Chapître VI.
Algorithmes d'approximation

1. Problèmes NP-complets

De nombreux problémes d'intérêt pratique sont NP-complets. En particulier une grande partie de problèmes des graphes et réseaux.

Livre recommandé:
Garey, Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, W.H. Freeman, San Francisco, 1979.

La plupart de problèmes sont trop importants pour qu'on laisse tomber, mais il y a très peu de chance de trouver un algorithme polynomial capable de résoudre un problème NP-complet exactement.

2. Attaquer un problème NP-complet

Supposons notre problème particulier (comme par exemple ENSEMBLE STABLE) est NP-complet. C'est-à-dire il ne faut pas perdre de temps en cherchant un **algorithme efficace** capable de le résoudre **pour toutes les entrées** avec **exactitude**.

Cependant sachant que le probléme est NP-complet nous essayons de relaxer une ou plusieurs des conditions trop strictes et alors nous arrivons à des differents types d'attaque possible:

0) petite entrée
1) algorithmes d'approximation
2) algorithmes randomisés
3) cas particuliers

3. Problème d'optimisation

Pour définir et parler d'un algorithme d'approximation il faut que le problème consideré est un **problème d'optimisation**.

-) Dans un problème d'optimisation il y a de nombreuses solutions possibles.

-) Chaque solution possible est affectée d'une valeur.

-) On souhaite trouver une solution ayant la valeur optimale (minimum ou maximum).

Exemple:

ENSEMBLE STABLE

-) donnée: graphe non-orienté $G = (V, E)$

-) solution: $I \subseteq V$ est une solution si I est un <u>stable</u> de G, c'est-à-dire quelque soit deux sommets $x \in I$ et $y \in I$ alors x et y ne sont pas adjacents.

-) valeur: $val(I) = |I|$

4. Algorithmes d'approximation

Livre recommandé:

Approximation Algorithms
V.V. Vazirani
Springer-Verlag, 2000.

Un algorithme qui retourne des solutions presque optimales et qui s'exécute un temps polynomial au pire des cas est appelé **algorithme d'approximation**.

C'est-à-dire on demande qu'un algorithme d'approximation est un **algorithme efficace** capable de résoudre le problème **pour toutes les entrées**, mais on laisse tomber le demande que l'algorithme doit trouver une solution optimale.

5. Mésurer la qualité d'un algorithme d'approximation

Supposons qu'on travaille sur un problème d'optimisation dans lequel chaque solution potentielle a un coût positif, et on souhaite trouver une solution presque optimal.

Soit $OPT(I)$ la valeur d'une solution optimal pour l'entrée I, et soit $ALG(I)$ la valeur de la solution retournée par l'algorithme A pour l'entrée I. On dit que le **facteur d'approximation** (souvent aussi appelé rapport d'approximation) de l'algorithme A est (au plus) $\varrho_A(n)$ si pour tout l'entrée I de taille n:

$$\max\left(\frac{ALG(I)}{OPT(I)}, \frac{OPT(I)}{ALG(I)}\right) \leq \varrho_A(n).$$

A. Couverture de sommet

Une **couverture de sommet** dans un graphe non orienté $G = (V, E)$ est un sous-ensemble $V' \subseteq V$ tel que si $\{u, v\} \in E$, alors soit $u \in V'$ soit $v \in V'$ (ou les deux). La taille d'une couverture de sommet est le nombre de sommets qu'elle contient.

Le **problème de la couverture de sommet** consiste à trouver une couverture de la taille minimale dans un graphe non orienté donné.

Le probléme de decision correspondant
$\{(G, k):\ G = (V, E)$ a une couverture de sommet V' tel que $|V'| \leq k\}$
est NP-complet.

1. Algorithme d'approximation

Bien qu'il puisse être difficile de trouver une couverture de sommet minimale dans un graphe G, il n'est pas très difficile de trouver une couverture de sommet presque optimal.

L'algorithme COUVERTURE-SOMMET-APPROCHÉE(G) prend en entrée un graphe non orienté G et retourne une couverture de sommet de G dont on peut être sûr que la taille n'est pas plus de deux fois plus grande que la taille d'une couverture de sommet minimale:

$$\forall G: \ ALG(G) \leq 2 \cdot OPT(G).$$

$COUVERTURE\text{–}SOMMET\text{–}APPROCHEE(G)$

$C \leftarrow \emptyset$
$U \leftarrow E$
tant que $U \neq \emptyset$
  faire soit $\{u, v\}$ une arête arbitraire de U
    $C \leftarrow C \cup \{u, v\}$
    supprimer de U toutes les arêtes incidentes soit à u soit à v
retourner C

Observation: Le temps d'exécution de cet algorithme est $O(m)$.

Exercice:
Donner une implémentation de l'algorithme COUVERTURE-SOMMET-APPROCHEE(G) qui s'exécute vraiment un temps en $O(m)$.

2. La qualité d'approximation

Théorème. *Le facteur d'approximation de l'algorithme COUVERTURE-SOMMET-APPROCHEE(G) est égal à 2.*

Démonstration: L'ensemble C retourné par l'algorithme est une couverture de sommet parce que l'algorithme boucle jusqu'à l'ensemble U d'arêtes non encore couverts par l'ensemble C courant est vide.

Pour comprendre que $ALG(G) \leq 2 \cdot OPT(G)$, soit M l'ensemble des arêtes qui ont été choisies à la ligne 4 de COUVERTURE-SOMMET-APPROCHEE. M est un couplage de G parce que pour chaque arête choisie à la ligne 4 tous les arêtes ayant une extrémité commune sont supprimer à la ligne 6.

Pour couvrir seulement les arêtes du couplage M il faut $|M|$ sommet. Par la construction de M on a $|C| = 2|M|$. C'est pourquoi $OPT(G) \geq |C|/2$, et $|C| = ALG(G) \geq OPT(G) \geq |C|/2$. $\square$

3. Exercice

Le professeur Tatin propose l'heuristique suivante pour résoudre le problème de la couverture de sommet.

On choisit de façon répété un sommet de plus haut degré, et on supprime toutes ses arêtes incidentes.

Donner un exemple montrant que l'heuristique du professeur n'a pas un facteur d'approximation inférieur ou égal à 2.

B. Couverture d'ensemble

1. Le problème

Entrée: ensemble fini X et une famille $\mathcal{F}$ de sous-ensembles de X tel que $X = \bigcup_{S \in \mathcal{F}} S$.

On dit qu'un sous-ensemble $S \in \mathcal{F}$ **couvre** ses éléments. $\mathcal{F}' \subseteq \mathcal{F}$ est appelé une **couverture d'ensemble** ou une couverture de l'instance $(X, \mathcal{F})$ si $X = \bigcup_{S \in \mathcal{F}'} S$.

Le **problème de la couverture d'ensemble** consiste à trouver un sous-ensemble de la taille minimale $\mathcal{C} \subseteq \mathcal{F}$ dont tous les membres couvrent X en entier: $X = \bigcup_{S \in \mathcal{C}} S$.

Le probléme de decision correspondant $\{(X, \mathcal{F}, k) :\ (X, \mathcal{F})$ a une couverture $\mathcal{F}'$ tel que $|\mathcal{F}'| \leq k\}$ est NP-complet.

2. Exemple

Le problème de la couverture d'ensemble est un modèle de problèmes combinatoires qu'on rencontre fréquemment.

Exemple:

X répresente un ensemble de compétences nécessaires pour résoudre un problème, et qu'on dispose d'un ensemble de personnes disponibles pour travailler à ce problème.

Chaque ensemble $S \in \mathcal{F}$ répresente les compétences d'une personne.

On souhaite créér un comité, compose du moins de personnes possible, tel que pour chaque compétence requise de X, un membre du comité possède cette compétence.

3. Algorithme d'approximation

Considérons un algorithme glouton pour le problème de la couverture d'ensemble.

Choix glouton:

La méthode gloutonne fonctionne en choisissant, à chaque étape, l'ensemble S de $\mathcal{F}$ qui couvre le plus d'éléments non encore couverts.

$COUVERTURE\!-\!ENSEMBLE\!-\!GLOUTON(X, \mathcal{F})$

$U \leftarrow X$
$\mathcal{C} \leftarrow \emptyset$
tant que $U \neq \emptyset$
    faire choisir un $S \in \mathcal{F}$ qui maximise $|S \cap U|$
        $U \leftarrow U \setminus S$
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{S\}$
retourner $\mathcal{C}$

4. Analyse du temps

Le nombre d'itérations de la boucle des lignes 3-6 est au plus égal à $\min(|X|, |\mathcal{F}|)$.

Le corps de la boucle peut être facilement implémenté pour s'exécuter en $O(|X| \cdot |\mathcal{F}|)$.

Il existe une implémentation de l'algorithme COUVERTURE-ENSEMBLE-GLOUTON$(X, \mathcal{F})$ **qui s'exécute un temps en**

$$O(|X| \cdot |\mathcal{F}| \cdot \min(|X|, |\mathcal{F}|)).$$

Exercice:

Montrer comment implémenter COUVERTURE-ENSEMBLE-GLOUTON de telle sorte qu'il puisse s'exécuter en temps $O(\sum_{S \in \mathcal{F}} |S|)$, c'est-à-dire en temps linéaire.

5. Le facteur d'approximation

Le d-ème nombre harmonique $\sum_{i=1}^{d} 1/i$ sera noté $H(d)$.

Théorème. *Le facteur d'approximation de COUVERTURE-ENSEMBLE-GLOUTON*$(X, \mathcal{F})$ *est au plus* $H(\max\{|S| : S \in \mathcal{F}\})$.

En utilisant $\ln d \leq H(d) \leq 1 + \ln d$ on obtient

Corollaire. *Le facteur d'approximation de COUVERTURE-ENSEMBLE-GLOUTON*$(X, \mathcal{F})$ *est au plus* $O(\ln |X| + 1)$.

Théorème. *Il y a un constant* $c > 0$ *tel qu'il n'existe pas d'algorithme d'approximation ayant un facteur au plus* $c \log n$ *excepté P=NP.*

Consulter

`http://www.nada.kth.se/~viggo/problemlist/compendium.html`

6. La démonstration

Théorème. *Le facteur d'approximation de COUVERTURE-ENSEMBLE-GLOUTON*$(X, \mathcal{F})$ *est au plus* $H(\max\{|S| : S \in \mathcal{F}\})$.

Idée de la démonstration:

La démonstration s'effectue en attribuant un **coût** à chaque ensemble choisi par l'algorithme,

en repartissant ce coût sur les éléments couverts pour la première fois,

puis en utilisant ces coûts pour déduire la relation souhaitée entre la taille $OPT(X, \mathcal{F})$ d'une couverture minimale et la taille $ALG(X, \mathcal{F})$ de la couverture retourné par l'algorithme.

Coûts:

Soit S_i le i-ème sous-ensemble choisi par l'algorithme.

L'algorithme subit un coût égal á 1 quand il ajoute S_i à $\mathcal{C}$.

On répartit ce coût dû au choix de S_i équitablement entre les éléments couverts pour la première fois par S_i.

Pour chaque $x \in X$, soit c_x le coût attribué à x. Chaque élément est attribué un coût une seule fois, quand il est couvert la première fois. C'est pourquoi

$$c_x = \frac{1}{|S_i \setminus (S_1 \cup S_2 \cup S_3 \cup \cdots S_{i-1})|}$$

où x est couvert par la première fois par S_i.

Cœur de la démonstration:

Soit $\mathcal{C}_{opt}$ une couverture minimale et soit $\mathcal{C}_{alg}$ la couverture trouvé par l'algorithme. Le coût total de $\mathcal{C}_{alg}$ a été réparti entre les éléments de X. Donc, puisque $\mathcal{C}_{opt}$ couvre X, on a

$$|\mathcal{C}_{alg}| = \sum_{x \in X} c_x \leq \sum_{S \in \mathcal{C}_{opt}} \sum_{x \in S} c_x.$$

Le reste de la preuve tient dans l'inégalité fondamentale ci-dessous, qui sera bientôt démontrée. Pour tout ensemble S appertenant à la famille $\mathcal{F}$:

$$\sum_{x \in S} c_x \leq H(|S|).$$

D'après les deux inégalités, on a

$$|\mathcal{C}_{alg}| \leq \sum_{S \in \mathcal{C}_{opt}} H(|S|) \leq |\mathcal{C}_{opt}| \cdot H(\max\{|S| : S \in \mathcal{F}\}),$$

ce qui prouve le théorème. $\square$

Démonstration de $\sum_{x \in S} c_x \leq H(|S|)$:

Pour tout ensemble $S \in \mathcal{F}$ et $i = 1, 2, \ldots, |\mathcal{C}|$, soit $u_i = |S \setminus (S_1 \cup S_2 \cup \cdots \cup S_i)|$
le nombre d'éléments de S non couverts après $S_1, S_2, \ldots, S_i$ ont été choisis par l'algorithme.

On définit $u_0 = |S|$.

Soit k le plus petit indice tel que $u_k = 0$.

Alors, $u_{i-1} \geq u_i$, et $u_{i-1} - u_i$ éléments de S sont couverts pour la première fois par S_i, pour $i = 1, 2, \ldots, k$.

Donc,

$$\sum_{x \in S} c_x = \sum_{i=1}^{k} (u_{i-1} - u_i) \cdot \frac{1}{S_i \setminus S_1 \cup S_2 \cup \cdots \cup S_{i-1}}.$$

$$|S_i \backslash (S_1 \cup \cdots \cup S_{i-1})| \geq |S \backslash (S_1 \cup \cdots \cup S_{i-1})| = u_{i-1}$$

parce que le choix glouton de S_i garantit que S ne peut pas couvrir plus de nouveaux éléments que S_i.

On obtient $\sum_{x \in S} c_x \leq \sum_{i=1}^{k} (u_{i-1} - u_i) \cdot \frac{1}{u_{i-1}}$.

Pour les entiers a et b, où $a < b$, on a

$$H(b) - H(a) = \sum_{i=a+1}^{b} \frac{1}{i} \geq \frac{b-a}{a}.$$

En se servant de cette inégalité, on obtient

$$\sum_{x \in S} \leq \sum_{i=1}^{k} (H(u_{i-1}) - H(u_i)) =$$

$$= H(u_0) - H(u_k) = H(u_0) - H(0) = H(|S|).$$