Cours « Conception et analyse d'algorithmes »
TD n^{o}6 – algorithmes d'approximation, exploration

1 Problème du placement d'usines

Une entreprise souhaite installer des usines (en nombre n) dans un nouveau territoire, elle a pour cela acquis n terrains et, après une étude minitieuse, on a pu déterminer pour chaque couple d'usines U_i, U_j le nombre $v_{i,j}$ de véhicules qui devront les relier chaque mois et pour chaque couple d'emplacements E_p, E_q le coût $c_{p,q}$ du voyage d'un véhicule entre ces emplacements. Il s'agit alors de déterminer l'affectation optimale des usines sur les emplacements.

i. Montre que cette recherche s'exprime comme un problème linéaire en nombres entiers.

ii. On souhaite résoudre le problème par une technique de *branch and bound*, pour cela chaque sommet de l'arbre d'exploration représentera une situation où certaines usines auront été affectées en des emplacements et les autres ne seront pas encore affectées.

Ou vous demande de donner une évaluation d'une borne inférieure du coût de la meilleure affectation possible à partir d'une situation donnée. Pour cela il faudra diviser les voyages dont on détermine le coût en trois catégories : ceux entre usines déjà affectées, ceux entre chaque usine affectée et les usines non-affectées et ceux entre les usines non-affectées. Vous utiliserez le fait que le produit scalaire de $a_1, \ldots, a_m$ par une permutation de $b_1, \ldots, b_m$ est minimal losrque l'on trie ces deux vecteurs et que l'on multiplie le plus grand a_i par le plus petit b_j et ainsi de suite.

iii. Quel est la complexité du calcul de la borne inférieure en fonction du nombre p d'usines affectées et $q = n - p$ d'usines non affectées.

iv. On suppose qu'il y a 4 usines A, B, C, D dans lesquelles le nombre d'echanges de véhicules par mois est donné par le tableau suivant :

	A	B	C	D
A	0	8	8	12
B	8	0	5	2
C	8	5	0	11
D	12	2	11	0

et que les coûts des voyages entre les emplacements a, b, c, d sont donnés par :

	a	b	c	d
a	0	2	2	1
b	2	0	2	3
c	2	2	0	3
d	1	3	3	0

Evaluez le coût de l'affectation :

$$A->a, \quad B->b, \quad C->c, \quad D->d$$

en déduire que le sous arbre pour lequel A est affecté à d ne sera pas exploré.

v. Pour l'exemple ci-dessus dessinez l'arbre d'exploration par la méthode branch and bound en faisant apparaître uniquement les sous-arbres explorés.

2 Algorithme approché pour un recouvrement minimal

Le problème RECOUVREMENT PAR ENSEMBLES s'énonce de la façon suivante :

Données : un ensemble fini E, une famille $(S_i)_{1 \leq i \leq k}$ de sous-ensembles de E et pour chaque S_i un poids $w(S_i)$ qui est un entier positif.

Il s'agit de déterminer p sous-ensembles $S_{j_1}, \ldots S_{j_p}$ dont l'union est E et dont la somme des poids est minimale. Soit plus formellement $J = \{j_1, j_2, \ldots, j_p\}$ tel que

$$\cup_{j \in J} S_j = E, \text{ et } \quad \sum_{j \in J} w(S_j) \text{ minimal.}$$

2.1 Recouvrement par ensembles est NP-complet

1. En utilisant un paramètre supplémentaire w, donner une version de décision de ce problème.

On se propose de montrer que cette version est NP-complète. Pour cela on réduit RECOUVREMENT PAR SOMMETS à ce problème. Etant donné un graphe $G = (V, E)$, on considère le RECOUVREMENT PAR ENSEMBLES de l'ensemble E des arêtes, en associant à chaque sommet v de V un sous-ensemble S_v de poids 1.

2. Comment choisir les ensembles S_v pour que le recouvrement de E par les S_v détermine un recouvrement par sommets du graphe G ?

3. Démontrer que le problème posé est NP-complet.

2.2 Algorithme approché glouton

On considère l'algorithme glouton suivant qui calcule un recouvrement J (où $|X|$ dénote le nombre d'éléments de X et $X \setminus Y$ l'ensemble des éléments de X qui ne sont pas dans Y) :

- Pour i de 1 à k Calculer $c(S_i) = w(S_i)/|S_i|$.
- $F \leftarrow \emptyset$, $J \leftarrow \emptyset$.
- Tant que $F \neq E$ Faire
  - Soit i tel que $c(S_i)$ soit minimal
  - $F \leftarrow F \cup S_i, \quad J \leftarrow J \cup \{i\}$
  - Recalculer les $c(S_i)$ par $c(S_i) = w(S_i)/|S_i \setminus F)|$.

Soit $(T_i)_{1\leq i\leq k}$ les ensembles du recouvrement ordonnés dans l'ordre dans lequel ils sont ajoutés par l'algorithme glouton.

1. Soit W_0 la somme des poids des sous-ensembles S_i qui figurent dans la solution optimale.

Montrer qu'au moins un de ces sous ensembles S_i satisfait $\frac{w(S_i)}{|S_i|} \leq \frac{W_0}{n}$

2. En déduire que $\frac{w(T_1)}{|T_1|} \leq \frac{W_0}{n}$

Soit $x_1, \ldots, x_n$ les éléments de E dans l'ordre dans lequel ils sont ajoutés dans F.

3. Montrer que le sous-ensemble T_i contenant x_j à l'étape où il entre dans F satisfait lors de cette étape : $c(T_i) \leq W_0/(n-j+1)$.

4. En déduire que l'algorithme glouton est H_n approché, où $H_n = 1 + 1/2 + \ldots + 1/n$.

5. Soit les $n+1$ sous-ensembles suivants de $E = \{1, 2, \ldots, n\}$, $S_i = \{i\}$ pour $1 \leq i \leq n$ et $S_{n+1} = E$ et soit $\varepsilon > 0$. Donner des poids à ces sous-ensembles pour que le résultat de l'algorithme glouton diffère de l'optimum d'un facteur $H_n/(1+\varepsilon)$.

2.3 Recouvrement de mots

Soit Σ un alphabet fini, un mot est une suite finie de lettres de Σ, la longueur d'un mot f est notée $\ell(f)$. On dit qu'un mot g est *facteur* du mot f si $f = f'gf''$, c'est à dire si f contient une suite consécutives de lettres qui forment un mot égal à g.

On s'intéresse dans la suite au problème P suivant :

Soit un ensemble $S = \{f_1, \ldots, f_n\}$ de mots, déterminer le plus court mot f contenant tous les f_i comme facteurs. On se propose d'appliquer les résultats qui précèdent à ce problème.

Deux mots f_i et f_j tels que les k dernieres lettres de f_i forment un mot égal à celui constitué par les k premiéres de f_j sont dits *k-compatibles*. On note dans ce cas σ_{ijk} le mot de longueur $n_i + n_j - k$ obtenu par concaténation de f_i et des $n_j - k$ derniers caractères de f_j (n_i et n_j sont les longueurs de f_i et f_j).

1. Soit $S = \{f_1 = ab^k, f_2 = b^kc, f_3 = b^{k+1}\}$. Que vaut σ_{12k} ? Quel est le f optimal ?

2. On considère l'algorithme glouton qui à chaque étape, détermine deux mots f_i et f_j de S qui sont k-compatibles pour un k maximal, et qui remplace alors f_i et f_j dans S par σ_{ijk}.

L'algorithme termine quand S est réduit à un seul mot. Montrer sur un exemple que cet algorithme est au mieux $(1/2)$-approché.

On associe à un problème P, donné par $S = \{f_1, \ldots, f_n\}$, un problème Q de recouvrement par sous-ensembles. Pour Q on pose $E = S$ et on construit des sous-ensembles S_u pour tous les mots u obtenus de la façon suivante : u est ou bien dans S ou bien égal à σ_{ijk} pour un couple (f_i, f_j) k-compatible ; S_u est alors l'ensemble des facteurs de u qui sont dans S, il a pour poids $w(S_u) = \ell(u)$ la longueur de u.

3. Soit $S_{\theta_1}, \ldots, S_{\theta_p}$ les ensembles donnant une solution optimale au problème Q, et W_0 le poids de cette solution. Donner une solution de longueur W_0 au problème P.

Soit f le mot solution optimale du problème P, dans lequel on suppose que deux mots de S ne sont pas facteur l'un de l'autre. On renumérote les f_i de sorte que f_i apparaît pour la première fois avant f_j dans f si et seulement si $i < j$. *Par abus de notation, f_i désignera uniquement sa première occurence dans f.*

On pose $b_1 = 1$, et soit e_1 le plus grand indice d'un mot de S qui a une intersection avec f_1 dans f. De même, si $1 \leq e_i < n$, on pose $b_{i+1} = e_i + 1$ et e_{i+1} le plus grand indice tel que $f_{e_{i+1}}$ ait une intersection avec $f_{b_{i+1}}$. On continue jusqu'à ce que $e_t = n$.

4. On note k_i la longueur de l'intersection entre f_{b_i} et f_{e_i}, et $\pi_i = \sigma_{b_i e_i k_i}$. Montrer que $S_{\pi_1}, \ldots, S_{\pi_t}$ est une solution au problème Q. Quel est son poids ?

5. Montrer que π_i et π_{i+2} sont des facteurs de f qui ne s'intersectent pas dans f. En déduire que $\sum_i \ell(\pi_i) \leq 2\ell(f)$.

6. En déduire un algorithme $2H_n$ approché qui trouve un mot f contenant tous les f_i comme facteurs.