fallecido** pasa a ser una celda vacía en la siguiente evolución
- Un individuo **sano** permanecerá sano siempre que no tenga contagiados en su entorno, en caso contrario se aplicarán las reglas de contagio en caso de contagiarse pasará la enfermedad con o sin síntomas
- Un individuo **inmune** seguirá siendo inmune de forma temporal, tras lo cual volverá a ser sano.

Una vez se calculan los estados de la nueva rejilla, aquellos indicados se moverán acorde a las reglas de movilidad. Tanto las reglas de contagio como las reglas de movilidad serán sujeto de estudio en el quinto capítulo de la memoria.

## **4.2.5 Recopilación de todos los enrejados**

Una vez se concluye en la creación del enrejado inicial o se termina la evolución en un paso del autómata celular, dicho enrejado es graficado como una matriz con un ListedColormap para que incluya los colores deseados y se guarda en local para no parar la ejecución del programa y sus sucesivas evoluciones del autómata celular así como para poder observar una vez estas concluyan la evolución de la pandemia con los parámetros introducidos sobre la población objetivo.

## **4.2.6 Extracción de datos finales**

Durante cada evolución del autómata celular, en unos arrays de alcance global se irán acumulando los datos de la pandemia, esto es, número de individuos sanos (tanto sanos, verdes, como inmunizados, azules), número de individuos inmunes en dicha evolución del autómata celular, número de contagiados por cada evolución, número de hospitalizados en cada paso y números de muertos totales durante la pandemia. Una vez se concluyan las evoluciones, se reunirán todas estas listas de datos y se guardarán junto con el resto de imágenes en una gráfica que represente estos datos recogidos en forma de gráfica de líneas en función de las evoluciones del autómata y número de individuos pertenecientes a cada grupo por en cada evolución.

## 4.3 Diseño final

---

Al final, una vez explicada la arquitectura y todas las partes del proyecto, solo queda decir que se ha conseguido solventar la ausencia absoluta de librerías para el trabajo con autómatas celulares en python, consiguiendo crear un enrejado acorde a unas determinadas reglas y evolucionando para simular una epidemia en una población objetivo.

Se ha optado por este diseño secuencial en vez de tratar de incorporar algún tipo de paralelismo durante, ya sea el relleno inicial del enrejado o la evolución del mismo en cada paso porque en todo momento se está trabajando sobre una única matriz, la cual habría de estar compartida entre los distintos procesos, lo cual imposibilita un relleno completamente aleatorio en paralelo o el cálculo de los individuos en movimiento ya que varios procesos podrían interbloquearse al intentar llegar a una misma nueva posición.

Una posible mejora sobre el diseño incorporando paralelismo hubiese sido dividir la matriz entre varios procesos, para que estos calculen el siguiente estado de cada celdilla de manera más eficiente que la implementada en secuencial, esperándose mutuamente para terminar todos antes de comenzar con el cálculo del movimiento en secuencial.

Con esto solo se conseguiría una mejora temporal, la cual aún llegando a poder ser significativa en rejillas de gran tamaño, no mejora de manera alguna el objetivo de poder simular la evolución de la pandemia de manera más exacta o precisa. Por lo que ante la complejidad técnica de hacer esto en paralelo en vez de en secuencial, al no aportar una mejor solución al problema planteado, se ha descartado esta implementación, haciendo la versión secuencial, que recordemos, generará los mismos resultados con algo más de tiempo.

Por todo esto, se ha conseguido un programa capaz de a partir de unos datos o parámetros propios de una pandemia proporcionados por un usuario, generar una posible evolución de dicha pandemia en una población objetivo y obtener a partir de esta unos datos sugeridos de la posible incidencia del virus en el futuro en caso de tratarse de una población aislada.



---

## Capítulo 5

### Presentación de la herramienta

---

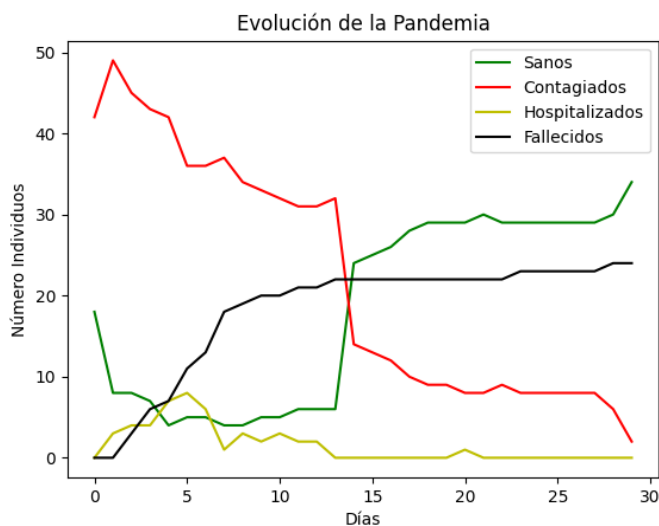
En este capítulo se expondrán las distintas reglas de infección y movilidad que se han explorado durante el proyecto, enseñando las gráficas en las cuales baso mi razonamiento para ir las mejorando llegando al escenario final en el cual se llega a un conjunto de reglas relativamente realista capaz de simular, siendo cada nuevo paso computacional representativo de un día, el comportamiento de la expansión de un virus en una población aislada.

#### 5.1 Reglas de infección

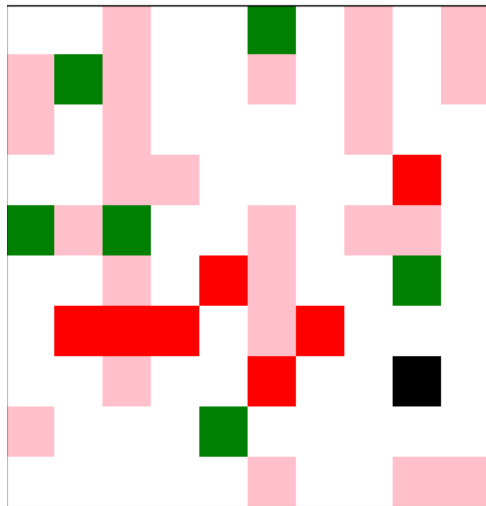
---

La primera aproximación a resolver este problema consiste en una manera simple y sencilla de simular las posibilidades de infección de un individuo. Como sabemos, ante una mayor exposición a cualquier tipo de virus, nos enfrentaremos a una mayor probabilidad de infección, por lo que decidimos, para poder hacer pruebas iniciales y comprobar que todo funcionase, que un individuo sano, se contagie al tener en su vecindario inmediato de Moore rango  $r = 1$  a una cantidad igual o mayor a tres contagiados. En cualquier otro caso el individuo permanecerá sano. Con esta aproximación y debido a las dificultades de expansión de la pandemia, no he implementado el posible estado de inmune, por lo que un individuo que pase la enfermedad, independientemente de si la pasa con síntomas o sin ellos, pasará una vez superada a individuo sano con plenas capacidades de contagio.

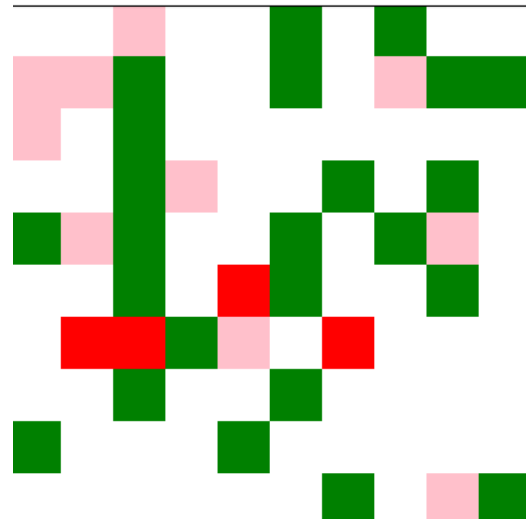
Con este set de reglas obtenemos las siguientes gráficas tras haber jugado con los parámetros para obtener un resultado que se asemeje al máximo a una evolución natural de una pandemia durante treinta días:



Como se puede observar en la anterior gráfica, esta aproximación presenta ciertos problemas evidentes. Tras el comienzo y una inicial subida muy pequeña, los casos de contagiados no hacen sino descender, lo que evidencia que no están surgiendo nuevos contagiados en la simulación, esto se evidencia a su vez en la representación de la matriz tras catorce días:



Día 14



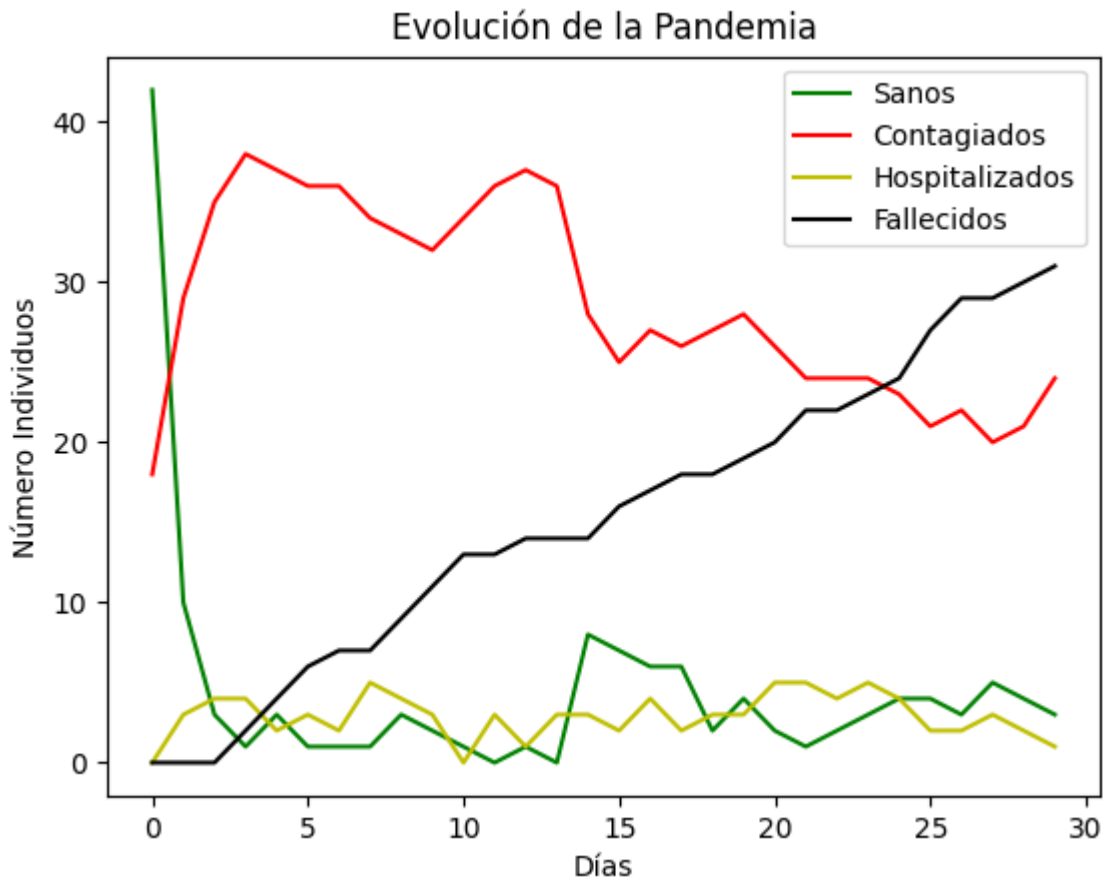
Día 15

Como se puede ver en las anteriores imágenes, aunque el comportamiento sea el esperado y con estos parámetros, se pueda ver ciertas similitudes con las gráficas que se pudieran crear recogiendo datos de una epidemia real, dado la escasísima tasa de infección existente, tras que los casos iniciales desaparecen, apenas si quedan unos días antes de que la población haya superado la pandemia.

Aparte, por este mismo motivo, para que se logre una gráfica como la anterior, se han de introducir de forma forzada un elevadísimo porcentaje de población inicialmente contagiada, en caso contrario la pandemia nunca se desarrollará en número de infectados, sin conseguir por ello similitud alguna con el COVID-19, la gripe estacional o cualquier otra enfermedad con un ratio de contagio mínimamente elevado.

Por ello, en vez de que los individuos sanos solo pudiendo contagiarse teniendo a tres o más contagiados en su vecindario más cercano, he realizado una segunda aproximación en la cual por cada infectado en su entorno, exista una probabilidad de enfermar.

Si se sigue este procedimiento con un 33.3% (para que se mantenga ese 100% de tasa de infección en individuos rodeados por al menos tres infectados) se obtiene la siguiente gráfica:

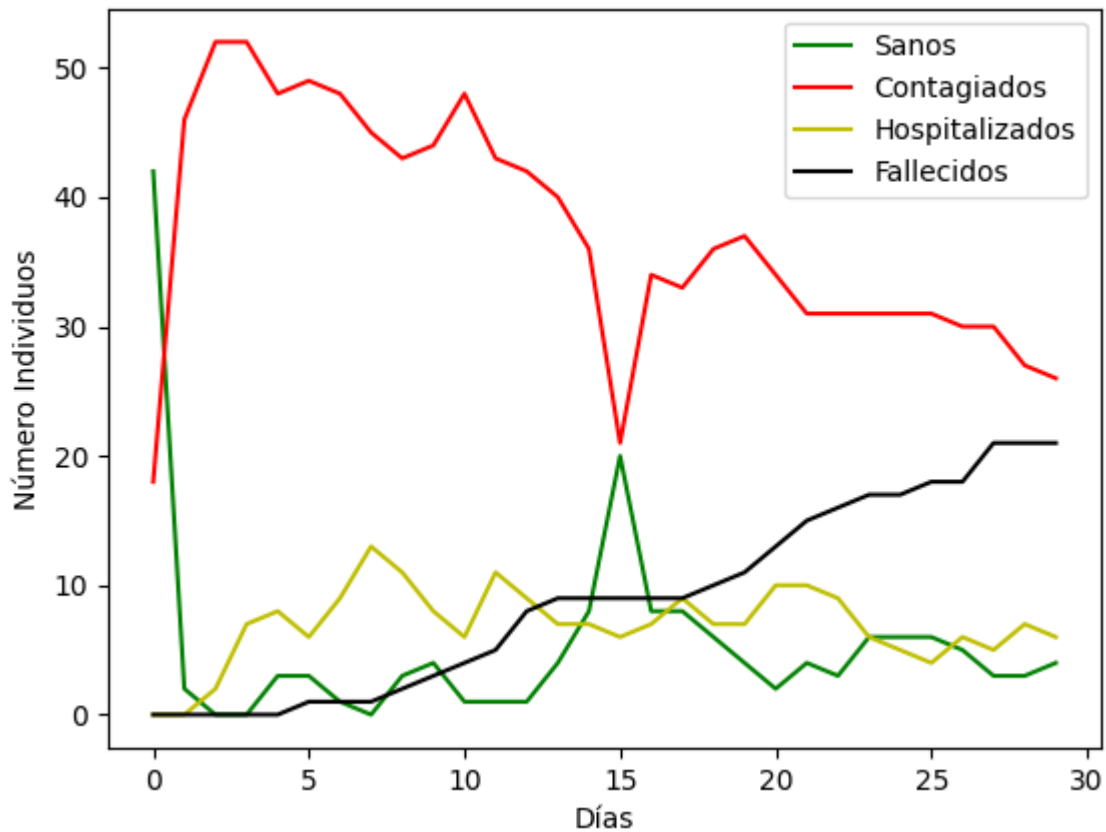


Como se puede ver, estas nuevas reglas de contagio hacen que el virus no pare de expandirse alcanzando a toda la población tras apenas cinco días, aún habiendo comenzado con solo un 30 % de la población infectada y sin disminuir ya que aunque tras quince días, se puede ver un pequeño número de individuos sanos, la población al no poder inmunizarse vuelve a ser inmediatamente víctima de la enfermedad diezmando en apenas treinta días a la mayor parte de la población.

Si se trabaja con un ratio de contagio mucho menor, de un 10% se obtiene la siguiente gráfica



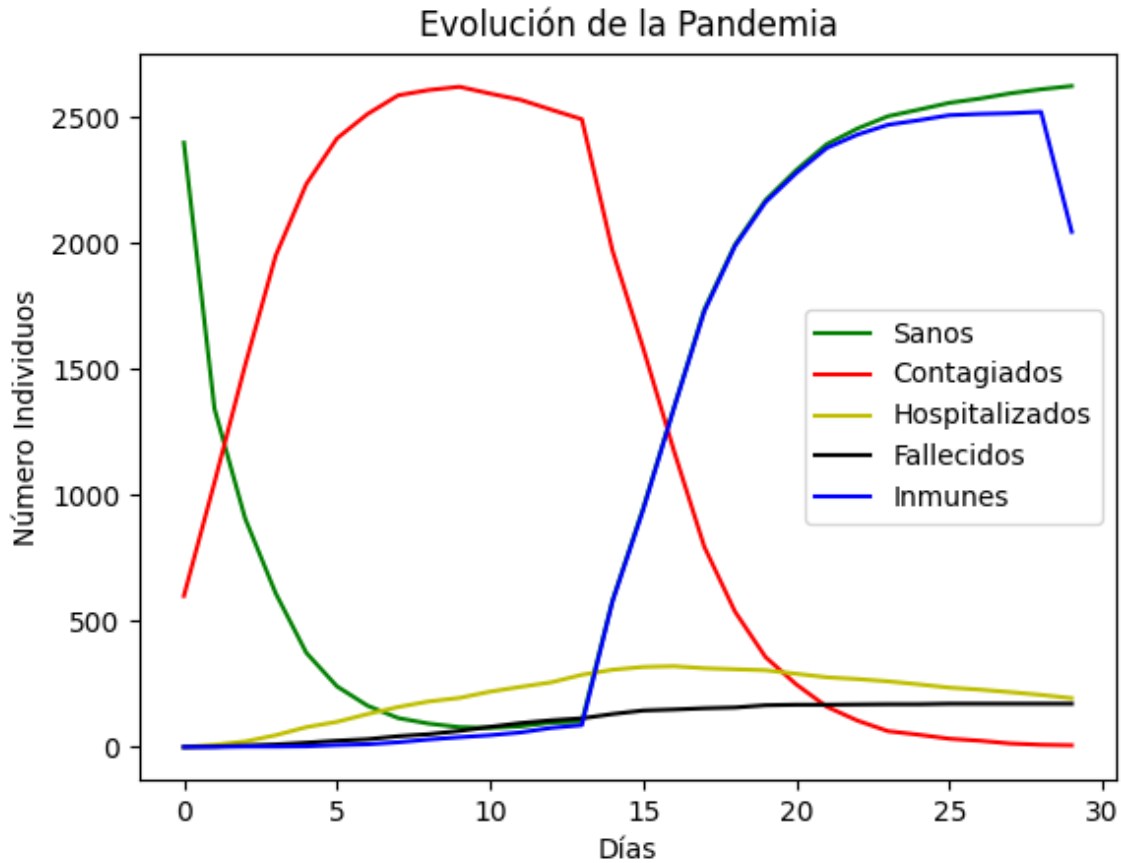
### Evolución de la Pandemia



Como se puede ver al tratarse de un ratio de contagio algo menor, la población no se ve tan mermada, aún así, al no poder desarrollar inmunidad frente al virus, tarde o temprano se vuelve al mismo escenario que con una mayor tasa de contagio.

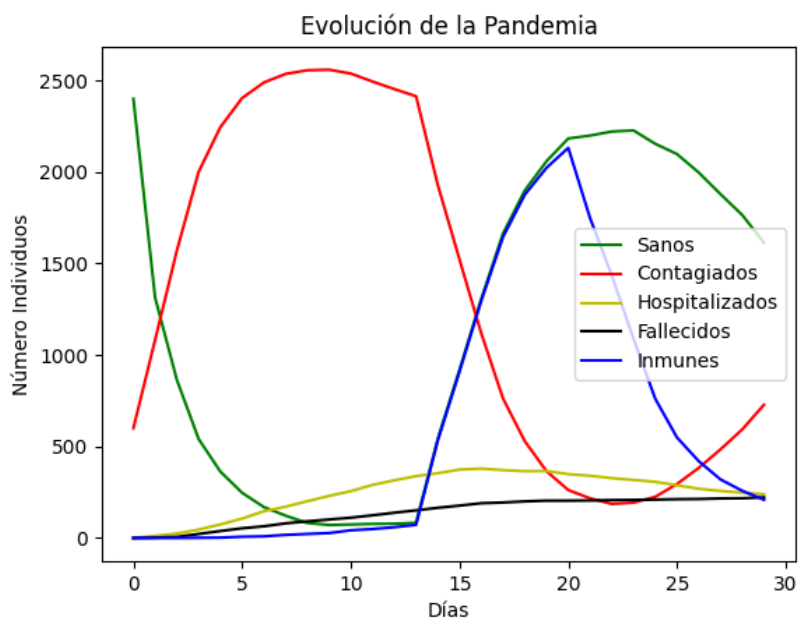
Por ello se ha de implementar un estado de inmunidad, el cual durará lo mismo que la enfermedad, 14 días durante los cuales no se podrá contraer ni propagar la enfermedad. Un individuo pasará a ser inmune en el momento el cual supere la enfermedad, ya sea tras una estancia hospitalaria o tras haber pasado los catorce días del momento del contagio inicial independientemente de si se sufrieron síntomas o no.

Esta tercera aproximación produce la siguiente gráfica con un ratio de contagio del 33%:



Como se puede ver, el periodo de inmunidad hace que una vez toda la población se inmunice, el virus desaparece, por eso creo que de cara a una mejor representación en las gráficas, se ha de reducir significativamente el tiempo inmunizado.

Una vez hecha esta modificación y se puede ver el siguiente cambio:



Aquí ya se puede ver una mejor representación de los datos que se podrían dar en una población aislada que se enfrenta a un virus. Como se puede ver, existe un comportamiento en olas para el número de infectados o contagiados y la presión hospitalaria llega a un punto álgido durante y justo tras los picos de dichas olas. Lo único que se le puede echar en cara es el drástico cambio en el número de individuos inmunes, ya que al crearse el enrejado inicial, todo aquel que queda contagiado, se contagia a la vez, haciendo que tras catorce días de convalecencia, se inmunizan todos de golpe creando el pico de inmunes entre los días quince y veinte. Tras lo cual este número baja radicalmente para luego volver a subir siguiendo la periodicidad existente en la evolución de cualquier pandemia.

A pesar de poder corregir este impacto inicial exagerado simplemente decretando que en vez de que al comenzar se rellene la celdilla, no con contagiados en el primer día de convalecencia sino en un número aleatorio de esos 14 días. He decidido no implementar esta modificación ya que para el caso de estudio resulta interesante simular que la pandemia comienza con un brote repentino que afecta a la población en un mismo instante temporal.

En el siguiente capítulo jugaremos con los parámetros de entrada para ver, en primer lugar, con qué rango de los valores de los parámetros se obtiene una simulación más realista y también veremos cómo se comporta la simulación ante abruptos cambios y qué incidencia tendrán sobre la pandemia.

## 5.2 Reglas de movilidad

---

Las reglas de movilidad son mucho más simples que las reglas de contagio, aquí existe un parámetro ajustable por el usuario el cual nos indica el porcentaje de individuos que se moverán a cada evolución del autómata celular.

En cada nueva evolución, en el método responsable de la misma, se elegirán un número equivalente de la población total proporcional al porcentaje por el usuario seleccionado, guardando las posiciones de los mismos en un array auxiliar para su posterior uso.

Para que un individuo sea elegible para moverse, este ha de estar en uno de los siguientes estados: sano, inmune, contagiado sintomático o contagiado asintomático.

Una vez se ha calculado el nuevo estado de cada celdilla en base al enrejado anterior, habiéndolo a su vez seleccionado cuáles son las posiciones de individuos que se deben mover, se procede a intentarlos mover uno a uno siguiendo el orden del de almacenamiento en el array.

Se han de buscar las posiciones libres en el vecindario de Moore inmediato (rango  $r = 1$ ) y se seleccionará una de ellas de forma aleatoria. En caso de que dicho individuo tenga todas sus celdillas colindantes ocupadas, dicho individuo no podrá moverse y permanecerá en su celdilla.



---

## Capítulo 6

### Ejemplos de escenarios: ejecuciones y gráficas

---

En este capítulo se mostrarán distintas ejecuciones jugando con los parámetros para averiguar con qué parámetros se consigue una simulación más realista y a continuación veremos el efecto que tendrá sobre la evolución del autómeta celular variar determinados parámetros, viendo si se comporta de una forma realista siguiendo el comportamiento que se esperaría en la evolución de la pandemia.

Para el proceso de experimentación, mantendremos las simulaciones a priori en cien días, para poder observar los cambios en la población, una vez hallados unos parámetros realistas, procederemos con simulaciones de mayor duración.

Antes que nada voy a enumerar los parámetros con los que vamos a experimentar:

En primer lugar, los parámetros propios de la simulación:

- **Dimensión rejilla:** la longitud en número de celdillas de cada lado del autómeta celular, en caso de por ejemplo introducir 50, el enrejado del autómeta celular será de un 50x50 de dimensiones haciendo así 2500 celdillas.
- **Porcentaje de celdas vacías:** con esto se decide la densidad de población del enrejado, en caso de querer simular una población urbana, se ha de introducir un valor mucho menor que en caso de querer simular un área rural con menor densidad poblacional. Aquí como en cada porcentaje, se está trabajando con tantos por uno, pudiéndose introducir un valor entre cero y uno.
- **Número de evoluciones:** el número total de evoluciones que se van a simular para la siguiente ejecución contando con la primera. A mayor número de evoluciones con un enrejado de mayor tamaño, mayor la tardanza de la simulación.
- **Porcentaje de individuos que se mueven:** con esto se decidirá cuál será el porcentaje de individuos con capacidad de movimiento que se moverán en cada evolución de la simulación. Con este valor se podrá ver de relevancia la importancia de medidas como el confinamiento domiciliario o las restricciones de movilidad que se han implementado como medidas estrella durante la pandemia del COVID-19 Europa y concretamente en España.

A continuación explicaré los parámetros propios personalizables por parte del usuario del virus:

- **% contagiados iniciales:** este será el porcentaje de la población que en la generación inicial de la rejilla se vea contagiada por la enfermedad durante el brote inicial de la misma.
- **% asintomáticos:** con este tanto por uno se decidirá cuál es el porcentaje dentro de los contagiados que pasarán la enfermedad de forma asintomática, evitando así posibles complicaciones que acaben en el hospital o el fallecimiento domiciliario
- **% diaria ser hospitalizado:** probabilidad en cada evolución de la simulación de en caso de haber sido contagiado y presentar síntomas, se requiera de atención hospitalaria y el individuo se tenga que hospitalizar.
- **% diaria fallecer en hospital:** probabilidad en cada evolución del autómata celular de que un individuo hospitalizado fallezca durante su estancia en un hospital.
- **% diario salir hospital:** probabilidad diaria de que un individuo hospitalizado, termine su convalecencia y vuelva al conjunto de individuos no hospitalizados entrando al conjunto de inmunizados, como ya se ha mencionado en anteriores ocasiones, la estancia en el hospital es a priori indefinida y su duración se verá regulada por este y el anterior parámetro en función del devenir del individuo.
- **% diaria fallecer en casa:** probabilidad en cada evolución del autómata celular de que se produzca un fallecimiento domiciliario fruto de complicaciones de la enfermedad, este parámetro y el de la probabilidad diaria de fallecimiento en el hospital (junto con la probabilidad de llegar al mismo) regularán la mortalidad de la enfermedad.
- **Ratio de contagio:** probabilidad por cada vecino adyacente en el vecindario de Moore de rango  $r = 1$  de contagiarse de la enfermedad, este parámetro regulará la capacidad y rapidez de propagación de la enfermedad.

Una vez listados los parámetros que el usuario podrá modificar para simular la evolución de la pandemia, procederemos a ajustarlos y ver cómo las modificaciones posibles, afectan al desarrollo de la enfermedad en el enrejado.

También, aunque ya se menciona en el apartado 4.2.1, comentaré el código de colores:

- **Celda vacía** = blanco
- **Individuo sano** = verde
- **Individuo inmune** = azul
- **Individuo contagiado sintomático** = rojo
- **Individuo contagiado asintomático** = rosa
- **Individuo hospitalizado** = amarillo
- **Individuo recientemente fallecido** = negro

A pesar de existir una interfaz con la cual el usuario pueda cambiar los parámetros, aquí los valores de los mismos serán extraídos mediante capturas directamente del código ya que

es más rápido y se mantendrán entre ejecuciones, facilitando enormemente las comparativas cuando solo se modifique un parámetro y el resto permanezcan constantes.

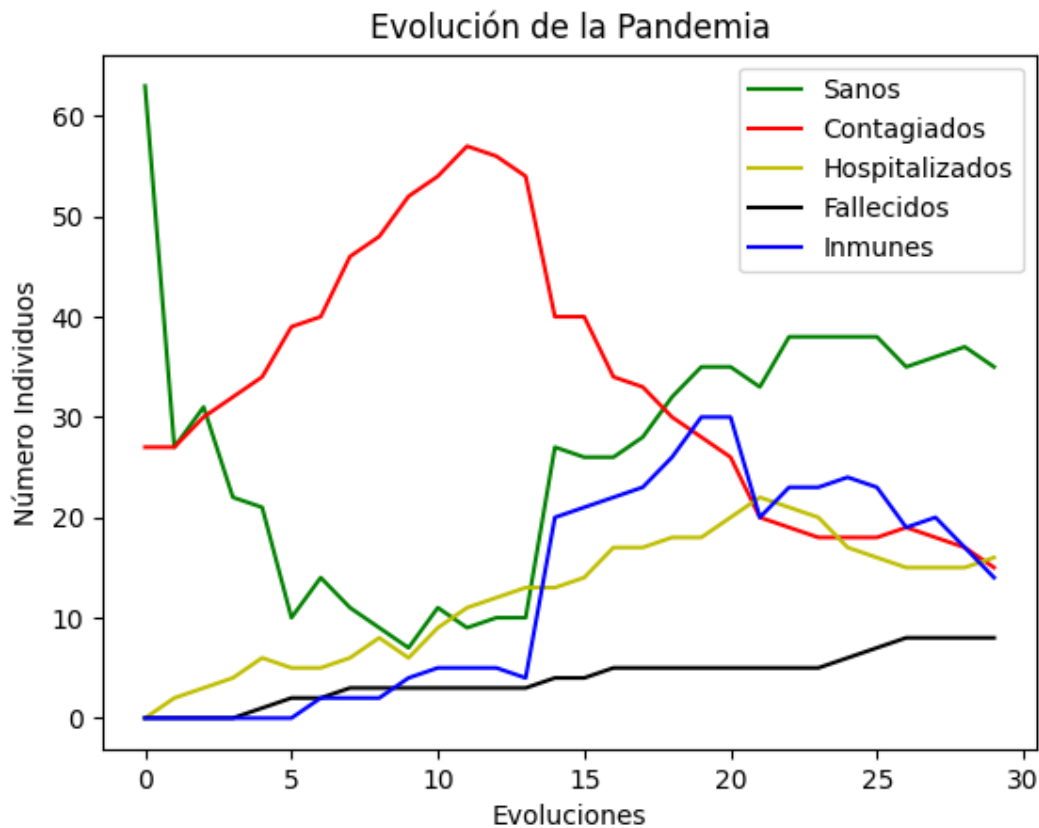
Comenzaremos con una exposición de una ejecución en una rejilla de tamaño pequeño para que se vea cada paso y poder explicar qué ocurre en cada etapa de la simulación con los siguientes parámetros:

```
# Parámetros:

# Necesarios para la creación de la rejilla inicial:
dimension_inicial_rejilla = 15
# Número de evoluciones
numero_de_evoluciones = 30
# Esto se podría extrapolar a densidad poblacional:
porcentaje_celdas_vacias = 0.6
# Porcentaje de individuos que se mueven:
porcentaje_individuos_mueven = 0.5
# Porcentaje de individuos sanos o enfermitos:
porcentaje_sanos = 0.7
porcentaje_contagiados = 0.3
porcentaje_asintomaticos = 0.05
#-----
# Datos del virus-----
porcentaje_de_hospitalizaciones = 0.02
porcentaje_fallecidos_en_hospital = 0.015
porcentaje_fallecidos_domesticos = 0.005
probabilidad_diaria_salir_del_hospital = 0.05
ratio_de_contagio = 0.1
#-----
```

Se produce la siguiente gráfica en las primeras 30 evoluciones del autómatas celular:

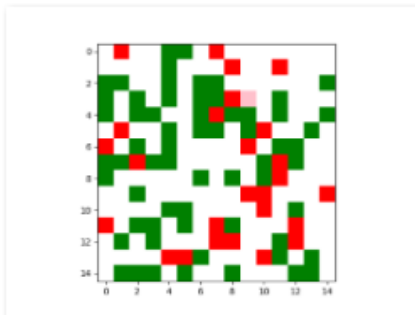




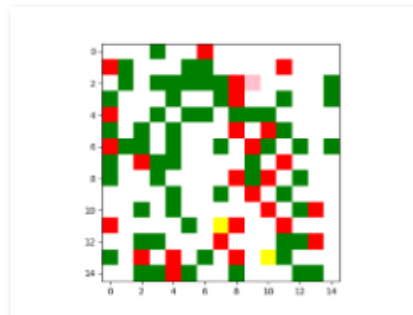
Donde se alcanza un pico de contagiados al final de los catorce días de convalecencia de los contagiados iniciales con concretamente 57 contagiados, alcanzando además en la evolución número 21 el pico de hospitalizaciones de la pandemia el cual se mantiene relativamente constante en las siguientes evoluciones con 22 hospitalizados. Durante esta simulación fallecen 8 individuos.

Como se desprende de la imagen, vemos una ola inicial de contagios masiva fruto del brote inicial, la cual produce un aumento drástico en el número de hospitalizaciones y fallecidos, tras el cese de este brote inicial y con la mayoría de población que acaba de ser contagiado y pasado la enfermedad, se aprecia que mientras dure la inmunidad de rebaño de estos contagiados iniciales, el número de contagiados no se mantiene mucho más bajo, en futuras simulaciones con más evoluciones, se podrá apreciar el carácter cíclico de las pandemias con numerosas olas en el número de contagios en cuanto la población pierde esa inmunidad de rebaño fruto del anterior pico. En este caso y con una inmunidad de tan solo cinco evoluciones, el periodo entre olas será mucho menor.

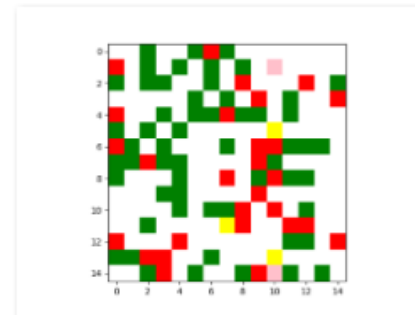
La representación gráfica del autómata celular siendo cada imagen una evolución del enrejado:



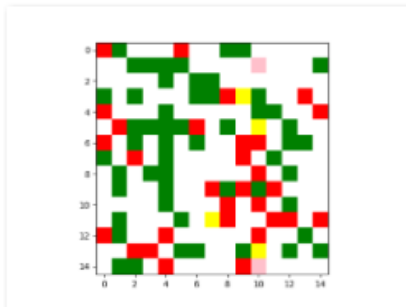
Enrejado1



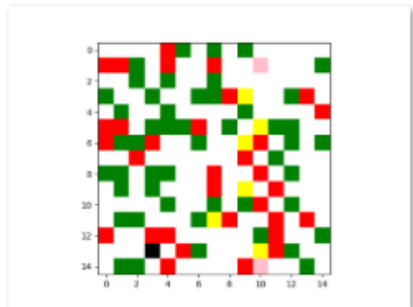
Enrejado2



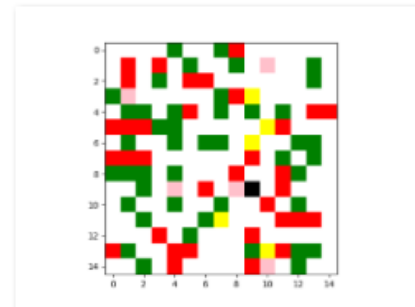
Enrejado3



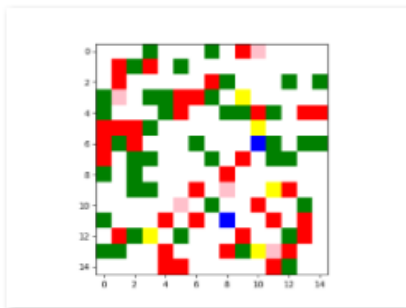
Enrejado4



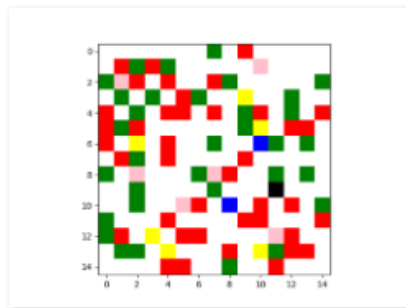
Enrejado5



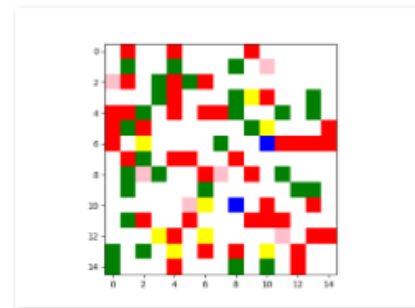
Enrejado6



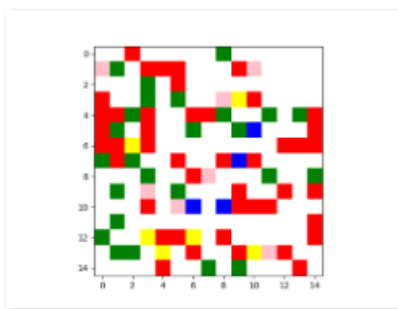
Enrejado7



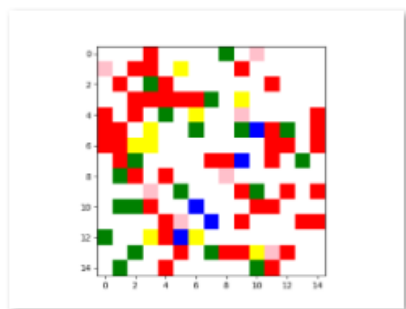
Enrejado8



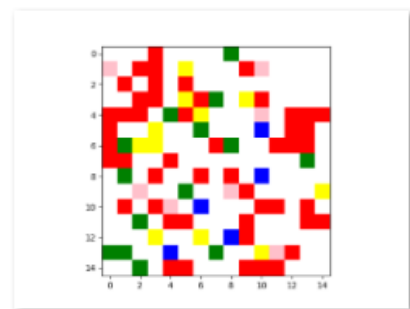
Enrejado9



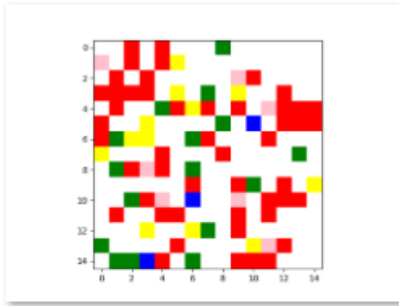
Enrejado10



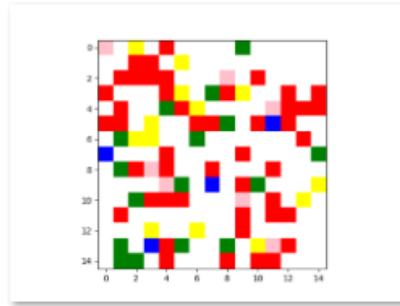
Enrejado11



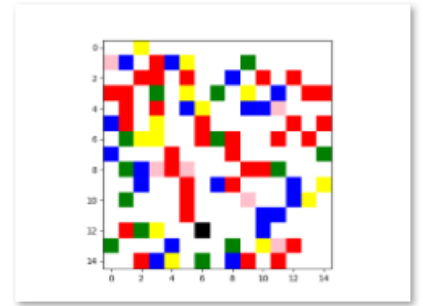
Enrejado12



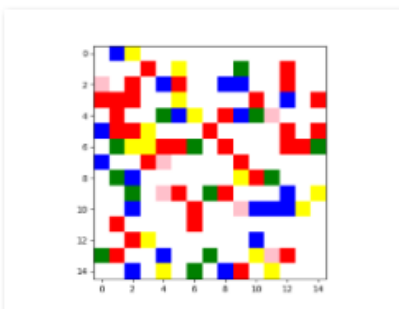
Enrejado13



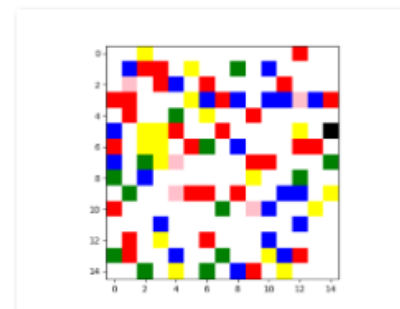
Enrejado14



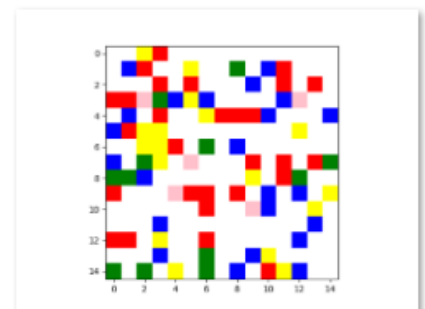
Enrejado15



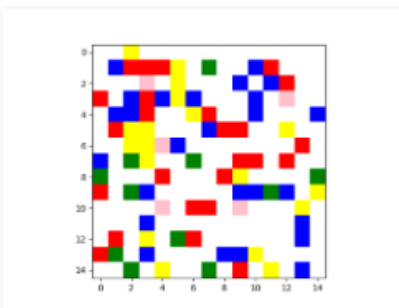
Enrejado16



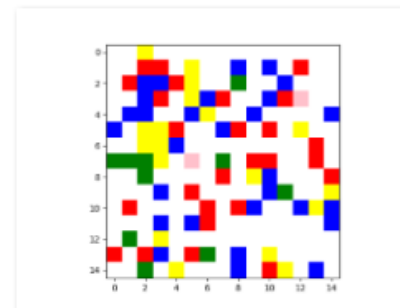
Enrejado17



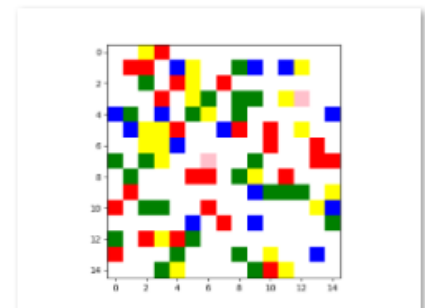
Enrejado18



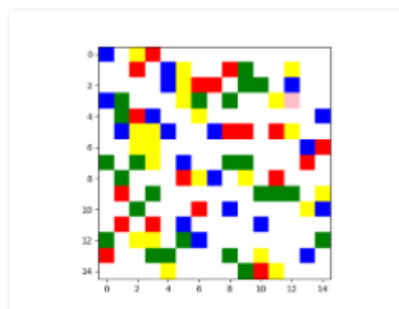
Enrejado19



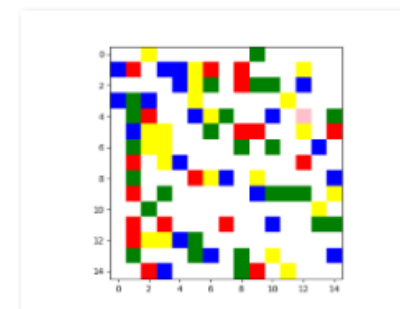
Enrejado20



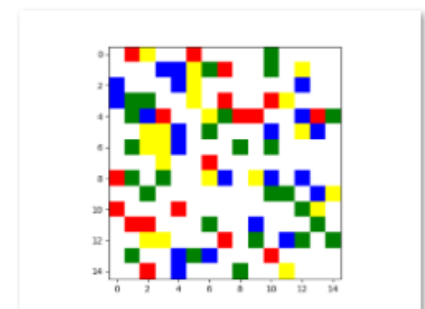
Enrejado21



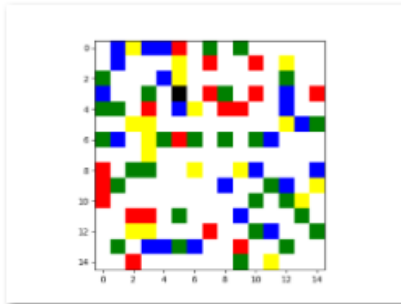
Enrejado22



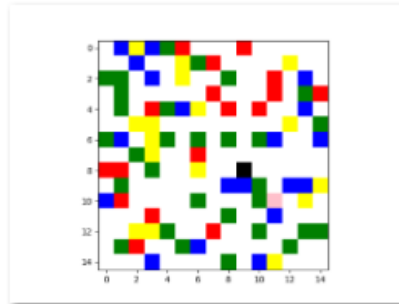
Enrejado23



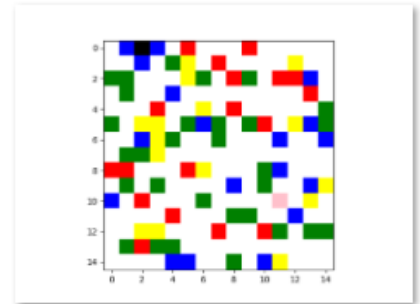
Enrejado24



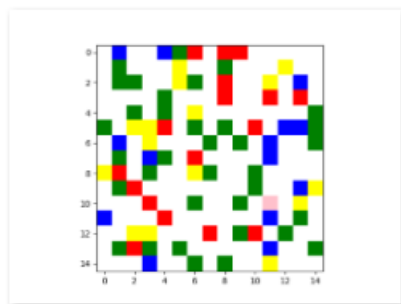
Enrejado25



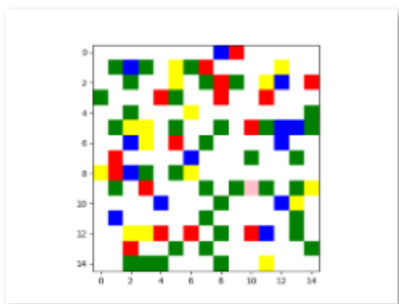
Enrejado26



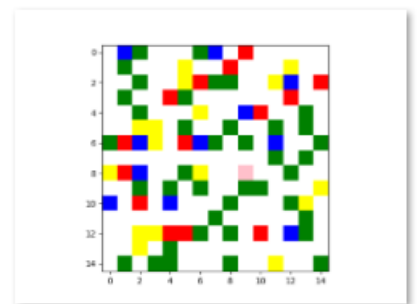
Enrejado27



Enrejado28



Enrejado29

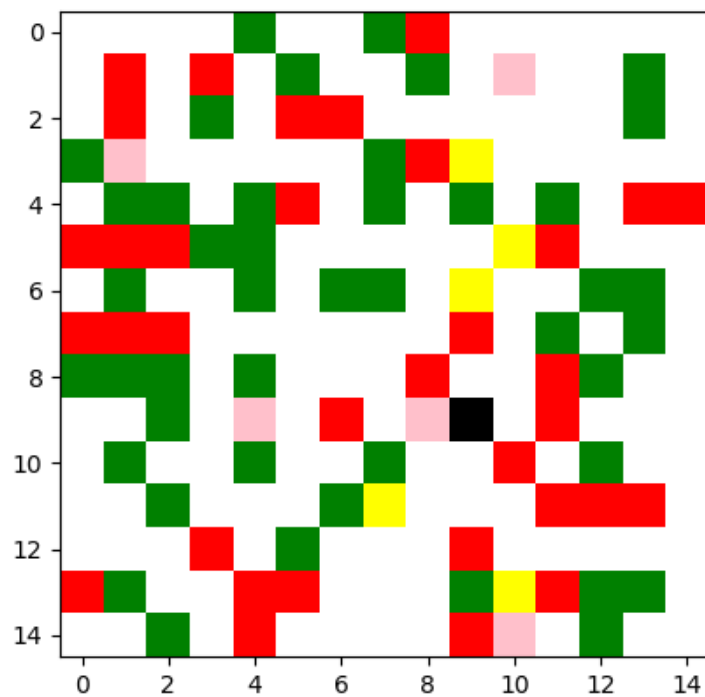
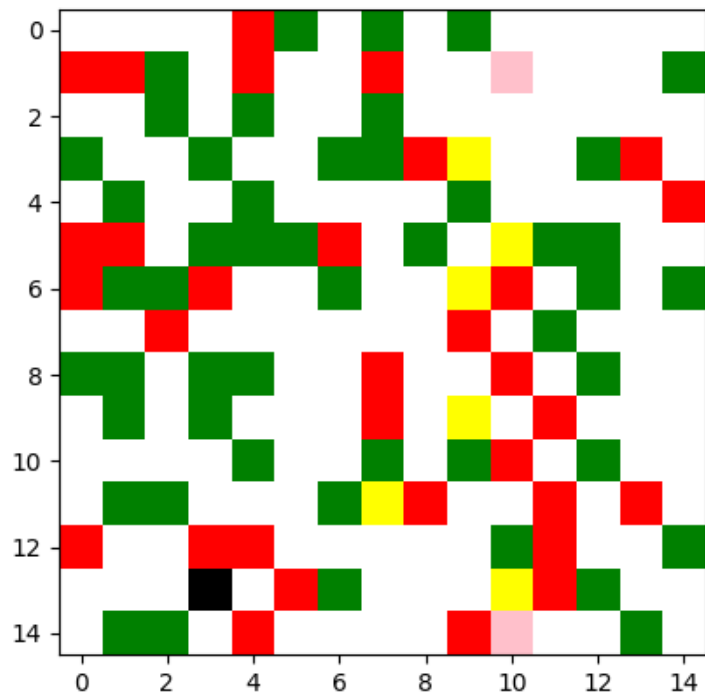


Enrejado30

Como se puede ver, todo lo mencionado a lo largo de la memoria funciona tal y como se describe, en esta ejecución como se desprende de la gráfica previamente enseñada, en las primeras cinco evoluciones predomina el verde propio de los individuos sanos, dando paso en las siguientes diez evoluciones del autómata a la peor etapa de enfermedad en donde el color mayoritario es el rojo y rosa de los contagiados. Una vez este pico de la pandemia comienza a verse en declive, surgiendo los inmunizados fruto de este pico, el enrejado pasa a verse dominado por los azules manteniendo entre las evoluciones 15 y 20 una inmunidad de rebaño la cual al desaparecer otorga cierta estabilidad a la pandemia que se mantiene hasta el final de la ejecución donde se ve un perfil diversificado en cuanto a colores.

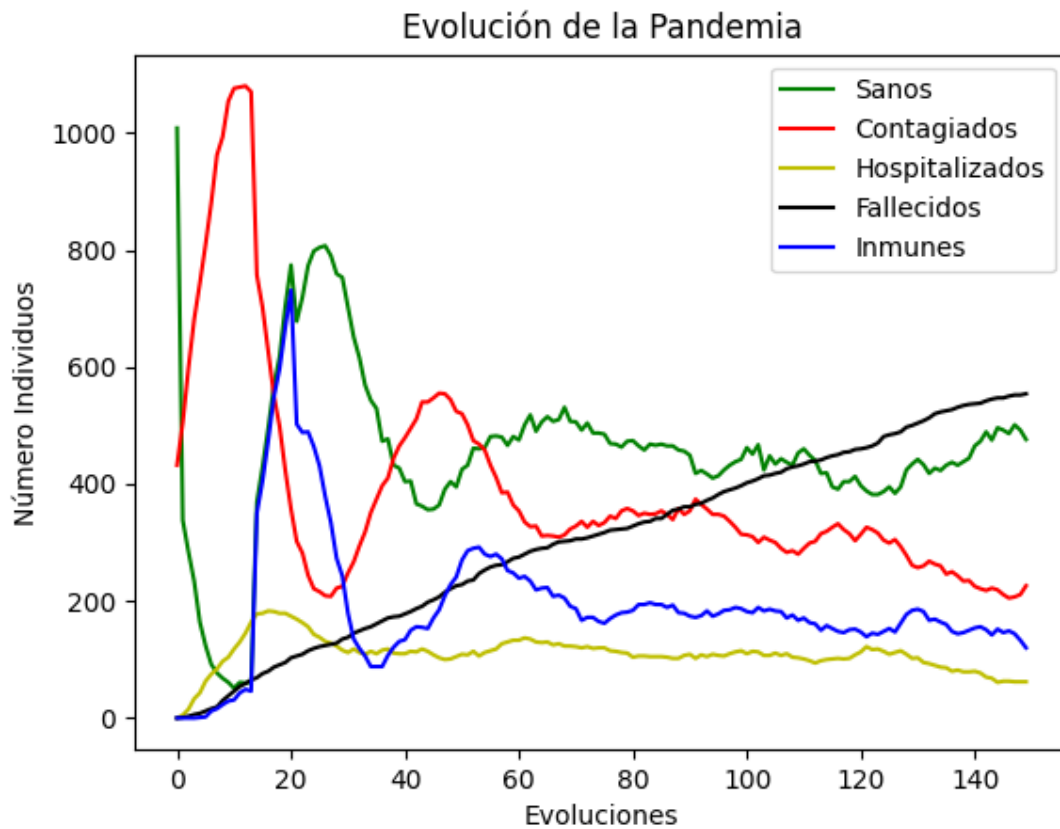
Los hospitalizados como es lógico alcanzarán su punto álgido tras la primera ola de contagios donde muchos de los inicialmente infectados se verán en el hospital combatiendo la enfermedad. En la evolución número 21 es donde esto mejor se ejemplifica. Los 8 fallecidos van surgiendo de forma esporádica durante las evoluciones siendo además el primer fallecido en la quinta evolución un fallecido doméstico, un fenómeno mucho menos probable que el resto de fallecidos de esta simulación que perecen en el hospital.

A continuación se expondrá un único paso de la simulación en mayor tamaño para que se pueda apreciar el cambio de enrejado en una única evolución del cuarto al quinto enrejado:

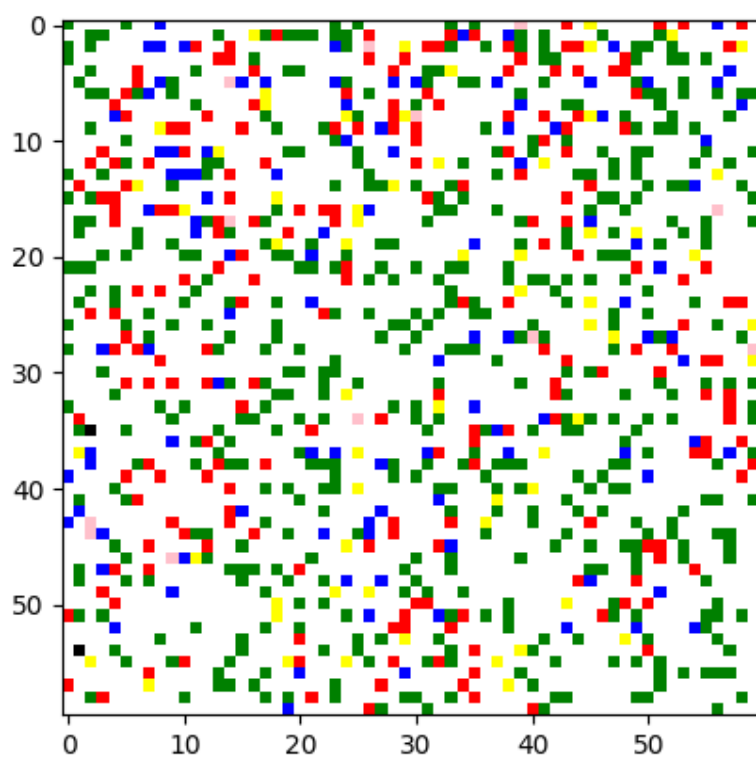
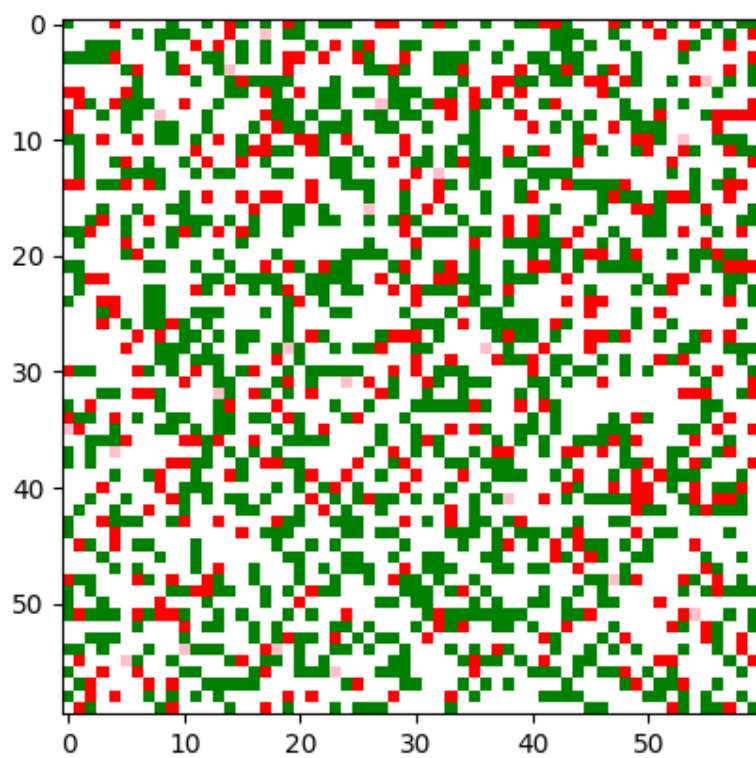


Una vez enseñado el funcionamiento de las evoluciones en el autómata celular, se procederá a hacer simulaciones con una rejilla mayor y más evoluciones para ver cómo se ve la pandemia simulada a mayor escala, en primer lugar mantendremos los mismos

parámetros tan solo multiplicaremos por cuatro la dimensión de la rejilla y simulando por 150 evoluciones

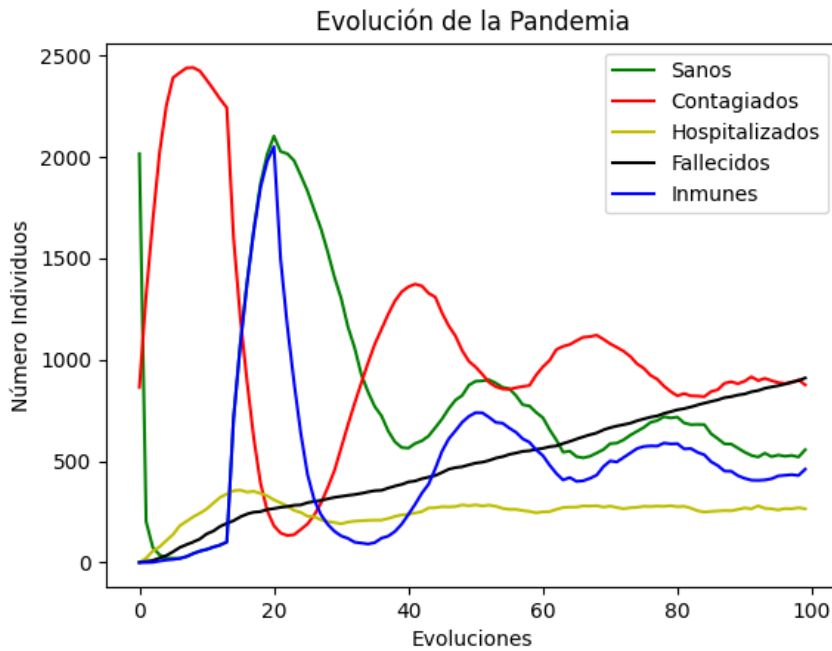


Aquí se vuelve a observar el mismo comportamiento cíclico con dos grandes olas antes de estabilizar el número de contagiados. En esta simulación a lo largo de 150 evoluciones el mayor número de contagiados simultáneos es de 1080 y en la evolución número 16 se alcanza un pico en las hospitalizaciones con 183 hospitalizados simultáneos, habiendo además un total de 554 fallecidos durante las 150 evoluciones simuladas. A modo de curiosidad, esta es la representación del enrejado en la primera y última evolución respectivamente:



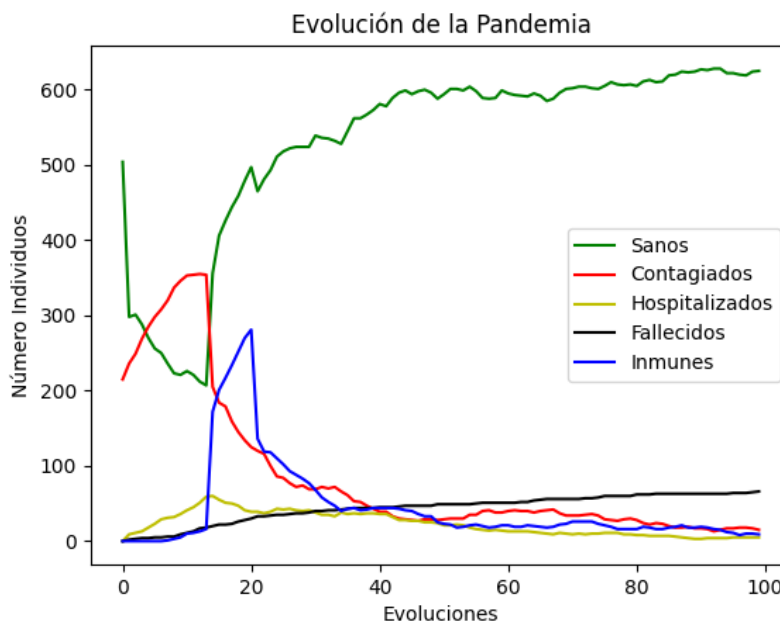
Ahora probemos a modificar algunos parámetros para ver cómo afectan a la simulación en 100 evoluciones y cómo cambia su gráfica, en primer lugar, manteniendo al resto constante, vamos a probar con una gran densidad poblacional, siendo el porcentaje de celdas vacías del 20% para simular una urbe, frente a una menor densidad poblacional, con un porcentaje de celdas vacías del 70%, también cambiaremos en ambos casos el porcentaje de individuos que se mueven por evolución al 15% para obtener datos más representativos en función de la densidad poblacional.

Densidad poblacional del 80%:



Con 910 fallecidos, lo que supone un % de fallecimientos del 31.60% sobre los individuos inicialmente generados y una máxima en el porcentaje de la población simultáneamente hospitalizada del 12.39% en la decimoquinta evolución.

Densidad poblacional del 20%:



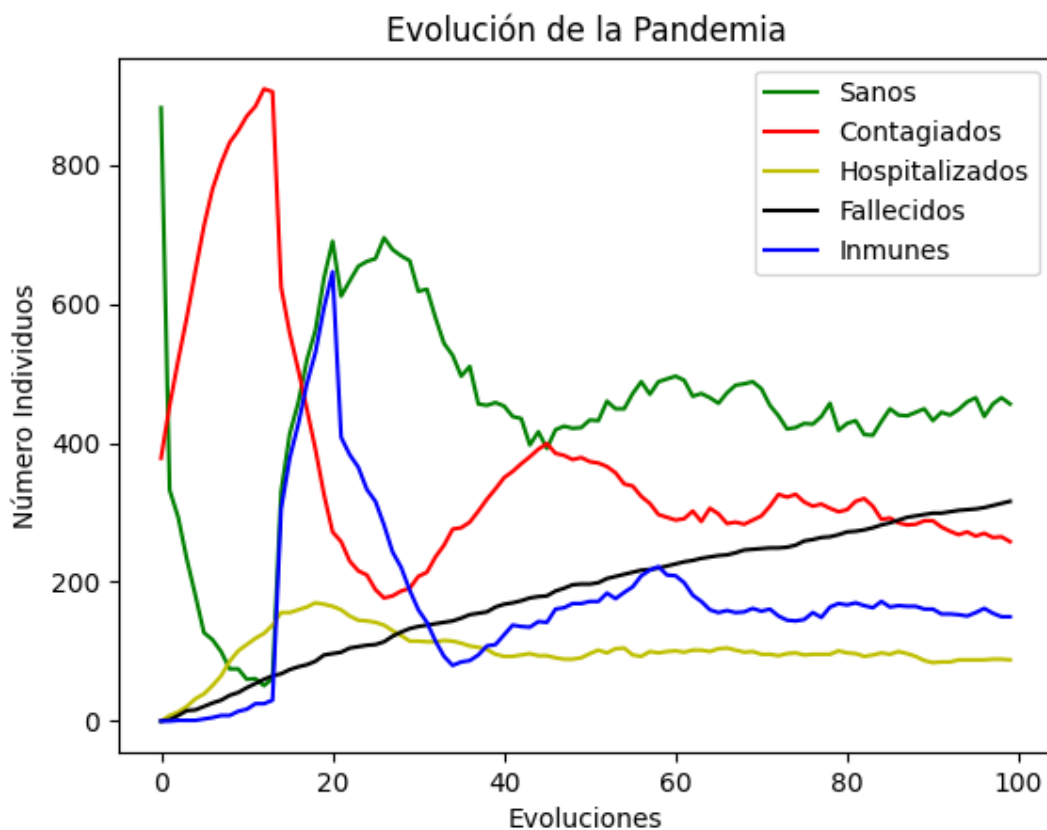


Como se puede observar en la gráfica un enorme porcentaje de la población ni siquiera llega a pasar la enfermedad y aunque esta no llega a desaparecer del todo en ningún momento, no sigue el anterior comportamiento cíclico. El pico inicial es mucho más bajo y una vez la mayoría de la población pasa la enfermedad, la pandemia se podría dar por controlada. Comparando cuantitativamente con respecto a la anterior simulación, el tanto por ciento de fallecidos sobre la población inicial es del 9.17% (22.4 puntos menos) y el punto de mayor presión hospitalario supone un porcentaje del 8.33% sobre la población (4.06 puntos menos).

Con esta comparativa se puede evidenciar claramente porqué por lo general cualquier tipo de enfermedad con un alto ratio de contagio tiende a producir efectos mucho más devastadores sobre poblaciones con mayor densidad frente a las áreas rurales donde es más raro que se puedan producir conglomeraciones multitudinarias siendo posibles focos de nuevos brotes.

A continuación pondremos una densidad poblacional fija del 35% y probaremos la diferencia entre un mayor porcentaje de individuos que se mueven o no. Esto es el equivalente de las restricciones de movilidad impuestas por numerosos gobiernos en Europa.

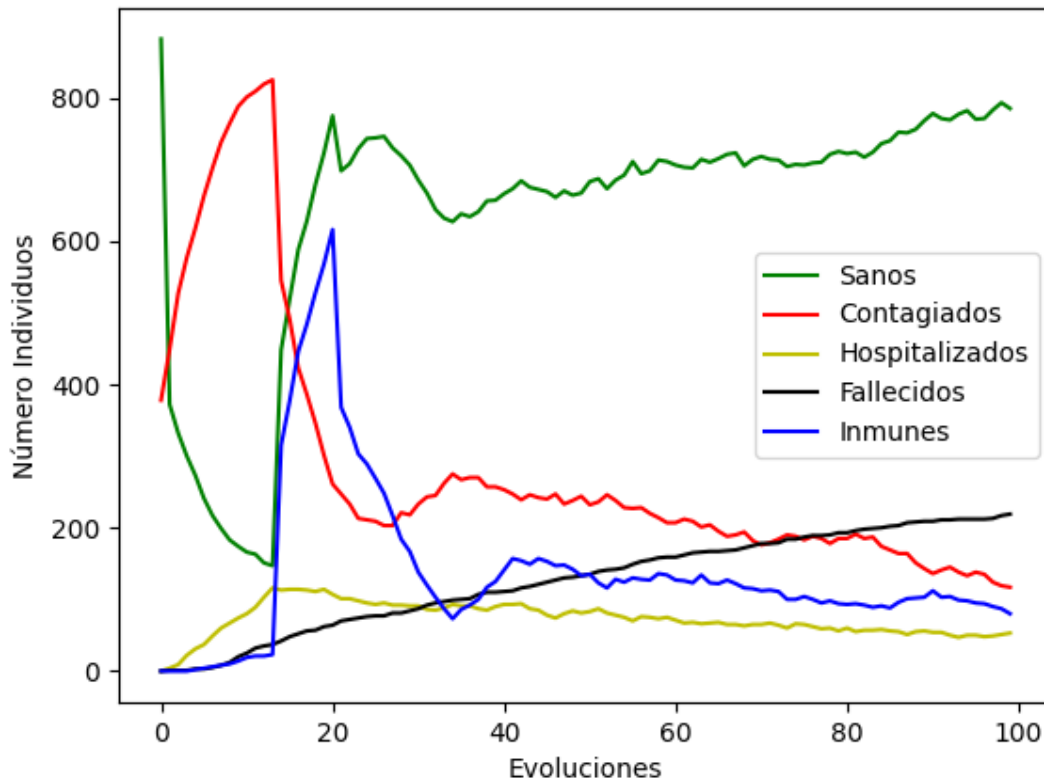
Tanto por ciento de individuos que se mueven cada evolución del 70%:



Con un % de fallecidos sobre la población inicial del 25.08%.

Tanto por ciento de individuos que se mueven cada evolución del 3%:

### Evolución de la Pandemia



Vemos como en todo momento el número de individuos sanos es significativamente mayor sin apenas haber olas significativas una vez la primera ola desaparece, obteniendo un % de fallecidos sobre la población inicial del 17.38% (una disminución en un 7.7%). Esta diferencia será mucho mayor si se hace una simulación con un menor número de contagiados, ya que sin desplazamientos y con esta densidad poblacional, se reducirá significativamente la hospitalización fruto de la primera oleada que es de donde se terminan produciendo un mayor número de decesos aunque con estos parámetros sigue habiendo una disminución de la presión hospitalaria y del número de fallecidos más que significativa.

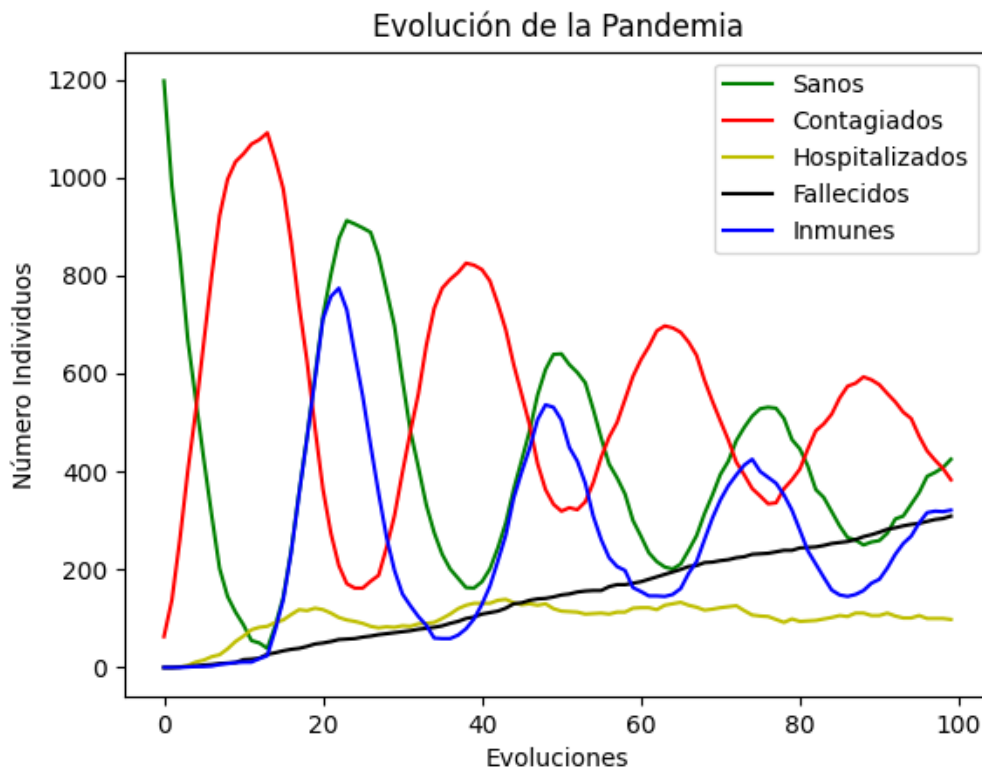
A continuación y para los siguientes parámetros:

```
# Parámetros:

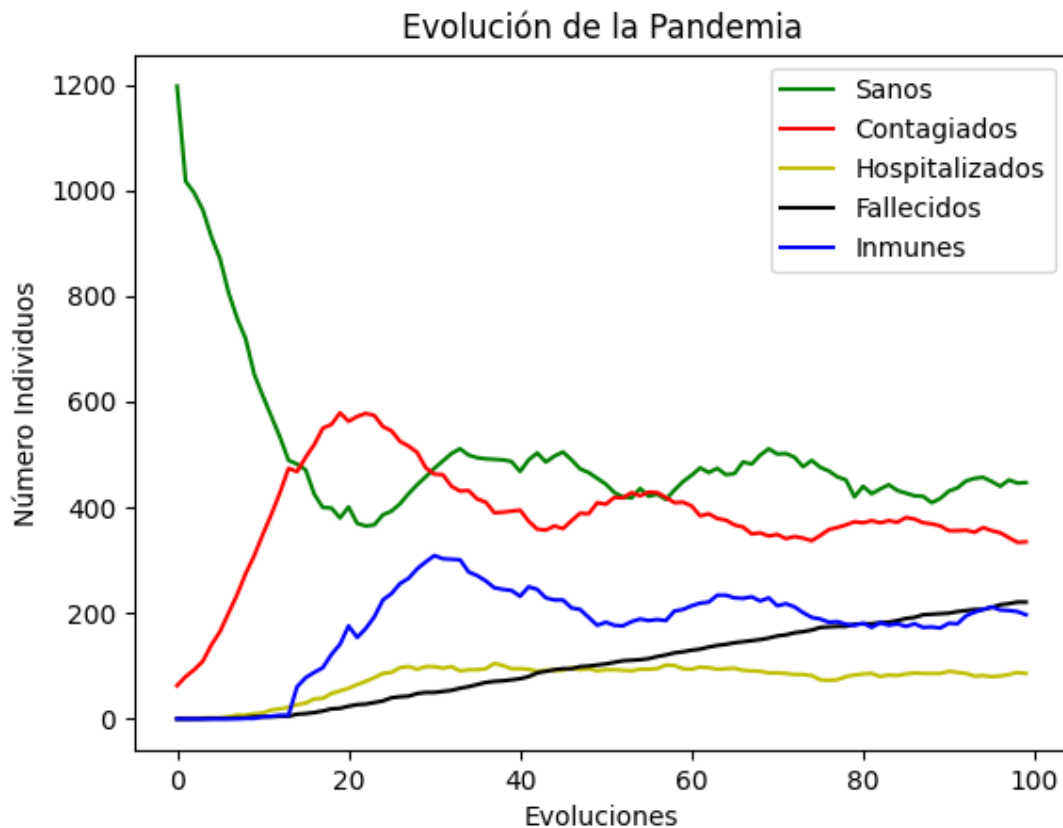
# Necesarios para la creación de la rejilla inicial:
dimension_inicial_rejilla = 60
# Número de evoluciones
numero_deevoluciones = 100
# Esto se podría extrapolar a densidad poblacional:
porcentaje_celdas_vacias = 0.65
# Porcentaje de individuos que se mueven:
porcentaje_individuos_mueven = 0.3
# Porcentaje de individuos sanos o enfermitos:
porcentaje_sanos = 0.95
porcentaje_contagiados = 0.05
porcentaje_asintomaticos = 0.3
#-----
# Datos del virus-----
porcentaje_de_hospitalizaciones = 0.02
porcentaje_fallecidos_en_hospital = 0.015
porcentaje_fallecidos_domesticos = 0.005
probabilidad_diaria_salir_del_hospital = 0.05
```

Cambiaremos el ratio de contagio, el cual apoyado por una menor incidencia inicial, producirá los siguientes cambios en las gráficas.

Con un ratio de contagio por vecino adyacente contagiado del 33% :



En cambio con un 10% :



Vemos como en la primera gráfica, se ve este comportamiento cíclico perfectamente simulado ya que en el momento en el cual los inmunes fruto de una ola dejan de serlo, vuelven a contagiarse lo cual hace que la presión hospitalaria se mantenga en todo momento por las nubes, llegando además en esta simulación al máximo en la evolución número 43 (cuando lo normal es que sea poco después de las 14 primeras evoluciones donde están hospitalizados los enfermos del primer brote). En cambio la virulencia de las olas en la segunda gráfica es mucho menor, así como la incidencia, siendo el mayor pico de la segunda gráfica 46.93% menor que el de la primera. También en la segunda gráfica se ve un tanto por ciento de fallecimientos en la población inicial del 17.54% frente al 24.52% de la primera muestra lo que no hace sino evidenciar la obvedad de la correcta simulación frente a las diferencias en el ratio de contagio.

Por último y a modo de curiosidad, cambiaré los parámetros respecto a los fallecimientos (los cuales en las anteriores simulaciones simulan una enfermedad con una altísima tasa de mortalidad) y dejaré al autómata simularse durante mil evoluciones para observar el comportamiento de la enfermedad con los siguientes parámetros:

```

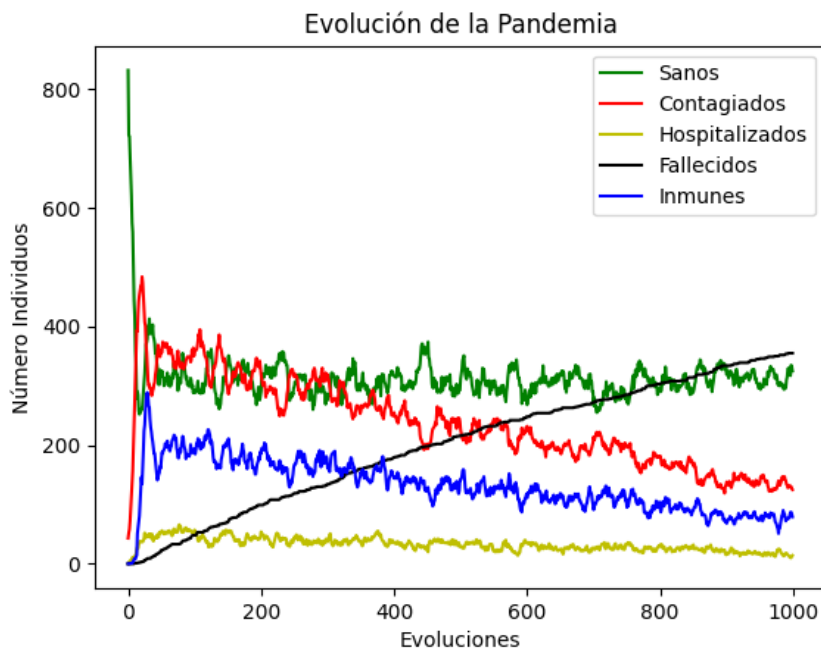
# Parámetros:

# Necesarios para la creación de la rejilla inicial:
dimension_inicial_rejilla = 50
# Número de evoluciones
numero_deevoluciones = 1000
# Esto se podría extrapolar a densidad poblacional:
porcentaje_celdas_vacias = 0.65
# Porcentaje de individuos que se mueven:
porcentaje_individuos_mueven = 0.3
# Porcentaje de individuos sanos o enfermitos:
porcentaje_sanos = 0.95
porcentaje_contagiados = 0.05
porcentaje_asintomaticos = 0.3
#-----
# Datos del virus-----
porcentaje_de_hospitalizaciones = 0.03
porcentaje_fallecidos_en_hospital = 0.01
porcentaje_fallecidos_domesticos = 0.0001
probabilidad_diaria_salir_del_hospital = 0.13

ratio_de_contagio = 0.13
#-----

```

Obteniendo la siguiente gráfica:



La cual aunque evidencia el correcto funcionamiento del autómata celular. Demuestra también que para obtener una simulación más realista tengo que bajar aún más los parámetros resultantes en la tasa de mortalidad que en 1000 evoluciones es del: 40.57%

---

## Capítulo 7

# Conclusiones

---

En este capítulo reflexionaré sobre el trabajo realizado y el nivel de logro del mismo como solución al problema planteado, expondré cuáles serían las posibles mejoras y apartados a añadir al proyecto en caso de continuarse y llegaré a las conclusiones oportunas acerca del trabajo realizado.

### 7.1 Simulación covid-19

---

Dada la inexistencia de herramientas técnicas que tanto nos hubieran podido ayudar sobre todo durante el comienzo de la emergencia sanitaria que ha dictado nuestra realidad durante ya más de un año, decidí escoger este trabajo y aunque se trate de un autómata celular con un set de reglas relativamente sencillo y de propósito general, siendo una prueba de concepto, veo indicado que una vez se ha probado en el capítulo anterior la relativa validez del método empleado, se pruebe con los parámetros del virus que inspiró la motivación para realizar este trabajo.

Esto supone una enorme dificultad ya que los datos en función de dónde sean recogidos o el momento de la extracción de los mismos puede variar enormemente. Por ello he decidido basarme en un documento redactado y actualizado a 25 de marzo de 2021 por el Centro de Coordinación de Alertas y Emergencias Sanitarias [5] en el cual se facilitan ciertos datos que con el siguiente post procesamiento servirán como parámetros para ver cuán precisa y útil podría llegar a ser una herramienta como esta en situaciones similares.

En este documento se estipula que las probabilidades de requerir hospitalización para un individuo residente en España es del 7.4% en caso de contraer el coronavirus. Dado que el parámetro equivalente es la probabilidad diaria de hospitalización para los contagiados sintomáticos, dividiré este número entre los 14 días de convalecencia para averiguar el parámetro. También se indica que la estancia media de un hospitalizado por COVID-19 es de 10 días, por lo que la probabilidad diaria de abandonar el hospital será de un diez por ciento, durante la cual el Centro de Coordinación de Alertas y Emergencias Sanitarias estipula en un 1.4% la probabilidad de defunción. Por lo que dada la estancia media de diez días en el hospital, resultará en el parámetro de probabilidad diaria de fallecimiento en el hospital de un 0.14 %.

Aparte, he encontrado en un artículo médico [6] que el tanto por ciento de personas que pasan la enfermedad sin síntomas es del 59% y por último, a pesar de no haber encontrado porcentajes oficiales de fallecimiento domiciliario, lo incorporaré con un valor insignificante ya que aunque los enfermos graves por lo general son trasladados a un hospital, han existido casos que ya sean por un motivo u otro han acabado pereciendo fruto de los síntomas del coronavirus en su propia residencia.

El resto de parámetros, como el ratio de contagio, el porcentaje de enfermos iniciales o el porcentaje de individuos que se mueven cada evolución no es tan fácil extraerlo, ya que

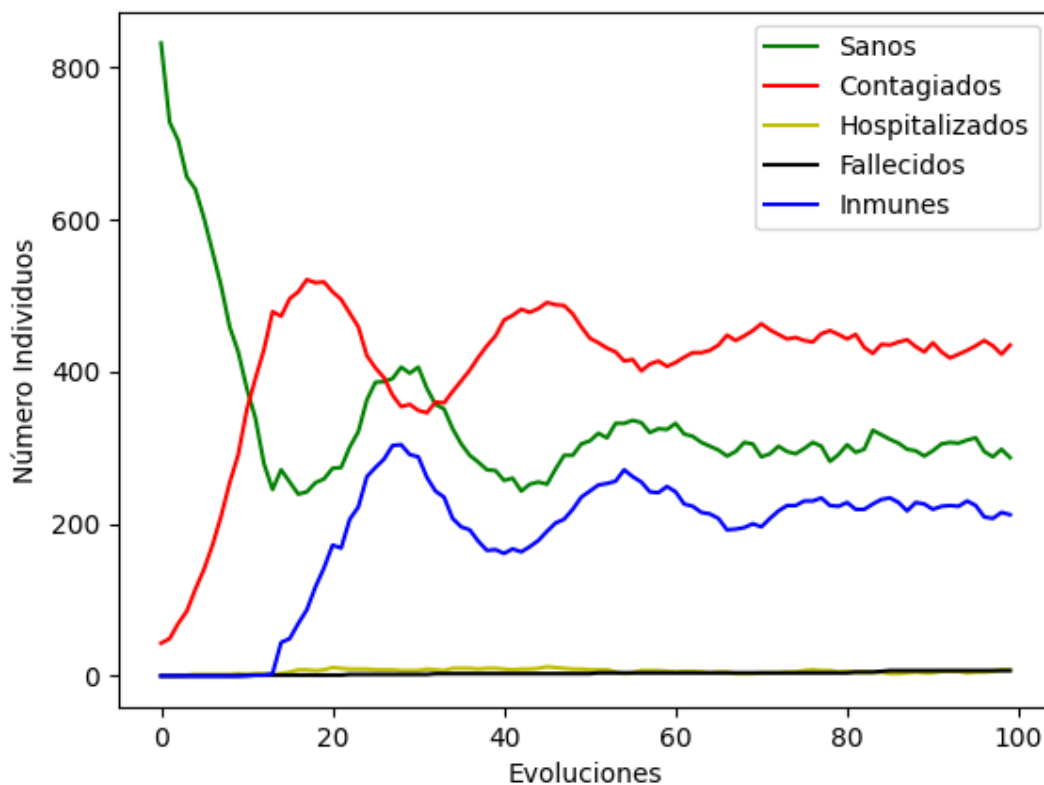
habría que de alguna manera extrapolarlo de datos reales, pero utilizaré los siguientes números con carácter orientativo:

```
# Parámetros:

# Necesarios para la creación de la rejilla inicial:
dimension_inicial_rejilla = 50
# Número de evoluciones
numero_deevoluciones = 100
# Esto se podría extrapolar a densidad poblacional:
porcentaje_celdas_vacias = 0.65
# Porcentaje de individuos que se mueven:
porcentaje_individuos_mueven = 0.3
# Porcentaje de individuos sanos o enfermitos:
porcentaje_sanos = 0.95
porcentaje_contagiados = 0.05
porcentaje_asintomaticos = 0.59
#-----
# Datos del virus-----
porcentaje_de_hospitalizaciones = 0.00514
porcentaje_fallecidos_en_hospital = 0.0014
porcentaje_fallecidos_domesticos = 0.0000001
probabilidad_diaria_salir_del_hospital = 0.1

ratio_de_contagio = 0.13
#-----
```

Evolución de la Pandemia



Durante la cual fallecen un individuos, lo que supone una tasa de mortalidad sobre la población inicial en 100 días del 0.114%.

Vemos con esto que son números perfectamente factibles y que la gráfica asemeja el comportamiento del coronavirus aún sin haberse incorporado medidas de protección en la simulación.



## 7.2 Posibles mejoras futuras

---

Dado que la herramienta pretende simular la evolución de una pandemia, eso se ha hecho en este proyecto, aún así, existen ciertos posibles mejoras a implementar para darle un mayor uso en caso de continuar su desarrollo.

- En primer lugar, una **mejora temporal** de las ejecuciones implementando paralelismo para el cálculo de las evoluciones tal y como se menciona en el apartado 4.3.
- También se habría de implementar un sistema de **detección y aislamiento** frente a la enfermedad en el cual se realice en cada evolución un porcentaje de test con un tanto por ciento de acierto sobre la población contagiada sintomática, forzando a un confinamiento domiciliario a aquellos positivos confirmados.
- La incorporación de **nuevos individuos** a la simulación a medida que transcurren las evoluciones. Dado que aquí se trataba de estudiar la incidencia de la pandemia sobre una población aislada objetivo, esto no se implementó. Pero para una herramienta más completa se tendrían que simular desplazamientos desde otros núcleos poblacionales con distintas tasas de incidencia.
- Incorporar una **diferenciación de edades** para que los parámetros del virus puedan variar en función de la edad y patologías previas de cada individuo (esto sería algo más complejo de hacer con autómatas celulares ya que se trata de hacer reglas más generales y no llevar un control tan estricto sobre cada individuo en particular).
- La incorporación tras un número ya sea aleatorio o introducido por el usuario de **campañas de vacunación** con distintos porcentajes de protección y ritmos de vacunación.
- La incorporación de posibles **variantes o mutaciones del virus** que suponga ligeros cambios sobre los parámetros inicial y dotando de mayor realismo a la simulación.

## 7.3 Conclusiones finales

---

Una vez realizado el trabajo, he de responder al logro en el propósito del mismo. Como se ha mencionado en múltiples ocasiones, se trata de una prueba de concepto acerca de si un autómatas celular puede simular el comportamiento de una pandemia basándose en un conjunto inicial de normas básicas. La respuesta a esta pregunta es un rotundo sí, hemos visto como apenas un mínimo set de normas que regulan la evolución entre los distintos estados, se ha logrado una herramienta capaz de simular gráficas y evoluciones realistas para la evolución de pandemias en poblaciones aisladas.

Creo que esta herramienta podría ser de utilidad para un ministerio como el de Salud pública para tener un informe más que presentar a su consejo de expertos en base al cual ellos tomarán las decisiones oportunas. Obviamente en caso de querer utilizarse como herramienta con uso real más allá de como prueba de concepto, yo personalmente optaría por introducir algunas de las mejoras propuestas en el anterior apartado para dotar al mismo de un mayor realismo.

El beneficio de usar autómatas celulares frente a otra tecnología para el desarrollo de una herramienta como esta es innegable. Otras tecnologías de inteligencia artificial que también podrían enfrentarse al problema como redes neuronales o aprendizaje máquina, tendrían necesariamente que contar con una ingente cantidad de data previamente recolectada la cual durante el comienzo tras un brote de una nueva enfermedad, todavía no estaría disponible. Sí que podrían, una vez se tengan estos datos, llegar a resultados mucho más precisos que casi cualquier herramienta basada en autómatas celulares. Pero tanto la cantidad previa de datos como el tiempo de computación invalidará cualquier uso en la primera respuesta ante el azote de una nueva emergencia sanitaria.

Por ello el uso de autómatas celulares tratando de simular comportamientos enormemente complejos de la naturaleza brilla por su capacidad de dar sentido a números a priori pseudo aleatorios y servir como una herramienta con base científica donde actualmente existe un vacío casi absoluto.

Con todo lo dicho en este trabajo, veo cumplido mi propósito de generar una herramienta capaz de en cierto modo poder predecir y simular la evolución de una pandemia mediante autómatas celulares.



## **Bibliografía:**

[0] = "A Contribution to the Mathematical Theory of Epidemics." by William Ogilvy Kermack and A. G. McKendrick.

Se puede encontrar el artículo completo aquí:

<https://royalsocietypublishing.org/doi/10.1098/rspa.1927.0118>

[1] =

[https://www.researchgate.net/publication/6330893\\_Excitable\\_Greenberg-Hastings\\_cellular\\_automaton\\_model\\_on\\_scale-free\\_networks](https://www.researchgate.net/publication/6330893_Excitable_Greenberg-Hastings_cellular_automaton_model_on_scale-free_networks)

[2]= <https://www.wolframscience.com/nks/>

[3] = <https://sites.google.com/site/jimmycifuentes/EvolvingCAMusic.pdf?attredirects=0>

[4] = <http://www.it.uc3m.es/jvillena/irc/practicas/09-10/04mem.pdf>

[5] =

[https://www.mscbs.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov/documentos/Documento\\_GRUPOS\\_PERSONAS.pdf](https://www.mscbs.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov/documentos/Documento_GRUPOS_PERSONAS.pdf)

[6] =

[https://www.consalud.es/pacientes/especial-coronavirus/asintomaticos-responsables-transmission-sars-cov-2-59-casos\\_90868\\_102.html](https://www.consalud.es/pacientes/especial-coronavirus/asintomaticos-responsables-transmission-sars-cov-2-59-casos_90868_102.html)

[7] = Introduction to Modeling of Complex Systems Using Cellular Automata by Alfons G. Hoekstra, Jiri Kroc, and Peter M.A. Sloot

Concretamente se ha extraído información de los siguientes capítulos:

- Introduction to Modeling of Complex Systems Using Cellular Automata
- Cellular Automata Composition Techniques for Spatial Dynamics Simulation
- Minimal Cellular Automaton Model of Inter-species Interactions: Phenomenology, Complexity and Interpretations