Vie Artificielle

Automates Cellulaires 2D (modélisation, feux de forêt)

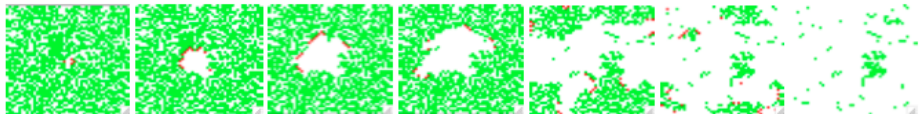
Mise à jour : Février 2018

Mots clés: *Feux de forêt.*

Une page est disponible sur le site http://pages.isir.upmc.fr/~bredeche *- Vous y trouverez des informations diverses (lien vers les sources, les sujets, etc.). En particulier vous trouverez un lien vers les codes sources donnés en annexe.*

Automates cellulaires 2D : Les feux de forêts [sur ordinateur]

On s'intéresse à la simulation du développement d'un feu de forêt. Pour mémoire, il s'agit d'un automate cellulaire 2D ou une cellule ne devient active que si un seul de ces voisins est actif (voisinage de von Neumann). Pour réaliser cet automate, on vous demande d'étudier le code source en annexe *ForestFire.java* (disponible sur le wiki) et de le modifier.



Les règles de mise à jour sont les suivantes (et leur interprétation):
- une cellule blanche reste blanche
  - ie. un emplacement vide reste vide
- une cellule verte reste verte sauf si au moins une des cellules voisines est rouge
  - ie. un arbre prend feu si au moins un de ces voisins est en feu
- une cellule rouge devient blanche
  - ie. un feu meurt en une itération

QUESTION 1. Programmer les règles de cet automate cellulaire. On vous demande ensuite de découvrir expérimentalement le seuil de percolation au dessus duquel la majorité de la forêt brûle.

QUESTION 2. Etendre l'automate cellulaire avec un état "cendres" (cellule de couleur noire) et modifiez la règle transformant une cellule rouge en cellule noire (ie. lorsqu'un feu meurt, on trouve des cendres). Une cellule noire disparaît après une itération.

QUESTION 3. Introduisez de nouvelles règles associées à une probabilité de réalisation.

- un arbre peut spontanément prendre feu (avec une probabilité *p1*)
- un emplacement vide peut spontanément voir apparaître un nouvel arbre (avec une probabilité *p2*)

Trouvez plusieurs couples de valeurs *p1* et *p2* tel que la forêt est une densité de 0.5 à long terme. Tracez le nombre d'arbres vivants au cours du temps à l'aide du script plot.py (cf. lien "tracer des valeurs CSV").

Astuce: Il va falloir mesurer l'évolution de la densité d'arbres au cours du temps et laisser tourner votre simulation jusqu'à stabilisation.

QUESTION 4. Jusqu'ici, vous avez utilisé une mise à jour synchrone (ie. en utilisant deux tableaux. L'un pour représenter l'état de la forêt à *t*-1, et l'autre pour calculer l'état de la forêt à *t*). Vous devez maintenant adapter votre code pour n'utiliser qu'un seul tableau. Pour cela, vous devez changer la méthode de mise à jour de l'automate pour que la mise à jour soit asynchrone. On vous demande d'implémenter les méthodes suivantes:

- Méthode 1: à chaque itération, on choisit une cellule au hasard que l'on met à jour. Cette méthode à l'avantage de la simplicité, mais il est possible que certaines cellules soient mises à jour plus souvent que d'autres.

- Méthode 2: à chaque itération, on choisit une cellule au hasard que l'on met à jour. Cette cellule doit être choisie dans la liste des cellules qui n'ont pas encore été mise à jour. Une fois toutes les cellules mises à jour, on recommence. Cette méthode à l'avantage de garantir que toutes les cellules sont mises à jour le même nombre de fois.

- Le seuil de percolation que vous aviez identifié précédemment est-il encore pertinent? Quels sont les seuils de percolation pour chaque méthode?

Suggestion: pour faire cette question, remettez temporairement les probabilités p1 et p2 à 0.

PREMIER PAS VERS LE PROJET.

A partir de la question 4, étendez votre feu de forêt pour intégrer la dispersion des cendres (une case noire redevient blanche après N=10 itérations).

Implémentez les règles permettant de modéliser l'étalement d'une quantité d'eau dans l'environnement. Vous pourrez par exemple modéliser l'eau présente en prenant en compte la quantité d'eau. Suggestion: utilisez une mise à jour asynchrone randomisé pour éviter les problèmes de concurrence d'accès.

Construisez un monde avec des arbres (+feu) et des étendues d'eau, et où les arbres (non brûlés) agissent comme barrage pour l'eau. Vous pourrez par exemple utiliser deux automates cellulaires différents et composer un affichage combinant les deux.

Etendez votre environnement avec une carte d'altitude (ie. une matrice qui pour chaque case de l'environnement donne l'altitude) et modifier les règles d'écoulement d'eau pour prendre en compte l'altitude dans la direction de l'écoulement. On souhaite pouvoir observer à la fois des écoulements et le remplissage d'un bassin (ie. une quantité d'eau initialement sur une cellule est répartie sur les cellules voisines, mais limitée par la géographie, ie. les cases dont l'altitude bloquent l'écoulement d'eau).

Suggestion pour le projet: l'intérêt d'une méthode de mise à jour asynchrone est que vous pourrez facilement moduler le nombre de mises à jour faites à chaque pas. Par exemple, vous pourrez décider qu'à chaque déplacement de vos agents dans l'environnement, seul 1/10e de votre environnement sera mis à jour. C'est très utile lorsque la mise à jour de votre environnement est très couteuse en temps, afin d'éviter qu'elle ralentisse l'ensemble de la simulation. De même

Annexe: ForestFire.java [à compléter]

```
public class ForestFire extends CAtoolbox {

     public static void main(String[] args) {

          int dx = 50;
          int dy = 50;

          int[][] tableauCourant = new int[dx][dy];
          int[][] nouveauTableau = new int[dx][dy];

          int delai = 500;//100;

          int nombreDePasMaximum = 10000;//1000;
          int it = 0;

          double densite = 0.55; //0.55; // seuil de percolation ^ 0.55

          // optionnel: initialise la visualisation dans une fenetre

          CAImageBuffer image = new CAImageBuffer(dx,dy);
          ImageFrame imageFrame = ImageFrame.makeFrame("Forest fire", image, delai, 200, 200 );

          // initialisation (peuple la foret)

         for ( int x = 0 ; x != dx ; x++ )
          for ( int y = 0 ; y != dy ; y++ )
               if ( densite >= Math.random() )
                    tableauCourant[(int)x][(int)y]=1; // tree

         tableauCourant[dx/2][dy/2] = 2; // burning tree

          // on fait tourner l'automate
          while ( it != nombreDePasMaximum )
          {
               // 1a - affiche de l'Etat courant dans la fenetre graphique (toutes les cellules)
               image.updateForest(tableauCourant);


               // 1 - mise a jour de l'automate (dans le tableau en tampon)
               for ( int x = 0 ; x != tableauCourant.length ; x++ )
                    for ( int y = 0 ; y != tableauCourant[0].length ; y++ )
                    {
                         /* **** A COMPLETER **** */
                    }


               // 2 - met a jour le tableau affichable
               for ( int x = 0 ; x != tableauCourant.length ; x++ )
                    for ( int y = 0 ; y != tableauCourant[0].length ; y++ )
                         tableauCourant[x][y] = nouveauTableau[x][y];


               it++;
```



```
               try {
                    Thread.sleep(delai);                // ne va pas trop vite...
               } catch (InterruptedException e)
               { }
          }

     }

}
```