Faculté des Sciences appliquées

MATH0488-1 Éléments de Processus Stochastiques

Méthodes de Monte Carlo par chaînes de Markov pour la détection de communautés dans un graphe

Professeur :
Pierre GEURTS

Groupe :
Maxime FIRRINCIELI
Romain LAMBERMONT
Arthur LOUIS

10 mai 2022

Table des matières

Table des figures

Liste des tableaux

Introduction

Ce projet a pour but de détecter les communautés dans un graphe grâce aux chaînes de Markov et par les méthodes de Monte-Carlo. L'utilisation de ces outils nous permettra de détecter efficacement les communautés dans un graphe. Le projet sera constitué en deux parties. Une première se focalisera sur la familiarisation avec les chaînes de Markov et les méthodes de Monte-Carlo, plus précisement l'algorithme de Metropolis-Hastings, tandis que la seconde sera le cœur du projet avec la recherche de communautés dans un graphe.

1 Première partie : chaînes de Markov et algorithme MCMC

1.1 Chaînes de Markov

Rappelons d'abord brievement ce qu'est une chaîne de Markov. Une chaîne de Markov est un outil mathématique stochastique, qui utilise un principe de "non-mémoire". Tout état d'un système est simplement calculé à partir du précédent, ce qui en facilite l'analyse.

Ces chaînes sont simplement décrites mathématiquement comme suit avec $X_1, X_2, ..., X_t$ une suite de variables aléatoires qui définit une chaîne de Markov si (pour t $>$ 1) elle suit cette relation :

$$\mathbb{P}(X_1, X_2, ..., X_t) = \mathbb{P}(X_1) \prod_{l=2}^{t} \mathbb{P}(X_l |\ X_{l-1})$$

1.1.1

Nous calculons donc pour des valeurs de t croissantes les différentes valeurs demandées, ici avec $t = 20$ (suffit pour avoir convergé) :

- Cas de base distribué uniformément : $\mathbb{P}(X_t = x) = \begin{pmatrix} 0.3488 & 0.0698 & 0.2326 & 0.3488 \end{pmatrix}$ avec $x = 1, 2, 3, 4$
- Cas de base fixé : $\mathbb{P}(X_t = x) = \begin{pmatrix} 0.3488 & 0.0698 & 0.2326 & 0.3488 \end{pmatrix}$ avec $x = 1, 2, 3, 4$
- $Q^t = \begin{pmatrix} 0.3488 & 0.0698 & 0.2326 & 0.3488 \\ 0.3488 & 0.0698 & 0.2326 & 0.3488 \\ 0.3488 & 0.0698 & 0.2326 & 0.3488 \\ 0.3488 & 0.0698 & 0.2326 & 0.3488 \end{pmatrix}$

On remarque donc bien une convergence vers des probabilités et ceci peut importe le cas de départ, on peut montrer cette convergence sur les figures ci-dessous 1 :

On remarque également qu'après une vingtaine d'itérations on obtient la relation suivante :

$$Q^t = Q^{t-1}$$

On en déduit donc que $\lim_{t\to\infty} Q^t \approx Q^{20}$, en effet chaque ligne de Q^t est égale à $[0.3488, 0.0698, 0.2326, 0.3488]$. On a déjà remarqué dans les figures 1 et 2 que peut importe le point de départ, on converge vers les mêmes probabilités après un nombre suffisant d'itérations on montre alors avec $\pi_{t,\text{uniforme}}$ représentant la distribution des probabilités au pas de temps t dans le cas d'une distribution de base uniforme et $\pi_{t,\text{fixé}}$ représentant la distribution des probabilités au pas de temps t dans le cas d'une distribution de base fixée sur le cas 3 :

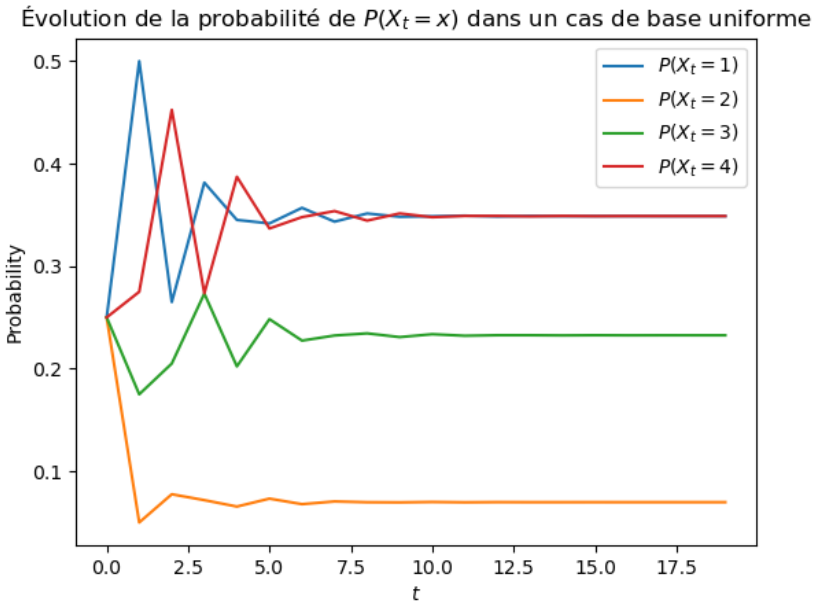

FIGURE 1 – Évolution des probabilités dans une distribution de départ uniforme

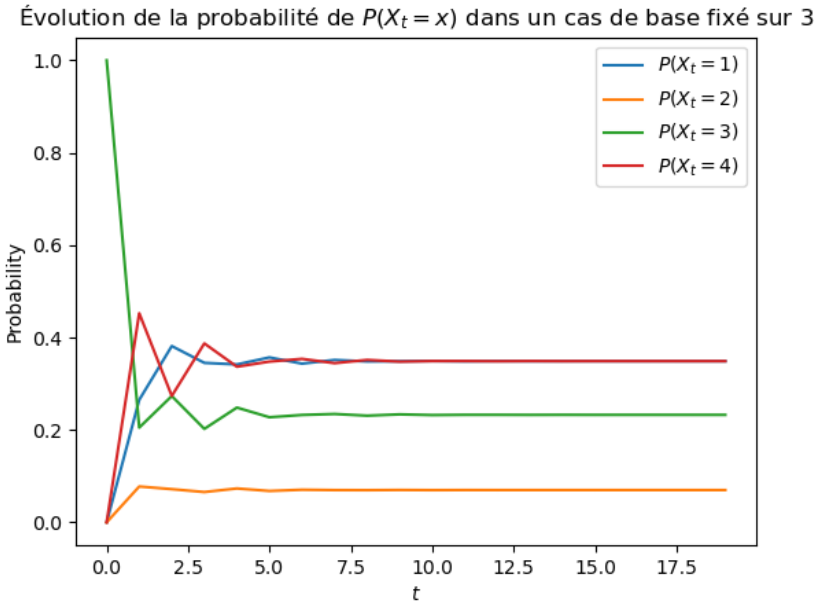

FIGURE 2 – Évolution des probabilités dans une distribution de départ fixée sur 3

$$\pi_{21,\text{uniforme}} = \pi_{20,\text{uniforme}} Q = \pi_{20,\text{uniforme}}$$
$$\pi_{21,\text{fixé}} = \pi_{20,\text{fixé}} Q = \pi_{20,\text{fixé}}$$

On dit d'une distribution qu'elle est stationnaire si $\pi_t = \pi_t Q$, nous considérons donc nos deux

distributions comme stationnaires et elles admettent les propriétés suivantes :

— Ergodicité si elle est égale aux lignes de $\lim_{t\to\infty} Q^t$, comme prouvé dans la sous-section 1.1.4.
— Unicité si la châine de Markov est irréductible.

1.1.2

Afin de déduire la distribution stationnaire π_∞ de notre chaîne qui est décrite comme suit :

$$[\pi_\infty]_j = \lim_{t\to\infty} \mathbb{P}(X_t = j)$$

Nous allons simplement calculer $\mathbb{P}(X_t)$ avec un grand t ce qui nous donne :

$$\pi_\infty = \begin{pmatrix} 0.3488 & 0.0698 & 0.2326 & 0.3488 \end{pmatrix}$$

1.1.3

Afin de vérifier les résultats obtenus, nous effectuons des réalisations de notre chaîne de Markov. Nous pouvons mettre en tableau la proportion de réalisation de chaque état lors de tests avec un nombre de pas croissant. Nous utilisons un point de départ distribué uniformément entre les 4 états car il a été prouvé plus haut que ça n'avait pas d'influence.

# Pas / État	1	2	3	4
100	0.35	0.05	0.21	0.39
1000	0.351	0.075	0.23	0.344
10000	0.346	0.064	0.236	0.354
100000	0.3501	0.0722	0.228	0.3497
1000000	0.34902	0.6757	0.23422	0.34919

TABLE 1 – Proportion des états obtenus en fonction de nombre de pas

En comparant ce tableau avec les résultats obtenus avec les valeurs précédement obtenus, on remarque bien que les proportions sont respectées et font sens, on converge bien vers les mêmes valeurs.

Pour montrer plus de détails, on montre qu'on approche bien des valeurs des lignes de $\lim_{t\to\infty} Q^t$ avec des fluctuations qui s'atténuent plus t augmente :

1.1.4

Après ces premières expériences sur les chaînes de Markov et après avoir remarqué que les distributions sont comme attendu stationnaires, nous pouvons conclure que la chaîne de Markov que nous avons réalisé est ergodique. Une telle chaîne permet d'obtenir la distribution stationnaire de deux manières différentes, soit en effectuant des réalisations différentes, soit en effectuant une chaîne assez longue.

Pour être ergodique, une chaîne doit respecter 3 conditions : avoir un espace finit, être irréductible et être apériodique. Analysons ces 3 conditions dans notre cas précis.

— Nous avons bien un espace finit d'états (1,2,3,4).

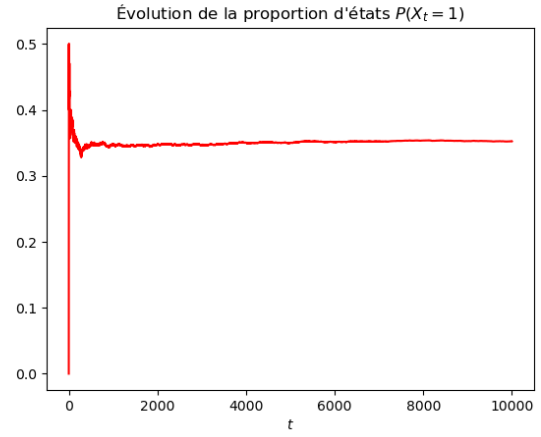

FIGURE 3 – Proportion de l'état 1

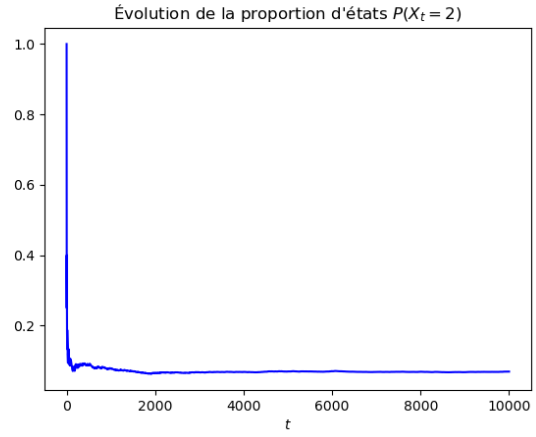

FIGURE 4 – Proportion de l'état 2

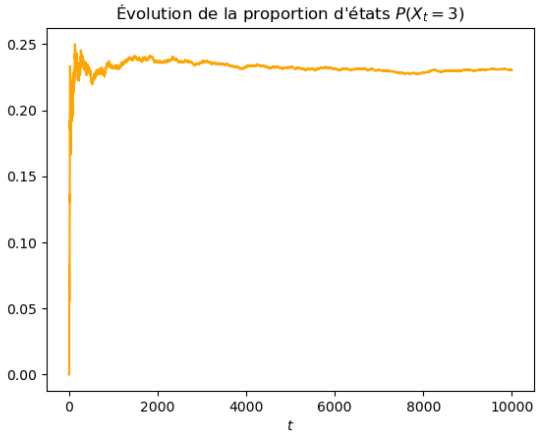

FIGURE 5 – Proportion de l'état 3

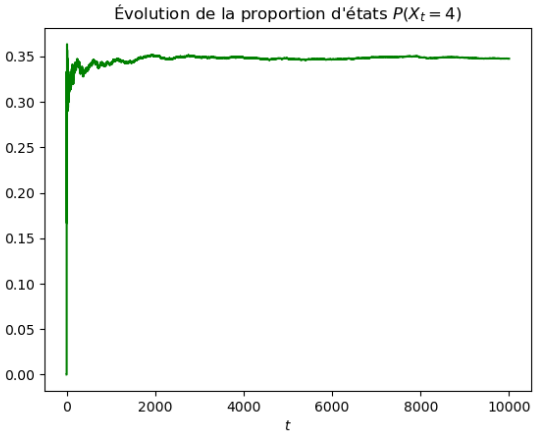

FIGURE 6 – Proportion de l'état 4

— Nous avons bien une chaîne irréductible car la probabilité d'accéder à chaque noeud depuis un autre noeud est non nulle. On peut remarquer cela en observant bien que les noeuds apparaissent plusieurs fois lors de nos réalisations de la chaîne de Markov. On ne reste pas bloqués dans un sous-espace d'états.

— Nous avons bien une chaîne de Markov apériodique car quand on calcule

$$Q = \begin{pmatrix} 0 & 0.1 & 0.1 & 0.8 \\ 1 & 0 & 0 & 0 \\ 0.6 & 0 & 0.1 & 0.3 \\ 0.4 & 0.1 & 0.5 & 0 \end{pmatrix} \quad Q^2 = \begin{pmatrix} 0.48 & 0.08 & 0.41 & 0.03 \\ 0 & 0.1 & 0.1 & 0.8 \\ 0.18 & 0.09 & 0.22 & 0.51 \\ 0.4 & 0.04 & 0.09 & 0.47 \end{pmatrix}$$

$$Q^3 = \begin{pmatrix} 0.338 & 0.051 & 0.104 & 0.507 \\ 0.48 & 0.08 & 0.41 & 0.03 \\ 0.426 & 0.069 & 0.295 & 0.21 \\ 0.282 & 0.087 & 0.284 & 0.347 \end{pmatrix}$$

Pour chaque état i, l'ensemble des nombres $n \in N_0$ tels que $Q^n_{i,i} \neq 0$ est $n = 2, 3, ...$ Son PGCD est 1, la chaîne est donc apériodique.

1.2 Méthode MCMC : analyse théorique dans le cas fini

Nous allons maintenant nous attarder sur l'aspect théorique de l'algorithme de Metropolis-Hastings avant de passer à l'aspect pratique.

1.2.1

On veut montrer que π_0 est une distribution stationnaire sachant que une matrice de transition Q et une distribution initiale π_0 d'une chaîne de Markov invariante dans le temps qui satisfont les équations de balance détaillée :

$$\forall i, j \in \{1, \dots, N\}, \pi_0(i)[Q]_{i,j} = \pi_0(j)[Q]_{j,i}$$

On reprend donc les définitions de π_0 et de Q :

$$\pi_0 = [P(X_0 = x_1) \dots P(X_0 = x_k) \dots P(X_0 = x_n)]$$

$$Q = \begin{pmatrix} P(X_1 = x_1|X_0 = x_1) & \dots & P(X_1 = x_n|X_0 = x_1) \\ \dots & P(X_1 = x_n|X_0 = x_n) & \dots \\ P(X_1 = x_1|X_0 = x_n) & \dots & P(X_1 = x_n|X_0 = x_n) \end{pmatrix}$$

Sachant que la chaîne de Markov est invariante, on a $Q_0 = Q_1 = \dots = Q_n = Q$ et on sait que la matrice de transition et la distribution initiale respectent les équations de balance détaillée :

$$\begin{aligned} \forall i, j \in \{1, \dots, N\}, \pi_0(i)[Q]_{i,j} &= \pi_0(j)[Q]_{j,i} \\ P(X_0 = x_i)P(X_1 = x_j|X_0 = x_i) &= P(X_0 = x_j)P(X_1 = x_i|X_0 = X_j) \\ P(X_0 = x_i)\frac{P(X_1 = x_j, X_0 = x_i)}{P(X_0 = x_i)} &= P(X_0 = x_j)\frac{P(X_1 = x_i, X_0 = x_j)}{P(X_0 = x_j)} \text{ par Bayes} \\ P(X_1 = x_j, X_0 = x_i) &= P(X_1 = x_i, X_0 = x_j) \forall i, j \end{aligned}$$

On sait également que au vu de la forme de π_0 et Q vue plus haut et que $\pi_1 = \pi_0 * Q$:

$$\begin{aligned} P(X_1 = x_j) &= \sum_{k=1}^{n} P(X_0 = x_k)P(X_1 = x_j|X_0 = x_k) \\ P(X_1 = x_j) &= \sum_{k=1}^{n} P(X_0 = x_k)\frac{P(X_1 = x_j, X_0 = x_k)}{P(X_0 = x_k)} \\ P(X_1 = x_j) &= \sum_{k=1}^{n} P(X_1 = x_j, X_0 = x_k) = \sum_{k=1}^{n} P(X_1 = x_k, X_0 = x_j) \\ P(X_1 = x_j) &= \sum_{k=1}^{n} P(X_1 = x_k, X_0 = x_j) = P(X_0 = x_j) \end{aligned}$$

On vérifie cela pour chaque j, ce qui montre que $\pi_0 = \pi_1$ et que π_0 est une distribution stationnaire. Cette distribution est unique si la chaîne de Markov induite par la matrice de transition Q est irréductible.

1.2.2

En remplaçant p_X par $f(x) = cp_X$, on recalcule la probabilté d'acceptation de l'algorithme de Metropolis-Hastings comme suit :

$$\alpha(y_{t+1}, x_t) = \min(1, \frac{cp_X(y_{t+1})}{cp_X(x_t)} \frac{q(x_t|y_{t+1})}{q(y_{t+1}|x_t)})$$

On peut denoter deux cas en renomant le deuxième membre du min en δ :

— $\delta < 1$: On obtient

$$\alpha(y_{t+1}, x_t) = \delta$$
$$\alpha(x_t, y_{t+1}) = \min(1, \frac{1}{\delta}) = 1$$

— $\delta > 1$: On obtient

$$\alpha(y_{t+1}, x_t) = 1$$
$$\alpha(x_t, y_{t+1}) = \delta$$

Peu importe le cas dans lequel on se trouve, on peut réécrire :

$$p_X(x_t)\alpha(y_{t+1}, x_t)q(y_{t+1}|x_t) = p_X(y_{t+1})\alpha(x_t, y_{t+1})q(x_t|y_{t+1})$$

On sait que $q(y_{t+1}|x_t)$ est la probabilité de générer y à l'instant $t+1$ selon la loi instrumentale q sachant que l'on se trouve en x à l'instant t. On sait également que notre probabilité d'acceptation du nouvel élément est α. On sait donc que :

$$\alpha(y_{t+1}, x_t)q(y_{t+1}|x_t) = Q(x_t, y_{t+1})$$

$$\alpha(x_t, y_{t+1})q(x_t|y_{t+1}) = Q(y_{t+1}, x_t)$$

ce qui nous permet d'écrire :

$$p_X(x_t)Q(x_t, y_{t+1}) = p_X(y_{t+1})Q(y_{t+1}, x_t)$$

Cette dernière équation est l'équation de balance détaillée et est vérifiée par notre chaîne de Markov.

1.3 Méthode MCMC : illustration sur un exemple simple

Nous allons maintenant nous attarder à un cas plus précis de l'algorithme de Metropolis-Hastings pour échantilloner des valeurs suivant la distribution binomiale définie comme suit avec p la probabilité de succès :

$$\mathbb{P}(X = k) = C_K^k p^k (1-p)^{K-k}$$

Pour ce faire nous allons utiliser la distribution de proposition énoncée avec $r \in]0, 1[$ comme suit :

$$q(y|x) = \begin{cases} r & \text{si } x = 0 \text{ et } y = 0 \\ (1-r) & \text{si } x = K \text{ et } y = K \\ r & \text{si } 0 < x \leq K \text{ et } y = x - 1 \\ (1-r) & \text{si } 0 \leq x < K \text{ et } y = x + 1 \\ 0 & \text{sinon} \end{cases}$$

1.3.1

Au vu du choix de notre distribution $q(y|x)$ qui est symétrique, on sait que lors du calcul de

$$\frac{p_X(y^{(t)})}{p_X(x^{(t-1)})}\frac{q(x^{(t-1)}|y^{(t)})}{q(y^{(t)}|x^{(t-1)})}$$

La deuxième partie de ce produit se simplifiera au vu de la symétrie de la distribution car

$$q(x^{(t-1)}|y^{(t)}) = q(y^{(t)}|x^{(t-1)})$$

On conclut donc que $q(y|x)$ est un bon choix de distribution pour notre algorithme de Metropolis-Hastings comme vu au point 1.2.2 .

1.3.2

Nous allons maintenant réaliser une grande réalisation de notre chaîne de Markov afin de constater la présence d'une convergence :

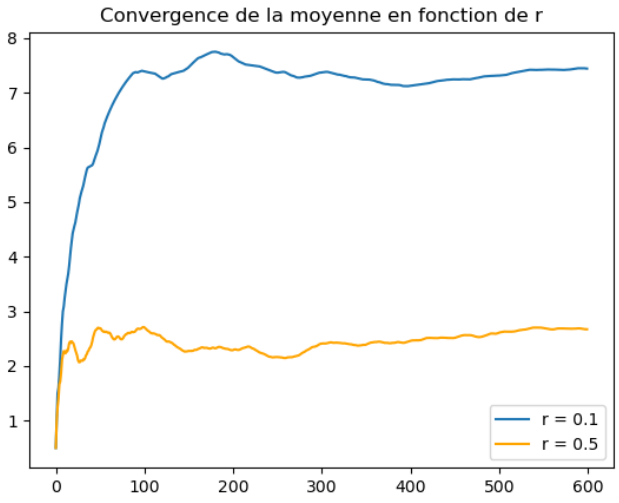

FIGURE 7 – Convergence de la moyenne pour deux valeurs de r

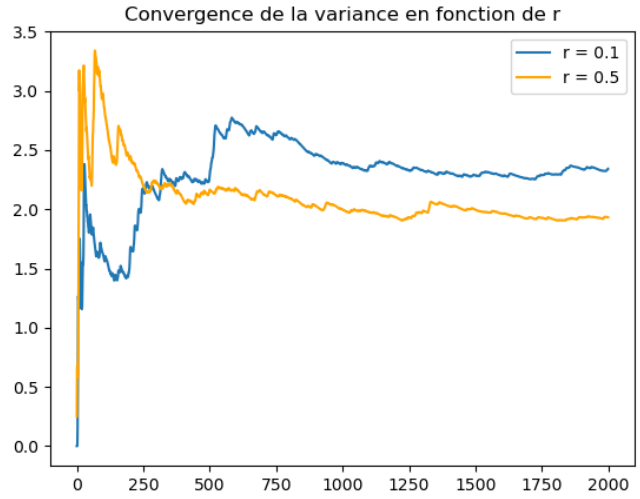

FIGURE 8 – Convergence de la variance pour deux valeurs de r

Au vu des deux figures ci-dessus, on remarque bien que les valeurs convergent et à des taux différents en fonction de r. En effet, plus r est grand, plus on converge rapidement et ce car plus r grandit, plus la proportion du cas $0 \leq x \leq K$ et $y = x - 1$ va prendre de l'importance dans notre distribution, ce qui provoquera une convergence.

1.3.3

On réalise des histogrammes des fréquences d'apparition pour les deux valeurs de r :

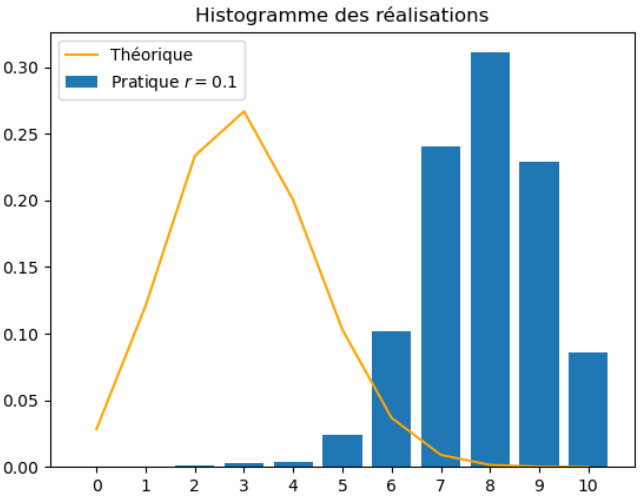

FIGURE 9 – Fréquences d'apparition pour r = 0.1

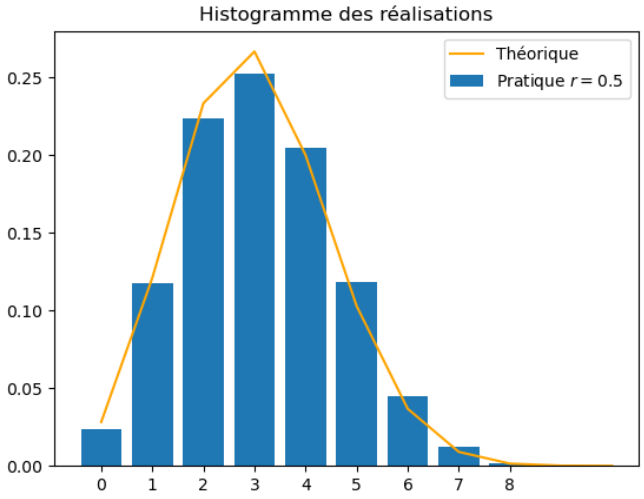

FIGURE 10 – Fréquences d'apparition pour r = 0.5

En comparant les résultats obtenus avec les figures obtenues à la section 1.3.2, on remarque bien qu'on converge vers les mêmes valeurs de moyenne et de variance. Par contre, en comparant ces mêmes résultats avec la distribution théorique, on remarque que un des r est beaucoup plus adapté à nos calculs. En effet, dans le cas où $r = 0.5$, nos fréquences d'apparition convergent bien vers les valeurs attendues par la distribution théorique.

2 Deuxième partie : détection de communautés dans un graphe par algorithmes MCMC

2.1 Etude théorique

2.1.1 Théorème de bayes, cas général

Pour commencer, rappelons nous le théorème de Bayes dans sa forme général :

$$\mathbb{P}(x|y) = \frac{\mathbb{P}(y|x) * \mathbb{P}(x)}{\mathbb{P}(y)}$$

Dans le cadre de ce projet, nous obtenons :

$$\mathbb{P}(x|G) = \frac{\mathbb{P}(G|x) * \mathbb{P}(x)}{\mathbb{P}(G)}$$

Regardons maintenant de plus près les 3 termes de notre équation, commençons par $\mathbb{P}(x)$:

$$\begin{aligned}\mathbb{P}(x) &= \prod_{u=1}^{n} p_{x_u} \\ &= \prod_{i=1}^{n} p_i^{|\Omega_i(x)|}\end{aligned}$$

Ensuite, pour la probabilité conditionnelle $\mathbb{P}(G|x)$:

$$\begin{aligned}\mathbb{P}(G|x) &= \prod_{1\leq u<v\leq n} W_{x_u,x_v}^{G_{u,v}}(1-W_{x_u,x_v})^{1-G_{u,v}} \\ &= \prod_{1\leq i\leq j\leq k} W_{i,j}^{N_{i,j}(x,G)}(1-W_{i,j})^{N_{i,j}^c(x,G)}\end{aligned}$$

Et pour finir, grâce à la lois des probabilités totales, nous pouvons transformer $\mathbb{P}(G)$:

$$\mathbb{P}(G) = \int \mathbb{P}(G|x)\,\mathrm{d}x$$

En recombinant ces résultats, nous pouvons exprimer $\mathbb{P}(x|G)$:

$$\mathbb{P}(x|G) = \frac{\prod_{i=1}^n p_i^{|\Omega_i(x)|}\prod_{1\leq i\leq j\leq k} W_{i,j}^{N_{i,j}(x,G)}(1-W_{i,j})^{N_{i,j}^c(x,G)}}{\int \mathbb{P}(G|x)\,\mathrm{d}x}$$

où :

$$\begin{aligned}N_{i,j}(x,G) &= \sum_{u<v,x_u=i,x_v=j} \mathbb{1}(G_{uv}=1) \\ N_{i,j}^c(x,G) &= \sum_{u<v,x_u=i,x_v=j} \mathbb{1}(G_{uv}=0) \\ &= |\Omega_i(x)| * |\Omega_j(x)| - N_{i,j}(x,G) \text{ si } i\neq j \\ &= \frac{|\Omega_i(x)| * (|\Omega_i(x)|-1)}{2} - N_{i,i}(x,G) \text{ si } i=j\end{aligned}$$

Dans le cas d'une distribution $SBM(N,K,p,A,B)$, la probabilité $\mathbb{P}(x|G)$ sera simplifiée.

$$\begin{aligned}\mathbb{P}(G|x) &= \prod_{1\leq i\leq j\leq k} A^{N_{i,i}(x,G)}(1-A)^{N_{i,i}^c(x,G)} \text{ si } i=j \\ &= \prod_{1\leq i\leq j\leq k} B^{N_{i,j}(x,G)}(1-B)^{N_{i,j}^c(x,G)} \text{ si } i\neq j\end{aligned}$$

2.1.2

Le terme qui va devenir un problème si N augmente, est le dénominateur de la fraction finale du point précédent. En effet, la complexité du calcul de l'intégral est exponentielle par rapport à N. Cependant, il n'est pas nécéssaire de connaitre ce terme car nous souhaitons maximiser $\mathbb{P}(x|G)$ et l'intégrale se simplifiera lors du calcul du taux d'acceptation α car elle est indépendante du vecteur x et donc constante au cours des itérations de l'algorithme de Metropolis-Hastings.

2.1.3

Pour que Metropolis-Hastings fonctionne au mieux, il faut que la chaine de Markov générée par celui-ci soit ergodique, c'est-à-dire que tout état soit atteignable depuis n'importe quel autre état, en un nombre fini de pas. (graphe connexe et irréductible)
Ici, dans notre distribution q_s, on vient modifier au hasard une composante à la fois du graphe ce qui nous assure de couvrir toutes les possibilités/états et qui nous assure donc une chaine de Markov ergodique.

2.1.4

Pour la section qui suit, nous considérons que $\mathbb{P}(x_t|G) < \mathbb{P}(x_{t-1}|G)$, si cette inégalité n'est pas respectée, nous considérerons $\alpha = 1$.
Voici donc l'expression développée de α

$$\begin{aligned}
\alpha &= \frac{\mathbb{P}(x_t|G)}{\mathbb{P}(x_{t-1}|G)} \\
&= \frac{\frac{\prod_{i=1}^n p_i^{|\Omega_i(x_t)|} \prod_{1\le i\le j\le k} W_{i,j}^{N_{ij}(x_t,G)} (1-W_{i,j})^{N^c_{i,j}(x_t,G)}}{\int \mathbb{P}(G|x)\,\mathrm{d}x}}{\frac{\prod_{i=1}^n p_i^{|\Omega_i(x_{t-1})|} \prod_{1\le i\le j\le k} W_{i,j}^{N_{i,j}(x_{t-1},G)} (1-W_{i,j})^{N^c_{i,j}(x_{t-1},G)}}{\int \mathbb{P}(G|x)\,\mathrm{d}x}} \\
&= \frac{\prod_{i=1}^n p_i^{|\Omega_i(x_t)|} \prod_{1\le i\le j\le k} W_{i,j}^{N_{ij}(x_t,G)} (1-W_{i,j})^{N^c_{i,j}(x_t,G)}}{\prod_{i=1}^n p_i^{|\Omega_i(x_{t-1})|} \prod_{1\le i\le j\le k} W_{i,j}^{N_{i,j}(x_{t-1},G)} (1-W_{i,j})^{N^c_{i,j}(x_{t-1},G)}}
\end{aligned}$$

Comme le taux α sera calculé numériquement, il est intéressant de calculer son logarithme. Comme nous obtiendrons des $\mathbb{P}(x|G)$ qui seront des très petits nombres, le logarithme va nous permettre de ne pas aller en dessous de l'Epsilon machine.

$$\log(\mathbb{P}(x|G)) = \sum_{i=1}^k |\Omega_i(x)| \log(p_i) + \sum_{1\le i\le j\le k} N_{i,j}(x,G)\log(W_{i,j}) + N^c_{i,j}(x,G)\log(1-W_{i,j})$$

$$\begin{aligned}
\log\alpha &= \log(\mathbb{P}(x_t|G)) - \log(\mathbb{P}(x_{t-1}|G)) \\
&= \sum_{i=1}^k |\Omega_i(x_t)| \log(p_i) + \sum_{1\le i\le j\le k} N_{i,j}(x_t,G)\log(W_{i,j}) + N^c_{i,j}(x_t,G)\log(1-W_{i,j}) \\
&\quad - \sum_{i=1}^k |\Omega_i(x_{t-1})| \log(p_i) - \sum_{1\le i\le j\le k} N_{i,j}(xt-1,G)\log(W_{i,j}) + N^c_{i,j}(xt-1,G)\log(1-W_{i,j})
\end{aligned}$$

En mettant en évidence tous les termes qui peuvent l'être, nous obtenons :

$$\begin{aligned}
\log\alpha = &\sum_{i=1}^k (|\Omega_i(x_t)| - |\Omega_i(x_{t-1})|)\log(p_i) + \sum_{1\le i\le j\le k} (N_{i,j}(x_t,G) - N_{i,i}(xt-1,G))\log(W_{i,j}) \\
&+ \sum_{1\le i\le j\le k} (N^c_{i,j}(x_t,G) - N^c_{i,j}(x_{t-1},G))\log(1-W_{i,j})
\end{aligned}$$

Ce calcul peut encore être simplifié en mettant à jour les nombres $N_{i,j}(x,G)$, $N^c_{i,j}(x,G)$ et $\Omega_i(x)$ à partir des itérations précédentes.
Nous l'avons implémenté dans notre algorithme pour améliorer grandement les temps d'éxécution.

2.2 Analyse expérimentale

2.2.1

Les graphiques qui vont suivre ont été généré avec une distribution $SBM(500, 2, [0.5, 0.5], a/500, b/500)$

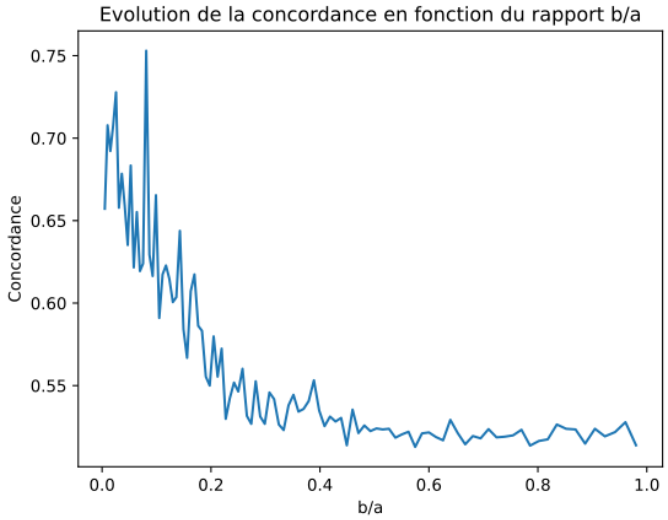

FIGURE 11 – Evolution de la concordance en fonction du rapport b/a

Ce graphique montre bien que dès que le rapport b/a augmente, la concordance diminue rapidement et tend vers 0.5 lorque b/a tend vers 1. C'est un résultat prévisible car plus le rapport est grand, moins il y a de communeautés de distinctes.

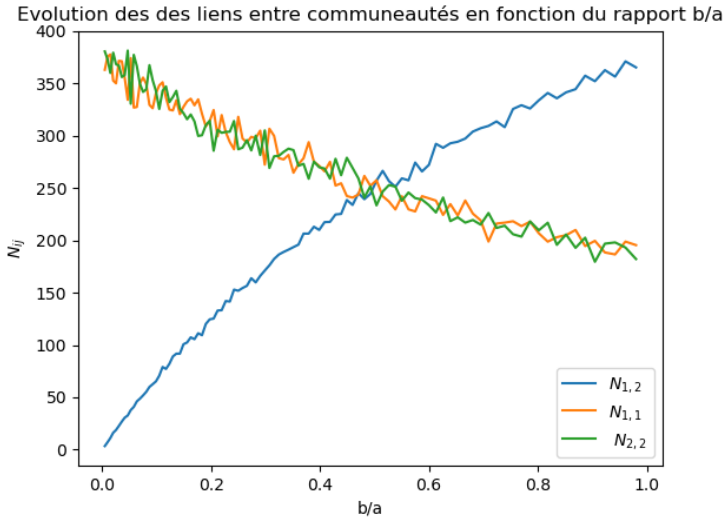

FIGURE 12 – Evolution de la concordance en fonction du rapport b/a

Ce graphique nous aide a comprendre plus visuellement ce que change le rapport b/a. On remarque que plus le rapport est grand, plus il y a de liens entre les communautés. Quand $b/a = 0.5$, il y a autant de liens entre communautés que à l'intérieur des communautés.

En comparant les 2 graphiques, on remarque rapidement que moins les communeautes ont de liens entre elles, plus l'algorithme de Metropolis-Hastings les détecte avec précisions. A partir d'un rapport $b/a = 0.2$ notre algorithme ne donne aucun résultats concluants

2.2.2

En considérant un degré moyen de 3 pour $\frac{a+b}{2}$, nous arrivons à ces conclusions :

$$\begin{aligned}
(a-b)^2 &> 2*(a+b) \\
(a-b)^2 &> 2*12 \\
(a-b) &> \sqrt{24} = 2*\sqrt{6} \\
a &> b + 2*\sqrt{6} \\
a &> 6 - a + 2*\sqrt{6} \\
2*a &> 6 + 2*\sqrt{6} \\
a &> 3 + \sqrt{6} \backsimeq 5,45
\end{aligned}$$

La condition n'est donc plus respectées pour le rapport $b/a = \frac{6-5,45}{5,45} = 0,092$

2.3 Application à un grand graphe

Notre implémentation

Pour implémenter l'algorithme de Metropolis-Hastings, nous avons décider de procéder étape par étape et cela se reflète bien dans nos fonctions. Chaque fonction a été une étape dans notre compréhension du problème.

Nous avons d'abord créé un générateur de distribution SBM pour bien comprendre les rouages de cette distribution. Ensuite, nous avons cherché un moyen de calculer la probabilité $\mathbb{P}(G)$ avant de nous rendre compte que cela nous serait inutile pour notre implémentation. Lorsque nous avons compris ça, nous avons pu avancer.

Nous avons poursuivi avec la fonction **computePX**. Lorsque nous avons simplifié au maximum notre problème pour une distribution SBM symétrique, nous nous sommes encore rendu compte que cette partie était aussi inutile car ce terme reste constant dans notre cas et ne change donc rien a la maximisation de $\mathbb{P}(x|G)$.

Nous avons donc abordé le dernier morceau qui était en fait le seul utile et nous avons rencontrer pas mal de problèmes d'abord du à de simples erreurs de code.

Une fois tous ces soucis réglés, nous avions un algorithme fonctionnel mais très loin d'être efficace. Nous avons donc fait des analyses temporelles pour trouver la partie de notre code qui prenait le plus de temps, c'était la fonction **computeN**. Nous avons compris à ce moment la que recalculer à chaque fois tous les éléments $N_{i,j}(x, G)$ n'était pas la bonne solution.

Nous avons alors implémenté une mise à jour de la variable N à partir de celle du pas précédent. Il nous a ensuite paru logique d'étendre cette méthode à la variable Om. Vous pouvez trouver nos étapes de mise à jour des variables dans la fonction **nextY**. Quand nous avons passé cette étape, le code à tout de suite été beaucoup plus efficace au point que nous avons doublé le nombre d'itérations pour atteindre 1.000.000 et cela nous a pris moins de temps que pour la version précédente.

Par après, nous nous sommes demandé comment éviter de tomber dans un minimum local et nous avons pensé que faire plusieurs tests en prenant différents vecteurs initiaux aléatoires permettrait de prendre en compte la possibilité d'un minimum local. L'arguments nbTests de notre fonction **mhALL** est la concrétisation de cette idée.

Nous avons alors lancé le test sur le graphe proposé pour le challenge et nous avons obtenus un très bon résultat ! ($\log(\mathbb{P}(G|x)\mathbb{P}(x)) = -1295577$)

Amélioration possible

Au fil des séances et des heures passées sur ce projet, nous avons compris que nous ne pourrions jamais atteindre un résultat parfait et cela pour plusieurs raisons.

Premièrement, nous avons des données réelles à analyser. Ces données ne peuvent pas correspondre exactement à une distribution théorique.

Ensuite, nous utilisons une distribution symétrique. En effet, nous avons considéré que la distribution était uniforme ($p_i = p_j, \forall i, j \in [1, k]$) et avons posé $W_{i,i} = A$ et $W_{i,j}, \forall i, j \in [1, k], i \neq j$. Nous savons cependant que $A = \frac{W_{1,1}+W_{2,2}}{2}$ alors que en pratique, les valeurs sur la diagonale de W sont assez différentes. Nous aurions pu chercher grâce à l'algorithme de Metropolis-Hastings la meilleur combinaison de $W_{1,1}$ et $W_{2,2}$ pour avoir une concordance encore plus proche.

Références

[Abb18] Emmanuel Abbe. Community detection and stochastic block models : Recent developments. Journal of Machine Learning Research, 18(177) :1–86, 2018.

[For10] Santo Fortunato. Community detection in graphs. Physics Reports, 486(3-5) :75 – 174, 2010.

[Wik22] Wikipedia. Stochastic block model. `https://en.wikipedia.org/wiki/Stochastic_block_model`

[Geu22] Pierre Geurts. Éléments de processus stochastiques. Méthodes de Monte Carlo par chaînes de Markov pour la détection de communautés dans un graphe.

[Qua22] Dirty Quant. The Metropolis-Hastings Algorithm (MCMC in Python). `https://www.youtube.com/watch?v=MxI78mpq_44&t=861s`