Simuler les feux de forêt

Victor Sarrazin

10 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 l'évolution des feux, afin de prévoir les zones à risques. Mais les simulations informatiques permettent aussi de tester l'impact de certaines transformations sur ces feux, afin de trouver des manières de réduire l'impact des catastrophes, sans pour autant dénaturer les forêts.

1 Automates cellulaires

1.1 Généralités

Pour réaliser nos modélisations, nous allons utiliser des automates cellulaires. Les automates cellulaires permettent de représenter des systèmes avec des intéractions locales entre les éléments qui constituent ces derniers. 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 case 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 utilisé. 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 limitant notamment sur les diagonales.

Définition 2. Le voisinage de *Moore* est composé du noeud et de ses 8 voisins.

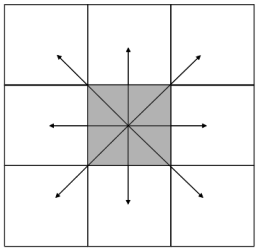

FIGURE 1 – Voisinage de Moore

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, mais dans un soucis de simplicité nous nous sommes restreints à un rayon.

1.2 Représentation d'un forêt

Afin de représenter une forêt avec un automate cellulaire nous avons défini une liste d'état que les cases pouvaient prendre.

Définition 3. Les états possibles sont :
- Arbres
- Arbres denses
- Champs
- Feu
- Case brulée*
- Eau*

A noter que les cases $(*)$ ne peuvent brûler.

Nous avons choisi de travailler avec une matrice de taille 256×256 pour des raisons techniques : au delà le temps de calcul devenait trop long et l'ajout de lignes/colonnes ne représentait pas un intérêt suffisant pour le faire par rapport au temps de calcul que celà impliquait.

2 Modèle d'Alexandridis

Dans un premier temps il est possible de faire une modélisation naïve des feux de forêt en associant à chaque case une probabilité fixe de brûler si un de ses voisins est en feu. Une telle tentative a été réalisée avec des résultats quelque peu satisfaisants, c'est pourquoi l'objectif de cette deuxième partie est de voir un modèle plus poussé de modélisation des feux de forêts, dit modèle d'*Alexandridis*.

2.1 Présentation du modèle

Le modèle d'*Alexandridis* prend en compte divers phénomènes pour modéliser cette propagation. Dans la suite nous avons choisi de nous restreindre à la densité de végétation, au type de végétation ainsi qu'au vent dans la zone. Le modèle original quant à lui prend aussi en compte 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 $m_{i,j}(t) =$ `feu` alors $m_{i,j}(t+1) =$ `brulé`
- Si $m_{i,j}(t) =$ `feu` alors $m_{i\pm1,j\pm1}(t+1) =$ `feu` avec une probabilité p_b
- Si $m_{i,j}(t) =$ `brulé` alors $m_{i,j}(t+1) =$ `brulé`

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

	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

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)

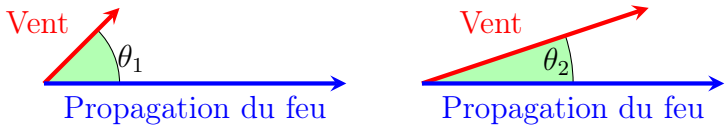

FIGURE 3 – Schémas pour le vent

2.2 Résultats

Dans un premier temps comparons les résultats de la modélisation naïve et celui du modèle d'*Alexandridis* :

On remarque que la solution naïve a une propagation plus lente et un front de feu plus étendu que celui du modèle d'*Alexandridis*.

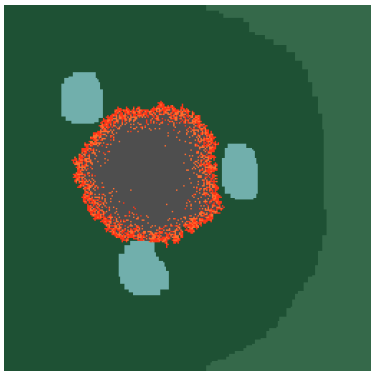

FIGURE 4 – Solution naïve

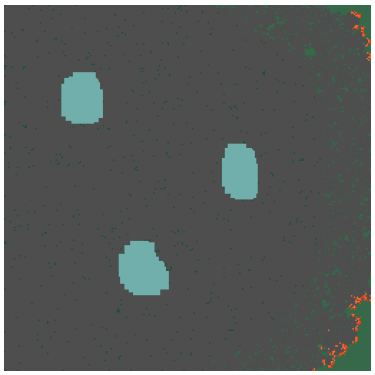

FIGURE 5 – *Alexandridis*

Comparons maintenant deux modélisations avec des densités de végétation différentes, et sous $15m/s$ de vent vers l'est :

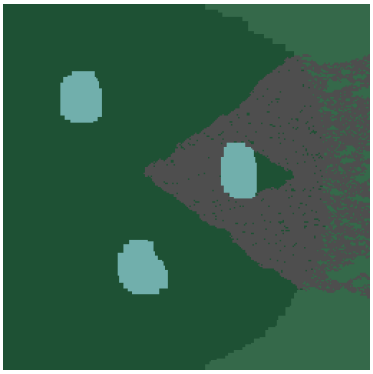

FIGURE 6 – Végétation normale

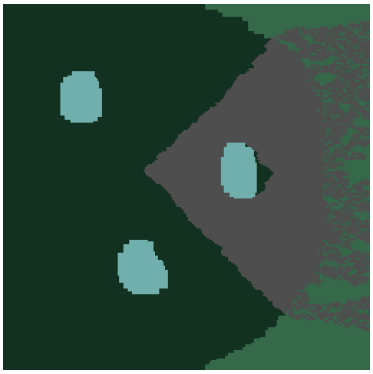

FIGURE 7 – Végétation dense

On remarque principalement deux effets notoires sur les deux modélisations.

D'une part, la progression dans une végétation normale est moins rapide (ce qui fait sens empiriquement), et donc le front est moins étendu. La surface brulée dans le cas de la densité normale présente aussi des trous là où celle de la densité élevée est lisse, ce phénomène est expliqué par le fait que p_b est plus élevée pour une densité plus forte.

D'autre part, dans les deux modélisations on voit clairement l'impact du vent : le front de feu s'est déplacé suivant un cône poussé par le vent.

2.3 Considérations algorithmiques

Cette sous partie a pour objectif de rendre compte de l'implémentation de la modélisation réalisée et de comprendre le choix de 256×256.

Voici les complexités des différentes fonctions créés :

— Calcul de p_b : $O(1)$
— Calcul du passage de t à $t+1$: $O(n^2)$ avec n la longueur de la grille, avec une estimation de 100 opérations élémentaires pour chaque case
— Affichage de l'état t : $O(m^2 \times n^2)$ avec n la longueur de la grille et m la taille d'une case représentée sur l'écran

Ainsi pour N itérations, on trouve une complexité totale de $O(m^2 \times n^2 \times N)$, numériquement avec $n = 256$, $m = 2$ et $N = 200$ on trouve $\approx 10^5$, et donc on en déduit grossièrement $\approx 10^7$ opérations élémentaires, pour un temps de calcul d'environ 5 secondes.

On comprend donc pourquoi augmenter la valeur de n (toutes choses égales par ailleurs) ou de N conduirait à des simulations plus longues.

3 Transformations

Puisque nous avons maintenant un modèle pour réaliser des simulations, nous allons nous intéresser aux transformations réalisables, notamment ici dans le cadre de cette étude aux tranchées et chemins.

3.1 Explications

Nous allons donc considérer la transformation suivante : l'ajout de tranchées ou de chemins forestiers, des transformations légères dans les forêts qui permettent de ne pas dénaturer ces dernières.

Proposition 4. On ajoute un nouvel état Chemins/Tranchées avec $p_{veg} = -0.55$ et $p_{den} = 0$ à notre modèle.

3.2 Résultats

L'objectif est maintenant de voir l'impact des tranchées, avec des configurations différentes de vent et de longueur de tranchées.

3.2.1 Tranchées/Sans tranchées

La première comparaison intéressante et de regarder la différence de propagation de feu avec et sans tranchées.

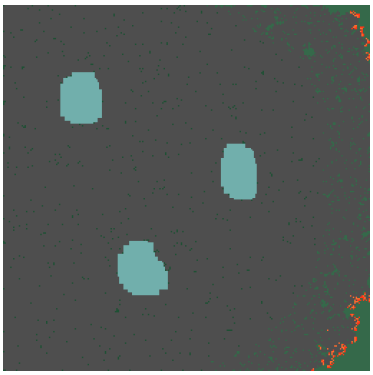

FIGURE 8 – Sans tranchées

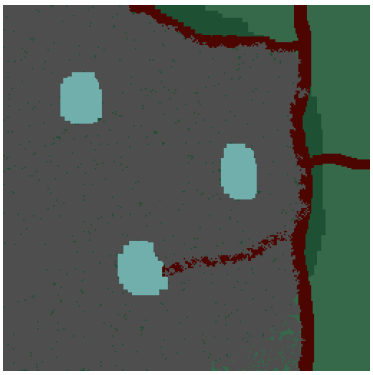

FIGURE 9 – Avec tranchées

On constate donc en comparant la FIGURE 8 et la FIGURE 9 que la présence de tranchées permet de bloquer le feu. Il est possible de critiquer le résultat dans la mesure où on a un nombre d'itérations limitées (ici 200), et que le feu aurait donc plus de temps pour se propager.

3.2.2 Vent faible/Vent fort

Dans un deuxième temps, comparons deux situations de vent différentes, dans le même sens mais avec une vitesse différente.

FIGURE 10 – 15 m/s de vent

FIGURE 11 – 30 m/s de vent

On constate donc en comparant la FIGURE 10 et la FIGURE 11 que la situation où le vent est plus fort rend moins efficace la tranchée/le chemin. Ce résultat est assez logique empiriquement dans la mesure où un vent plus fort permet au feu d'aller plus loin, et donc de passer plus facilement la tranchée.

Il est tout de même intéressant de mentionner que même si le feu passe outre la tranchée, moins de 25% de cette dernière est brulée à la fin (dans la zone impactée par le feu)

3.2.3 Tranchée mince/Tranchée large

Dans un dernier temps, comparons deux tailles de tranchée.

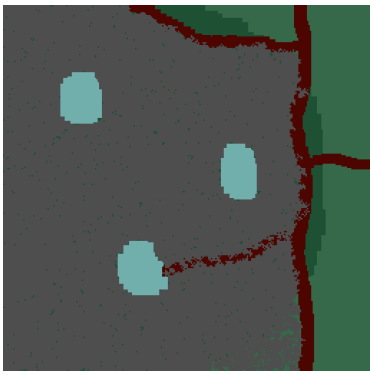

FIGURE 12 – Tranchée de 8m

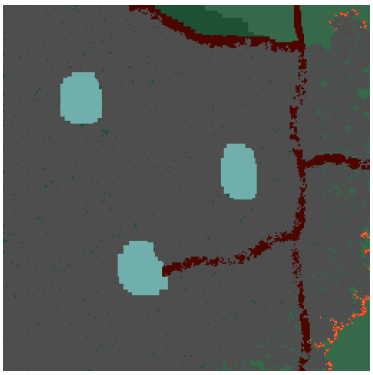

FIGURE 13 – Tranchée de 4m

On constate en comparant la FIGURE 12 et la FIGURE 13 que plus une tranchée est large, plus elle réduit les possibilités de propagation du feu. Entre ces deux simulations c'est la grande tranchée verticale qui a été réduite horizontalement et on voit bien l'impact de ce changement.

De plus on observe sur la FIGURE 13 que le front de feu est encore présent, et donc que si la simulation durait plus longtemps une zone encore plus vaste serait brûlée.

3.3 Possibles améliorations

Il est possible d'aller plus loin dans les simulations, nous nous sommes limités à un modèle intermédiaire, prenant en compte le type de végétation, la densité de cette dernière ainsi que le vent.

Il serait donc envisageable de prendre en compte d'autres paramètres dans la modélisation tels que l'élévation, l'humidité ou la température.

De plus il est aussi possible de considérer d'autres transformations telles que la réduction de la densité de végétation car on a vu en FIGURE 7 qu'une végétation dense favorisait la propagation, ou encore considérer une intéraction plus complexe avec les cours d'eau/lacs, ici considérés uniquement comme des zones bloquant totalement le feu alors qu'elles peuvent lui permettre de tout de même progresser.

4 Annexe

4.1 Choix d'implémentation

Pour implémenter les différents modèles nous avons choisi d'utiliser le langage `C99`. Le choix du langage a été motivé principalement par deux aspects :

— Le `C` permet par son bas niveau de gagner un certain temps d'exécution par rapport à d'autres langages, utile ici au vu du nombre de calculs à effectuer dans nos grilles.
— Le `C` étant plus utilisé que le `OCaml`, il bénéficie par conséquent de plus de bibliothèques externes.

Nous avons utilisé les bibliothèques non standard suivantes :

— `SDL2` : Pour réaliser une interface graphique afin de visualiser les évolutions.
— `cJSON` : Pour manipuler des fichiers `JSON` et représenter les grilles afin de pouvoir les importer.
— `png` : Pour créer des fichiers `PNG` et les manipuler en `C` afin d'exporter nos simulations.

Pour créer plus facilement les grilles, nous avons créé un site web avec `React.JS`.