Simuler les feux de forêt

Victor Sarrazin

25 mai 2025

Introduction

Avec le changement climatique, les feux de forêt sont de plus en plus fréquents et dévastateurs, à l'image de ceux en Californie en janvier 2025.

Dans ce cadre, les modélisations informatiques des feux de forêt permettent de simuler leur évolution et ainsi de prévoir les zones à risque. Elles offrent également la possibilité de tester l'impact de certaines transformations sur ces incendies, afin d'identifier des moyens de réduire les conséquences des catastrophes sans dénaturer les forêts.

1 Automates cellulaires

Pour réaliser nos modélisations, nous allons utiliser des automates cellulaires. Ces outils permettent de représenter des systèmes avec des interactions locales entre les éléments qui les constituent. Les automates cellulaires ont notamment été popularisés avec *Le Jeu de la Vie* de *Conway*.

Définition 1. Un automate cellulaire est la donnée d'un triplet (Q, M, f) avec :
- Q un ensemble d'états
- M une matrice de taille $n \times m$, où chaque $m_{i,j}$ représente une cellule de la grille
- $f : \mathcal{M}_{n,m}(Q) \longrightarrow \mathcal{M}_{n,m}(Q)$ une *fonction de transition* qui à une grille renvoie la grille suivante

La définition de la fonction de transition dépend donc du type de voisinage considéré. Il existe deux principaux types de voisinages : celui de *von Neumann* et celui de *Moore*. Nous avons choisi d'utiliser le voisinage de *Moore* pour nos modélisations pour prendre en compte le plus d'intéractions possibles, celui de *von Neumann* étant plus limitatif, notamment en ce qui concerne les interactions diagonales.

Définition 2. Le voisinage de *Moore* est composé de la cellule centrale et de ses 8 voisins adjacents (horizontalement, verticalement et diagonalement).

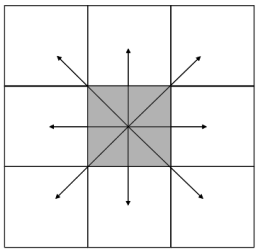

FIGURE 1 – Voisinage de Moore

Il serait possible dans une optique d'avoir des modèles encore plus précis d'utiliser des voisinages de *Moore* étendus, avec plusieurs rayons de voisins. Cependant par souci de simplicité nous nous sommes restreints à un rayon.

2 Modèle d'Alexandridis

Dans un premier temps, il est possible de réaliser une modélisation simple des feux de forêt en associant à chaque cellule une probabilité fixe de s'enflammer si l'un de ses voisins est en feu. Une telle tentative a donné des résultats mitigés. C'est pourquoi l'objectif de cette deuxième partie est de présenter un modèle plus avancé de modélisation des feux de forêts, appelé modèle d'*Alexandridis*.

2.1 Présentation du modèle

Le modèle d'*Alexandridis* prend en compte divers phénomènes pour simuler la propagation des incendies. Par la suite, nous avons choisi de nous concentrer sur la densité de végétation, le type de végétation ainsi que le vent dans la zone. Le modèle original prend également en considération l'élévation du terrain.

Proposition 1. On utilise les règles de transition suivantes, pour $(i, j) \in \mathbb{N}^2$, $t \in \mathbb{N}$:
- Si $m_{i,j}(t) = \texttt{feu}$ alors $m_{i,j}(t+1) = \texttt{brulé}$
- Si $m_{i,j}(t) = \texttt{feu}$ alors $m_{i\pm1,j\pm1}(t+1) = \texttt{feu}$ avec une probabilité p_b
- Si $m_{i,j}(t) = \texttt{brulé}$ alors $m_{i,j}(t+1) = \texttt{brulé}$

Proposition 2. La probabilité p_b qu'une cellule brûle est définie par $p_b = p_h(1+p_{veg})(1+p_{den})p_{vent}$ avec $p_h = 0.27$ une constante, et p_{veg} et p_{den} des coefficients relatifs au type de cellule.

Proposition 3. La probabilité p_{vent} liée au vent est définie par
$p_{vent} = \exp(0.045v) \times \exp(0.131v \times (\cos(\theta) - 1))$ avec θ l'angle entre la propagation du feu et la direction du vent, et v la vitesse du vent (en m/s)

	p_{veg}	p_{den}
Arbres	0.3	0.3
Arbres denses	0.3	0
Champs	-0.1	0

FIGURE 2 – Probabilités p_{veg} et p_{den} selon le type de végétation

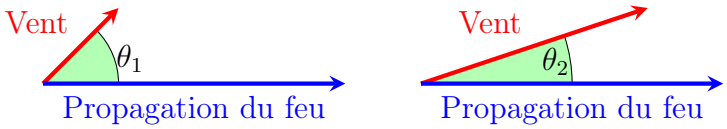

FIGURE 3 – Configurations de vent

On a $\theta_1 > \theta_2$ et une vitesse plus élevée dans la seconde situation, ainsi la probabilité p_{vent} sera plus élevée dans la seconde situation.

2.2 Résultats

Voici un résultat de modélisation de feu, avec un vent de $15m/s$ orienté vers l'est :

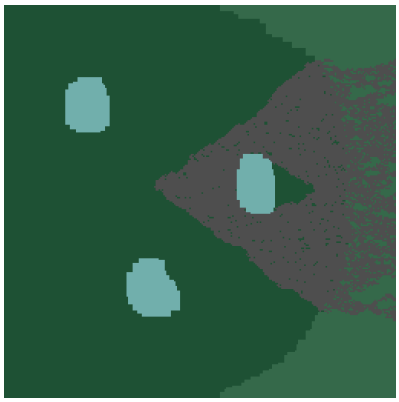

FIGURE 4 – Simulation Alexandridis

3 Hashlife

Afin de réaliser des modélisations plus conséquentes (notamment en augmentant la taille de la grille), il faut trouver une manière de réduire le nombre de calculs. C'est l'objectif de l'algorithme

de mémoïsation `Hashlife`, notamment utilisé dans le logiciel `Golly` [1] pour les simulations du *Jeu de la Vie* de *Conway*.

3.1 Principe de Hashlife

Au lieu de calculer chaque cellule indépendamment, l'algorithme considère des groupements de cellules, dits *macro-cellules*, de taille $2^n \times 2^n$, que l'on peut redécouper en 4 quadrants de taille $2^{n-1} \times 2^{n-1}$ (partie en bleu sur la figure).

On appelle *résultat* la macro-cellule centrale de taille $2^{n-1} \times 2^{n-1}$ (partie en rouge sur la figure).

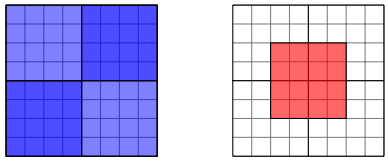

FIGURE 5 – Une macro-cellule de taille $2^3 \times 2^3$

Ce découpage en macro-cellules permet de calculer le résultat sans considérer aucune information extérieure à la macro-cellule pendant 2^{n-2} unités de temps. Ce calcul est effectué récursivement selon deux cas :

— Cas de base ($n = 2$) : Le résultat est composé de seulement 4 cellules. Dans ce cas, nous le calculons directement en appliquant les règles de notre automate cellulaire.

— Cas récursif ($n > 2$) : Nous avons 4 quadrants de taille $2^{n-1} \times 2^{n-1}$ pour lesquels nous calculons récursivement leur résultat (illustré en bleu). Ensuite, il est nécessaire de calculer les 5 macro-cellules de taille $2^{n-2} \times 2^{n-2}$ (en rouge) en tenant compte des macro-cellules de taille $2^{n-1} \times 2^{n-1}$ associées (exemple en vert).

Enfin, après ces précalculs, les quatre macro-cellules de taille 2^{n-2} (en jaune) peuvent être calculées pour obtenir le résultat recherché.

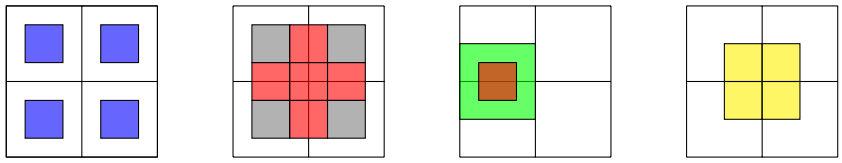

FIGURE 6 – Calcul dans le cas $n > 2$

Si, dans le cas $n = 2$, le nombre d'opérations est similaire à celui de la simulation naïve, le cas $n > 2$ permet une réelle économie de calcul.

1. https ://golly.sourceforge.io/

3.2 Mémoïsation

Cependant, `Hashlife` ne se contente pas de réduire le nombre de calculs par itération. L'algorithme tire parti du fait que les automates cellulaires présentent des répétitions de motifs durant leur exécution, notamment avec les *gliders* dans le *Jeu de la Vie* qui se déplacent jusqu'aux bordures (infinies) de la grille en répétant 4 fois les mêmes motifs :

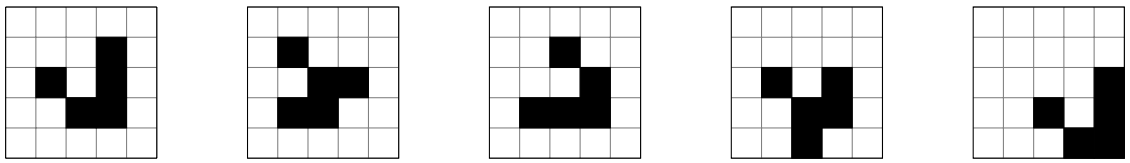

FIGURE 7 – Un *glider* et sa périodicité

C'est pourquoi `Hashlife` réalise également une mémoïsation des résultats de chaque quadrant à l'aide d'une table de hachage, d'où le nom de l'algorithme. Cela permet de réutiliser au maximum les calculs et d'éviter de recalculer les situations récurrentes.

Mais le nombre de configurations différentes pour une macro-cellule de taille $2^n \times 2^n$ étant majoré par 2^{4^n}, il est nécessaire de mettre en place des techniques efficaces de mémoïsation pour ne pas saturer la mémoire.

Une technique est de ne considérer qu'une instance de chaque macro-cellule existante dans la grille, et de fonctionner avec un système de pointeurs. Cette technique permet aussi de hasher beaucoup plus facilement chaque grille afin de réduire au maximum la taille de chaque couple clé/valeur dans la table de hashage

3.3 Complexité

En raison du nombre de configurations, il est complexe d'avoir une complexité théorique. Empiriquement dans des cas non dégénérés on observe une complexité en temps logarithmique en la taille de la grille. Cependant la mise en place d'un bon *garbage collector* pour réduire au maximum les utilisations de mémoire extrême viennent augmenter cette complexité pour les cas avec peu de répétitions.

Bien que `Hashlife` réduise drastiquement le temps de calcul pour de grandes grilles et permette de s'affranchir de leurs bornes en exploitant les motifs répétitifs, il existe toujours des cas dégénérés où l'algorithme a une complexité similaire à celle de l'approche naïve si les motifs ne se répètent jamais [2].

2. L'algorithme `Quicklife` permet de résoudre ce problème

4 Conclusion

Les automates cellulaires permettent de simuler de nombreux systèmes à interactions, tels que les feux de forêt. Cependant, pour obtenir des simulations d'une taille convaincante (et dans un temps raisonnable), il est nécessaire d'implémenter des algorithmes plus avancés comme `Hashlife`.