INF554: Utilisation de l'aléatoire en algorithmique	**Automne 2014**
Cours 4 — 6 octobre	
Enseignant : Frédéric Magniez	*Rédacteur : Daniel Lévy*

4.1 Préliminaires

4.1.1 Algorithme d'approximation

Problème d'optimisation :

- **Input** : muni d'une fonction cost
- **Output** : qui minimise ou maximise la fonction cost. Cette valeur est appelée *valeur optimum* (OPT).

Exemple 1. *(Cardinality) Vertex Cover*
- ***Input*** *: Graphe non-orienté $G = (V, E)$.*
- ***Output*** *: Ensemble $S \subseteq V$ de plus petite taille tel que $\forall e \in E, e \bigcap S = \varnothing$*

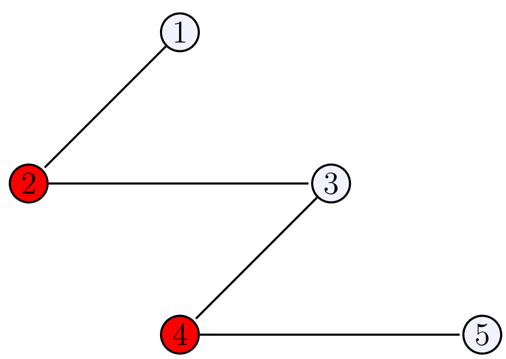

Figure 4.1. Exemple de vertex cover symbolisé en rouge.

Pour un graphe étoilé par rapport à $u \in V$, $S = \{u\}$ convient et $|\mathsf{Opt}| = 1$.
Pour un graphe complet, on a : $|\mathsf{Opt}| = |V| - 1, \forall v \in V, V - \{v\}$ convient.
Dans le cas d'une ligne de taille $2n$, $|\mathsf{Opt}| = n$.

Définition 4.1. *Soit $\alpha \geq 1$. A est un algorithme qui calcule une $\alpha-$approximation d'un problème P si pour toute entrée x :*
- *Max : $\mathrm{cost}\,(A(x)) \leq \mathrm{cost}\,(\mathsf{Opt}(x)) \leq \alpha\mathrm{cost}\,(A(x))$*
- *Min : $\mathrm{cost}\,(\mathsf{Opt}(x)) \leq \mathrm{cost}\,(A(x)) \leq \alpha\mathrm{cost}\,(\mathsf{Opt}(x))$*

On peut généraliser aux algorithmes probabilistes Las Vegas ou Monte Carlo. α est appelé le facteur (ou ratio) d'approximation.

Retour sur l'exemple

Algorithme. Calculer un couplage M maximal pour l'inclusion (algorithme glouton linéaire). Renvoyer $S = V(M)$ où $V(M)$ est l'ensemble des sommets composant les arêtes de M.

Proposition 4.1.2. *Cette algorithme est une 2-approximation de Vertex Cover.*

Preuve: S est un set cover par maximalité de M. Montrons que : $|S| \leq 2|\mathsf{Opt}|$

On montre en fait que $|\mathsf{Opt}| \leq |M|$. En effet, comme Opt est vertex cover, à toute arête de M correspond un noeud différent de Opt. De plus, $2|M| = |S|$ nous donne bien le résultat. □

4.1.3 La méthode probabiliste

Le principe de cette méthode est que, afin de montrer que quelque chose existe, on montre qu'il existe avec une probabilité non nulle.

Théorème 4.2. *Si* $\binom{n}{k} < 2^{\binom{k}{2}-1}$, *alors il est possible de colorier les arêtes du graphe complet* K_n *à* n *sommets avec deux couleurs, tel qu'il n'y ait pas de clique* K_k *monochrome.*

Exemple 2. *Si on fait le calcul avec* $n = 1000$ *et* $k = 20$, *on a le résultat.*

Preuve: On définit une fonction Color : $\{1, 2, \dots, n\} \to \{0, 1\}$ de manière probabiliste par chaque arête de $e \in \binom{n}{2}$ est coloriée uniformément au hasard.

Numérotons toutes les k-cliques de 1 à $\binom{n}{k}$.
$A_i := \mathbf{Pr}\left[\text{La i-ème clique est monochromatique}\right] = \frac{1}{2}^{\binom{k}{2}-1}$
On a : $\mathbf{Pr}\left[\text{Aucune clique est monochromatique}\right] = 1 - \mathbf{Pr}\left[\exists i, A_i\right] = 1 - \mathbf{Pr}\left[\bigcup_i A_i\right]$.

Or, $\mathbf{Pr}\left[\bigcup_i A_i\right] \leq \sum_i \mathbf{Pr}(A_i) = \binom{n}{k} \frac{1}{2}^{\binom{k}{2}-1}$.

D'après l'hypothèse sur n et k, on a bien $\mathbf{Pr}\left[\text{Aucune clique est monochromatique}\right] > 0$.

□

4.2 Argument par linéarité de l'espérance

Principe 1. *Soit* S *un espace probabiliste,* X *une variable aléatoire définie sur* S *telle que* $\mathbf{E}(X) = \mu$. *Alors,* $\mathbf{Pr}(X \geq \mu) > 0$ *et* $\mathbf{Pr}(X \leq \mu) > 0$ *et il existe une instance de* S *pour laquelle* $X \geq \mu$.

Définition 4.3. $G = (V, E)$ *un graphe non-orienté. Une cut de* G *est une partition de* V *en deux parties* $(S, V \setminus S)$.

Problème :

- **Input** : $G = (V, E)$
- **Output** : Cut maximal

Résoudre ce problème est utile en physique statistique, en circuit design (connaître le nombre minimal de superpositions de fils par exemple...).

Théorème 4.4. *Il existe une cut S de taille plus grande que* $\dfrac{|E|}{2} \geq \dfrac{|\mathsf{Opt}|}{2}$.

Preuve: Pour chaque sommet v, on décide uniformément, aléatoirement et indépendamment si $v \in S$. Pour chaque arête, on définit $X_e = \mathbf{1}_{e \in S \times V \setminus S}$.
Donc $\text{Taille}(S) = \sum_{e \in E} X_e = \text{Cut}(S)$. On a alors,

$$\mathbf{E}\left(\text{Cut}(S)\right) = \sum_{e \in E} \mathbf{E}(X_e) = \sum_{e=\{u,v\} \in E} \mathbf{Pr}\left((u \in S, v \notin S)\text{ou}(u \notin S, v \in S)\right) = \frac{|E|}{2}$$

□

D'après le principe énoncé au début, on dispose de S tel que : $\text{cut}(S) \geq \dfrac{|E|}{2}$.

Corollaire 4.5. *Il existe un algorithme probabiliste qui calcule en moyenne une 2-approximation de cut maximal en temps* $O\left(|V|\right)$.

Corollaire 4.6. *Il existe un algorithme déterministe qui calcule en moyenne une 2-approximation de cut maximal en temps* $O\left(|V||E|\right)$.

Preuve: La preuve du premier corollaire est facile, on tire simplement au hasard chaque sommet pour S.
Soit $V = \{1, 2, \cdots, n\}$.

$$\mathbf{E}\left(\text{cut}(S)\right) = \frac{1}{2}\mathbf{E}_{S|1 \in S}\left(\text{cut}(S)\right) + \frac{1}{2}\mathbf{E}_{S|1 \notin S}\left(\text{cut}(S)\right) \geq \frac{|E|}{2}$$

Ainsi, nécessairement l'un des deux termes de la somme est forcément supérieur à $\dfrac{|E|}{2}$
De manière générale, on sait calculer $\mathbf{E}\left(\text{cut}(S)|A \subset S, B \bigcap S = \varnothing\right) = \sum_{e \in E} \underbrace{\mathbf{E}\left(X_e|A \subset S, B \bigcap S = \varnothing\right)}_{=0,1\text{ou}\frac{1}{2}}$

La dérandomization consiste à calculer successivement les espérances conditionnelles (conditionnées aux choix préalablement faits), en s'assurant de toujours rester supérieur à l'espérance initiale. □

Problème du Max-SAT

- **Input** : $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$
- **Output** : $a \in \{0,1\}^n$ qui maximise le nombre de clauses satisfaites

Théorème 4.7. *Max 2SAT est NP-difficile.*

Théorème 4.8. *Il existe un algorithme probabiliste qui calcule une 2-approximation de Max SAT en moyenne.*

Lemme 4.9. *Pour une clause C à k variables (non-dégénérée), on a $\mathbf{Pr}(C(a) = 1) = 1 - 2^{-k}$.*

Preuve (du lemme): Soit $C = X_{i_1} \vee X_{i_2} \vee \cdots \vee X_{i_k}$. Cette clause est satisfaite si sa négation ne l'est pas. Comme les i_j sont distincts (C est non-dégénérée), on dénombre un seul n-uplet qui convient, d'où le résultat. □

Preuve (du théorème): On construit un algorithme qui renvoie $a \in \{0,1\}^n$ uniformément choisi au hasard. On appelle Z_C la variable aléatoire définie par $Z_C = \mathbf{1}_{C(a)=1}$. Le nombre de clauses satisfaites par s'écrit alors $A = \sum\limits_{C \in \varphi} Z_C$.

$$\mathbf{E}\left(\sum_{C \in \varphi} Z_C\right) = \sum_{C \in \varphi} \mathbf{E}(Z_C) \geq \underbrace{\frac{1}{2}}_{\text{cas } k=1} \times n \geq \frac{n}{2}$$

□

Observation 1. *Si chaque clause a exactement k variable, on a une $\dfrac{1}{1 - 2^{-k}}$-approximation.*

Observation 2. *De la même façon que précédemment on peut dérandomizer.*

4.3 Echantillonage et modification

Max indépendant set

Définition 4.10. *$G = (V, E)$, $S \subset V$ est indépendant si $E \bigcap (S \times S) = \varnothing$.*

- **Input** : G
- **Output** : Ensemble indépendant S de taille maximale

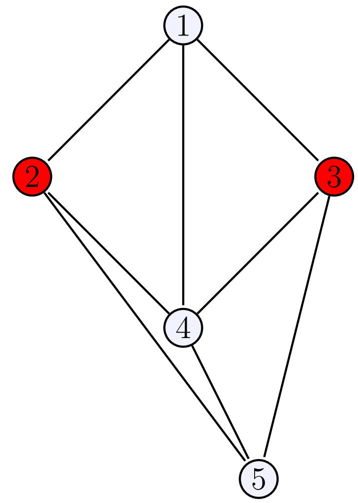

Figure 4.2. Exemple de set indépendant symbolisé en rouge.

Algorithme. $n = |V|, m = |E|$.

1. $d = \dfrac{2m}{n}$
2. Pour chaque noeud $v \in V$. Supprimer v avec une probabilité $1 - \frac{1}{d}$ (et toutes les arêtes incidentes).
3. Pour chaque arête restante, la supprimer ansi qu'un de ses sommets choisis aléatoirement.
4. Renvoyez S l'ensemble des sommets restants.

Théorème 4.11. *S est un ensemble indépendant de G, $\mathbf{E}(S) \geq \dfrac{n^2}{2m}$.*

Preuve: $X = \#\{\text{noeuds survivants après 1.}\}$. $\mathbf{E}(X) = \dfrac{n}{d} = \dfrac{n^2}{2m}$.

$Y = \#\{\text{arêtes survivantes après 1.}\}$. $\mathbf{E}(Y) = \dfrac{m}{d^2} = \dfrac{n^2}{4m}$.

$|S| \geq |X| - |Y| = \dfrac{n^2}{4m}$. $\square$