

TIPE : Etude de la propagation d'incendies en milieu urbain

Aimine MEDDEB - Numéro de candidat : 11314

2022 - 2023



1 Position du problème

2 Modélisation du phénomène

3 Simulation expérimentale

4 Théorie mathématiques

5 Analyse des résultats

6 Annexe

Enjeux



Ville de New York.



Projet New Murabba en Arabie Saoudite

Enjeux



- Le dernier rapport du GIEC prévoit une intensification des incendies comme c'est déjà le cas dans les forêts.
 - La chaleur des villes permet en plus un départ de feu facile.

Intérêt de la modélisation

- Phénomène trop complexe à modéliser dans son entièreté.
- Trop de facteur interviennent.
- Dépend de la géographie de la ville.

On adopte un modèle mathématique simple et générale comme première approximation.

Modélisation en deux dimensions

- Grille régulière où les sommets sont les bâtiments

Modélisation en deux dimensions

- Grille régulière où les sommets sont les bâtiments
 - la transmission de l'incendie entre deux bâtiment est représenté par une arrête.

Modélisation en deux dimensions

- Grille régulière où les sommets sont les bâtiments
 - la transmission de l'incendie entre deux bâtiment est représenté par une arrête.
 - Si le feu a pu se propager l'arrête est fermé, et ouverte sinon.

Modélisation en deux dimensions

- Grille régulière où les sommets sont les bâtiments
- la transmission de l'incendie entre deux bâtiment est représenté par une arrête.
- Si le feu a pu se propager l'arrête est fermé, et ouverte sinon.
- Chaque arrête a une probabilité p d'être fermée et $1-p$ d'être ouverte .

Modélisation en deux dimensions

- Grille régulière où les sommets sont les bâtiments
- la transmission de l'incendie entre deux bâtiment est représenté par une arrête.
- Si le feu a pu se propager l'arrête est fermé, et ouverte sinon.
- Chaque arrête a une probabilité p d'être fermée et $1-p$ d'être ouverte .
- On obtient finalement un graphe dont les composantes connexes représentent l'ensemble des bâtiments brûlés si un incendie se déclarait dans l'un de ces immeubles.

Exemple

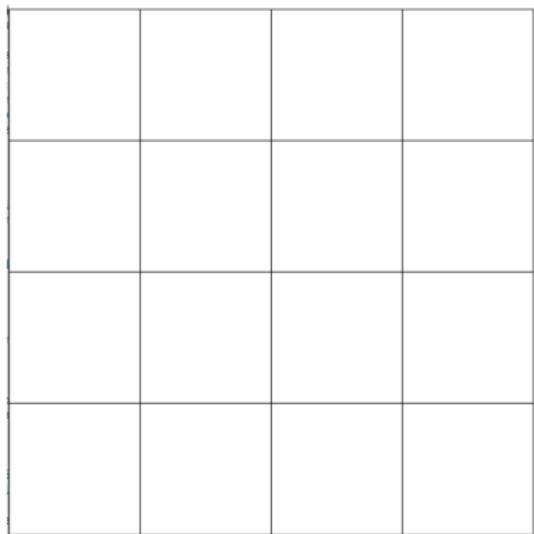


Figure: Grille initiale

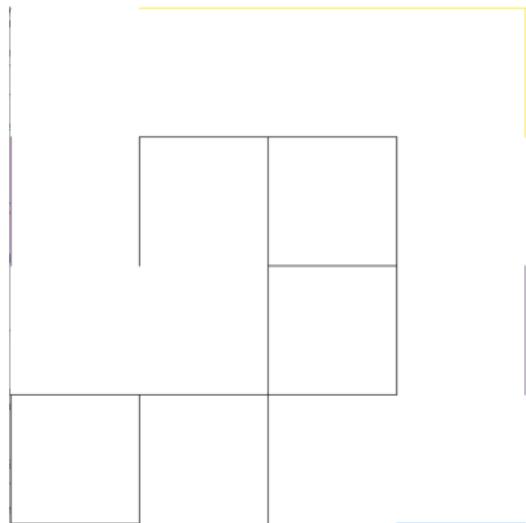


Figure: Grille après percolation

Utilisation

- Notre objectif est de minimiser la taille de la plus grande composante connexe.
- Nous allons étudier la taille de cette dernière en fonction de p .
- Dans le modèle mathématiques, on suppose une grille de taille infini. Nous allons donc réaliser des simulations sur des grilles de grandes tailles et analyser la pertinence de cette hypothèse.

Dimension 2

Attentes de la modélisation

- Pour des grandes valeurs de p (proche de 1) : une seule grande composante connexe.
 - Pour des petites valeurs de p (proche de 0) : plusieurs petites composantes connexes.

Dimension 2

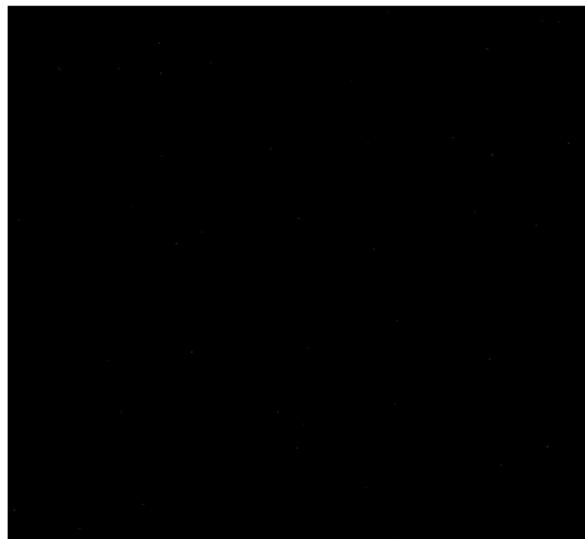
Exemples



Graphe $(\mathbb{Z}^2, \mathbb{E}^2)$ pour $p = 0.2$ et $n = 3000$

Dimension 2

Exemples



Graphe $(\mathbb{Z}^2, \mathbb{E}^2)$ pour $p = 0.8$ et $n = 3000$

Position du problème

Modélisation du phénomène

Simulation expérimentale

Théorie mathématiques

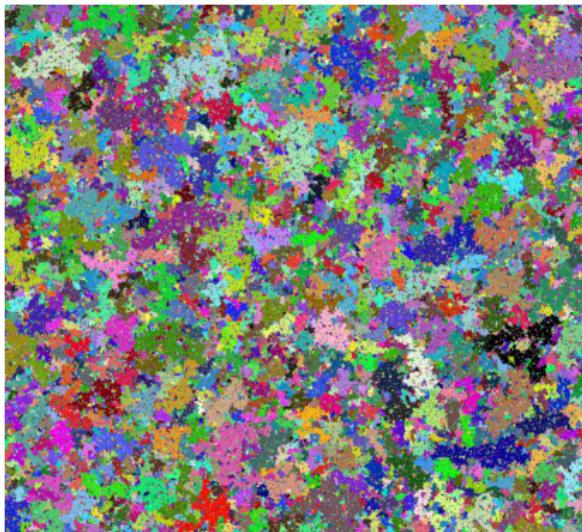
Analyse des résultats

Dimension 2

Transition de phase

Dimension 2

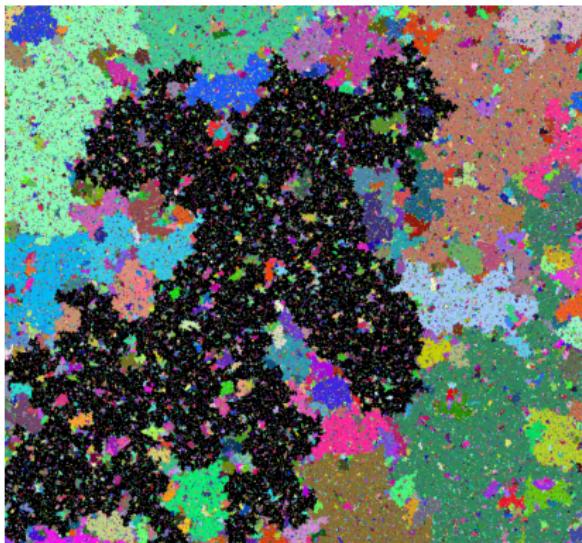
Transition de phase



Graphe $(\mathbb{Z}^2, \mathbb{E}^2)$ pour $p = 0.49$ et $n = 3000$

Dimension 2

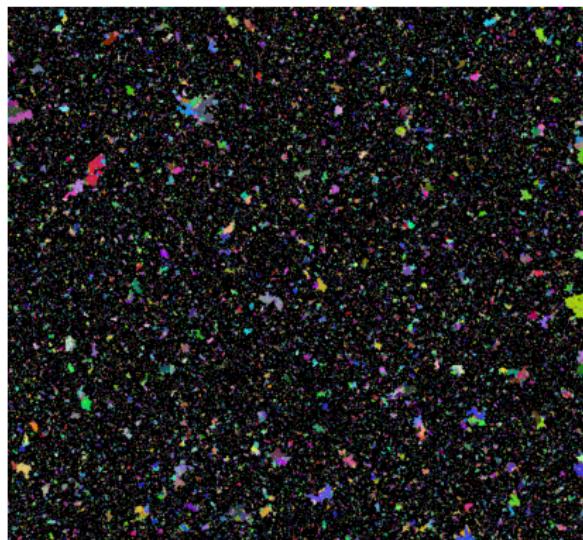
Transition de phase



Graphe $(\mathbb{Z}^2, \mathbb{E}^2)$ pour $p = 0.50$ et $n = 3000$

Dimension 2

Transition de phase



Graphe $(\mathbb{Z}^2, \mathbb{E}^2)$ pour $p = 0.51$ et $n = 3000$

Dimension 2

Transition de phase et probabilité critique

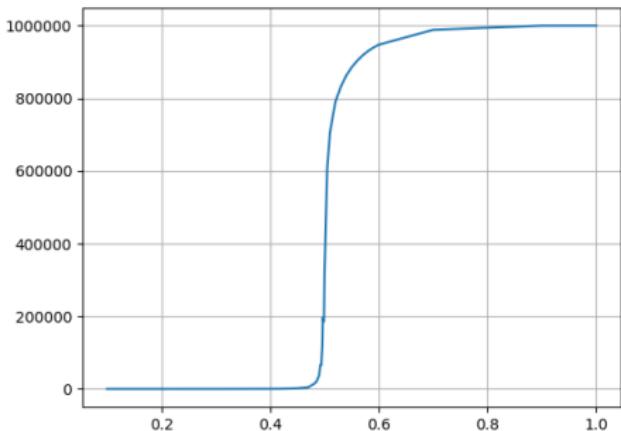


Figure représentant la taille de la plus grande composante connexe d'un graphe de taille 1000x1000 en fonction de p .

Dimension 2

Première conjecture

Conjecture 1

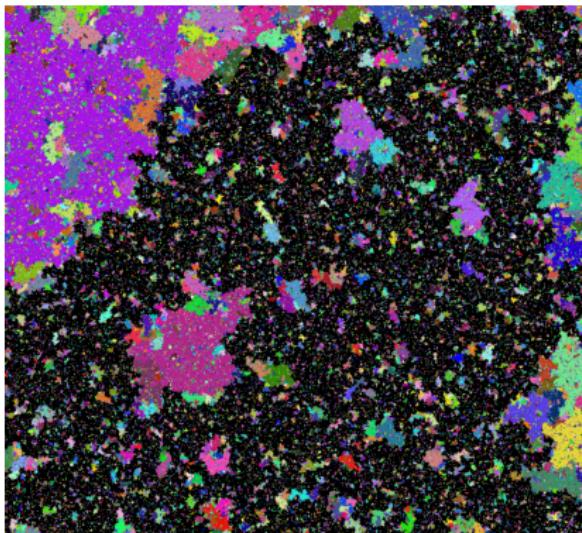
Le modèle contient un phénomène de transition de phase à la probabilité critique $p_c = 0,5$

Étude pour $p < p_c$

- Dans le cadre de notre étude on peut vouloir étudier le comportement de la plus grande composante connexe pour $p < p_c$.
- En effet dans le cadre de la lutte contre un incendie voir comment croit ou décroît la zone potentiellement touché par le feu est intéressant.

Dimension 2

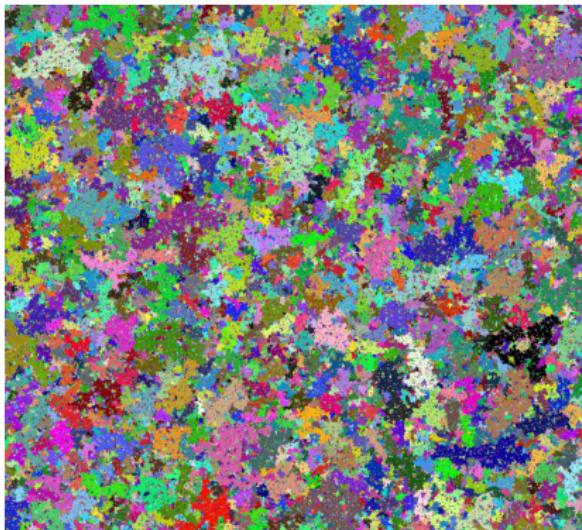
Étude de la décroissance de la taille des composantes connexes



Graphe $(\mathbb{Z}^2, \mathbb{E}^2)$ pour $p = 0.50$ et $n = 3000$

Dimension 2

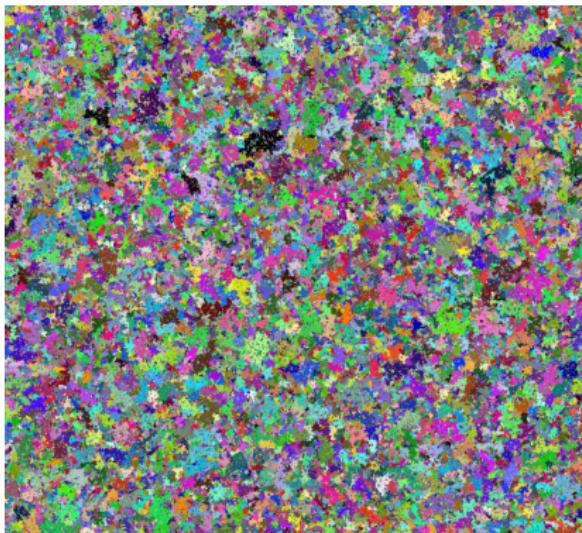
Étude de la décroissance de la taille des composantes connexes



Graphe $(\mathbb{Z}^2, \mathbb{E}^2)$ pour $p = 0.49$ et $n = 3000$

Dimension 2

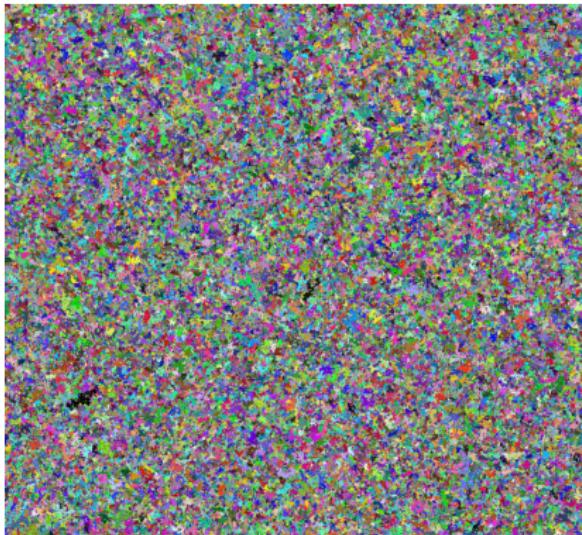
Étude de la décroissance de la taille des composantes connexes



Graphe $(\mathbb{Z}^2, \mathbb{E}^2)$ pour $p = 0.48$ et $n = 3000$

Dimension 2

Étude de la décroissance de la taille des composantes connexes



Graphe $(\mathbb{Z}^2, \mathbb{E}^2)$ pour $p = 0.46$ et $n = 3000$

Dimension 2

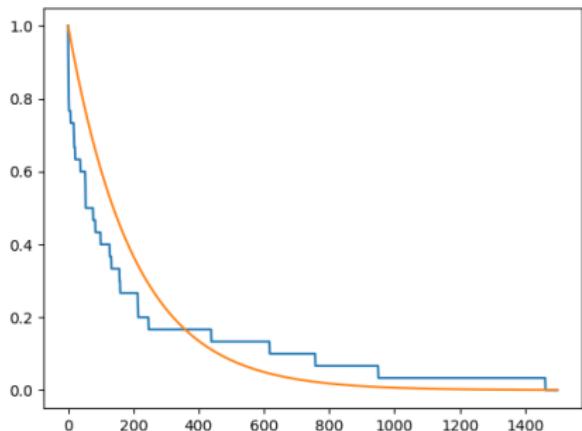
Étude de la décroissance de la taille des composantes connexes



Graphe $(\mathbb{Z}^2, \mathbb{E}^2)$ pour $p = 0.40$ et $n = 3000$

Dimension 2

Analyse de la décroissance



Probabilité que la taille de la plus grande composante connexe soit supérieure à n en fonction de n

Dimension 2

Conjecture

On note $C(x)$ la composante connexe de x pour x un élément de \mathbb{Z}^2

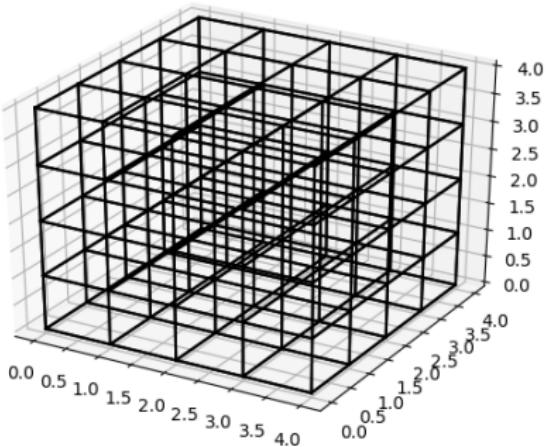
Conjecture 2

Pour $p < p_c$, il existe une constante $\lambda(p)$ telle que:

$$\forall n \in \mathbb{N}, \mathbb{P}(|C(x)| \geq n) \leq \exp(-\lambda(p)n)$$

Dimension 3

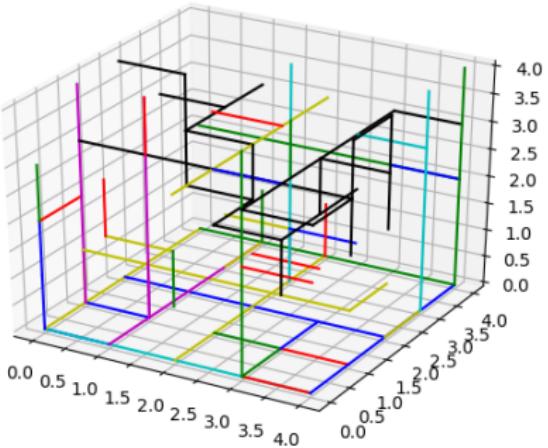
Dimension 3: Modélisation



Graphe $(\mathbb{Z}^3, \mathbb{E}^3)$ pour $n = 5$

Dimension 3

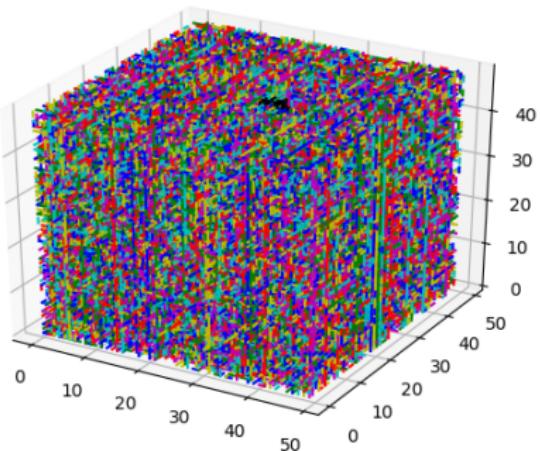
Dimension 3: Modélisation



Graphe $(\mathbb{Z}^3, \mathbb{E}^3)$ percolé pour $n = 5$

Dimension 3

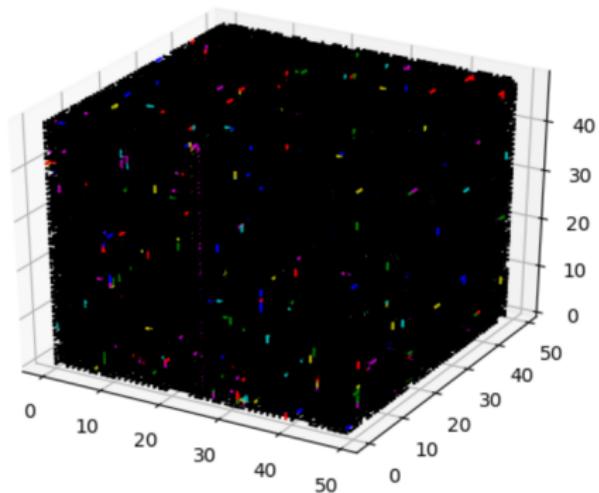
Étude de la percolation en dimension 3



Graphe $(\mathbb{Z}^3, \mathbb{E}^3)$ avec pour $p = 0.2$ et $n = 5$

Dimension 3

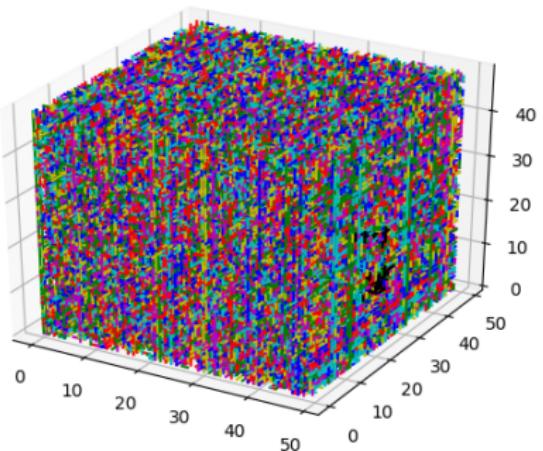
Étude de la percolation en dimension 3



Graphe $(\mathbb{Z}^3, \mathbb{E}^3)$ avec pour $p = 0.4$ et $n = 5$

Dimension 3

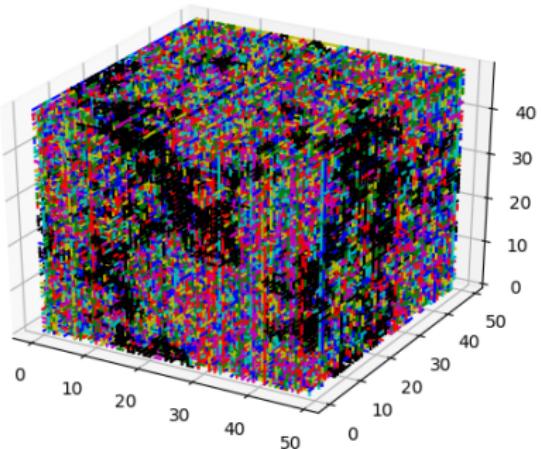
Étude de la transition de phase en dimension 3



Graphe $(\mathbb{Z}^3, \mathbb{E}^3)$ avec pour $p = 0.24$ et $n = 5$

Dimension 3

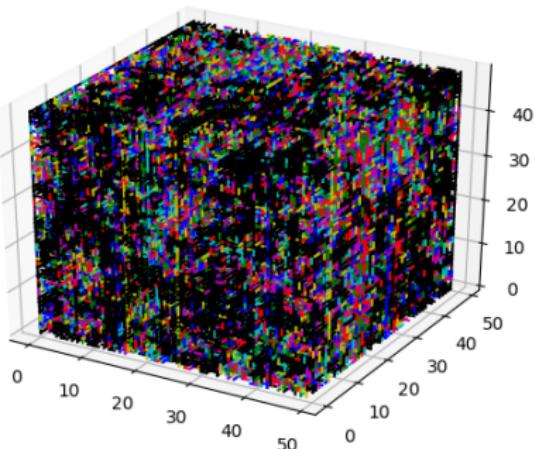
Étude de la transition de phase en dimension 3



Graphe $(\mathbb{Z}^3, \mathbb{E}^3)$ avec pour $p = 0.25$ et $n = 5$

Dimension 3

Étude de la transition de phase en dimension 3



Graphe $(\mathbb{Z}^3, \mathbb{E}^3)$ avec pour $p = 0.26$ et $n = 5$

Dimension 3

Transition de phase en dimension 3

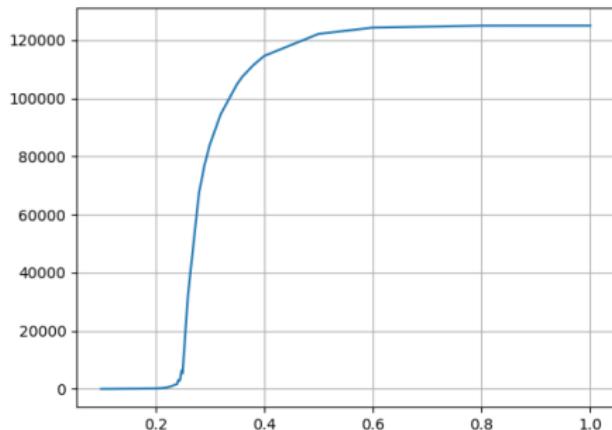


Figure représentant la taille de la plus grande composante connexe d'un graphe de taille 50x50x50 en fonction de p .

Dimension 3

Conjectures en dimension 3

Conjecture

Le modèle de percolation en dimension 3 présente un phénomène de transition de phase avec une probabilité critique $p_c \simeq 0,247$

Position du problème

Modélisation du phénomène

Simulation expérimentale

Théorie mathématiques
● ●
888

Analyse des résultats /

Probabilité critique et transition de phase

Définitions

Probabilité critique et transition de phase

Définitions

Définition

On définit le graphe aléatoire infini $(\mathbb{Z}^2, \mathbb{E}^2)$ avec $\mathbb{E}^2 = \{e = (x, y) \in \mathbb{Z}^2 \times \mathbb{Z}^2, \|x - y\| = 1, \mathbb{P}_p(e) = 1\}$ lorsque \mathbb{P}_p suit la loi de Bernoulli de paramètre $p \in [0, 1]$.

Probabilité critique et transition de phase

Définitions

Définition

On définit le **graphe aléatoire infini** $(\mathbb{Z}^2, \mathbb{E}^2)$ avec

$\mathbb{E}^2 = \{e = (x, y) \in \mathbb{Z}^2 \times \mathbb{Z}^2, \|x - y\| = 1, \mathbb{P}_p(e) = 1\}$ lorsque \mathbb{P}_p suit la loi de Bernoulli de paramètre $p \in [0, 1]$.

Définition

- Dans un tel graphe on dit qu'il y a percolation lorsque il existe une composante connexe de taille infini.
 - On appelle probabilité de percolation en 0 :
 $\theta(p) = \mathbb{P}(\text{Card}(C(0)) = +\infty).$
 - On note $p_c(d) = \sup(\{p \in [0, 1], \theta(p) = 0\})$ que l'on nomme seuil critique de percolation.

Théorème de non-trivialité de p_c

Pour avoir une véritable transition de phase il faut que $p_c \neq 1$ et $p_c \neq 0$

Théorème 1 (preuve en annexe)

$$0 < p_c < 1$$

On connaît aussi le comportement de $\theta(p)$ pour $p > p_c$ et $p < p_c$

Théorème de 2 (preuve en annexe)

Le seuil critique p_c met en évidence une transition de phase :

- si $p < p_c$, alors il n'y a presque sûrement pas percolation
- si $p > p_c$, alors il y a presque sûrement percolation
- si $p = p_c$, on ne peut rien dire a priori

Décroissance exponentielle

Théorème de décroissance exponentielle de la taille des composantes connexes

En conservant les notations introduites précédemment on a :

Théorème 2

Pour $p < p_c$, on dispose de $\lambda(p)$ tel que

Autres résultats

Théorème (Harris, 1960)

$$\theta(p_c) = 0$$

Décroissance exponentielle

Autres résultats

Théorème (Harris, 1960)

$$\theta(p_c) = 0$$

Théorème (Kesten, 1980)

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

Décroissance exponentielle

Recherches récentes

Recherches plus récentes

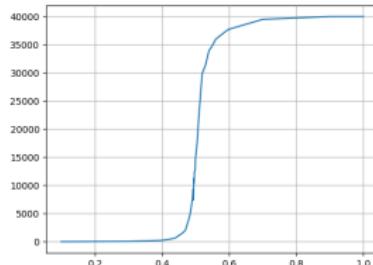
- Extension des résultats en dimension 3
 - Recherche sur des graphes non carré (réseaux triangulaire ou en nid d'abeille)

Cohérence

- Simulations informatiques cohérentes avec la théorie mathématiques

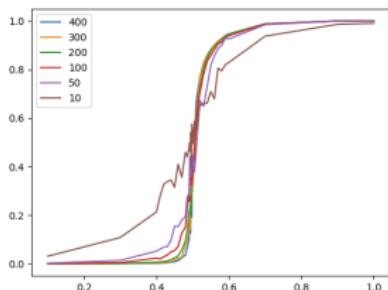
Cohérence

- Simulations informatiques cohérentes avec la théorie mathématiques
- Les simulations peuvent correspondre à la réalité



Taille de la plus grande composante connexe d'un graphe de taille 200×200 en fonction de p

Cohérence



Taille de la plus grande composante connexe en fonction de p pour différente valeur de n .

Résultat important et interprétation

Résultats

- Transition de phase à la probabilité $p_c = 0.5$ en dimension 2 et $p_c \simeq 0.247$.
- Décroissance exponentielle pour $p < p_c$.
- Le couloir représentent les arrêtes.
- Les sites représentent les pièces.
- Les arrêtes ouvertes symbolisent la présence dans le couloir d'un extincteur ou d'une porte coupe-feu.

Conclusion

- On peut ainsi agir sur la valeur de p pour être en dessous de la percolation critique
 - Les zones denses sans protection contre le feu représentent un grand danger en cas d'incendie (bidonville)
 - Du fait de la décroissance exponentielle, on a grandement intérêt à diminuer p en plaçant des dispositifs anti-feu régulièrement

Démonstration du théorème 1 ($0 < p_c < 1$)

Théorème 1

$$0 < p_c < 1$$

Démonstration du théorème 1 ($0 < p_c < 1$)

Théorème 1

$$0 < p_c < 1$$

Preuve en dimension 2 (minoration $p_c > 0$)

Soient $p \in [0, 1]$ et $k \in \mathbb{N}$.

$$\begin{aligned}\theta(p) &= \mathbb{P}(\text{Card}(C(0)) = +\infty) \\ &\leq \mathbb{P}(\text{Card}(C(0)) \geq k) \\ &= \mathbb{P}\left(\bigcup_{\substack{y_0, \dots, y_n \text{ chemin} \\ y_0 = 0}} \{\forall 0 \leq i \leq k-1, y_i \text{ est relié à } y_{i+1}\}\right)\end{aligned}$$



Démonstration du théorème 1 ($0 < p_c < 1$)

Majoration $p_c < 1$

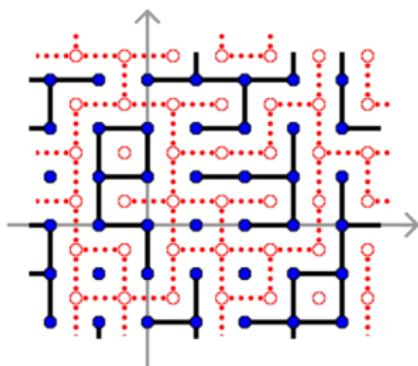
$$\begin{aligned}\theta(p) &\leq \sum_{\substack{y_0, \dots, y_n \text{ chemin} \\ y_0=0}} \mathbb{P}(\{\forall 0 \leq i \leq k-1, y_i \text{ est relié à } y_{i+1}\}) \\ &= p^k \times (\text{nombre de chemins de longueur } k \text{ issus de 0}) \\ &\leq 4 \cdot 3^{k-1} \cdot p^n\end{aligned}$$

$$\text{donc } \forall p < \frac{1}{3}, \ 0 \leq \theta(p) \leq 4p \cdot (3 \times p)^{k-1} \xrightarrow{k \rightarrow \infty} 0,$$

d'où $p_c > \frac{1}{3}$ et donc $p_c > 0$.

Démonstration du théorème 1 $0 < p_c < 1$)

Preuve en dimension 2 (majoration $p_c < 1$)



$\text{Card}(C(0)) < +\infty \Leftrightarrow$ il existe un circuit dans le dual entourant 0



Démonstration du théorème 1 ($0 < p_c < 1$)

$$1 - \theta(p) = \mathbb{P}(\{\text{il existe un circuit dans le dual entourant } 0\})$$

$$\leq \mathbb{P}\left(\bigcup_{y_0^*, \dots, y_{n-1}^*} \{\forall 0 \leq i \leq k-1, y_i^* \text{ est relié à } y_{i+1}^*\}\right)$$

$$\leq \sum_{k \geq 4} \sum_{y_0^*, \dots, y_{n-1}^*} \mathbb{P}(\{\forall 0 \leq i \leq k-1, y_i^* \text{ est relié à } y_{i+1}^*\})$$

$$\leq \sum_{k \geq 4} (1-p)^k \cdot k \cdot (4 \cdot 3^{k-1}) \xrightarrow{p \rightarrow 1} 0$$

donc $\theta(p) \xrightarrow{p \rightarrow 1} 1$ et $\exists p_0 < 1, \forall p > p_0, \theta(p) > 0$ d'où $p_c < 1$.

Démonstration du théorème 2

Théorème 2

Le seuil critique p_c met en évidence une transition de phase :

- si $p < p_c$, alors il n'y a presque sûrement pas percolation
- si $p > p_c$, alors il y a presque sûrement percolation
- si $p = p_c$, on ne peut rien dire

Démonstration du théorème 2

Théorème 2

Le seuil critique p_c met en évidence une transition de phase :

- si $p < p_c$, alors il n'y a presque sûrement pas percolation
- si $p > p_c$, alors il y a presque sûrement percolation
- si $p = p_c$, on ne peut rien dire

Loi du 0-1 de Kolmogorov (non démontrée)

Si $(X_n)_{n \in \mathbb{N}}$ est une suite de variables aléatoires indépendantes, si A est un événement indépendant de tout ensemble fini de variables $(X_i)_{i \in I}$, alors $\mathbb{P}(A) \in \{0, 1\}$.

Démonstration du théorème 2

On rappel qu'il y a percolation lorsqu'il existe une composante connexe de taille infini.

Cela ce représente par l'évènement:

$$(\exists x \in \mathbb{Z}^2, \text{Card}(C(x)) = +\infty)$$

Et pour $p \in [0, 1]$, on note: $\psi(p)$ la probabilité de cet évènement

Démonstration du théorème 2

On applique la loi 0-1 de Kolmogorov à l'évènement "il y a percolation" dans la mesure où, un nombre fini d'arêtes n'a pas d'influence sur le caractère fini ou non de $C(x)$

Donc $\forall p \in [0, 1], \psi(p) \in \{0, 1\}$

- si $p < p_c$, alors $\forall x \in \mathbb{Z}^d, \mathbb{P}(\text{Card}(C(x)) = +\infty) = 0$, donc en particulier :
 $\forall n \in \mathbb{N}, \mathbb{P}(\exists x \in [-n, n]^d, \text{Card}(C(x)) = +\infty) = 0$.

Et par continuité croissante on a :

$$\mathbb{P}(\exists x \in \mathbb{Z}^d, \text{Card}(C(x)) = +\infty) = \mathbb{P}(\bigcup_{n \in \mathbb{N}} (\exists x \in [-n, n]^d, \text{Card}(C(x)) = +\infty)) = 0$$

On a donc $\psi(p) = 0$.

Démonstration du théorème 2

- si $p > p_c$,
alors $\psi(p) \geq \theta(p) > 0$
et donc $\psi(p) = 1$.
- si $p = p_c$, alors la loi du 0-1 donne $\psi(p) = 0$ ou 1 mais on ne peut pas déterminer la valeur simplement

Réalisation du graphe percolé

```
11 #Les directions possibles dans Z^2
12 dir = [(1, 0), (0, -1), (-1, 0), (0, 1)]
13
14 #Construction de la grille initiale
15 def grille(etat, n):
16     for i in range(n):
17         etat.append([1])
18         for j in range(n):
19             etat[i].append([0, 0, 0, 0, 0])
20 """
21 Un sommet contient l'information de ses 4 arêtes voisine (0 si ouverte, 1 si fermé) et une
22 autre valeur qui servira pour un parcours du graphe plus tard
23 """
24 #Supprime une arête pour un sommet et son homologue
25 def delete(etat, x, y, direction):
26     x2, y2 = dir[direction][0] + x, dir[direction][1] + y
27
28     etat[x][y][direction] = 1
29     etat[x2][y2][(2 + direction) % 4] = 1
30
31 #Réalisation de la percolation
32 def percolateur(etat, n, p):
33     for i in range(n):
34         for j in range(n):
35             for k in range(1, 3): #On parcours le graphe en traitant pour chaque sommet
36                 uniquement les arêtes droite et gauche
37                 if random.random() < p:
38                     delete(etat, i, j, k)
39
40 #Programme qui renvoie un graphe percolé
41 def general(n,p):
42     g=[]
43     grille(g,n)
44     percolateur(g,n,p)
45     return g
```

Analyse des composantes connexes

```

47 ##ANALYSE DES COMPOSANTE CONNEXES
48
49 #Renvoie la liste des arêtes ouverte voisine
50 def voisins(etat, n, x, y):
51     listevoisins = []
52     for direction in range(4):
53         x2, y2 = dir[direction][0] + x, dir[direction][1] + y
54         if (x2 >= 0 and y2 >= 0 and x2 < n and y2 < n):
55             if (etat[x][y][direction] == 1):
56                 listevoisins.append((x2, y2))
57
58     return listevoisins
59 #A partir d'un sommet, et d'un numéro, renvoie la taille de la composante connexe associé et marque
60 #tous les sommets qui en font partie à l'aide d'un parcours en profondeur
61 def marquage(etat, n, x, y, marque):
62     compteur = 0
63     queue = []
64     etat[x][y][4] = marque
65     queue.append((x,y))
66     while queue != []:
67         v = queue.pop(0)
68         i, j = v
69         for voisin in voisins(etat, n, i, j):
70             i2, j2 = voisin
71             if etat[i2][j2][4] == 0:
72                 etat[i2][j2][4] = marque
73                 compteur += 1
74                 queue.append(voisin)
75
76     return compteur
77 #Renvoie la taille et le numéro de la plus grande composante connexe
78 def plusgrandecomposante(etat, n):
79     marque = 1
80     max = -1
81     for i in range(1, n):
82         for j in range(1, n):
83             if etat[i][j][4] == 0:
84                 marque += 1
85                 compteur = marquage(etat, n, i, j, marque)
86                 if compteur > max:
87                     max = compteur
88                     idmax = marque
89
90     return (idmax,max)

```

Affichage graphique

```
88 ##AFFICHAGE GRAPHIQUE DU GRAPHE
89
90 def graphisme(etat, n, surface, taille, idmax):
91     ratio = taille / n
92     couleurs = {}
93     for i in range(n):
94         for j in range(n):
95             for k in range(1, 3):
96                 marque = etat[i][j][4]
97                 if marque in couleurs:
98                     marquecouleurs = couleurs[marque]
99                 else:
100                     if marque == idmax:
101                         marquecouleurs = (0, 0, 0)
102                     else:
103                         marquecouleurs = (random.randrange(0, 256), random.randrange(0, 256),
104                                         random.randrange(0, 256))
105                         couleurs[marque] = marquecouleurs
106                 i2, j2 = dir[k]
107                 i2, j2 = i + i2, j + j2
108
109                 if etat[i][j][k] == 1:
110                     pygame.draw.line(surface, couleurs[marque], (i * ratio, j * ratio), (i2 * ratio,
111                                         j2 * ratio))
112 def afficher(N=1000,p=0.49):
113     etat = []
114     grille(stat, N)
115     percolateur(etat, N, p)
116     idmax = plusgrandecomposante(etat, N)[0]
117     pygame.init()
118     display = pygame.display.set_mode((700,700))
119     pygame.display.set_caption('Percolation dans Z^2')
120     display.fill((255,255,255))
121     graphisme(etat, N, display, 700, idmax)
122     pygame.display.update()
123     while not closed:
124         for event in pygame.event.get():
125             if event.type == pygame.QUIT:
126                 closed = True
127     pygame.quit()
```

percolation en 3D

```
130 import networkx as nx
131 import matplotlib.pyplot as plt
132 import random
133
134
135
136
137 from mpl_toolkits.mplot3d.art3d import Poly3DCollection as poly
138
139 def percolation_3D (n,p):
140     # Créer le graphe de Z^3
141
142     graphe = nx.grid_graph(dim=[n, n, n], periodic=True)
143
144     # Retire chaque arête avec une probabilité 1-p!!!!!!!!!!!!problème copy dico!!!!!
145     retire=[]
146     for arete in graphe.edges():
147         if random.random() > p:
148             retire.append(arete)
149     graphe.remove_edges_from(retire)
150     # Trouver les composantes connexes
151     composantesconnexes = list(nx.connected_components(graphe))
152
153     # Trouver la plus grande composante connexe
154     maxcomposante = max(composantesconnexes, key=len)
155
156     # Afficher les arêtes du graphe avec les composantes connexes
157     fig = plt.figure()
158     ax = fig.add_subplot(111, projection='3d')
159     couleurs = ['b', 'g', 'r', 'c', 'm', 'y']
160     for arete in graphe.edges():
161         u, v = arete
162         if u in maxcomposante and v in maxcomposante:
163             ax.plot([u[0], v[0]], [u[1], v[1]], [u[2], v[2]], c='k')
164         else:
165             couleur = random.choice(couleurs)
166             ax.plot([u[0], v[0]], [u[1], v[1]], [u[2], v[2]], c=couleur)
167
168 plt.show()
```

Statistiques

```

171 ##Statistiques
172 #Trace la taille moyenne de la plus grande composante connexe en fonction de p
173 def Montecarlo(n,q):
174     L=[];U=[0.1,0.3,0.4,0.41,0.42,0.43,0.44,0.45,0.46,0.47,0.48,0.485,0.49,0.491,0.492,0.493,0.494,0.495,0.496,0.497,0.498,0.499,0.5,0.505,0.51,0.52,0.53,0.54,0.55,0.56,0.57,0.58,0.59,0.6,0.7,0.9,1]
175     for d in U :
176         S=0
177         print(d)
178         for i in range (q):
179             g=[]
180             grille(g,n)
181             percolateur(g,n,d)
182             S+=plusgrandecomposante(g,n)[1]
183         L.append(S/q)
184     plt.figure()
185     plt.plot(U,L)
186     plt.grid()
187     plt.show()
188 #La même chose en 3D
189 def MC3D(n,q):
190     L=[];U=[0.1,0.2,0.21,0.22,0.23,0.24,0.242,0.244,0.246,0.247,0.248,0.249,0.25,0.255,0.26,0.28,0.29,
191     0.3,0.32,0.35,0.36,0.38,0.4,0.5,0.6,0.8,0.9,1]
192     for p in U :
193         S=0
194         print(p)
195         for i in range (q):
196             graphe = nx.grid_graph(dim=[n, n, n])
197             retire=[]
198             for arete in graphe.edges():
199                 if random.random() > p:
200                     retire.append(arete)
201             graphe.remove_edges_from(retire)
202             #graphe.remove_edge(*arete)
203             # Trouver les composantes connexes
204             composantesconnexes = list(nx.connected_components(graphe))
205             # Trouver la plus grande composante connexe
206             maxcomposante = max(composantesconnexes, key=len)
207             S+=len(maxcomposante)
208         L.append(S/q)
209     plt.figure()
210     plt.plot(U,L)
211     plt.grid()
212     plt.show()

```

Statistiques

```

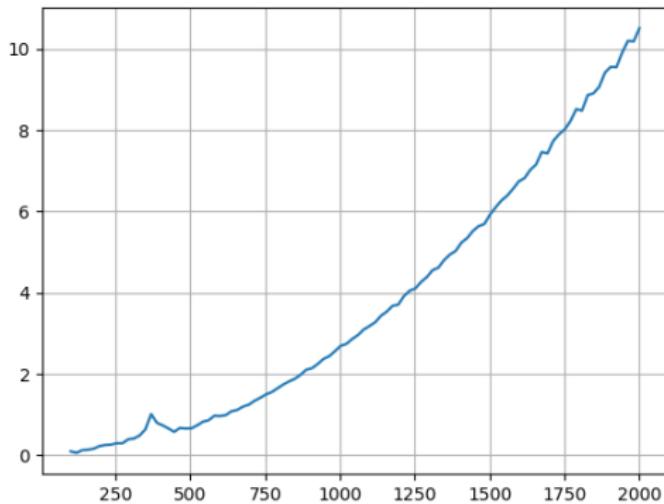
213 #Trace la probabilité d'avoir une plus grande composante connexe de taille supérieur à n en fonction
214 de n
215 def taille(n,p,q,l):
216     L=[]
217     for i in range (q):
218         L.append(marquage(general(n,p),n,n//2,n//2,1))
219     X = [l for i in range (n//2)];P=[];E=[]
220     for i in X:
221         print (i)
222         S=0
223         for c in L:
224             if c>=i:
225                 S+=1
226         S=S/q
227         P.append(S)
228         E.append(np.exp(-(i/l)))
229     plt.figure()
230     plt.plot(X,P)
231     plt.plot(X,E)
232     plt.show()
233 #Trace theta(p)en fonction de p pour différente valeur de n afin de voir jusqu'à où la simulation est
234 crédible
235 def cred (q):
236     N=[400,300,200,100,50,10];R=[];U=[0.1,0.3,0.4,0.41,0.42,0.43,0.44,0.45,0.46,0.47,0.48,0.485,0.49,0
237 ,491,0.492,0.493,0.494,0.495,0.496,0.497,0.498,0.499,0.5,0.505,0.51,0.52,0.53,0.54,0.55,0.56,0.57,0.58
238 ,0.59,0.6,0.7,0.9,1]
239     for n in N:
240         L=[]
241         for d in U :
242             S=0
243             print(d,n)
244             for i in range (q):
245                 g=[]
246                 grille(g,n);percolateur(g,n,d)
247                 S+=plusgrande(G(g)[1]/n**2
248             print (S)
249             L.append(S/q)
250         R.append(L)
251     plt.figure()
252     for i in range (len(R)):
253         plt.plot(U,R[i],label=N[i])
254     plt.legend(loc='upper left')
255     plt.show()

```

Complexité

```
256 ##Complexité
257
258
259 def Chronométrier(Programme, args) :
260     début = time()
261     _ = Programme(*args)
262     fin = time()
263     return fin - début
264
265
266 def Temps_moyen(Programme, n, nb_rep) :
267     temps = 0
268     for p in lnpinspace(0, 1, nb_rep) :
269         temps += Chronométrier(Programme, (n,p))
270     return temps/nb_rep
271
272 def Complexité(Programme, min = 100, max = 2000, nb_points = 20, nb_rep = 2) :
273     X = np.linspace(min, max, nb_points)
274     Y = []
275     i = 0
276     for x in X :
277         Y.append((Temps_moyen(Programme, int(x), nb_rep))**1/2)
278         i += 1
279         print(str(100*i//len(X))+"%")
280     plt.plot(X, Y)
281     plt.grid(True)
282     plt.show()
283     return X, Y
```

Complexité



Courbe du temps de réponse en fonction de n