Arbres couvrants aléatoires uniformes

Julien Bureaux et Ilia Smilga
sous la direction de Nicolas Curien.

On s'intéresse ici au problème de trouver un algorithme efficace pour tirer uniformément un arbre couvrant d'un graphe donné. On va d'abord définir la notion d'arbre couvrant et d'arbre couvrant uniforme, puis présenter deux algorithmes qui répondent au problème posé : l'algorithme d'Aldous-Broder et l'algorithme de Wilson. On va enfin proposer une implémentation de chacun de ces algorithmes, et comparer en pratique leur vitesse d'exécution.

Table des matières

1 Généralités sur les arbres couvrants

1.1 Définitions de base sur les graphes

On rappelle les définitions classiques suivantes :

- Un *graphe* est un couple $G = (S, A)$ où S est l'ensemble des *sommets* et $A \subset \binom{S}{2} = \{\{s, t\} \mid (s, t) \in S^2, s \neq t\}$ est l'ensemble des *arêtes*.
- Un graphe $G = (S, A)$ est dit *fini* si S est fini, *vide* si $S = \emptyset$.
- Soient $G = (S, A)$ un graphe et $s, t \in S$ deux sommets. Un *chemin* (de *longueur* n) de s à t dans G est un $n+1$-uplet $(s = s_0, \ldots, s_n = t)$ de sommets tel que pour tout $i \in \{0, \ldots, n-1\}$ on ait $\{s_i, s_{i+1}\} \in A$. Par convention on dit que pour tout sommet s il existe un chemin de longueur 0 de s à s. On définit une relation d'équivalence sur les sommets de G par $s \sim_G t$ si et seulement s'il existe un chemin de s à t dans G. Les classes d'équivalence pour $\sim_G$ sont appelée *composantes connexes* de G.
- Soient $G = (S, A)$ un graphe et $s \in S$. Si G admet un chemin de longueur non nulle de s à s qui n'utilise que des arêtes distinctes, on dit que G admet un *cycle*.
- Un graphe $G = (S, A)$ est *connexe* si et seulement si G a une seule composante connexe.
- Un *arbre* est un graphe connexe sans cycle.
- Soit $G = (S, A)$ un graphe. Un *sous-graphe* de G est un graphe $G' = (S', A')$ tel que $S' \subset S$ et $A' \subset A \cap \binom{S'}{2}$. Un *sous-graphe couvrant* de G est un graphe $G' = (S, A')$ tel que $A' \subset A$. Un *arbre couvrant* de G est un sous-graphe couvrant de G qui est un arbre. On note Couv_G l'ensemble des arbres couvrants de G.
- Soit $G = (S, A)$ un graphe et $s \in S$. Le *voisinage* de s est l'ensemble $V_G(s) = \{t \in S \mid \{s, t\} \in A\}$. Le *degré* de s est l'entier $\deg(s) = \left|V_G(s)\right|$.

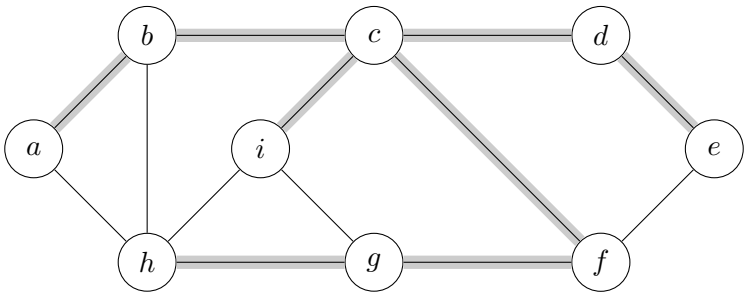


FIGURE 1 – Exemple de graphe avec un arbre couvrant mis en évidence.

1.2 Caractérisation des arbres couvrants

Soit $G = (S, A)$ un graphe fini non vide. Sans perte de généralité, on peut supposer que $S = \{1, \ldots, n\}$ avec $n \in \mathbb{N}^*$. Soit $s_1 \in S$. On considère l'algorithme d'exploration suivant :

1. On initialise R à $\{s_1\}$, S' à $\emptyset$, et A' à $\emptyset$.
2. Tant que R n'est pas vide on répète les étapes 3 et 4.
3. On retire le plus petit élément s de R et on l'ajoute à S'.
4. Pour chaque sommet $t \in S \setminus (R \cup S')$ tel que $\{s, t\} \in A$ on ajoute t à R et on ajoute l'arête $\{s, t\}$ à A'.

Proposition 1. *À la fin de l'exécution, S' est la composante connexe de s_1 dans G et $G' = (S', A')$ est un graphe connexe qui possède $|S'| - 1$ arêtes.*

Preuve. La première chose à voir est qu'à tout moment, R et S' sont disjoints et donc qu'à chaque exécution de l'étape 3, l'ensemble $S' \subset S$ croît strictement. L'algorithme s'arrête donc nécessairement au bout d'un temps fini.

Numérotons les sommets $s_1, \dots, s_n$ de S' de sorte que s_i soit le sommet choisi à la i-ème exécution de l'étape 3. On a alors par récurrence que pour tout $j \in \{2, \dots, n\}$, il existe un unique $i \in \{1, \dots, j-1\}$ tel que $a_j = \{s_i, s_j\} \in A'$. On en déduit en particulier que pour tout $j \in \{2, \dots, n\}$ il existe un chemin de s_1 à s_j dans G', et donc que G' est connexe. De plus on a exactement $A' = \{a_j \mid 2 \leq j \leq n\}$ et donc $|A'| = n - 1 = |S'| - 1$.

Finalement, soit t un sommet appartenant à la composante connexe de s_1, c'est à dire tel qu'il existe un chemin $(s_1 = t_1, \dots, t_r = t)$ dans G. Supposons qu'il existe un $j \in \{2, \dots, r\}$ minimal tel que $t_j \notin S'$. Par minimalité il existe alors $i \in \{1, \dots, n\}$ tel que $t_{j-1} = s_i$ et donc t_j serait ajouté à R à la i-ème exécution de l'étape 4, ce qui est absurde car tout sommet ajouté dans R finit dans S'. Un tel t n'existe donc pas et comme G' est un sous-graphe connexe de G, c'est exactement la composante connexe de s_1. □

Corollaire 1. *Soit $G = (S, A)$ un graphe fini non vide et sans cycle. On a $|A| \leq |S| - 1$ avec égalité si et seulement si G est connexe.*

Preuve. Soient $C_1, \dots, C_k$ les composantes connexes de G. On note $G_i = (C_i, A_i)$ le sous-graphe obtenu par l'algorithme d'exploration à partir d'un sommet quelconque de C_i. Soit $\{s, t\} \in A$. Il existe $i \in \{1, \dots, k\}$ tel que C_i contienne à la fois s et t. Comme G_i est connexe il existe un chemin de s à t dans G_i n'utilisant que des arêtes distinctes, et donc A_i contient $\{s, t\}$ car sinon on aurait un cycle dans G. On en déduit que les $(A_i)_{1 \leq i \leq k}$ forment une partition de A, d'où

$$|A| = \sum_{i=1}^{k} |A_i| = \sum_{i=1}^{k} (|S_i| - 1) = |S| - k \leq |S| - 1.$$

Le cas d'égalité équivaut à ce que $k = 1$, c'est à dire à ce que G soit connexe. □

Corollaire 2. *Soit $G = (S, A)$ un graphe fini non vide et connexe. On a $|A| \geq |S| - 1$ avec égalité si et seulement si G est sans cycle.*

Preuve. On considère le sous-graphe $G' = (S', A')$ obtenu par l'algorithme précédent à partir d'un sommet s_1 quelconque. Comme G est supposé connexe on a d'après la Proposition 1 que $S' = S$, d'où $|A| \geq |A'| = |S'| - 1 = |S| - 1$. Si G est sans cycle, l'inégalité fournie par le Corollaire 1 montre que $|A| = |S| - 1$. Réciproquement, si G possède un cycle $(s_0, s_1, \dots, s_k, s_0)$, alors $\tilde{G} = (S, A \setminus \{s_0, s_1\})$ est encore connexe et on a donc $|A| - 1 \geq |S| - 1$. □

En combinant ces deux résultats on obtient immédiatement la caractérisation suivante :

Théorème 1. *Soit $G = (S, A)$ un graphe fini non vide. Si deux des propriétés suivantes sont vérifiées alors la troisième aussi, et G est un arbre :*

(i) G est connexe ; *(ii) G est sans cycle ;* *(iii) $|A| = |S| - 1$.*

Corollaire 3. *Tout graphe connexe fini admet au moins un arbre couvrant.*

Preuve. Soit $G = (S, A)$ un tel graphe. Si $S \neq \emptyset$, il suffit de considérer le sous-graphe connexe $G' = (S', A')$ construit par l'algorithme d'exploration à partir d'un sommet quelconque. Par connexité de G on a $S' = S$, et comme $|A'| = |S'| - 1$, le sous-graphe couvrant G' est en fait un arbre couvrant de G par le théorème. □

1.3 Intérêt des arbres couvrants

Les arbres couvrants possèdent de nombreuses applications pratiques ainsi que d'intéressantes propriétés théoriques. En particulier :

- Ils sont utilisés dans le cadre des technologies de *réseaux maillés* où les hôtes sont reliés de proche en proche sans hiérarchie centrale. On a alors besoin de pouvoir calculer un sous-réseau tel que deux hôtes puissent toujours être reliés, et ce de manière unique. Dans le cas contraire, la présence de boucles est à l'origine de *tempêtes de diffusions* qui paralysent le réseau. L'algorithme utilisé pour générer les arbres couvrants nécessaires est le protocole STP (spanning tree protocol) défini dans le standard IEEE 802.1D.
- Souvent on associe à chaque arête un poids (on dit encore un coût) et on chercher un arbre couvrant de poids minimal. On se donne par exemple un ensemble de villes dont on connaît la distance qui les sépare et on veut construire un réseau routier de longueur minimale qui permet de toutes les relier. Ce type de problèmes peut être résolu avec les algorithmes de Prim et Kruskal (exposés tous les deux dans [5] par exemple).

1.4 Arbres couvrants du graphe complet

On dénombre ici les arbres couvrants du graphe complet. Le cas général du dénombrement des arbres couvrants est donné par le théorème de Kirchhoff (en annexe).

Définition 1. Soit $n \in \mathbb{N}^*$. On appelle *graphe complet* de taille n le graphe K_n dont l'ensemble des sommets est $\{1, \ldots, n\}$ et l'ensemble des arêtes est $\big\{\{i, j\} \mid 1 \leq i < j \leq n\big\}$.

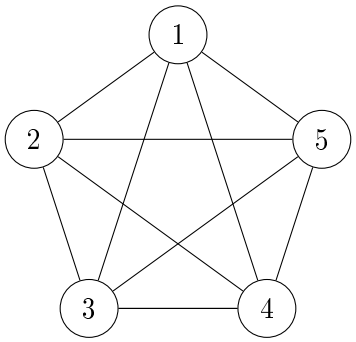


FIGURE 2 – Le graphe complet K_5.

Proposition 2. *Pour $n \geq 2$, on a* $\left|\mathrm{Couv}_{K_n}\right| = n^{n-2}$.

La première preuve de ce résultat est due à Cayley. Nous allons donner ici une preuve plus concise attribuée à André Joyal par Aigner et Ziegler [1].

Preuve. Commençons par remarquer que Couv_{K_n} est exactement l'ensemble des arbres à n sommets distingués (appelés arbres de Cayley).

Soient A un arbre de sommets $S = \{1, \ldots, n\}$, et $i, j \in S$ deux sommets. On appelle *arbre doublement enraciné en i et j* l'arbre A dans lequel on distingue les sommets i et j, noté $A_{i,j}$.

Le cardinal de l'ensemble des arbres doublement enracinés est clairement $n^2 |\mathrm{Couv}_{K_n}|$. Montrons par ailleurs qu'on peut mettre cet ensemble en bijection avec l'ensemble des applications de S dans lui-même. Si f est une telle application, notons $C_f = \left\{x \in S \mid \exists i \in \mathbb{N}^*,\ f^i(x) = x\right\}$, $c_1 < c_2 < \cdots < c_k$ les éléments de C_f et posons $d_i = f(c_i)$ pour $1 \leq i \leq k$. On considère alors le graphe H_f d'ensemble de sommets S et d'ensemble d'arêtes $\left\{\{d_i, d_{i+1}\} \mid 1 \leq i \leq k-1\right\} \cup \left\{\{j, f(j)\} \mid j \notin C_f\right\}$ dans lequel on distingue les sommets d_1 et d_k. Ce graphe est clairement connexe (pour tout $x \in S$ il existe $i \in \mathbb{N}$ tel que $f^i(x) \in C_f$) et sans cycle (puisque les cycles engendrés par f sont à support dans C_f qui ne correspond qu'à une ligne dans H_f). Il s'agit donc bien d'un arbre couvrant qui est de plus doublement enraciné en d_1 et d_k.

Inversement, si on se donne sur S un arbre doublement enraciné en s et t, il existe un unique chemin entre s et t passant par les sommets $s = d_1, d_2, \ldots, d_k = t$ dans l'ordre. On reconstruit alors l'application f de S dans lui-même en définissant pour tout $x \in S$:

- si x n'est pas un des d_i, $f(x)$ est le premier sommet sur le chemin reliant x à un des d_i
- si $x = d_i$ avec $1 \leq i \leq k$, $f(x)$ est le i-ème plus petit élément de $\{d_j \mid 1 \leq j \leq k\}$

On a ainsi exhibé une bijection et sa bijection réciproque. Le nombre d'arbres doublement enracinés est donc $|S^S| = n^n$, ce qui permet de conclure. □

1.5 Position du problème

Soit G un graphe connexe fini. Notre objectif est de construire une variable aléatoire $\mathcal{A}$ de loi uniforme sur Couv_G. Une solution naïve consiste à construire tous les arbres couvrants de G puis à en tirer un au hasard de manière uniforme, mais cela demande beaucoup de temps de calcul et de mémoire. En effet, on vient de voir que le nombre d'arbres couvrants peut être très grand en fonction du nombre de sommets. On commence par présenter l'algorithme d'Aldous-Broder qui fait beaucoup mieux en moyenne, puis l'algorithme de Wilson qui est encore plus efficace.

2 Algorithme d'Aldous-Broder

Cet algorithme est tiré d'un article de 1990 dû à David Aldous et intitulé *The Random Walk Construction of uniform Spanning Trees and Uniform Labelled Trees* [2]. Il a également été découvert de manière indépendante par Andrei Broder [3]. Toute sa beauté réside dans sa simplicité et dans le fait qu'il génère un arbre couvrant uniforme sans même avoir besoin de connaître le nombre d'arbres couvrants.

2.1 Présentation de l'algorithme

- On simule une marche aléatoire simple issue d'un sommet arbitraire et on marque les sommets rencontrés au fur et à mesure.
- Dès qu'on rencontre un sommet non marqué, on ajoute l'arête que l'on vient d'emprunter à l'arbre déjà construit.
- On s'arrête dès que tous les sommets sont marqués.

2.2 Formalisation

On se fixe un graphe fini $G = (S, A)$ contenant au moins un sommet $x_0 \in S$.

Définition 2. On note $SRW_G^{x_0}$ la loi d'une chaîne de Markov sur S issue de x_0 dont la fonction de transition P est donnée par

$$P(s,t) = \frac{1}{\deg(s)} \times \mathbb{1}_{\{t \in V(s)\}} \qquad ((s,t) \in S^2).$$

On appelle *marche aléatoire (simple)* sur G issue de x_0 un processus suivant la loi $SRW_G^{x_0}$.

Proposition 3. *La matrice de transition P admet une mesure de probabilité réversible μ donnée par*

$$\mu(s) = \frac{\deg(s)}{\deg(G)} \quad (s \in S) \qquad \text{où } \deg(G) = \sum_{s \in S} \deg(S).$$

Preuve. La vérification est immédiate. □

Définition 3. On note SRW_G la loi d'une chaîne de Markov stationnaire de fonction de transition P et dont la loi instantanée est μ. On appelle *marche aléatoire stationnaire* sur G un processus $(X_n)_{n \geq 0}$ à valeurs dans S et dont la loi est SRW_G.

Définition 4. On note $\mathcal{E}_G$ l'ensemble des suites $(x_n)_{n \geq 0}$ à valeurs dans S vérifiant les deux conditions suivantes :

(i) pour tout $n \geq 0$, $\{x_n, x_{n+1}\} \in A$;

(ii) il existe $N \geq 0$ tel que $\{x_n \mid 0 \leq n \leq N\} = S$.

Définition 5. Soit $\mathbf{x} \in \mathcal{E}_G$. Pour tout sommet $s \in S$ on définit le *temps d'atteinte* $T_s(\mathbf{x})$ de s par $\mathbf{x}$,

$$T_s(\mathbf{x}) = \min\{i \geq 0 \mid x_i = s\}.$$

On définit alors le *temps de parcours* de G par $\mathbf{x}$,

$$T(\mathbf{x}) = \max_{s \in S} T_s(\mathbf{x}).$$

Définition 6. Soit $\mathbf{x} \in \mathcal{E}_G$. On appelle *sous-graphe d'Aldous* construit à partir de $\mathbf{x}$ le sous-graphe couvrant $\mathcal{A}(\mathbf{x})$ dont l'ensemble des arêtes est

$$\{\{x_{T_s(\mathbf{x})-1}, x_{T_s(\mathbf{x})}\} \mid s \in S \setminus \{x_0\}\}.$$

Proposition 4. *Soit $\mathbf{x} \in \mathcal{E}_G$. Le sous-graphe d'Aldous $\mathcal{A}(\mathbf{x})$ construit à partir de $\mathbf{x}$ est un arbre couvrant de G.*

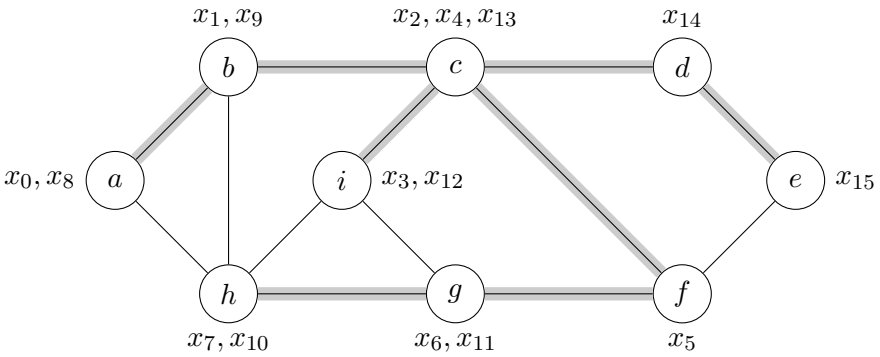


FIGURE 3 – Construction de l'arbre d'Aldous obtenu à partir d'une suite de sommets $\mathbf{x}$ commençant par $a, b, c, i, c, f, g, h, a, b, h, g, i, c, d, e, \ldots$

Preuve. Le sous-graphe $\mathcal{A}(\mathbf{x})$ possède exactement une arête par sommet de $S \setminus \{x_0\}$, soit $n-1$ en tout. Quitte à renuméroter correctement les sommets $s_0, \ldots, s_{n-1}$ on peut supposer que $0 = T_{s_0}(\mathbf{x}) < \cdots < T_{s_{n-1}}(\mathbf{x}) = T(\mathbf{x})$ et on montre alors par récurrence sur k que pour tout $k \in \{1, \ldots, n-1\}$, il existe un chemin de $s_0 = x_0$ à s_k dans $\mathcal{A}(\mathbf{x})$ ne passant que par des sommets de $\{s_0, \ldots, s_{k-1}\}$. Il en découle que $\mathcal{A}(\mathbf{x})$ est connexe, ce qui permet de conclure avec le Théorème 1. □

Proposition 5. *Soit $\mathbf{X} = (X_n)_{n\geq 0}$ une marche aléatoire sur G. Presque sûrement $\mathbf{X} \in \mathcal{E}_G$.*

Preuve. La propriété (i) est immédiatement vérifiée pour une marche aléatoire. Pour (ii), comme G est connexe et fini, la chaîne de Markov $(X_n)_{n\geq 0}$ est récurrente irréductible. □

Le résultat important et le plus délicat à prouver à propos de l'algorithme d'Aldous-Broder est le suivant.

Théorème 2. *Soit $\mathbf{X} = (X_n)_{n\geq 0}$ une marche aléatoire sur G issue de x_0. Le sous-graphe d'Aldous $\mathcal{A}(\mathbf{X}) = \mathcal{A}(X_0, X_1, \ldots)$ suit la loi uniforme sur* Couv_G.

2.3 Chaînes de Markov stationnaires bilatères

La preuve du théorème utilise fortement le résultat suivant qui découle directement du théorème de Kolmogorov.

Théorème 3. *Soient E un espace d'états discret, Q une fonction de transition sur E et μ une mesure de probabilité stationnaire pour Q. On note $\nu_{n,m}$ la mesure sur $E^{\{n,\ldots,m\}}$ définie par*

$$\nu_{n,m}(x_n, x_{n+1}, \ldots, x_m) = \mu(x_n)Q(x_n, x_{n+1}) \ldots Q(x_{m-1}, x_m).$$

Il existe une unique mesure de probabilité ν sur $E^{\mathbb{Z}}$ telle que pour tous $m, n \in \mathbb{Z}$, $m \leq n$, on ait $pr_{m,n}\nu = \nu_{m,n}$. Toute suite $\mathbf{X} = (X_n)_{n\in\mathbb{Z}}$ de loi ν possède la propriété de Markov, est stationnaire de loi instantanée μ, et a pour fonction de transition Q. On dit que $\mathbf{X}$ est une chaîne de Markov stationnaire bilatère. *De plus, le processus $\tilde{\mathbf{X}} = (\tilde{X}_n)_{n\in\mathbb{Z}}$ défini par $\tilde{X}_n = X_{-n}$ est une chaîne de Markov bilatère stationnaire de loi instantanée μ et de fonction de transition $\tilde{Q}$ donnée par*

$$\tilde{Q}(y,x) = \frac{\mu(x)Q(x,y)}{\mu(y)}.$$

De plus, si la loi stationnaire μ est réversible par rapport à Q alors $\mathbf{X}$ et $\tilde{\mathbf{X}}$ ont la même loi.

2.4 Preuve de l'uniformité

Définition 7. Soit $G = (S, A)$. On note Couv_G^r l'ensemble des *arbres couvrants enracinés en un sommet* r, c'est à dire l'ensemble des couples (H, r) où $H \in \mathrm{Couv}_G$ est un arbre couvrant de G et $r \in S$ est un sommet distingué appelé la *racine* de (H, r). L'arbre enraciné (H, r) sera plus simplement noté Hr. On pose

$$\mathrm{Couv}_G^* = \bigcup_{s\in S} \mathrm{Couv}_G^s .$$

Définition 8. Soient $H \in \mathrm{Couv}_G$ un arbre couvrant et $s, t \in S$ deux sommets distincts. On note $1\mathrm{er}_H(s \to t)$ le premier sommet rencontré sur l'unique chemin de s à t dans l'arbre H.

Soit $Hr \in \mathrm{Couv}_G^*$ un arbre enraciné et $r' \in V_G(r)$. On considère le graphe H' obtenu à partir de H en lui ajoutant l'arête $\{r, r'\}$ et en lui retirant $\{r', 1\mathrm{er}_H(r' \to r)\}$. On note alors $T(H, r, r')$ le triplet $(H', r', 1\mathrm{er}_H(r' \to r))$ et

$$T_{r'} : (H, r) \in \mathrm{Couv}_G \times V_G(r') \longmapsto (H', r').$$

De même, on notera $S(H, r, r')$ le triplet $(H', 1\mathrm{er}_H(r \to r'), r)$ où H' est le graphe formé à partir de H en lui ajoutant l'arête $\{r, r'\}$ et en lui retirant $\{r, 1\mathrm{er}_H(r \to r')\}$. On peut de plus vérifier que $S \circ T = T \circ S$.

Proposition 6. *Pour tout sommet $r' \in s$, l'application $T_{r'}$ est à valeurs dans* Couv_G^*.

Preuve. Posons $s = 1\mathrm{er}_H(r' \to r)$. Si $r' \in V_H(r)$ alors $s = r$ et $H' = H$. Sinon $s \notin \{r, r'\}$. On note $\tilde{H}$ le graphe H auquel on a retiré l'arête $\{s, r'\}$, C_s et $C_{r'}$ les composantes connexes respectives de s et r' dans $\tilde{H}$. Alors r est dans C_s et l'ajout de l'arête $\{r, r'\}$ rend connexe le graphe. Comme H et H' ont le même nombre d'arêtes ceci permet de conclure. □

Proposition 7. *Soit $\mathbf{x} = (x_n)_{n\geq 0} \in \mathcal{E}_G$ et $s \in S$ tel que $\{s, x_0\} \in A$. On a alors*

$$T\big(\mathcal{A}(\mathbf{x}), x_0, s\big) = \big(\mathcal{A}(s, x_0, \dots), s, x_{T_s(\mathbf{x})-1}\big).$$

Preuve. Si l'on ajoute s en tête de $\mathbf{x}$ alors la première visite de x_0 correspond à l'utilisation de l'arête $\{s, x_0\}$, puis on suit l'ordre des sommets de $\mathbf{x}$. On obtient donc $\mathcal{A}(s, x_0, x_1, \dots)$ à partir de $\mathcal{A}(x_0, x_1, \dots)$ en ajoutant l'arête $\{s, x_0\}$ et en retirant $\{x_{T_s(\mathbf{x})-1}, s\}$ (comme $T_s(s, x_0, \dots) = 0$ elle n'a pas à être prise). Ceci correspond exactement au calcul de $T\big(\mathcal{A}(\mathbf{x}), x_0, s\big)$. Il est par ailleurs clair que $1\mathrm{er}_{\mathcal{A}(\mathbf{x})}(s \to x_0) = x_{T_s(\mathbf{x})-1}$ puisque $\mathcal{A}$ est un arbre couvrant et donc contient un et un seul chemin entre deux sommets donnés quelconques. □

Remarque. Ce résultat est la clef de la démonstration. On aura aussi besoin de remarquer que pour tout $H'r' \in \mathrm{Couv}_G^*$, on a

$$\left|T_{r'}^{-1}(H'r')\right| = \deg r',$$

ce qui découle directement des définitions de T et S.

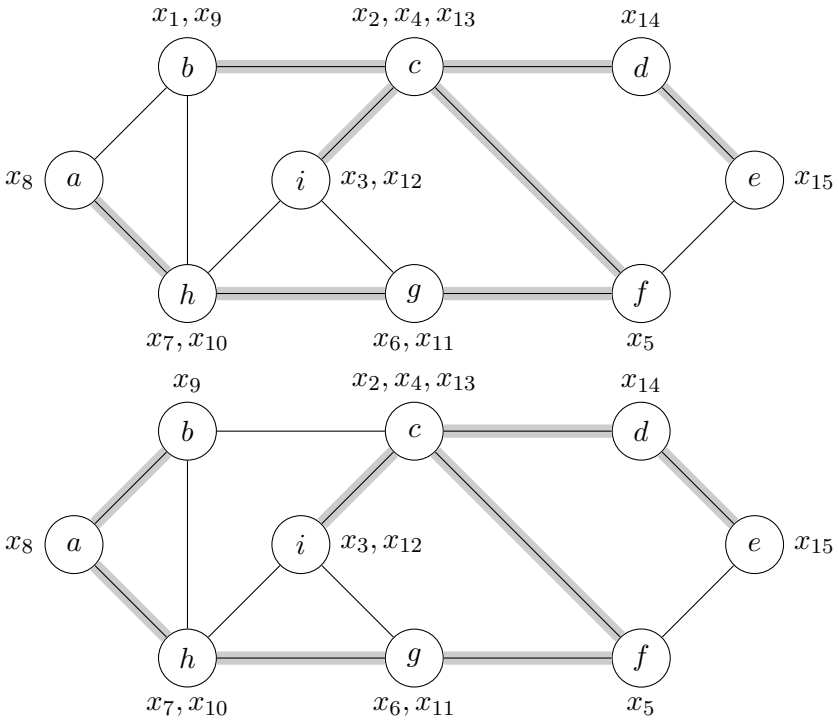


FIGURE 4 – Modifications de l'arbre d'Aldous obtenues lorsqu'on « oublie » x_0 puis x_1.

Soit $(X_n)_{n\in\mathbb{Z}}$ une marche aléatoire stationnaire sur G. Pour tout $n \in \mathbb{Z}$ on pose $\mathcal{A}_n = \mathcal{A}(X_n, X_{n+1}, \dots)$ et on note $\mathcal{H}_n R_n = \mathcal{A}_{-n} X_{-n}$.

Proposition 8. *Le processus $(\mathcal{H}_n R_n)_{n\geq 0}$ est une chaîne de Markov sur Couv_G^* de fonction de transition donnée par*

$$Q(Hr, H'r') = P(r, r') \times \mathbb{1}_{\{T_{r'}(Hr) = H'r'\}} = \frac{1}{\deg r} \times \mathbb{1}_{\{T_{r'}(Hr) = H'r'\}}.$$

Preuve. Pour tout $n \geq 0$ on a d'après la proposition précédente et en utilisant la réversibilité de la chaîne $(X_n)_{n\in\mathbb{Z}}$ que

$$\begin{aligned}
&\mathbb{P}(\mathcal{H}_{n+1}R_{n+1} = H_{n+1}r_{n+1} \mid \mathcal{H}_0R_0 = H_0r_0, \dots, \mathcal{H}_nR_n = H_nr_n) \\
&\qquad = \mathbb{P}(T_{R_{n+1}}(H_nr_n) = H_{n+1}r_{n+1} \mid \mathcal{H}_0R_0 = H_0r_0, \dots, \mathcal{H}_nR_n = H_nr_n) \\
&\qquad = \mathbb{P}(T_{r_{n+1}}(H_nr_n) = H_{n+1}r_{n+1}, R_{n+1} = r_{n+1} \mid R_n = r_n) \\
&\qquad = \mathbb{1}_{\{T_{r_{n+1}}(H_nr_n) = H_{n+1}r_{n+1}\}} \times \mathbb{P}(X_{-n-1} = r_{n+1} \mid X_{-n} = r_n) \\
&\qquad = P(r_n, r_{n+1}) \times \mathbb{1}_{\{T_{r_{n+1}}(H_nr_n) = H_{n+1}r_{n+1}\}}.
\end{aligned}$$

□

Proposition 9. *La fonction de transition Q admet une mesure de probabilité stationnaire ν définie par*

$$\nu(Hr) = \frac{\deg(r)}{\deg(G)|\operatorname{Couv}_G|} \qquad (Hr \in \operatorname{Couv}_G^*).$$

Preuve. Soit $H'r' \in \operatorname{Couv}_G^*$ fixé. On a alors,

$$\sum_{(H,r)\in\operatorname{Couv}_G^*} \nu(Hr)Q(Hr, H'r') = \sum_{\substack{(H,r)\in\operatorname{Couv}_G^* \\ T_{r'}(Hr)=H'r'}} \deg(r) \times \frac{1}{\deg r} = \deg r'$$

d'après la remarque plus haut sur le cardinal de $T_{r'}^{-1}$. □

Or cette probabilité ne dépend pas de l'arbre couvrant mais uniquement de sa racine dont on sait déjà qu'elle suit la loi stationnaire μ proportionnelle au degré. On en déduit que $(\mathcal{H}_n R_n)_{n\geq 0}$ est une chaîne de Markov stationnaire. En particulier on a donc pour tout $H \in \operatorname{Couv}_G^*$,

$$\mathbb{P}(\mathcal{A}_0 = H) = \mathbb{P}(\mathcal{H}_0 = H) = \sum_{r\in S} \mathbb{P}(\mathcal{H}_0 R_0 = Hr) = \frac{1}{|\operatorname{Couv}_G|}.$$

À ce stade on a seulement prouvé le résultat suivant :

Proposition 10. *Soit $(X_n)_{n\geq 0}$ une marche aléatoire stationnaire sur G. Alors $\mathcal{A} = \mathcal{A}(X_0, X_1, \dots)$ suit la loi uniforme sur* Couv_G.

Il nous manque encore un dernier point pour pouvoir conclure.

Proposition 11. *Les variables aléatoires $\mathcal{A}_0$ et X_0 sont indépendantes.*

Preuve. Pour tout arbre couvrant $H \in \operatorname{Couv}_G$ et tout sommet $r \in S$ on a

$$\mathbb{P}(\mathcal{A}_0 = H \text{ et } X_0 = r) = \nu(Hr) = \frac{\deg(r)}{\deg(G)|\operatorname{Couv}_G|} = \mathbb{P}(\mathcal{A}_0 = H) \times \mathbb{P}(X_0 = r).$$

□

Il en découle que la loi de $\mathcal{A}_0$ ne dépend pas de celle de X_0. En particulier le résultat de la proposition précédente est encore vrai si l'on considère la loi de $\mathcal{A}_0$ conditionnellement à $X_0 = s_0$.

2.5 Temps d'exécution pour le graphe complet

Proposition 12. *Soit $n \in \mathbb{N}^*$. On considère une marche aléatoire $\mathbf{X} = (X_k)_{k\geq 0}$ issue de 1 sur K_n. Soit $\tau_n = T(\mathbf{X})$ le temps de parcours de K_n par $\mathbf{X}$ tel que définit plus haut. Lorsque l'on fait tendre n vers $+\infty$ on a*

$$\mathbb{E}(\tau_n) \sim n \ln n, \qquad \operatorname{Var}(\tau_n) \sim \frac{(n\pi)^2}{6}.$$

De plus $\left(\dfrac{\tau_n}{n \ln n}\right)$ converge en probabilité vers 1.

Preuve. Soit $(\mathcal{F}_n)_{n\geq 0}$ la filtration canonique associée à $\mathbf{X}$. On pose $t_0 = 0$, $E_0 = \{1\}$, et pour $0 \leq i < n-1$,

$$t_{i+1} = \min\{k \geq t_i \mid X_k \notin E_i\}, \qquad E_i = \{X_0, \ldots, X_{t_i}\}.$$

de sorte que le temps de parcours qui nous intéresse est $\tau_n = t_{n-1}$. De plus les t_i sont des temps d'arrêt par rapport à $(\mathcal{F}_n)_{n\geq 0}$ et sont presque sûrement finis puisque majorés par τ_n.

Pour tout $i \geq 0$ on peut écrire d'après la propriété de Markov forte que

$$t_{i+1} - t_i = \min\{k \geq 0 \mid X_{t_{i+k}} \notin E_i\} = \min\{k \geq 0 \mid Y_k^{(i)} \notin E_i\}$$

où les $\left(Y^{(i)}\right)$ sont $n-1$ suites de variables aléatoires deux à deux indépendantes et respectivement indépendantes de $(X_k)_{0\leq k\leq t_i}$, et suivent respectivement les lois de marches aléatoires sur K_n issues des X_{t_i} (notons en particulier que les $t_{i+1} - t_i$ sont indépendants). Ainsi pour tout $0 \leq i < n-1$ l'accroissement $t_{i+1} - t_i$ a-t-il la même loi que

$$\Delta_i = \min\{k \geq 0 \mid U_k^{(i)} \notin \{1, \ldots, i+1\}\}$$

où chaque $\left(U^{(i)}\right)$ est une suite de variables aléatoires indépendantes de même loi $\mathcal{U}_{n-1}$ la loi uniforme sur $\{1, \ldots, n-1\}$. Pour tout $0 \leq i < n-1$, Δ_i suit la loi géométrique de paramètre $p_i = \frac{i+1}{n-1}$. On a donc finalement

$$\mathbb{E}(\tau_n) = \mathbb{E}\left(\sum_{i=0}^{n-2}(t_{i+1} - t_i)\right) = \sum_{i=0}^{n-2}\mathbb{E}(t_{i+1} - t_i) = (n-1)\sum_{i=0}^{n-2}\frac{1}{i+1} \sim n\ln n,$$

$$\mathrm{Var}(\tau_n) = \mathrm{Var}\left(\sum_{i=0}^{n-2}(t_{i+1} - t_i)\right) = \sum_{i=0}^{n-2}\mathrm{Var}(T_{i+1} - T_i) = (n-1)\sum_{i=0}^{n-2}\frac{n-2-i}{(i+1)^2} \sim \frac{(n\pi)^2}{6}.$$

Soit alors $\varepsilon > 0$. On a d'après l'inégalité de Markov que

$$\begin{aligned}
\mathbb{P}\left(\left|\frac{\tau_n}{n\ln n} - 1\right| \geq \varepsilon\right) &= \mathbb{P}(|\tau_n - n\ln n| \geq \varepsilon n\ln n) \\
&\leq \frac{\mathrm{Var}(\tau_n - n\ln n) + (\mathbb{E}(\tau_n - n\ln n))^2}{(\varepsilon n\ln n)^2} \\
&= \frac{\mathrm{Var}(\tau_n) + o((n\ln n)^2))}{(\varepsilon n\ln n)^2} \\
&= \frac{\pi^2 + o(1)}{6(\ln n)^2} + o(1) = o(1)
\end{aligned}$$

D'où la deuxième partie de la proposition. □

2.6 Une formule de dénombrement

Une conséquence de l'uniformité de l'arbre d'Aldous-Broder est que si l'on est capable de calculer la probabilité de sortie d'un arbre couvrant particulier alors on peut en déduire le nombre d'arbres couvrants du graphe. C'est par exemple le cas lorsque le graphe admet un chemin Hamiltonien (c'est à dire un chemin qui passe une et une seule fois par chaque sommet).

Proposition 13. *Supposons que $s_1, s_2, \ldots, s_n$ soit un chemin Hamiltonien dans G. Soit $B_i = \{s_1, \ldots, s_i\}$ et soit ρ_i la probabilité que la marche aléatoire sur G initialisée en s_i retourne en s_i avant de quitter B_i. Alors*

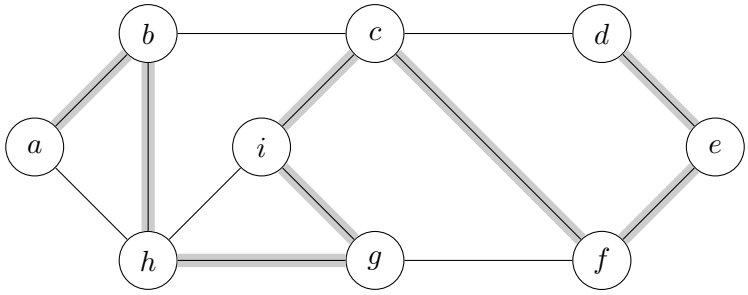


FIGURE 5 – Exemple de graphe avec un chemin Hamiltonien.

$$N(G) = \prod_{i=1}^{n-1} \deg(s_i)(1-\rho_i).$$

Preuve. Soit a le chemin $(s_1, \ldots, s_n)$ vu comme un arbre enraciné en s_1. On considère l'arbre $\mathcal{A}$ construit lors de la marche aléatoire sur G issue de s_1. On définit le temps d'arrêt correspondant à la sortie hors de B_i,

$$U_i = \min\{k \geq 1 \mid X_k \notin B_i\}.$$

On peut alors écrire que

$$\mathbb{P}(\mathcal{A} = a) = \prod_{i=1}^{n-1} \mathbb{P}_{s_i}\big((X_{U_i-1}, X_{U_i}) = (s_i, s_{i+1})\big).$$

Or, partant de s_i, le nombre moyen de visites en s_i avant de quitter B_i est $1/(1-\rho_i)$ et chaque visite sera suivie d'une traversée de (s_i, s_{i+1}) avec une probabilité $1/\deg(s_i)$. On a donc

$$\mathbb{P}_{s_i}\big((X_{U_i-1}, X_{U_i}) = (s_i, s_{i+1})\big) = \frac{1}{\deg(s_i)} \frac{1}{1-\rho_i},$$

et le résultat en découle en appliquant le Théorème 2 d'uniformité. □

3 Algorithme de Wilson

3.1 Description de l'algorithme

On se donne, pour toute cette section, un graphe fini connexe $G = (S, A)$. On commence par définir la notion de marche aléatoire à boucles effacées.

Définition 9. Soit $X = (X_0, X_1, \ldots)$ une marche sur G. On note $BE(X)$ (comme *loop-erased*) la *marche à boucles effacées* correspondante, qui est une suite de chemins finis du graphe, avec la propriété qu'aucun de ces chemins ne passe deux fois par le même point. On la définit par récurrence :

- $BE(X)_0 = (X_0)$
- Pour $n > 0$, on pose $BE(X)_{n-1} = (X'_0, \ldots, X'_k)$ et on distingue deux cas :
  - s'il existe un indice $i \leq k$ tel que $X_n = X'_i$ (il est alors unique), on pose $BE(X)_n = (X'_0, \ldots, X'_{i-1}, X'_i)$;
  - sinon, on pose $BE(X)_n = (X'_0, \ldots, X'_{k-1}, X'_k, X_n)$.

On vérifie facilement que $BE(X)_n$ est encore un chemin fini de G qui ne passe pas deux fois par le même point.

Voici alors la description de l'algorithme de Wilson. Il est entendu que tous les tirages aléatoires réalisés lors de l'exécution de l'algorithme sont indépendants.

On commence par choisir un ordre $(s_0, \ldots, s_{n-1})$ des sommets du graphe. (On verra que l'ordre des sommets n'a aucune importance, donc on n'a pas besoin de préciser la loi.) On appelle $\mathcal{A} = (S_\mathcal{A}, A_\mathcal{A})$ le futur arbre couvrant, qu'on initialise à $(\{s_0\}, \emptyset)$. Ensuite, pour chaque sommet restant s (dans l'ordre choisi : on commence par s_1 et on finit par s_{n-1}), on fait une marche aléatoire à boucles effacées partant de s, et on s'arrête dès qu'on atteint l'arbre $\mathcal{A}$, i.e. on prend le chemin $BE(X)_i = (X^i_0, \ldots, X^i_{k_i})$ où i est le plus petit indice tel que $X^i_{k_i} \in S_\mathcal{A}$. On incorpore alors le chemin obtenu dans l'arbre, c'est-à-dire que pour tout j tel que $0 \leq j < k_i$, on rajoute X^i_j dans $S_\mathcal{A}$ et (X^i_j, X^i_{j+1}) dans $A_\mathcal{A}$. (En particulier, si s est déjà dans l'arbre, on a $i = 0$ et on ne fait donc aucune modification sur l'arbre.)

Il est facile de montrer (par récurrence) qu'à chaque étape, $\mathcal{A}$ est un arbre. Or par construction, le résultat final contient tous les sommets. C'est donc un arbre couvrant de G.

Théorème 4. *Quelle que soit la façon dont on choisit l'ordre des sommets, l'algorithme de Wilson permet de tirer un arbre couvrant uniforme de G.*

3.2 Preuve de l'algorithme

Preuve. On fixe, une fois pour toutes, un ordre arbitraire sur les sommets $(s_0, \ldots, s_{n-1})$ du graphe.

On va commencer par faire quelques reformulations successives de l'algorithme, qui ne changent pas la loi du résultat. D'abord, la plus importante : on va essayer de regrouper tous les tirages aléatoires au même endroit, de sorte que le reste de l'algorithme devienne déterministe. On va se donner :

- un ensemble M, indexé par $(S \setminus \{s_0\}) \times \mathbb{N}$, de variables aléatoires toutes indépendantes et telles que $M_{s,n}$ soit un voisin uniforme de s. Ce sont ces variables qui vont indiquer les directions à prendre à partir de chaque sommet ; comme n'importe quel sommet peut être

visité un nombre de fois arbitrairement grand, on a besoin d'une liste infinie de directions pour chaque sommet.
- un ensemble P, indexé par $S \setminus \{s_0\}$, de compteurs à valeurs dans $\mathbb{N}$. La lettre P signifie « profondeur » ; l'image à retenir est celle d'un empilement de flèches en chaque sommet, qu'on dépile après avoir suivi chaque flèche.

Dans chacune des variantes suivantes de l'algorithme, on va commencer par faire un tirage de M et initialiser tous les P_s à 0 (cette partie sera omise par souci de concision). De plus, on convient que les sommets sont toujours parcourus dans l'ordre qu'on a fixé ; ainsi, dans tout ce qui suit, il faut comprendre l'expression « pour chaque sommet, on fait ... » comme un raccourci pour dire « pour i allant de 1 à $n-1$, on fait avec s_i ... ».

Remarque. Pour tirer M, il faut faire un nombre infini de tirages, ce qui est bien entendu impossible à réaliser en pratique. Cependant, d'un point de vue théorique, on peut attribuer un sens tout à fait précis à cette action. Il ne s'agit donc pas d'un vrai algorithme, mais d'une sorte d'algorithme abstrait qui ne sert qu'à la démonstration.

Reformulation 1. *On applique l'algorithme de Wilson tel quel, à ceci près qu'à chaque fois qu'on a besoin de choisir un voisin aléatoire d'un sommet s, on prend M_{s,P_s} puis on incrémente P_s.*

Justification. La seule différence par rapport à la version originelle est le moment où se font les tirages aléatoires. La loi, et l'indépendance, de toutes les variables utilisées est préservée. □

Reformulation 2. *On applique encore un algorithme similaire : à partir de chaque sommet, on essaye de rejoindre la partie de l'arbre déjà construite en suivant les flèches M_{s,P_s}, mais cette fois-ci, on n'incrémente plus les P_s à chaque fois. Si on y arrive, on rajoute le chemin suivi dans l'arbre. Si on échoue, à savoir si on tombe sur un sommet s_i déjà visité pendant cette étape (donc si on détecte un cycle), on incrémente en même temps les P_s de chaque sommet du cycle avant de repartir de s_i.*

Justification. On va montrer que, pour une même valeur de M, cette version de l'algorithme va donner le même résultat que la version précédente. Pour cela, il suffit de vérifier que dans les deux versions, à chaque fois qu'on visite un sommet s, on suit la même flèche sortante, donc que la valeur de P_s est la même.

Associons pour cela à chaque sommet $s \in S \setminus \{s_0\}$ un nouveau compteur A_s, qui vaut 1 si le sommet est déjà intégré à l'arbre, 1 s'il appartient au chemin qu'on est en train de construire (sans compter son extrémité finale), et 0 sinon. Ainsi, quand on visite une arête, on incrémente le A_s qui correspond à son origine ; quand on efface un cycle, on décrémente tous les A_s correspondants ; enfin, quand on adjoint un nouveau chemin au graphe, on ne touche pas aux A_s. Or :
- dans la version précédente de l'algorithme, on incrémente chaque P_s à chaque fois qu'on incrémente le A_s correspondant ;
- dans la version actuelle, on incrémente P_s à chaque fois qu'on décrémente A_s ;
- chaque A_s vaut 0 au début, et vaut 0 à chaque fois qu'on doit sortir de s (en effet, si A_s vaut 1, cela signifie soit que $s \in S_{\mathcal{A}}$, et alors on s'arrête là, soit qu'on a détecté un cycle, et alors avant de poursuivre son chemin, on efface le cycle donc en particulier on remet A_s à 0) ;
- enfin, A_s ne prend par définition que les valeurs 0 et 1.

Tout ceci montre qu'à chaque visite de s, P_s a la même valeur dans les deux versions de l'algorithme. □

Remarquez bien ce qui se passe : à chaque fois qu'on détecte un cycle, on incrémente les compteurs, de façon à voir le graphe qui est « en dessous ». Ce procédé est appelé *dépilement de cycle* (« *cycle popping* » en anglais).

Définition 10. On appelle *graphe visible* le graphe orienté formé par les arêtes (s, M_{s,P_s}). Il est donc caractérisé par la propriété suivante : tout sommet autre que s_0 est de degré sortant 1, et s_0 est de degré sortant 0.

Reformulation 3. *On vérifie si le graphe visible est un arbre, par l'algorithme évident : à partir de chaque sommet, on essaye de rejoindre la racine (ou, ce qui revient au même, la partie du graphe déjà vérifiée). Dès qu'on détecte un cycle (i.e. on retombe sur un sommet déjà visité avant d'avoir atteint la partie vérifiée), on le dépile, et on poursuit la vérification. S'il n'y a plus de cycles à dépiler, on a gagné : le graphe visible est bien un arbre.*

Justification. Comme on l'a vu, à chaque fois qu'on visite une arête d'origine s, A_s vaut 0, donc $s \notin S_{\mathcal{A}}$. Ainsi, chaque cycle détecté est disjoint de $\mathcal{A}$ et n'interfère pas avec la partie déjà vérifiée. Mis à part cela, c'est exactement le même algorithme, seul le point de vue change : au lieu de construire l'arbre petit à petit, on le considère dans sa totalité et on le modifie au fur et à mesure. □

La démonstration repose alors sur le lemme suivant :

Lemme 1. *Fixons une valeur de M. Alors l'ordre dans lequel on dépile les cycles lors de l'exécution de l'algorithme n'influe pas sur le résultat. Autrement dit, ou bien toute suite maximale de dépilements de cycles est infinie, ou bien elle se termine toujours sur un ensemble de valeurs P^{fin} des compteurs P qui ne dépend pas de la suite de cycles.*

Preuve. L'observation clé est la suivante : dans tout graphe visible, deux cycles distincts sont nécessairement disjoints. En effet, sachant que tout sommet est l'origine d'au plus une arête, si deux cycles partagent un sommet, de proche en proche, on voit qu'ils sont nécessairement confondus.

On va alors montrer qu'il existe un ensemble $\mathcal{C}$ (fini ou infini) de cycles potentiellement dépilables (formellement, ce sont des parties de $S \times \mathbb{N}$, car il faut indiquer non seulement par où ils passent mais aussi leur « profondeur » en chaque point), tel que toute suite maximale de dépilements ait pour image (c'est-à-dire ensemble de termes) $\mathcal{C}$ tout entier. Moralement, tous ces cycles sont disjoints, donc tous les dépilements de cycles « commutent ».

Plus rigoureusement, soit C un cycle d'un graphe visible à un moment donné, et $(C_1, \ldots, C_n)$ une suite de cycles, ne contenant pas C, qu'on peut dépiler en partant de ce même graphe visible. Montrons, par récurrence, qu'après avoir dépilé tous les C_i, on peut encore dépiler C. Le cas $n = 0$ est trivial. Sinon, à l'étape i, on constate que C_i est disjoint de C : on peut donc dépiler C_i et continuer grâce à l'hypothèse de récurrence.

Soient deux suites maximales qui n'ont pas le même ensemble de termes. En considérant le premier cycle de l'une qui n'appartient pas à l'autre, on aboutit à une contradiction.

Il est facile de voir que l'algorithme termine presque sûrement, en considérant sa version initiale (toute marche aléatoire sur un ensemble fini correspond à une chaîne de Markov récurrente, donc visite presque sûrement tous les sommets). Reste à vérifier que le résultat a la bonne loi.

Étant donnée une valeur P des compteurs, on pose :

$$\begin{aligned} M_{<P} &= (M_{s,p})_{\substack{s\in S\setminus\{s_0\}\\ p<P_s}} && \text{(la partie de } M \text{ « au-dessus » de } P\text{)}\\ M_P &= (M_{s,P_s})_{s\in S\setminus\{s_0\}} && \text{(le graphe situé à la profondeur } P\text{)}\\ C(M) &= M_{<P^{\mathrm{fin}}(M)} && \text{(la partie à dépiler)}\\ A(M) &= M_{P^{\mathrm{fin}}(M)} && \text{(l'arbre final)} \end{aligned}$$

Fixons une valeur $A^0 = (A^0_s)_{s\in S\setminus\{s_0\}}$ de l'arbre final. On veut montrer que $\mathbb{P}(A(M) = A^0)$ est constante. Pour le faire, fixons une valeur P^0 des compteurs P, et une valeur $C^0 = (C^0_{s,p})_{\substack{s\in S\setminus\{s_0\}\\ p<P^0_s}}$ de la partie à dépiler. On suppose que C^0 est vraiment « dépilable », c'est-à-dire qu'il existe une valeur de M telle que $C(M) = C^0$. On s'intéresse alors à la probabilité conditionnelle

$$\mathbb{P}(A(M) = A^0 | C(M) = C^0).$$

On a $C(M) = C^0$ si et seulement si $P^{\mathrm{fin}}(M) = P^0$ et $M_{<P^0} = C^0$. Remarquons qu'avec la propriété qu'on a exigé de C^0, et en vertu du lemme ci-dessus, si $M_{<P^0} = C^0$ et M_{P^0} est un arbre, alors $P^{\mathrm{fin}}(M) = P^0$; c'est donc encore une condition équivalente. Sous cette hypothèse, on a de plus $A(M) = A^0$ si et seulement si $M_{P^0} = A^0$, puisque A^0 est effectivement un arbre. On a donc

$$\mathbb{P}(A(M) = A^0 | C(M) = C^0) = \mathbb{P}(M_{P^0} = A^0 | M_{<P^0} = C^0 \text{ et } M_{P^0} \text{ est un arbre}).$$

Puisque P^0 est fixé, les variables M_{P^0} et $M_{<P^0}$ sont indépendantes, donc on obtient en fait

$$\mathbb{P}(A(M) = A^0 | C(M) = C^0) = \mathbb{P}(M_{P^0} = A^0 | M_{P^0} \text{ est un arbre}).$$

De plus, la loi M_{P^0} ne dépend pas de P^0 :

$$\mathbb{P}(M_{P^0} = A^0 | M_{P^0} \text{ est un arbre}) = \text{const} = \mathbb{P}(M_0 = A^0 | M_0 \text{ est un arbre}),$$

d'où finalement

$$\mathbb{P}(A(M) = A^0) = \mathbb{P}(M_0 = A^0 | M_0 \text{ est un arbre}).$$

L'arbre donné par l'algorithme a bien une loi uniforme. En particulier, cette loi ne dépend pas de l'ordre des sommets qu'on a fixé ; on en déduit que, même si cet ordre est choisi aléatoirement et quelle que soit sa loi, l'arbre aura une loi uniforme.

Remarque. Tout ce qu'on a fait s'adapte mot pour mot aux graphes orientés, et on peut même l'adapter aux graphes avec des arêtes pondérées (auquel cas on n'obtient plus un arbre uniforme, mais un arbre dont la probabilité est proportionnelle au produit des poids de ses arêtes.)

3.3 Temps d'exécution sur le graphe complet

On établit ici un résultat assez surprenant :

Proposition 14. *Le temps d'exécution de l'algorithme de Wilson sur un graphe complet à n sommets est, en moyenne, en $\underset{n\to\infty}{O}(n)$.*

Preuve. On fixe une fois pour toutes $n \geq 1$.

On commence par remarquer que l'ensemble des états par lesquels l'algorithme peut passer forme une chaîne de Markov dont il suffit d'estimer le temps moyen de parcours pour obtenir le résultat.

Plus précisément, pour décrire l'état de la mémoire à un moment donné, on pose :

- $t \in \mathbb{N}$ l'étape de l'exécution de l'algorithme ;
- $\mathcal{A}(t)$ la partie de l'arbre déjà construite à l'étape t, et $a(t) = |S_{\mathcal{A}(t)}|$ le nombre de sommets déjà incorporés ;
- $\mathcal{B}(t) = (\mathcal{B}_0(t), \ldots, \mathcal{B}_{b(t)}(t))$ la future nouvelle branche qu'on est en train de construire, et $b(t)$ sa longueur.

Le sommet $\mathcal{B}_{b(t)}(t)$ joue un rôle particulier : c'est celui que l'algorithme est en train d'examiner, c'est donc le seul auquel il ne pourra pas revenir à l'étape suivante. C'est pour cette raison qu'on n'a pas compté ce sommet, c'est-à-dire qu'on a pris $b(t)$ et non pas $b(t)+1$ comme longueur.

Remarque. Ici, « à l'étape t » signifie en fait « juste avant le t-ième saut de la marche aléatoire ». On considère donc qu'un effacement de boucle, une extension de l'arbre ou le choix d'un nouveau sommet de départ ne constituent pas une étape séparée, mais sont exécutés à la fin d'une étape. Ainsi, $\mathcal{B}_{b(t)}(t)$ est toujours en dehors de $\mathcal{A}(t)$ et distinct des autres $\mathcal{B}_i(t)$. Pour limiter les disjonctions de cas pénibles au minimum, on convient que, avant de terminer l'algorithme, on se place sur un sommet fictif qui ne fait pas partie du graphe, et on reste indéfiniment dans cet état. Avec cette convention, on a, pour tout $t \geq t_{\text{fin}}$, $a(t) = n$ et $b(t) = 0$, où t_{fin} désigne le premier instant où $a(t)$ atteint n.

Fixons un instant t tel que $a(t) < n$. A l'étape $t+1$, on va visiter un voisin uniforme de $\mathcal{B}_{b(t)}(t)$. Comme le graphe est complet, c'est en fait un sommet tiré uniformément dans $S \setminus \{\mathcal{B}_{b(t)}(t)\}$. Appelons s ce nouveau sommet ; il peut alors se passer trois choses :

- Avec probabilité $\frac{a(t)}{n-1}$, $s \in \mathcal{A}(t)$. Alors on adjoint $\mathcal{B}$ à $\mathcal{A}$, puis on choisit pour $\mathcal{B}$ un nouveau point de départ $\mathcal{B}_0(t+1)$ qui n'est pas encore dans l'arbre. On obtient $a(t+1) = a(t) + b(t) + 1$ (car il faut aussi compter $\mathcal{B}_{b(t)}(t)$) et $b(t+1) = 0$.
- Pour tout i tel que $0 \leq i < b(t)$, avec probabilité $\frac{1}{n-1}$, $s = \mathcal{B}_i(t)$. Alors on efface la boucle et on continue, ce qui mène à $a(t+1) = a(t)$ et $b(t+1) = i$.
- Enfin, avec probabilité $\frac{n-1-a(t)-b(t)}{n-1}$, s est un des sommets restants. Alors on étend simplement $\mathcal{B}$ et on poursuit, ce qui donne $a(t+1) = a(t)$ et $b(t+1) = b(t)+1$.

Si $a(t) = n$, par convention, on a $a(t+1) = a(t) = n$ et $b(t+1) = b(t) = 0$ avec probabilité 1.

On remarque alors qu'on n'a même pas besoin de décrire entièrement l'état de la mémoire : le couple $X(t) = (a(t), b(t))$ contient toutes les informations nécessaires et a déjà la loi d'une chaîne de Markov. Résumons tout ce qu'on sait dessus :

- $X(0) = (1, 0)$.
- $\forall t \in \mathbb{N}, X(t) \in \mathcal{X}$, où on pose

$$\mathcal{X} = \{(a, b) | a \geq 1, b \geq 0, a + b \leq n - 1\} \cup \{(n, 0)\}.$$

– Si $a < n$, on a :

$$\begin{cases} \mathbb{P}((a,b) \to (a,b+1)) = \frac{n-1-a-b}{n-1} & \text{pour } b < n-1-a \\ \mathbb{P}((a,b) \to (a,i)) = \frac{1}{n-1} & \text{pour } 0 \leq i < b \\ \mathbb{P}((a,b) \to (a+b+1,0) = \frac{a}{n-1} & \\ \mathbb{P}((a,b) \to (a',b') = 0 & \text{sinon.} \end{cases}$$

– Enfin, on a :

$$\begin{cases} \mathbb{P}((n,0) \to (n,0)) = 1 \\ \mathbb{P}((n,0) \to (a',b')) = 0 \quad \text{sinon,} \end{cases}$$

où $\mathbb{P}((a,b) \to (a',b'))$ désigne la matrice de transition : c'est un raccourci pour dire $\mathbb{P}(X(t+1) = (a',b')|X(t) = (a,b))$. On voit donc que le seul état récurrent de cette chaîne, qui est même un état absorbant, est l'état final $X_{\text{fin}} = (n,0)$. On va donc chercher le temps moyen de parcours $\mathbb{E}(t_{\text{fin}})$.

On introduit alors $T_{a,b}$ le temps moyen de parcours en partant de l'état (a,b). La valeur qu'on veut déterminer est en fait $T_{1,0}$, et on dispose d'un système de relations récursives :

$$\begin{cases} T_{a,b} = 1 + \sum_{(a',b') \in \mathcal{X}} \mathbb{P}((a,b) \to (a',b')) T_{a',b'} \quad \text{pour } a < n \\ T_{n,0} = 0, \end{cases}$$

soit concrètement, pour $a < n$:

$$T_{a,b} = 1 + \frac{a}{n-1} T_{a+b+1,0} + \frac{n-1-a-b}{n-1} T_{a,b+1} + \frac{1}{n-1} \sum_{i=0}^{b-1} T_{a,i}.$$

Remarque. Si $b = n-1-a$, $T_{a,b+1}$ ne fait pas partie de l'ensemble d'états qu'on a déclaré, mais comme la probabilité d'y passer est alors nulle, ce n'est pas un problème.

On peut encore réécrire cette égalité :

$$-\sum_{i=0}^{b-1} T_{a,i} + (n-1) T_{a,b} - (n-1-a-b) T_{a,b+1} = n-1 + a T_{a+b+1,0}.$$

En faisant varier b de 0 à $n-1-a$, on reconnaît ici une identité matricielle :

$$\begin{pmatrix} n-1 & -(n-1-a) & 0 & \cdots & 0 \\ -1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -2 & 0 \\ \vdots & & \ddots & \ddots & -1 \\ -1 & \cdots & \cdots & -1 & n-1 \end{pmatrix} \begin{pmatrix} T_{a,0} \\ T_{a,1} \\ \vdots \\ \vdots \\ T_{a,n-1-a} \end{pmatrix} = \begin{pmatrix} n-1+aT_{a+1,0} \\ n-1+aT_{a+2,0} \\ \vdots \\ \vdots \\ n-1+aT_{n,0} \end{pmatrix}.$$

On pose :

$$M_k = \begin{pmatrix} 0 & k-1 & & & \\ 1 & \ddots & \ddots & & \\ \vdots & \ddots & \ddots & 2 & \\ \vdots & & \ddots & \ddots & 1 \\ 1 & \cdots & \cdots & 1 & 0 \end{pmatrix}.$$

On a donc

$$\begin{pmatrix} T_{a,0} \\ \vdots \\ T_{a,n-1-a} \end{pmatrix} = ((n-1)I_{n-a} - M_{n-a})^{-1} \begin{pmatrix} n-1+aT_{a+1,0} \\ \vdots \\ n-1+aT_{n,0} \end{pmatrix},$$

ce qui nous donne une relation de récurrence sur les $T_{a,0}$. Cependant, avant de pouvoir l'exploiter, il faut arriver à inverser la matrice qui apparaît ci-dessus ; et avant d'inverser la matrice, on aura d'abord besoin de calculer son déterminant.

3.3.1 Calcul du polynôme caractéristique de M_k

Pour cela, on va introduire une autre famille de matrices : on pose, pour tout $k \in \mathbb{N}$,

$$P_k = |XI_k - M_k|$$

et pour tout $k > 0$

$$\tilde{P}_k = \left|\begin{array}{c|ccc} -1 & -(k-1) & 0 & \cdots & 0 \\ \hline -1 & & & & \\ \vdots & & XI_{k-1} - M_{k-1} & & \\ -1 & & & & \end{array}\right|.$$

Par récurrence, on va démontrer simultanément les deux identités suivantes :

$$\begin{cases} \forall k \in \mathbb{N}, P_k = (X+1)^{k-1}(X-(k-1)) \\ \forall k > 0, \tilde{P}_k = -(X+1)^{k-1}. \end{cases}$$

En effet :

- $P_0 = 1 = (X+1)^{-1}(X-(-1))$.
- $P_1 = |X| = X = (X+1)^0(X-0)$.
- $\tilde{P}_1 = |-1| = -1 = -(X+1)^0$.
- Soit $k > 1$, et supposons l'égalité vraie au rang $k-1$. Alors on décompose sur la première

ligne :

$$
\begin{aligned}
P_k &= \left|\begin{array}{cc|ccccc}
X & -(k-1) & 0 & \cdots & \cdots & 0 \\
-1 & X & -(k-2) & 0 & \cdots & 0 \\
\hline
-1 & -1 & & & & \\
\vdots & \vdots & & XI_{k-2} - M_{k-2} & & \\
-1 & -1 & & & &
\end{array}\right| \\
&= X \left|\begin{array}{c|cccc}
X & -(k-2) & 0 & \cdots & 0 \\
\hline
-1 & & & & \\
\vdots & & XI_{k-2} - M_{k-2} & & \\
-1 & & & &
\end{array}\right| + (k-1) \left|\begin{array}{c|cccc}
-1 & -(k-2) & 0 & \cdots & 0 \\
\hline
-1 & & & & \\
\vdots & & XI_{k-2} - M_{k-2} & & \\
-1 & & & &
\end{array}\right| \\
&= XP_{k-1} + (k-1)\tilde{P}_{k-1} \\
&= X(X+1)^{k-2}(X-(k-2)) - (k-1)(X+1)^{k-2} \\
&= (X^2 - (k-2)X - (k-1))(X+1)^{k-2} \\
&= (X+1)(X-(k-1))(X+1)^{k-2} \\
&= (X+1)^{k-1}(X-(k-1)),
\end{aligned}
$$

et de même

$$
\begin{aligned}
\tilde{P}_k &= -P_{k-1} + (k-1)\tilde{P}_{k-1} \\
&= -(X+1)^{k-2}(X-(k-2)) - (k-1)(X+1)^{k-2} \\
&= (-X + (k-2) - (k-1))(X+1)^{k-2} \\
&= -(X+1)(X+1)^{k-2} \\
&= -(X+1)^{k-1}.
\end{aligned}
$$

3.3.2 Expression explicite de la relation de récurrence

On a :

$$
\begin{pmatrix} T_{a,0} \\ \vdots \\ T_{a,n-1-a} \end{pmatrix} = ((n-1)I_{n-a} - M_{n-a})^{-1} \begin{pmatrix} n-1+aT_{a+1,0} \\ \vdots \\ n-1+aT_{n,0} \end{pmatrix}.
$$

On remarque d'abord que la somme des coefficients d'une ligne de M_k vaut toujours $k-1$, d'où

$$
((n-1)I_{n-a} - M_{n-a}) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = ((n-1) - (n-a-1)) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = a \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}
$$

et donc

$$
\begin{pmatrix} T_{a,0} \\ \vdots \\ T_{a,n-1-a} \end{pmatrix} = \frac{n-1}{a} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} + a((n-1)I_{n-a} - M_{n-a})^{-1} \begin{pmatrix} T_{a+1,0} \\ \vdots \\ T_{n,0} \end{pmatrix}.
$$

En particulier, on a donc

$$T_{a,0} = \frac{n-1}{a} + a\sum_{i=1}^{n-a}(((n-1)I_{n-a} - M_{n-a})^{-1})_{1,i}T_{a+i,0}.$$

Reste à calculer les coefficients de la matrice inverse qui nous intéressent. Soit i un indice tel que $1 \leq i \leq n-a$; on a alors :

$$(((n-1)I_{n-a} - M_{n-a})^{-1})_{1,i} = (-1)^{i+1}\frac{\mathbf{M}_{i,1}((n-1)I_{n-a} - M_{n-a})}{|(n-1)I_{n-a} - M_{n-a}|},$$

où $\mathbf{M}_{i,j}$ désigne le mineur obtenu en enlevant la i-ième ligne et la j-ième colonne. Calculons-le explicitement :

$$\begin{aligned}
\mathbf{M}_{i,1}((n-1)I_{n-a} - M_{n-a}) &= \left|\begin{array}{ccc|c} -(n-a-1) & & & \\ * & \ddots & & 0 \\ * & * & -(n-a-i+1) & \\ \hline * & * & * & (n-1)I_{n-a-i} - M_{n-a-i} \end{array}\right| \\
&= \left(\prod_{j=1}^{i-1}(-(n-a-j))\right) P_{n-a-i}(n-1) \\
&= (-1)^{i-1}\frac{(n-a-1)!}{(n-a-i)!}n^{n-a-i-1}(a+i),
\end{aligned}$$

donc

$$\begin{aligned}
(((n-1)I_{n-a} - M_{n-a})^{-1})_{1,i} &= \frac{(n-a-1)!}{(n-a-i)!}\frac{n^{n-a-i-1}(a+i)}{n^{n-a-1}a} \\
&= \frac{(n-a-1)!}{(n-a-i)!}\frac{a+i}{n^i a}.
\end{aligned}$$

On obtient finalement une forme explicite de la relation de récurrence voulue :

$$T_{a,0} = \frac{n-1}{a} + a\sum_{i=1}^{n-a}\frac{(n-a-1)!}{(n-a-i)!}\frac{a+i}{n^i a}T_{a+i,0}.$$

3.3.3 Résolution de la relation de récurrence

On va vérifier que la solution est $T_{a,0} = \frac{(n-a)(n-1)(a+1)}{na}$. On procède par récurrence descendante sur a (comme le suggère la forme de la relation de récurrence) :

- $T_{n,0} = 0 = \frac{(n-n)(n-1)(n+1)}{n^2}$.
- Supposons-le vrai au rang $a+1$. On a alors (en adoptant la convention usuelle $\frac{1}{n!} = 0$ pour

$n < 0$) :

$$\begin{aligned}
T_{a,0} &= \frac{n-1}{a} + a\sum_{i=1}^{n-a} \frac{(n-a-1)!}{(n-a-i)!}\frac{a+i}{n^i a}\frac{(n-a-i)(n-1)(a+i+1)}{n(a+i)} \\
&= \frac{n-1}{a}\left(1 + a\sum_{i=1}^{n-a} \frac{(n-a-1)!}{(n-a-i-1)!}\frac{(a+i+1)}{n^{i+1}}\right) \\
&= \frac{n-1}{a}\left(1 + a\sum_{i=1}^{n-a} \frac{(n-a-1)!}{(n-a-i-1)!}\frac{(n-(n-a-i-1))}{n^{i+1}}\right) \\
&= \frac{n-1}{a}\left(1 + a\left(\sum_{i=1}^{n-a} \frac{(n-a-1)!}{(n-a-i-1)!}\frac{1}{n^i} - \sum_{i=1}^{n-a} \frac{(n-a-1)!}{(n-a-i-2)!}\frac{1}{n^{i+1}}\right)\right) \\
&= \frac{n-1}{a}\left(1 + a\frac{n-a-1}{n}\right) \\
&= \frac{n-1}{a}\frac{n+an-a^2-a}{n} \\
&= \frac{(n-1)(n-a)(a+1)}{an}.
\end{aligned}$$

En posant $a = 1$, on trouve finalement la formule étonnamment simple

$$\mathbb{E}(t_{\text{fin}}(n)) = 2\frac{(n-1)^2}{n}.$$

Asymptotiquement, cela donne

$$\mathbb{E}(t_{\text{fin}}(n)) \underset{n\to\infty}{\sim} 2n.$$

ce qui veut dire qu'en moyenne, l'algorithme visite chaque sommet seulement 2 fois !

Dans cette estimation du temps d'exécution, on a négligé les effacements de boucles, les extensions de l'arbre et les choix des nouveaux sommets. Pour les deux derniers points, il est facile de voir qu'on a un nombre constant d'opérations élémentaires par sommet du graphe, donc le tout se fait en temps linéaire. Quant aux effacements de boucles, lors de la preuve de l'algorithme, on a vu qu'ils prenaient autant de temps (à un terme linéaire près) que le parcours du graphe proprement dit. La complexité totale de l'algorithme est donc bien linéaire en n.

A Implémentation des algorithmes en OCaml

A.1 Aldous-Broder

```
type dir = N | E | S | W
type vertex = int * int
let random_dir () = match Random.int 4 with
  | 0 -> N
  | 1 -> E
  | 2 -> S
  | 3 -> W
let move dir (i, j) = match dir with
  | N -> (i-1, j)
  | E -> (i, j+1)
  | S -> (i+1, j)
  | W -> (i, j-1)

let n = 10
let m = 10

let seen = Array.init n (fun i -> Array.make m false)
let seen_counter = ref 0
let laby = ref []
let cursor = ref (0, 0)

let is_valid (i, j) = i >= 0 && i < n && j >= 0 && j < m
let rec random_neighbour v =
  let res = move (random_dir ()) v in
  if is_valid res then res else random_neighbour v
(* On tire un des quatre voisins potentiels de v. S'il sort de la grille, on réessaye. *)

let main = begin
Random.self_init ();
seen.(0).(0) <- true;
incr seen_counter;
while !seen_counter < n*m do
  let (i, j) = !cursor in
  assert (is_valid (i, j));
  let (i', j') = random_neighbour (i, j) in
  if not seen.(i').(j') then begin
    laby := ((i, j), (i', j')) :: !laby;
    seen.(i').(j') <- true;
    incr seen_counter
  end;
  cursor := (i', j')
```

```
done;
end
```

A.2 Wilson

```
(* Définition du graphe par listes d'adjacences *)

let make_neighbours n m =
        let dirs = [(0,1) ; (0, -1) ; (1, 0) ; (-1, 0)] in
        let add_vec (x,y) (dx, dy) = (x + dx, y + dy) in
        let valid_coord n x = x >= 0 && x < n in
        let valid_pos (x,y) = valid_coord n x && valid_coord m y in

        fun pos ->
                let candidates = List.map (add_vec pos) dirs in
                List.filter valid_pos candidates
;;

(* Accès à un élément aléatoire d'un liste *)

let rand_elt xs =
        let len = List.length xs in
        let i = Random.int len in
                List.nth xs i
;;

(** Début du programme *)

let n,m = 10,10 ;;
let neighbours = make_neighbours n m;;
let rand_neighbour pos = rand_elt (neighbours pos);;

(* Marquage des sommets *)

type mark = White | Grey | Black ;;
let current = ref 0 ;;
let bound = n * m ;;
let marked = Array.make_matrix n m White ;;
let edges = ref [] ;;
let time = ref 0 ;;

exception Finished ;;

let next () =
        while marked.(!current / m).(!current mod m) <> White do
```

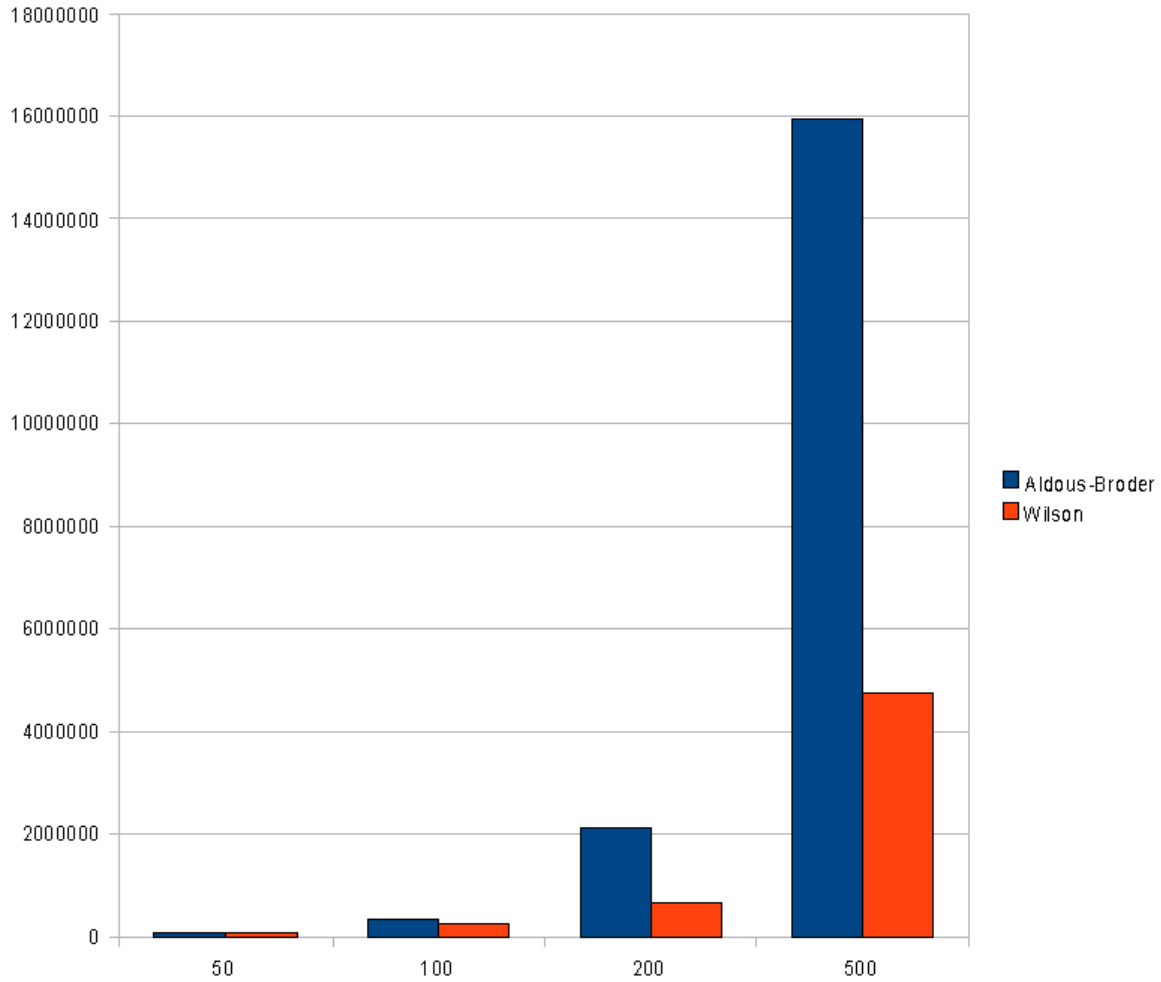
```
                incr current ;
                if !current >= bound then raise Finished
        done ;
        (!current / m, !current mod m)
;;

let rec erase pos = function
        | (x,y) :: q ->
                (* Printf.printf "effacement de %d,%d\n" x y; *)
                if  (x,y) = pos
                        then pos :: q
                        else begin
                                marked.(x).(y) <- White ;
                                erase pos q
                        end
        | [] -> []
;;

let rec to_edges p = function
        | [] -> ()
        | (x,y)::ps' ->
                marked.(x).(y) <- Black ;
                edges := (p,(x,y))::(!edges) ;
                to_edges (x,y) ps'
;;

let rec loop_erased_walk path =
        let (x,y) = rand_neighbour (List.hd path) in
        incr time;
        (* Printf.printf "voisin au hasard : %d,%d\n" x y; *)
        match marked.(x).(y) with
                | Black -> to_edges (x,y) path
                | Grey -> loop_erased_walk (erase (x,y) path)
                | White ->
                        marked.(x).(y) <- Grey ;
                        loop_erased_walk ((x,y) :: path)
;;

let wilson () =
        Random.self_init ();
        marked.(0).(0) <- Black;
        try
                while true do
                let (x,y) = next () in
                incr time;
```

```
                (* Printf.printf "point de départ : %d,%d\n" x y; *)
                marked.(x).(y) <- Grey;
                loop_erased_walk [(x,y)]
                done
        with
                | Finished -> Printf.printf "Temps de parcours : %d\n" !time
;;
```

A.3 Comparaison expérimentale

On donne pour chaque taille de graphe le temps moyen (nombre d'étapes) réalisé par les deux algorithmes sur 20 simulations :

Taille	50×50	100×100	200×200	500×500
Aldous-Broder	66734	337359	2124577	15959901
Wilson	43244	253756	634284	4765663



B Théorème de Kirchhoff

Définition 11. Soit $G = (S, A)$ un graphe connexe fini pour lequel on fixe une orientation arbitraire de ses arêtes. On définit alors :

- la *matrice d'incidence* $I_G = (i_{s,a})_{(s,a)\in S\times A}$ où $i_{s,a} = \begin{cases} -1 & \text{si } s \text{ est la source de } a \\ 1 & \text{si } s \text{ est le but de } a \\ 0 & \text{sinon} \end{cases}$
- la *matrice d'adjacence* $A_G = (a_{s,t})_{(s,t)\in S^2}$ où $a_{s,t} = \begin{cases} 1 & \text{si } s \text{ et } t \text{ sont adjacents} \\ 0 & \text{sinon} \end{cases}$
- la *matrice des degrés* $D_G = (d_{s,t})_{(s,t)\in S^2}$ où $d_{s,t} = \begin{cases} \deg(s) & \text{si } s = t \\ 0 & \text{sinon} \end{cases}$
- la *matrice Laplacienne* $L_G = (l_{s,t})_{(s,t)\in S^2}$ où $l_{s,t} = \begin{cases} \deg(s) & \text{si } s = t \\ -1 & \text{si } s \text{ et } t \text{ sont adjacents} \\ 0 & \text{sinon} \end{cases}$

Théorème 5. *Pour tout graphe G connexe fini, le nombre $N(G)$ d'arbre couvrants de G est donné par un mineur diagonal quelconque de sa matrice Laplacienne.*

B.1 Démonstration

On commence par introduire et démontrer la formule de Binet-Cauchy, résultat classique sur les déterminants dont on aura besoin pour la preuve.

Si n, p, r, s sont quatre entiers tels que $r \leq n$ et $s \leq p$, $M \in \mathcal{M}_{n,p}(\mathbb{R})$, $1 \leq i_1 < \cdots < i_r \leq n$ et $1 \leq j_1 < \cdots < j_s \leq p$, on note $M\begin{pmatrix} i_1 & \dots & i_r \\ j_1 & \dots & j_s \end{pmatrix}$ la sous-matrice extraite de M dont on n'a gardé que les lignes $i_1, \dots, i_r$ et les colonnes $j_1, \dots, j_s$. On notera également $M_1, \dots, M_p$ les vecteurs colonne de M.

Lemme 2 (formule de Binet-Cauchy)**.** *Soient $m, n, p, r \in \mathbb{N}$ quatre entiers vérifiant $r \leq \min(n, p)$, $P \in \mathcal{M}_{n,m}(\mathbb{R})$ et $Q \in \mathcal{M}_{m,p}(\mathbb{R})$ deux matrices, et $1 \leq i_1 < \cdots < i_r \leq n$, $1 \leq j_1 < \cdots < j_r \leq p$ des entiers. On a l'égalité suivante*

$$\left|(PQ)\begin{pmatrix} i_1 & \dots & i_r \\ j_1 & \dots & j_r \end{pmatrix}\right| = \sum_{1\leq k_1<\cdots<k_r\leq m} \left|P\begin{pmatrix} i_1 & \dots & i_r \\ k_1 & \dots & k_r \end{pmatrix}\right| \left|Q\begin{pmatrix} k_1 & \dots & k_r \\ j_1 & \dots & j_r \end{pmatrix}\right|$$

Preuve. Comme $\mathrm{rg}(PQ) \leq \min(\mathrm{rg}(P), \mathrm{rg}(Q)) \leq m$ le cas où $m < r$ se ramène simplement à l'égalité $0 = 0$. On peut donc supposer que $r \leq m$. Une deuxième simplification consiste à remarquer qu'on n'utilise dans les deux termes de la formule que les lignes $i_1, \dots, i_r$ de P et les colonnes $j_1, \dots, j_r$ de Q, ce qui permet de supposer que $n = p = r$. Il s'agit alors d'évaluer le déterminant de PQ, ce pour quoi on peut faire jouer la multilinéarité :

$$\begin{aligned}|PQ| &= |((PQ)_1, \dots, (PQ)_r)| \\ &= \left| \left(\sum_{k_1=1}^{m} P_{k_1} q_{k_1,1}, \dots, \sum_{k_r=1}^{m} P_{k_r} q_{k_r,r} \right) \right| \\ &= \sum_{1 \leq k_1, \dots, k_r \leq m} q_{k_1,1} \dots q_{k_r,r} |(P_{k_1}, \dots, P_{k_r})|\end{aligned}$$

Or $|(P_{k_1}, \dots, P_{k_r})|$ est nul dès que les $k_1, \dots, k_r$ ne sont pas tous distincts. S'ils le sont, on pose σ la permutation de $\mathfrak{S}_r$ telle que les $j_i = k_{\sigma(i)}$ sont dans l'ordre croissant, auquel cas le déterminant précédent vaut $\varepsilon(\sigma)\,|(P_{j_1}, \dots, P_{j_r})|$. Il vient alors

$$\begin{aligned}|PQ| &= \sum_{1 \leq j_1 < \dots < j_r \leq m} \sum_{\sigma \in \mathfrak{S}_r} \varepsilon(\sigma) q_{j_1,\sigma(1)} \dots q_{j_r,\sigma(r)} |(P_{j_1}, \dots, P_{j_r})| \\ &= \sum_{1 \leq j_1 < \dots < j_2 \leq m} \left| P \begin{pmatrix} 1 & \dots & r \\ j_1 & \dots & j_r \end{pmatrix} \right| \left| Q \begin{pmatrix} j_1 & \dots & j_r \\ 1 & \dots & r \end{pmatrix} \right|\end{aligned}$$

ce qui achève la preuve. □

Le lemme suivant se montre par un simple calcul matriciel et permet de relier les différentes matrices caractéristiques d'un graphe donné.

Lemme 3. *Pour tout graphe G non forcément connexe et toute orientation de ses arêtes on a*

$$I_G {}^t I_G = L_G = D_G - A_G$$

Revenons à la preuve du théorème lui-même. Soit donc $G = (S, A)$ un graphe (connexe) à n sommets (comme on la fait remarquer plus tôt il a au moins $n-1$ arêtes). Pour tout sommet $s \in S$ on note L_s la sous-matrice extraite de L_G dont on a retiré la s-ième ligne et la s-ième colonne, et I_s la sous-matrice extraite de I_G dont on a retiré la s-ième ligne. D'après la formule de Binet-Cauchy, en notant $M_{n-1}(I_s)$ l'ensemble des sous-matrices $(n-1) \times (n-1)$ de I_s, on peut écrire

$$|L_s| = \sum_{J \in M_{n-1}(I_s)} |J| |{}^t J| = \sum |J|^2$$

Afin d'évaluer la somme, remarquons qu'on peut faire correspondre chaque sous-matrice $(n-1) \times (n-1)$ de I_s avec le sous-graphe $\phi(J)$ de G dont l'ensemble des sommets est S mais dont on ne garde que $n-1$ arêtes (correspondant aux $n-1$ colonnes).

Lemme 4. *Pour toute sous-matrice J de taille $(n-1) \times (n-1)$ et extraite de I_s on a*

$$|J|^2 = \begin{cases} 1 & \textit{si } \phi(J) \textit{ est couvrant} \\ 0 & \textit{sinon} \end{cases}$$

Preuve. Si $\phi(J)$ est couvrant alors c'est un arbre puisqu'il a exactement $n-1$ arêtes. Il contient donc au moins un sommet de degré 1 distinct de s que l'on notera s_1. Si on supprime s_1 et

la seule arête a_1 qui lui est incidente on obtient encore un arbre et on peut itérer le processus jusqu'à ce que l'arbre soit réduit à s. Si $s_1, s_2, \ldots, s_{n-1}$ et $a_1, a_2, \ldots, a_{n-1}$ sont respectivement les sommets et les arêtes supprimés à la première, la seconde, ..., la n-ième étape, alors dans les bases $(s_1, s_2, \ldots, s_{n-1})$ et $(a_1, a_2, \ldots, a_{n-1})$ la matrice J est triangulaire supérieure avec des ± 1 sur la diagonale. Il vient donc finalement $|J| = \pm 1$.

Maintenant, si $\phi(J)$ n'est pas couvrant, on peut trouver un sommet t qui n'est pas dans la composante connexe de s. Soit T la composante connexe de t dans $\phi(J)$. Alors la sommet des lignes correspondant à T dans J est nulle, et donc la famille des vecteurs lignes de J n'est pas linéairement indépendante. Par conséquent $|J| = 0$. □

En appliquant le lemme la somme précédente devient donc :

$$|L_s| = \sum_{a \in \mathrm{Couv}(G)} 1 = N(G)$$

B.2 Cas des graphes réguliers

Définition 12. Soit $k \in \mathbb{N}$. Un graphe est *k-régulier* si chacun de ses sommets est de degré k.

On commence par donner quelques propriétés du *spectre d'adjacence* (valeurs propres de la matrice d'adjacence) d'un tel graphe.

Lemme 5. *Soit G un graphe connexe k-régulier à n sommets et de spectre d'adjacence $\alpha_1 \geq \cdots \geq \alpha_n$. Alors pour tout $i \in \{1, \ldots, n\}$ on a $|\alpha_i| \leq k$ et $k = \alpha_1$ est valeur propre de multiplicité 1.*

Preuve. Soit λ une valeur propre de A_G et $v = (v_s)_{s \in S}$ un vecteur propre associé à λ. Soit t tel que $|v_t| = \max_{s \in S} |v_s|$. On peut supposer que $v_t > 0$ quitte à remplacer v par $-v$ et on a alors :

$$|\lambda| v_t = |\lambda v_t| = \left| \sum_{s \in S} a_{t,s} v_s \right| \leq \sum_{s \in S} a_{t,s} |v_s| \leq k v_t$$

D'où la première partie de l'énoncé. Pour la seconde partie il est clair que tout vecteur constant non nul est vecteur propre de A_G associé à la valeur propre k. Par ailleurs, en remplaçant λ par k dans l'inégalité précédente on voit que pour les k voisins s de t on a $v_s = v_t$ et on en déduit par récurrence sur la distance à t que v est constant. k est donc de multiplicité 1. □

Théorème 6. *Soit G un graphe k-régulier à n sommets et $k = \alpha_1 > \alpha_2 \geq \cdots \geq \alpha_n$ son spectre d'adjacence. Le nombre d'arbres couvrants de G est alors donné par*

$$N(G) = \frac{1}{n} \prod_{i=2}^{n} (k - \alpha_i)$$

Preuve. On exprime de deux manières différentes $c_X(\chi_{L_G})$, le coefficient en X du polynôme caractéristique $\chi_{L_G} = \det(L_G - X I_n)$ de la matrice Laplacienne de G. Si on note $\lambda_1 \leq \cdots \leq \lambda_n$ les valeurs propres de L_G on a d'après les relations coefficients-racines

$$c_X(\chi_{L_G}) = -\sum_{i=1}^{n} \prod_{j \neq i} \lambda_j$$

De plus il se trouve que pour tout $i \in \{1, \ldots, n\}$ on a $\lambda_i = k - \alpha_i$ puisque la matrice D_G est scalaire dans le cas d'un graphe k-régulier. En particulier $\lambda_1 = 0$ et le résultat précédent se simplifie donc en

$$c_X(\chi_{L_G}) = -\prod_{i=2}^{n}(k - \alpha_i)$$

D'autre part, ce coefficient est aussi donné par l'opposé de la somme des mineurs diagonaux de L_G (il suffit par exemple de dériver et de prendre $X = 0$ pour le voir). Mais d'après le théorème précédent tous ces mineurs sont égaux à $N(G)$ et donc le coefficient cherché vaut $-nN(G)$. En égalant ces deux expressions il vient

$$N(G) = \frac{1}{n}\prod_{i=2}^{n}(k - \alpha_i)$$

□

Références

[1] Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK.* Springer-Verlag, 3ème edition, octobre 2003.

[2] David J Aldous. The random walk construction of uniform spanning trees and uniform labelled trees. *SIAM J. Discrete Math.*, 3 :450–465, 1990.

[3] Andrei Broder. Generating random spanning trees. In *Proc. 30'th IEEE Symp. Found. Comp. Sci.*, pages 442–447, 1989.

[4] Philippe Chassaing. Arbres couvrants aléatoires. Thème d'Agrégation.

[5] Thomas H. Cormen, Charles Leiserson, Ron Rivest, and Cliff Stein. *Introduction à l'algorithmique.* Dunod, 2nde edition, 2002.

[6] Geoffrey Grimmett. *Probability on Graphs*, chapter 2. x, 2009.

[7] Vincent Pilaud. Arbres couvrants & théorie algébrique des graphes. Thème d'Agrégation, 2006.

[8] David Bruce Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the 28'th annual ACM symposium on Theory of computing*, pages 296 – 303, 1996.

Nous remercions également Nicolas Curien pour nous avoir encadré et assisté dans notre travail.