Agrégation externe de mathématiques, texte d'exercice diffusé en 2012
Épreuve de modélisation, option informatique

Résumé : le problème du rangement de boîtes est un des classiques de l'optimisation combinatoire. Dans ce problème, on veut ranger un ensemble de boîtes, de volumes variés, dans des conteneurs ayant tous le même volume.

Nous nous proposons d'étudier diverses propriétés du rangement optimal. L'obtention d'un tel rangement optimal constitue un problème NP-complet.

Nous analysons ensuite diverses stratégies (ou *heuristiques*) donnant un résultat pas trop éloigné de l'optimum : rangement séquentiel (*next fit*), puis stratégie *first fit*. Le cas où cette dernière stratégie est appliquée à une instance décroissante fait l'objet d'une étude particulière.

Mots clefs : optimisation combinatoire, NP-complétude

► *Il est rappelé que le jury n'exige pas une compréhension exhaustive du texte. Vous êtes laissé(e) libre d'organiser votre discussion comme vous l'entendez. Des suggestions de développement, largement indépendantes les unes des autres, vous sont proposées en fin de texte. Vous n'êtes pas tenu(e) de les suivre. Il vous est conseillé de mettre en lumière vos connaissances à partir du fil conducteur constitué par le texte. Le jury appréciera que la discussion soit accompagnée d'exemples traités sur ordinateur.*

1. Introduction

1.1. *Définitions et notations*

► Nous nous intéressons au problème suivant : on dispose de *conteneurs*, tous de même capacité entière $c \geq 2$, dans lesquels on souhaite ranger des *objets* de volumes respectifs $X_1, X_2, \ldots, X_n$, lequels appartiennent tous à l'intervalle $[\![1, c]\!]$. La forme d'un objet importe peu : on peut imaginer que c'est un sac de céréales, déformable à loisir, mais sans changement de son volume.

► Une *instance* du problème est un couple $(X; c)$, où c est un entier au moins égal à 2, et X une liste $X = (X_1, X_2, \ldots, X_n)$ d'entiers compris entre 1 et c inclus.

► Soit p un naturel non nul. Une application $f : [\![1, n]\!] \mapsto [\![1, p]\!]$ est un *rangement* de l'instance $(X; c)$ si elle est surjective et vérifie, pour tout $i \in [\![1, p]\!]$:

$$\sum_{j \in \Omega_i} X_j \leq c$$

où $\Omega_i = f^{-1}\big(\{i\}\big)$. Avec le rangement f, l'objet numéro j est placé dans le conteneur numéro $f(j)$. La surjectivité de f traduit le fait qu'aucun conteneur n'est vide ; la deuxième condition traduit le fait qu'aucun conteneur ne déborde (le volume total des objets placés dans un conteneur est au plus égal au volume de ce dernier).

() Option informatique

▶ Le nombre p de conteneurs requis pour le rangement f est noté $\|f\|$, c'est le *coût* de ce rangement. Nous notons $\mathscr{V}(X) = \sum_{1\leq i\leq n} X_i$ le volume total des objets à ranger. Il est clair qu'il existe au moins un rangement de l'instance $(X;c)$, et que l'ensemble des rangements de $(X;c)$ est fini.

1.2. *Propriétés d'un rangement optimal*

▶ Parmi tous les rangements possibles de l'instance $(X;c)$, il en existe au moins un de coût minimal. Nous noterons désormais $\omega(X;c) = \min\{\|f\|\}$ où f parcourt l'ensemble des rangements de $(X;c)$. Un rangement f de X tel que $\|f\| = \omega(X;c)$ sera dit *optimal*.

▶ La fonction ω vérifie diverses propriétés :
- la majoration $\mathscr{V}(X) \leq c\omega(X;c)$ est immédiate ; un raisonnement très simple nous donne $\omega(X;c) \leq \left\lceil \frac{2\mathscr{V}(X)}{c} \right\rceil$
- soit $Y = (Y_1, Y_2, \ldots, Y_n)$ une autre famille d'entiers, tous compris entre 1 et c inclus ; si $X_i \leq Y_i$ pour tout $i \in [\![1,n]\!]$, alors $\omega(X;c) \leq \omega(Y;c)$

▶ Est-il intéressant de grouper les objets deux par deux ? Pour répondre à cette question, vous pouvez étudier le signe de la quantité :

$$\omega(X_1+Y_1, X_2+Y_2, \ldots, X_n+Y_n; c) - \omega(X_1, X_2, \ldots, X_n, Y_1, Y_2, \ldots, Y_n; c)$$

où les X_k et les Y_k sont tels que $2 \leq X_k + Y_k \leq c$ pour $1 \leq k \leq n$.

▶ Il n'est pas toujours avantageux de remplir complètement un conteneur. On peut construire, pour $n \geq 6$, une instance $(X_1, X_2, \ldots, X_n; c)$ vérifiant les deux conditions suivantes :
- **C1** : $X_1 + X_2 + X_3 = c$
- **C2** : $\omega(X_1, X_2, \ldots, X_n; c) < 1 + \omega(X_4, X_5, \ldots, X_n; c)$

On peut même construire une instance vérifiant **C1**, **C2** et la condition supplémentaire suivante :

- **C3** : la seule partie I de $[\![1,n]\!]$ telle que $\sum_{i\in I} X_i = c$ est $\{1,2,3\}$.

2. Aspect algorithmique

2.1. *Analyse de la stratégie de rangement séquentiel (*next fit*)*

▶ La stratégie de rangement séquentiel (*next fit*) consiste à loger des objets de numéros consécutifs dans le conteneur courant, tant que ceci est possible. Si l'espace disponible n'est pas suffisant pour l'objet courant, on «referme» le conteneur, et on alloue un nouveau conteneur. Cette stratégie autorise un fonctionnement «en ligne» : la prise de décision peut se faire en un temps borné indépendamment de l'instance.

► Formellement, si $(X_1, X_2, \ldots, X_n; c)$ est l'instance à ranger, le premier conteneur accueillera les objets X_1 à X_k, où k est le plus grand indice $j \in [\![1, n]\!]$ tel que $\sum_{1 \leq i \leq j} X_i \leq c$. Si $k = n$, c'est fini ; sinon, on range l'instance $(X_{k+1}, \ldots, X_n; c)$ dans un nouveau conteneur (ou plusieurs), en appliquant la même stratégie.

► Nous noterons $C_{\text{nf}}(X; c)$ le coût du rangement obtenu par application de la stratégie de rangement séquentiel à l'instance $(X; c)$.

► Avec cette stratégie, le facteur 2 dans la majoration $C_{\text{nf}}(X; c) \leq \left\lceil \frac{2\mathscr{V}(X)}{c} \right\rceil$ est optimal : pour $\varepsilon > 0$ et $n \geq 1$, on peut construire une instance $(X_1, X_2, \ldots, X_p; c)$ vérifiant $\frac{C_{\text{nf}}(X; c)}{\omega(X; c)} > 2 - \varepsilon$ et $p \geq n$.

2.2. *Analyse de la stratégie* first fit

► La stratégie *first fit* consiste à loger chaque objet dans le conteneur de plus petit numéro susceptible de l'accueillir ; à défaut, on alloue un nouveau conteneur. Voici une description imagée de cette stratégie : les objets arrivent entre les mains d'un manutentionnaire, en tête d'un train dont chaque wagon porte un conteneur ; le manutentionnaire se dirige vers la queue du train en inspectant les wagons, à la recherche du premier conteneur offrant une place suffisante ; il y loge l'objet, puis retourne à la tête du train. Lorsque tous les objets ont été chargés, il repart vers la queue du train, et, au besoin, sépare le dernier wagon ayant reçu un objet du premier wagon n'en ayant reçu aucun (pour éviter à la locomotive de tracter des wagons vides).

► Nous noterons $C_{\text{ff}}(X; c)$ le coût du rangement obtenu par application de la stratégie *first fit* à l'instance $(X; c)$. Avec cette stratégie, tous les conteneurs (sauf peut-être un) sont remplis à plus de 50%. On peut construire une instance $(X; c)$ telle que $\frac{C_{\text{ff}}(X; c)}{\omega(X; c)} = \frac{5}{3}$.

2.3. *Analyse de la stratégie* first fit *avec une instance décroissante*

► Nous dirons qu'une instance $(X; c)$ est *décroissante* lorsque $X_1 \geq X_2 \geq \ldots \geq X_n$. On analyse dans cette partie la qualité du rangement obtenu lorsque la stratégie *first fit* est appliquée à une telle instance.

► Soit $(X; c)$ une instance décroissante pour laquelle un rangement optimal requiert p conteneurs. Nous allons établir une majoration du nombre de conteneurs supplémentaires nécessaires lorsque l'on applique la stratégie *first fit*. Nous supposons donc $C_{\text{ff}}(X; c) > p$. Voici une démarche :

- montrez que les conteneurs d'indice supérieur à p ne contiennent que des objets de volume au plus égal à $c/3$;
- montrez qu'à eux tous, ces conteneurs contiennent au plus $p - 1$ objets ;
- pour $p \geq 2$, montrez que $C_{\text{ff}}(X; c) \leq \frac{4p + 1}{3}$;

– qu'en est-il du cas $p = 1$?

3. Résultats théoriques

► On vérifie facilement que le problème étudié est dans la classe NP.

► Ce problème est (hélas) NP-complet. En effet, on sait que le problème PARTITION (qui consiste à trouver une partition de l'ensemble $\{a_1, \ldots, a_n\}$ d'entiers en deux sous-ensembles de même somme) est NP-complet.

Exercice de programmation :

► *Il vous est demandé de rédiger un programme conforme aux spécifications ci-dessous dans l'un des langages C, Caml ou Java à votre choix. Ce programme devra être accompagné d'un exemple d'exécution permettant d'en vérifier le bon fonctionnement. La clarté et la concision du programme seront des éléments importants d'appréciation pour le jury.*

Programmez la mise en œuvre des stratégies *next fit* et *first fit*.

Suggestions pour le développement

► *Soulignons qu'il s'agit d'un menu à la carte et que vous pouvez choisir d'étudier certains points, pas tous, pas nécessairement dans l'ordre, et de façon plus ou moins fouillée. Vous pouvez aussi vous poser d'autres questions que celles indiquées plus bas. Il est très vivement souhaité que vos investigations comportent une partie traitée sur ordinateur et, si possible, des représentations graphiques de vos résultats.*

– donnez des exemples de problèmes pratiques qui se ramènent au problème étudié ici ;
– pour la partie 1.2 :
  – calculez $\omega(X)$ sur un exemple de petite taille ;
  – construisez les exemples indiqués en fin de cette partie ;
– pour la partie 2.1 :
  – prouvez la majoration et son optimalité ;
– pour la partie 2.2 :
  – construisez une instance réalisant l'égalité indiquée ;
– pour la partie 2.3 :
  – construisez une instance décroissante X de longueur 6, vérifiant $\omega(X;c) = 2$ et $C_{\text{ff}}(X;c) = 3$;
  – construisez une instance décroissante X de longueur au moins n, vérifiant $\omega(X;c) = 2$ et $C_{\text{ff}}(X;c) = 3$;
  – démontrez la majoration indiquée ;
– pour la partie 3 :
  – prouvez que le problème est dans la classe NP ;

() Option informatique

– décrivez la réduction indiquée.