Examen ALGO2

1 Applications du lemme local de Lovász

Soit $A_1, \dots, A_k$ des événements. Soit $J \subseteq \{1, \dots, k\}$. L'événément A_i est mutuellement indépendants de $\{A_j \mid j \in J\}$ si pour toute $J_1, J_2 \subseteq J$ avec $J_1 \cap J_2 = \emptyset$, on a :

$$\mathbb{P}(A_i \cap \bigcap_{j \in J_1} A_j \cap \bigcap_{j \in J_2} \overline{A_j}) = \mathbb{P}(A_i) \times \mathbb{P}(\bigcap_{j \in J_1} A_j \cap \bigcap_{j \in J_2} \overline{A_j}).$$

Dans cet exercice, on admet le lemme suivant :
Lemme local de Lovász *Soit $A_1, \cdots, A_k$ des événements, $p \in [0,1]$ et $d \in \mathbb{N}$. On suppose que pour tout $i \in \{1, \dots, k\}$,*

1. *$\mathbb{P}(A_i) < p$;*
2. *A_i est mutuellement indépendant des autres événements sauf au plus d, c'est-à-dire : il existe $S \subseteq \{1, \dots, k\} \setminus \{i\}$, $|S| = d$, tel que A_i soit mutuellement indépendant de $(\{A_j \mid j \in \{1, \dots, k\} \setminus \{i\}) \setminus S\}$.*

Alors si $p \times (d+1) \times e < 1$ (où e est la base du logarithme népérien) alors $\mathbb{P}(\bigcap_{i=1}^{k} \overline{A_i}) > 0$.

Si les A_i sont des *mauvais* événements, le lemme local de Lovász donne une condition suffisante pour qu'il existe des situations où aucun de ces mauvais événements n'a lieu en même temps. Il est en particulier utilisé pour étudier les algorithmes probabilistes.

Coloriage aléatoire de cycle Soit $\mathcal{C}_n$ le graphe constitué d'un cycle sur n sommets. Étant données c couleurs, on colorie $\mathcal{C}_n$ de façon naïve en affectant indépendamment à chaque sommet une couleur aléatoire, en suivant une distribution uniforme sur les couleurs.

1. Déterminer la probabilité qu'une arête soit monochromatique (c'est-à-dire, qu'elle ait les deux extrémités de la même couleur).

2. Calculer l'espérance du nombre d'arêtes monochromatiques.

3. Donner une borne sur le nombre c de couleurs pour pouvoir appliquer le lemme local de Lovász et en déduire que l'algorithme naïf a une probabilité non nulle de colorier correctement le cycle $\mathcal{C}_n$ (de façon à ce que deux sommets voisins n'est pas la même couleur). Comparer ce résultat à une coloration optimale.

k-SAT On considère le problème k-SAT qui restreint les instances de SAT à des formules ayant exactement k littéraux par clause.

4. Soit φ une formule en forme normale conjonctive avec k littéraux exactement par clause, et telle que chaque variable apparaît dans au plus ℓ clauses.
Donner une contrainte sur ℓ en fonction de k de façon à garantir que φ soit satisfiable. On considérera les événements A_i = « la clause i n'est pas satisfaite » en supposant qu'une valuation est choisie aléatoirement en suivant une distribution uniforme.

2 Non-approximabilité

Le problème du *Bin Packing* consiste à ranger des objets de différentes tailles en un nombre minimal de boîtes.

BIN PACKING

ENTRÉES	–	des rationnels $x_1, \dots, x_n \in]0, 1]$
SORTIE	–	le plus petit entier m pour lequel il existe $f : \{1, \dots, n\} \to \{1, \dots, m\}$ tel que, pour tout $b \in \{1, \dots, m\}$, $\sum_{i \in f^{-1}(b)} x_i \leq 1$.

On veut prouver que BIN PACKING n'admet pas d'algorithme en temps polynomial avec un facteur d'approximation meilleur que 3/2. Pour cela, considérons le problème suivant :

PARTITION

ENTRÉE	–	$a_1, \dots, a_n \in \mathbb{N}$, écrits en binaire
SORTIE	–	Oui s'il existe $A \subseteq \{1, \dots, n\}$ tel que $\sum_{i \in A} a_i = \sum_{i \notin A} a_i$, non sinon.

On admet que PARTITION est NP-complet.

5. Soit I une instance de PARTITION. Construire une instance I' de BIN PACKING telle que I admet une solution si et seulement si I' admet une solution à deux boites.

6. Conclure que, si $P \neq NP$, il n'existe pas d'algorithme en temps polynomial de facteur d'approximation strictement plus petit que 3/2.

3 Algorithmes d'approximation gloutons

On considère encore le problème BIN PACKING (voir section précédente). D'abord, on s'intéresse à un algorithme d'approximation glouton qui prend les éléments dans un ordre arbitraire. Il place au fur et à mesure chaque élément dans la première boite dans laquelle il rentre, et entame une nouvelle boite si l'élément ne rentre nul part. On note m le nombre de boites et f la fonction de rangement calculés par cet algorithme.

7. Montrer qu'au moins $m - 1$ paquets sont remplis à plus de la moitié de leur capacité. En déduire que cet algorithme glouton a un facteur d'approximation 2.

On considère maintenant une variante de l'algorithme précédent dans laquelle les éléments sont préalablement triés, avant d'être considérés par ordre décroissant de taille : $x_1 \geq x_2 \geq \dots \geq x_n$. On note f la fonction de rangement et m le nombre de boites calculés par l'algorithme. On pose $b = \lceil \frac{2m}{3} \rceil$.

8. Supposons que l'algorithme met dans la boite b un élément de taille strictement plus grosse que 1/2, i.e., il existe i tel que $f(i) = b$ et $x_i > 1/2$. Montrer que la solution optimale m^* vérifie $m^* \geq \frac{2}{3}m$.

Supposons maintenant que la boite b ne contienne pas d'éléments de taille $> 1/2$. On note $S := \sum_i x_i$ la somme de toutes les tailles des objets, et T le niveau de remplissage minimal des boites 1 à $b - 1$.

9. Montrer que $S > T \times (b - 1) + (2(m - b) + 1)(1 - T)$.

10. En déduire $S > \lceil 2m/3 \rceil - 1$. *Indice* : Utiliser $\lceil 2m/3 \rceil \leq 2m/3 + 2/3$.

11. Conclure que l'algorithme a un facteur d'approximation 3/2.