ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

PROJET DE MATHÉMATIQUES

BACHELOR SEMESTRE 5

Méthode probabiliste et Lemme local de Lovász

Auteur :
Samuel REGAMEY

Assistante responsable :
Ghid MAATOUK

Professeur :
Amin SHOKROLLAHI

Janvier 2012

Table des matières

Résumé

La méthode probabiliste est une méthode non-constructive permettant de prouver l'existence d'objets mathématiques. Ce travail s'articule autour du lemme local de Lovász, qui est un outil puissant lorsque nous étudions des événements ayant "suffisamment peu" de dépendance entre eux. Nous étudierons notamment le problème du routage, dont une solution importante utilise ce lemme. Enfin, nous étudierons une preuve constructive de ce lemme.

1 Introduction

La méthode probabiliste est une méthode permettant de prouver de manière non constructive l'existence d'objets mathématiques avec des propriétés données. Introduite par Paul Erdős, elle consiste à construire aléatoirement un objet, et de montrer qu'il y a une probabilité strictement positive pour que l'objet ainsi obtenu ait la propriété souhaitée. On aura ainsi prouvé l'existence d'un tel objet.

Après avoir illustré cette méthode par quelques exemples simples, nous aborderons le lemme local de Lovász. Ce lemme permet de montrer que si on a un ensemble fini d'événements, avec pour chacun d'eux une probabilité "suffisamment petite" et avec "suffisamment peu de dépendances entre eux", alors la probabilité qu'aucun de ces événements n'ait lieu est strictement positive. Nous donnerons quelques exemples assez simples d'applications de ce lemme, avant d'aborder un problème plus conséquent, le problème du routage. Dans un réseau donné, on dispose de paquets que l'on doit faire circuler via des chemins donnés. Le but est de trouver un schedule pour ces paquets qui ait une durée d'exécution assez petite. Enfin, nous terminerons en étudiant une preuve constructive du lemme local.

2 Quelques exemples illustrant la méthode probabiliste

Pour mieux illustrer la méthode probabiliste générale, nous allons commencer par quelques exemples simples d'applications. Ils sont tirés du livre The Probabilistic Method [1]. Le premier nous fournira un résultat concernant la 2-colorabilité d'hypergraphes. Le second nous donnera une borne inférieure aux nombres de Ramsey.

Définitions 2.1. Un *hypergaphe* est une paire $H = (V, E)$ où V est un ensemble fini dont les éléments sont les *sommets* de l'hypergraphe, et où E est un sous-ensemble fini de $\mathcal{P}(V)$, dont les éléments sont appelés les *arêtes*. Si chaque arête de H est un sous-ensemble de V de cardinalité exactement n, l'hypergraphe est dit *n-uniforme*. Un *2-coloriage* d'un hypergraphe $H = (V, E)$ est une application $c : V \to \{0, 1\}$, qui à chaque sommet v associe une couleur. Un hypergraphe H est dit *2-coloriable* si existe un 2-coloriage c de H tel qu'aucune arête n'est monochromatique (c'est-à-dire, $\forall e \in E$, $\exists u, v \in e$ tels que $c(u) \neq c(v)$). On définit $m(n)$ comme le plus petit entier k tel qu'il existe un hypergraphe n-uniforme avec k arêtes qui ne soit pas 2-coloriable.

Proposition 2.2. $m(n) \geq 2^{n-1}$.

Démonstration. Soit $H = (V, E)$ un hypergraphe n-uniforme avec strictement moins que 2^{n-1} arêtes. Colorions alors aléatoirement, de manière équiprobable et indépendante, chacun des sommets de V par 2 couleurs (rouge et bleu). Pour chaque arête e, soit A_e l'événement "e est monochromatique".

On a $\mathbb{P}(A_e) = 2^{1-n}$, car la probabilité que chacun des sommets de e soit bleu est de $\frac{1}{2}$, et comme les couleurs des sommets sont indépendantes, on a donc que la probabilité que tous les sommets soient bleus vaut $\frac{1}{2^n}$. De même avec la couleur rouge. Comme les événements B_e "tous les sommets de e sont bleus" et R_e "tous les sommets de e sont rouges" sont disjoints, on a que $\mathbb{P}(A_e) = \mathbb{P}(B_e \cup R_e) = \mathbb{P}(B_e) + \mathbb{P}(R_e) = \frac{1}{2^n} + \frac{1}{2^n} = 2^{1-n}$.

Ainsi

$$\mathbb{P}\left(\bigcup_{e\in E} A_e\right) \leq \sum_{e\in E} \mathbb{P}(A_e) = |E| \cdot 2^{1-n} < 2^{n-1} \cdot 2^{1-n} = 1.$$

Et donc

$$\mathbb{P}\left(\bigcap_{e\in E} \overline{A_e}\right) = \mathbb{P}\left(\overline{\bigcup_{e\in E} A_e}\right) = 1 - \mathbb{P}\left(\bigcup_{e\in E} A_e\right) > 0.$$

La méthode probabiliste nous montre donc qu'il existe un 2-coloriage de H tel qu'aucune arête ne soit monochromatique, donc H est 2-coloriable. Ceci permet de conclure que $m(n) \geq 2^{n-1}$. □

Définition 2.3. Le nombre de Ramsey $R(k, l)$ est le plus petit entier n tel que, pour tout 2-coloriage en rouge et bleu des arêtes du graphe complet sur n sommets K_n ; on ait ou bien un sous-graphe complet K_k induit par K_n dont toutes les arêtes sont rouges, ou bien un sous-graphe complet K_l induit par K_n dont toutes les arêtes sont bleues.

Proposition 2.4. *Si* $\binom{n}{k} \cdot 2^{1-\binom{k}{2}} < 1$, *alors* $R(k, k) > n$.

Ce résultat permet de calculer que $R(k, k) > \lfloor 2^{k/2} \rfloor$ pour tout $k \geq 3$.

Démonstration. Soit un coloriage aléatoire des arêtes du graphe complet K_n, où chaque arête est coloriée indépendamment en rouge ou en bleu de manière équiprobable. Pour W un ensemble d'exactement k sommets de K_n, soit A_W l'événement "toutes les arêtes de K_W sont de la même couleur", où K_W est le sous-graphe complet sur W induit par K_n muni du 2-coloriage correspondant.

Comme le graphe K_W contient $\binom{k}{2}$ arêtes, de manière analogue à la preuve de la proposition 2.2, on obtient que $\mathbb{P}(A_W) = \frac{2}{2^{\binom{k}{2}}} = 2^{1-\binom{k}{2}}$.

Observons maintenant qu'il y a $\binom{n}{k}$ choix possibles d'ensembles W. Ainsi, on a que la probabilité $\mathbb{P}(A)$ qu'au moins un événement A_W ait lieu est au plus $\binom{n}{k} \cdot 2^{1-\binom{k}{2}} < 1$. Donc à l'inverse, la probabilité $\mathbb{P}(\overline{A})$ qu'aucun événement A_W n'ait lieu vaut $1 -$

$\mathbb{P}(A) > 0$. Ainsi, on vient de montrer, à l'aide de la méthode probabiliste, qu'il existe un 2-coloriage de K_n, tel qu'il n'admette aucun sous-graphe induit de cardinalité k monochromatique. Ceci montre donc que $R(k,k) > n$. □

3 Le Lemme local de Lovász

Nous allons maintenant montrer le résultat principal autour duquel s'articule notre travail. Le lemme local de Lovász s'applique quand on considère des événements avec des probabilités "suffisamment petites", et avec "suffisamment peu" de dépendances. Il est très utile pour montrer l'existence d'objets mathématiques. Nous nous inspirons ici de la preuve du livre The Probabilistic Method [2].

Lemme 3.1. Lemme local de Lovász. *Soient $A_1, A_2, A_3, ..., A_n$ des événements dans un espace de probabilité arbitraire. Un graphe orienté (ou digraphe) $D = (V, E)$ avec $V = \{1, 2, 3, ..., n\}$ est appelé un graphe de dépendance pour les événements $A_1, A_2, ..., A_n$ si pour tout entier i avec $1 \leq i \leq n$, on a que A_i est indépendant de la collection d'événements $\{A_j : (i,j) \notin E\}$ (c'est-àdire que $\forall T \subset \{1, 2, ..., n\}$ tel que $\forall t \in T, (i,t) \notin E$; on a : $\mathbb{P}\left(A_i \left| \bigwedge_{t \in T} \overline{A_t} \right.\right) = \mathbb{P}(A_i)$).*

S'il existe $D = (V, E)$ un graphe de dépendance et $x_1, x_2, ..., x_n$ des réels positifs strictement inférieurs à 1 tels que pour tout $1 \leq i \leq n$:

$$\mathbb{P}(A_i) \leq x_i \cdot \prod_{(i,j) \in E} (1 - x_j) \ ; \tag{1}$$

alors

$$\mathbb{P}\left(\bigwedge_{i=1}^{n} \overline{A_i}\right) \geq \prod_{i=1}^{n} (1 - x_i) > 0.$$

Démonstration. Soit H_s l'hypothèse suivante :

$$\mathbb{P}\left(A_i \left| \bigwedge_{j \in S} \overline{A_j} \right.\right) \leq x_i,$$

et montrons par récurrence sur s que H_s est vrai pour tout $S \subset \{1, 2, ..., n\}$ avec $|S| = s$ et $s < n$.

Pour $s = 0$, H_s est facilement vérifiée : comme par hypothèse on a $0 < x_j < 1 \quad \forall j$, on a que $\prod_{(i,j) \in E} (1 - x_j) < 1$, et donc (1) implique directement $\mathbb{P}(A_i) \leq x_i$, ce qui prouve H_0.

Supposons maintenant que $H_{s'}$ soit vrai $\forall s' < s$, et soit S avec $|S| = s$. Posons $S_1 = \{j \in S : (i,j) \in E\}$ et $S_2 = S \backslash S_1$. Observons maintenant que (par définition de

la probabilité conditionnelle) :

$$\mathbb{P}\left(A_i \middle| \bigwedge_{j\in S} \overline{A_j}\right) = \frac{\mathbb{P}\left(A_i \wedge \left(\bigwedge_{j\in S_1} \overline{A_j}\right) | \bigwedge_{l\in S_2} \overline{A_l}\right)}{\mathbb{P}\left(\bigwedge_{j\in S_1} \overline{A_j} | \bigwedge_{l\in S_2} \overline{A_l}\right)}. \tag{2}$$

Pour majorer la fraction (2), nous allons en majorer le numérateur et minorer le dénominateur.

Concernant le numérateur, comme on a par hypothèse que A_i est indépendant de la collection $\{A_j : (i,j) \notin E\} \supset \{A_l : l \in S_2\}$, on a $\mathbb{P}\left(A_i \middle| \bigwedge_{l\in S_2} \overline{A_l}\right) = \mathbb{P}(A_i)$, ce qui nous donne :

$$\mathbb{P}\left(A_i \wedge \left(\bigwedge_{j\in S_1} \overline{A_j}\right) \middle| \bigwedge_{l\in S_2} \overline{A_l}\right) \leq \mathbb{P}\left(A_i \middle| \bigwedge_{l\in S_2} \overline{A_l}\right) = \mathbb{P}(A_i) \leq x_i \cdot \prod_{(i,j)\in E} (1-x_j).$$

Concernant le dénominateur, nous allons utiliser l'hypothèse de récurrence. Posons $S_1 = \{j_1, j_2, ..., j_r\}$. Si $r = 0$, c'est-à-dire $S_1 = \emptyset$, le dénominateur vaut 1, et H_s est clairement vraie. Si $r > 0$, on se retrouve avec :

$$\mathbb{P}\left(\bigwedge_{j\in S_1} \overline{A_j} \middle| \bigwedge_{l\in S_2} \overline{A_l}\right) = \mathbb{P}\left(\overline{A_{j_1}} \wedge \overline{A_{j_2}} \wedge \ldots \wedge \overline{A_{j_r}} \middle| \bigwedge_{l\in S_2} \overline{A_l}\right)$$

$$= \left(\mathbb{P}(\overline{A_{j_1}}) \middle| \bigwedge_{l\in S_2} \overline{A_l}\right) \cdot \left(\mathbb{P}(\overline{A_{j_2}}) \middle| \overline{A_{j_1}} \wedge \bigwedge_{l\in S_2} \overline{A_l}\right) \cdot \ldots \cdot \left(\mathbb{P}(\overline{A_{j_r}}) \middle| \overline{A_{j_1}} \wedge \overline{A_{j_2}} \wedge \ldots \wedge \overline{A_{j_{r-1}}} \wedge \bigwedge_{l\in S_2} \overline{A_l}\right)$$

$$= \left(1 - \mathbb{P}\left(A_{j_1} \middle| \bigwedge_{l\in S_2} \overline{A_l}\right)\right) \cdot \left(1 - \mathbb{P}\left(A_{j_2} \middle| \overline{A_{j_1}} \wedge \bigwedge_{l\in S_2} \overline{A_l}\right)\right) \cdot \ldots \cdot$$

$$\cdot \left(1 - \mathbb{P}\left(A_{j_r} \middle| \overline{A_{j_1}} \wedge \overline{A_{j_2}} \wedge \ldots \wedge \overline{A_{j_{r-1}}} \wedge \bigwedge_{l\in S_2} \overline{A_l}\right)\right).$$

Notons que $\forall k$:

$$\left(1 - \mathbb{P}\left(A_{j_k} \middle| \overline{A_{j_1}} \wedge \overline{A_{j_2}} \wedge \ldots \wedge \overline{A_{j_{k-1}}} \wedge \bigwedge_{l\in S_2} \overline{A_l}\right)\right) = \left(1 - \mathbb{P}\left(A_{j_k} \middle| \bigwedge_{l\in S'_k} \overline{A_l}\right)\right),$$

avec $S'_k = S_2 \cup \{j_1, j_2, ..., j_{k-1}\}$, et donc $|S'_k| < s$. Ainsi, par l'hypothèse de récurrence, $\mathbb{P}\left(A_{j_k} \middle| \bigwedge_{l\in S'_k} \overline{A_l}\right) \leq x_{j_k}$ et donc $\left(1 - \mathbb{P}\left(A_{j_k} \middle| \bigwedge_{l\in S'_k} \overline{A_l}\right)\right) \geq (1 - x_{j_k})$. Il s'ensuit au final que :

$$\mathbb{P}\left(\bigwedge_{j\in S_1} \overline{A_j} \middle| \bigwedge_{l\in S_2} \overline{A_l}\right) \geq \prod_{k=1}^{r} (1 - x_{j_k}) = \prod_{(i,j)\in E} (1 - x_j).$$

Avec ces bornes sur le numérateur et le dénominateur, il devient clair que l'hypothèse H_s est vraie, et donc par récurrence $\mathbb{P}\left(A_i \middle| \bigwedge_{j\in S} \overline{A_j}\right) \leq x_i$, $\forall S \subset \{1, 2, ..., n\}$.

Désormais, on peut conclure :

$$\mathbb{P}\left(\bigwedge_{i=1}^{n}\overline{A_i}\right) = \mathbb{P}(\overline{A_1})\cdot\mathbb{P}(\overline{A_2}\,|\,\overline{A_1})\cdot\ldots\cdot\mathbb{P}\left(\overline{A_n}\,\middle|\,\bigwedge_{i=1}^{n-1}\overline{A_i}\right)$$

$$= (1-\mathbb{P}(A_1))\cdot\left(1-\mathbb{P}\left(A_2\,|\,\overline{A_1}\right)\right)\cdot\ldots\cdot\left(1-\mathbb{P}\left(A_n\,\middle|\,\bigwedge_{i=1}^{n-1}\overline{A_i}\right)\right) \geq \prod_{i=1}^{n}(1-x_i).$$

avec l'inégalité qui est tirée du résultat montré par récurrence, et qui nous amène au résultat souhaité. □

Ce lemme est néanmoins assez peu commode à utiliser, au vu de la forme contraignante des hypothèses dont on a besoin. C'est pourquoi on utilise généralement son corollaire, qui correspond à un cas symétrique de ce lemme et qui est en général plus facile à utiliser. Nous nous inspirons ici du livre The Probabilistic Method [3].

Corollaire 3.2. Cas symétrique du lemme local de Lovász. *Soient $A_1, A_2, ..., A_n$ des événements d'un espace de probabilité arbitraire. Si A_i est mutuellement indépendant d'une collection qui comprend tous les événements A_j sauf au plus d d'entre eux, et qu'il existe p tel que $\mathbb{P}(A_i) \leq p \quad \forall\, 1 \leq i \leq n$ et*

$$e \cdot p \cdot (d+1) \leq 1 \quad ; \tag{3}$$

Alors

$$\mathbb{P}\left(\bigwedge_{i=1}^{n}\overline{A_i}\right) > 0.$$

Démonstration. Si $d = 0$, les événements sont mutuellement indépendants et on a donc clairement $\mathbb{P}\left(\bigwedge_{i=1}^{n}\overline{A_i}\right) = \prod_{i=1}^{n}\mathbb{P}(\overline{A_i}) \geq (1-p)^n > 0$.

Sinon, on utilise le cas général du lemme local 3.1 : on pose $x_i = \frac{1}{d+1} \quad \forall i$. On considère un graphe de dépendance $D = (V, E)$ tel que $\forall i,\ |\{j : (i,j) \in E\}| \leq d$ (Il existe un tel graphe de dépendance par hypothèse). On a ainsi :

$$x_i \cdot \prod_{(i,j)\in E}(1-x_j) \geq \frac{1}{d+1}\cdot\left(1-\frac{1}{d+1}\right)^d.$$

En observant que (3) implique que $p \leq \frac{1}{e\cdot(d+1)}$ et en supposant que pour tout $d \geq 1$, $\left(1-\frac{1}{d+1}\right)^d > \frac{1}{e}$ (on montrera ce résultat plus loin), on se retrouve avec, $\forall i$:

$$x_i \cdot \prod_{(i,j)\in E}(1-x_j) \geq \frac{1}{d+1}\cdot\left(1-\frac{1}{d+1}\right)^d \geq \frac{1}{e\cdot(d+1)} \geq p \geq \mathbb{P}(A_i).$$

De fait, nous entrons bien dans les hypothèses du lemme 3.1, qui permet de conclure que $\mathbb{P}\left(\bigwedge_{i=1}^{n}\overline{A_i}\right) > 0$.

Il ne nous reste donc plus qu'à montrer que pour tout $d \geq 1$, $\left(1 - \frac{1}{d+1}\right)^d > \frac{1}{e}$: Comme ln est concave, on a que $\forall t$, $\ln(1 + \frac{1}{t}) < \frac{1}{t}$). (Car la tangente de ln en 1 est d'équation $y = x - 1$, et que la concavité implique que ln est toujours sous sa tangente, donc : $\ln(x) \leq x - 1 \Rightarrow \ln\left(1 + \frac{1}{t}\right) \leq \frac{1}{t} \Rightarrow t \cdot \ln\left(1 + \frac{1}{t}\right) \leq 1$.)

Ainsi, on a que

$$\left(1 + \frac{1}{x}\right)^x = e^{x \cdot \ln(1 + \frac{1}{x})} \leq e. \tag{4}$$

Il ne nous reste plus à observer que :

$$\left(\left(1 - \frac{1}{x+1}\right)^x\right)^{-1} = \left(\frac{x}{x+1}\right)^{-x} = \left(\left(\frac{x}{x+1}\right)^{-1}\right)^x = \left(\frac{x+1}{x}\right)^x = \left(1 + \frac{1}{x}\right)^x \leq e.$$

D'où, pour $x > 0$: $\left(1 - \frac{1}{x+1}\right)^x \geq e^{-1}$.

□

4 Quelques exemples simples d'applications

Nous allons maintenant commencer par montrer deux exemples simples d'applications du lemme, qui sont tirés du livre The Probabilistic Method [4]. Le premier concerne la 2-colorabilité d'hypergraphes, tandis que le second permet d'améliorer la borne sur les nombres de Ramsey de la forme $R(k, k)$ que nous avions obtenue à la proposition 2.4.

Proposition 4.1. *Soit $H = (V, E)$ un hypergraphe dans lequel chaque arête contient au moins k sommets, et supposons que chaque arête de H partage des sommets avec au plus d autres arêtes. Si $e \cdot (d + 1) \leq 2^{k-1}$, alors H est 2-coloriable.*

Démonstration. Colorions de manière aléatoire, indépendante et équiprobable chaque arête de H en rouge ou bleu. Pour chaque arête f, soit A_f l'événement "f est monochromatique". De manière analogue à la preuve de la proposition 2.2, comme chaque arête possède au moins k sommets, on a $\mathbb{P}(A_f) \leq 2^{1-k}$. De plus, l'événement A_f est clairement mutuellement indépendant de tous les événements $A_{f'}$ tels que f' n'est pas l'une des au plus d arêtes qui partagent des sommets avec f. Ainsi, comme on a : $e \cdot (d + 1) \leq 2^{k-1} \Rightarrow e \cdot (d + 1) \cdot 2^{1-k} \leq 1$; on peut appliquer le corollaire 3.2 du cas symétrique du lemme local, qui nous garantit l'existence d'un coloriage de H tel qu'aucun A_f n'ait lieu, et donc que H est bien 2-coloriable. □

Proposition 4.2. *Si $e \cdot \left(\binom{k}{2} \cdot \binom{n}{k-2} + 1\right) \cdot 2^{1-\binom{k}{2}} < 1$, alors $R(k, k) > n$.*

Ce résultat permet de calculer que :

$$R(k, k) > \frac{\sqrt{2}}{e} \cdot (1 + o(1)) \cdot k \cdot 2^{k/2}$$

Démonstration. On procède de la même manière que pour la preuve de la proposition 2.4, avec un 2-coloriage aléatoire des arêtes du graphe complet $K_n = (V, E)$. Pour tout ensemble S de k sommets distincts de V, on pose A_S l'événement "le sous-graphe complet K_S de K_n sur S est monochromatique". De la même manière qu'en 2.4, $\mathbb{P}(A_S) = 2^{1-\binom{k}{2}}$. Observons que A_S est mutuellement indépendant de $\{A_T : |S \cap T| < 2\}$ (en effet, comme S et T ont au plus un sommet commun, K_S et K_T n'ont pas d'arêtes communes). Or $|\{A_T : |S \cap T| \geq 2\}| \leq \binom{k}{2} \cdot \binom{n}{k-2}$. (Il y a $\binom{k}{2}$ choix possibles pour deux sommets communs à S et T, et $\binom{n-2}{k-2}$ choix possibles pour les $k-2$ autres sommets de T.) Ainsi, les hypothèses nous permettent d'appliquer le corollaire 3.2 avec $d = \binom{k}{2} \cdot \binom{n}{k-2}$ et $p = 2^{1-\binom{k}{2}}$, ce qui nous garantit l'existence d'un coloriage tel qu'aucun événement A_S n'ait lieu, donc tel qu'aucun sous-graphe sur k sommets ne soit monochromatique. Ainsi, $R(k,k) > n$. □

Proposition 4.3. *Soit $G = (V, E)$ un graphe simple et supposons qu'à chaque $v \in V$ est associé un ensemble $S(v)$ de couleurs, de cardinalité au moins $10d$, où $d \geq 1$. Supposons de plus que pour chaque $v \in V$ et $c \in S(v)$, il y a au plus d voisins u de v tels que $c \in S(u)$. Alors il existe un coloriage de G qui associe à chaque sommet v une couleur de son ensemble associé $S(v)$.*

Cette proposition est un exercice du livre The Probabilistic Method [5] (donné sans résolution). Voici ma proposition de solution :

Démonstration. Nous allons le démontrer pour le cas où $S(v)$ est de cardinalité exactement $10d$ pour tout $v \in V$. En effet, pour le cas général, nous pouvons "tronquer" arbitrairement les ensembles de couleurs de chaque sommet pour retomber sur ce cas particulier, et la solution trouvée sera également une solution du cas général.

Soit $C : V \to \bigcup_{v \in V} S(v)$ un coloriage de V qui associe à chaque sommet $v \in V$ une couleur de $S(v)$ aléatoirement avec équiprobabilité pour chaque couleur.

Considérons pour chaque arête $i = (u, v) \in E$ et chaque $c \in S(u) \cap S(v)$; l'événement A_{ic} : "$C(v) = C(u) = c$".

Comme $S(v)$ et $S(u)$ sont de cardinalité $10d$, on a que $P(A_{ic}) \leq 1/100d^2$.

Observons que l'événement A_{ic} est mutuellement indépendant de l'ensemble de tous les événements privé des événements A_{jc_j} , A_{lc_l} et $A_{ic'}$ avec $j = (u, w) \in E$ tel que $c_j \in S(w) \cap S(u)$; $l = (v, w') \in E$ tel que $c_l \in S(w') \cap S(v)$; $c' \in S(u) \cap S(v)$. En effet, pour tous les autres événements, ou bien ils concernent des arêtes disjointes de l'arête i ; ou bien ils ont une probabilité nulle et n'ont donc aucune influence.

On a au plus $10d^2$ événements A_{jc_j}. En effet, pour chacun des d voisins w de u, on a au plus $10d$ couleurs c_j dans $S(w) \cap S(u)$. De même pour A_{lc_l}. Enfin, on a au plus $10d - 1$ événements $A_{ic'}$ avec $c \neq c'$, correspondant à la cardinalité de $S(u) \cap S(v)$ qui est bornée par la cardinalité $10d$ de $S(u)$.

Observons maintenant que :

$$e \cdot \frac{1}{100 \cdot d^2} \cdot (2 \cdot 10 \cdot d^2 + 10 \cdot d) =$$

$$e \cdot \left(\frac{20 \cdot d^2}{100 \cdot d^2} + \frac{10 \cdot d}{100 \cdot d^2} \right) =$$

$$e \cdot \left(\frac{1}{5} + \frac{1}{10d} \right) \leq e \cdot \left(\frac{1}{5} + \frac{1}{10} \right) = e \cdot \frac{3}{10} < 1$$

Nous sommes donc en mesure d'appliquer le corollaire pour le cas symétrique du lemme local de Lovász, qui nous garantit donc que : $P[\bigwedge_{ic=1}^{n} \overline{A_{ic}}] > 0$, où $n = 10d \cdot m$ avec $m = |E|$. □

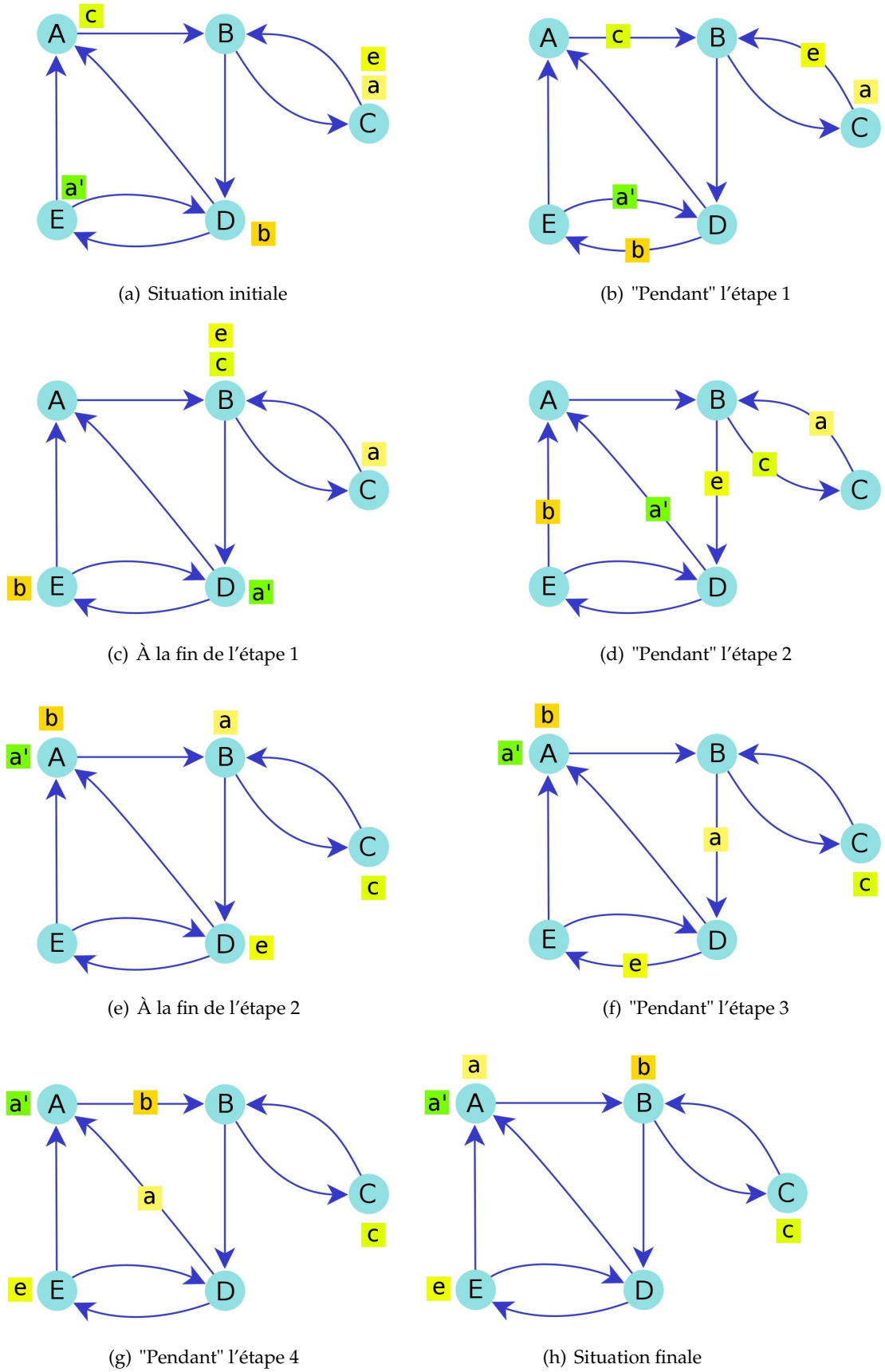
5 Le problème de routage

5.1 Présentation du problème et algorithme naïf

On considère un réseau, c'est à dire un graphe $G = (V, E)$, ainsi que n paquets assortis chacun d'un chemin dans G. On doit acheminer chacun des paquet de son sommet initial à son sommet de destination, en suivant son chemin associé. Chaque arête est munie d'une file d'attente, et à chaque pas de temps, un paquet peut soit attendre dans une file d'attente, soit passer d'une file d'attente à une autre en traversant l'arête de son chemin correspondante. Le but est de trouver un schedule, qui détermine les mouvements des paquets étape par étape. On cherche à minimiser le nombre d'étapes nécessaires pour que tous les paquets atteignent leur destination, ainsi que la taille nécessaire pour les files d'attentes. La *dilation* est la taille maximale des chemins considérés. La *congestion* est le nombre maximal de paquet dont les chemins associés empruntent une même arête. Le *routing time* est le nombre d'étapes que prend un schedule. La *mémoire tampon* requise par un routage correspond au nombre maximal de paquets qui se retrouvent dans la file d'attente d'une même arête à un moment donné.

La figure 1 illustre un exemple de routage avec un routing time de 4,

Un algorithme naïf nous donne le schedule suivant : pour un réseau avec des paquets de congestion C et de dilation D, on divise le schedule en D séquences. À chaque séquence on fait passer un paquet d'un sommet au suivant selon son chemin, ainsi, à la fin des D séquences, chacun des paquets aura atteint sa destination. Comme à chaque séquence, on peut avoir jusqu'à C paquets qui doivent traverser la même arête, en faisant passer ces C paquets "un à un", on obtient un schedule où chaque séquence peut prendre jusqu'à C étapes. Ainsi, on a un routing time en $\mathcal{O}(C \cdot D)$.



(a) Situation initiale

(b) "Pendant" l'étape 1

(c) À la fin de l'étape 1

(d) "Pendant" l'étape 2

(e) À la fin de l'étape 2

(f) "Pendant" l'étape 3

(g) "Pendant" l'étape 4

(h) Situation finale

FIGURE 1 – Un exemple de routage

5.2 Minimiser le routing time

Il est possible d'améliorer le routing time par rapport à celui de l'algorithme naïf. Dans cette section, on reprend la construction d'un routage avec routing time en $\mathcal{O}(C+D)$ donnée dans le livre de C. Scheideler Universal Routing Strategies for Interconnection Networks [7], qui lui même s'inspire des travaux de Leighton, Maggs et Rao.

Théorème 5.1. *Pour un ensemble de paquets avec des chemins simples (c'est-à-dire qu'aucun chemin ne passe deux fois par la même arête) de congestion C et de dilation D, il existe un schedule pour acheminer tous les paquets avec un routing time en $\mathcal{O}(C+D)$.*

Démonstration. Un *t-intervalle* est une suite de t étapes consécutives. Un *t-frame* est un t-intervalle qui commence à un entier multiple de t. Soit $I_0 = \max\{C, D\}$ et $I_j = \log I_{j-1}$ pour tout $j \geq 1$.

Pour construire notre schedule, on va effectuer une succession d'affinage à partir d'un schedule initial S_1, qui ne respecte pas nécessairement la contrainte qu'une arête ne peut être empruntée que par un seul paquet à chaque étape. Ainsi, on assigne à chaque paquet un délai initial δ entre 0 et $C-1$, choisi de manière aléatoire, équiprobable et indépendante. Ainsi, dans S_1, le paquet attend sur son sommet initial pendant δ étapes avant d'avancer à chaque étape suivante, "sans s'arrêter", jusqu'à son sommet final. Le routing time de S_1 est donc $D+C$.

Maintenant, on utilise le corollaire du lemme local de Lovász pour montrer que si on a un I_1 suffisamment grand, alors, avec une probabilité strictement positive, la congestion pour chaque arête sur tout ${I_1}^3$-intervalle est au plus $C_1 = (1+\frac{3}{I_1}) \cdot {I_1}^3$ (c'est-à-dire que pour chaque arête il y a au plus C_1 paquets qui transitent par cette arête au cours d'un ${I_1}^3$-intervalle quelconque). Ainsi, il existe une assignation de délais tel que dans S_1, sur tout ${I_1}^3$-intervalle, la congestion de toute arête est au plus C_1.

Pour appliquer le corollaire, on associe à chaque arête e un "mauvais" événement A_e : "il y a davantage que $C_1 = (1+\frac{3}{I_1}) \cdot {I_1}^3$ paquets qui transitent par e lors d'un ${I_1}^3$-intervalle quelconque".

Observons que l'événement A_e est mutuellement indépendant de tous les événements A_f, pour f une arête telle qu'aucun paquet passant par e ne passe aussi par f. (L'événement A_e est uniquement dépendant des délais assignés aux paquets transitant par e, mais si aucun de ces paquets ne passe par f, savoir que des événements A_f ont lieu ne change rien à la probabilité de A_e.) Ainsi, on peut prendre $d = C \cdot D$.

Majorons maintenant la probabilité qu'un "mauvais événement" ait lieu. Considérons une arête e et J un ${I_1}^3$-intervalle. Numérotons les paquets traversant e de 1 à $m \leq C$. Pour tout $i \in \{1, 2, ..., m\}$, soit X_i la variable aléatoire qui vaut 1 si et seulement si le paquet i transite par e pendant J. Soit $X = \sum_{i=1}^{m} X_i$. Puisque les délais assignés aux paquets sont déterminés de manière aléatoire uniforme sur $\{0, 1, ..., C-1\}$, la probabilité qu'un paquet traverse e pendant une étape donnée est inférieure ou égale à $1/C$, et donc la probabilité qu'il transite par e pendant une des ${I_1}^3$ étapes de J

est bornée par ${I_1}^3/C$. Ainsi $\mathbb{P}[X_i = 1] \leq {I_1}^3/C \quad \forall i \in \{1, 2, ...m\}$. D'où :

$$\mathbb{E}[X] = \sum_{i=1}^{m} \mathbb{P}[X_i = 1] \leq \sum_{i=1}^{m} \frac{{I_1}^3}{C} \leq {I_1}^3. \tag{5}$$

Le théorème de Chernoff nous donne la borne suivante : si on a une collection de variables aléatoires mutuellement indépendantes $X_1, X_2, ..., X_n$, et $\mathbb{E}[\sum_{i=1}^{n} X_i] = \mu$, alors, $\forall \epsilon \geq 0$ on a :

$$\mathbb{P}\left(\sum_{i=1}^{n} X_i \geq (1+\epsilon) \cdot \mu\right) \leq \left(\frac{e^{\epsilon}}{(1+\epsilon)^{(1+\epsilon)}}\right)^{\mu} \quad ;$$

et pour $\epsilon < 1$, on a de plus :

$$\mathbb{P}\left(\sum_{i=1}^{n} X_i \geq (1+\epsilon) \cdot \mu\right) \leq \exp\left(\frac{-\mu \cdot \epsilon^2}{2}\right).$$

Ainsi, en appliquant la borne de Chernoff avec $\epsilon = \frac{3}{I_1}$, (5) nous donne que :

$$\mathbb{P}[X \geq C_1] = \mathbb{P}[X \geq (1 + \frac{3}{I_1}) \cdot {I_1}^3] \leq \exp\left(-\left(\frac{3}{I_1}\right)^2 \cdot \frac{{I_1}^3}{2}\right) \leq 2^{-5I_1}.$$

Puisque pour chaque arête e il y a au plus $(C + D)$ ${I_1}^3$-intervalles à considérer, on a que la probabilité d'un "mauvais événement" A_e est inférieure ou égale à $(D + C) \cdot 2^{-5I_1}$. Observons encore que pour $I_1 \geq 4$ (ce qui implique $C, D \geq 16$), comme $32 \cdot e \cdot (256 + 1) < 2^{20}$, on a $32 \cdot 2^{-20} < 1/(e \cdot (256 + 1))$, on se retrouve avec :

$$\mathbb{P}(A_e) \leq (D + C) \cdot 2^{-5I_1} < \frac{1}{e \cdot (C \cdot D + 1)}.$$

Ainsi, on peut appliquer le corollaire 3.2 du lemme local de Lovász avec $p = \frac{1}{e \cdot (C \cdot D + 1)}$ et $d = C \cdot D$, puisqu'on a bien $e \cdot p \cdot (d + 1) < 1 \ \forall I_1 \geq 4$, et donc, il existe un schedule S_1 avec une congestion pour tout ${I_1}^3$-intervalle bornée par $C_1 = \left(1 + \frac{3}{I_1}\right) \cdot {I_1}^3$. Observons encore que S_1 a un longueur (C+D).

Maintenant, nous décomposons S_1 en ${I_1}^4$-frame, et regardons séparément chaque frame comme un problème de routage indépendant : pour chaque paquet, son sommet d'origine est son sommet au début de la frame, et son sommet de destination est son sommet à la fin de la frame dans S_1. Comme précédemment, nous allons affiner S_1 en assignant pour chaque frame un délai initial à chaque paquet qui traverse au moins une arête durant cette frame. Nous obtiendrons ainsi un schedule S_2, en juxtaposant les schedule correspondant aux frames.

Considérons une frame F, et assignons à chaque paquet un délai initial entre 0 et $({I_1}^3 - {I_2}^3)$. Ainsi, le routing time du schedule de F est au plus ${I_1}^4 + {I_1}^3$. De la même manière que précédemment, on utilise le corollaire du lemme local de Lovász pour montrer qu'il existe un assignement de délais pour les paquets tels que pour tout

${I_2}^3$-intervalle, la congestion est au plus :

$$C_2 = \left(1 + \frac{3}{I_1}\right) \cdot \left(1 + \frac{4}{I_2}\right) \cdot \left(1 + \frac{1}{1 - (I_2/I_1)^3}\right) \cdot {I_2}^3.$$

Observons que le schedule S_2 ainsi obtenu aura une longueur :

$$\frac{(C+D)}{{I_1}^4} \cdot \left({I_1}^4 + {I_1}^3 - {I_2}^3\right) = (C+D) \cdot \left(1 + \frac{{I_1}^3}{{I_1}^4} - \frac{{I_2}^3}{{I_1}^4}\right)$$

$$= (C+D) \cdot \left(1 + \frac{1}{I_1} - \frac{{I_2}^3}{{I_1}^4}\right) \leq (C+D) \cdot \left(1 + \frac{1}{I_1}\right).$$

(Ici nous avons omis les petites variations induites par le fait qu'on doive prendre des parties entières, mais qui ne changent pas grand chose au final.)

Nous continuons de la sorte à décomposer les S_i, jusqu'à ce qu'on arrive au rang k, tel que $I_k = 4$. (Si on devait avoir $I_k < 4$, pour la k-ième étape d'affinage, on remplace en fait les I_i par des I'_i que l'on définit de la manière suivante : $I'_i = I_i$ pour $i = 0, 1$; $I'_2 = \max\{4, s\}$ avec s le plus petit entier tel que $\log^*(s) = \lfloor \log^* I_0 \rfloor$; et $I'_{i+1} = \log I'_i$ pour $i \geq 2$. Nous retombons ainsi, avec quelques légères variations, avec les même propriétés que précédemment.) On pourrait montrer qu'on obtient ainsi un schedule S_k de longueur totale au plus :

$$(D+C) \cdot \prod_{j=1}^{k-1} \left(1 + \frac{1}{I_j}\right) \leq 1,1 \cdot (C+D) \ ,$$

et qui pour chaque ${I_k}^3$-intervalle admet une congestion C_k d'au plus :

$$ {I_k}^3 \cdot \left(1 + \frac{3}{I_1}\right) \cdot \prod_{j=1}^{k-1} \left(1 + \frac{4}{I_{j+1}}\right) \cdot \left(\frac{1}{1 - \left(I_{j+1}/I_j\right)^3}\right) \leq 2,5 \cdot {I_k}^3.$$

Maintenant nous décomposons S_k en frames de longueur $4^3 = 64$, et nous allons considérer une frame F donnée. Nous assignons à chaque paquet un délai initial entre 0 et 64 comme précédemment. Nous allons à nouveau utiliser le corollaire du lemme local de Lovász afin d'exhiber un assignement des délais qui nous convienne.

De la même manière que précédemment, un événement A_e est mutuellement indépendant de l'ensemble de tous les événements A_g tel qu'aucun paquet passant par e lors de F ne passe par l'arête g lors de F. Or, comme il y a au plus $C_k = 2,5 \cdot 64$ paquets passant par e, et que ces derniers passent par au plus 64 autre arêtes (64 est la longueur maximale de F), on a que A_e est mutuellement indépendant d'un ensemble de tous les événements A_g privé d'au plus $d = 2,5 \cdot 64^2$ événements.

Le nombre moyen de paquets traversant une même arête à une étape donnée est borné par 2,5 au vu de la congestion C_k relative à tout ${I_k}^3$-intervalle. Soit X une variable aléatoire qui compte le nombre de paquets transitant par une arête e lors

d'une étape donnée. En appliquant la borne de Chernoff avec $\epsilon = 6$, on obtient :

$$\mathbb{P}[X \geq (1+\epsilon) \cdot 2,5] \leq \left(\frac{e^6}{(6+1)^{6+1}} \right)^{2,5}.$$

Puisqu'il y a au plus $2 \cdot 64$ pas de temps à considérer, la probabilité qu'au moins $(1+\epsilon) \cdot 2,5$ paquets transitent par une arête e lors d'une étape quelconque est bornée comme suit :

$$\mathbb{P}(A_e) \leq 128 \cdot \left(\frac{e^6}{7^7} \right)^{2,5} < \frac{1}{e \cdot \left(2,5 \cdot 64^2 + 1 \right)} \tag{6}$$

(la dernière inégalité se montre par calcul brut.)

Ainsi, nous sommes en mesure d'appliquer le corollaire du lemme local de Lovász avec $d = 2,5 \cdot 64^2$ et $p = 128 \cdot \left(\frac{e^6}{7^7} \right)^{2,5}$, car (6) implique que $e \cdot p \cdot (d+1) < 1$. Il existe donc un assignement de délais tels que au plus $(1+6) \cdot 2,5$ paquets passent par une arête lors d'une étape quelconque. En faisant passer "un à un" ces paquets en leur consacrant une étape chacun à la manière de l'algorithme naïf ; on obtient pour ces frames un schedule avec un routing time de $2 \cdot 64 \cdot 7 \cdot 2,5$, qui respecte la contrainte que pour chaque étape et chaque arête, au plus un paquet transite par cette arête.

En juxtaposant les schedules de chacune des frames, qui sont en nombre au plus $\frac{1,1 \cdot (C+D)}{64}$, on obtient un schedule final de longueur $1,1 \cdot 2 \cdot 7 \cdot 2,5 \cdot (C+D) \leq 39 \cdot (C+D)$. (Ici encore, nous avons négligé quelques détails concernant des parties entières, qui ajoutent en fait quelques constantes. Néanmoins, on a bien un schedule final avec un routing time en $\mathcal{O}(C+D)$.)

Ceci prouve donc le résultat souhaité. □

Remarque. Dans la démonstration qui précède, nous n'avons pas parlé de la taille nécessaire pour les files d'attentes aux arêtes. En fait, le schedule construit lors de la preuve peut nécessiter une mémoire tampon à chaque arête de taille jusqu'à $\mathcal{O}((\log(C+D))^3)$. Dans les frames du dernier affinage, comme a une congestion à chaque pas de temps d'au plus $2,5 \cdot 7$, on pourrait se satisfaire de mémoire tampon de taille constante si on les regardent comme un problème de routage indépendant. Néanmoins, dans ces frames, nous ne tenons plus compte des paquets qui "ne bougent pas" lors de l'intervalle de temps considérés. De fait, à chaque arête, il peut y avoir jusqu'à $2,5 \cdot 7$ paquets qui attendent de se mouvoir dans r étapes. Ainsi, on doit considérer l'ensemble des valeurs possibles de r dans ces dernières frames. Comme on fait abstractions des mémoires tampon nécessaires aux sommets initiaux et de destination des paquets, on n'a qu'à considérer la sommes des valeurs maximales des délais supplémentaires distribués ensuite à chaque affinage. C'est à dire $({I_1}^3 - {I_2}^3) + ({I_2}^3 - {I_3}^3) + \ldots$. Ainsi on se retrouve bien avec une mémoire tampon nécessaire en $\mathcal{O}({I_1}^3)$, et comme $I_1 = \max\{\log C, \log D\}$, on a bien une mémoire tampon requise à chaque arête en $\mathcal{O}((\log(C+D))^3)$.

Néanmoins, toujours à l'aide du lemme de Lovász, il est possible de construire un routage avec une mémoire tampon nécessaire à chaque arrête de taille constante égale à 3. Ce routage conserve un routing time en $\mathcal{O}(C+D)$, mais la constante correspondante est beaucoup plus importante que pour celui nous avons développé ici.

6 Une preuve constructive du Lemme

Dans cette section, nous étudions une preuve constructive du lemme local de Lovász 3.1, donnée par R.A. Moser dans son travail [6]. La version du lemme ainsi démontrée est néanmoins légèrement plus faible. Elle suppose par exemple qu'il existe un ensemble fini de variables aléatoires mutuellement indépendantes telles que les événements considérés dans le lemme soient déterminés par les valeurs que prennent ces variables.

6.1 Un algorithme simple

Soit $\mathcal{P}$ une collection finie de variables aléatoires mutuellement indépendantes dans un espace de probabilité Ω. Soit A un événement de Ω déterminé par les variables de $\mathcal{P}$. On voit facilement qu'il existe un unique sous ensemble minimal de $\mathcal{P}$ dont les variables déterminent A, et on note ce sous-ensemble $\mathrm{vbl}(A)$.

Soit $\mathcal{A}$ une famille finie d'événements de Ω déterminés par $\mathcal{P}$. Soit $G_{\mathcal{A}}$ le graphe de dépendance tel que les sommets sont les éléments de $\mathcal{A}$, et tel qu'il admette une arête (A, B) si et seulement si $\mathrm{vbl}(A) \cap \mathrm{vbl}(B) \neq \emptyset$ et $A \neq B$. On note $\Gamma(A) = \Gamma_{\mathcal{A}}(A)$ l'ensemble des voisins de A dans $G_{\mathcal{A}}$. On pose également $\Gamma^+(A) = \Gamma(A) \cup \{A\}$.

Le but dans cette section sera de montrer le théorème suivant :

Théorème 6.1. *Avec $\mathcal{A}$ et $\mathcal{P}$ définis comme précédemment, s'il existe x une assignation de réels $x : \mathcal{A} \to]0, 1[$ telle que, $\forall A \in \mathcal{A}$:*

$$\mathbb{P}(A) \leq x(A) \cdot \prod_{B \in \Gamma(A)} (1 - x(B)) \quad ;$$

alors il existe une assignation de valeurs pour les variables de $\mathcal{P}$ telle qu'aucun des événements de $\mathcal{A}$ n'ait lieu.

La preuve s'articule autour d'un algorithme simple : pour chaque variable $P \in \mathcal{P}$ on note v_P la valeur qu'elle prend. On commence par une évaluation aléatoire des variables. Si il existe un événement A qui a lieu pour une évaluation donnée, on dit que celle-ci *viole* A. Si il existe des événements de $\mathcal{A}$ qui sont violés par l'évaluation courante, on choisit arbitrairement A l'un d'entre eux, et on procède à un ré-échantillonnage de A. C'est à dire que l'on ré-évalue chacune des variables de $\mathrm{vbl}(A)$, indépendamment et selon leur loi de probabilité, ce qui nous donne une nouvelle évalution des variables de $\mathcal{P}$ (on ne change pas l'évalutaion des variables de $\mathcal{P} \setminus \mathrm{vbl}(A)$). On répète ce processus et au final on obtient une évaluation des variables de $\mathcal{P}$ qui ne viole aucun événements de $\mathcal{A}$. Cet algorithme correspond au pseudo-code suivant :

for all $P \in \mathcal{P}$ **do**
  $v_P \leftarrow$ une évaluation aléatoire de P ;
end for
while $\exists A \in \mathcal{A}$ tel que A est violé si $P = v_P$ pour tout $P \in \mathcal{P}$ **do**
  on choisit arbitrairement un événement violé $A \in \mathcal{A}$;

for all $P \in \mathcal{P}$ **do**
$v_P \leftarrow$ une nouvelle évaluation de P ;
end for
end while
return $(v_P)_{P\in\mathcal{P}}$;

Néanmoins, a priori, un tel algorithme pourrait ne jamais se terminer. Notre but va justement être de montrer que sous les hypothèses du théorème 6.1, pour un événement A, l'espérance du nombre de ré-échantillonnage de A par cet algorithme avant de retourner une évaluation satisfaisante est au plus $x(A)/(1-x(A))$. Ainsi, l'espérance du nombre de ré-échantillonages total pour l'algorithme est au plus : $\sum_{A\in\mathcal{A}} \frac{x(A)}{1-x(A)}$.

Dans l'algorithme, nous choisissons arbitrairement un des événements violés. Soit $C : \mathbb{N} \to \mathcal{A}$ le *journal d'exécution* qui liste les choix effectués. Si l'algorithme termine, C n'est en fait que défini jusqu'au nombre de ré-échantillonnages effectués par l'algorithme, et non sur tout $\mathbb{N}$.

Un *arbre témoin* $\tau = (T, \sigma_T)$ consiste en un arbre enraciné fini T muni d'une fonction d'étiquetage $\sigma_T : V(T) \to \mathcal{A}$ de ses sommets vers des événements tel que les enfants d'un sommet $u \in V(T)$ reçoivent une étiquette dans $\Gamma^+(\sigma_T(u))$. Si pour tout sommet, chacun de ses enfants reçoit une étiquette distincte de celles de ses "frères", on dit que l'arbre témoin est *propre*. Pour raccourcir les notations, on pose $V(\tau) := V(T)$ et pour tout $v \in V(\tau)$, on écrit $[v] := \sigma_T(v)$.

Pour un journal C donné, on va maintenant, à chaque étape de ré-échantillonnage t construire un arbre témoin $\tau_C(t)$ qui puisse servir de "justification" quant à la nécessité du ré-échantillonnage concerné. Pour ce faire, on pose $\tau_C^{(t)}(t)$ comme étant une racine isolée étiquetée $C(t)$. On va maintenant construire $\tau_C^{(i)}(t)$ pour $i < t$ par récurrence : Si il existe au moins un sommet $v \in \tau_C^{(i+1)}(t)$ tel que $C(i) \in \Gamma^+([v])$, alors on choisit arbitrairement parmi ces sommets un de ceux dont la distance à la racine est maximale, on lui attache un nouveau sommet u avec $[u] = C(i)$, et on obtient ainsi $\tau_C^{(i)}(t)$. Si il n'existe pas de tel sommet $v \in \tau_C^{(i+1)}(t)$ tel que $C(i) \in \Gamma^+([v])$, alors on "saute" simplement l'étape i en posant $\tau_C^{(i)}(t) := \tau_C^{(i+1)}(t)$. Enfin, on définit $\tau_C(t) := \tau_C^1(t)$.

On dit que l'arbre témoin τ *apparaît* dans le journal C si il existe $t \in \mathbb{N}$ tel que $\tau_C(t) = \tau$.

Lemme 6.2. *Soit τ un arbre témoin et C un journal produit par l'algorithme.*

1. *Si τ apparaît dans C, alors τ est propre.*
2. *La probabilité que τ apparaisse dans C est au plus $\prod_{v\in V} \mathbb{P}[[v]]$.*

Démonstration. (1) : Comme τ apparaît dans le journal C, il existe $t \in \mathbb{N}$ tel que $\tau_C(t) = \tau$. Pour un sommet $v \in V(\tau)$, soit $d(v)$ sa profondeur, c'est à dire sa distance par rapport à la racine, et soit $q(v)$ l'étape à laquelle v a été attaché à l'arbre témoin, c'est-à-dire que q correspond au plus grand entier tel que v est contenu dans $\tau_C^{(q)}(t)$.

Tout d'abord, observons que si $q(u) < q(v)$ pour des sommets $u, v \in V(\tau)$ (ce qui implique que u ait été "attaché à l'arbre" après v) et si $\text{vbl}([u]) \cap \text{vbl}([v]) \neq \emptyset$, alors (par construction) $d(u) > d(v)$. Ainsi si deux sommets $u, v \in V$ ont la même profondeur $d(u) = d(v)$, alors $\text{vbl}([u]) \cap \text{vbl}([v]) = \emptyset$. Ceci implique que deux enfants d'un même sommet (qui ont donc la même profondeurs) sont forcément distincts, et on peut ainsi conclure que τ est propre.

(2) : Considérons le procédé suivant, que l'on nommera τ-*check* : dans un ordre à profondeur décroissante (par exemple, l'inverse de l'ordre breadth-first search), on visite les sommets de τ et pour un sommet v, on prend une évaluation aléatoire de des variables de $\text{vbl}([v])$ (selon leur loi de probabilité et indépendamment d'éventuelles précédentes évaluations). On dit que le τ-check *passe* si à chaque visite de sommet l'événement $[v]$ était violé pour l'évaluation considérée.

Clairement, la probabilité que le τ-check passe est exactement $\prod_{v \in V(\tau)} \mathbb{P}([v])$.

De manière intuitive, on voit que si τ apparait dans C, c'est que "le τ-check effectué par l'algorithme" passe forcément. En effet, au cours de l'algorithme, on procède à des ré-échantillonnages d'événements violés, ce qui correspond bien à notre construction du τ-check. Plus proprement, on peut montrer que si on prend la même "source aléatoire" pour l'algorithme et le τ-check (un retirage d'une variable étant remplacé par la lecture de la valeur suivante dans une table de valeurs prises successivement pour la variable à retirer, à la manière de ce qui sera fait dans plus tard pour l'algorithme déterministe) ; alors si τ apparait dans C c'est forcément que le τ-check passe.

Ceci implique donc que la probabilité que τ apparaisse dans C vaut au plus la probabilité que le τ-check passe, ce qui complète la preuve de ce lemme.

□

Pour un événement $A \in \mathcal{A}$, soit N_A la variable aléatoire qui compte le nombre de fois où l'événement A a été ré-échantillonné durant l'exécution de l'algorithme. Si C est le journal d'exécution de l'algorithme, N_A représente le nombre de fois où A apparaît dans C. Observons que N_A correspond également au nombre d'arbres témoins distincts de C dont le sommet racine a pour étiquette l'événement A. En effet, si t_i est la i-ième étape avec $C(t_i) = A$, alors clairement, l'arbre témoin $\tau_C(t_i)$ comporte i sommets dont l'étiquette correspond à l'événement A (car le seul moyen que l'événement $C(t')$ n'apparaisse pas dans $\tau_C(t_i)$ pour $t' < t_i$, est que $\text{vbl}(C(t')) \cap \text{vbl}(C(t_i)) = \emptyset$, ce qui est trivialement faux pour $C(t') = C(t_i)$). Ainsi, puisque pour $i \neq j$, $\tau_C(t_i)$ et $\tau_C(t_j)$ comportent un nombre différent de sommets dont l'étiquette correspond à l'événement A, on a bien $\tau_C(t_i) \neq \tau_C(t_j)$. Comme à chacune des N_A étapes t_i telles que $C(t_i) = A$, $\tau_C(t_i)$ a pour racine A, et que ces $\tau_C(t_i)$ sont tous distincts, on a bien au final N_A arbres témoins dont la racine a pour étiquette A qui apparaissent dans C. (Puisque pour un t'' tel que $C(t'') = B \neq A$, la racine de $\tau_C(t'')$ a pour étiquette $B \neq A$.)

Ainsi on peut borner l'espérance de N_A simplement en sommant les bornes données par le lemme 6.2 sur la probabilité d'occurrence des différents arbres témoins propres.

Soit $A \in \mathcal{A}$. Nous allons maintenant nous intéresser au processus de Galton-Watson, qui permet de générer des arbres témoins propres ayant pour racine un sommet ayant pour étiquette l'événement A. L'étape initiale consiste en un singleton dont l'étiquette est A. À chaque étape suivante, on considère indépendamment chacun des sommets v produits à l'étape précédente, et pour chaque événement $B \in \Gamma^+([v])$, aléatoirement et indépendamment, on attache à v un sommet fils ayant pour étiquette B avec une probabilité $x(B)$; ou bien on n'attache pas de tel sommet (avec donc une probabilité $1 - x(B)$. Ce processus continue jusqu'à ce qu'il s'arrête naturellement car aucun nouveau sommet n'aura été attaché lors d'une étape. (Il est donc possible que le processus ne se termine jamais.)

On définit $x'(B) := x(B) \cdot \prod_{C \in \Gamma(B)}(1 - x(C))$.

Lemme 6.3. *Soit τ un arbre témoin propre avec comme racine un sommet dont l'étiquette correspond à l'événement A. La probabilité p_τ que l'algortihme de Galton Watson décrit ci-avant nous produise exactement l'arbre τ vaut :*

$$p_\tau = \frac{1 - x(A)}{x(A)} \cdot \prod_{v \in V((\tau)} x'([v]).$$

Démonstration. Pour un sommet $v \in V(\tau)$ on note $W_v \subset \Gamma^+([v])$ l'ensemble des éléments de $\Gamma^+([v])$ qui n'apparaissent pas comme étiquette d'un des fils de v dans τ. Comme chaque décision de l'algorithme est indépendante, que la probabilité que chacun des sommets v de τ apparaisse vaut $x([v])$ (sauf pour la racine v_r) ; et que la probabilité que chacun des événements u dans chacun des W_v n'apparaisse pas est (1-x([u])), on a :

$$p_\tau = \prod_{v \in V(\tau) \setminus v_r} x([v]) \cdot \prod_{v \in V(\tau)} \prod_{u \in W_v} (1 - x([u])) = \frac{1}{x(A)} \cdot \prod_{v \in V(\tau)} \left(x([v]) \cdot \prod_{u \in W_v} (1 - x([u])) \right).$$

On souhaiterait maintenant de se débarrasser du W_v. Or, chacun des les événements dans $\Gamma^+(v)$ qui ne sont pas dans W_v correspondent à des fils de v dans τ. Ainsi, on peut récrire l'expression en remplaçant W_v par $\Gamma^+(v)$ (pour peu qu'on remplace $x([v])$ par $\frac{x([v])}{1-x([v])}$ afin de compenser le facteur $(1 - x([v])$ en trop qu'on a compté lors du produit sur $\Gamma^+(v')$, où v' est le père de v). Il ne nous reste qu'à observer que le sommet racine est le seul n'ayant pas de père, et on obtient le résultat suivant :

$$p_\tau = \frac{1 - x(A)}{x(A)} \cdot \prod_{v \in V(\tau)} \left(\frac{x([v])}{1 - x([v])} \cdot \prod_{u \in \Gamma^+([v])} (1 - x([u])) \right).$$

En remplaçant les voisinages inclusifs $\Gamma^+(v)$ par des voisinages exclusifs $\Gamma(v)$, cette

expression se simplifie en :

$$p_\tau = \frac{1-x(A)}{x(A)} \cdot \prod_{v \in V(\tau)} \left(x([v]) \cdot \prod_{u \in \Gamma([v])} (1 - x([u])) \right) = \frac{1-x(A)}{x(A)} \cdot \prod_{v \in V(\tau)} x'([v]),$$

ce qui correspond au résultat que l'on souhaitait montrer. □

Nous sommes maintenant en mesure de prouver facilement que, sous les hypothèses du théorème 6.1, l'espérance du nombre de ré-échantillonnages utilisés lors de l'algorithme est au plus $\sum_{A \in \mathcal{A}} \frac{x(A)}{1-x(A)}$.

Démonstration. Soit $\mathcal{T}_A$ l'ensemble de tous les arbres témoins propres dont la racine est un sommet avec A comme étiquette. Ainsi, $\mathbb{E}(N_A) = \mathbb{E}(|\{\tau \in \mathcal{T}_A : \tau \text{ apparaît dans} C\}|)$. Soit Z_τ la variable aléatoire qui vaut 1 si τ apparaît dans C, 0 sinon. On a ainsi :

$$\mathbb{E}(N_A) = \mathbb{E}\left(|\{\tau \in \mathcal{T}_A : \tau \text{ apparaît dans } C\}|\right) = \mathbb{E}\left(\sum_{\tau \in \mathcal{T}_A} Z_\tau\right) = \sum_{\tau \in \mathcal{T}_A} \mathbb{E}(Z_\tau)$$

$$= \sum_{\tau \in \mathcal{T}_A} \mathbb{P}(Z_\tau) = \sum_{\tau \in \mathcal{T}_A} \mathbb{P}["\tau \text{ apparaît dans } C"] \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in V(\tau)} \mathbb{P}[[v]] \leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in V(\tau)} x'([v]).$$

La première inégalité vient du lemme 6.2 tandis que la seconde est une conséquence directe des hypothèses.

Observons que le lemme 6.3 nous donne que $p_\tau = \frac{1-x(A)}{x(A)} \cdot \prod_{v \in V((\tau)} x'([v]) \Rightarrow \prod_{v \in V((\tau)} x'([v]) = p_\tau \cdot \frac{x(A)}{1-x(A)}$, ce qui nous permet de poursuivre :

$$\begin{aligned} \mathbb{E}(N_A) &\leq \sum_{\tau \in \mathcal{T}_A} \prod_{v \in V(\tau)} x'([v]) = \sum_{\tau \in \mathcal{T}_A} \frac{x(A)}{1-x(A)} \cdot p_\tau \\ &= \frac{x(A)}{1-x(A)} \cdot \sum_{\tau \in \mathcal{T}_A} p_\tau \leq \frac{x(A)}{1-x(A)}, \end{aligned}$$

où la dernière égalité est induite du fait que l'algorothme de Galton-Watson ne produit qu'un seul arbre à la fois, et pas nécessairement dans $\mathcal{T}_A$ car potentiellement infini.

Ceci permet donc de conclure que l'espérance du nombre de ré-échantillonnage utilisé par l'algorithme est au plus $\sum_{A \in \mathcal{A}} \mathbb{E}(N_A) = \sum_{A \in \mathcal{A}} \frac{x(A)}{1-x(A)}$. □

On rappelle que si l'algorithme termine, il existe une assignation des variables telle qu'aucun des événements de $\mathcal{A}$ ne soit violé. Or, sous les hypothèses du théorème 6.1, on vient de montrer que l'espérance du nombre de ré-échantillonnages est un nombre fini (car $\mathcal{A}$ est fini et que $x(A) < 1 \ \forall A$) ; ce qui implique donc forcément qu'il y a une assignation de valeurs pour les variables de $\mathcal{P}$ telle qu'aucun événement de $\mathcal{A}$ ne soit violé (sinon, l'algorithme ne terminerait jamais et on n'aurait pas une espérance du nombre de ré-échantillonnages finie). Ceci prouve donc le théorème 6.1.

6.2 L'algorithme "en parallèle"

Dans cette section, nous allons légèrement modifier l'algorithme précédent afin d'en obtenir un encore plus performant. Au lieu, à chaque pas, de ré-échantillonner un événement violé choisi arbitrairement, on va en ré-échantillonner plusieurs, en faisant attention à ce que les événements concernés ne dépendent d'aucune variable commune entre eux. Ainsi, à chaque pas, on va considérer le sous-graphe G_i du graphe $G_{\mathcal{A}}$ de dépendance induit par l'ensemble des sommets correspondant aux événements violés, et on va considérer un ensemble maximal indépendant S. (C'est à dire, $S \subset G_i$ tel que $\forall A, B \in S$, $\text{vbl}(A) \cap \text{vbl}(B) = \emptyset$; et $|S| \geq |S'| \quad \forall S'$ tel que $A', B' \in S' \Rightarrow \text{vbl}(A) \cap \text{vbl}(B)$. On sait également que trouver un tel ensemble S est un problème qui peut être résolu en temps polynomial.) Ensuite, on effectue un ré-échantillonnage des variables de $\bigcup_{A \in S} \text{vbl}(A)$ (toujours indépendamment, toujours selon leurs lois de probabilité, et toujours sans modifier les autres variables), et on procède ainsi de suite, jusqu'à ce qu'il n'y ait plus d'événement violé.

En pseudo-code, voici notre algorithme :

for all $P \in \mathcal{P}$ **do**
  $v_P \leftarrow$ une évaluation aléatoire de P ;
end for
while $\exists A \in \mathcal{A}$ tel que A est violé si $P = v_P$ pour tout $P \in \mathcal{P}$ **do**
  $S \leftarrow$ un ensemble indépendant maximal dans le sous-graphe de $G_{\mathcal{A}}$ induit par tous les événements de $\mathcal{A}$ violés quand $P = v_P$ pour tout $P \in \mathcal{P}$;
  for all $P \in \bigcup_{A \in S} \text{vbl}(A)$ **do**
    $v_P \leftarrow$ une nouvelle évaluation de P ;
  end for
end while
return $(v_P)_{P \in \mathcal{P}}$;

Cet algorithme pouvant être vu comme un cas particulier du précédent, les théorèmes que nous avons vu précédemment sont toujours valables. Afin de pouvoir munir cet algorithme d'une borne logarithmique sur l'espérance du nombre de ré-échantillonnage, on va devoir reformuler le lemme avec des conditions légèrement plus restrictives :

Théorème 6.4. *Soit $\mathcal{P}$ un ensemble fini de variables mutuellement dans un espace de probabilité. Soit $\mathcal{A}$ un ensemble fini d'événements déterminés par ces variables. Si il existe $\epsilon > 0$ et une assignation de réels $x : \mathcal{A} \to]0, 1[$ tels que, pour tout A dans $\mathcal{A}$:*

$$\mathbb{P}[A] \leq (1 - \epsilon) \cdot x(A) \cdot \prod_{B \in \Gamma_{\mathcal{A}}(A)} (1 - x(B)),$$

alors la version "en parallèle" de l'algorithme admet une espérance pour le nombre d'étapes qu'il prendra en $\mathcal{O}\left(\frac{1}{\epsilon} \log \sum_{A \in \mathcal{A}} \frac{x(A)}{1 - x(A)}\right)$.

Soit C un journal d'exécution de cet algorithme. En choisissant arbitrairement un ordre sur les événements violés choisis à chaque étape de ré-échantillonnage, et en considérant que l'on a ré-échantillonné ces événements un à un, on obtient un journal

de l'algorithme séquentiel (le premier algorithme que nous avons construit). Soit S_j le segment du journal C correspondant à la j-ième étape de l'algorithme "en parallèle". On rappelle la *profondeur* d'un arbre témoin est la profondeur maximale d'un sommet dans cet arbre.

Lemme 6.5. *Si $t \in S_j$, alors la profondeur de $\tau_C(t)$ est $j-1$.*

Démonstration. Soit t_k le premier nombre du segment S_k et soit $\tau_k = \tau_C^{(t_k)}(t)$ pour $k \leq j$.

Comme les événements ré-échantillonnés lors de la j-ième étape de l'algorithme "en parallèle" sont indépendants, la racine est le seul sommet de τ_j. Pour $k < j$ on obtient τ_k à partir de τ_{k+1} en y attachant des sommets correspondant à la k-ième étape de l'algorithme "en parallèle". Comme les étiquettes de ces sommets correspondent à des événements indépendants entre eux, ils ne peuvent accroître que d'une unité la profondeur de l'arbre.

Pour voir qu'ils accroissent forcément d'une unité la profondeur, on procède par l'absurde : soit k' le plus grand entier tel que la profondeur de $\tau_{k'}$ est égale à la profondeur de $\tau_{k'+1}$. On considère un sommet v de $\tau_{k'+1}$ avec profondeur maximale. Ce sommet correspond donc à un événement $[v]$ ré-échantillonné après la k'-ième étape de l'algorithme "en parallèle". Si $\tau_{k'}$ n'a pas de sommet de profondeur supérieure à celle de v, c'est donc qu'aucun événement de $\Gamma^+([v])$ n'a été ré- échantillonné durant la k'-ième étape. Donc soit v devait déjà être violé à la k-ième étape, ce qui contredit la maximalité de l'ensemble indépendant d'événements violés que l'on ré-échantillonne ; soit v a été violé suite à un ré-échantillonnage ultérieur à l'étape $k+1$, ce qui signifie que v appartenait déjà à $\tau_{k'+2}$, ce qui contredirait la maximalité de k' (Comme la profondeur de τ_{k+1} correspond à la profondeur de v, et que la profondeur de τ_{k+2} est d'au moins celle de v, puisque $v \in \tau_{k+2}$, on devrait nécessairement avoir que la profondeur des deux arbres τ_{k+1} et τ_{k+2} sont égales). On arrive donc a une contradiction.

Ceci permet de conclure que la profondeur de $\tau_C(t) = \tau_1$ vaut bien $j-1$. □

On est maintenant en mesure de prouver le théorème 6.4 :

Démonstration. Soit $Q(k)$ la probabilité que l'algorithme prenne au moins k étapes. Par le lemme 6.5 qui précède, un arbre témoin d'une profondeur $k-1$ doit donc apparaître dans son journal d'exécution C. Notons qu'un arbre de profondeur $k-1$ a au moins k sommets. Soit $\mathcal{T}_A(k)$ l'ensemble des arbres dans $\mathcal{T}_A$ qui ont au moins k sommets. On a :

$$Q(k) \leq \sum_{A \in \mathcal{A}} \sum_{\tau \in \mathcal{T}_A(k)} \mathbb{P}[\tau \text{ apparaît dans le journal } C]$$

$$\leq \sum_{A \in \mathcal{A}} \sum_{\tau \in \mathcal{T}_A(k)} \prod_{v \in V(\tau)} \mathbb{P}[[v]] \leq (1-\epsilon)^k \cdot \sum_{A \in \mathcal{A}} \sum_{\tau \in \mathcal{T}_A} \prod_{v \in V(\tau)} x'([v]),$$

Où la dernière inégalité est une des conséquences directes des hypothèses du théorème 6.4 tandis que les deux premières sont obtenues de manière analogue à ce qui a été

fait dans la démonstration en page 19. Ainsi on a :

$$Q(k) \leq (1-\epsilon)^k \cdot \sum_{A\in\mathcal{A}} \sum_{\tau\in\mathcal{T}_A(k)} \prod_{v\in V(\tau)} x'([v]) = (1-\epsilon)^k \cdot \sum_{A\in\mathcal{A}} \sum_{\tau\in\mathcal{T}_A(k)} \frac{x(A)}{1-x(A)} \cdot p_\tau$$

$$= (1-\epsilon)^k \cdot \sum_{A\in\mathcal{A}} \frac{x(A)}{1-x(A)} \sum_{\tau\in\mathcal{T}_A(k)} \cdot p_\tau \leq (1-\epsilon)^k \cdot \sum_{A\in\mathcal{A}} \frac{x(A)}{1-x(A)}. \qquad (7)$$

Pour simplifier l'écriture, on pose $\alpha(\mathcal{A}) = \sum_{A\in\mathcal{A}} \frac{x(A)}{1-x(A)}$, et $k_0 = 2 \cdot \frac{1}{\epsilon} \cdot \log \alpha(\mathcal{A})$.
Observons que :

$$(1-\epsilon)^{k_0} \cdot \alpha(\mathcal{A}) = (1-\epsilon)^{2\cdot\frac{1}{\epsilon}\cdot\log\alpha(\mathcal{A})} \cdot \alpha(\mathcal{A}).$$

Par des arguments similaires à ceux donnés lors de la preuve du corollaire 3.2 (4), on peut montrer que $(1-\epsilon)^{\frac{1}{\epsilon}} \leq e^{-1}$. Ainsi on se retrouve avec :

$$(1-\epsilon)^{k_0} \cdot \alpha(\mathcal{A}) \leq e^{-2\cdot\log\alpha(\mathcal{A})} \cdot \alpha(\mathcal{A}) = e^{-2\cdot\frac{\ln\alpha}{\ln 2}} \cdot \alpha(\mathcal{A}) = \alpha(\mathcal{A})^{-\frac{2}{\ln 2}} \cdot \alpha(\mathcal{A}) \leq (\alpha(\mathcal{A}))^{-1}.$$

Désignons par $Q'(k)$ la probabilité que l'algorithme prenne exactement k étapes. Clairement, $Q'(k) \leq Q(k)$.

Maintenant, si on calcule l'espérance du nombre d'étapes N_p que prendra l'algorithme, on obtient :

$$\mathbb{E}(N_p) = \sum_{k=0}^{\infty} k \cdot Q'(k) \leq \sum_{k=0}^{2\cdot k_0} k \cdot Q'(k) + \sum_{k=2\cdot k_0+1}^{\infty} k \cdot Q(k).$$

Or,

$$\sum_{k=2\cdot k_0+1}^{\infty} k \cdot Q(k) = \sum_{j=k_0+1}^{\infty} (j+k_0) \cdot Q(j+k_0) = \sum_{j=k_0+1}^{\infty} (j+k_0) \cdot (1-\epsilon)^j \cdot (1-\epsilon)^{k_0} \cdot \alpha(\mathcal{A})$$

$$\leq \sum_{j=k_0+1}^{\infty} (j+k_0) \cdot (1-\epsilon)^j \cdot (\alpha(\mathcal{A}))^{-1} \leq \sum_{j=k_0+1}^{\infty} (2j) \cdot (1-\epsilon)^j \cdot (\alpha(\mathcal{A}))^{-1}$$

$$= 2 \cdot (\alpha(\mathcal{A}))^{-1} \sum_{j=k_0+1}^{\infty} (1-\epsilon)^{j-1} \cdot j \cdot \epsilon \cdot \frac{1}{\epsilon} \cdot (1-\epsilon).$$

Observons que l'espérance $\sum_{j=0}^{\infty} j \cdot (1-\epsilon)^{j-1} \cdot \epsilon$ d'une variable aléatoire géométrique avec paramètre ϵ vaut $1/\epsilon$. Ainsi, on se retrouve avec l'inégalité suivante :

$$\sum_{k=2\cdot k_0+1}^{\infty} k \cdot Q'(k) \leq 2 \cdot (\alpha(\mathcal{A}))^{-1} \cdot \epsilon \cdot \frac{1}{\epsilon} \cdot (1-\epsilon) \cdot \frac{1}{\epsilon} = 2 \cdot (\alpha(\mathcal{A}))^{-1} \cdot (1-\epsilon) \cdot \frac{1}{\epsilon}.$$

Ainsi, $\sum_{k=2\cdot k_0+1}^{\infty} k \cdot Q'(k)$ est en $\mathcal{O}(\frac{1}{\epsilon})$ pour $\alpha(\mathcal{A}) > 1$ (nous nous intéressons aux cas où $\alpha(\mathcal{A})$ tend vers l'infini et ϵ tend vers 0).

Clairement,

$$\sum_{k=0}^{2\cdot k_0} k \cdot Q'(k) \leq 2 \cdot k_0 = 4 \cdot \frac{1}{\epsilon} \cdot \log \alpha(\mathcal{A}),$$

et donc :

$$\mathbb{E}(N_p) = \sum_{k=0}^{\infty} k \cdot Q'(k) = 4 \cdot \frac{1}{\epsilon} \cdot \log \alpha(\mathcal{A}) + \mathcal{O}\left(\frac{1}{\epsilon}\right)$$

Ceci nous permet donc de conclure que l'espérance $\mathbb{E}(N_p)$ est bien un nombre en $\mathcal{O}\left(\frac{1}{\epsilon} \cdot \log \alpha(\mathcal{A})\right)$ (pour $\alpha(\mathcal{A}) > 2$), et prouve donc le théorème 6.4. □

6.3 Une variante déterministe de l'algorithme

En s'appuyant sur l'algorithme "en parallèle", on pourra montrer qu'il existe une variante déterministe qui construit en un temps polynomial une assignation telle qu'aucun événement de $\mathcal{A}$ ne soit violé. Pour ce faire on aura besoin de quelques hypothèses supplémentaires. Ainsi, on aboutira au théorème suivant :

Théorème 6.6. *Soit $\mathcal{P} = \{P_1, P_2, ..., P_n\}$ un ensemble fini de variables aléatoires mutuellement indépendantes dans un espace de probabilité, chacun des ces variables prenant des valeurs dans un domaine fini D_i. Soir $\mathcal{A}$ un ensemble de m événements déterminés par ces variables. Considérons la taille du problème comme étant $s := m + n + \sum_{i=1}^{n} |D_i|$. Supposons qu'il existe un algorithme qui puisse calculer, pour chaque $A \in \mathcal{A}$ et chaque évaluation partielle $(v_i \in D_i)_{i \in I}$, $I \subset \{1, 2, ..., n\}$ la probabilité conditionnelle $\mathbb{P}[A | \forall i \in I : P_i = v_i]$ en temps polynomial en fonction de s. Supposons de plus que le degré maximal dans un graphe de dépendance $G_{\mathcal{A}}$ soit borné par une constante, c'est-à-dire qu'il existe une constante k telle que $\forall A \in \mathcal{A} : |\Gamma_{\mathcal{A}}(A)| \leq k$. S'il existe une constante $\epsilon > 0$ et une assignation de réels $x : \mathcal{A} \rightarrow]0, 1[$ telle que, pour tout A dans $\mathcal{A}$ on ait :*

$$\mathbb{P}[A] \leq (1 - \epsilon) \cdot x(A) \cdot \prod_{B \in \Gamma(A)} (1 - x(B)),$$

alors un algorithme déterministe peut trouver en un temps polynomial en fonction de s une évaluation des variables telle qu'aucun événement de $\mathcal{A}$ n'ait lieu.

Nous allons maintenant présenter dans les grandes lignes l'algorithme correspondant, ce qui consistue une esquisse de preuve pour ce théorème.

Pour rendre notre algorithme déterministe, on va remplacer les retirages aléatoires pour les valeurs prises par les variables à chaque ré-échantillonnage, par une table de valeurs. Soit $(v_i^{(j)} \in D_i)_{1 \leq i \leq n, j \in \mathbb{N}}$ les séquences de valeurs de ces variables. Ainsi, on suppose que le j-ième retirage d'une variable P_i de l'algorithme "en parallèle" retournera la valeur $v_i^{(j)}$. On dit qu'un arbre témoin τ est *consistant* avec les séquences $(v_i^{(j)} \in D_i)_{1 \leq i \leq n, j \in \mathbb{N}}$ si, lorsqu'on les substitue à la source aléatoire, le τ-check passe. On peut montrer à partir du résultat (7) de la preuve du théorème 6.4, que si les valeurs $v_i^{(j)}$ sont choisies aléatoirement, indépendamment et selon leur loi de probabilité, alors l'espérance du nombre d'arbres témoins comprenant au moins k sommets et qui sont consistants avec les séquences correspondantes, était bornée par $(1 - \epsilon)^k \cdot \sum_{A \in \mathcal{A}} \frac{x(A)}{1 - x(A)}$. De plus, si les poids $x(A)$ admettent une borne supérieure strictement plus petite que 1 (cas auquel on peut toujours se ramener à partir des hypothèses), on pourrait montrer que ce nombre est borné en $\mathcal{O}(m \cdot (1 - \epsilon)^k)$. Ceci

permettrait de montrer qu'il existe une constante c telle que l'espérance du nombre d'arbres témoins de taille au moins $c \cdot \log m$ qui sont consistants avec la table des séquences soit au plus $1/2$. Ainsi, avec une probabilité d'au moins $1/2$, aucun arbre témoin de taille au moins $c \cdot \log m$ n'est consistant avec la table. Puisqu'en ce cas, aucune variable ne peut être retirée plus de $c \cdot \log m$ fois, on peut se restreindre à la construction de séquences $(v_i^{(j)} \in D_i)_{1 \leq i \leq n, 0 \leq j \leq c \log m}$ telles qu'aucun "grand" arbre ne devienne consistant. De telles valeurs peuvent être calculées séquentiellement en utilisant la méthode des probabilités conditionnelles. Mais tout d'abord, nous devons nous assurer que le nombre d'arbres témoins que nous devrons considérer n'est pas trop important, d'où le lemme suivant :

Lemme 6.7. *Supposons que le degré maximum des sommets du graphe de dépendance des événements est borné par k comme dans les hypothèses du théorème 6.6. Soit $u \in \mathbb{N}$. S'il existe un arbre témoin de taille au moins u qui est consistant avec des séquences d'évaluation $(v_i^{(j)} \in D_i)_{1 \leq i \leq n, j \in \mathbb{N}}$ données, alors il existe aussi un arbre témoin de taille comprise entre u et $(k+1) \cdot u$ qui est consistant avec ces séquences.*

Démonstration. Nous allons procéder par l'absurde. Soit τ un contre-exemple de taille minimale, c'est-à-dire un arbre témoin de taille plus grande que $(k+1) \cdot u$, consistant avec une table de valeur telle qu'il n'existe aucun arbre consistant de taille entre u et $(k+1) \cdot u$. Par la borne concernant les degrés dans le graphe de dépendance, chaque sommet de τ a au plus $k+1$ enfants. Soient $w_1, w_2, ..., w_j$ avec $j \leq k+1$ les enfants immédiats de la racine. Maintenant on va construire j arbres distincts consistants avec la table comme suit : On parcourt τ de la manière usuelle niveau par niveau "bottom-up", en partant du plus bas niveau et en remontant jusqu'à la racine. On considère la liste d'événements correspondants ainsi obtenue comme un journal C' de longueur $|\tau|$ de l'algorithme séquentiel. On construit maintenant pour les j "avant-dernières" étapes de ré-échantillonnage (qui concernent donc les événements $[w_1], [w_2], ..., [w_n]$), les arbres témoins correspondant $\tau_i := \tau_{C'}(|\tau| - i)$ pour $i = 1, 2, ..., j$. Par construction, chaque τ_i est consistant avec la table de séquences. De plus, comme τ_i contient forcément au moins autant de sommets que le sous-arbre de τ enraciné en w_i, on doit avoir au moins un i tel que τ_i contient au moins $(|\tau| - 1)/(k+1)$ sommets. Puisque τ contient au moins $(k+1) \cdot u$ sommets, τ_i en contient forcément u. Ainsi, τ_i contredit forcément une des hypothèses : celle de la non-consistance des arbres témoins de taille entre u et $(k+1) \cdot u$), ou celle de la minimalité de τ comme contre-exemple. □

Nous pouvons maintenant présenter le fonctionnement de l'algorithme de façon intuitive :

On commence par énumérer tous les arbres témoins de taille entre $c \cdot \log m$ et $(k+1) \cdot c \cdot \log m$ dans une liste L. La borne sur le degré du graphe de dépendance nous garantit qu'ils sont en nombre polynomial. Maintenant, dans un ordre quelconque qui parcourt toutes les paires (i, j) avec $1 \leq i \leq n$ et $0 \leq j \leq c \cdot \log m$, on va calculer une à une les valeurs des $v_i^{(j)}$. Pour chaque paire (i, j), on va considérer chaque valeur possible de D_i que l'on pourrait assigner à $v_i^{(j)}$, puis calculer l'espérance conditionnelle du nombre d'arbres qui deviennent consistant avec la table partielle des valeurs que l'on a déjà choisies jusque là, et en considérant les valeurs

pas encore calculées comme aléatoires. Au début, quand aucune valeur n'a été calculée dans la table, l'espérance est inférieure à 1/2. Ainsi, à chaque pas, on peut clairement choisir une valeur pour la variable concernée qui préserve cette espérance conditionnelle en dessous de 1/2. Une fois que toutes les valeurs sont fixées, le nombre d'arbres témoins consistants est déterministe, et doit donc correspondre à un entier inférieur à 1/2. Ainsi, il n'y a plus aucun arbre consistant de taille au moins $c \cdot \log m$ avec la table des valeurs ainsi obtenue. On peut donc lancer l'algorithme en prenant cette table comme source aléatoire, et on est assuré qu'il termine en au plus $c \cdot \log m$ étapes.

Ceci permet bien d'aboutir à un temps polynomial pour l'algorithme ainsi construit.

7 Conclusion

Dans ce projet, nous avons eu un bon aperçu de ce qu'est la méthode probabiliste et de la force que celle-ci peut avoir pour montrer certains résultats. Nous avons notamment travaillé autour du lemme local de Lovász, dont les hypothèses peuvent paraître quelques peu abstraites et lourdes à vérifier, mais qui s'avère néanmoins très puissant et trouve des applications dans de nombreux problèmes concrets, dont celui du routage que nous avons abordé.

Nous avons terminé en étudiant une preuve constructive pour une variante de ce lemme. Ce résultat est d'autant plus puissant qu'il aboutit à un algorithme déterministe et en temps polynomial, capable de trouver une assignation de valeurs pour les variables correspondant aux événements qui nous intéressent, telle qu'aucun "mauvais événement" n'ait lieu.

Références

[1] N. Alon, J.H. Spencer et P. Erdős : *The Probabilistic Method*, 1991
(Chapitre 1 :*The basic method*, Prop. 1.1 p.3 et Prop. 3.1 p.8)

[2] N. Alon, J.H. Spencer et P. Erdős : *The Probabilistic Method*, 1991
(Chapitre 5 :*The local lemma*, Lemme 1.1 pp.53-55)

[3] N. Alon, J.H. Spencer et P. Erdős : *The Probabilistic Method*, 1991
(Chapitre 5 :*The local lemma*, Cor. 1.2 p.55)

[4] N. Alon, J.H. Spencer et P. Erdős : *The Probabilistic Method*, 1991
(Chapitre 5 :*The local lemma*, Prop. 2.1 p.56 et Prop. 3.1 p.57)

[5] N. Alon et J.H. Spencer : *The Probabilistic Method*, 2008
(Chapitre 5 : exercice 3 p.82)

[6] R.A. Moser et G.Tardos : *A constructive proof of the general Lovász Local Lemma*, mai 2009 (http ://arxiv.org/abs/0903.0544)

[7] C. Scheideler : *Universal Routing Strategies for Interconnection Networks*, 1998
(Chapitre 6 : *Offline Routing Protocols*)
(http ://www.springerlink.com/content/978-3-540-64505-4)