Facultat d'Informàtica de Barcelona

GIA UPC

Optimització

Treballant amb CA i heurístiques

Alumnes :
Abril Risso Matas
Irene Pumares Benaiges

Tutors :
Pau Fontseca

May 6, 2024

Contents

1 Part 1

1.1 Introducció

En aquesta primera part de la pràctica, ens endinsarem en la programació i simulació d'autòmats cel·lulars. L'objectiu serà implementar un autòmat cel·lular que segueixi les regles de Wolfram, un conjunt de regles elementals que es poden consultar a MathWorld. Aquestes regles defineixen el comportament de les cel·les en una graella unidimensional i són un exemple clàssic en l'estudi dels sistemes complexos i la teoria del caos.

Tant en aquesta primera part com en la següent s'utilitzarà el llenguatge *Python* i específicament per aquesta primera part s'ha emprat l'eina Chat GPT versió 4 per la facilitació de generació del codi.

Es generalitzarà l'autòmat per permetre combinar diverses regles en un mateix sistema i finalment es mostrarà l'evolució temporal de l'autòmat mitjançant *Pygame*.

1.2 Implementació del Codi

1.2.1 regla_wolfram

Aquesta funció converteix un número de regla en la seva representació binària. Cada bit de la representació binària correspon a una possible configuració de tres cel·les veïnes (esquerra, centre, dreta), i determina l'estat de la cel·la central en el següent pas de temps.

La funció rep com a paràmetre una regla de wolfram que prèn un número del 1 fins al 256, i fa servir la funció de Numpy **np.binary_repr()** per convertir aquesta regla a una cadena de caràcters que representen el seu valor en binari (cada element es un 0 o un 1).

L'element de cada posició ens indica si la cel·la central serà activa o inactiva en el següent pas del temps.

1.2.2 actualitzar

Aquesta funció és essencial per l'evolució de l'autòmat cel·lulat basat en la regla específica indicada.

La funció calcula l'estat següent de l'autòmat cel·lular utilitzant l'estat actual i una regla binària proporcionada. L'objectiu és determinar si cada cel·la estarà activa (1) o inactiva (0) al següent pas de temps basant-se en l'estat de les cel·les veïnes.

La funció rep com a paràmetres **estat_actual** que representa l'estat actual de l'autòmat en un array de Numpy on cada element es 0 o 1 i **regla_binaria** que representa l'array de Numpy obtinguda a la funció *regla_wolfram*.

Primer de tot, es crea un array de zeros amb la mateixa mida que estat_actual per emmagatzemar el proper estat de l'autòmat. Seguidament, genera un array d'índexs que exclou el primer i l'últim element, evitant així problemes amb cel·les que no tenen tres veïns.
A continuació, defineix tres arrays (esquerra, centre, dreta) que representen els estats dels veïns a l'esquerra, al centre i a la dreta de cada cel·la, respectivament i utilitza una fórmula per convertir la configuració local de tres cel·les en un índex únic que es correspon amb un bit en la regla_binaria. Aquest índex s'obté invertint el valor decimal resultant de la configuració de veïns, ja que les regles de Wolfram s'enumeren de dreta a esquerra. Finalment, assigna a estat_nou els valors corresponents de regla_binaria basant-se en els patrons calculats, actualitzant així l'estat de cada cel·la (excepte les dels extrems).

1.2.3 combinar

Aquesta funció permet la combinació de diversos estats d'autòmats cel·lulars mitjançant operacions lògiques bit a bit.

S'ha decidit que aquestes operacions lògiques siguin: and, or i xor. Per tant, l'usuari podrà decidir quina d'aquestes operacions lògiques aplicar canviant el paràmetre *metode*

per **'and', 'or' o 'xor'**.

La funció rep com a paràmetres *estats*, que és una llista d'arrays de Numpy on cada array representa l'estat d'un autòmat cel·lular en un moment donat, i el paràmetre *metode* que, com ja s'ha comentat abans, representa l'operació lògica triada per combinar les diverses regles.

La funció retorna un array de NumPy que representa l'estat combinat resultant de l'aplicació de l'operació bit a bit escollida a tots els estats d'entrada.

1.2.4 evolucio

Aquesta funció utilitza la llibreria *Pygame* per visualitzar l'evolució temporal d'un autòmat cel·lular en una interfície gràfica, mostrant cada estat del procés en una graella visual. Això permet observar detalladament com es desenvolupen els diferents estats de l'autòmat a través del temps.

La funció comença amb la inicialització de *Pygame* i la configuració de la mida de la finestra gràfica basada en les dimensions de l'array estats_combinats multiplicat pel tamany_pixel, determinant l'espai visual de cada cel·la.

Per cada cel·la de l'autòmat, es determina el color segons l'estat: negre per a l'estat actiu (1) i blanc per a l'estat inactiu (0).

1.2.5 automata_celular

Aquesta funció té l'objectiu de cridar a les funcions anteriors per generar, combinar i visualitzar els autòmates cel·lulars donat un conjunt de regles de Wolfram.

Comença definint una llista buida estats per emmagatzemar els resultats de cada regla aplicada. Per cada regla especificada en el paràmetre *regles*, converteix la regla en la seva forma binària utilitzant la funció *regla_wolfram*. Seguidament, inicialitza un array de zeros que representarà l'evolució de l'autòmat al llarg del nombre especificat de *passos*, amb la mida especificada per *mida*. La simulació comença amb una condició inicial on només la cel·la central està activa.

A continuació, per cada pas temporal, utilitza la funció *actualitzar* per calcular l'estat següent de l'autòmat basant-se en l'estat anterior, aplicant la regla binària corresponent. Aquest procés es repeteix per cada regla, i cada estat resultant s'afegeix a la llista estats.

Si hi ha més d'un estat en la llista *estats*, els estats són combinats utilitzant l'operació lògica especificada en metode ('and', 'or' o 'xor') per la funció *combinar*. Si només hi ha un estat, aquest es pren com l'estat combinat final. Finalment, la funció *evolucio* és cridada per visualitzar l'estat combinat en el temps.

1.3 Assumicions en el codi

1. **Inicialització en el centre:** La simulació de l'autòmat cel·lular comença amb una única cèl·lula activa en el centre de la primera fila (estat inicial).

2. **Condicions de la frontera:** Són gestionades de manera que les cel·les dels extrems de l'array (o fronteres del sistema d'automàt cel·lular) no s'actualitzen durant la simulació. Això significa que el comportament d'aquestes cel·les dels extrems es manté constant, o bé continuen amb el seu estat inicial durant tota la simulació.

3. **Combinació de regles:** Prèviament a la creació del codi, s'ha decidit que la combinació de les regles es farà un cop creats els autòmats individuals, és a dir, cada automat implementarà una regla de Wolfram específica i per la combinació d'aquests es farà mitjançant la combinació binària de regles, on el resultat serà la combinació binària de les dues regles.

2 Part 2

2.1 Introducció

En aquesta segona part de la pràctica, es modelitzarà la propagació d'un incendi forestal utilitzant dades que simulen condicions reals. Aquesta part implica l'ús d'un model computacional que ens permetrà definir una funció d'actualització per l'autòmat.

Per realitzar aquesta tasca farem ús de dades estructurades segons el format de IDRISI32. Aquestes dades seran obtingudes a partir de la creació de fitxers on cadascun d'ells aporta informació específica sobre les característiques del terreny que poden influir en la propagació de l'incendi.

El model desenvolupat tal i com s'explicarà a continuació, gestiona 3 capes diferents. Les quals són carregades des de fitxers independents (humitat, vegetació i estat de l'incendi). La última capa (estat de l'incendi) no serà carregada des d'un fitxer ja que totes les caselles s'inicialitzen amb el mateix estat. Per tant, no cal un fitxer específic per proporcionar aquesta informacio ja que en tots els incendis serà la mateixa.

Aquesta part de la pràctica ens permetrà aplicar conceptes teòrics sobre models dinàmcs a una situació pràctica. D'aquesta manera obtindrem una millor comprensió sobre com els diversos factors interactuen en el fenòmen d'incendi forestal.

2.2 Implementació del Codi

El model té tres capes principals que són essencials per a la simulació del foc: humitat, vegetació i estat de l'incendi.

Les variables 'humitat' i 'vegetació' són els elements clau per comprendre i simular la propagació del foc. A partir dels seus valors, es pot construir la tercera capa, l'estat de l'incendi, que representa el comportament del foc.

- Humitat: nombre d'hores durant les quals la cel·la no es crema
- Vegetació: nombre d'hores que triga a cremar-se una cel·la

El codi s'ha dividit en dos fitxers principals. El primer fitxer, `main`, és el punt d'entrada del programa i s'encarrega d'inicialitzar la simulació. El segon fitxer, `funcions_auxiliars`, conté les funcions específiques per a realitzar tasques com la creació de llacs, la inicialització de vegetació, la simulació de la propagació de l'incendi, i la visualització de les diferents capes.

Aquesta divisió permet que el fitxer principal es centri en el flux general de la simulació, mentre que les funcions auxiliars s'ocupen de les operacions més detallades.

Per la creació de les capes, es llegiran dos fitxers diferents en format IDRISI32 per cada capa.

- .doc: Aquest fitxer descriu la capa de dades corresponent. Ens servirà per obtenir les dimensions de la capa i així permetre la seva generació.
- .img: Aquest fitxer representa la informació de cada cel·la de la capa, corresponent al nivell d'humitat, de vegetació o l'estat de l'incendi.

2.2.1 main

Aquest és el fitxer principal des del que s'executa el programa, és a dir, és l'script que s'ha d'executar per a iniciar el programa que genera l'incendi.

En primer lloc, s'importen totes les llibreries necessàries per al programa, així com el fitxer `funcions_auxiliars`.

Aleshores conté el bloc (`if __name__ == "__main__"`) que assegura que el codi s'executi només si es crida directament, evitant la execució no intencionada quan s'importa des d'un altre mòdul. Dins d'aquest bloc conté tot el fluxe principal del programa, que crida a les funcions necessàries per al funcionament del programa.

Quan s'executa aquest fitxer, es demana a l'usuari dos valors d'entrada: la mida de la graella i el nombre de llacs, assegurant-se que siguin vàlids abans de continuar. Aquests valors s'assignen a les variables corresponents i aleshores es comencen a crear els fixters,

i per a això, s'imprimeix un missatge que ho indica.

A continuació, es procedeix a la creació de dos fitxers .img i .doc per cada capa de la simulació on ens aportan la informació necessària sobre l'estat inicial del terreny i de l'incendi. En primer lloc, es crea la matriu NumPy que representarà la humitat. Per fer-ho, es crida la funció auxiliar `crear_llacs()`, passant-li com a paràmetres els dos valors obtinguts de l'usuari. Un cop s'ha creat la matriu de la humitat, s'utilitza la funció `genera_fitxer_img()` i `genera_fitxer_doc()` per a crear els fitxers corresponents per aquesta capa. El mateix procediment s'aplica per a la matriu que representa la vegetació, però aquesta vegada es crida la funció `inicialitzar_vegetacio()` per crear la matriu.

Després de crear cada un d'aquests fitxers .img i .doc, s'informa a l'usuari mitjançant un missatge que indica que el fitxer s'ha creat correctament.

Per a l'estat de l'incendi, es genera una matriu de zeros, representant que cap casella està cremant-se inicialment.

Tot seguit, es procedeix a llegir les dades dels fitxers `humitat.img`, `humitat.doc` i `vegetacio.img`, `vegetacio.doc` creats anteriorment cridant a la funció `llegir_dades`. On, aquesta funció crida a `llegir_dades_doc` i `llegir_dades_img`, en aquestes funcions, com s'ha mencionat anteriorment, es llegeix les dimensions de la graella i les dades de cadascuna de les caselles per la capa corresponent respectivament.

En el nostre cas, s'ha decidit que la mida de la graella sigui quadrada perquè posteriorment en els experiments, és més fàcil controlar-los i modificar condicions o paràmetres per estudiar el seu impacte en la propagació del foc. Per tant, una quadrícula regular ens permetrà fer rèpliques dels experiments d'una manera consistent.

En aquest punt, es pregunta a l'usuari si vol que hi hagi vent en la simulació o no (y/n). En cas que la resposta sigui afirmativa, es demana que especifiqui la direcció del vent (nord, sud, est, oest). També es demana el nombre de focs inicials que hi haurà a la simulació.

En qualsevol cas, si l'entrada de l'usuari no és vàlida, es mostra un missatge d'error i es torna a sol·licitar la resposta fins que sigui correcta. Cada resposta obtinguda s'emmagatzema en la variable corresponent. Si l'usuari decideix que no vol vent, la variable `direccio_vent` s'estableix a None.

Aleshores d'inicien els focs. S'itera pel nombre de focs proporcionat per l'usuari i per a cada un, escull una posició aleatòria (fila i columna) de la graella de manera que mai es tracti d'un llac, i s'inicialitza el foc establint l'estat d'incendi d'aquella casella en 1.

A continuació, el codi inicia un bucle infinit que es manté actiu mentre la finestra de gràfics segueixi oberta. Aquest bucle permet que el programa es pugui aturar en qualsevol moment simplement tancant la finestra.

A cada iteració del bucle, es crida la funció `visualitzar_capes` per actualitzar la

visualització de les capes d'humitat, vegetació i estat de l'incendi. Després, es crida la funció `actualitzar_incendi` per simular el següent pas de la propagació de l'incendi. El contador pas s'incrementa per indicar el nombre d'iteracions.

El bucle també comprova si l'incendi s'ha extingit completament mitjançant la variable `tot_apagat`. Si aquesta variable és True, s'imprimeix un missatge que indica que l'incendi s'ha extingit completament, i el bucle es trenca, posant fi al programa.

2.2.2 funcions_auxiliars

Aquest fitxer conté les diverses funcions necessàries per al funcionament del programa que seran cridades des del fitxer principal `main` per a poder simular l'incendi. D'aquesta manera, el fitxer principal pot centrar-se en el flux de la simulació, mentre que les funcions definides aquí realitzen tasques específiques que contribueixen a la simulació general de l'incendi.

2.2.2.1 genera_fitxer_doc

La funció genera_fitxer_doc serveix per crear el fitxer de documentació (.doc) que descriu les característiques de la capa corresponent, típicament utilitzat en contextos on els fitxers de dades necessiten acompanyar-se d'una explicació sobre el seu format i contingut. Aquesta funció es pot usar per documentar fitxers de dades en simulacions o en sistemes de processament de dades geogràfiques. Podem trobar, el nom de la capa, tipus de dades, tipus de fitxer, dimensions (columnes i files), sistema i unitats de referència, distància unitària, coordenades mínimes i màximes (X i Y), errors de posició, resolució, valors mínims i màxims, unitats dels valors, errors dels valors, i informació sobre flags o categories de llegenda.

2.2.2.2 genera_fitxer_img

La funció genera_fitxer_img està dissenyada per crear un fitxer d'imatge (.img) a partir d'una matriu de dades. Converteix la matriu bidimensional 'dades' (que seran els np arrays de `humitat` i `vegetació` en una matriu unidimensional (d'una columna) amb el mètode `flatten`.

2.2.2.3 llegir_dades_doc

La funció llegir_dades_doc obra i llegeix un fitxer de documentació (.doc) que conté metadades sobre una capa. Aquesta funció extreu les dimensions de la graella de la capa corresponent dimensions utilitzant expressions regulars per buscar les cadenes de text que comencen amb 'columns' i 'rows', convertint els números en format strings a enters i retorna aquestes dimensions.

2.2.2.4 llegir_dades_img

La funció llegir_fitxer_img s'encarrega de llegir un fitxer d'imatge (.img), assumint que conté valors numèrics representats en format columna. Aquesta funció utilitza el nombre de columnes i files proporcionats com a paràmetres per reestructurar la llista de valors llegits del fitxer en una matriu amb les dimensions especificades.

2.2.2.5 llegir_dades

La funció llegir_dades realitza la lectura completa dels conjunts de dades per `humitat` i `vegetació`. Aquesta funció primer crida a llegir_dades_doc per obtenir les dimensions de cada graella a partir dels fitxers de documentació associats als fitxers d'imatge d'`humitat` i `vegetació`.

Amb aquestes dimensions, es procedeix a llegir els fitxers .img utilitzant llegir_fitxer_img, retornant les matrius de dades per `humitat` i `vegetació` que seràn utilitzades posteriorment.

2.2.2.6 crear_llacs

Aquesta funció s'encarrega, tal i com el seu nom indica, de generar els llacs aleatòriament a la graella, i d'aquesta manera, crear la capa d'humitat. Per tant, rep com a paràmetres el nombre de llacs que es volen crear i les dimensions de la graella, i retorna la capa d'humitat, és a dir, la matriu NumPy amb nombres del 0 al 5, indicant les hores que triga en evaporar-se la humitat en cada casella, abans de començar a cremar-se.

Primerament es verifica si es volen crear llacs. En cas que no es desitgi crear-ne cap (`num_llacs == 0`), es genera una matriu de la mida de la graella proporcionada amb nombres aleatoris entre 0 i 5 i es retorna com a resultat.

En cas contrari, es crea una matriu similar, però amb nombres entre 0 i 3. Aleshores, es repeteix el mateix procediment tants cops com nombre de llacs especificat, per a crear els llacs i ajustar la humitat de les caselles properes. L'objectiu és que les caselles més properes al llac tinguin una humitat més elevada, mentre que la humitat disminueix gradualment a mesura que s'allunyen del llac, tot mantenint una certa aleatorietat.

Es comença inicialitzant el llac en una casella aleatòria de la graella. Per garantir que aquestes caselles no es puguin cremar mai, se'ls assigna un valor d'humitat infinit. A continuació, s'estableix la mida del llac de manera aleatòria dins d'un rang, per a evitar que siguin massa petits 0 massa grans.

Després, s'inicialitzen dues noves variables: `caselles_llac` i `num_caselles_llac`. La primera és una llista emmagatzema les caselles que pertanyen al llac que s'està creant, on cada element és una tupla que conté les coordenades (fila i columna) de la casella del llac. La segona és un comptador que registra el nombre de caselles que pertanyen al llac.

Aleshores comença un bucle que acaba quan el llac arriba a la mida especificada anteriorment. Per a crear el llac de forma aleatòria, es selecciona una casella que ja formi part del llac (de la llista `caselles_llac`) i s'afegeixen totes les caselles adjacents com a part del llac comprovant que no en formin part ja i que es trobin dins la graella.

Un cop els llacs han sigut creats, es procedeix a afegir humitat a les zones més properes. Per aconseguir-ho, s'itera per cada casella de la graella que no tingui assignada humitat infinita, és a dir, que no formi part d'un llac. Per a cada una d'elles, es calcula la distància fins a la casella més propera que forma part d'un llac. Aleshores se li assigna una probabilitat en funció d'aquesta distància, la qual serà més alta com més propera sigui a un llac.

En funció d'aquesta probabilitat, s'assigna o no, de manera aleatòria un valor d'humitat més alt del que ja tingués, ja que en aquest punt s'assignen valors d'entre 4 i 5. Aquest procediment permet establir un gradient d'humitat des dels llacs cap a les zones més llunyanes, amb una major probabilitat d'increment d'humitat en les àrees més properes als llacs.

2.2.2.7 inicialitzar_vegetacio

Aquesta funció s'encarrega de crear la capa d'inicialització de la vegetació, és a dir, genera la matriu que contindrà els valors de vegetació per a cada casella.

La funció comença generant una matriu de probabilitats aleatòries de mida `graella_size` x `graella_size`, on cada element és un valor entre 0 i 1.

A continuació, s'aplica un filtre Gaussià a la matriu amb un paràmetre `sigma` = 1.5, que controla el grau de suavització. D'aquesta manera s'eliminen els valors extrems i es creen unes transicions més suaus entre les diferents àrees.

Després de suavitzar les probabilitats, la funció normalitza els valors per garantir que oscil·lin entre 0 i 1.

Aleshores, es fa un altre escalat per obtenir valors entre 1 i 10, que representen la quantitat de vegetació en cada casella. Això permet simular les hores que cada casella tarda a cremar-se segons la seva densitat de vegetació.

Finalment, la funció retorna la matriu resultant, que representa la capa d'inicialització de la vegetació.

2.2.2.8 actualitzar_incendi

Aquesta funció s'encarrega de simular la propagació de l'incendi a través de la graella, modificant la capa d'humitat, la de vegetació i la d'estat d'incendi, que són les tres matrius que retorna la funció, juntament amb una variable booleana que indica l'estat global del foc, és a dir, si està apagat o no.

Com a paràmetres, la funció rep la matriu NumPy corresponent a cada una de les tres capes: humitat, vegetació i estat_incendi, així com la mida de la graella i la direcció del vent, en cas que n'hi hagi.

Primer de tot, es crea una còpia de la matriu que simula l'estat de l'incendi i s'inicialitza la variable booleana `tot_apagat` a `True`.

Aleshores comença un bucle que recorre tota la graella, casella per casella, utilitzant el valor de la variable proporcionada `graella_size`. Dins el bucle, es comprova si la casella ja s'està cremant (`estat_incendi[i, j] == 1`) o si pot començar a cremar-se (`estat_incendi[i, j] == 0`).

En el primer cas, si la casella s'està cremant, es modifica el valor de la variable booleana a Fale, indicant que encara hi ha caselles cremant-se a la graella. A continuació es verifica si la vegetació és superior 0 (`vegetacio[i, j] > 0`). En cas que es cumpleixi, es redueix en 1 el valor de vegetació per a la casella en qüestió. En cas contrari, es marca com a completament cremada assignant-li el valor 2 a l'estat d'incendi.

Posteriorment, es redueix la humitat en 1 a les caselles adjacents (contant les diagonals) a l'actual, si en tenen (`humitat[i + di, j + dj] > 0`). En aquest punt comença la influència del vent en l'incendi. Es verifica que `direccio_vent is not None`, és a dir, que l'usuari hagi indicat que si que vol que hi hagi vent i la direcció. En cas afirmatiu, es calcula les noves coordenades (fila i columna) en la direcció proporcionada. Si la nova casella no té humitat, s'inicia un altre foc en aquesta posició. Així, el vent pot accelerar la propagació de l'incendi cap a certes direccions en la graella.

En el segon cas, quan es tracta d'una casella que encara no ha començat a cremar ni està cremada completament, aleshores es comprova si alguna de les caselles veïnes s'està cremant i si la casella actual té humitat 0. Si es compleixen aquestes dues condicions, s'inicia el foc en la casella.

2.2.2.9 visualitzar_capes

Aquesta funció crea una visualització de tres capes: la humitat, la vegetació i una combinació de les dues anteriors, juntament amb l'estat de l'incendi.

Primer de tot, es creen tres subgràfiques, formant una matriu d'una fila i tres columnes de subgràfiques.

En la primera subgràfica es mostra la humitat. Per a representar-la, s'han sobreposat dues capes. Primer de tot, una que pinta de blau les caselles d'humitat infinita (llacs), i a sobre, s'ha afegit la matriu d'humitat amb una paleta de blaus, on els valors més foscos indiquen una humitat més elevada.

En la segona subgràfica es mostra la vegetació, utilitzant una paleta de colors verds per representar els diferents nivells de vegetació.

En la tercera subgràfica, es mostra l'estat de l'incendi. S'utlitza una paleta de colors vermells per a representar els tres estats de l'incendi (per cremar, cremant-se i completament cremat).

En la tercera subgràfica, es presenta la combinació dels dos estats anteriors, juntament amb la propagació del foc. També consisteix en dues capes sobreposades, com en la segona subgràfica. La primera capa mostra tota la graella verda, ressaltant els llacs amb color blau. La segona conté la propagació de l'incendi, col·locant-la per sobre, tot emmascarant les caselles que no s'estàn cremant per a que es pugui veure la primera capa.

Finalment, s'utilitza `plt.pause(0.0001)` per a fer una pausa breu entre iteracions. Això permet visualitzar els gràfics en cada hora de la simulació, facilitant la observació de la propagació de l'incendi.

2.3 Diagrama de Flux

La següent imatge mostra el diagrama de flux de l'incendi, representant el comportament d'una casella de la graella, aplicable per a cada una d'elles. D'aquesta manera, es pot comprendre com es propaga l'incendi en una casella en concret i per conseqüent, com es propaga per al tot el conjunt de caselles que formen la graella. S'ha realitzat el diagrama de flux que representa el codi explicat anteriorment juntament amb les extensions de la creació dels llacs i l'aplicació del vent en la propagació de l'incendi.

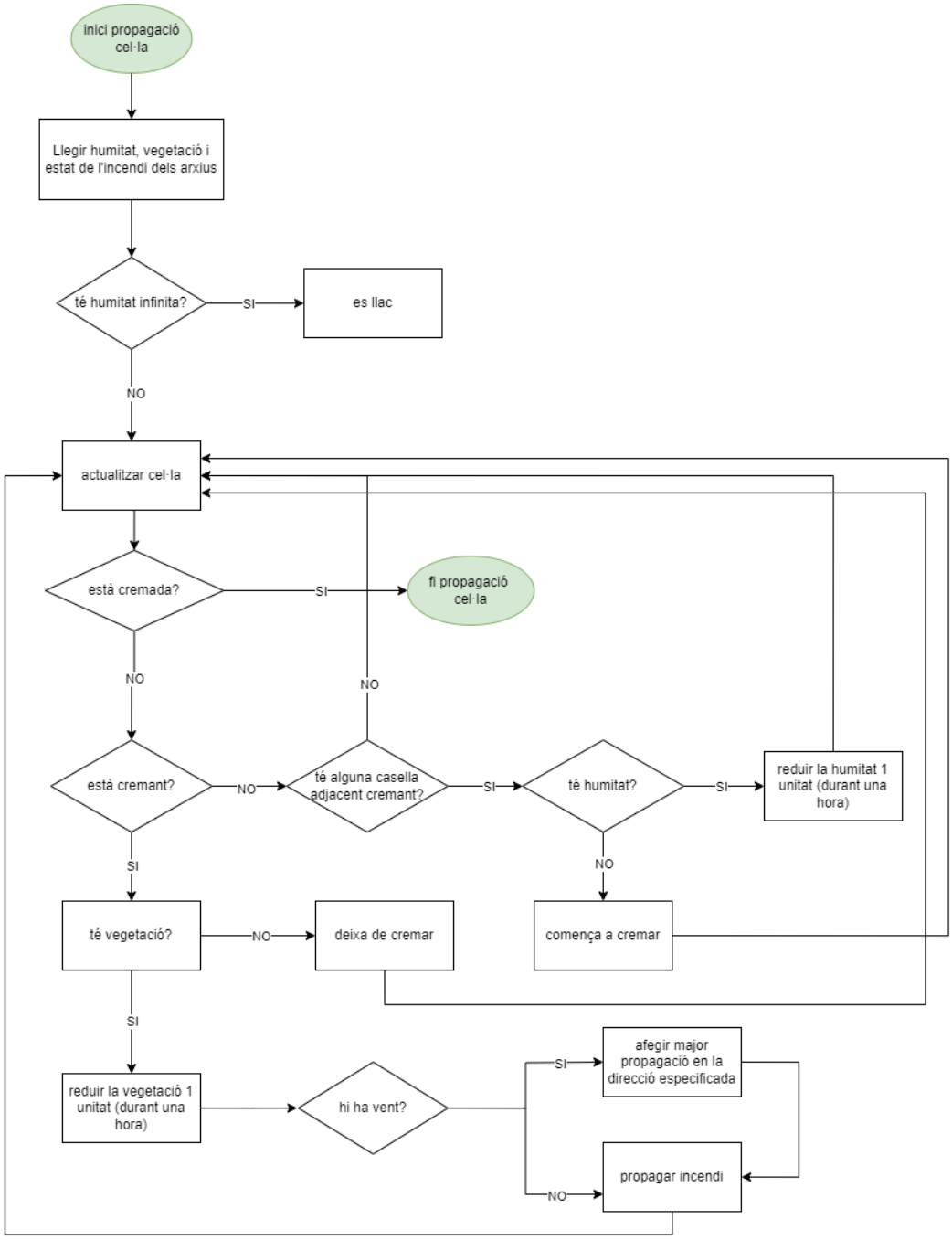


Figure 1: Diagrama de Flux

Aquest diagrama representa la propagació d'una cel·la de la graella de la representació de l'incendi.

Primer de tot, llegim els valors dels fitxers .img de cada capa per saber el nivell d'humitat i de vegetació i l'estat de l'incendi en la casella corresponent.

Un cop hem llegit els valors, comprovem si la cel·la actual té humitat infinita, ja que si en té, indica que aquesta cel·la forma part d'un llac i, per tant, no es podrà cremar.

Seguidament, si la cel·la no forma part d'un llac, actualitzem els valors de la cel·la amb les dades llegides anteriorment (nivell d'humitat i vegetació i l'estat de l'incendi).

Comprovem si la cel·la ja està cremada. Si ho està, vol dir que aquesta cel·la ja no propagarà més el foc ja que ja no en té, per tant, fi de la propagació. Si no està cremada, fem una nova comprovació: la cel·la està cremant?

Si la resposta és afirmativa, ens indica que la humitat és 0 (ja que es un requisit en el nostre codi perquè una cel·la pugui començar a cremar), per tant, comprovem si la cel·la té vegetació. Si encara en té, reduim una unitat el nivell de vegetació (la qual cosa trigarà una hora en fer-se). Seguidament, comprovem si hi ha vent en la simulació de l'incendi, si hi ha, cadrà afegir una major propagació en les cel·les en direcció al vent (donat per l'usuari), si no hi ha vent, es propagarà l'incendi únicament a les cel·les adjacents (contant les diagonals) a la cel·la en la que ens trobem. Si no té vegetació, indica que la cel·la ja ha cremat tota la vegetació de la cel·la i, per tant, no queda res per cremar (la cel·la ha deixat de cremar). Tant pel cas afirmatiu com pel negatiu, actualitzarem els valors la cel·la ja que aquests han canviat (nivell de vegetació o estat de l'incendi).

En cas que la cel·la no estigui cremant, comprovem que alguna casella adjacent a aquesta estigui cremant ja que si alguna està cremant, és necessària la propagació de l'incendi en la casella actual. Si la cel·la actual té humitat, disminuim el nivell d'humitat 1 unitat durant una hora i actualitzem la cel·la. Si la cel·la no té humitat, cal canviar, l'estat de l'incendi a cremant, per tant, tornem a actualitzar la cel·la.

2.4 Experiments

En aquesta secció, es realitzaran diversos experiments per avaluar la influència de diversos factor en el comportament del foc, és a dir, en relació a les hores que triga a cremar-se per complet la graella.

Per mantenir la consistència i assegurar la validesa de les conclusions, algunes paràmetres determinats es mantindran constants en tots els experiments, excepte aquells que es modifiquin específicament per estudiar el seu efecte.

Els paràmetres següents es mantindran constants per a tots els experiments, a menys que es faci una excepció:

- `graella_size` = 70
- `numero_llacs` = 0
- `direccio_vent` = None
- `numero_focs` = 1
- Rang de valors de la humitat: 0-5
- Rang de valors de la vegetació: 1-10

Per assegurar la reproductibilitat i reduir la variabilitat associada a l'aleatorietat, cada experiment es repeteix cinc vegades. Utilitzem un conjunt específic de llavors per a la generació de nombres aleatoris. Això garanteix que les simulacions es poden reproduir exactament per validar resultats o realitzar més experiments.

Les llavors utilitzades per a cada iteració són:

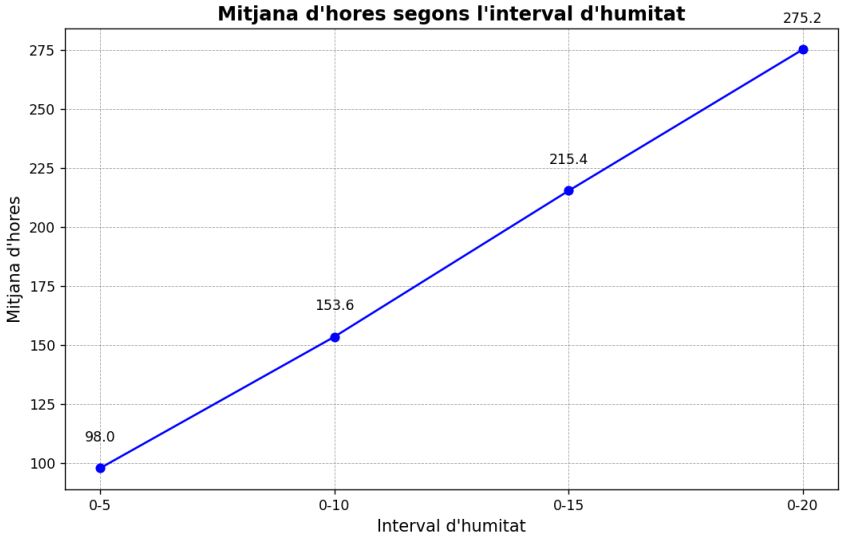
- Iteració 1: 123456789
- Iteració 2: 987654321
- Iteració 3: 20242025
- Iteració 4: 54321
- Iteració 5: 111222333

Com que els experiments depenen fortament de la posició inicial del foc i altres factors aleatoris, és crucial repetir cada experiment diverses vegades per obtenir una visió més clara del comportament general. Les llavors específiques permeten que cada iteració es reprodueixi exactament, fent possible la comparació i l'anàlisi de resultats.

EXPERIMENT 1. Què passa quan augmentem l'interval de la humitat?

Interval d'humitat	**0-5**	**0-10**	**0-15**	**0-20**
Iteració 1	72	155	240	266
Iteració 2	93	134	225	295
Iteració 3	90	179	165	268
Iteració 4	128	189	209	276
Iteració 5	107	111	238	271

Table 1: Taula d'hores segons l'interval d'humitat



En aquest experiment, s'ha investigat l'efecte de l'augment de l'interval de valors que pren la humitat, que indica el nombre d'hores que una casella triga a quedar sense humitat.

Com era d'esperar, a mesura que l'interval de valors de la humitat s'incrementa, el temps total que tarda la graella a cremar-se completament també augmenta. Aquesta relació sembla ser lineal, en altres paraules, com més humitat té la graella, més temps triga el foc a cremar-la completament.

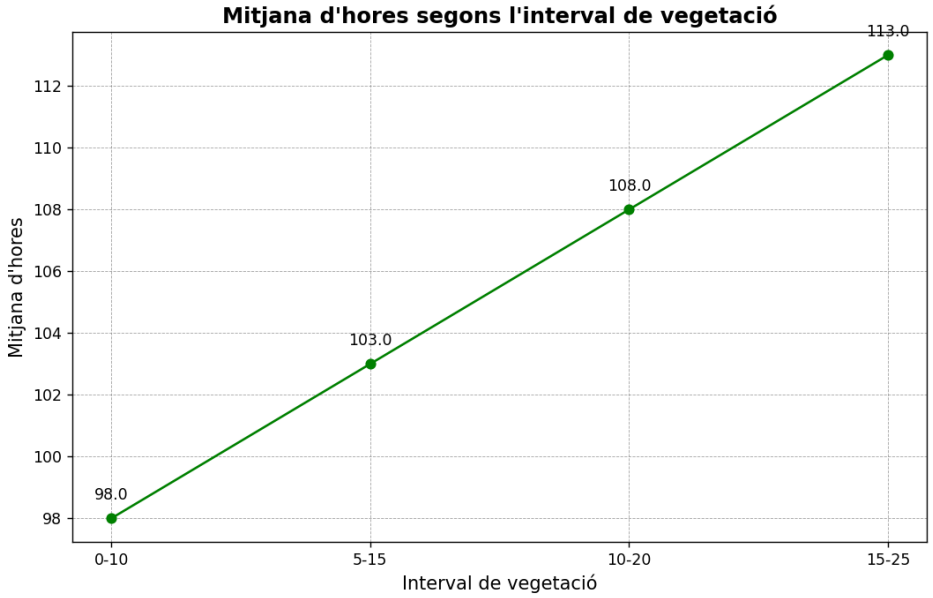
Un resultat inesperat ha sigut que, amb els intervals d'humitat més grans, algunes caselles de la graella no arriben a cremar-se mai. Això es deu al fet que el codi simula que una casella només redueix la seva humitat si té alguna casella adjacent cremant i només pot començar a cremar-se si no té humitat i té una casella adjacent cremant. En casos on la humitat és molt alta i no hi ha suficient vegetació, la humitat pot ser tan elevada que, quan finalment s'asseca (si es que arriba a assecar-se), ja no hi ha caselles adjacents (contant les diagonals) amb foc. Això pot crear zones que no arriben a cremar-se.

Aquest resultat indica que el model utilitzat per simular el foc té aspectes realistes, ja que en el món real també es podrien donar situacions similars. En àrees amb humitat elevada, el foc podria no propagar-se completament, deixant zones sense cremar.

EXPERIMENT 2. Què passa quan augmentem l'interval de la vegetació?

Interval de vegetació	0-10	5-15	10-20	15-25
Iteració 1	72	77	82	87
Iteració 2	93	98	103	108
Iteració 3	90	95	100	105
Iteració 4	128	133	138	143
Iteració 5	107	112	117	122

Table 2: Taula d'hores segons l'interval de vegetació



En aquest cas, s'ha volgut analitzar la influència de la vegetació en la propagació del foc. La vegetació representa el temps que triga una casella a cremar-se completament un cop s'ha encès.

Per explorar aquest efecte, s'ha anat modificant l'interval de valors de vegetació en increments de 5.

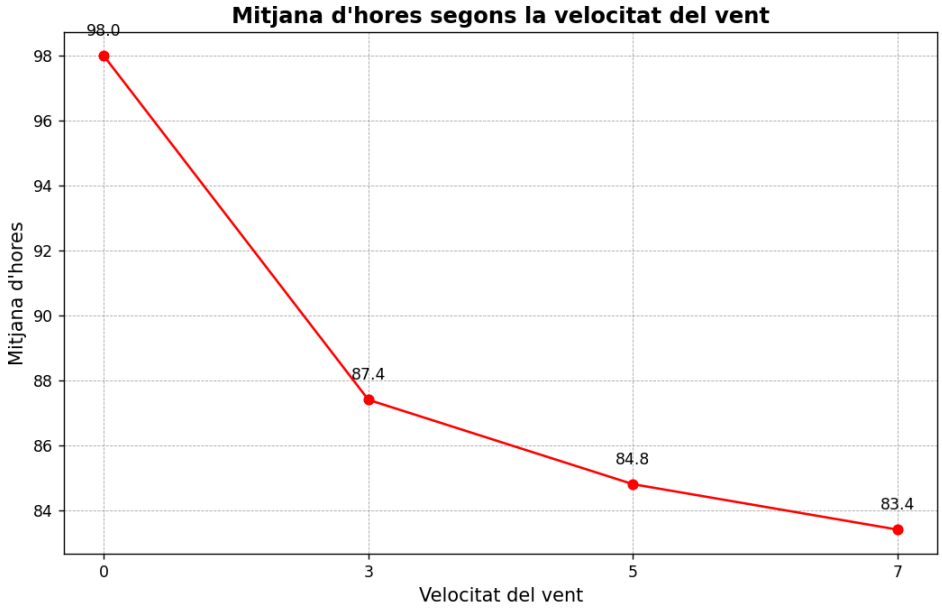
S'ha pogut observar que cada increment de 5 unitats en la vegetació resulta en un augment exactament de 5 hores en el temps total de crema de la graella. Aquest fet indica que l'augment de la vegetació afecta de manera lineal el temps que triga una casella a cremar-se completament. A mesura que augmenta la vegetació, el foc es propaga més lentament, requerint més temps per consumir totes les caselles.

La relació lineal observada en aquest experiment suggereix que la vegetació és un factor predictible en el comportament del foc. Això implica que, controlant la densitat i la distribució de la vegetació, es pot estimar amb el temps necessari perquè el foc es propagui en un entorn donat.

EXPERIMENT 3. Què passa quan afegim vent amb diferents velocitats?

Velocitat del vent	0	3	5	7
Iteració 1	72	69	66	64
Iteració 2	93	88	86	85
Iteració 3	90	90	90	90
Iteració 4	128	109	107	106
Iteració 5	107	81	75	72

Table 3: Taula d'hores segons la velocitat del vent



Aquest experiment ha servit per a observar el comportament del foc a l'afegir-hi un factor extern: el vent. S'ha executat la simulació per a vent amb diferents velocitats i s'ha assignat `direccio_vent` = 'est', per a fer totes les simulacions amb el mateix valor.

S'ha observat que el vent si que contribueix a l'acceleració de la propagació de l'incendi. A mesura que la velocitat del vent augmenta, el temps total per cremar la graella disminueix.

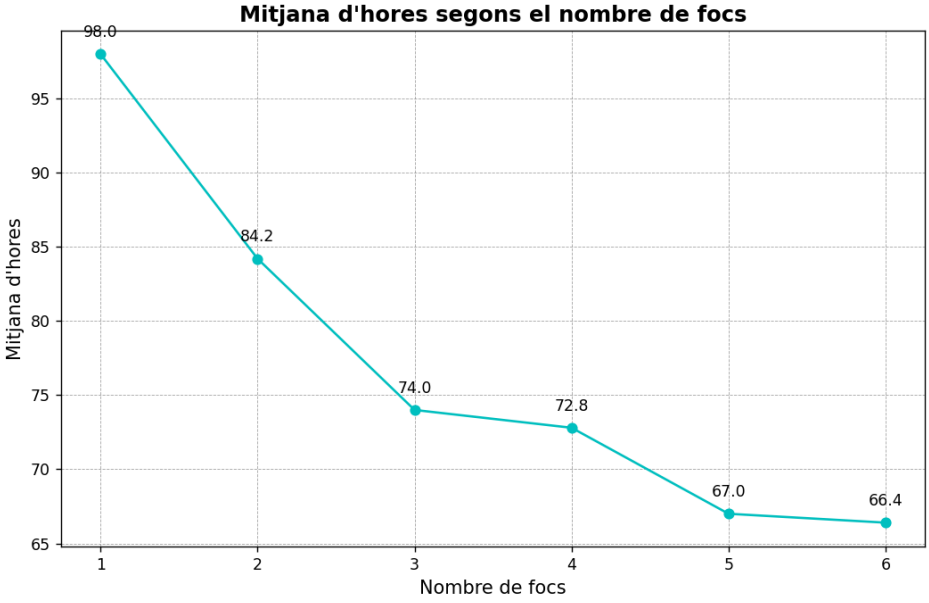
Tot i això, s'ha observat que, en particular a la tercera iteració, el temps no varia amb l'augment de la velocitat del vent. Això es pot explicar pel fet que, amb aquesta llavor, el foc s'inicialitza en una posició bastant a la dreta de la graella. L'efecte del vent fa que el foc s'expandeixi ràpidament cap al límit dret, però després encara ha de propagar-se cap a l'esquerra, i el vent no té impacte en aquesta direcció.

Aquesta observació també explica per què la disminució del temps de propagació es fa més lenta a mesura que la velocitat del vent augmenta: el foc arriba més ràpidament al límit dret, però ha de recórrer tota la graella per cremar-se completament.

EXPERIMENT 4. Què passa quan augmentem el nombre focs?

Nombre de focs	1	2	3	4	5	6
Iteració 1	72	71	63	63	63	63
Iteració 2	93	84	79	73	73	73
Iteració 3	90	87	85	85	67	64
Iteració 4	128	109	82	82	76	76
Iteració 5	107	70	61	61	56	56

Table 4: Taula d'hores segons el nombre de focs



Aquest experiment intenta explorar l'impacte del nombre de focs inicials en la rapidesa de propagació de l'incendi. La simulació s'ha executat amb diferents escenaris, augmentant el nombre de focs fins a 6.

Els resultats han mostrat el que s'esperava: a mesura que s'augmenta el nombre de focs inicials, el temps total es redueix considerablement, ja que més focs impliquen una major àrea cremant simultàniament, accelerant la propagació de l'incendi per tota la graella.

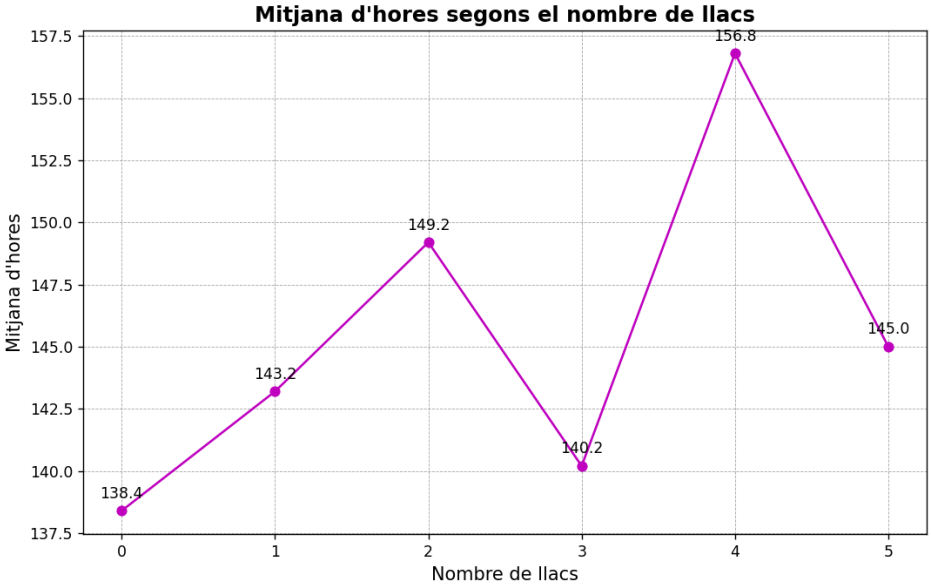
Tanmateix, també s'ha observat que l'efecte de l'augment de focs no és sempre lineal. A mesura que s'incrementa el nombre de focs, la disminució del temps de propagació és menys dràstica. Aquesta diferència pot ser deguda a l'aleatorietat: si els focs inicials es troben propers entre si, actuen com un únic foc més gran. En canvi, si estan més distribuïts inicialment, la propagació és molt més ràpida.

Per tant, podem concloure que, tot i que l'augment del nombre de focs inicials tendeix a accelerar la propagació del foc, aquesta depèn en gran part del lloc on s'inicien els focs.

EXPERIMENT 5. Què passa quan afegim llacs?

Nombre de llacs	0	1	2	3	4	5
Iteració 1	138	133	120	107	198	125
Iteració 2	156	160	129	186	141	127
Iteració 3	138	134	164	142	122	156
Iteració 4	129	147	175	148	129	136
Iteració 5	131	142	158	118	194	181

Table 5: Taula d'hores segons el nombre de focs



Aquest experiment busca trobar la relació entre el nombre de llacs en un terreny i la rapidesa de propagació de l'incendi.

En aquest cas, s'ha fet ús d'una graella de mida 100, per a poder experimentar amb més nombre de llacs, ja que a la graella de 70 no hi caben tants llacs. S'ha fet la simulació amb escenaris fins a 5 llacs.

Tot i que es podria esperar que un nombre més gran de llacs ralentitzi la propagació del foc perquè hi ha menys àrea vegetal, la situació no és ben bé així. Les zones properes als llacs tenen una humitat més alta, per lo que triguen més a començar a cremar-se.

La gràfica resultant no mostra una tendència clara; els valors semblen fluctuar de manera aleatòria. Això pot ser degut a que els resultats depenen en gran part de la ubicació en la graella tant dels llacs, com de l'inici del foc.

A causa d'aquesta variabilitat, és difícil extreure conclusions definitives sobre la relació entre el nombre de llacs i la rapidesa del foc. No obstant això, en general, els resultats suggereixen que la presència de llacs pot retardar la propagació de l'incendi.

3 Conclusions

3.1 Part 1

La primera part de la pràctica ha consistit en la implementació i simulació d'un autòmat cel·lular utilitzant les regles de Wolfram, que serveixen per descriure els comportaments dinàmics en una graella unidimensional. A través de l'ús del llenguatge de programació Python i la biblioteca Pygame, hem pogut visualitzar la progressió temporal dels estats de l'autòmat.

Amb aquesta part de la pràctica hem pogut reforçar la comprensió dels principis bàsics dels autòmats cel·lulars i la utilitat de la programació en la modelització de sistemes complexos.

3.2 Part 2

En aquesta segona part de la pràctica, s'ha realitzat la modelització de la propagació d'un incendi forestal utilitzant un autòmat cel·lular, representat per tres capes diferents: humitat, vegetació, i estat de l'incendi. A través de la implementació d'un model computacional en Python, s'ha simulat la interacció entre aquests components per predir i visualitzar el comportament d'un incendi en un entorn forestal controlat.

La pràctica ha demostrat com la variació en els nivells d'humitat i la distribució de la vegetació afecten la velocitat i direcció de la propagació del foc. A més, la capacitat de manipular variables ambientals com la presència de llacs o la direcció del vent ha permès visualitzar diferents escenaris d'incendis forestals.

La capacitat de preveure i comprendre la dinàmica dels incendis forestals és crucial per al desenvolupament d'estratègies més efectives de prevenció i resposta a emergències, destacant la rellevància d'aquestes tècniques en l'enginyeria ambiental i la gestió del risc.

3.3 Utilització de la IA

Durant la realització de la pràctica, hem utilitzat l'eina de Chat GPT versió 4. Aquesta tecnologia ens ha facilitat la creació del codi. Mitjançant una descripció detallada del que volíem fer, Chat GPT ens ha generat suggeriments de codi, consells sobre l'estructura del programa i ens ha ajudat a resoldre dubtes específics que ens han sorgit durant la implementació. Aquesta eina ens ha permès realitzar la tasca en un temps reduït.