Algorithmes approchés

VLADY RAVELOMANANA

IRIF – UMR CNRS 8243
Université de Paris 7
vlad@irif.fr

– M1 Algorithmique avancée –

Plan de ce cours

- Algorithmes approchés.
- Approximation et schémas.
- Limite de l'approximabilité.
- Non approximabilité.
- Complexité paramétrique.

Le problème du rangement optimal ou “bin packing”

Définition.

Input: n objets de taille $a_1, a_2, \cdots, a_n$ (avec $0 < a_i \leq 1$).
Tâche: Trouver un rangement par paquets de taille 1 qui **minimise** le nombre de paquets.

Le problème du rangement optimal ou “bin packing”

Définition.

Input: n objets de taille $a_1, a_2, \cdots, a_n$ (avec $0 < a_i \leq 1$).
Tâche: Trouver un rangement par paquets de taille 1 qui **minimise** le nombre de paquets.

Un exemple.

Input: 0.3, 0.2, 0.2, 0.2, 0.2, 0.4, 0.5
Un rangement en 3 paquets: [0.3, 0.2, 0.2, 0.2], [0.2, 0.4] puis [0.5]
Un rangement optimal: [0.3, 0.2, 0.5] puis [0; 2, 0.2, 0.2, 0.4]

Le problème du rangement optimal ou “bin packing”

Définition.

Input: n objets de taille $a_1, a_2, \cdots, a_n$ (avec $0 < a_i \leq 1$).
Tâche: Trouver un rangement par paquets de taille 1 qui **minimise** le nombre de paquets.

Un exemple.

Input: 0.3, 0.2, 0.2, 0.2, 0.2, 0.4, 0.5
Un rangement en 3 paquets: [0.3, 0.2, 0.2, 0.2], [0.2, 0.4] puis [0.5]
Un rangement optimal: [0.3, 0.2, 0.5] puis [0; 2, 0.2, 0.2, 0.4]

Le glouton naturel: l’algorithme “First-Fit”.

1. Mettre objet après objet dans un des paquets ouverts si celui-ci peut encore recevoir l’objet considéré. Dans le cas contraire, ouvrir un nouveau paquet.
2. Refaire le point précédent tant qu’il y a des objets à ranger.

First-fit trouve une solution approchée

Théorème.

L'algorithme First-Fit est un algorithme d'approximation à facteur constant 2:
le problème Bin-Packing est donc dans **APX**.

First-fit trouve une solution approchée

Théorème.

L'algorithme First-Fit est un algorithme d'approximation à facteur constant 2:
le problème Bin-Packing est donc dans **APX**.

Preuve.

- On va supposer que First-Fit utilise m paquets.
- Si 2 paquets sont remplis moins que la moitié, alors les objets du second paquet peuvent être mis dans le premier. Donc au moins $(m-1)$ paquets sont plus qu'à moitié remplis. Si on note par $S_{\mathtt{opt}}$ le nombre de paquets optimal, on a alors

$$S_{\text{opt}} \geq \sum_{i=1}^{n} a_i > \frac{m-1}{2}.$$

La première inégalité est justifiée par le fait qu'il faut ranger tous les a_i.
La seconde inégalité est due au fait que $m-1$ paquets sont plus qu'à moitié remplis. Donc

$$2S_{\text{opt}} \geq m$$

Un meilleur algorithme pour Bin-Packing.

Un glouton amélioré.

1. Ordonner par ordre décroissant les objets.
2. Mettre les objets dans cet ordre dans le premier paquet disponible.
3. Refaire le point précédent tant qu'il y a des objets à ranger.

Un meilleur algorithme pour Bin-Packing.

Un glouton amélioré.

1. Ordonner par ordre décroissant les objets.
2. Mettre les objets dans cet ordre dans le premier paquet disponible.
3. Refaire le point précédent tant qu'il y a des objets à ranger.

Sur notre exemple.

Input: 0.3, 0.2, 0.2, 0.2, 0.2, 0.4, 0.5
Tri: 0.5, 0.4, 0.3, 0.2, 0.2, 0.2, 0.2
Un rangement en 3 paquets: [0.5, 0.4], [0.3, 0.2, 0.2, 0.2], [0.2]

Un meilleur algorithme pour Bin-Packing.

Un glouton amélioré.

1. Ordonner par ordre décroissant les objets.
2. Mettre les objets dans cet ordre dans le premier paquet disponible.
3. Refaire le point précédent tant qu'il y a des objets à ranger.

Sur notre exemple.

Input: 0.3, 0.2, 0.2, 0.2, 0.2, 0.4, 0.5
Tri: 0.5, 0.4, 0.3, 0.2, 0.2, 0.2, 0.2
Un rangement en 3 paquets: [0.5, 0.4], [0.3, 0.2, 0.2, 0.2], [0.2]

Théorème.

L'algorithme amélioré utilise au plus $1 + \frac{3}{2} S_{\text{opt}}$ paquets.

Un meilleur algorithme pour Bin-Packing.

Un glouton amélioré.

1. Ordonner par ordre décroissant les objets.
2. Mettre les objets dans cet ordre dans le premier paquet disponible.
3. Refaire le point précédent tant qu'il y a des objets à ranger.

Sur notre exemple.

Input: 0.3, 0.2, 0.2, 0.2, 0.2, 0.4, 0.5
Tri: 0.5, 0.4, 0.3, 0.2, 0.2, 0.2, 0.2
Un rangement en 3 paquets: [0.5, 0.4], [0.3, 0.2, 0.2, 0.2], [0.2]

Théorème.

L'algorithme amélioré utilise au plus $1 + \frac{3}{2} S_{\text{opt}}$ paquets.

Preuve.

- Si tous les objets ont tous leur poids $> 1/3$, la situation après le tri est $o_1 \geq \cdots o_i > \frac{2}{3} \geq o_{i+1} \geq \cdots o_j > \frac{1}{2} \geq o_{j+1} \geq \cdots o_n > \frac{1}{3}$ glouton est donc **optimal**. De même si chaque paquet contient un objet de poids $> 1/3$.
- Sinon à part un paquet tous les autres sont remplis à plus de $2/3$.

Le problème du rangement et limite de l'approximabilité

Théorème.

S'il existe un algorithme polynomial ε approché avec $\varepsilon < 1/3$ pour le problème du rangement optimal alors $\mathcal{P} = \mathcal{NP}$.

Le problème du rangement et limite de l'approximabilité

Théorème.

S'il existe un algorithme polynomial ε approché avec $\varepsilon < 1/3$ pour le problème du rangement optimal alors $\mathcal{P} = \mathcal{NP}$.

Preuve.

L'idée de la preuve est

- S'il existe un tel algorithme alors on aura un **`algorithme polynomial`** pour le **problème de partition** qui est $\mathcal{NP}$-complet.

Le problème de partition consiste à partitionner un ensemble d'entiers S en S_1 et S_2 de telle sorte à $\sum_{s \in S_1} s = \sum_{s \in S_2} s = \frac{S}{2}$ (ce problème est difficile!). Pour montrer le th., on considère le problème de partition $S = \{s_1, s_2, \cdots, s_n\}$ et le problème du rangement avec comme poids total possible $P = \frac{1}{2} \sum e_i$ (au lieu de 1.0). On va supposer qu'on a un algorithme $\varepsilon < 1/3$ approché pour Bin-Packing.
Si l'algorithme trouve 2 paquets alors c'est une solution à notre problème de partition. Sinon l'algorithme trouve un nombre de paquets > 2 et donc nous n'avons pas de solution à partition car $\frac{|\text{solution}| - s_{\text{opt}}}{|\text{solution}|} \geq \frac{3-2}{3} > \varepsilon$.

Le problème du rangement et limite de l'approximabilité

Théorème.

S'il existe un algorithme polynomial ε approché avec $\varepsilon < 1/3$ pour le problème du rangement optimal alors $\mathcal{P} = \mathcal{NP}$.

Preuve.

L'idée de la preuve est

- S'il existe un tel algorithme alors on aura un **`algorithme polynomial`** pour le **problème de partition** qui est $\mathcal{NP}$-complet.

Le problème de partition consiste à partitionner un ensemble d'entiers S en S_1 et S_2 de telle sorte à $\sum_{s \in S_1} s = \sum_{s \in S_2} s = \frac{S}{2}$ (ce problème est difficile!). Pour montrer le th., on considère le problème de partition $S = \{s_1, s_2, \cdots, s_n\}$ et le problème du rangement avec comme poids total possible $P = \frac{1}{2} \sum e_i$ (au lieu de 1.0). On va supposer qu'on a un algorithme $\varepsilon < 1/3$ approché pour Bin-Packing.
Si l'algorithme trouve 2 paquets alors c'est une solution à notre problème de partition. Sinon l'algorithme trouve un nombre de paquets > 2 et donc nous n'avons pas de solution à partition car $\frac{|\text{solution}| - s_{\text{opt}}}{|\text{solution}|} \geq \frac{3-2}{3} > \varepsilon$.

Corollaire.

Si $\mathcal{P} \neq \mathcal{NP}$ alors **PTAS** $\subsetneq$ **APX**.

Plan de ce cours

- Algorithmes approchés.
- Approximation et schémas.
- Limite de l'approximabilité.
- Non approximabilité.
- Complexité paramétrique.

Inapproximabilité du problème du voyageur de commerce

Définition.

Input: Un graphe G avec des arêtes valuées (lire: sommets=villes/arêtes=distance).
Tâche: Trouver un ordre de visite de toutes les villes minimisant la distance parcourue.

Inapproximabilité du problème du voyageur de commerce

Définition.

Input: Un graphe G avec des arêtes valuées (lire: sommets=villes/arêtes=distance).
Tâche: Trouver un ordre de visite de toutes les villes minimisant la distance parcourue.

Théorème.

S'il existe un algorithme ε approché avec $\varepsilon < 1$ pour le problème du voyageur de commerce alors $\mathcal{P} = \mathcal{NP}$.

Inapproximabilité du problème du voyageur de commerce

Définition.

Input: Un graphe G avec des arêtes valuées (lire: sommets=villes/arêtes=distance).
Tâche: Trouver un ordre de visite de toutes les villes minimisant la distance parcourue.

Théorème.

S'il existe un algorithme ε approché avec $\varepsilon < 1$ pour le problème du voyageur de commerce alors $\mathcal{P} = \mathcal{NP}$.

Corollaire.

TSP ou le problème du voyageur de commerce n'est pas dans la classe **APX**.

Preuve de l'inapproximabilité de TSP

Preuve.

- Le problème du circuit hamiltonien consiste à trouver un cycle passant une et exactement une fois par tous les sommets. Ce problème est $\mathcal{NP}$-complet.
- Etant donné un graphe G, on construit un graphe G' **valué** de la façon suivante: soit k un entier fixé mais $\geq \frac{1}{1-\varepsilon}$, et x et y deux sommets de G

$$\begin{cases} v(x,y) = 1 \text{ dans G' si } (x,y) \text{ est une arête de } G \\ v(x,y) = nk+1 \text{ sinon .} \end{cases}$$

- Si on dispose d'un algorithme d'approximation pour le problème du voyageur du commerce alors G admet un cycle hamiltonien sinon le poids de l'ordre de visite trouvé est $S \geq (n-1)+(nk+1)$ mais l'approximation nous dit $\frac{n+nk-S_{\text{opt}}}{n+nk} \leq \varepsilon$ on a $S_{\mathtt{opt}} \geq n(1+k)(1-\varepsilon) > n$!!!

Plan de ce cours

- Algorithmes approchés.
- Approximation et schémas.
- Limite de l'approximabilité.
- Non approximabilité.
- Complexité paramétrique.

FPT.

Problème de décision

Avec des données $\mathcal{D}$ et une propriété P, il s'agît d'écrire un algorithme capable de décider si **OUI** ou **NON**, $\mathcal{D}$ satisfait P.

FPT.

Problème de décision

Avec des données $\mathcal{D}$ et une propriété P, il s'agît d'écrire un algorithme capable de décider si **OUI** ou **NON**, $\mathcal{D}$ satisfait P.

Problème de décision/d'optimisation et paramètre

A un problème de décision:

$\mathcal{D}$ satisfait ou non P

on peut toujours associer un problème d'optimisation

Maximiser/minimiser une partie de $\mathcal{D}$ satisfaisant P

et un paramètre presque naturel

A-t-on une partie de `taille k` de $\mathcal{D}$ satisfaisant P

FPT.

Problème de décision

Avec des données $\mathcal{D}$ et une propriété P, il s'agît d'écrire un algorithme capable de décider si **OUI** ou **NON**, $\mathcal{D}$ satisfait P.

Problème de décision/d'optimisation et paramètre

A un problème de décision:

$\mathcal{D}$ satisfait ou non P

on peut toujours associer un problème d'optimisation

Maximiser/minimiser une partie de $\mathcal{D}$ satisfaisant P

et un paramètre presque naturel

A-t-on une partie de `taille k` de $\mathcal{D}$ satisfaisant P

La classe "Fixed-parameter tractable" (FPT)

Un algorithme s'exécutant de manière générale peut avoir une complexité de la forme

`(une partie exponentielle en k)` $\times$ (polynomial en la taille totale des données)

Complexité paramétrée: exemples

Problème:	MINIMUM VERTEX COVER	MAXIMUM INDEPENDENT SET
Données:	Un graphe G et un entier k	Un graphe G et un entier k
Question:	Peut-on couvrir les arêtes de G avec k sommets?	Existe-il un ens. indépendant de taille k?
Complexité:	$\mathcal{NP}$-complet	$\mathcal{NP}$-complet
Alg. basé sur l'énumération	$O(n^k)$	$O(n^k)$
FPT? ,	il existe un algorithme en $O(2^k n^2)$	Aucun algorithme en $n^{o(k)}$ connu!