Optimització

Pràctica 3
Simulació amb autòmats cel·lulars

Grau en Intel·ligència Artificial

Curs 2024/2025

Laida Queral Llopart
Jaume Mora i Ladària

Índex

1 Introducció

En aquesta pràctica hem treballat amb autòmats cel·lulars com a eina per simular fenòmens temporals i espacials. El treball s'ha dividit en dues sessions: la primera, que ens ha servit per utilitzar les regles de Wolfram per a autòmats cel·lulars unidimensionals; i la segona centrada en un cas d'aplicació real: la propagació d'un incendi a partir de dades ambientals.

El codi s'ha desenvolupat en Python, utilitzant estructures de dades bàsiques i biblioteques gràfiques. S'han generat tant imatges com GIFs per visualitzar l'evolució dels sistemes.

2 Primera sessió: Autòmats cel·lulars de Wolfram

2.1 Objectiu

El primer objectiu és implementar un autòmat cel·lular unidimensional (1D) que evolucioni segons una de les 256 regles elementals de Wolfram, i representar la seva evolució gràficament. En el segon pas, s'ha generalitzat el sistema per a poder combinar dues regles en una sola simulació.

2.2 Funcions bàsiques

2.2.1 Conversió de regla a binari

Per poder aplicar una regla de Wolfram, cal primer convertir el número de regla a la seva representació binària de 8 bits. Això ens permet associar cada patró de 3 bits (una cel·la i els seus veïns) a un nou valor:

```
def get_rule_binary(rule_number):
    return np.array([int(b) for b in np.binary_repr(rule_number,
       width=8)])
```

Per exemple, la regla 30 es representa com `00011110`, la qual cosa implica un conjunt determinat d'actualitzacions per a cada combinació de 3 bits.

2.2.2 Evolució d'un pas temporal

Aquesta funció aplica la regla binària sobre una fila d'estat actual per obtenir el nou estat de la següent línia temporal:

```
def evolve_1d(state, rule_bin):
    new_state = np.zeros_like(state)
    for i in range(1, len(state) - 1):  # Evitem les vores
        neighborhood = state[i-1:i+2]
        index = 7 - int(''.join(map(str, neighborhood)), 2)
        new_state[i] = rule_bin[index]
    return new_state
```

S'agafa la cel·la actual i els seus dos veïns, es construeix el patró binari, es calcula l'índex corresponent i s'obté el nou valor a partir de la regla.

2.2.3 Simulació completa

La funció següent rep un número de regla, i simula l'evolució temporal completa de l'autòmat, partint d'una única cel·la activa al centre:

```
def simulate_1d_automaton(rule_number, size=101, steps=100):
    rule_bin = get_rule_binary(rule_number)
    states = np.zeros((steps, size), dtype=int)
    states[0, size // 2] = 1  # Inicialitzacio

    for t in range(1, steps):
        states[t] = evolve_1d(states[t-1], rule_bin)
```

```
8
9       return states
```

El resultat és una matriu 2D de dimensions `(steps, size)` on cada fila representa l'estat de totes les cel·les en un moment concret.

2.3 Visualització

L'autòmat s'ha visualitzat mitjançant imatges en escala de grisos, on les cel·les actives es mostren en negre:

```
1 def plot_1d_automaton(states, rule_number):
2     plt.imshow(states, cmap="binary", interpolation="nearest")
3     plt.title(f"Automat 1D - Regla {rule_number}")
4     plt.show()
```

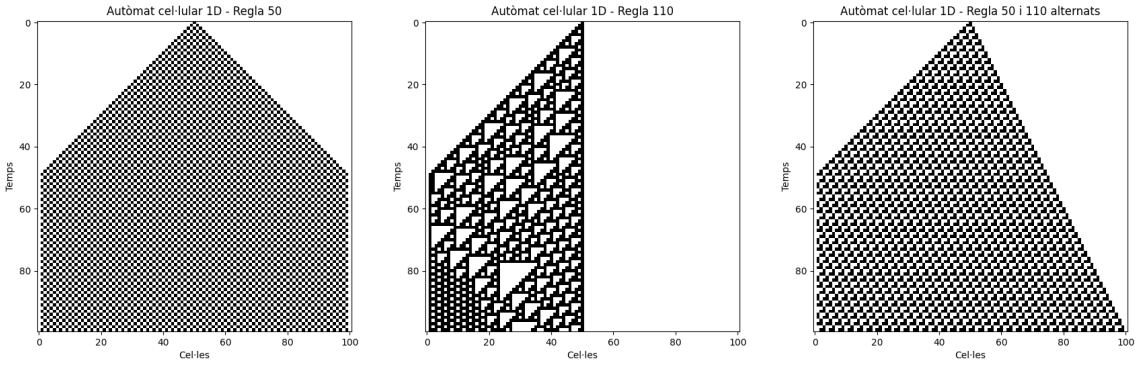
2.4 Simulació amb regles combinades

Per simular sistemes multicapa, en aquest cas s'ha fet amb dues capes i s'ha implementat una variant on s'alternen dues regles de manera intercalada:

```
1  def simulate_multirule_1d(rule1, rule2, size=101, steps=100):
2      rule_bin1 = get_rule_binary(rule1)
3      rule_bin2 = get_rule_binary(rule2)
4
5      states = np.zeros((steps, size), dtype=int)
6      states[0, size // 2] = 1
7
8      for t in range(1, steps):
9          if t % 2 == 0:
10             states[t] = evolve_1d(states[t-1], rule_bin1)
11         else:
12             states[t] = evolve_1d(states[t-1], rule_bin2)
13
14     return states
```

Això genera comportaments més complexos que poden emular sistemes híbrids.

2.5 Resultats



(a) Regla 50 (b) Regla 110 (c) Regles 50 i 110 alternades

Figura 1: Autòmat cel·lular per diferents regles de Wolfram

Les imatges mostren com diferents regles de Wolfram poden generar patrons molt diversos. La Regla 50 dona lloc a una estructura totalment regular i previsible; en canvi, la Regla 110 genera patrons complexos amb comportaments que semblen caòtics, tot i tenir regles simples. Finalment, la combinació alternada de les Regles 50 i 110 genera una dinàmica híbrida que alterna simetries amb irregularitats, cosa que reflecteix com petites variacions en les regles poden donar lloc a resultats significativament diferents.

3 Segona sessió: Simulació d'incendi forestal

3.1 Objectiu

Construir un sistema d'autòmat cel·lular bidimensional capaç de simular la propagació d'un incendi forestal, a partir de dades ambientals reals: capa de vegetació (hores que crema) i capa d'humitat (retard per encendre's). Posteriorment, s'afegeix una tercera capa de vent per simular direccions preferents de propagació, que era opcional.

3.2 Lectura de dades

Les dades s'han carregat a partir del format binari IDRISI32, amb una funció auxiliar que converteix els fitxers '.img' a matrius de NumPy:

```
def read_img(path, shape):
    with open(path, 'rb') as f:
        data = np.fromfile(f, dtype=np.int16)
    return data.reshape(shape)
```

3.3 Estats i capes

Cada cel·la pot estar en estat:

- 0 = pendent (encarq no s'ha cremat).
- 1 = en procés de cremar-se.
- 2 = ja cremada.

A més, es mantenen dues matrius auxiliars:

- **humidity_timer**: nombre d'hores que s'ha exposat a foc.
- **vegetation_timer**: nombre d'hores que porta cremant-se.

3.4 Algorisme de propagació (sense vent)

La funció principal responsable de l'evolució de l'autòmat itera sobre totes les cel·les i va actualitzant-ne l'estat segons l'informació de les dues capes (la vegetació i la humitat). El comportament es diferencia en funció de si la cel·la ja està cremant o encara està pendent de cremar-se:

- **Cel·les cremant-se** (`state[i, j] == 1`):
  Aquestes cel·les han estat activades en un pas anterior i es troben actualment cremant-se. En cada instant de temps:
  - S'incrementa el temporitzador de vegetació: `vegetation_timer[i, j] += 1`.
  - Quan aquest temporitzador és igual al valor de `vegetation[i, j]`, es considera que la cel·la ha acabat de cremar i ja es considera zona cremada.

  Això modela el temps que triga una cel·la abans de ser cremada i mentre s'està cremant.

- **Cel·les pendents de ser cremades** (`state[i, j] == 0`):
  Aquestes cel·les encara no s'han cremat però podran encendre's si tenen almenys un veí en cremant-se. En aquest cas:
  - Es recorren les vuit cel·les veïnes.
  - Si es detecta almenys un veí amb `state == 1`, s'incrementa el temporitzador d'humitat de la cel·la actual: `humidity_timer[i, j] += 1`.
  - Quan el valor del temporitzador supera o iguala `humidity[i, j]`, la cel·la comença a cremar: `state[i, j] = 1`.

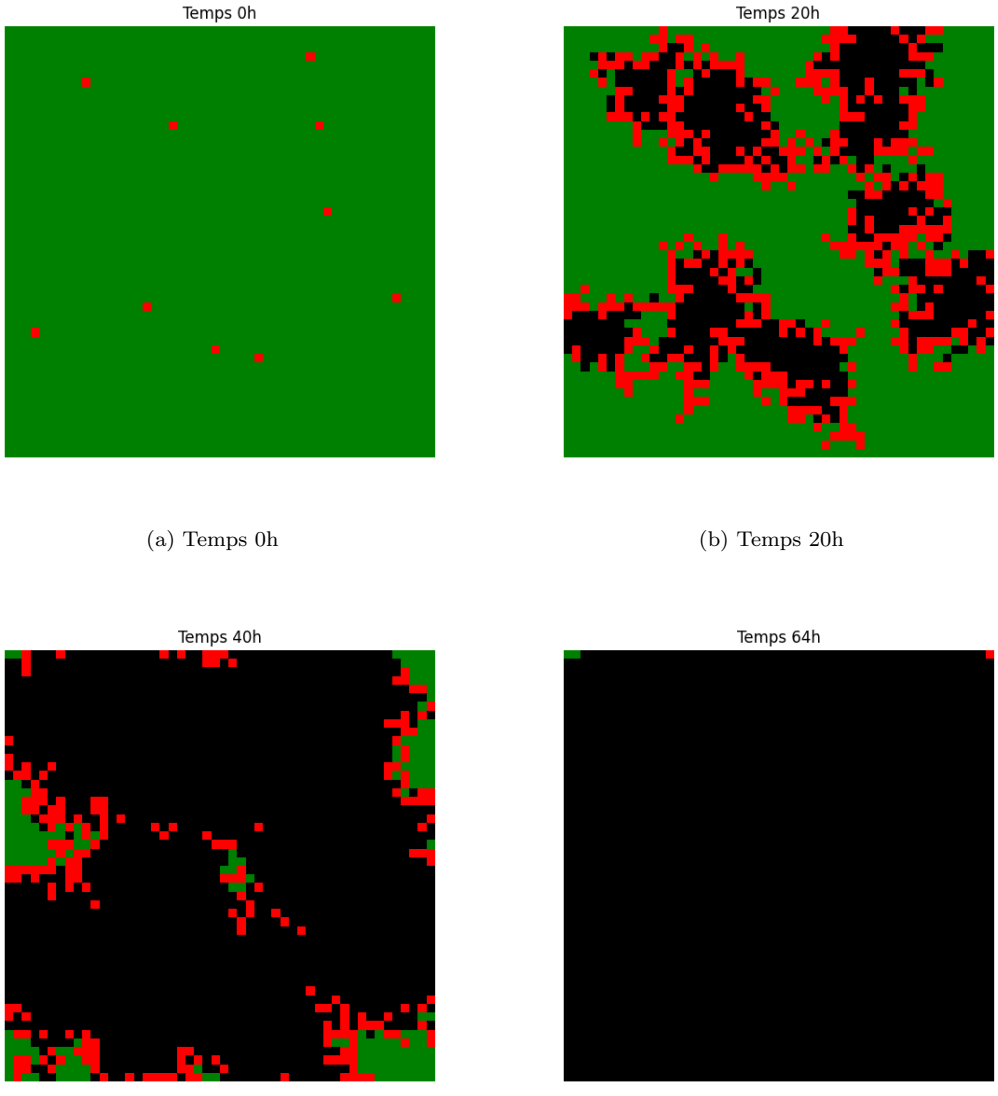
  Aquest procés modela el retard introduït per la humitat de cada cel·la. Com més humitat més li costarà començar a cremar, necessitarà més temps amb contacte amb el foc per començar.

Anem un instant de temps després d'un altre perquè així podem controlar de manera realista el comportament del foc, tenint en compte quant triga a començar a encendre's i a cremar-se.

El codi que implementa el que s'acaba d'explicar és el següent:

```
if state[i, j] == 1:
    new_vegetation_timer[i, j] += 1
    if new_vegetation_timer[i, j] >= vegetation[i, j]:
        new_state[i, j] = 2
elif state[i, j] == 0:
    for ni, nj in neighbors:
        if state[ni, nj] == 1:
            new_humidity_timer[i, j] += 1
            if new_humidity_timer[i, j] >= humidity[i, j]:
                new_state[i, j] = 1
```

3.5 Resultats sense vent



(a) Temps 0h

(b) Temps 20h

(c) Temps 40h

(d) Temps 64h

Figura 2: Evolució de la propagació d'un incendi amb vent en diferents temps

Clicant en aquest enllaç es mostra el GIF amb tota l'evolució de l'incendi sense vent: `https://ja.cat/YzcAg`

En aquest cas, la propagació del foc depèn exclusivament de la distribució de la humitat i la vegetació, així com de la proximitat entre cel·les actives. A l'inici, el foc s'inicia de forma aleatòria en 10 cel·les (per a tots els casos hem posat una seed al codi que es podria treure però així podem mantenir-ho reproduïble). Aquestes comencen a cremar i transmeten el foc a les cel·les veïnes segons les condicions de combustibilitat.

Observem com la propagació s'expandeix de manera radial, al costat d'allà on comença a cremar, òbviament, seguint formes irregulars causades per la variabilitat de l'humitat i vegetació de les capes. Les zones amb vegetació densa i baixa humitat afavoreixen una expansió més ràpida, mentre que àrees amb humitat elevada retarden el foc.

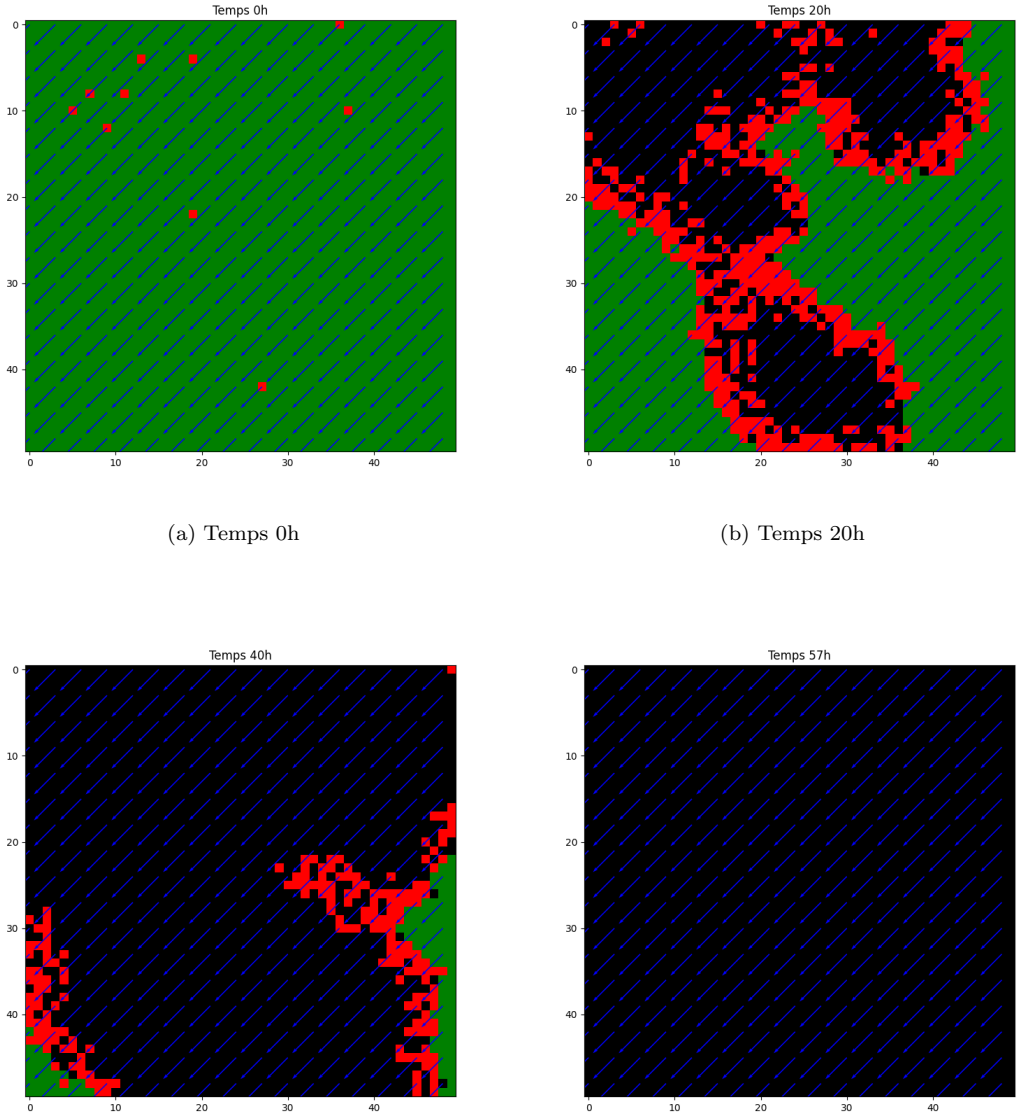
3.6 OPTATIU: Algorisme de propagació amb vent

Aquest model és idèntic a l'anterior però, per tal de fer-lo més realista, s'ha introduït una tercera capa ambiental: el vent. Aquesta capa és una matriu de la mateixa mida que la resta, on a cada cel·la se li assigna una direcció de vent codificada com un enter entre 0 i 7. Cada valor representa una de les vuit direccions possibles en una graella bidimensional (Nord, Nord-Est, Est, etc.).

Aquest petit fragment de codi mostra com s'incrementa el valor de `humidity_timer` d'una cel·la en funció de la coincidència entre la direcció del vent local i la direcció des d'on arriba el foc. Si el vent i la propagació són en la mateixa direcció, la cel·la acumula calor més ràpidament, i per tant superarà abans el seu llindar d'humitat, encenent-se abans.

```
if d == wind_dir:
    new_humidity_timer[ni, nj] += 4  # propagacio afavorida pel vent
else:
    new_humidity_timer[ni, nj] += 1  # propagacio normal
```

3.7 OPTATIU: Resultats amb vent



(a) Temps 0h

(b) Temps 20h

(c) Temps 40h

(d) Temps 57h

Figura 3: Evolució de la propagació d'un incendi amb vent en diferents temps

Clicant en aquest enllaç es pot visualitzar el GIF complet amb tota l'evolució temporal de l'incendi amb vent: `https://ja.cat/eH5QY`

En aquest cas, s'ha afegit una capa de vent que introdueix una direcció preferent de propagació del foc a cada cel·la. La direcció s'ha assignat aleatòriament, però es manté fixa durant tota la simulació. Quan el foc es transmet en la mateixa direcció que el vent local, el procés de combustió s'accelera, afavorint una propagació més ràpida en aquest sentit.

Els resultats mostren clarament com la propagació ja no és simètrica com en el cas sense vent, sinó que es veu infuenciada per la direcció del vent. El foc avança amb més intensitat i rapidesa en les regions on la direcció del vent coincideix amb la distribució favorable de vegetació i baixa humitat. Això provoca formes més allargades.

L'efecte del vent no només accelera la propagació, sinó que també influeix en la manera en com cremen les zones.

3.8 Diagrama de flux

A continuació, es presenta un diagrama de flux que descriu el comportament de cada cel·la dins del model d'autòmat cel·lular implementat. Aquest diagrama mostra el procés de transició d'estats segons les condicions definides per la humitat `humidity` i la vegetació `vegetation`, així com l'efecte del vent en la propagació del foc.

Inicialment, cada cel·la es troba en estat `Idle`. Quan rep foc d'una cel·la veïna en combustió, entra en fase de calentament (`Heating`), durant la qual acumula impuls a través del paràmetre `humidity_timer`. Aquest valor s'incrementa de manera diferent segons la direcció del vent: si el foc prové d'una direcció alineada amb el vent dominant, l'increment és de +4, i +1 en cas contrari.

Quan el `humidity_timer` supera el llindar marcat per la humitat de la cel·la, aquesta comença a cremar (`Burning`). Durant aquest estat, es propaga el foc a les cel·les veïnes i es consumeix progressivament la vegetació (`vegetation_timer`). Finalment, quan aquesta esgota el seu valor, la cel·la entra en estat `Burnt`.

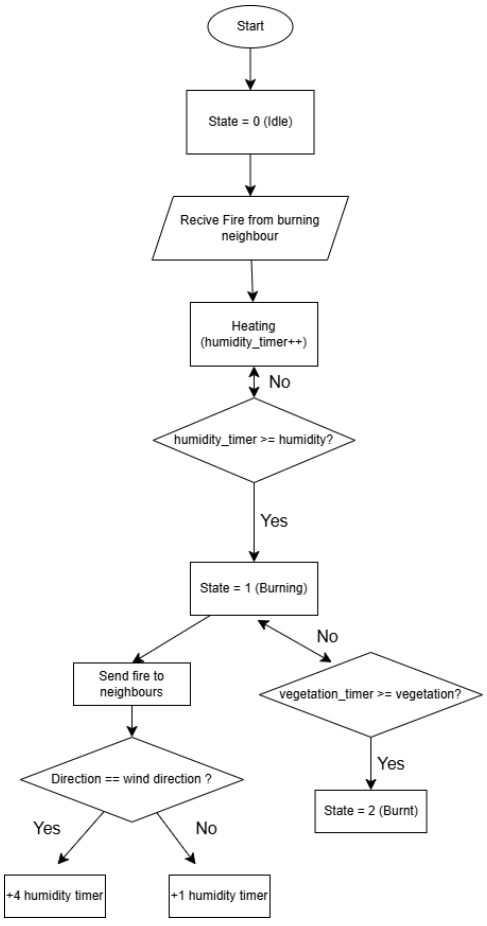


Figura 4: Diagrama de flux del model de propagació (SDL)

4 Ús de la Intel·ligència Artificial

Al llarg del desenvolupament d'aquesta pràctica s'ha fet un ús puntual d'eines com ChatGPT, especialment per resoldre errors de codi i crear les estructures bàsiques i inicials.

La IA ha estat un complement útil, però en cap moment ha substituït la feina de disseny, anàlisi ni desenvolupament. Ens ha ajudat, sobretot, en moments on calia desbloquejar errors concrets o accelerar la creació d'algun fragment de codi senzill, com ara la inicialització de les matrius o la lectura i creació d'alguns fitxers que no enteníem.

En general, ha ajudat a resoldre gairebé tots els errors que trobàvem.

5 Conclusions

Aquesta pràctica ha permès entendre i aplicar els conceptes d'autòmats cel·lulars tant des d'un punt de vista més teòric (primera sessió) com pràctic (segona sessió). En una primera fase s'ha treballat amb regles elementals de Wolfram per observar-ne els comportaments propis així com combinar-los, mentre que en una segona fase s'ha aplicat a un escenari realista com seria un incendi forestal.

A nivell de codi, s'han desenvolupat funcions que gestionen l'evolució d'estats en el temps, tenint en compte diverses capes d'informació (vegetació, humitat i vent) que interactuen entre elles. S'ha tingut molt en compte la visualització d'aquests reesultats per a poder intepretar perfectament el comportament del sistema i validar que l'algoritme funciona de manera coherent.

Finalment, cal destacar que la pràctica ha requerit una anàlisi acurada del problema, una estructuració modular del codi i una bona comprensió dels autòmats cel·lulars. El resultat final és un sistema que és capaç de simular perfectament el comportament d'un incendi per cel·les.