



Percolation : étude d'une transition de phase

| | | |
|----------|--|-----------|
| 1 | Introduction | 2 |
| 2 | Histoire et définitions | 4 |
| 3 | Phénomène de seuil | 5 |
| 3.1 | Définitions complémentaires | 5 |
| 3.2 | Outils probabilistes | 6 |
| 3.3 | Résultats théoriques | 8 |
| 4 | Un joli résultat | 11 |
| 5 | Amas infini (complément ?) | 12 |
| 6 | Ouverture | 14 |
| 7 | Annexes | 16 |
| 7.1 | Démonstrations | 16 |
| 7.2 | Programmes des modélisations | 20 |
| 7.3 | Sources | 30 |

« Transition, transformation, conversion. »

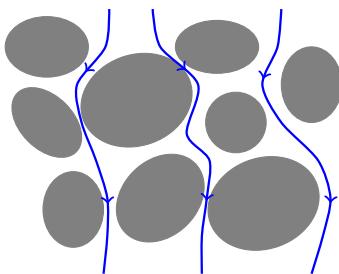
1 Introduction

Du grain de café au grain de sable

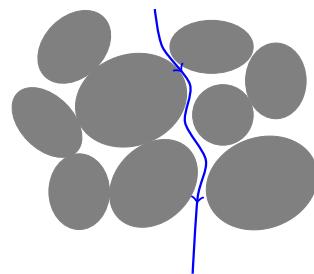
Lorsque l'on prépare un café, on dose sa "force" par la vitesse de l'écoulement de l'eau dans le filtre (que l'on peut assimiler à une roche poreuse). Plus elle est lente, plus il sera corsé, mais il faut avant tout que l'eau puisse atteindre la tasse. Comment peut-on s'en assurer ?



Crédits : café, roche



Milieu perméable



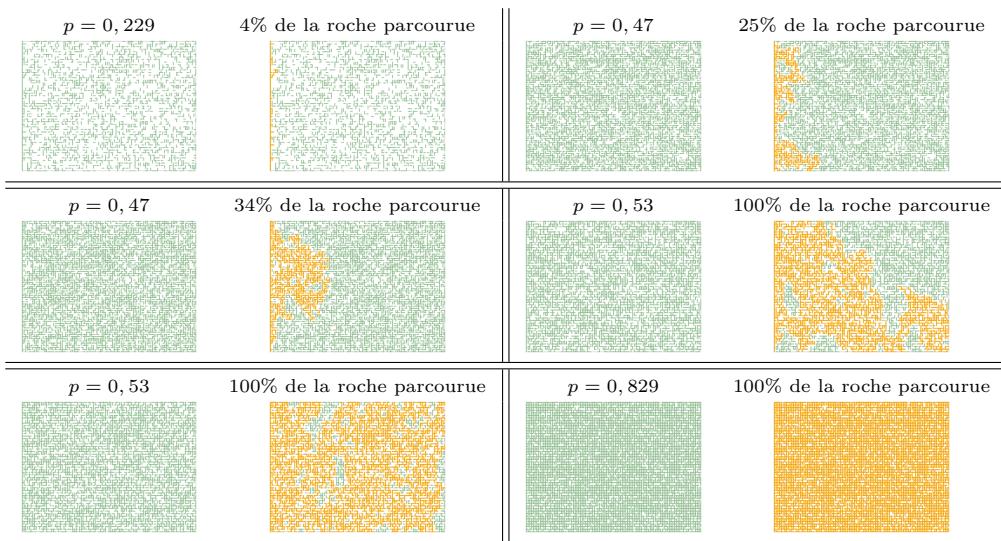
Milieu moins perméable

Dans ces deux cas (roche poreuse et café, qui s'avèrent donc être les mêmes pour nous) le phénomène qui nous permet d'étudier l'écoulement est appelé la **percolation**. Commençons par une modélisation !

On représente un **réseau carré** de dimension 50×100 dans \mathbb{Z}^2 où chaque **lien** à une même probabilité $p \in [0; 1]$ d'exister (il est modélisé en **vert** et on dit alors qu'il est **ouvert**, par opposition aux liens **fermés** qui ne sont pas représentés).

On part ensuite de la gauche et on cherche tous les chemins existants, à l'aide d'un parcours de graphe en profondeur, en suivant les liens ouverts afin de créer une **composante connexe ouverte**, aussi appelée **amas** ou **cluster** en anglais (modélisé en **orange** ci dessous).

Pour des valeurs de p distinctes, on observe que la taille de la composante connexe ouverte varie, et plus précisément qu'autour de $\frac{1}{2}$, sa taille évolue drastiquement.



[Voir algorithme](#)

On observe effectivement que pour $p \leq 0,47$, la composante connexe ouverte est réduite et ne traverse pas la moitié du réseau, tandis que pour $p \geq 0,53$, elle est bien plus imposante, couvrant plus de la moitié du réseau et le traversant entièrement.

Problématiques

On semble donc faire face à un **seuil**, ce qui motive les questionnements suivants : qu'est-ce que ce seuil auquel on fait face lors de nos simulations et quelle est son influence sur la percolation ?



2 Histoire et définitions

La théorie de la percolation a été mise en place au milieu des années 1950 par deux chercheurs, Simon BROADBENT et John HAMMERSLEY lorsqu'ils cherchaient à expliquer en modélisant numériquement le dysfonctionnement de masques à gaz encrassés. Ces masques étaient faits en partie de charbon, qui grâce à sa porosité particulière, sert de "tamis moléculaire".

Définitions

La **théorie de la percolation** est une branche des mathématiques et de la physique statistique qui s'intéresse aux caractéristiques de milieux aléatoires et plus précisément aux ensembles de sommets connectés dans un graphe aléatoire.

Le **phénomène de percolation** est un effet de seuil : une faible variation de paramètres différencie la transmission limitée à quelques voisins proches ou bien à des voisins arbitrairement loin.

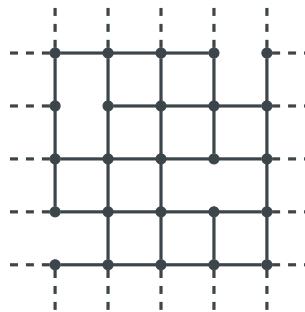
On dit alors qu'il y a **percolation** dans un réseau si une quantité suffisante d'éléments (de nœuds) fusionnent afin de créer un amas de taille proportionnelle à celle du réseau (et donc infini si le réseau l'est aussi).

On distingue la **percolation de lien** qui attribue une probabilité de présence à un lien de la **percolation de site** qui attribue une probabilité de présence à un site.

3 Phénomène de seuil

3.1 Définitions complémentaires

On définit le graphe infini $\mathbb{L}_2 := (\mathbb{Z}^2, E^2)$ où $E^2 = \{(u, v) \in (\mathbb{Z}^2)^2 \mid \|u - v\|_1 = 1\}$. Nous nous limitons dans ce devoir à l'étude de réseaux carrés issus de \mathbb{L}_2 (voir l'exemple ci dessous). Lorsque le réseau est fini (respectivement infini), nous le modélisons à l'aide d'un graphe $G = (S, A)$ fini (respectivement infini), planaire, simple, non-orienté et connexe. Les ensembles au plus dénombrables S et A représentent respectivement les sommets et les arcs.



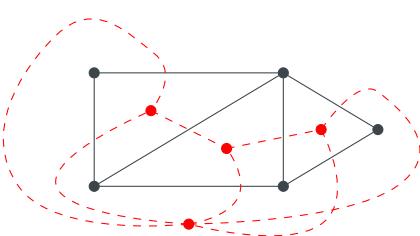
Exemple d'un réseau carré infini de dimension 2 sujet à une percolation de liens

Définitions

Un **modèle de percolation** (ici par lien) est une famille de variables aléatoires $(X_a)_{a \in A}$ à valeurs dans $\{0, 1\}$. Pour $a \in A$, le lien a est ouvert si $X_a = 1$ et fermé sinon.

Une **percolation de Bernoulli de paramètre p** est une percolation où la famille de variables aléatoires est indépendante et identiquement distribuées et chaque variable suit une loi de Bernoulli de même paramètre $p \in [0; 1]$.

Étant donné un graphe G non orienté planaire, on construit son **graphe dual** G' en associant à chaque face f de G un sommet de G' . Ensuite, pour chaque arête a de G séparant deux faces, on relie les sommets de G' associés aux deux faces par une arête a' de G' qui coupe a .



Exemple d'un graphe dual (en rouge).



Notations :

$\{x \rightsquigarrow y\} := \{\omega \in \{0,1\}^A \mid x \rightsquigarrow^\omega y\}$ l'évènement : « il existe un chemin entre $x \in S$ et $y \in S$. »

$\{x \rightsquigarrow \infty\}$ l'évènement : « il y a percolation en $x \in S$ ».

$\theta_x : p \mapsto P_p(\{x \rightsquigarrow \infty\})$ la probabilité qu'il y ait percolation en $x \in S$ pour le paramètre p .

$\Pi : p \mapsto P_p(\bigcup_{x \in S} \{x \rightsquigarrow \infty\})$ la probabilité qu'il y ait percolation.

3.2 Outils probabilistes

On travaillera avec l'espace probabilisé $(\Omega, \mathcal{A}, P_p)$ où

- $\Omega = \prod_{a \in A} \{0,1\} = \{0,1\}^A \simeq \mathcal{P}(A)$
- $\mathcal{A} = \mathcal{P}(\{0,1\})^{\otimes A}$ est tribu produit
- $P_p = \prod_{a \in A} \mu_p$ où μ_p est la mesure sur $\{0,1\}$ tel que $\mu_p(\{0\}) = 1-p$ et $\mu_p(\{1\}) = p$

Une configuration d'un graphe $G = (S, A)$ sera donc notée $\omega = (\omega_a)_{a \in A} \in \{0,1\}^A$. On retrouve qu'une arête a est ouverte dans la configuration ω ssi $\omega_a = 1$.

Une relation d'ordre

On muni Ω de la relation d'ordre suivante :

$\omega \preceq \omega'$ ssi toute arête ouverte de ω est ouverte dans ω' .

Ainsi $A \in \mathcal{A}$ est dit croissant si

$$\forall (\omega, \omega') \in A \times \Omega \quad \omega \preceq \omega' \Rightarrow \omega' \in A.$$

Tribu asymptotique :

Soit $(\mathcal{A}_k)_{k \in \mathbb{N}}$ une suite de tribus indépendantes sur (Ω, \mathcal{A}, P) .

On note \mathcal{B}_n la tribu engendrée par $(\mathcal{A}_k)_{k \geq n}$.

La tribu $\mathcal{B}_\infty = \bigcap_{n \in \mathbb{N}} \mathcal{B}_n$ est appelée tribu terminale.

Une autre définition de la tribu asymptotique est qu'elle est constituée des événements dit de queue, i.e. dont la réalisation dépend d'une suite de variables aléatoires indépendantes mais ne dépend d'aucun sous-ensemble fini de ces variables.

**Loi du zero-un de Kolmogorov :**

Tout événement de la tribu asymptotique est de probabilité 0 ou 1.

[Voir preuve](#)

Inégalité FKG :

Soit J au plus dénombrable dont chaque élément est dans l'état 0 avec une probabilité p ou bien 1 avec une probabilité $1 - p$.

On pose $\Omega = \{0, 1\}^J$.

Soit deux parties croissantes (au sens de \preceq) A et B de Ω , alors

$$P(A \cap B) \geq P(A)P(B).$$

[Voir preuve](#)



3.3 Résultats théoriques

Proposition

Il existe une probabilité critique $p_c \in [0; 1]$ telle que $\begin{cases} \theta_x(p) = 0 & \text{si } p \leq p_c \\ \theta_x(p) > 0 & \text{si } p > p_c \end{cases}$ et cela peu importe $x \in S$.

Preuve (existence d'une probabilité critique)

On définit une telle probabilité p_c comme

$$p_c := \sup(\{p \in [0; 1] \mid \theta_x(p) = 0\}) \text{ pour } x \in S.$$

Celle-ci existe bien car l'ensemble de départ est majoré.

Soit $p \in [0; p_c[$ (sous réserve d'existence, on ne traite pas le cas d'égalité) :

$p < p_c$ et p_c est la borne supérieure de l'ensemble des zéros de θ_x donc c'est le plus petit majorant de cet ensemble et p est alors un zéro de θ_x : il n'y a pas percolation.

Soit $p \in]p_c; 1]$ (sous réserve d'existence) :

$p > p_c$ donc p n'est pas un zéro de θ_x , et comme il s'agit d'une probabilité, elle est strictement positive en ajoutant l'argument de positivité.

Soit $(x, y) \in S^2$.

$$\begin{aligned} \theta_x(p) &= P_p(\{x \rightsquigarrow \infty\}) \\ &\geq P_p(\{x \rightsquigarrow y\} \cap \{y \rightsquigarrow \infty\}) \\ &\geq P_p(\{x \rightsquigarrow y\}) \cdot P_p(\{y \rightsquigarrow \infty\}) \\ &\quad \text{car } \{x \rightsquigarrow y\} \text{ et } \{y \rightsquigarrow \infty\} \text{ sont croissants donc l'inégalité FKG s'applique} \\ &= \underbrace{P_p(\{x \rightsquigarrow y\})}_{>0} \cdot \theta_y(p) \end{aligned}$$

Puis $\theta_x(p) = 0 \iff \theta_y(p) = 0$ donc θ_x et θ_y s'annulent mutuellement. □



Proposition

La probabilité de percolation sur un réseau infini vérifie

$$\Pi(p) = \begin{cases} 0 & \text{si } p \leq p_c \\ 1 & \text{si } p > p_c \end{cases}$$

d'où l'appellation "transition de phase".

Preuve (transition de phase)

L'événement $\bigcup_{x \in S} \{x \rightsquigarrow \infty\}$ est un événement de queue et donc, d'après la loi du zéro-un de Kolmogorov

$$\Pi(p) \in \{0, 1\}.$$

Pour $p \in [0; p_c]$:

$$\Pi(p) \leq \sum_{x \in S} \theta_x(p) = 0 \quad \text{d'après l'inégalité de Boole}$$

d'où $\Pi(p) = 0$.

Pour $p \in]p_c; 1]$:

$$\{y \rightsquigarrow \infty\} \subset \bigcup_{x \in S} \{x \rightsquigarrow \infty\}.$$

Donc par croissance de P_p ,

$$0 < P_p(\{y \rightsquigarrow \infty\}) = \theta_y(p) \leq P_p(\bigcup_{x \in S} \{x \rightsquigarrow \infty\}) = \Pi(p)$$

et donc $\Pi(p) = 1$.

□

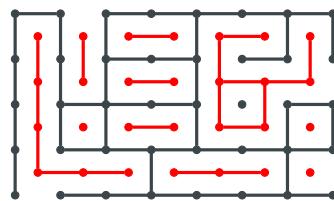
Proposition HARRIS-KESTEN (1960, 1980)

La probabilité critique pour une percolation de lien dans un réseau carré en dimension 2 vaut $p_c = \frac{1}{2}$.

Remarque : pour une percolation de site dans un même réseau, $p_c \approx 0,5928$.

Preuve (valeur de la probabilité critique)

Prenons pour convention que si une arête est ouverte dans le graphe, elle est fermé dans le dual et inversement.

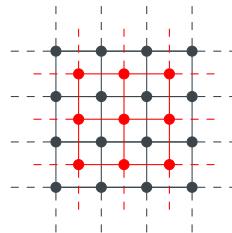


Exemple

Avec p_c^* la probabilité critique du graphe dual, il paraît "intuitivement" que

$$p_c^* = 1 - p_c.$$

Ensuite, le graphe dual de \mathbb{L}_2 n'est autre que lui-même :



Ainsi $p_c = \frac{1}{2}$.

□

4 Un joli résultat

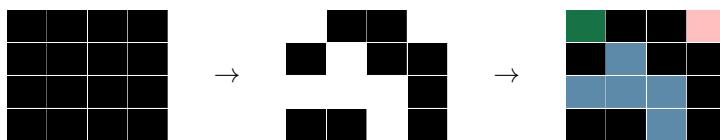
Théorème

Pour $p > p_c$, il existe presque sûrement un unique amas infini sur \mathbb{L}_2 .

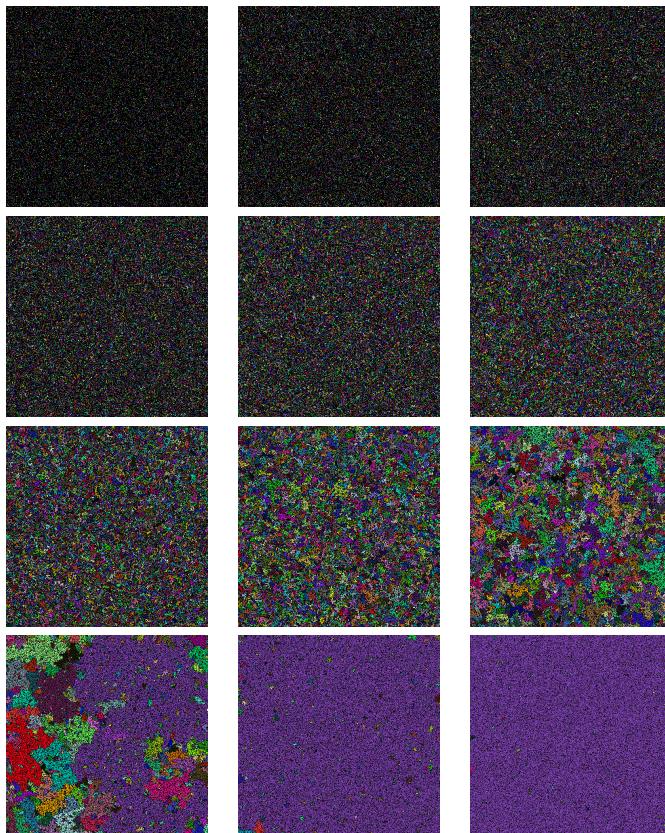
La modélisation suivante permet de mettre en exergue ce résultat :

on représente ici ce qu'on appellera plus tard une percolation de site : on remplit une grille de 500 par 500 de cases colorées avec une probabilité p ou bien noires avec une probabilité $1 - p$.

Ensuite, on recherche tous les amas et on leurs donne des couleurs distinctes.



En faisant varier la probabilité p en commençant à 0,15 avec un pas de 0,05, on observe ces changements :



[Voir algorithme](#)



5 Amas infini (complément?)

Dans cette partie, nous allons démontrer trois théorèmes concernant les amas infinis de certains graphes bien particuliers.

Définitions

Pour (G, \star) un groupe et S une partie génératrice finie de G , on appelle **graphe de Cayley** le graphe $\Gamma(G, S)$ (que l'on notera Γ par abus) tel que chaque sommet représente un élément de G et deux sommets g_1 et g_2 sont liés si et seulement si il existe s un élément de S tel que $g_1 = s \star g_2$.

Un graphe Γ est dit **transitif** si pour tout sommet v , il existe un automorphisme de graphe f tel que $f(v) = v$.

Autrement dit, ce graphe est le même sous tous les angles.

Soit Γ un graphe de Cayley sur lequel on opère une percolation de Bernoulli de paramètre p . On note Γ_p le sous graphe obtenu.

On appelle **percolation sur le sous graphe** $H \subset \Gamma$ le sous graphe de Γ_p restreint aux sommets de H , que l'on note $X(H)$.

Enfin, pour Γ un graphe de Cayley, v un sommet de ce graphe et $k \in \mathbb{N}^*$, on appelle **boule ouverte de centre v et de rayon k** l'ensemble

$$\mathcal{B}(v, k) := \{\omega \in \Gamma \mid |v \rightsquigarrow w| \leq k\}$$

où $|\cdot|$ désigne le plus court chemin.

On appelle **jeu d'arc** associé l'ensemble

$$\mathcal{B}_E(v, k) := \{e \in \mathcal{B}(v, k)^2\}.$$

On remarque que l'on a alors $\lim_{k \rightarrow \infty} \mathcal{B}_E(v, k) = \Gamma$.

Proposition

Soit $\Gamma(G, S)$ un graphe de Cayley infini, connexe et transitif sur lequel on réalise une percolation de Bernoulli de paramètre p .

Le nombre d'amas infini(s) de Γ est de 0, 1 ou ∞ .

Voir preuve



Définitions

Soit $\Gamma(G, S)$ un graphe de Cayley et $H \subset \Gamma$. On définit la *frontière* de H comme suit :

$$\partial H := \{(u, v) \in H \times \Gamma \setminus H\}.$$

Un graphe de Cayley $\Gamma(G, S)$ est dit *moyennable* si et seulement si

$$h(G) := \inf_{H \subset \Gamma} \frac{|\partial H|}{|H|} = 0$$

où $|H|$ est le nombre de sommets de H .

Proposition

Soit $\Gamma(G, S)$ un graphe de Cayley connexe, transitif et moyennable.
Alors pour tout $p \in]p_c; 1]$, Γ possède un unique amas infini.

[Voir preuve](#)

Proposition

Pour $p > p_c$, il existe presque sûrement un unique amas infini sur \mathbb{L}_2 .

[Voir preuve](#)

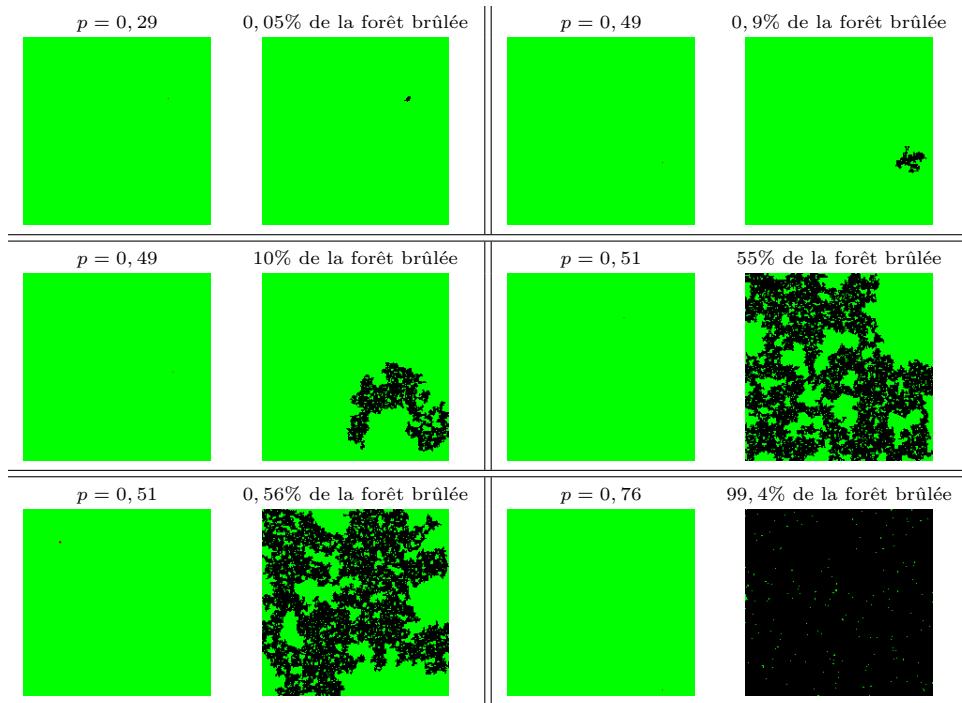
6 Ouverture

Aujourd’hui, la percolation à de nombreuses applications en physique et en informatique, dont :

- les feux de forêt ;
- la modélisation d’épidémies ;
- le passage de l’archipel d’îles au continent (Pierre-Gilles DE GENNES, 1976) ;
- la modélisation de réseaux de communication ;
- l’aimantation (modèle d’ISING)
- la transition sol-gel ;
- la conduite du changement.

Réalisons une modélisation simplifiée de feu de forêt : un arbre au hasard dans une forêt de 200 par 200 prend feu (cause naturelle ou volontaire !) et propage l’incendie ou non dans chacune des quatre directions avec une même probabilité p .

On obtient les résultats suivant :



[Voir algorithme](#)



On remarque tout de même que des hypothèses fortes ont été faites :

- la forêt est considérée comme homogène : les arbres sont également répartis sur \mathbb{Z}^2 , de même hauteur ;
- le feu se propage comme s'il n'y avait pas de vent : il n'y a pas de direction "favorisée" ;
- on considère le temps comme une variable discrète : un arbre qui commence à brûler à l'instant t sera carbonisé à l'instant $t + 1$.

La encore, il est flagrant qu'avant $p = \frac{1}{2}$, la taille de l'amas reste largement inférieure à celle du réseau, tandis que pour des valeurs supérieures, il atteint (plus ou moins rapidement) une taille proportionnelle à celle du réseau. Par exemple pour $p = 0,76$, l'amas est presque de la même taille (cela est moins vrai pour $p = 0,51$ ce qui s'explique par le fait que la grille n'est pas assez grande).



7 Annexes

7.1 Démonstrations

Preuve (loi du zero-un de Kolmogorov)

[Retour à l'énoncé](#)

Soit $B \in \mathcal{B}_\infty$. On considère la classe monotone des évènements indépendants de B :

$$\mathcal{M} = \{A \in \mathcal{A} \mid P(A \cap B) = P(A)P(B)\}.$$

Montrons que $\mathcal{B}_\infty \subset \mathcal{M}$, ainsi on aura B indépendant de lui même, i.e. $P(B) = P(B \cup B) = P(B)^2$ et $P(B) \in \{0, 1\}$.

On pose $\mathcal{C}_n := \sigma(\mathcal{A}_0, \dots, \mathcal{A}_n)$.

Les tribus de départ étant indépendantes, on a pour tout $n \in \mathbb{N}$ que \mathcal{C}_n et \mathcal{B}_{n+1} le sont aussi.

Par conséquent les tribus \mathcal{C}_n et \mathcal{B}_∞ sont indépendantes pour tout $n \in \mathbb{N}$.

On en déduit

$$\forall A \in \bigcap_{n=1}^{\infty} \mathcal{C}_n \quad \forall B \in \mathcal{B}_\infty \quad P(A \cap B) = P(A)P(B).$$

Ceci entraîne que $\mathcal{C}_\infty := \bigcap_{n=1}^{\infty} \mathcal{C}_n \subset \mathcal{M}$ et par théorème des classes monotones $\sigma(\mathcal{C}_\infty) = \mathcal{M}(\mathcal{C}_\infty) \subset \mathcal{M}$.

D'autre part,

$$\forall k \in \mathbb{N} \quad \mathcal{A}_k \subset \mathcal{C}_k \subset \mathcal{C}_\infty \subset \sigma(\mathcal{C}_\infty)$$

donc

$$\forall n \in \mathbb{N} \quad \mathcal{B}_n = \sigma(\mathcal{A}_k \mid k \geq n) \subset \sigma(\mathcal{C}_\infty)$$

i.e.

$$\mathcal{B}_\infty \subset \mathcal{M}.$$

□

Preuve (inégalité FKG)

[Retour à l'énoncé](#)

Soient A et B idoines.

Déjà, l'application $(\omega, \omega') \mapsto (\mathbb{1}_A(w) - \mathbb{1}_A(w'))(\mathbb{1}_B(w) - \mathbb{1}_B(w'))$ définie sur Ω^2 où $\Omega = \{0, 1\}^{\mathbb{Z} \times \mathbb{Z}}$ est positive.

En effet,

si $(\omega, \omega') \in A^2$ (ou $(\omega, \omega') \in B^2$), l'image est nulle,
si $(\omega, \omega') \in A \times \Omega \setminus A$

□



Preuve (quantification du nombre d'amas infini(s))

[Retour à l'énoncé](#)

Notons N le nombre d'amas infinis du sous graphe $\Gamma_p \subset \Gamma$.

Posons

$\forall n \in \mathbb{N} \quad D_n : \text{« } \Gamma_p \text{ possède } n \text{ composantes connexes ouvertes infinis »}.$

Supposons par l'absurde qu'il existe un entier naturel n tel que $P(D_n) \in]0; 1[$.

$$\text{Posons } I_{n,v,k} := \begin{cases} 0 & \text{si } P(D_n|X(\mathcal{B}_E(v,k))) \leq \frac{1}{2} \\ 1 & \text{si } P(D_n|X(\mathcal{B}_E(v,k))) > \frac{1}{2} \end{cases} \text{ et } I_{D_n} := \begin{cases} 0 & \text{si } D_n \\ 1 & \text{sinon} \end{cases}.$$

Soit v un sommet de Γ .

Comme $\lim_{k \rightarrow \infty} X(\mathcal{B}_E(v,k)) = \Gamma_p$, alors $\lim_{k \rightarrow \infty} I_{n,v,k} = I_{D_n}$.

Prenons $(w_k)_{k \in \mathbb{N}^*}$ une suite de sommets tels que

$$\forall k \in \mathbb{N}^* \quad d(w_k, v) \geq 2k.$$

On a

$$\mathcal{B}_E(v, k) \cap \mathcal{B}(v, k) = \emptyset \tag{1}$$

Ensuite, comme Γ est transitif, $I_{n,v,k}$ et $I_{n,w_k,k}$ ont la même distribution et donc $\lim_{k \rightarrow \infty} I_{n,w_k,k} = I_{D_n}$.

Ainsi,

$$\lim_{k \rightarrow \infty} P(I_{n,w_k,k} = I_{n,v,k} = I_{D_n}) = 1 \tag{2}$$

Cependant, d'après (1), $I_{n,w_k,k}$ et $I_{n,v,k}$ sont indépendants (l'état de n'importe quelle arête de Γ est indépendante des autres), alors

$$\begin{aligned} P(I_{n,w_k,k} = 1 - I_{n,v,k} = 1) &= P((I_{n,w_k,k} = 1) \cap (I_{n,v,k} = 0)) \\ &= P(I_{n,w_k,k} = 1)P(I_{n,v,k} = 0) \end{aligned}$$

donc

$$\lim_{k \rightarrow \infty} P(I_{n,w_k,k} = 1 - I_{n,v,k} = 1) = \underbrace{P(D_n)}_{>0} \underbrace{(1 - P(D_n))}_{>0} > 0.$$

Cela contredit (2) ce qui est absurde : l'affirmation initiale est fausse et pour tout entier naturel n , $P(D_n) \in \{0, 1\}$.

Supposons maintenant qu'il existe n fini et supérieur à 2 tel que $P(D_n) = 1$. Comme Γ connexe, il existe $k \in \mathbb{N}^*$ et v un sommet tel que la probabilité que $\mathcal{B}(v, k)$ intersecte chacun des n amas soit strictement positive.

De plus, comme $\mathcal{B}(v, k)$ est de cardinal fini, il existe une probabilité non nulle qu'il existe des chemins de $\mathcal{B}(v, k)$ reliant les amas infinis.

On a alors $P(N = 1) > 0$ ce qui contredit $P(D_n) = 1$ et aboutit donc à une absurdité : on a bien que N prend pour valeur 0, 1 ou ∞ .

□



Preuve (unicité de l'amas infini d'un certain type de graphe)

[Retour à l'énoncé](#)

Soit $p \in]p_c; 1]$ le paramètre de notre percolation (ainsi il existe un cluster infini dans Γ).

Un sommet $v \in \Gamma$ est appelé une **trifurcation** si et seulement si v fait partie d'un amas infini, v possède seulement 3 sommets voisins et enlever v de Γ sépare l'amas en trois amas infinis disjoints.

Supposons que Γ possède au moins une trifurcation. Sinon, la probabilité qu'un sommet v soit une trifurcation est nulle.

Soit $A \subset \Gamma$ un ensemble fini de trifurcations toutes contenues dans le même amas C . $v \in A$ est appelé **membre extérieur** si au moins deux des amas infinis créés si l'on retire v ne contiennent aucun éléments de A .

On a que A contient des membres extérieurs.

En effet, prenons $v_1 \in A$ et supposons que v_1 n'en est pas un. Alors $C \setminus \{v_1\}$ est réunion de trois amas infinis disjoints dont deux intersectent A . Soient v_2 et v_3 de tels éléments. Supposons que v_3 n'est pas non plus membre extérieur, alors $C \setminus \{v_3\}$ est aussi réunion de trois amas infinis disjoints dont l'un contient v_1 et v_2 et au moins l'un des deux autres amas contient un $v_4 \in A$. Maintenant, si v_4 n'est pas membre extérieur, le retirer nous donne un amas contenant v_1 , v_2 et v_3 et au moins un autre contenant un v_5 .

Or comme A est fini, ce processus s'arrête pour un certain $v_n \in A$. De plus, $C \setminus \{v_n\}$ est formé de la réunion de trois amas infinis dont l'un contient v_1, \dots, v_{n-1} et les deux autres n'intersectent pas A . C'est pourquoi A contient un membre extérieur.

On peut montrer par récurrence qu'avec un amas infini C , retirer $j \in \mathbb{N}^*$ trifurcations nous donne $j + 2$ amas infinis disjoints.

En effet, en retirer une nous donne trois amas infini disjoints. Supposons le résultat vrai pour $j \in \mathbb{N}^*$ et prenons $T \subset C$ un ensemble de $j + 1$ trifurcations. Soit v un membre extérieur de T , par hypothèse $C \setminus \{T \setminus \{v\}\}$ est formé de $j + 2$ amas infinis. Comme v est membre extérieur, $C \setminus \{v\}$ est réunion de trois amas infinis disjoints dont un au moins intersecte T . Alors, retirer v de $C \setminus \{T \setminus \{v\}\}$ crée un amas infini disjoint supplémentaire, ce qui en fait $j + 3$ au total et prouve l'hypothèse de récurrence au rang $j + 1$.

Soit $W \subset \Gamma$ un sous graphe fini contenant $j \in \mathbb{N}^*$ trifurcations d'amas infini C . Alors C doit intersecte ∂W sur au moins $j + 1$ sommets. Alors W ne peut contenir plus de $|\partial W| - 2$ trifurcations.

Soit p_t la probabilité qu'un sommet soit une trifurcation. Comme Γ est transitif, p_t dépend d'aucun sommet en particulier. De plus, en posant $T(W)$ le nombre de trifurcation de W , alors on a que $T(W) \leq |\partial W| - 2$ et $E(T(W)) = p_t |W|$.

De ce fait,

$$p_t = \frac{E(T(W))}{|W|} \leq \frac{|\partial W| - 2}{|W|}.$$

De plus, Γ est moyennable par hypothèse donc on peut rendre la partie droite de l'équation aussi petite que l'on veut par choix de W . Cela implique que $p_t = 0$ et donc aucun sommet n'est trifurcation.



Maintenant supposons par l'absurde qu'il existe une infinité d'amis infinis. On peut trouver un sous graphe fini W connexe tel qu'au moins trois amas infinis disjoints intersectent W . Comme W fini, il existe un nombre fini de percolations sur W et n'importe laquelle à une probabilité non nulle.

Alors, comme au moins trois amas infinis disjoints intersectent W , l'événement « il existe des chemins dans W depuis chacun des trois amas qui s'intersectent en un seul point » est de probabilité non nulle. Cette événement crée une trifurcation, donc la probabilité d'existence d'une trifurcation est non nulle. Ainsi $p_t > 0$ sur Γ ce qui est absurde car contredit l'hypothèse initiale.

Ainsi, on a prouvé par l'absurde qu'avec $p \in]p_c; 1]$, il ne peut exister qu'un unique amas infini.

□

Preuve (unicité de l'amas infini sur \mathbb{L}_2)

[Retour à l'énoncé](#)

Soit $\varepsilon \in \mathbb{R}_+^*$ et $H = \{-n, n\} \times \{-n, n\}$ la boîte de côté n et de centre l'origine. On a alors $|H| = 4n^2$ et $|\partial H| = 4(2n - 1)$.

Si l'on prend $n > \frac{1}{\varepsilon} \pm \sqrt{\frac{1}{\varepsilon^2} - \frac{1}{\varepsilon}}$, alors

$$\frac{|H|}{|\partial H|} = \frac{2n - 1}{n^2} < \varepsilon$$

c'est à dire que l'on peut rendre $\frac{|H|}{|\partial H|}$ aussi petit que l'on veut.

Alors $h(\mathbb{L}_2) = 0$ donc \mathbb{L}_2 est moyennable et d'après le théorème précédent, s'il existe, l'amas infini de \mathbb{L}_2 est unique.

□



7.2 Programmes des modélisations

Code de l'algorithme modélisant une roche poreuse

[Voir la modélisation](#)

```
1 # Date : 14/07/2024 - 16/07/2024
2 # Balthazar RICHARD
3 # Analyses de percolations dans des réseaux carrés aléatoires
4 # modélisant l'écoulement d'eau dans une roche poreuse ou bien un
5 # percolateur de café.
6
7
8 import random
9 import matplotlib.pyplot as plt
10 import numpy as np
11 import os
12 os.system("clear")
13
14
15 # Modélisation de la recherche de la composante connexe dans un
16 # réseau carre aléatoire de taille n*m
17
18 def percolation(n, m, p, r=0) :
19
20     # Modélisation du réseau
21     grid = [list() for i in range(2*n+1)]
22     q = 0
23     for i in range(2*n+1) :
24         grid[i] = [list() for j in range(2*m+1)]
25         if i%2!=0 :
26             grid[i][0].append([[0, 0], [n-i//2, n-i//2-1]])
27             grid[i][0].append("darkseagreen")
28         else :
29             grid[i][0] = False
30         for j in range(1, 2*m+1) :
31             q = random.randint(1,100)/100
32             grid[i][j] = list()
33             if (i%2==1 and j%2==1) or (i%2!=1 and j%2!=1) :
34                 grid[i][j] = False
35             else :
36                 # Vertical
37                 if i%2==0 and j%2!=0 :
38                     grid[i][j].append([[j//2, j//2+1], [n-i//2, n
-i//2]])
39                 # Horizontal
40                 elif i%2!=0 and j%2==0 :
41                     grid[i][j].append([[j//2, j//2], [n-i//2, n-i
//2-1]])
42                 if q<p or p==1 :
43                     grid[i][j].append("darkseagreen")
44                 else :
45                     grid[i][j].append("white")
```



```
46
47     plt.plot([0, 0, m, m, m, m, 0], [n, n, n, n, 0, 0, 0], ,
48     gainsboro') # Cadre du réseau
49
50     for i in range(2*n+1) :
51         for j in range(2*m+1) :
52             if grid[i][j] != False :
53                 if grid[i][j][1] != "white" :
54                     plt.plot(grid[i][j][0][0], grid[i][j][0][1] ,
55                     color=grid[i][j][1])
56
57     # Calcul de la composante connexe
58     # Initialisation :
59     for i in range(2*n+1) :
60         if i%2!=0 :
61             if grid[i][0][1]=="darkseagreen" :
62                 grid[i][0][1] = 'Orange'
63
64     stack = list()
65     visited = [[False for j in range(2*m+1)] for i in range(2*n+1)]
66     stack.append([1, 0])
67     while stack != [] :
68         ind = stack.pop()
69         i = ind[0]
70         j = ind[1]
71         if visited[i][j] == False :
72             if grid[i][j][1] == "darkseagreen" :
73                 grid[i][j][1] = "Orange"
74                 # Vertical
75                 if i%2!=0 and j%2==0 :
76                     if i-1>0 and j-1>0 and grid[i-1][j-1][1]!='
77                     white' :
78                         stack.append([i-1, j-1])
79                     if i-2>0 and grid[i-2][j][1]!='white' :
80                         stack.append([i-2, j])
81                     if i-1>0 and j+1<2*m+1 and grid[i-1][j+1][1]!='
82                     white' :
83                         stack.append([i-1, j+1])
84                     if i+1<2*n+1 and j+1<2*m+1 and grid[i+1][j
85                     +1][1]!='white' :
86                         stack.append([i+1, j+1])
87                     if i+2<2*n+1 and grid[i+2][j][1]!='white' :
88                         stack.append([i+2, j])
89                     if i+1<2*n+1 and j-1>0 and grid[i+1][j-1][1]!='
90                     white' :
91                         stack.append([i+1, j-1])
92                         # Horizontal
93                     elif i%2==0 and j%2!=0 :
94                         if j-2>0 and grid[i][j-2][1]!='white' :
95                             stack.append([i, j-2])
```



```
90             if i-1>=0 and j-1>=0 and grid[i-1][j-1][1]!='white' :
91                 stack.append([i-1, j-1])
92             if i-1>=0 and j+1<2*m+1 and grid[i-1][j+1][1]!='white' :
93                 stack.append([i-1, j+1])
94             if i+1<2*n+1 and j-1>=0 and grid[i+1][j-1][1]!='white' :
95                 stack.append([i+1, j-1])
96             if i+1<2*n+1 and j+1<2*m+1 and grid[i+1][j+1][1]!='white' :
97                 stack.append([i+1, j+1])
98             if j+2<2*m+1 and grid[i][j+2][1]!='white' :
99                 stack.append([i, j+2])
100    visited[i][j] = True
101
102 # Donnée de l'atteinte du bord droit par la composante connexe
103 goal = False
104 for i in range(2*n+1) :
105     if grid[i][2*m] != False :
106         if grid[i][2*m][1] == "Orange" :
107             goal = True
108
109 # Calcul de la profondeur de la composante connexe
110 dept = 0
111 for j in range(2*m+1) :
112     for i in range(2*n+1) :
113         if grid[i][j] != False :
114             if grid[i][j][1] == "Orange" :
115                 dept = j//2
116
117 # Affichages
118 print("Probabilité d'ouverture d'un lien p =", p)
119 print("Profondeur de la composante connexe ouverte :", dept,
      "\n\n")
120 title = "Probabilité d'ouverture d'un lien p = " + str(p)
121 plt.axis(False)
122 plt.title(title)
123 nom = 'percolation_eau_0'+str(r)+'.svg'
124 plt.savefig(nom)
125
126 plt.pause(0.1)
127 for j in range(2*m+1) :
128     for i in range(2*n+1) :
129         if grid[i][j] != False :
130             if grid[i][j][1] == "Orange" :
131                 plt.plot(grid[i][j][0][0], grid[i][j][0][1],
132                          'Orange')
133
134 plt.draw()
```



```
134     plt.pause(1)
135
136     nom2 = 'percolation_eau_'+str(r)+'.svg'
137     plt.savefig(nom2)
138
139     plt.clf()
140     return goal
```



Code de l'algorithme modélisant une percolation de site

[Voir la modélisation](#)

```
1 # Date : 16/07/2024
2 # Balthazar RICHARD
3 # Algorithme modélisant une percolation de site
4
5
6 import random
7 import matplotlib.pyplot as plt
8 import numpy as np
9 from PIL import Image
10 from PIL import GifImagePlugin
11 import os
12 os.system("clear")
13
14
15 # Fonction qui crée un tableau de n couleurs distinctes ni noir,
16 # ni blanche
17
18 def colors_tab(n) :
19     colors = list()
20     for i in range(n) :
21         r = 0
22         g = 0
23         b = 0
24         while (r,g,b) == (0,0,0) or (r,g,b) == (255,255,255) or (r,g,b)
25             == (203,51,255) or (r,g,b) in colors :
26             r = random.randint(0,255)
27             g = random.randint(0,255)
28             b = random.randint(0,255)
29             colors.append((r,g,b))
30
31     return colors
32
33
34 # Modélisation d'une percolation de Bernoulli de paramètre p
35
36 def percolation(n, p, r='') :
37
38     print("\nProbabilité p =", p, "\n")
39
40     # Modélisation du réseau de la forêt
41     grid = [list() for i in range(n)]
42     for i in range(n) :
43         grid[i] = [list() for j in range(n)]
44         for j in range(n) :
45             q = random.randint(1,100)/100
46             if q < p or p==1 :
47                 grid[i][j] = 0
48             else :
49                 grid[i][j] = 1
```



```
48 # Système sans mise en évidence des différents amas
49 im = Image.new('RGB',(n,n))
50 for i in range(n):
51     for j in range(n):
52         if grid[i][j] == 0:
53             im.putpixel((i,j),(203,51,255))
54         else :
55             im.putpixel((i,j),(0,0,0))
56
57 # Recherche des amas
58 clusters = list()
59 in_cluster = [[False for j in range(n)] for i in range(n)]
60 tmp = list()
61
62 stack = list()
63 visited = [[False for j in range(n)] for i in range(n)]
64 for x in range(n) :
65     for y in range(n) :
66         if grid[x][y] == 0 and in_cluster[x][y] == False :
67             stack.append([x,y])
68             tmp.append([x,y])
69             while stack != [] :
70                 ind = stack.pop()
71                 i = ind[0]; j = ind[1]
72                 if visited[i][j] == False :
73                     in_cluster[i][j] = True
74                     # Gauche
75                     if i-1>=0 and grid[i-1][j] == 0 and
76                     in_cluster[i-1][j] == False :
77                         stack.append([i-1, j])
78                         tmp.append([i-1, j])
79
80                     # Droite
81                     if i+1<n and grid[i+1][j] == 0 and
82                     in_cluster[i+1][j] == False :
83                         stack.append([i+1, j])
84                         tmp.append([i+1, j])
85
86                     # Bas
87                     if j-1>=0 and grid[i][j-1] == 0 and
88                     in_cluster[i][j-1] == False :
89                         stack.append([i, j-1])
90                         tmp.append([i, j-1])
91
92                     # Haut
93                     if j+1<n and grid[i][j+1] == 0 and
94                     in_cluster[i][j+1] == False :
95                         stack.append([i, j+1])
96                         tmp.append([i, j+1])
97                         visited[i][j] = True
```



```
95             clusters.append(tmp)
96             tmp = list()
97
98             colors = colors_tab(len(clusters))
99             color = (128,68,181)
100
101            maxi = 0
102            ind_maxi = 0
103            for l in range(len(clusters)) :
104                if len(clusters[l]) > maxi :
105                    maxi = len(clusters[l])
106                    ind_maxi = l
107
108            for l in range(len(clusters)) :
109                while clusters[l] != [] :
110                    ind = clusters[l].pop()
111                    i = ind[0]
112                    j = ind[1]
113                    if l != ind_maxi :
114                        im.putpixel((i,j), colors[l])
115                    else :
116                        im.putpixel((i,j), color)
117
118
119            titre = 'image_'+str(r)+'_'+str(p)+'.png'
120            im.save(titre, 'png')
```



Code de l'algorithme modélisant un feu de forêt.

[Voir la modélisation](#)

```
1 # Date : 16/07/2024
2 # Balthazar RICHARD
3 # Analyses de percolations dans un réseau carre modélisant
4 # un feu de forêt.
5
6
7 import random
8 import matplotlib.pyplot as plt
9 import numpy as np
10 from PIL import Image
11 from PIL import GifImagePlugin
12 import os
13 os.system("clear")
14
15
16 # Modélisation de la propagation du feu.
17
18 def percolation(n, m, p, r=''):
19
20     print("\nProbabilité de transmission du feu d'un arbre à l' autre :", p)
21
22     # Modélisation du réseau de la forêt
23     grid = [list() for i in range(n)]
24     for i in range(n) :
25         grid[i] = [list() for j in range(m)]
26         for j in range(m) :
27             grid[i][j] = 0
28     burning = list()
29     neighbor = list()
30     burnt = list()
31
32     # Forêt sans feu
33     im = Image.new('RGB',(n,m))
34     for i in range(n):
35         for j in range(m):
36             im.putpixel((i,j),(0,255,0))
37     im.save('foret_0.png', 'png')
38
39     # Premier arbre en feu
40     x = random.randint(0,n-1)
41     y = random.randint(0,m-1)
42     burning.append([x, y])
43     grid[x][y] = 1
44     im.putpixel((x,y),(194,24,7))
45     im.save('foret_1.png', 'png')
46
47
48     # Percolation
```



```
49     alpha = 2
50     while burning != [] or neighbor != [] :
51         while burning != [] :
52             ind = burning.pop(0)
53             x = ind[0]; y = ind[1]
54
55             # Gauche
56             if x-1>=0 and grid[x-1][y] == 0:
57                 q = random.randint(1, 100)/100
58                 if q<p or p==1 :
59                     neighbor.append([x-1, y])
60                     grid[x-1][y] = 1
61                     im.putpixel((x-1,y),(255,0,0))
62
63             # Droite
64             if x+1<n and grid[x+1][y] == 0:
65                 q = random.randint(1, 100)/100
66                 if q<p or p==1 :
67                     neighbor.append([x+1, y])
68                     grid[x+1][y] = 1
69                     im.putpixel((x+1,y),(255,0,0))
70
71             # Bas
72             if y-1>=0 and grid[x][y-1] == 0:
73                 q = random.randint(1, 100)/100
74                 if q<p or p==1 :
75                     neighbor.append([x, y-1])
76                     grid[x][y-1] = 1
77                     im.putpixel((x,y-1),(255,0,0))
78
79             # Haut
80             if y+1<m and grid[x][y+1] == 0:
81                 q = random.randint(1, 100)/100
82                 if q<p or p==1 :
83                     neighbor.append([x, y+1])
84                     grid[x][y+1] = 1
85                     im.putpixel((x,y+1),(255,0,0))
86
87             burnt.append(ind)
88             grid[x][y] = 2
89             im.putpixel((x,y),(0,0,0))
90
91             burning = neighbor
92             neighbor = list()
93             titre = 'foret_+'+str(alpha)+'.png'
94             im.save(titre, 'png')
95             alpha += 1
96
97             alive = 0
98             burnt2 = 0
99             for i in range(n) :
```



```
100     for j in range(m) :
101         if grid[i][j] == 0 :
102             alive += 1
103         elif grid[i][j] == 2 :
104             burnt2 += 1
105
106     print("Au total,", burnt2, "arbre(s) brûlé(s) et", alive, "
107     arbre(s) encore vert.\n")
108
109 # Création d'un GIF
110
111 imgs = list()
112 im1 = Image.open('foret_0.png')
113
114 for i in range(1, alpha) :
115     noms = 'foret_'+str(i)+'.png'
116     imgs.append(Image.open(noms))
117 im1.save('percolation_foret_'+str(r)+'.gif', save_all=True,
append_images=imgs, duration=alpha/3, loop=0, optimize=True,
include_color_table=True, interlace=True)
```



7.3 Sources

- [1] BERGLUND (N) : [Probabilités et Physique Statistique, 28 juin 2013.](#)
- [2] SPECTRAL COLLECTIVE : [Percolation: a Mathematical Phase Transition, 9 août 2022.](#)
- [3] JARRY (M), CAMPET (P) : [Modélisation d'un feu de forêt, Juin 2010.](#)
- [4] LESBRE (D) : [Rapport_TIPE_Percolation, 2018.](#)
- [5] RUCH (J), CHABANOL (M-L) : [RAPPELS de PROBABILITÉ, 2013, 2014.](#)

Source pour le complément :

- [6] HECKEL (R) : [PERCOLATION AND THE EXISTENCE OF THE INFINITE OPEN CLUSTER, 2008.](#)