MPI: Proba & Approx

Cours

- Algorithme déterministe, de Las Vegas, Monte Carlos.
- Problème de décision, approximation, instance positive, instance négative.
- Fonction de cout, algorithme d'approximation.

Question de Cours

- Qu'est-ce qu'un algorithme de Monte-Carlo et de Las vegas ?
- Comment transformer un problème d'approximation en un problème de décision ?

Petites Questions

- Soit P un problème de minimisation et $\overline{P}$ le problème de décision associé. On suppose que $\overline{P}$
- Quelle est la probabilité que la fonction suivante termine ? `random()` renvoie un flottant aléatoire de manière uniforme dans $[0; 1[$

```
void f() {
  if (random() < 0.5f) {f();f();}
  return;
}
```

A ajouter

- Polya's Urn, Exchangeable random variables
- Chinese restaurant process
- Indian bufffet process
- Moris approximate counting algorithm https://en.wikipedia.org/wiki/Approximate_counting_algorithm (count to n with $\log \log n$ bits)

Espérence pour tout choisir

On choisi uniformément des entiers dans $\{1, ..., n\}$ avec remis j'usqu'à ce que tous les entiers ont été choisi. On note X la variable aléatoire du temps d'exécution. Montrer que $\mathbb{E}[X] = n \ln n + o(n \ln n)$

Le programme de Von Neumann

Soit `bool random(float p);` une fonction qui génère le booléen `true` avec probabilité p et le booléen `false` avec probabilité $1-p$.

1. Soit $0 < p < 1$. Montrer que le code suivant renvoie `true` avec probabilité $1/2$:

```
bool von_neumann(float p) {
  bool a = random(p);
  bool b = random(p);
  if (a!= b) { return a; }
  if (a == b) { return von_neumann(p);};
};
```

2. Que veux l'espérence du temps d'exécution en fonction de p ?

2-approximation du circuit eulérien (TSP)

Soit $G = (V, E)$ un graphe non orienté pondéré par $w : E \to \mathbb{R}_+^*$. On note pour $c := (c_i)_{i \leq n}$ un chemin $w(c) = \sum_{i<n} w((c_i, c_{i+1}))$ son poids. Similairement, on définit $w(T)$ le poids d'un arbre T sur G comme la somme du poids de toutes ses arêtes. Un circuit est un cycle où l'on a le droit de repasser sur des sommets déjà vu.

On cherche un circuit passant par tous les sommets au moins une fois et pouvant repasser sur lui-même tel que le poids de tout le chemin soit minimal. On note w^* le poids du circuit de poids minimal.

Question 1 On suppose que tous les aretes du graphe sont de poids différents. Montrer que le plus petit arbre couvrant est unique.

Question 2 Montrer que si T est arbre couvrant de poids minimal, alors $w(T) < w^*$

Question 3 En déduire via un parcours de cet arbre un circuit repassant sur lui-même c de poids $w(c) < 2w^*$

Question 4 Proposer un algorithme pour calculer un tel arbre. Quel est sa complexité ?

3/2-approx de TSP

On considère le problème du voyageur de commerce (TSP) :

- **Entrée** Un graphe $G = (V, E)$ avec $d : V^2 \longrightarrow \mathbb{R}_+$ une distance (respecte les axiomes de la distance).
- **Sortie** Le poids minimum d'un cycle passant par tous les sommets.

On note $w(C)$ pour C un chemin (ou arbre) son poids. On pose T^* un arbre couvrant de G' de poids minimal et I l'ensemble des sommets impair de T^*. On note w^{OPT} le poids optimal d'une instance du problème.

Question 1 Montrer que $w(T^*) < w^{\mathrm{OPT}}$. En déduire une 2-approximation du problème en utilisant un parcours de T^*

Question 2 Montrer que $|I|$ est pair et en déduire que G' restraint à I possède un couplage de poids minimal que l'on notera M.

Question 3 Montrer que si G est un graphe connexe tel que $\forall v \in V, \deg v \in 2\mathbb{N}$, alors G possède un cycle eulérien (passant par toutes les aretes). En déduire que le multigraphe $T^* + M$ possède un cycle eulérien que l'on notera h.

Question 4 Montrer que le problème du voyageur de commerce admet une $\frac{3}{2}$-approximation.

C-approx du couplages maximum[1]

Pour M un couplage, on note $S(M)$ les sommets saturé de M.

On considère l'algorithme suivant:

Entrée: G un graphe
$M \leftarrow$ un couplage maximal de G
tant que $\exists \{u', u\}, \{u, v\}, \{v, v'\} \in E$ avec $\{u, v\} \in M$ et $u', v' \notin S(M)$:
$\quad M \longleftarrow (M \setminus \{u, v\{\}\}) \cup \{\{u', u\}, \{v, v'\}\}$
fin tant que
retourner M

1. Quelle est la complexité de l'algorithme ?
2. Montrer que l'algorithme est une C-approx du couplage maximum pour un certain C que l'on déterminera.

On change la boucle pour chercher un chemin alternant P de longueur $2t + 1$, et si on en trouve un on effectue $M \longleftarrow (M \setminus P) \cup (P \setminus M)$. L'algorithme donné correspond donc au cas $t = 1$.

3. Donner le facteur d'approximation en fonction de t, et en déduire un *PTAS*, c'est-à-dire que pour tout $\varepsilon > 0$ on a un algorithme polynomial qui renvoie une $(1 - \varepsilon)$-approximation. On déterminera exactement la complexité en fonction de ε.

[1] TD11 L3 ENS Lyon Algo 1

Vertex cover

Soit $G = (V, E)$ un graphe. On dit que $S \subseteq V$ est *une couverture* si pour toute arête $e \in E$, au moins une des extrémitées de e est dans S. On s'intéresse au problème VERTEX-COVER suivant:

- **Entrée:** $G = (V, E)$ un graphe
- **Sortie:** Une couverture C de taille minimale en cadinal

1. Donner la version problème de décision de VERTEX-COVER.

On considère un algorithme glouton qui, tant qu'il reste des arêtes au graphe, en choisit une et retire les deux sommets du graphe. On note C les arêtes choisies la fin.

2. Montrer que si S est une couverture, alors $|S| \leq 2|C|$
3. En déduire une 2 approximation de VERTEX-COVER

On considère l'algorithme qui, tant que le graphe possède encore des arêtes, prend un sommet de degré non nul au hasard uniformément, le choisi et retire toutes les arêtes relié à ce sommet.

4. Quel est ce type d'algorithme ? Soit $G = (V, E)$ un graphe, on note V_f la variable aléatoire de l'ensemble choisi à la fin de l'exécution de l'algorithme sur G. Montrer que

$$\mathbb{E}\big[|V_f|\big] = \sum_{v \in V} \frac{\deg v}{\deg v + 1}$$

Set cover

On considère le problème SET-COVER suivant:

Entrée: $n, m \in \mathbb{N}, U_1, ..., U_n \subseteq [\![1; m]\!]$
Sortie: Le plus petit I en cardinal tel que $\bigcup_{i \in I} U_i = [\![1; m]\!]$

Dans toute cette colle on supposera que toutes les entrées sont telles que $\bigcup_{i \leq n} U_i = [\![1; m]\!]$

Question 1 Résoudre le problème pour les instances de la forme $n \in \mathbb{N}; m := n + 1; U_i = \{i, i + 1\}$

Question 2 Donner une $\log(m)$-approximation de SET-COVER. *Ind: On fera un algorithme glouton et on pourra utiliser le fait que* $\frac{1}{2} + ... + \frac{1}{n} \leq \log n$

On considère maintenant que la sortie doit être un I qui maximise le nombre de $x \in [\![1; m]\!]$ tel que x appartienne à un nombre impair d'ensembles de $(U_i)_{i \in I}$

Question 3 Soit $(U_i)_{i \leq n}$ une instance du problème. On choisi avec probabilité $\frac{1}{2}$ chaque U_i. On note N la variable aléatoire du nombre de $x \in [\![1; m]\!]$ appartennant à un nombre impair d'ensembles choisi.

Question 4 Montrer que $\mathbb{E}[N] = \frac{m}{2}$

Question 5 En déduire un algorithme un algorithme de Las Vegas qui est une 2-approximation. Existe t'il une 2-approximation déterministe ?

2-approx de Bin-Packing[2]

On s'intéresse au problème `BIN-PACKING` suivant :

- **Entrée:** Des objets de tailles $t_1, ..., t_n \in]0, 1]$
- **Sortie** Le plus petit nombre de boites $n \in \mathbb{N}$ de capacité 1 nécéssaire pour ranger tout les objets.

On note OPT le nombre minimal de boîtes nécessaires.

On étudie l'algorithme `Next-Fit` suivant :

$N, T \longleftarrow 0$
Pour i allant de 1 à n:
    Si $T + t_i \leq 1$ **alors:**
        $T \longleftarrow T + t_i$
    Sinon:
        $T \longleftarrow t_i$
        $N \longleftarrow N + 1$
Retourner N

1. Appliquer `Next-Fit` à la suite $(0.4, 0.7, 0.3, 0.6, 0.5, 0.5)$. Que vaux N à la fin?
2. Montrer que si `Next-Fit` ouvre une nouvelle boîte, alors la boîte précédente et la nouvelle boîte ont ensemble une charge strictement supérieure à 1.
3. On suppose que `Next-Fit` utilise k boites. Montrer que :

$$\text{OPT} \geq \sum_{i=1}^{n} t_i > \frac{k-1}{2}.$$

4. En déduire que `Next-Fit` est une 2-approximation.
5. Est-ce que `Next-Fit` est meilleur qu'une 2-approximation?

[2] CCINP 2025

$\log(n)$-approx de MLST

Un *multi-graphe* est un graphe dans lequel on peut avoir plusieurs fois une arête entre 2 sommets. On considère le problème MIN-COLOR-TREE suivant:

- **Entrée:** $G = (S, A)$ un multigraphe et $c : A \longrightarrow \mathbb{N}$ une coloration des arrettes.
- **Sortie:** Un arbre couvrant $T = (S, A')$ de G qui minimise $\mathrm{Card}(\{c(e) : e \in A'\})$

On note n le nombre de sommets et m le nombre d'arêtes.

1. Donner la version problème de décision, et montrer qu'elle est NP.
2. Montrer que si toutes les arêtes sont de couleur différentes alors il existe un algorithme polynomial.

On considère l'algorithme glouton suivant. Il initialise une structure union-find avec une classe par sommet et, a chaque étape, on choisit parmi les couleurs non encore choisies une couleur qui minimise le nombre de composantes après ajout de toutes les arêtes de cette couleur. Une fois que toutes les classes sont regroupées, alors il retourne un arbre couvrant des arêtes des couleurs choisies.

5. Montrer que l'algorithme renvoie bien une solution.

On note q le nombre de couleurs de la solution optimale et p_i le nombre de classes de la structure union-find à l'étape i de l'algorithme.

6. Montrer que

$$p_{i+1} - 1 \leq \left(1 - \frac{1}{q}\right)(p_i - 1)$$

7. En déduire une $\log(n)$-approximation.

Arbres ternaires complets (+Jeux)[3]

Un arbre T est dit ternaire complet si chaque noeud interne possède exactement 3 fils et que les feuilles sont toutes à la meme profondeur. On note $\mathcal{F}(T)$ les feuilles de T et $\mathcal{N}(T)$ les noeuds de T (incluant les feuilles).

On dispose d'une valuation $\sigma : \mathcal{F}(T) \longrightarrow \{V, F\}$ qui associe à chaque feuille une valeur booléene. On étend σ à $\mathcal{N}(T)$ en posant $\sigma(u) = b$ pour tout $u \in \mathcal{N}(T)$ ayant deux fils u_1, u_2 tels que $\sigma(u_1) = \sigma(u_2) = b$. On cherche à évaluer la valeur de σ à la racine.

1. Exprimer $|\mathcal{F}(T)|$ et $|\mathcal{N}(T)|$ en fonction de la hauteur de l'arbre T.

On considère l'algorithme probabiliste récursif qui consiste à tirer aléatoirement deux enfants u_1 et u_2, et n'évaluer σ sur u_3 l'enfant non choisi seulement si $\sigma(u_1) \neq \sigma(u_2)$.

2. Quel type d'algorithme s'agit t'il? Montrer que l'éspérence du nombre de feuilles que l'algorithme visite et inférieure à $\left(\frac{8}{3}\right)^h$ pour h la hauteur de T.
3. En déduire un algorithme probabiliste qui, en espérance, est de complexité $O(|\mathcal{N}(T)|^{0.9})$ (donc sous-linéaire). *On indique que* $\log_3(8) \approx 1,893$

On cherche maintenant à montrer qu'un algorithme déterministe correct doit forcément inspecter toutes les feuilles. On pose $n = |\mathcal{F}(T)|$. Soit $F \subseteq \mathcal{F}(T)$, on dit que F *détermine* un noeud $u \in \mathcal{N}(T)$ soit si $u \in F$ ou s'il a deux fils u_1 et u_2 déterminés par F tels que $\sigma(u_1) = \sigma(u_2)$

4. Montrer que pour tout $u \in \mathcal{N}(T)$ non déterminé par F, on peut modifier les valeurs de σ sur $\mathcal{F} \setminus F$ de manière à changer la valeur de $\sigma(u)$.
5. On considère le jeu à deux joueurs suivant : à chacun des $n - 1$ tours, Alice choisi une feuille $x \in \mathcal{F}(T)$ non déjà choisit et Bob décide si $\sigma(x) = V$ ou $\sigma(x) = F$. Alice gagne si l'ensemble des $n - 1$ feuilles détermine la racine, et Bob gagne dans le cas contraire. Montrer que Bob dispose d'une stratégie gagnante.
6. En déduire qu'un algorithme déterministe ne peut pas faire mieux que du $O(|\mathcal{N}(T)|)$.

[3]Exercice 100% recopié de Florian Bourse : https://www.di.ens.fr/~fbourse/enseignement/Proba.pdf

Unique graphe infini aléatoire

On dit que deux graphe $G = (V, E)$ et $G' = (V', E')$ sont isomorphe s'il existe $\varphi : V \longrightarrow V'$ bijective tel que $(u, v) \in E \Leftrightarrow (\varphi(u), \varphi(v)) \in E'$

On cherche à montré que les graphes aléatoire infinis dénombrable sont isomorphes avec probabilité 1. On considère pour cela les variables aléatoire de bernoulli $\left(X_{u,v}\right)_{u,v\in\mathbb{N}}$ indépendantes de probabilité p qui représente s'il existe une arête entre u et v. On note G_p cette famille de variables aléatoire.

Question 1 Quelle est la probabilité que le graphe possède le chemin $(0, 1, 2, ..., k)$?

On dit qu'un graphe infini possède *la propriété de Rado* si pour tout $U, V \subseteq \mathbb{N}$ disjoint fini, il existe $x \in \mathbb{N}$ tel que x soit connecté à tout U et a aucun V.

Question 2 Soit $U, V \subseteq \mathbb{N}$ fini, montrer que la probabilité qu'un $x \in \mathbb{N} \setminus (U \cup V)$ satisfaite la propriété de rado est non nul. En déduire l'existence d'un tel x avec probabilité 1.

Question 3 Montrer que si G et G' possède la propriété de Rado, alors ils sont isomorphes.

Question 4 Généraliser la question 3 pour montrer que la probabilité que G_p soit isomorphe a $G_{p'}$ est 1 si $0 < p, p' < 1$.

$(\Delta + 1)$-approx de Maximum independant set

Question de Cours Rapeller la définition d'un algorithme de monte-carlo et d'un algorithme de las-vegas.

Question 1 Montrer que si L est un langage hors-contexte alors $\overline{L} = \{\overline{w} : w \in L\}$ (l'ensemble des mirroirs de L) l'est aussi.

On considère le problème MIS suivant:

Entrée: Un graphe $G = (V, E)$ non orienté
Sortie: Un $I \subseteq V$ de cardinal maximal tel que $E \cap I^2 = \emptyset$

Pour $G = (V, E)$ un graphe, on note $\Delta(G) = \max_{v \in V} \deg(v)$.

Question 2 Donner la version problème de décision de MIS

Question 3 Donner un $(\Delta(G) + 1)$-approximation de MIS qui tourne en $O(|S|^2)$

On considère l'algorithme qui, tant que le graphe possède encore des arêtes, prend un sommet de degré non nul au hasard uniformément, le choisi et retire toutes les arêtes relié à ce sommet.

Question 4 Quel est ce type d'algorithme ? Soit $G = (V, E)$ un graphe, on note V_f la variable aléatoire de l'ensemble choisi à la fin de l'exécution de l'algorithme sur G. Montrer que

$$\mathbb{E}\big[|V_f|\big] = \sum_{v \in V} \frac{\deg v}{\deg v + 1}$$

Coloration aléatoire

Question de Cours Montrer que les langages réguliers sont hors-contexte

Soit $k \in \mathbb{N}$ fixé. On regarde le problème du k-PATH:

Entrée: Un graphe $G = (V, E)$ non orienté **Sortie:** Est-ce qu'il existe un chemin de longueur k ?

On définit une k-coloration comme une fonction $\mu : V \longrightarrow \{1, .., k\}$. On dit qu'un chemin est arc-en-ciel si tous les sommets sont de couleur différente.

Question 1 Quelle est la probabilité qu'un chemin de longueur k soit arc-en-ciel ?

Question 2 Donner un algorithme de programmation dynamique qui, soit G et une k-coloration, vérifie si G possède un chemin arc-en-ciel en $O\big(2^k \times |E|\big)$.

Question 3 En déduire un algorithme à erreur unilateralle de même probabilité que celle obtenu question 1. Quel type d'algorithme est-ce ?

Question 4 En déduire un algorithme à erreur unilateralle de probabilité $\frac{1}{2}$. Quel est son temps d'exécution ?

K-Centres

On fixe $V \subseteq \mathbb{R}^d$ un jeu de donnée. Pour $\vec{y} \in \mathbb{R}^d$ on définit $d(\vec{y}, V) = \max_{\vec{x} \in V} \|\vec{y} - \vec{x}\|$. On cherche à choisir k points de V tel qu'ils minimise $\max_{v \in V} d(v, X)$. On regarde donc le k-CENTRES suivant:

- **Entrée:** $v_1, ..., v_n \subseteq \mathbb{R}^d$ une liste de points finis
- **Sortie** : un $X := \{x_1, ..., x_k\} \subseteq V$ qui minimise $\max_{v \in V} d(v, X)$

Question 1 On cherche à montrer que l'algorithme du cours de la question ne donne pas de bonne approximation du problème des k-centres. Pour cela on pose $k = 2$, $d = 1$ et on pose N points en 0, N points en 1 et un point en D. Montrer que pour des bonnes valeurs de N et D on peut avoir le ratio $\frac{\text{opt-k-moyennes}}{\text{opt-k-centre}}$ arbitrairement large.

Question 2 Proposer un algorithme glouton pour le problème des k-centres.

Question 3 On note $p_1, ..., p_k$ les k points retrourné par notre algorithme et $p_1^*, ..., p_k^*$ la solution optimale. On partitionne $V = \bigsqcup_{i \in \mathbb{N}} V_i$ avec V_i l'ensemble des points les plus proches de p_i^*. Montrer que

- Si $\forall i, V_i \cap \{p_1, ..., p_n\} \neq \emptyset$ alors on a une 2-approximation
- Sinon, on a aussi une 2-approximation

Paire de points

Pour tout $x, y \in \mathbb{R}$ deux points distincts inconnus, on te donne avec probabilité $\frac{1}{2}$ le point x et sinon le point y. Tu peux décider de garder ou d'échanger de points avec une certaine probabilité qui dépend seulement du point que l'on te donne. Avec quelle probabilité dois-tu échanger de manière à ce que quel que soit la paire $(x, y) \in \mathbb{R}^2$ tu prend le plus grand point avec probabilité strictement plus grande que $\frac{1}{2}$?