Méthode probabiliste et dérandomisation

François Schwarzentruber

7 avril 2021

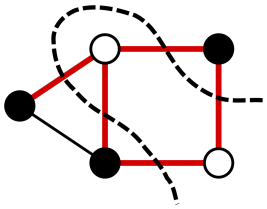
— La *méthode probabiliste* est l'art de démontrer l'existence d'objets par des arguments probabilistes.
— La *dérandomisation* consiste à construire un algorithme déterministe à partir d'un algorithme probabiliste, en gardant les même garanties.

1 MAXCUT

Définition 1 (taille d'une coupe) Soit $C = \{V_0, V_1\}$ une coupe dans un graphe non orienté. La taille de C, notée $\#C$, est le nombre d'arêtes allant d'un sommet de V_0 à un sommet de V_1.

Définition 2 (coupe maximale) Une coupe maximale est une coupe de taille maximale.

Exemple 3 (coupe maximale de taille 5)



Définition 4 **MAXCUT**
entrée : un graphe G non orienté
sortie : une coupe maximale

1.1 Algorithme naïf

Placer chaque sommet de manière aléatoire uniforme dans V_0 ou dans V_1.

1.2 Existence d'une solution

Théorème 5 Soit $G = (V, E)$. Il existe une coupe de G de taille au moins $\frac{|E|}{2}$.

Démonstration. Soit C la coupe calculée par l'algorithme naïf. On note, pour tout sommet u :

$$X_u = \begin{cases} 0 & \text{si } u \in V_0 \\ 1 & \text{si } u \in V_1. \end{cases}$$

$$\frac{\#C = \sum_{(u,v)\in E} 1_{X_u \neq X_v}}{\mathbb{E}(\#C) = \sum_{(u,v)\in E} \mathbb{E}(1_{X_u \neq X_v})} \qquad \mathbb{E}(1_{X_u \neq X_v}) = \mathbb{P}(X_u \neq X_v)$$

$$\frac{\mathbb{P}(X_u \neq X_v) = \mathbb{P}(X_u=0 \text{ et } X_v=1) + \mathbb{P}(X_u=1 \text{ et } X_v=0)}{\mathbb{P}(X_u \neq X_v) = \mathbb{P}(X_u=0)\mathbb{P}(X_v=1) + \mathbb{P}(X_u=1)\mathbb{P}(X_v=0)} \qquad \mathbb{P}(X_u \neq X_v) = \frac{1}{2}$$

$$\mathbb{E}(\#C) = \frac{|E|}{2}$$

Comme l'espérance de la taille de la coupe calculée est plus grande que $\frac{|E|}{2}$, il existe au moins une coupe de cette taille.

■

1.3 Algorithme d'approximation probabiliste en moyenne

Définition 6 Étant donné un problème d'optimisation, un algorithme probabiliste $\mathcal{A}$ est une approximation de facteur $\rho(n)$ *en moyenne* si pour toute entrée de taille n, $\mathcal{A}$ calcule une solution `sol` avec

$$\begin{cases} \frac{\mathbb{E}(\text{coût}(\texttt{sol}))}{\text{coût d'une solution optimale}} \leq \rho(n) & \text{si c'est un problème de minimisation ;} \\ \\ \frac{\mathbb{E}(\text{coût}(\texttt{sol}))}{\text{coût d'une solution optimale}} \geq \rho(n) & \text{si c'est un problème de maximisation.} \end{cases}$$

Proposition 7 L'algorithme probabiliste naïf pour le calcul d'une coupe est une $\frac{1}{2}$-approximation de **MAXCUT** en moyenne.

Démonstration.

$$\frac{\mathbb{E}(\#C) = \frac{|E|}{2} \qquad \text{coût d'une solution optimale} \leq |E|}{\frac{\mathbb{E}(\#C)}{\text{coût d'une solution optimale}} \geq \frac{1}{2}}$$

■

Proposition 8 La probabilité que l'algorithme naïf renvoie une coupe de taille $\geq \frac{|E|}{2}$ est d'au moins $\frac{1}{1+\frac{|E|}{2}}$. Autrement dit :

$$\mathbb{P}(\#C \geq \frac{|E|}{2}) \quad \geq \quad \frac{1}{1+\frac{|E|}{2}}.$$

Démonstration. Afin de minorer la probabilité $p := \mathbb{P}(\#C \geq \frac{|E|}{2})$, calculons $\mathbb{E}(\#C)$:

$$\begin{aligned}
\frac{|E|}{2} = \mathbb{E}(\#C) &= \sum_{i \leq |E|} i\ \mathbb{P}(\#C = i) && \text{Définition de l'espérance} \\
&= \sum_{i \leq \frac{|E|}{2}-1} i\ \mathbb{P}(\#C = i) \quad + \quad \sum_{i \geq \frac{|E|}{2}} i\ \mathbb{P}(\#C = i) && \text{Découpage de la somme} \\
&\leq (\frac{|E|}{2} - 1) \sum_{i \leq \frac{|E|}{2}-1} \mathbb{P}(\#C = i) \quad + \quad |E| \sum_{i \geq \frac{|E|}{2}} \mathbb{P}(\#C = i) && \text{Majoration des } i \\
&\leq (\frac{|E|}{2} - 1)\mathbb{P}(\#C < \frac{|E|}{2}) \quad + \quad |E|\ \mathbb{P}(\#C \geq \frac{|E|}{2}) && \text{Probabilité d'unions disjointes} \\
&= (\frac{|E|}{2} - 1)(1-p) \quad + \quad |E|\ p && \text{Renommage} \\
&= \frac{|E|}{2} - 1 \quad + \quad p(1 + \frac{|E|}{2}). && \text{Regroupement}
\end{aligned}$$

D'où la proposition.

■

Proposition 9 Le nombre moyen de répétitions de l'algorithme naïf pour trouver une coupe de taille au moins $\frac{|E|}{2}$ est $1 + \frac{|E|}{2}$.

Démonstration. Quand on répète un tirage de Bernouilli de probabilité q, alors l'espérance du nombre de répétitions T jusqu'à succès est $\frac{1}{q}$. En effet :

$$\begin{aligned}
\mathbb{E}(T) &= \sum_{i \geq 1} \mathbb{P}(T \geq i) && \text{Définition alternative de l'espérance} \\
&= \sum_{i \geq 1} (1-q)^{i-1} && \text{au moins } i-1 \text{ échecs} \\
&= \sum_{i \geq 0} (1-q)^{i} && \text{décalage de l'indice} \\
&= \frac{1}{1-(1-q)} && \text{série convergente} \\
&= \frac{1}{q} = 1 + \frac{E}{2} && \text{car ici } q = \frac{1}{1+\frac{|E|}{2}}
\end{aligned}$$

■

Remarque 10 On obtient donc un algorithme de type Las Vegas pour le calcul d'une coupe contenant au moins la moitié des arêtes.

2 Dérandomisation

Question 11 Les algorithmes probabilistes sont-ils nécessaires ?

2.1 Méthode des probabilités conditionnelles

Cette approche est dûe à Erdös et Selfridge [ES73]. Illustrons la sur l'algorithme naïf probabiliste pour **MAXCUT**. On note C la coupe calculée par l'algorithme probabiliste, et $\#C$ sa taille. Il y a une exécution de l'algo probabiliste naïf avec $\#C \geq \frac{|E|}{2}$.

Idée de l'algorithme déterministe.
Placer de manière déterministe les sommets dans X ou Y de façon à conserver l'invariant

$$\mathbb{E}(\#C \mid \text{choix déterministes déjà faits}) \geq \frac{|E|}{2}.$$

Les sommets sont notés $V = \{1, \ldots, n\}$. On note, pour tout sommet u :

$$X_u = \begin{cases} 0 & \text{si } u \in X \\ 1 & \text{si } u \in Y. \end{cases}$$

Plus formellement, l'algorithme déterministe va choisir $j_1, \ldots, j_n \in \{0,1\}$ de façon à avoir pour tout $i \in \{0, \ldots, n\}$:

$$\mathbb{E}(\#C \mid X_1{=}j_1, \ldots, X_i{=}j_i) \geq \frac{|E|}{2}$$

Notation 12 On note $X_{1..i} = j_{1..i}$ au lieu de $X_1{=}j_1, \ldots, X_i{=}j_i$.

Initialisation. Au début, $\mathbb{E}(\#C) \geq \frac{|E|}{2}$ donc tout va bien.
Induction. Supposons que nous ayons choisi $j_1, \ldots, j_{i-1} \in \{0,1\}$ avec

$$\mathbb{E}(\#C \mid X_{1..i-1}{=}j_{1..i-1}) \geq \frac{|E|}{2}.$$

Or

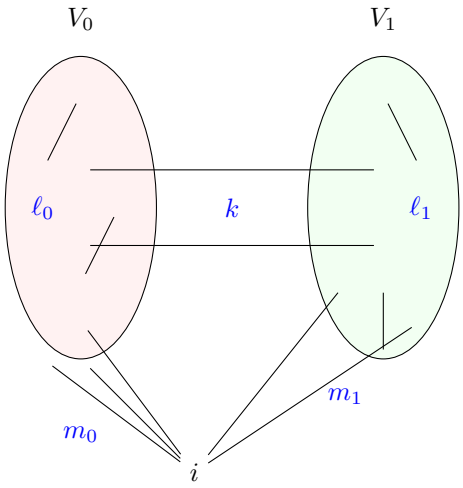
$$\mathbb{E}(\#C \mid X_{1..i-1}{=}j_{1..i-1}) = \frac{1}{2}\mathbb{E}(\#C \mid X_{1..i-1}{=}j_{1..i-1}, X_i{=}0) + \frac{1}{2}\mathbb{E}(\#C \mid X_{1..i-1}{=}j_{1..i-1}, X_i{=}1)$$

et donc soit $\mathbb{E}(\#C \mid X_{1..i-1}{=}j_{1..i-1}, X_i{=}0) \geq \frac{|E|}{2}$ ou $\mathbb{E}(\#C \mid X_{1..i-1}{=}j_{1..i-1}, X_i{=}1) \geq \frac{|E|}{2}$.

L'algorithme déterministe consiste à choisir $j_i \in \{0,1\}$ avec $\mathbb{E}(\#C \mid X_{1..i}{=}j_{1..i}) \geq \frac{|E|}{2}$.

Fait 13

$$\mathbb{E}(\#C \mid X_{1..i-1}{=}j_{1..i-1}, X_i{=}0) = k + m_1 + \frac{\alpha}{2} \quad \text{et} \quad \mathbb{E}(\#C \mid X_{1..i-1}{=}j_{1..i-1}, X_i{=}1) = k + m_0 + \frac{\alpha}{2}$$



avec
- deux ensembles $V_0 := \{j \in \{1,\dots,i-1\} \mid X_j = 0\}$ et $V_1 := \{j \in \{1,\dots,i-1\} \mid X_j = 1\}$;
- ℓ_0 arêtes entre sommets de V_0 ;
- ℓ_1 arêtes entre sommets de V_1 ;
- k arêtes entre un sommet de V_0 et un de V_1 ;
- m_0 arêtes entre sommets de V_0 et le sommet i ;
- m_1 arêtes entre sommets de V_1 et le sommet $i.$;
- $\alpha = |E| - m_0 - m_1 - k - \ell_0 - \ell_1$ arêtes touchant un sommet de $\{i+1,\dots,n\}$.

DÉMONSTRATION. Les arêtes entre sommets de la même patate ne comptent pas. Les arêtes entre les deux patates comptent chacunes pour 1. Enfin, seules les arêtes reliant i à la patate dans laquelle i ne va pas être comptent. Enfin, les arêtes touchant un sommet de $\{i+1,\dots,n\}$ ne sont pas encore choisies donc on a une contribution de $\frac{1}{2}$. ■

2.2 Algorithme déterministe

entrée : un graphe non orienté $G = (V, E)$
sortie : une coupe (X, Y) de taille plus grande que $\frac{|E|}{2}$
fonction gloutonMaxCutApprox($G = (V, E)$)
    $X := \emptyset$
    $Y := \emptyset$
    pour tout sommet $v \in V$ **faire**
        si v a plus de voisins dans X que dans Y **alors**
            $Y := Y \cup \{v\}$
        sinon
            $X := X \cup \{v\}$

2.3 État de l'art

Le meilleur ratio pour **MAXCUT** est environ 1/0.878, voir [GW95]. C'était la première fois que la programmation semidéfinie est utilisée pour concevoir un algo d'approximation. C'est une forme d'optimisation avec des expressions quadratiques. Une instance se transforme en :

$$\begin{array}{l}\text{maximiser } \sum_{(i,j)\in E} \frac{1-v_i v_j}{2} \\ \{\ v_i \in \{-1,1\}\end{array}$$

où $v_i = -1$ signifie que $i \in X$ et $v_i = 1$ que $i \in Y$. Puis ils font de la relaxation, et algo probabiliste (on fera pareil après pour **MAXSAT**). La borne 1/0.878 est optimale si la *conjecture des jeux uniques* est vraie [KKMO07].

3 Théorème d'Adleman

Théorème 14 (Théorème d'Adleman) Tout problème de décision L dans RP peut se dérandomiser de manière non-uniforme : il existe un polynôme P, et pour tout entier n, il existe un algorithme déterministe qui décide l'ensemble des instances positives de taille n en temps $P(n)$.

DÉMONSTRATION. Sans perte de généralité, nous supposons que tous les mots sont sur $\Sigma = \{0, 1\}$. Soit $L \in RP$. Il existe un algorithme A en temps polynomial $Q(n)$ tel que pour tout instance x :
- Si $x \in L$ alors $\mathbb{P}(A(x)$ renvoie 'oui'$) \geq \frac{1}{2}$;
- Si $x \notin L$ alors $\mathbb{P}(A(x)$ renvoie 'non'$) = 1$.

Considérons plutôt que l'algorithme A est déterministe, et prend aussi un mot aléatoire R de longueur $Q(n)$ tiré uniformément. On a alors pour toute instance x :
- Si $x \in L$ alors $\mathbb{P}(A(x, R)$ renvoie 'oui'$) \geq \frac{1}{2}$;
- Si $x \notin L$ alors $\mathbb{P}(A(x, R)$ renvoie 'non'$) = 1$.

Fixons n. Considérons maintenant l'algorithme B qui prend en entrée x de taille n ainsi que $n+1$ mots $r_1, \ldots, r_{n+1}$ chacun de longueur $Q(n)$:

fonction $B(x, r_1, \ldots, r_{n+1})$
  pour $i := 1..n+1$ **faire**
    si $A(x, R_i)$ renvoie 'oui' **alors**
      renvoyer 'oui'
  renvoyer 'non'

Si x de taille n avec $x \notin L$, alors pour tout $r_1, \ldots, r_{n+1} \in \{0,1\}^{Q(n)}$, $B(x, r_1, \ldots, r_{n+1})$ renvoie 'non'.

Notons $L^{=n}$ l'ensemble des instances positives de L taille n, i.e. $L \cap \{0,1\}^n$. Soit $R_1, \ldots, R_{n+1}$ des variables aléatoires indépendantes qui donnent un mot dans $\{0,1\}^{Q(n)}$ de manière uniforme.

Fait 15 Soit $x \in L^{=n}$ on a :

$$\mathbb{P}(B(x, R_1, \ldots, R_{n+1}) \text{ renvoie 'non'}) \leq \frac{1}{2^{n+1}}$$

Démonstration.

$$\begin{aligned}
\mathbb{P}(B(x, R_1, \ldots, R_{n+1}) \text{ renvoie 'non'}) &= \mathbb{P}(A(x, R_1) \text{ et } \ldots \text{ et } A(x, R_{n+1}) \text{ renvoient non}) \\
&= \mathbb{P}(A(x, R_1) \text{ renvoie non}) \times \cdots \times \mathbb{P}(A(x, R_{n+1}) \text{ renvoient non}) \\
&\qquad \text{car les } R_i \text{ sont des variables indépendantes} \\
&\leq \frac{1}{2} \times \cdots \times \frac{1}{2} = \frac{1}{2^{n+1}}.
\end{aligned}$$

■

Fait 16

$$\mathbb{P}(\text{pour tout } x \in L^{=n},\ B(x, R_1, \ldots, R_{n+1}) \text{ renvoie 'oui'}) > 0.$$

Démonstration.

$$\begin{aligned}
\mathbb{P}(\text{il existe } x \in L^{=n},\ B(x, R_1, \ldots, R_{n+1}) \text{ renvoie 'non'}) &= \mathbb{P}\Big(\bigcup_{x \in L^{=n}} \{B(x, R_1, \ldots, R_{n+1}) \text{ renvoie 'non'}\}\Big) \\
&\leq \sum_{x \in L^{=n}} \mathbb{P}(B(x, R_1, \ldots, R_{n+1}) \text{ renvoie 'non'}) \\
&\leq \sum_{x \in L^{=n}} \frac{1}{2^{n+1}} \text{ par le Fait 15} \\
&\leq \frac{2^n}{2^{n+1}} = \frac{1}{2} \text{ car } |L^{=n}| \leq 2^n.
\end{aligned}$$

■

Par la méthode probabiliste, il existe des mots $r_1, \ldots, r_{n+1}$ de longueur $Q(n)$ tel que pour tout $x \in L^{=n}$, $B(x, r_1, \ldots, r_{n+1})$ renvoie 'oui'.

L'algorithme déterministe qui décide les instances positives de taille n est $B(\_, r_1, \ldots, r_{n+1})$.

■

Définition 17 P/*poly* est la classe des problèmes de décision L qui admette un algorithme A en temps polynomial, et une suite de mots $\alpha_0, \alpha_1, \ldots$ de longueur $poly(n)$ tels que pour toute instance de taille n,

$$x \in L \text{ ssi } A(x, \alpha_n) \text{ renvoie 'oui'.}$$

Théorème 18 (théorème d'Adleman) RP $\subseteq$ P/*poly*.

Références

[ES73] Paul Erdoes and JL Selfridge. On a combinatorial game. *Journal of Combinatorial Theory, Series A*, 14(3) :298–301, 1973.

[GW95] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6) :1115–1145, 1995.

[KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps ? *SIAM Journal on Computing*, 37(1) :319–357, 2007.