Aspect probabilistes pour l'informatique

Table des matières

Chapitre 1

Rappels de combinatoire

Principe fondamental de dénombrement

On réalise deux expériences : la première peut produire l'un quelconque de m résultats et, pour chacun d'entre eux, il y a n résultats possibles pour la seconde expérience. Alors, il existe $m \times n$ résultats possibles pour les deux expériences prises ensemble.

Si r expériences sont réalisées avec la première pouvant produire n_1 résultats, chacun entrainant n_2 résultats possibles pour la deuxième expérience, et ainsi de suite, alors il existe $n_1 \times n_2 \times \cdots \times n_r$ résultats possibles pour les r expériences prises ensemble.

Permutations

Si E est un ensemble fini à n éléments, on appelle permutation de E une suite ordonnée de n éléments distincts de E. Le nombre de permutations de n objets est $n! = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$.

Si les éléments de E sont partiellement indiscernables, c'est-à-dire qu'il y a n_1 objets indiscernables, qu'on arrive à discerner de n_2 objets indiscernables entre eux, et ainsi de suite jusqu'à n_r objets indiscernables entre eux, le nombre de permutations des $n = n_1 + n_2 + \cdots + n_r$ objets est $\frac{n!}{n_1! n_2! \cdots n_r!}$.

Arrangements et permutations

Si E est un ensemble fini à n éléments et p un entier tel que $1 \leq p \leq n$, on appelle p-arrangement de E une suite de p éléments distincts de E. Le nombre de p-arrangements de E est $n(n-1)(n-2)\cdots(n-p+1) = \frac{n!}{(n-p)!}$. On appelle p-combinaison de E un sous-ensemble de E contenant p éléments (l'ordre des éléments ne compte donc plus, par rapport aux arrangements). Le nombre de p-combinaisons de E est $\binom{n}{p} = \frac{n!}{p!(n-p)!}$, appelé coefficient binomial.

Les coefficients binomiaux sont reliés par de nombreuses identités telles que $\binom{n}{p} = \binom{n}{n-p}$ ou

$$\binom{n}{p} = \binom{n-1}{p-1} + \binom{n-1}{p}$$

qui sert à construire le triangle de Pascal. On a également la formule du binôme de Newton :

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$$

Si, plutôt que d'extraire de E un sous-ensemble, on cherche à diviser l'ensemble en r groupes de tailles respectives $n_1, n_2, \ldots, n_r$, on a $\frac{n!}{n_1! n_2! \cdots n_r!}$: ce sont les coefficients multinomiaux (qui généralisent bien le cas où on divise E en deux sous-ensembles de tailles p et $n-p$).

Cela permet de trouver facilement le nombre de solutions à valeurs entières à une équation de la forme $x_1 + x_2 + \cdots + x_r = n$: si on se restreint aux solutions à valeurs strictement positives, il y a $\binom{n-1}{r-1}$ solutions ; si on compte les solutions à valeurs positives ou nulles, il y en a $\binom{n+r-1}{r-1}$.

Chapitre 2

Probabilités discrètes

2.1 Évènements

La définition mathématique de la notion de probabilité se base sur la construction de mondes différents, chacun donnant un des résultats possibles d'une expérience de pensée probabiliste. On note souvent Ω l'univers des mondes possibles. Si on considère le lancer d'un dé à 6 faces, on peut donc considérer un univers $\Omega = \{1, 2, 3, 4, 5, 6\}$ consistant en l'ensemble des tirages possibles. Souvent, on ne s'intéresse pas tant à un élément ω de l'univers Ω en isolation, mais plutôt à une partie de Ω, qu'on appelle *évènement*. Par exemple, « tirer une face paire » est un évènement de l'expérience précédente. Une fois définie un univers, on cherche à évaluer, quantitativement, les résultats qui sont plus ou moins susceptibles de se produire. On associe donc à chaque ω une grandeur numérique.

Dans le cas discret qui va nous intéresser dans ce chapitre, l'univers Ω est un ensemble fini ou dénombrable (imaginez par exemple le cas où l'univers doit compter le nombre d'opérations élémentaires effectuées par un algorithme ; on a alors $\Omega = \{0, 1, 2, 3, \ldots\} \cup \{+\infty\}$). On associe alors à chaque élément ω sa probabilité $p(\omega) \in [0, 1]$. Pour un évènement A, on lui associe donc une probabilité en sommant les probabilités des évènements élémentaires le composant : $\mathbf{P}(A) = \sum_{\omega \in A} p(\omega)$. Par souci de normalisation, on souhaite avoir toujours $\mathbf{P}(\Omega) = 1$.

Notez que cette définition ensembliste induit de fait que $\mathbf{P}(\emptyset) = 0$, $\mathbf{P}(\overline{A}) = 1 - \mathbf{P}(A)$ (avec $\overline{A} = \Omega \backslash A$ l'évènement complémentaire de A) et $\mathbf{P}(A \uplus B) = \mathbf{P}(A) + \mathbf{P}(B)$ si A et B sont deux parties disjointes de Ω. Lorsque A et B ne sont pas disjointes, on a tout de même $\mathbf{P}(A \cup B) \leq \mathbf{P}(A) + \mathbf{P}(B)$, ce qui se généralise à n évènements en :

$$\mathbf{P}(A_1 \cup \cdots \cup A_n) \leq \mathbf{P}(A_1) + \cdots + \mathbf{P}(A_n)$$

Une formule plus fine, dite d'*inclusion-exclusion*, permet de trouver la probabilité d'une union d'évènements (non nécessairement disjoints) :

$$\mathbf{P}(A \cup B) = \mathbf{P}(A) + \mathbf{P}(B) - \mathbf{P}(A \cap B)$$

qui se généralise pour une union de n évènements en

$$\mathbf{P}(A_1 \cup \cdots \cup A_n) = \sum_{k=1}^{n} (-1)^{k-1} \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq n} \mathbf{P}(A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k})$$

Deux évènements A et B sont dits *indépendants* lorsque la connaissance de la réalisation (ou non) de l'un ne modifie pas notre connaissance de la probabilité de l'autre : cela s'écrit formellement $\mathbf{P}(A \cap B) = \mathbf{P}(A) \times \mathbf{P}(B)$. Une autre façon de le décrire est que la connaissance de l'évènement A n'influe pas sur l'évènement B : la probabilité de A *sachant* B est égale à la probabilité de A. Dans le cas général, on appelle « probabilité de l'évènement A sachant l'évènement B », qu'on

note $\mathbf{P}(A \mid B)$, le quotient $\mathbf{P}(A \cap B)/\mathbf{P}(B)$, défini uniquement si $\mathbf{P}(B) \neq 0$. Les probabilités conditionnelles permettent par exemple de calculer la probabilités d'une intersection de n évènements non nécessairement indépendants, à l'aide de la formule des *probabilités composées* :

$$\mathbf{P}(A_1 \cap \cdots \cap A_n) = \mathbf{P}(A_1) \cdot \mathbf{P}(A_2|A_1) \cdot \mathbf{P}(A_3|A_1 \cap A_2) \cdots \mathbf{P}(A_n|A_1 \cap A_2 \cap \cdots \cap A_{n-1})$$

Les probabilités conditionnelles permettent également de calculer la probabilité d'un évènement en le décomposant sur une partition de Ω, c'est-à-dire un ensemble $\{B_i \mid 1 \leq i \leq n\}$ d'évènements disjoints tels que $\biguplus_{i=1}^{n} B_i = \Omega$: on a alors pour tout évènement A

$$\mathbf{P}(A) = \sum_{i=1}^{n} \mathbf{P}(A \mid B_i)\mathbf{P}(B_i)$$

C'est la formule des *probabilités totales*.

Finalement, la formule de Bayes (au cœur de l'*inférence bayésienne* très utile en intelligence artificielle) permet de déterminer la probabilité de A sachant B, en fonction des probabilité de A, de B et de B sachant A :

$$\mathbf{P}(A \mid B) = \frac{\mathbf{P}(B \mid A)\mathbf{P}(A)}{\mathbf{P}(B)}$$

Combinée avec la formule des probabilités totales, on obtient pour une partition $\{A_i \mid 1 \leq i \leq n\}$ de Ω :

$$\mathbf{P}(A_i \mid B) = \frac{\mathbf{P}(B \mid A_i)\mathbf{P}(A_i)}{\sum_j \mathbf{P}(B \mid A_j)\mathbf{P}(A_j)}$$

Application : accès simultanés à une base de données

Considérons la situation où $n \geq 2$ processus $P_1, P_2, \ldots, P_n$ accèdent de façon compétitive à une base de données. La base peut être utilisée par un seul processus à la fois. On discrétise le temps qui est donc divisé en tours $\{1, 2, 3, 4, \ldots\}$. Pendant un tour, si au moins deux processus essaient d'accéder à la base, alors elle devient verrouillée pendant ce tour. On se demande comment proposer une méthode équitable d'accès pour chaque processus.

Étudions la méthode probabiliste suivante : pour une valeur $0 < p \leq 1$ à déterminer, à chaque tour, chaque processus P_i accède à la base aléatoirement avec probabilité p, de manière indépendante du choix des autres processus.

Considérons les événements suivants :
$A[i,t]$: « P_i essaie d'accéder à la base au tour t »
$S[i,t]$: « P_i réussit à accéder à la base au tour t »
$F[i,t]$: « P_i ne réussit à accéder à la base dans aucun des tours $1, 2, \ldots, t$ »
Sans même avoir à décrire formellement l'univers et les probabilités, on sait que (on note $\overline{A}$ l'évènement complémentaire $\Omega \setminus A$) :

$$\mathbf{P}(A[i,t]) = p, \qquad \mathbf{P}(\overline{A[i,t]}) = 1 - p, \qquad \text{et} \qquad S[i,t] = A[i,t] \cap \left(\bigcap_{j \neq i} \overline{A[j,t]} \right) .$$

Les évènements $A[i,t]$, $\overline{A[j,t]}$ et $\overline{A[k,t]}$ sont indépendants pour i, j et k trois processus distincts (puisque les tirages aléatoires se font de manière indépendante), donc

$$\mathbf{P}(S[i,t]) = \mathbf{P}(A[i,t]) \times \prod_{j \neq i} \mathbf{P}(\overline{A[j,t]}) = p(1-p)^{n-1} .$$

Notons $f(p) = p(1-p)^{n-1}$. Intuitivement, on cherche à maximiser cette probabilité, il est donc naturel d'essayer de choisir p maximisant cette fonction. La dérivée de f vaut $f'(p) = (1-p)^{n-1} - (n-1)p(1-p)^{n-2}$. Elle est donc nulle si et seulement si $p = \frac{1}{n}$, et on peut vérifier

qu'il s'agit d'un maximum de la fonction f. Par la suite, posons donc $p = \frac{1}{n}$. Par conséquent, $\mathbf{P}(S[i,t]) = \frac{1}{n}(1-\frac{1}{n})^{n-1}$.

Puisque $n \geq 2$, on peut voir que la fonction $n \mapsto (1-\frac{1}{n})^n$ croit de $\frac{1}{4}$ à $\frac{1}{e}$, alors que la fonction $n \mapsto (1-\frac{1}{n})^{n-1}$ décroit de $\frac{1}{2}$ à $\frac{1}{e}$. Ainsi, $\frac{1}{en} \leq \mathbf{P}(S[i,t]) \leq \frac{1}{2n}$, c'est-à-dire $\mathbf{P}(S[i,t]) = \Theta(\frac{1}{n})$.

On cherche à étudier l'évolution de $\mathbf{P}(F[i,t])$ lorsque t grandit. Or, tous les tours étant indépendants,

$$\mathbf{P}(F[i,t]) = \mathbf{P}\left(\bigcap_{r=1}^{t} \overline{S[i,r]}\right) = \prod_{r=1}^{t} \mathbf{P}(\overline{S[i,r]})\,.$$

Puisque $\mathbf{P}(\overline{S[i,r]}) \leq 1 - \frac{1}{en}$, on a

$$\mathbf{P}(F[i,t]) \leq \left(1-\frac{1}{en}\right)^t\,.$$

Au tour $t = en$ en particulier,

$$\mathbf{P}(F[i,en]) \leq \left(1-\frac{1}{en}\right)^{en} \leq \frac{1}{e}\,.$$

Cela veut dire qu'après $O(n)$ tours, la probabilité d'échec est bornée par une constante $\frac{1}{e}$.

Maintenant, si $t = (en)\cdot(c\log n)$, alors

$$\mathbf{P}(F[i,t]) \leq \left(\left(1-\frac{1}{en}\right)^{en}\right)^{c\log n} \leq e^{-c\log n} = \frac{1}{n^c}\,.$$

Ainsi, après $O(n\log n)$ tours, la probabilité d'échec est beaucoup plus petite (qu'après $O(n)$ tours) et bornée par $\frac{1}{n^c}$.

Considérons finalement l'évènement F qu'un des processus n'a pas réussi à accéder la base dans aucune des $t = (en)\cdot(c\log n)$ premiers tours. Si $c = 2$ (c'est-à-dire, $t = 2en\log n$), alors

$$\mathbf{P}(F) = \mathbf{P}\Big(\bigcup_{i=1}^{n} F[i,t]\Big) \leq \sum_{i=1}^{n} \mathbf{P}(F[i,t]) \leq n\cdot\frac{1}{n^2} = \frac{1}{n}\,.$$

En conclusion, avec une probabilité $\geq 1-\frac{1}{n}$, les n processus réussissent à accéder à la base au moins une fois après $2en\log n$ tours.

2.2 Variables aléatoires discrètes

Soit $(\Omega, \mathbf{P})$ un espace probabiliste, c'est-à-dire un univers et une fonction de probabilité. Une *variable aléatoire* X est une application $X\colon \Omega \to \mathbb{R}$ associant à chaque élément de l'univers une valeur. Pour tout ensemble $A \subseteq \mathbb{R}$ (suffisamment régulier, mais nous ne rentrerons pas dans ces technicités, dans ce cours), l'ensemble des expériences $\omega \in \Omega$ telles que $X(\omega) \in A$ est un évènement qu'on note $[X \in A]$: on note $\mathbf{P}[X \in A]$ sa probabilité. On appelle *loi* de la variable X la fonction décrivant la probabilité des évènements liés à X.

En fait, il suffit de connaître la probabilité des évènements $[X \leq x]$ pour tout $x \in \mathbf{R}$ pour connaître toute la loi de X : on appelle *fonction de répartition* la fonction $F\colon x \mapsto \mathbf{P}[X \leq x]$. La fonction de répartition caractérise la loi. Par exemple, la probabilité $\mathbf{P}[a < X \leq b]$ est obtenu en remarquant que $[a < X \leq b] = [X \leq b] \setminus [X \leq a]$ de sorte que $\mathbf{P}[a < X \leq b] = F(b) - F(a)$.

Notons $X(\Omega) = \{x_k \mid k \in \mathbf{N}\}$ l'image de la variable aléatoire X (on rappelle que Ω est ici un ensemble au plus dénombrable, la variable X est en particulier dite discrète). L'*espérance* $\mathbf{E}(X)$ d'une telle variable aléatoire est définie par

$$\mathbf{E}(X) = \sum_{k\in\mathbf{N}} x_k \times \mathbf{P}[X = x_k]\,.$$

Elle n'est définie que si la somme infinie converge (si $X(\Omega)$ est finie, l'espérance existe donc nécessairement). L'espérance permet de donner une estimation de la probabilité que la variable X soit « grande » :

Théorème 2.2.1 (Inégalité de Markov)**.** *Soit X une variable aléatoire discrète, positive, admettant une espérance. Pour tout $t > 0$,*

$$\mathbf{P}[X \geq t] \leq \frac{\mathbf{E}(X)}{t}$$

Une propriété cruciale est la linéarité de l'espérance : si X et Y sont deux variables aléatoires sur Ω, on note $X + Y$ la variable aléatoire égale à $X(\omega) + Y(\omega)$ pour tout $\omega \in \Omega$, et alors, on a $\mathbf{E}(X + Y) = \mathbf{E}(X) + \mathbf{E}(Y)$.

Une autre statistique intéressante d'une variable aléatoire est sa variance, définie par $\mathbf{V}(X) = \mathbf{E}\big((X - \mathbf{E}(X))^2\big)$ (à nouveau uniquement si la somme potentiellement infinie converge). Elle permet de quantifier la dispersion des valeurs possibles de X par rapport à sa moyenne (son espérance), au sens précisé par le théorème suivant :

Théorème 2.2.2 (Inégalité de Bienaymé-Tchebychev)**.** *Soit X une variable aléatoire discrète admettant une variance. Notons $m = \mathbf{E}(X)$ l'espérance de X et σ_X^2 sa variance. Alors, pour tout $\varepsilon > 0$, on a*

$$\mathbf{P}[|X - m| > \varepsilon] \leq \frac{\sigma_X^2}{\varepsilon^2}$$

ou encore

$$\mathbf{P}[|X - m| > \varepsilon\sigma_X] \leq \frac{1}{\varepsilon^2}$$

La linéarité de l'espérance implique que la variance peut aussi s'écrire

$$\mathbf{V}(X) = \mathbf{E}(X^2) - \mathbf{E}(X)^2$$

où on note X^2 la variable aléatoire associant à toute expérience ω, $X(\omega)^2$. La variance, étant une quantité d'ordre 2, n'est pas linéaire. Elle le devient dès lors que les variables aléatoires sont *indépendantes*. Une famille de variables aléatoires $(X_1, X_2, \ldots, X_n)$ est dite indépendante dès lors que pour toute valeurs possibles $x_1, x_2, \ldots, x_n$ des n variables, on a

$$\mathbf{P}[X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n] = \prod_{i=1}^{n} \mathbf{P}[X_i = x_i]$$

Ainsi, si deux variables X et Y sont indépendantes, alors on a bien $\mathbf{V}(X + Y) = \mathbf{V}(X) + \mathbf{V}(Y)$. Cela provient du fait que $\mathbf{E}(XY) = \mathbf{E}(X)\mathbf{E}(Y)$ dans ce cas.

Pour des raisons d'unité (la variance est une quantité d'ordre 2), on utilise parfois plutôt l'écart-type $\sigma_X = \sqrt{\mathbf{V}(X)}$ d'une variable X.

2.2.1 Loi uniforme

La loi la plus simple est la loi uniforme décrivant des chances équiprobables sur un ensemble fini. Par exemple, considérons un lancer de dé à 6 faces. On peut considérer l'ensemble $\{1, 2, \ldots, 6\}$ avec $\mathbf{P}[X = j] = \frac{1}{6}$ pour tout $j = 1, \ldots, 6$, alors

$$\mathbf{E}(X) = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + \cdots + 6 \cdot \frac{1}{6} = \frac{6 \times 7}{2 \times 6} = \frac{7}{2}\,.$$

Cela s'interprète comme le fait que la valeur moyenne d'une face de dé est $3,5$.

Plus généralement, pour n éléments, on a $\mathbf{P}[X = j] = \frac{1}{n}$ pour tout $j \in \{1, 2, \ldots, n\}$. L'espérance vaut $\mathbf{E}(X) = \frac{n+1}{2}$ et la variance $\mathbf{V}(X) = \frac{n^2-1}{12}$.

2.2.2 Loi de Bernoulli

Considérons l'expérience consistant au lancer d'une pièce ayant probabilité p de tomber sur *pile* et $1-p$ sur *face* : la variable aléatoire X prenant la valeur 1 lorsque la pièce tombe sur *pile* et 0 lorsqu'elle tombe sur *face* suit une loi de Bernoulli. Son espérance est $\mathbf{E}(X) = p$ et sa variance $\mathbf{V}(X) = p(1-p)$.

2.2.3 Loi binomiale

La loi binomiale modélise une expérience consistant en la réalisation de n épreuves de Bernoulli indépendantes de même loi de paramètre p : on note X le nombre total de succès. Par exemple, on peut lancer n fois de suite une pièce déséquilibrée, donnant *pile* avec probabilité p, et compter le nombre de *pile* obtenus au cours des n lancers. La variable X prend donc les valeurs $\{0, 1, 2, \ldots, n\}$ et on a, pour $0 \leq k \leq n$,

$$\mathbf{P}[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$$

puisqu'il faut choisir k lancers réussis sur les n.

L'espérance de la loi binomiale de paramètres n et p est $\mathbf{E}(X) = np$ puisque c'est la somme de n variables de Bernoulli. Sa variance vaut $\mathbf{V}(X) = np(1-p)$.

Un résultat intéressant venant de la répétition d'expériences de Bernoulli indépendantes est la loi des grands nombres, qu'on ne citera que dans sa version faible dans ce cours :

Théorème 2.2.3. *Soit $(X_n)_{n\in\mathbf{N}}$ une suite de variables indépendantes suivantes toutes la même loi de Bernoulli de paramètre p. Pour tout n, notons $M_n = (X_1 + \cdots + X_n)/n$ la moyenne des n premières variables. Alors la suite $(M_n)_{n\geq 1}$ des moyennes converge vers p, au sens où, pour tout $\varepsilon > 0$,*

$$\mathbf{P}[|M_n - p| > \varepsilon] \underset{n\to\infty}{\longrightarrow} 0$$

2.2.4 Loi géométrique

Avec une variable de Bernoulli, on peut vouloir calculer le nombre moyen (« l'espérance du nombre ») de lancers jusqu'au premier lancer tombant sur *pile*. Notons X la variable aléatoire égale au nombre de lancers : elle suit une loi géométrique. L'univers correspond donc à une séquence infinie de lancers de face, c'est-à-dire $\Omega = \{pile, face\}^{\mathbf{N}}$. Notons que $\mathbf{P}[X = j] = (1-p)^{j-1} \cdot p$: pour qu'on ait besoin d'exactement j lancers, il faut obtenir *face* aux $j-1$ premiers lancers et *pile* au j-ième lancer, tous les lancers étant indépendants. Notez qu'il est possible qu'on ne rencontre jamais *pile*, on a alors $X = +\infty$: cependant,

$$\mathbf{P}[X < +\infty] = \sum_{j=1}^{+\infty} \mathbf{P}[X = j] = \sum_{j=1}^{+\infty} p(1-p)^{j-1} = \frac{p}{1-(1-p)} = 1$$

L'évènement $[X = +\infty]$ est donc négligeable.

L'espérance de X vaut $\mathbf{E}(X) = \frac{1}{p}$. Cela s'interprète de la façon suivante : si on a une expérience de Bernoulli avec probabilité de succès $p > 0$, alors l'espérance du nombre de répétitions jusqu'au premier succès est $\frac{1}{p}$. La variance de la variable X suivant une loi géométrique est $\mathbf{V}(X) = \frac{1-p}{p^2}$.

Les lois géométriques vérifient les propriétés de non vieillissement : on dit qu'elles sont sans mémoire. Dans le cas d'attente d'un tirage gagnant au pile ou face, la probabilité de devoir attendre au moins 5 lancers avec le premier succès est la même que la probabilité de devoir attendre au moins 15 lancers lorsqu'on n'a eu aucun succès pendant les 10 premiers. Formellement,

$$\mathbf{P}[X > k] = \sum_{n=k+1}^{\infty} p(1-p)^{n-1} = (1-p)^k$$

d'où

$$\mathbf{P}[X > k+\ell \mid X > \ell] = \frac{\mathbf{P}[X > k+\ell, X > \ell]}{\mathbf{P}[X > \ell]} = \frac{\mathbf{P}[X > k+\ell]}{\mathbf{P}[X > \ell]} = \frac{(1-p)^{k+\ell}}{(1-p)^{\ell}} = (1-p)^k = \mathbf{P}[X > k]$$

Chapitre 3

Algorithmes randomisés

Il existe plusieurs manières d'utiliser l'aléa lors de la conception d'algorithmes. La première consiste à effectuer des actions aléatoirement dans l'objectif de calculer le résultat dans un temps que l'on espère (au sens mathématique) faible. La seconde consiste à calculer avec une complexité garantie un résultat qui est très probablement correct. Ces deux types d'algorithmes portent des noms :

— les algorithmes de Las Vegas : solution garantie, complexité espérée ;
— les algorithmes de Monte Carlo : solution probablement correcte, complexité garantie.

Se contenter d'un algorithme probabiliste permet souvent d'obtenir des algorithmes plus performants. Nous allons en étudier quelques exemples.

3.1 Tri rapide et selection

Commençons par un exemple d'algorithme de Las Vegas, produisant une permutation triée d'un tableau, dont la complexité n'est bonne qu'en moyenne : il s'agit du tri rapide.

Dans l'algorithme de tri rapide d'un tableau contenant n éléments distincts $S = \{a_1, \dots, a_n\}$, on choisit un élément a de S comme pivot et on définit les ensembles $S_< = \{a_i \mid a_i < a\}$, $S_= = \{a_i \mid a_i = a\}$, et $S_> = \{a_i \mid a_i > a\}$. Récursivement, on effectue un appel de l'algorithme sur $S_<$ et/ou $S_>$. On dit que a est un bon pivot si les ensembles $S_<$ et $S_>$ sont relativement petits, c'est-à-dire contiennent au plus une fraction d'éléments par rapport à S (disons au plus $\frac{3}{4}n$ éléments par exemple). Ce pivot peut être choisi de manière aléatoire, auquel cas les algorithmes deviennent eux-même aléatoires : il s'agit alors d'algorithmes de Las Vegas, puisque l'algorithme reste correct quel que soit le choix du pivot. Cependant la complexité de l'algorithme dépend du choix du pivot. Avant toute chose, essayons de comprendre quelle est la probabilité de choisir un *bon pivot*.

Pour répondre à cette question, considérons le tableau trié $[b_1, \dots, b_n]$ contenant les éléments de $S = \{a_1, \dots, a_n\}$. Notons que les bons pivots sont exactement les éléments b_i avec $\frac{1}{4}n \leq i \leq \frac{3}{4}n$. Donc parmi les n éléments de S, $\frac{n}{2}$ sont des bons pivots. Autrement dit, pour tout élément $a \in S$, $\mathbf{P}[a \text{ est un bon pivot}] = \frac{1}{2}$.

À la lumière de ce calcul, on peut modifier l'algorithme de tri rapide de façon à vérifier à chaque fois si le pivot courant est un bon pivot et le tirer de nouveau si ce n'est pas le cas. Le tirage aléatoire d'un tel pivot suit donc une loi géométrique de paramètre $1/2$, dont on sait que l'espérance vaut 2 : ainsi, en moyenne, cet algorithme tirera le bon pivot deux fois. Puisque l'obtention d'un bon pivot à chaque étape induit une complexité en $O(n \log n)$ (essayez de vous en convaincre !), on en déduit que cet algorithme modifié a alors une complexité moyenne de l'ordre de $O(n \log n)$.

En fait, même sans modification, l'algorithme de tri rapide a une complexité en moyenne (c'est-à-dire si l'on considère une permutation aléatoire des éléments en entrée) de l'ordre de $O(n \log n)$, comme vous le verrez en TD.

Une modification très simple de l'algorithme de tri rapide permet aussi de trouver le k-ième plus petit élément d'un tableau quelconque, avec un algorithme randomisé de complexité en moyenne inférieure à $4n$. Asymptotiquement, il existe aussi des algorithmes déterministes atteignant une complexité linéaire (optimale asymptotiquement).

Étudions désormais un autre algorithme randomisé strictement meilleur (en espérance) que tous les algorithmes déterministes connus : c'est à nouveau un algorithme de type Las Vegas en ce sens qu'il fournit assurément la bonne réponse, mais avec une complexité moyenne qu'on va contrôler. On suit une approche radicalement différente, non récursive. L'idée est de trouver rapidement deux éléments du tableau a et b tels que, *avec très forte probabilité*, les deux propriétés suivantes sont satisfaites :

— le k-ième élément, qu'on note $S_{(k)}$ dans la suite, se trouve dans l'ensemble P des éléments qui sont entre a et b (dans l'ordre croissant) ;
— l'ensemble P est *petit*.

Si ces deux conditions sont remplis, on peut alors, en comparant a et b à chaque élément de S, calculer à la fois P, les rangs $r_S(a)$ et $r_S(b)$ de a et b respectivement, puis trier complètement P pour pouvoir trouver $S_{(k)}$, le $k - r_S(a) + 1$-ième élément de P. La définition de *petit* doit donc permettre de trier complètement P sans affecter sensiblement la complexité de l'algorithme.

Pour trouver les deux éléments a et b, on tire au hasard un nombre grand (mais petit vis-à-vis de n) d'éléments de S : les rangs de ces éléments sont alors répartis uniformément entre 1 et n. On va en tirer une proportion $1/n^{1/4}$, c'est-à-dire $n^{3/4}$ éléments de S (avec remise pour simplifier l'étude qui suit). Voici donc l'algorithme proposé :

1 $x := kn^{-1/4}$; $\ell := \max(\lfloor x - \sqrt{n} \rfloor, 1)$; $h := \max(\lfloor x + \sqrt{n} \rfloor, n)$;
2 Tirer indépendamment $m = \lceil n^{3/4} \rceil$ éléments aléatoires de S (avec remise) pour former un multi-ensemble T;
3 Trier T (en temps $O(n^{3/4} \log n)$);
4 $a := T[\ell]$; $b := T[h]$;
5 En comparant a et b à chaque élément de S, déterminer $r_S(a)$, $r_S(b)$, $\{s \in S \mid s \leq a\}$, $\{s \in S \mid a \leq s \leq b\}$, $\{s \in S \mid b \leq s\}$;
6 Si $k < n^{1/4}$ alors $P := \{s \in S \mid s \leq b\}$
7 sinon si $k > n - n^{1/4}$ alors $P := \{s \in S \mid s \geq a\}$ sinon $P := \{s \in S \mid a \leq s \leq b\}$;
8 Si $|P| > 4n^{3/4} + 2$ ou $S_{(k)} \notin P$, retourner en 2;
9 Trier P (en temps $O(n^{3/4} \log n)$);
10 Retourner $P[k - r_S(a) + 1]$

Si les étapes 1 à 8 sont exécutées une seule fois, alors la complexité de l'algorithme est $2n + O(n^{3/4} \log n)$. Montrons que l'algorithme a une complexité moyenne en $2n + o(n)$. On va même montrer mieux : avec probabilité $1 - o(1)$, l'algorithme a une complexité $2n + o(n)$. Pour cela, il suffit de majorer la probabilité des deux évènements $A = [S_{(k)} \notin P]$ et $B = [|P| > 4n^{3/4} + 2]$: nous allons essentiellement montrer que ces deux probabilités tendent vers 0. Le principe est que, puisque l'ensemble T comprend une proportion $1/n^{1/4}$ d'éléments de S, le rand de a dans S devrait être de l'ordre de $k - n^{3/4}$ et celui de b de l'ordre de $k + n^{3/4}$. Ainsi, $S_{(k)}$ devrait, avec bonne probabilité, se trouver entre a et b et l'ensemble P devrait contenir de l'ordre de $2n^{3/4}$ éléments. Par conséquent, il devrait être possible de prouver que les deux évènements considérés sont peu probables, pour que peu que les variables aléatoires $r_S(a)$ et $r_S(b)$ s'écartent peu de leur espérance (on utilisera l'inégalité de Bienaymé-Tchebychev pour démontrer cela).

Probabilité de $A = [S_{(k)} \notin P]$ On se place dans le cas $n^{3/4} \leq k \leq n - n^{3/4}$ (faites les autres cas en exercice !), de sorte que $\ell = \lfloor x - \sqrt{n} \rfloor$ et $h = \lfloor x + \sqrt{n} \rfloor$. Soit X le nombre d'éléments de T qui sont inférieurs ou égaux à $S_{(k)}$: X suit une loi binomiale de paramètres m (taille de T) et k/n (probabilité pour un élément aléatoire d'être inférieure ou égal à $S_{(k)}$). On a donc $\mathbf{E}(X) = km/n = x$ et $\mathbf{V}(X) = km/n(1 - k/n) \leq km/n$.

L'évènement A correspond au cas où $S_{(k)}$ est soit plus petit que $T_{(\ell)}$ (ce qui signifie que X est inférieure à $\mathbf{E}(X) - \sqrt{n}$), soit plus grand que $T_{(h)}$ (auquel cas X est supérieure à $\mathbf{E}(X) + \sqrt{n}$). Par l'inégalité de Bienaymé-Tchebychev (avec $\varepsilon = \sqrt{n}$),

$$\mathbf{P}[S_{(k)} \notin P] = \mathbf{P}[|X - \mathbf{E}(X)| > \sqrt{n}] \leq \frac{\mathbf{V}(X)}{n} \leq \frac{km}{n^2} \leq n^{-1/4}$$

Notons que cette probabilité tend donc vers 0 lorsque n tend vers l'infini.

Probabilité de $B = [|P| > 4n^{3/4} + 2]$ On se place dans le cas $n^{3/4} \leq k \leq n - n^{3/4}$ (faites les autres cas en exercice !), soit $\sqrt{n} \leq x \leq n^{3/4} - \sqrt{n}$. Le nombre d'éléments de P est $|P| = r_S(b) - r_S(a) + 1$: par conséquent B se produit si et seulement si $r_S(b) - r_S(a) \geq 4n^{3/4} + 2$. Cela nécessite que $r_S(b) \geq k + 2n^{3/4} + 1$ ou que $r_S(a) \leq k - 2n^{3/4} - 1$ (sinon, on a $r_S(b) - r_S(a) < k + 2n^{3/4} + 1 - (k - 2n^{3/4} - 1) = 4n^{3/4} + 2$). Majorons donc la probabilité des évènements $C = \ll r_S(b) \geq k + 2n^{3/4} + 1 \gg$ et $D = \ll r_S(a) \leq k - 2n^{3/4} - 1 \gg$.

L'évènement C se produit si, parmi les m choix indépendants d'éléments de S qui sont choisis pour T, le h-ième plus petit est plus grand que $k + 2n^{3/4} + 1$ éléments de S, c'est-à-dire si moins de h des éléments choisis pour T sont parmi les $k + 2n^{3/4} + 1$ plus petits de S. Notons Y la variable aléatoire égale au nombre des éléments de T qui sont choisis parmi les $k + 2n^{3/4} + 1$ plus petits de S. Chacun des éléments de T étant choisi uniformément, indépendamment des autres, Y suit une loi binomiale de paramètres m et $(k + 2n^{3/4} + 1)/n$: Y est d'espérance $\mathbf{E}(Y) = x + 2\sqrt{n} + O(n^{-1/4}) = h + \sqrt{n} + O(n^{-1/4})$ et de variance $\mathbf{V}(Y) \leq \mathbf{E}(Y) \leq n^{3/4} + \sqrt{n} + O(n^{-1/4})$. On a donc, par l'inégalité de Bienaymé-Tchebychev,

$$\begin{aligned}
\mathbf{P}(C) &= \mathbf{P}(B \leq h) \\
&= \mathbf{P}(B - h - \sqrt{n} \leq -\sqrt{n}) \\
&\leq \mathbf{P}(B - \mathbf{E}(Y) < -\sqrt{n}) \\
&\leq \mathbf{P}(|B - \mathbf{E}(Y)| > \sqrt{n}) \\
&\leq \frac{n^{3/4} + O(\sqrt{n})}{n} = O(n^{-1/4})
\end{aligned}$$

De même, on trouve $\mathbf{P}(D) = O(n^{-1/4})$. Ces deux probabilités tendent aussi vers 0 lorsque n tend vers l'infini.

Conclusion Ainsi, avec probabilité $1 - O(n^{-1/4})$, l'algorithme se termine après un tour, avec une complexité en $2n + O(n^{3/4} \log n) = 2n + o(n)$. De plus, le nombre moyen de tentatives nécessaires pour obtenir un ensemble T convenable est une variable géométrique de paramètre $1 - O(n^{-1/4})$ et donc d'espérance $1 + O(n^{-1/4})$ (on utilise ici le fait que $\frac{1}{1-x} \sim 1 + x$) : par conséquence le nombre moyen de comparaisons de l'algorithme est également en $2n + o(n)$.

Cet algorithme est donc essentiellement un algorithme de type Monte Carlo (avec la particularité de signaler les cas d'échec) qu'on répète jusqu'à obtenir un succès. Il se trouve simplement que sa probabilité de succès dès la première tentative tend vers 1 lorsque la taille du problème tend vers l'infini.

3.2 Algorithme de Karger pour la coupe minimum

Soit $G = (V, E)$ un graphe (non-orienté) et $c \colon E \to \mathbf{N}$ une fonction de capacité. On cherche comment calculer efficacement une coupe minimum, c'est-à-dire un ensemble de sommets $U \subsetneq V$, $U \neq \emptyset$ et $U \neq V$, qui minimise $\sum_{e \in \delta(U)} c(e)$, avec $\delta(U) = \{e = (u, v) \in E \mid u \in U, v \notin U\}$.

En passant via une alternative orientée (il suffit de remplacer chaque arête non orientée e par deux arcs entre la même paire de sommets, dans chaque sens, ayant la même capacité $c(e)$), un calcul de flot maximum de la source s à la cible t nous assure de pouvoir trouver une (s, t)-coupe minimale, c'est-à-dire une coupe U telle que $s \in U$ et $t \notin U$, minimale parmi toutes ces coupes.

Ensuite, il faut tester toutes les paires (s,t) possibles et garder celles ayant la plus petite coupe. Cet algorithme, en utilisant l'algorithme d'Edmonds-Karp pour le calcul des flots maximums, a une complexité asymptotique $O(n^3m^2)$.

Utiliser un algorithme randomisé pour calculer plus rapidement une coupe qui est minimum avec forte probabilité.

Nous commençons par montrer que les arêtes d'une coupe minimum sont rares dans le graphe.

Lemme 3.2.1. *Notons $C = \sum_{e\in E} c(e)$. Soit k la capacité minimum d'une coupe de G. Alors $C \geq nk/2$ (c'est-à-dire $k \leq 2C/n$).*

Démonstration. Pour tout sommet u, la coupe $\delta(u)$ (ou $\delta(\{u\})$) a une capacité au moins égale à la coupe minimum. En sommant sur tous les sommets, on obtient :

$$\sum_{u\in V}\sum_{e\in\delta(u)} c(e) \geq nk$$

Par ailleurs, dans cette sommation, chaque arête est comptée deux fois, une fois à chacune de ses extrémités. Ce qui donne :

$$\sum_{u\in V}\sum_{e\in\delta(u)} c(e) = 2\sum_{e\in E} c_e = 2C$$

Donc $2C \geq nk$, c'est-à-dire $C \geq nk/2$. □

Ainsi :

Lemme 3.2.2. *Soit U une coupe minimum de G. Soit e une arête choisie aléatoirement avec une distribution proportionnelle aux capacités des arêtes (c'est-à-dire $\mathbf{P}(e \text{ est choisie}) = c_e/C$). Alors la probabilité que e soit dans la coupe minimum considérée est*

$$\mathbf{P}(e \in \delta(U)) \leq \frac{2}{n}\,.$$

Démonstration. En utilisant le lemme précédent, il vient

$$\mathbf{P}(e \in \delta(u)) = \sum_{e\in\delta(U)} \frac{c_e}{C} = \frac{k}{C} \leq \frac{2}{n}\,.$$ □

Nous pouvons donc choisir une arête aléatoirement et avoir une bonne probabilité qu'elle ne fasse pas partie de la coupe minimum U. Notez qu'il peut y avoir plusieurs coupes minimums, et que possiblement toutes les arêtes sont dans au moins une coupe minimum. Il est donc important ici que U soit une coupe minimum fixée, et ce que nous allons trouver, c'est un algorithme qui trouve cette coupe U avec une probabilité $\Omega(1/n^2)$.

Une fois choisie une arête qui ne sera pas dans la coupe, nous pouvons identifier ses deux extrémités en un seul sommet.

Définition 3.2.1. *Soit $G = (V,E)$ un multi-graphe (c'est-à-dire un graphe orienté dans lequel on a le droit d'avoir plusieurs arcs e ayant la même source $src(e)$ et la même destination $dst(e)$) et $e = (u,v) \in E$. Le graphe G/e,* contraction *du graphe G par l'arête e, est le multi-graphe (V',E') défini par :*

- *$V' = V \setminus \{v\}$,*
- *$E' = E \setminus \{e\}$,*
- *$dst_{G'}(e') = dst_G(e')$ si $dst_G(e') \neq v$, $dst_{G'}(e') = u$ sinon,*
- *$src_{G'}(e') = src_G(e')$ si $src_G(e') \neq v$, $src_{G'}(e') = u$ sinon.*

Ainsi, seul l'arc $e = (u,v)$ disparait, les autres arcs incidents à v sont incidents à u dans G/e. Notons que toute coupe $\delta(U')$ dans G/e est aussi une coupe $\delta(U)$ dans G avec $U = U'$ si $u \notin U'$ et $U = U' \cup \{v\}$ si $u \in U'$. Donc la capacité minimale d'une coupe de G' est au moins celle d'une coupe de G.

L'algorithme randomisé est le suivant : choisir une arête aléatoirement avec une distribution proportionnelle aux capacités des arêtes, la contracter, et répéter ces deux opérations jusqu'à avoir un graphe à deux sommets. Un graphe à deux sommets possède une seule coupe.

Bornons la probabilité que l'algorithme choisisse une arête de $\delta(U)$. Pour cela, notons A_i l'évènement « à la i-ième étape, l'algorithme tire une arête qui n'est pas dans $\delta(U)$ », pour $1 \leq i \leq n-2$. La probabilité que l'arête choisie à la première étape soit dans $\delta(U)$ est au plus $2/n$, donc

$$\mathbf{P}(A_1) \geq 1 - \frac{2}{n}$$

Supposons ensuite que A_1 est vérifié. À la deuxième étape, il reste donc $n-1$ sommets, donc la probabilité de tirer une arête de $\delta(U)$ devient $2/(n-1)$, d'où

$$\mathbf{P}(A_2 \mid A_1) \geq 1 - \frac{2}{n-1}$$

À la i-ème étape, la nombre de sommets résiduels est $n-i+1$, donc la probabilité de tirer une arête de la coupe, sachant qu'on en a tiré aucune avant, est $2/(n-i+1)$, de sorte que

$$\mathbf{P}(A_i \mid A_1 \cap \cdots \cap A_{i-1}) \geq 1 - \frac{2}{n-i+1}$$

Par la formule des probabilités composées, la probabilité de ne tirer aucune des arêtes de la coupe (pendant les $n-2$ étapes) est donc

$$\begin{aligned}
\mathbf{P}(A_1 \cap \cdots \cap A_{n-2}) &= \mathbf{P}(A_1) \cdot \mathbf{P}(A_2|A_1) \cdot \mathbf{P}(A_3|A_1 \cap A_2) \cdots \mathbf{P}(A_{n-2}|A_1 \cap A_2 \cap \cdots \cap A_{n-3}) \\
&\geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) \\
&= \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} \\
&= \frac{\prod_{i=1}^{n-2} i}{\prod_{i=3}^{n} i} \\
&= \frac{2}{n(n-1)}
\end{aligned}$$

La probabilité de découverte de la coupe $\delta(U)$ est donc supérieure à $2/n(n-1)$. L'algorithme a donc de fortes chances de ne pas la trouver. Cependant, si on répète $n(n-1)/2$ fois la recherche, en faisant des choix probabilistes indépendants, la probabilité de ne toujours pas avoir trouvé la coupe minimum devient inférieure à

$$\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)/2} < \frac{1}{e}$$

Il suffit de renvoyer la coupe de capacité minimum parmi les $n(n-1)/2$ coupes qu'on obtient ainsi : avec probabilité au moins $1 - 1/e$, il s'agit d'une coupe minimum.

Comme pour l'exemple des requêtes dans les bases de données, si on pousse jusqu'à $n(n-1)\ln n/2$ tentatives, la probabilité de ne pas avoir trouvé la coupe minimum devient inférieure à

$$\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)\ln n/2} < \left(\frac{1}{e}\right)^{\ln n} = \frac{1}{n}$$

On peut donc faire tendre la probabilité d'échec vers 0 en augmentant le nombre de tentatives. C'est un exemple d'algorithme de Monte Carlo qui a une complexité garantie, et fournit un résultat correct avec une grande probabilité.

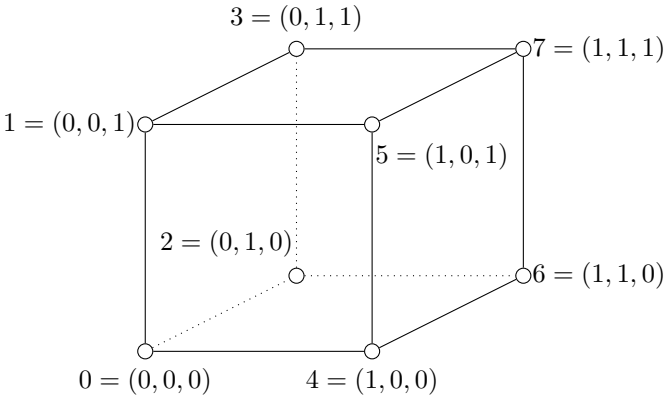
3.3 Routage dans un hypercube

Pour finir, revenons à un algorithme de Las Vegas pour un problème de routage dans un réseau. On modélise un tel réseau par un graphe dirigé à N sommets, chacun ayant un identifiant unique entre 1 et N, dont les arcs représentent les liens de communication. Chaque lien peut transporter une unité de message (un paquet) pendant une étape. À chaque étape, un sommet peut envoyer au plus un paquet à chacun de ses voisins. On considère le problème du routage par permutation où chaque sommet v souhaite envoyer un paquet au sommet $\pi(v)$ avec $\pi\colon V \to V$ une permutation des sommets de V : on cherche le nombre d'étapes totales nécessaires pour avoir transporté les N paquets à leur destinataire. Le chemin emprunté par un paquet de v à $\pi(v)$ est appelée une route. Le long de son chemin, un paquet peut devoir attendre dans un sommet du fait que l'arc suivant est occupé par un autre paquet. On suppose donc que chaque arc est muni d'une file d'attente qui héberge les paquets en attente d'être transmis via cette arc. Un algorithme de routage doit spécifier une route pour chaque paquet ainsi qu'une discipline de file pour résoudre les conflits entre paquets qui doivent être simultanément transmis sur un même arc.

On s'intéresse plus spécifiquement à la recherche d'un algorithme *sans mémoire*, c'est-à-dire tel que la route de v à $\pi(v)$ ne dépende que de v et $\pi(v)$, et pas de $\pi(u)$ pour $u \neq v$. Un tel algorithme est beaucoup plus simple à implémenter de manière distribuée. Il n'est pas très difficile de se convaincre que tout algorithme déterministe sans mémoire pour le problème de routage par permutation nécessite $\Omega(\sqrt{N/d})$ étapes dans le pire des cas, dans un réseau à N sommets où chaque sommet à degré d.

Considérons un cas particulier suffisamment général : le cas de l'hypercube de dimension n. On considère donc le graphe $G = (V, E)$ avec $V = \{0,1\}^n$ et $E = \{(u, v) \mid \|u - v\|_1 = 1\}$: ainsi, (u, v) est un arc si et seulement si au plus un bit diffère entre u et v. Il y a donc ici $N = 2^n$ sommets.

Un algorithme déterministe sans mémoire très simple pour router un paquet d'un sommet u à un sommet v consiste à aller d'abord au sommet obtenu en inversant le bit le plus à droite (le bit de poids le plus faible) qui est différent entre u et v, puis aller ensuite au sommet obtenu en inversant de plus le second bit le plus à droite, etc. jusqu'à arriver en v : on parle de *route par bit-fixing*. Par exemple, pour aller de $(1, 0, 1, 1)$ à $(0, 0, 0, 0)$ dans l'hypercube 4-dimensionnel, le paquet traverse d'abord $(1, 0, 1, 0)$, puis $(1, 0, 0, 0)$ sur sa route avant d'arriver à destination. En terme d'entiers représentés par ces vecteurs de bits, cela revient à dire que pour cheminer dans l'hypercube de dimension 4 du sommet 11 au sommet 0, on choisit de passer par les sommets 10 puis 8. Pour l'hypercube de dimension 3 (donc le cube), voici la représentation des sommets avec l'entier représentant chacun, ainsi que sa séquence de bits.



Par ailleurs, on suppose dans la suite que la discipline de file est FIFO (premier arrivé, premier sorti) en résolvant des arrivées simultanément arbitrairement : en fait, la discipline n'influe pas sur l'analyse qui suit et l'on pourrait en fixer une autre (comme, par exemple, toujours privilégier les paquets originaires du plus petit sommet possible pour un ordre arbitraire).

Si on considère une dimension n paire, et la permutation π telle que pour tout sommet $u = (a, b)$

(où a et b ont dimension $n/2$) on a $\pi(u) = (b, a)$, alors on obtient que la route par bit-fixing induit un nombre total d'étapes de l'ordre de $\Omega(\sqrt{2^n/n})$, comme la borne inférieure précédente l'affirme.

Étudions donc un algorithme randomisé bien plus efficace : il utilise un nombre $O(n)$ étapes avec probabilité proche de 1. C'est donc un algorithme de Las Vegas, toujours correct, mais de bonne complexité seulement avec une forte probabilité. Le principe est de décomposer le routage en deux phases : pour chaque sommet v,

— dans la phase 1, on choisit aléatoirement une destination intermédiaire $\sigma(v)$ (σ n'est pas nécessairement une permutation donc) et on route le paquet de v à $\sigma(v)$ en utilisant la route par bit-fixing ;

— dans la phase 2, on utilise la route par bit-fixing pour router le paquet de chaque sommet $\sigma(v)$ à $\pi(v)$.

Puisque chaque paquet choisit sa destination intermédiaire indépendamment des autres paquets, cet algorithme de routage est sans mémoire.

On analyse le nombre d'étapes nécessaires au routage dans la suite. Par symétrie des deux phases, il suffit de montrer que la phase 1 termine en un nombre d'étapes $O(n)$ avec forte probabilité. Pour chaque sommet v, on note γ_v la route empruntée par le paquet de v à $\sigma(v)$ dans la phase 1. Fixons un sommet u. Si u atteint sa destination à l'étape t alors que le chemin γ_v à longueur $|\gamma_v|$, on définit son retard comme $t - |\gamma_v|$: c'est le nombre d'étapes où le paquet a dû attendre dans une file. Voici le lemme clé de l'analyse :

Lemme 3.3.1. *Soit $S = \{v \in \{0,1\}^n \mid \gamma_u$ et γ_v partagent au moins un arc$\}$. Alors le retard de u est au plus $|S|$.*

Démonstration. La route par bit-fixing implique facilement que pour tous $u, v \in \{0,1\}^n$, si γ_u et γ_v se rencontrent puis divergent, ils ne peuvent plus se rencontrer à nouveau.

Fixons donc $u \in \{0,1\}^n$ et $\gamma_u = (e_1, e_2, \ldots, e_k)$. Pour $t \geq 1$, on définit le décalage de v au temps t, noté $\mathrm{dec}_t(v)$, comme étant $t - j$ si le paquet issu de v attend de passer l'arc e_j lorsque l'étape t commence, et 0 si v a déjà atteint sa destination au temps t. Lorsque $\mathrm{dec}_t(v) = d > 0$ et $\mathrm{dec}_{t+1}(v) = 0$, on dit que v est éjecté avec décalage d : on a alors $\mathrm{dec}_{t'}(u) = 0$ pour tout $t' \geq t+1$.

Notons T l'étape où u atteint sa destination : $\mathrm{dec}_1(u) = 0$ et $\mathrm{dec}_T(u) = T - k$ est le retard de u. Il suffit donc de montrer que $\mathrm{dec}_T(u) \leq |S|$.

Utilisons pour cela un argument de remplissage en prouvant que si $\mathrm{dec}_T(u) \geq d$ pour un certain $d \geq 1$ alors au moins un sommet v est éjecté avec décalage d. Ainsi, si $\mathrm{dec}_T(u) = L$, il doit exister des sommets éjectés avec un décalage $d = 1, 2, \ldots, L$, donc $L \leq |S|$ (puisque, d'après la première remarque de la preuve, chaque sommet est éjecté au plus une fois).

Considérons donc un entier ℓ tel que $\mathrm{dec}_t(v) = d$ pour un certain t et v. Supposons que t et v sont choisis de sorte que t est le dernier instant où $\mathrm{dec}_t(v) = d$: un tel instant existe puisque $\mathrm{dec}_{T+1}(v) = 0$ pour tout v. Soit $j = t - d$. Puisque $\mathrm{dec}_t(v) = d$, le paquet issu de v attend de passer l'arc e_j lorsque l'étape t commence. Un paquet traverse donc l'arc e_j lors de l'étape $t + 1$: c'est le paquet issu d'un certain sommet w. Montrons donc que w est éjecté avec décalage d au temps t : en effet, si w n'est pas éjecté, alors $\mathrm{dec}_{t+1}(w) = (t + 1) - (j + 1) = d$ ce qui contredit l'hypothèse sur t et v. □

Pour $v \in \{0,1\}^n$, on définit alors une variable aléatoire

$$H_{uv} = \begin{cases} 1 & \text{si } \gamma_u \text{ et } \gamma_v \text{ partagent au moins un arc} \\ 0 & \text{sinon} \end{cases}$$

Ainsi, par le lemme précédent,

$$\text{retard de } u \leq |S| = \sum_v H_{uv}$$

Par suite, majorer le probabilité que le retard de u soit grand revient à majorer la probabilité que $\sum_v H_{uv}$ soit grand. On peut donc appliquer les inégalités de Markov ou de Bienaymé-Tchebychev

pour cela : malheureusement cela n'est pas suffisant et n'utilise pas la particularité que la variable aléatoire $\sum_v H_{uv}$ est une somme de variables aléatoires indépendantes (puisque les destinations intermédiaires $\sigma(v)$ sont choisies uniformément et indépendamment). Une autre inégalité existe pour ce cas-là, qu'on commence donc par découvrir.

Inégalité de Chernoff Alors que les inégalités de Markov et de Bienaymé-Tchebychev sont des inégalités utilisant l'espérance et la variance pour trouver des bornes linéaires ou quadratiques sur la queue d'une distribution de variable aléatoire (« quelle est la probabilité qu'une variable X soit β plus grande que sa moyenne ? », par exemple), l'inégalité de Chernoff permet d'obtenir une décroissance exponentielle beaucoup plus forte. Elle se base sur la fonction génératrice des moments $\mathbf{E}(e^{tX})$ qu'on appelle ainsi du fait qu'elle peut se décomposer en série entière $\sum_{k\geq 0} \frac{t^k \mathbf{E}(X^k)}{k!}$, la quantité $\mathbf{E}(X^k)$ s'appelant le moment d'ordre k de X. En contrepartie, l'inégalité de Chernoff nécessite de pouvoir décomposer la variable X comme une somme $X_1 + \cdots + X_n$ de variables aléatoires à valeurs dans $\{0, 1\}$ et indépendantes : la somme en exposant se transforme alors en produit dont on arrive à borner l'espérance.

Théorème 3.3.1 (Inégalité de Chernoff)**.** *Soit $X_1, \ldots, X_n$ des variables aléatoires de Bernoulli indépendantes telles que $\mathbf{P}[X_i = 1] = p_i$, avec $0 < p_i < 1$ pour tout i. Alors, si $X = X_1 + \cdots + X_n$, $m = \mathbf{E}(X) = p_1 + \cdots + p_n$ et $\delta > 0$*

$$\mathbf{P}[X > (1+\delta)m] < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^m$$

Démonstration. Soit t un réel strictement positif. Alors

$$\mathbf{P}[X > (1+\delta)m] = \mathbf{P}[\exp(tX) > \exp(t(1+\delta)m)]$$

Appliquons l'inégalité de Markov à droite :

$$\mathbf{P}[X > (1+\delta)m] < \frac{\mathbf{E}(\exp(tX))}{\exp(t(1+\delta)m)}$$

L'inégalité est ici bien stricte car l'hypothèse sur p_i implique que X peut prendre plusieurs valeurs (refaites la preuve pour vous en assurer !). Observons alors que

$$\mathbf{E}(\exp(tX)) = \mathbf{E}\left(\prod_{i=1}^n \exp(tX_i)\right)$$

Puisque les X_i sont indépendants, on prouve aisément que les variables $\exp(tX_i)$ sont aussi indépendantes, de sorte que $\mathbf{E}(\prod_{i=1}^n \exp(tX_i)) = \prod_{i=1}^n \mathbf{E}(\exp(tX_i))$. La variable e^{tX_i} a la valeur e^t avec probabilité p_i et 1 avec probabilité $1 - p_i$, on peut donc facilement calculer son espérance, d'où

$$\mathbf{P}[X > (1+\delta)m] < \frac{\prod_{i=1}^n (1 + p_i(e^t - 1))}{\exp(t(1+\delta)m)}$$

Grâce à l'inégalité $1 + x < e^x$ avec $x = p_i(e^t - 1)$, on obtient

$$\begin{aligned}\mathbf{P}[X > (1+\delta)m] &< \frac{\prod_{i=1}^n \exp(p_i(e^t - 1))}{\exp(t(1+\delta)m)} \\ &= \frac{\exp(\sum_{i=1}^n p_i(e^t - 1))}{\exp(t(1+\delta)m)} \\ &= \frac{\exp(m(e^t - 1))}{\exp(t(1+\delta)m)}\end{aligned}$$

En choisissant alors t donnant la meilleure borne possible (ce qu'on peut faire en dérivant le membre droit), c'est-à-dire $t = \ln(1+\delta)$, on obtient la borne souhaitée. □

En particulier, on obtient aisément une borne plus large :

$$\mathbf{P}[X > (1+\delta)m] < \left(\frac{e}{1+\delta}\right)^{(1+\delta)m}$$

qui permet d'obtenir la borne suivante, dès lors que $\delta > 2e - 1 \approx 4,4$:

$$\mathbf{P}[X > (1+\delta)m] < 2^{-(1+\delta)m} \tag{3.1}$$

Utilisation de l'inégalité de Chernoff pour le routage dans l'hypercube On ne peut pas directement appliquer l'inégalité de Chernoff à $\sum_v H_{uv}$ car $\mathbf{E}(H_{uv})$ est difficile à estimer. On fait donc un détour. Pour tout arc $e \in E$, on note $N(e)$ le nombre de chemins γ_v qui utilise l'arc e, pour $v \in \{0,1\}^n$. Par symétrie, on a $\mathbf{E}(N(e)) = \mathbf{E}(N(e'))$ pour deux arcs $e, e' \in E$ quelconques. Ainsi, pour tout $e_0 \in E$,

$$|E|\mathbf{E}(N(e_0)) = \sum_{e\in E} \mathbf{E}(N(e)) = \sum_{v\in\{0,1\}^n} \mathbf{E}(|\gamma_v|)$$

La longueur de γ_v étant le nombre de bits différents dans v et $\sigma(v)$, on a aisément $\mathbf{E}(|\gamma_v|) = n/2$. Par conséquent,

$$\mathbf{E}(N(e_0)) = \frac{2^n n}{2|E|} = \frac{2^n n}{2 \times 2^n n} = \frac{1}{2}$$

Ainsi, pour une route $\gamma = (e_1, e_2, \ldots, e_k)$ fixée, on a

$$\mathbf{E}(|\{v \mid \gamma \text{ et } \gamma_v \text{ partagent au moins un arc}\}|) \leq \mathbf{E}\left(\sum_{i=1}^{k} N(e_i)\right) = \frac{k}{2} \leq \frac{n}{2}$$

Cela implique finalement que

$$\mathbf{E}\left(\sum_v H_{uv}\right) \leq \frac{n}{2}$$

On applique alors la borne de Chernoff (3.1) avec $\delta = 5 > 2e - 1$ pour conclure que

$$\mathbf{P}[\text{retard de } u > 3n] < 2^{-3n}$$

En majorant la probabilité d'une union par la somme des probabilités, puis en prenant le complémentaire, on obtient

$$\mathbf{P}[\forall u \in \{0,1\}^n \text{ retard de } u \leq 3n] > 1 - 2^{-3n}2^n = 1 - 2^{-2n}$$

Ainsi, avec probabilité au moins $1 - 2^{-2n}$, chaque paquet arrive à destination de la phase 1 en au plus $3n$ étapes. En cumulant les deux phases et en majorant 2×2^{-2n} par $1/N = 2^{-n}$, on obtient

Théorème 3.3.2. *Avec probabilité au moins $1 - 1/N$, chaque paquet atteint sa destination en au plus $6n$ étapes.*

Chapitre 4

Chaînes de Markov

4.1 Marcheur aléatoire

Une marche aléatoire dans un graphe orienté ou non orienté $G = (V, E)$ est la création d'un chemin dans G depuis un sommet $u \in V$ donné où chaque successeur est choisi aléatoirement de manière uniforme sur l'ensemble de ses voisins : si on se trouve dans le sommet $v \in V$, on choisit comme successeur un des sommets w tels que $(v, w) \in E$, avec probabilité $1/|\{(v, w) \in E\}|$. Le choix à une étape est donc indépendant des choix faits aux étapes précédentes : si on note V_i la variable aléatoire décrivant le sommet où se trouve le marcheur aléatoire après i étapes, cela veut dire que, pour $v_1, v_2, \ldots, v_i, v_{i+1} \in V$:

$$\mathbf{P}[V_{i+1} = v_{i+1} \mid V_1 = v_1, V_2 = v_2, \ldots, V_i = v_i] = \mathbf{P}[V_{i+1} = v_{i+1} \mid V_i = v_i]$$

c'est-à-dire que la probabilité $\mathbf{P}[V_{i+1} = v_{i+1}]$ ne dépend que de la position à l'étape précédente.

Voici quelques questions qu'on peut se poser dans ce cadre :

— quel est le nombre moyen d'étapes nécessaires pour aller d'un sommet u à un sommet v pour la première fois ? On parle de *temps moyen de premier passage*.

— à partir d'un sommet u, combien d'étapes sont nécessaires en moyenne pour visiter tous les sommets du graphe ? On parle de *temps moyen de couverture*.

On peut résoudre certaines de ces questions (en particulier pour des cas particuliers tels qu'un graphe en forme de segment, ou une clique) avec des arguments de probabilités élémentaires. Par exemple, on peut montrer de manière élémentaire que le temps moyen de couverture depuis une extrémité d'un graphe en forme de segment à n sommets est $(n-1)^2$, alors que le temps moyen de couverture d'un graphe complet à n sommet est de l'ordre de $n \log n$.

Les marches aléatoires sont la source d'innombrables algorithmes randomisés. Utilisons par exemple une marche aléatoire pour résoudre le problème 2-SAT de logique. Il s'agit d'un cas particulier du problème SAT, de satisfaisabilité d'une formule de la logique propositionnelle, au cas de formules en forme canonique conjonctive $\bigwedge_i (\ell_{i,0} \vee \ell_{i,1} \vee \cdots \vee \ell_{i,n_i})$ (avec $\ell_{i,j}$ un littéral, c'est-à-dire une variable propositionnelle ou sa négation), où chaque clause admet deux littéraux (c'est-à-dire $n_i = 2$). Le problème SAT est NP-complet, mais le problème 2-SAT est lui résoluble en temps polynomial. Appliquons une sorte de marche aléatoire dans l'hypercube de dimension n (où n est le nombre de variables dans la formule) pour le résoudre très simplement. Les sommets du graphe que nous considérons alors consistent en les valuations des variables propositionnelles. Les arêtes du graphe consistent à modifier de la manière suivante une telle valuation :

— à partir d'une valuation ν des variables ne satisfaisant pas la formule, on choisit aléatoirement uniformément une clause de la formule qui n'est pas satisfaite ;

— on choisit alors aléatoirement uniformément l'un des deux littéraux de la clause et on inverse sa valuation pour trouver la nouvelle valuation.

On poursuit la marche aléatoire tant qu'on n'a pas rencontré une valuation satisfaisant la formule. Montrons que si la formule est satisfaisable, le temps moyen pour que la marche aléatoire

précédente ait trouvé une valuation la satisfaisant est en $O(n^2)$ où n est le nombre de variables. On peut aisément en déduire un algorithme de Monte-Carlo pour décider si une formule de 2-SAT est satisfaisable.

Pour cela, changeons de modélisation. Fixons une valuation ν satisfaisant la formule (qui existe puisqu'on suppose désormais la formule satisfaisable). Pour une autre valuation ν', conservons uniquement en mémoire le nombre X de variable dont les valeurs coincident dans ν et ν' : ce nombre fait donc partie de $\{0, 1, 2, \ldots, n\}$. À chaque étape de l'algorithme, on change la valeur d'une variable exactement de sorte que X change d'une unité : si la valeur est $0 < i < n$ à une étape, à l'étape suivante la valeur est $i-1$ ou $i+1$. Si la valeur est 0, on ne peut que passer à 1 et on a trouvé la valuation ν dès lors que la valeur devient égale à n. Puisque nous considérons à chaque étape une clause insatisfaite, on est sûr qu'un des deux littéraux au moins a une valeur courante différente de celle dans ν : ainsi, avec probabilité au moins $1/2$, on augmente de 1 la valeur de X. Autrement dit, la valeur de X se déplace sur le segment $\{0, 1, 2, \ldots, n\}$ avec une probabilité supérieure à $1/2$ d'être augmentée en une étape. Ce n'est pas exactement une marche aléatoire, mais il s'agit d'un modèle plus générale de chaîne de Markov que nous introduisons désormais.

4.2 Chaînes de Markov discrètes

Une chaîne de Markov est un processus stochastique défini sur un ensemble (fini ou dénombrable) S d'états à l'aide d'une matrice P de probabilités de transitions. La matrice P a une ligne et une colonne par état de S. La chaîne de Markov est dans un état à la fois et réalise une transition par temps $t = 1, 2, \ldots$. La case $P_{i,j}$ de la matrice est la probabilité qu'étant dans l'état i, le prochain état sera l'état j. Ainsi, pour tout $i, j \in S$, on a $0 \leq P_{i,j} \leq 1$ et $\sum_{j \in S} P_{i,j} = 1$. Cette dernière égalité signifie que chaque ligne de la matrice P somme à 1.

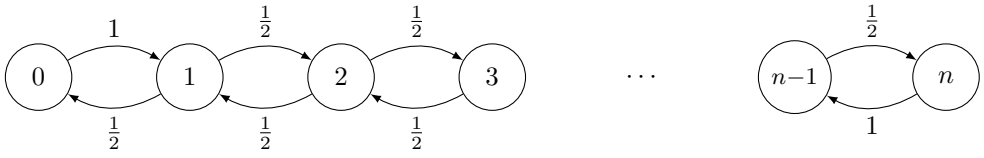
La propriété cruciale d'une chaîne de Markov est son absence de mémoire : le comportement futur de la chaîne ne dépend que de son état courant, pas de son histoire. Cela explique la modélisation à base d'une matrice de probabilités de transitions. Si on note X_t l'état de la chaîne de Markov à l'étape t, alors on a (comme pour la marche aléatoire) :

$$\mathbf{P}[X_{t+1} = j \mid X_0 = i_0, X_1 = i_1, \ldots, X_t = i] = \mathbf{P}[X_{t+1} = j \mid X_t = i] = P_{i,j}$$

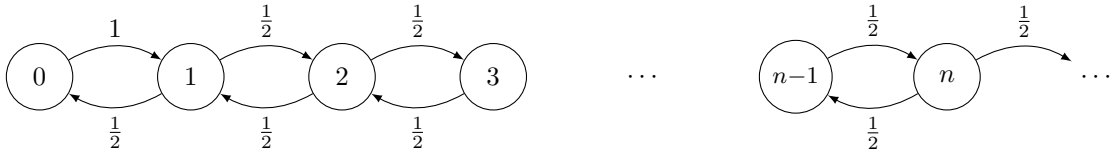
La marche aléatoire sur un segment de n sommets peut donc s'écrire de la manière suivante : $S = \{0, 1, 2, \ldots, n-1\}$ et

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 \end{pmatrix}$$

Cette matrice est la matrice d'adjacence d'un graphe pondéré, qu'on appelle *graphe sous-jacent de la chaîne de Markov*. Sur l'exemple précédent, il s'agit du graphe



Une chaîne de Markov peut être infinie (dénombrable). Par exemple, on peut généraliser la marche aléatoire précédente au cas d'une demi-droite infinie. Les états sont alors $S = \mathbf{N}$ et la matrice P est nulle sauf les coefficients $P_{0,1} = 1$ et $P_{i-1,i} = P_{i,i+1} = \frac{1}{2}$ pour $i > 0$:



Pour initialiser une chaîne de Markov, on donne un état initial. En général, on autorise que l'état initial soit une variable aléatoire avec une distribution quelconque $\pi^{(0)}$ associant à chaque état s la probabilité $\pi_s^{(0)}$ qu'il soit l'état initial : on a donc $\sum_{s\in S}\pi_s^{(0)} = 1$. Après une étape, une nouveau distribution de probabilité $\pi^{(1)}$ sur les états est obtenue. Plus généralement, après t étapes, on note $\pi^{(t)}$ la distribution obtenue. Par la formule des probabilités totales, on sait que

$$\pi_s^{(t+1)} = \mathbf{P}[X_{t+1} = s] = \sum_{i\in S}\mathbf{P}[X_t = i]\mathbf{P}[X_{t+1} = s \mid X_t = i] = \sum_{i\in S}\pi_i^{(t)}P_{i,s}$$

On reconnait là un produit matriciel entre un vecteur ligne $\pi^{(t)}$ et une matrice carrée P, de sorte qu'on peut réécrire cette égalité via

$$\pi^{(t+1)} = \pi^{(t)}P$$

En particulier, en notant P^t le produit $\underbrace{P \times P \times \cdots \times P}_{t \text{ fois}}$ (et donc P^0 est la matrice identité), on a pour tout $t \geq 0$

$$\pi^{(t)} = \pi^{(0)}P^t$$

En particulier, une distribution π est dite *stationnaire*, si $\pi = \pi P$: si la chaîne de Markov est initialisée dans cette distribution stationnaire, alors la distribution n'évolue plus et reste constamment la même. Certaines chaînes ne possèdent pas de distribution stationnaire ; d'autres en possèdent plusieurs ; certaines, enfin, en possèdent une unique qui contient alors de riches informations sur la chaîne de Markov, comme nous allons le voir.

Revenons au problème initial consistant à calculer un temps moyen de premier passage ou un temps moyen de couverture dans une chaîne de Markov. Le temps moyen de premier passage menant d'un état i à un autre état j de la chaîne de Markov qu'on notera $h_{i,j}$ (pour *hitting time*) est une espérance :

$$h_{i,j} = \sum_{t>0} t a_{i,j}^{(t)}$$

où $a_{i,j}^{(t)}$ est la probabilité que l'état j soit atteint pour la première fois à l'étape $t > 0$ en partant de l'état i :

$$a_{i,j}^{(t)} = \mathbf{P}[X_t = j \wedge \forall 1 \leq s \leq t-1\ X_s \neq j \mid X_0 = i]$$

À partir de ces probabilités, on peut aussi en déduire la probabilité que l'état j soit visité un jour en partant de l'état i :

$$f_{i,j} = \sum_{t>0} a_{i,j}^{(t)}$$

Noter que si $f_{i,j} < 1$, cela veut dire qu'il y a une probabilité non nulle de ne jamais visiter l'état i, ce qui implique que l'espérance $h_{i,j}$ n'est pas définie, c'est-à-dire vaut $h_{i,j} = \infty$. Mais la réciproque n'est pas forcément vraie : il se peut qu'on puisse aller de i à j avec probabilité 1, mais que le temps moyen pour se faire soit infini. Cela nous amène à distinguer trois types d'états :

- un état i tel que $f_{i,i} < 1$ (et donc $h_{i,i} = \infty$) est dit *transient* : on finit par le quitter et ne jamais y revenir ;
- un état i tel que $f_{i,i} = 1$ est dit *récurrent* :
  - si, de plus, $h_{i,i} = \infty$, l'état i est dit *récurrent nul* ;
  - dans le cas contraire, l'état i est dit *récurrent non-nul*.

Dans une chaîne de Markov finie, aucun état ne peut être récurrent nul : tous les états sont donc soit transient, soit récurrent non-nul. Dans la suite de ce cours, on se limite aux chaînes de Markov finis : on appelle donc simplement récurrents les états récurrents non-nuls.

On l'a vu, une chaîne de Markov finie peut être représentée comme un graphe orienté dont les sommets sont les états de la chaîne et dans lequel il existe un arc de i à j si $P_{i,j} > 0$: dans la suite, on confond la chaîne de Markov et son graphe sous-jacent. Une composante fortement connexe du graphe de la chaîne est dite *terminale* dès lors qu'aucun arc n'en sort. En partant d'un état d'une composante fortement connexe de la chaîne, il y a une probabilité non nulle d'aller à tout autre sommet de la même composante en un nombre fini d'étapes. Si de plus C est une composante terminale, cette probabilité est 1 puisque la chaîne ne peut pas sortir de C une fois qu'elle y est entrée. Un état est donc persistant si et seulement s'il fait partie d'une composante terminale.

Une chaîne de Markov est dite *irréductible* si son graphe est fortement connexe : il n'y a donc qu'une unique composante terminale qui est la chaîne entière. Tous les états y sont donc persistants. Les chaînes irréductibles admettent une unique distribution stationnaire qui vérifie de plus les propriétés suivantes :

Théorème 4.2.1. *Toute chaîne de Markov (finie) irréductible possède une unique distribution stationnaire π : pour tout état i, π_i est non nul et vaut $\pi_i = 1/h_{i,i}$. De plus, si l'on note $N(i,t)$ le nombre de fois que la chaîne de Markov visite l'état i dans les t premières étapes (en partant d'une distribution initiale quelconque), alors*

$$\lim_{t\to\infty} \frac{N(i,t)}{t} = \pi_i$$

Ce théorème dit en particulier que le temps moyen de premier retour dans un état i d'une chaîne irréductible vaut $1/\pi_i$. Il « suffit » donc de savoir trouver l'unique distribution stationnaire, vérifiant $\pi = \pi P$, pour en déduire des informations sur le temps moyen de premier retour.

Il pourrait être tentant de déduire du théorème précédent que la distribution stationnaire est la distribution limite si on laisse tourner la chaîne de Markov à partir de n'importe quelle distribution initiale. Il n'en est malheureusement rien. Considérer par exemple une chaîne de Markov à deux états 1 et 2 telle que

$$P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Une distribution $\pi = (a, b)$ est stationnaire si et seulement si $\pi = \pi P$, c'est-à-dire $a = b$. Du fait que $a + b = 1$, on en déduit que $a = b = 1/2$ et on vérifie que $(1/2, 1/2)$ est bien une distribution stationnaire de la chaîne. Elle est irréductible, donc on sait que c'est la seule distribution stationnaire. Le résultat du théorème sur le temps moyen de premier retour est aussi vrai : il faut toujours 2 étapes pour revenir dans un état une fois qu'on l'a quitté. Par contre, cette chaîne n'admet pas forcément une distribution limite. Si on part de la distribution $\pi = (1, 0)$ par exemple, alors on atteint successivement les distributions $(0, 1), (1, 0), (0, 1)$, etc. Ainsi, $(\pi^{(n)})_{n\in\mathbf{N}}$ ne converge pas nécessairement vers la distribution limite. Dans cet exemple, la raison en est la présence d'un unique cycle de longueur 2 qui sert de vase communicant empêchant de converger vers une distribution particulière. C'est en fait la seule raison comme le montre le prochain théorème.

La *périodicité* d'un état i (faisant partie d'une composante fortement connexe non réduite à ce seul état i) est le plus grand commun diviseur des longueurs des cycles auxquels il appartient. Un état est dit *périodique* s'il a une périodicité différente de 1 ; il est dit *apériodique* s'il a périodicité 1. Une chaîne de Markov dont tous les états sont apériodiques est elle-même appelée *apériodique*.

Théorème 4.2.2. *Une chaîne de Markov (finie) irréductible et apériodique (on dit parfois qu'une telle chaîne de Markov est* ergodique*) admet une unique distribution limite : pour toute distribution $\pi^{(0)}$, la suite $(\pi^{(n)})_{n\in\mathbf{N}}$ des distributions $\pi^{(n)} = \pi^{(0)}P^n$ converge vers l'unique distribution stationnaire.*

4.3 Temps moyen de couverture d'une marche aléatoire

Revenons finalement au temps moyen de couverture d'une marche aléatoire, en nous concentrant sur le cas d'un graphe non orienté $G = (V, E)$. La chaîne de Markov correspondant à cette marche aléatoire a pour états les sommets V et pour transitions de probabilité

$$P_{i,j} = \begin{cases} \frac{1}{\text{degré}(i)} & \text{si } (i,j) \in E \\ 0 & \text{sinon} \end{cases}$$

Noter la ressemblance avec la matrice d'adjacence du graphe G. Étudions désormais des propriétés de cette chaîne de Markov.

Lemme 4.3.1. *La chaîne de Markov associée à la marche aléatoire sur G est irréductible si et seulement si le graphe G est connexe.*

Démonstration. Immédiat. □

On suppose donc dans la suite que le graphe G est connexe.

Lemme 4.3.2. *La chaîne de Markov associée à la marche aléatoire sur G est apériodique si et seulement si le graphe G n'est pas biparti.*

Démonstration. Un graphe est biparti si et seulement s'il n'a pas de cycles ayant un nombre impair d'arêtes. De plus, il existe toujours, dans un graphe connexe non réduit à un sommet, un chemin de longueur 2 d'un sommet à lui-même. Si le graphe est biparti, la chaîne est donc périodique (de période 2 pour tous les états). Si le graphe n'est pas biparti alors, puisqu'il est connexe, il possède un cycle non trivial de longueur impaire de tout sommet vers lui-même. La chaîne est donc apériodique. □

Il est temps d'appliquer les résultats vus précédemment :

Lemme 4.3.3. *Si le graphe G est connexe et n'est pas biparti, alors la chaîne de Markov converge vers une distribution stationnaire π telle que $\pi_v = \frac{degré(v)}{2|E|}$.*

Démonstration. Puisque la chaîne est irréductible et apériodique, la chaîne possède une unique distribution stationnaire π, qui est aussi la distribution limite. Il suffit donc de vérifier que la distribution donnée dans l'énoncé vérifie bien $\pi = \pi P$, pour pouvoir conclure par unicité. D'abord, c'est bien une distribution de probabilités puisque $\sum_{v \in V} d(v) = 2|E|$. De plus, en notant $N(v)$ l'ensemble des voisins de v dans le graphe, on a

$$(\pi\, P)_v = \sum_{u \in N(v)} \frac{\text{degré}(u)}{2|E|} \frac{1}{\text{degré}(u)} = \frac{\text{degré}(v)}{2|E|} = \pi_v$$

□

Rappelons qu'on a noté $h_{i,j}$ le nombre moyen d'étapes pour atteindre j depuis i. Avant de l'utiliser pour borner le temps moyen de couverture, commençons par un résultat intermédiaire qu'on prouve en changeant de chaîne de Markov :

Lemme 4.3.4. *Si le graphe G est connexe et non biparti, et si $\{u, v\} \in E$, alors $h_{u,v} + h_{v,u} \leq 2|E|$.*

Démonstration. On change de point de vue en considérant désormais une chaîne de Markov définie sur les arêtes de G : l'état courant est plus précisément la donnée d'une arête et de son sens de traversée, c'est donc un arc (dirigé). Il y a donc $2|E|$ états dans cette nouvelle chaîne de Markov. La matrice de transition Q a pour coefficients non nuls tous les

$$Q_{(u,v),(v,w)} = P_{v,w} = \frac{1}{\text{degré}(v)}$$

Cette matrice a une particularité : elle est doublement stochastique, au sens que non seulement ses lignes somment à 1 (comme toute matrice de transition de probabilités), mais aussi ses colonnes :

$$\begin{aligned}\sum_{x\in V, y\in N(x)} Q_{(x,y),(v,w)} &= \sum_{u\in N(v)} Q_{(u,v),(v,w)} \\ &= \sum_{u\in N(v)} P_{v,w} \\ &= \text{degré}(v) \times \frac{1}{\text{degré}(v)} = 1\end{aligned}$$

Cela peut se réécrire $(1,1,\ldots,1)Q = (1,1,\ldots,1)$. Dans ce cas particulier, la distribution uniforme π (telle que $\pi_{(u,v)} = \frac{1}{2|E|}$) est donc une distribution stationnaire.

Cette nouvelle chaîne de Markov est irréductible et apériodique, donc le temps moyen entre deux passages via l'arc dirigé (v,u) est $2|E|$.

Considérons désormais la quantité $h_{u,v}+h_{v,u}$ dans la chaîne de Markov originelle. Interprétons-la comme le temps moyen pour la marche partant dans le sommet u de visiter le sommet v avant de repartir vers u. Conditionné par l'évènement que la première arrivée dans u était par l'arc (v,u), on conclut que le temps moyen pour aller de là à v et de retourner à u par l'arc (v,u) est $2|E|$. L'absence de mémoire de la chaîne de Markov nous permet de retirer ce conditionnement : puisque la suite de transitions tirées après u est indépendante du fait qu'on est arrivé dans u via (v,u) au début, le temps moyen pour aller de u à v puis pour retourner à u par l'arc (v,u) est $2|E|$. C'est donc une borne supérieure sur $h_{u,v} + h_{v,u}$, qui est le temps d'aller de u à v puis à u (sans contrainte sur l'arc qu'on doit prendre pour cela). □

Nous pouvons alors en déduire :

Théorème 4.3.1. *Le temps moyen de couverture d'un graphe G connexe et non biparti depuis un sommet arbitraire v_0 est majoré par $2|E|(|V|-1)$.*

Démonstration. Soit T un arbre couvrant de G. Il existe un cycle (eulérien) sur cet arbre qui traverse chaque arête de T deux fois (une fois dans chaque direction). Soit $v_0, v_1, \ldots, v_{2|V|-2} = v_0$ la suite de sommets du tour. Le temps moyen pour visiter successivement les sommets du tour dans l'ordre est une borne supérieure sur le temps moyen de couverture. Ainsi, ce temps moyen de couverture est majoré par

$$\sum_{i=0}^{2|V|-3} h_{v_i,v_{i+1}} = \sum_{\{u,w\}\in T} (h_{u,w} + h_{w,u}) \leq 2|E|(|V|-1)$$

□

En particulier, pour la marche aléatoire sur un segment comprenant n sommets, le temps moyen de couverture est majoré par $2n^2$ (on peut en fait montrer que, dans ce cas précis, la valeur exacte est n^2). La chaîne de Markov concernant l'algorithme randomisé de résolution du problème 2-SAT ressemble à cette marche aléatoire sur le segment, à la différence près que la chaîne a une plus forte probabilité de se décaler vers les valeurs croissantes des états... Cela implique que le temps moyen de couverture de cette chaîne, et en particulier le temps moyen de premier passage de n'importe quel état à l'état n, est majoré par $2n^2$. Grâce à l'inégalité de Markov, on peut en déduire que si on laisse tourner l'algorithme pendant $4n^2$ étapes, on a probabilité $1/2$ de se tromper seulement.

4.4 Applications au graphe du web

Les algorithmes de recherche tels que Google (au début de son existence en tout cas, puisque les méthodes changent continuellement) utilisent l'algorithme PageRank (inventé par Larry Page, co-fondateur de Google) pour ordonner les résultats de la recherche, en fonction de la popularité

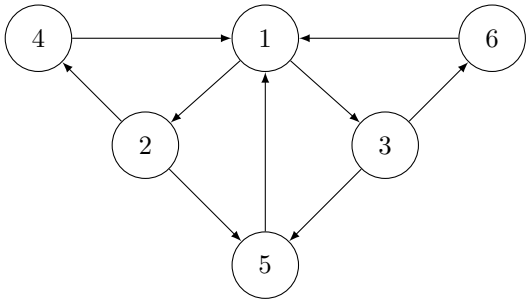
des pages web. L'algorithme suppose qu'Internet consiste en une collection de pages web avec des hyperliens pointant vers d'autres pages web. L'algorithme de PageRank simule un surfeur aléatoire qui visite Internet en décidant (probabilistiquement) s'il clique sur l'un des hyperliens de la page qu'il visite, ou s'il entre une nouvelle URL dans la barre d'adresse.

Supposons qu'il y a N pages web dans le monde, nommées $p_1, \ldots, p_N$. L'algorithme de PageRank crée une chaîne de Markov avec les N états $\{1, \ldots, N\}$. Si la page p_i a des liens vers toutes les autres pages du web, alors on saute de p_i à p_j avec probabilités $P_{i,j} = \frac{1}{N-1}$ pour tout $j \neq i$. Si la page p_i a des liens vers $N' < N-1$ autres pages, alors on pose $P_{i,j} = \frac{0.85}{N'}$ si p_i a un lien vers la page p_j et $P_{i,j} = \frac{0.15}{N-N'-1}$ sinon, pour tout $j \neq i$. La distribution initiale de la chaîne de Markov est uniforme sur tous les états.

Dès que $N > 2$, le graphe de la chaîne de Markov est fortement connexe, donc la chaîne est irréductible. De plus, la chaîne contient des cycles de toutes les longueurs supérieures ou égales à 2, donc la chaîne est apériodique. Ainsi, les théorèmes 4.2.1 et 4.2.2 assurent qu'il existe une unique distribution stationnaire qu'on peut obtenir en laissant tourner la chaîne de Markov pendant suffisamment longtemps. Cette distribution stationnaire est une mesure de la popularité des pages, puisque π_i est égal au nombre moyen de fois que la page p_i est vue pendant un surf aléatoire. Cette mesure de popularité reflète donc qu'une page est populaire si elle a beaucoup de pages populaires qui pointent vers elle...

Notez que, si on ne donne pas la possibilité au surfeur aléatoire de sauter vers une page web autre que celles pointées par la page courante, alors la chaîne de Markov n'est plus irréductible. Cela empêche donc la convergence de la distribution de la chaîne. Il est donc important de fixer une petite probabilité (ici 15%) de recommencer un nouveau surf aléatoire plutôt que de continuer le surf courant (avec 85% de chances).

Par exemple, le réseau ci-dessous est périodique (de période 3), mais la chaîne de Markov possède un graphe sous-jacent complet ce qui explique qu'elle soit irréductible et apériodique.



Commençons par étudier la chaîne de Markov de la marche aléatoire dans ce graphe. En partant de la distribution initiale uniforme, voici les distributions qu'on observe pas après pas :

$$
\begin{gathered}
(0.17, 0.17, 0.17, 0.17, 0.17, 0.17) \\
(0.5, 0.08, 0.08, 0.08, 0.17, 0.08) \\
(0.33, 0.25, 0.25, 0.04, 0.08, 0.04) \\
(0.17, 0.17, 0.17, 0.13, 0.25, 0.13) \\
(0.5, 0.08, 0.08, 0.08, 0.17, 0.08) \\
(0.33, 0.25, 0.25, 0.04, 0.08, 0.04) \\
(0.17, 0.17, 0.17, 0.13, 0.25, 0.13) \\
(0.5, 0.08, 0.08, 0.08, 0.17, 0.08) \\
\vdots
\end{gathered}
$$

On remarque qu'on atteint un cycle de longueur 3 très rapidement... Il n'y a donc pas de distribution limite. Par contre, la chaîne étant irréductible, elle possède une unique distribution stationnaire :

$$(0.33, 0.17, 0.17, 0.08, 0.17, 0.08)$$

La chaîne de Markov modifiée pour permettre de calculer le PageRank présente, elle, la dynamique suivante, toujours en partant de la distribution uniforme :

$$\begin{gathered}(0.17, 0.17, 0.17, 0.17, 0.17, 0.17)\\(0.44, 0.10, 0.10, 0.10, 0.16, 0.10)\\(0.33, 0.21, 0.21, 0.07, 0.11, 0.07)\\(0.25, 0.16, 0.16, 0.12, 0.20, 0.12)\\(0.38, 0.13, 0.13, 0.10, 0.16, 0.10)\\\vdots\\(0.325, 0.162, 0.162, 0.096, 0.158, 0.096)\\(0.322, 0.161, 0.161, 0.098, 0.161, 0.098)\\(0.327, 0.160, 0.160, 0.098, 0.159, 0.097)\\\vdots\end{gathered}$$

On a donné ci-dessus les distributions $\pi^{(0)}$ à $\pi^{(4)}$, puis $\pi^{(17)}$ à $\pi^{(19)}$. On voit que cette fois-ci une distribution limite se dessine. En effet, la distribution stationnaire est :

$$(0.333, 0.165, 0.165, 0.100, 0.164, 0.100)$$

C'est d'ailleurs le score PageRank pour ce réseau. La page p_1 est donc la plus populaire, suivie par les pages p_2, p_3, puis la page p_5 de peu, puis les pages p_4 et p_6 à égalité.

Calculer la distribution stationnaire pour le gigantesque graphe du web est impossible. Par contre, simuler un surfeur aléatoire est chose aisée. On voit que si on attend suffisamment longtemps, on obtient une distribution proche de la distribution stationnaire comme distribution limite.

Que se passe-t-il désormais si des pages web s'ajoutent ou se suppriment, ou bien que des hyperliens sont ajoutés ou supprimés entre les pages existantes ? La chaîne de Markov du calcul du PageRank change, mais reste irréductible et apériodique. Une première possibilité est de refaire tout le travail en partant depuis la nouvelle distribution uniforme. En fait, il y a bien mieux : on peut repartir d'une distribution quelconque, et en particulier d'une adaptée de celle obtenue précédemment comme distribution limite. En effet, le théorème 4.2.2 assure que la distribution stationnaire est la limite de toute dynamique dans la chaîne de Markov, en particulier partant de n'importe quelle distribution initiale. En pratique, cela permet d'atteindre beaucoup plus rapidement la distribution limite.

4.5 Algorithme de Metropolis et application à la cryptographie

De nombreux problèmes sont désormais résolus à l'aide de simulation probabiliste, en marchant aléatoirement dans une chaîne de Markov bien choisie. Un exemple intéressant [1] provient d'un cours de Stanford, où des étudiants ont appliqué cette technique pour décoder des messages que s'échangent les détenus d'une prison, tels que celui présenté en FIGURE 4.1. En supposant que le code est un simple chiffrement par substitution, c'est-à-dire que chaque symbole remplace une lettre, un chiffre, un espace ou un signe de ponctuation, il s'agit donc de trouver une fonction de substitution

$$f\colon \{\text{symboles du code}\} \longrightarrow \{\text{alphabet usuel}\}$$

Une approche standard pour décrypter un tel message consiste à utiliser les statistiques de la langue (ici supposée être l'anglais) du message, pour deviner le choix le plus probable pour f et regarder si le message décrypté fait sens. Malheureusement, simplement remplacer les lettres les plus fréquentes du message par les lettres les plus fréquentes en anglais échoue dans cet exemple.

1. Exemple tiré de *The Markov chain Monte Carlo revolution* par Persi Diaconis, dans Journal : Bull. Amer. Math. Soc. 46 (2009), pages 179-205.

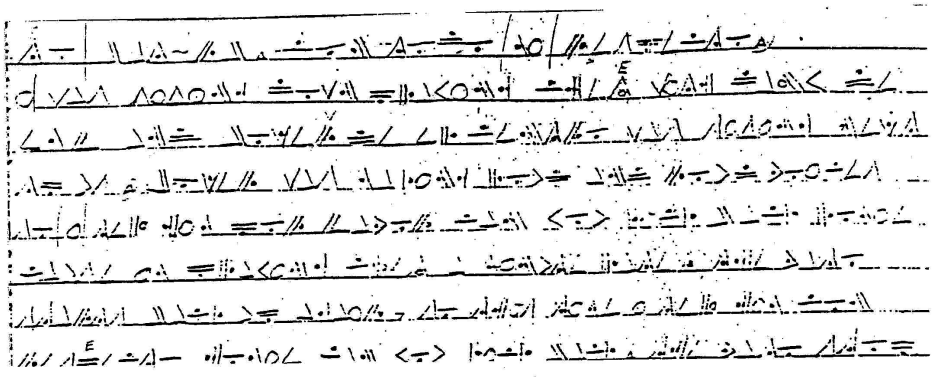


FIGURE 4.1 – Un message codé

```
ENTER HAMLET HAM TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIS
NOBLER IN THE MIND TO SUFFER THE SLINGS AND ARROWS OF OUTRAGEOUS
FORTUNE OR TO TAKE ARMS AGAINST A SEA OF TROUBLES AND BY OPPOSING END
```

FIGURE 4.2 – Un passage de *Hamlet*

On utilise donc des statistiques plus détaillées. Pour obtenir ces statistiques, on utilise un corpus, c'est-à-dire un texte suffisamment long de la langue choisie, et on enregistre les transitions du premier ordre, c'est-à-dire pour chaque pair de symbole (x, y) la proportion où ces deux symboles se sont retrouvés consécutifs dans le corpus. Cela donne une matrice M de transitions. On peut alors associer une mesure de plausibilité à f via

$$\mathrm{Pl}(f) = \prod_i M(f(s_i), f(s_{i+1}))$$

où (s_i) est la suite de symboles du message codé. Les fonctions f ayant une plausibilité haute sont des bonnes candidates pour le décryptage. Pour trouver les fonctions f maximisant la plausibilité, on peut utiliser un algorithme de Metropolis, qui consiste en une marche aléatoire accélérée dans une chaîne de Markov :

- On commence avec un choix préliminaire de fonction f.
- On calcule $\mathrm{Pl}(f)$.
- On change f en f_* en réalisant une transposition aléatoire à partir de f en inversant deux symboles.
- On calcule $\mathrm{Pl}(f_*)$; s'il est strictement plus grand que $\mathrm{Pl}(f)$, on accepte f_*.
- Sinon, on lance une pièce biaisée ayant probabilité $\mathrm{Pl}(f_*)/\mathrm{Pl}(f)$ de succès ; en cas de succès, on accepte f_*, sinon, on reste avec f.

L'algorithme continue ainsi en essayant d'améliorer la fonction f courante à l'aide de transpositions aléatoires. Le lancer de pièce permet de retomber vers une fonction moins plausible, empêchant ainsi l'algorithme de rester coincé dans un *maximum local* de la fonction de plausibilité.

Évidemment, l'espace des fonctions f possibles est gigantesque (disons de l'ordre de $40! \approx 8 \times 10^{47}$ environ). Pourquoi l'algorithme de Metropolis devrait trouver la bonne réponse ? Avant de lancer l'algorithme sur le message du prisonnier, faisons un test avec un passage de *Hamlet*, donné en FIGURE 4.2.

Pour tester l'algorithme, on commence par mélanger les symboles de l'alphabet, puis on lance l'algorithme de Metropolis. La FIGURE 4.3 montre une exécution, l'entier à gauche étant le numéro d'itérations. Après 100 itérations, on ne comprend rien, mais après 2000 itérations, le message décrypté fait sens. Il reste essentiellement identique si on continue l'exécution.

On peut donc faire tourner l'algorithme sur le message du prisonnier. Une portion du résultat final est donné en FIGURE 4.4. Le décodage semble avoir fonctionné : notez en particulier le fait que

```
  100 ER ENOHDLAE OHDLO UOZEOUNORU O UOZEO HD OITO HEOQSET IUROFHE HENO ITORUZAEN
  200 ES ELOHRNDE OHRNO UOVEOULOSU O UOVEO HR OITO HEOQAET IUSOPHE HELO ITOSUVDEL
  300 ES ELOHANDE OHANO UOVEOULOSU O UOVEO HA OITO HEOQRET IUSOFHE HELO ITOSUVDEL
  400 ES ELOHINME OHINO UOVEOULOSU O UOVEO HI OATO HEOQRET AUSOWHE HELO ATOSUVMEL
  500 ES ELOHINME OHINO UODEOULOSU O UODEO HI OATO HEOQRET AUSOWHE HELO ATOSUDMEL
  600 ES ELOHINME OHINO UODEOULOSU O UODEO HI OATO HEOQRET AUSOWHE HELO ATOSUDMEL
  900 ES ELOHANME OHANO UODEOULOSU O UODEO HA OITO HEOQRET IUSOWHE HELO ITOSUDMEL
 1000 IS ILOHANMI OHANO RODIORLOSR O RODIO HA OETO HIOQUIT ERSOWHI HILO ETOSRDMIL
 1100 ISTILOHANMITOHANOT ODIO LOS TOT ODIOTHATOEROTHIOQUIRTE SOWHITHILOTEROS DMIL
 1200 ISTILOHANMITOHANOT ODIO LOS TOT ODIOTHATOEROTHIOQUIRTE SOWHITHILOTEROS DMIL
 1300 ISTILOHARMITOHAROT ODIO LOS TOT ODIOTHATOENOTHIOQUINTE SOWHITHILOTENOS DMIL
 1400 ISTILOHAMRITOHAMOT OFIO LOS TOT OFIOTHATOENOTHIOQUINTE SOWHITHILOTENOS FRIL
 1600 ESTEL HAMRET HAM TO CE OL SOT TO CE THAT IN THE QUENTIOS WHETHEL TIN SOCREL
 1700 ESTEL HAMRET HAM TO BE OL SOT TO BE THAT IN THE QUENTIOS WHETHEL TIN SOBREL
 1800 ESTER HAMLET HAM TO BE OR SOT TO BE THAT IN THE QUENTIOS WHETHER TIN SOBLER
 1900 ENTER HAMLET HAM TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIS NOBLER
 2000 ENTER HAMLET HAM TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIS NOBLER
```

FIGURE 4.3 – Exécution de l'algorithme de Metropolis sur le passage de *Hamlet*

```
  to bat-rb. con todo mi respeto. i was sitting down playing chess with
danny de emf and boxer de el centro was sitting next to us. boxer was
making loud and loud voices so i tell him por favor can you kick back
homie cause im playing chess a minute later the vato starts back up again
so this time i tell him con respecto homie can you kick back.  the vato
stop for a minute and he starts up again so i tell him check this out shut
the f**k up cause im tired of your voice and if you got a problem with it
we can go to celda and handle it. i really felt disrespected thats why i
told him. anyways after i tell him that the next thing I know that vato
slashes me and leaves. dy the time i figure im hit i try to get away but
the c.o. is walking in my direction and he gets me right dy a celda. so i
go to the hole. when im in the hole my home boys hit doxer so now "b" is
also in the hole. while im in the hole im getting schoold wrong and
```

FIGURE 4.4 – Le message décodé grâce à l'algorithme de Metropolis

l'algorithme a fonctionné alors même que la langue utilisée est un mélange d'anglais non littéraire, d'espagnol et de jargon prisonnier.

Essayons désormais de comprendre l'algorithme utilisé. Tout d'abord, remarquons que la fonction de plausibilité, une fois normalisée, donne une distribution de probabilité sur l'ensemble fini de fonctions de décodage f possibles. Plus précisément, si on pose

$$z = \sum_f \mathrm{Pl}(f)$$

alors la fonction π définie pour toute fonction de décodage f par

$$\pi(f) = z^{-1}\mathrm{Pl}(f)$$

est une distribution de probabilité sur les fonctions de décodage. Noter qu'on ne peut pas calculer z en pratique. Le problème qu'on cherche à résoudre consiste donc à tirer de manière répétée une fonction f suivant la distribution de probabilité π. Comment faire ? L'algorithme de Metropolis propose une solution.

Supposons donc qu'on ait un ensemble S d'états et une distribution de probabilité π sur S qu'on souhaite simuler (et qu'on ne connait possiblement qu'à une constante de normalisation près) vérifiant $\pi_i > 0$ pour tout état i. On considère une chaîne de Markov quelconque sur S, avec une matrice de probabilité de transitions Q, vérifiant $Q_{i,j} > 0$ si et seulement si $Q_{j,i} > 0$. Cette

chaîne n'est donc pas reliée à la distribution π. L'algorithme de Metropolis consiste à modifier Q en une nouvelle chaîne de Markov régie par une matrice de probabilité de transitions P ayant pour distribution stationnaire π. Elle utilise un *lancer de pièce* pour prendre une décision : on le simule à l'aide d'un ratio $A_{i,j} = \pi_j Q_{j,i}/\pi_i Q_{i,j}$, défini dès lors que $Q_{i,j} \neq 0$. La chaîne P est définie par

$$P_{i,j} = \begin{cases} Q_{i,j} & \text{si } i \neq j, A_{i,j} \geq 1 \\ Q_{i,j} A_{i,j} & \text{si } i \neq j, A_{i,j} < 1 \\ Q_{i,j} + \displaystyle\sum_{k | A_{i,k} < 1} Q_{i,k}(1 - A(i,k)) & \text{si } i = j \end{cases}$$

L'interprétation de cette nouvelle chaîne est la suivante : depuis l'état i, on choisit j avec probabilité $Q_{i,j}$; si $A_{i,j} \geq 1$, on saute en j ; sinon, on lance une pièce avec probabilité $A_{i,j}$ de succès et on saute en y seulement en cas de succès. Notons que la constante de normalisation de π disparait dans le calcul du ratio $A_{i,j}$: il est donc inutile de la connaître.

La nouvelle chaîne de Markov satisfait $\pi_i P_{i,j} = \pi_j P_{j,i}$, par symétrie du ratio $A_{i,j}$. Ainsi, on a

$$(\pi P)_j = \sum_i \pi_i P_{i,j} = \sum_i \pi_j P_{j,i} = \pi_j \sum_i P_{i,j} = \pi_j$$

puisque les lignes de P somment à 1 (comme toute matrice de probabilité de transitions). La distribution π est donc une distribution stationnaire de la nouvelle chaîne de Markov. Si la chaîne est bien irréductible et apériodique, alors, par les théorèmes 4.2.1 et 4.2.2, cette distribution stationnaire est unique et toute simulation de la chaîne converge vers cette distribution stationnaire.

Pour l'exemple du message du prisonnier, l'ensemble d'états est celui des fonctions injectives de l'espace des symboles (de taille m) à l'espace de l'alphabet anglais (de taille $n \geq m$). L'ensemble d'états est donc de taille $n(n-1)\cdots(n-m+1)$. La chaîne de Markov initiale choisie est spécifiée par les transpositions de symboles : pour f, f^*, on a donc

$$Q_{f,f^*} = \begin{cases} \frac{1}{n(n-1)(n-m+1)(n-m+2)} & \text{si } f, f^* \text{ diffèrent en au plus deux symboles} \\ 0 & \text{sinon} \end{cases}$$

Notons que $Q_{f,f^*} = Q_{f^*,f}$ de sorte que le ratio A_{f,f^*} vaut ici $\pi_{f^*}/\pi_f = \mathrm{Pl}(f^*)/\mathrm{Pl}(f)$, comme dans l'algorithme.