GIPAD – GS2 – Recherche Opérationnelle
Correction devoir final APP2

2 décembre 2009

---

1 Modèles de packing

Problème 1 *Une grille de calcul est composée d'ordinateurs interconnectés sur lesquelles peuvent s'exécuter des applications informatiques. Dans la suite, on suppose que tous les ordinateurs sont disponibles pour exécuter un ensemble donné d'applications, toutes de durées identiques. À chaque application, est associée une valeur de priorité d'exécution.*

1. ordinateurs mono-processeurs / tâches prioritaires
2. ordinateurs multi-processeurs / ordinateurs OU tâches prioritaires
3. ordinateurs multi-processeurs + mémoire partagée / ordinateurs prioritaires
4. ordinateurs multi-processeurs + mémoire partagée / applications distribuées / ordinateurs ET tâches prioritaires

---

2 Capacitated Facility Location Problem (CFL)

Question 1 ***(6+2 points)***

Q1.1. (2) modéliser ce problème par un programme linéaire en nombres entiers (CFL) ;

Q1.2. (2) formuler la relaxation lagrangienne $L(\mathsf{CFL}, \mu)$ de (CFL) de paramètre $\mu \in \mathbb{R}^n_+$ obtenue par dualisation des contraintes d'approvisionnement des clients ;

Q1.3. (1) formuler le dual lagrangien de (CFL) correspondant à cette dualisation ;

Q1.4. (1) montrer que $L(\mathsf{CFL}, \mu)$ se décompose en m sous-problèmes indépendants notés $L_j(\mathsf{CFL}, \mu)$ pour tout $j = 1, \dots, m$;

Q1.5. (2*) déterminer la valeur optimale de $L_j(\mathsf{CFL}, \mu)$, en déduire un algorithme pour résoudre $L(\mathsf{CFL}, \mu)$ et donner sa complexité.

3 Programmation dynamique : alignement de séquences ADN

Problème 2 alignement de séquences ADN.
Un fragment d'ADN consiste en une séquence finie d'éléments de l'ensemble $\Sigma = \{A, C, G, T\}$. Le problème d'alignement de deux fragments X et Y consiste à déterminer une sous-suite commune de X et de Y de longueur maximale.

On cherche à calculer $Z = z_1 z_2 \dots z_k$, une plus longue sous-suite commune de deux fragments $X = x_1 x_2 \dots x_m$ et $Y = y_1 y_2 \dots y_n$, par programmation dynamique. On utilisera pour cela les éléments de notation et de proposition suivants :

Proposition 1 *Soient $1 \leqslant i \leqslant m$ et $1 \leqslant j \leqslant n$ et soit $Z = z_1 z_2 \dots z_k$ une plus longue sous-suite commune de X_i et Y_j, alors :*

1. *si $x_i = y_j$, alors $z_k = x_i$ et Z_{k-1} est une plus longue sous-suite commune de X_{i-1} et Y_{j-1} ;*
2. *si $x_i \neq y_j$, alors Z_k est une plus longue sous-suite commune de : soit X_{i-1} et Y_j, soit X_i et Y_{j-1} ;*

Question 2 ***(6+3 points)***
Pour tout $1 \leqslant i \leqslant m$ et $1 \leqslant j \leqslant n$, on note $c_{i,j}$ la plus grande longueur de sous-suite commune de $X_i = x_1 x_2 \dots x_i$ et de $Y_j = y_1 y_2 \dots y_j$.

Q2.1. (1) montrer que $c_{i,j} = \max(c_{i,j-1}, c_{i-1,j})$ si $x_i \neq y_j$, pour tout $1 \leqslant i \leqslant m$ et $1 \leqslant j \leqslant n$;

Q2.2. (2) exprimer par une relation de récurrence sur $0 \leqslant i \leqslant m$ et $0 \leqslant j \leqslant n$, la valeur $c_{i,j}$: la longueur de la plus longue sous-suite commune de $X_i = x_1 x_2 \dots x_i$ et de $Y_j = y_1 y_2 \dots y_j$.

Q2.3. (3) proposer un algorithme de programmation dynamique de calcul de la longueur de la plus longue sous-suite commune de X et Y, et donner sa complexité ;

Q2.4. (3*) proposer un algorithme de calcul d'une plus longue sous-suite Z et de l'alignement optimal (X^Z, Y^Z) correspondant.

Correction Problème 1.

On note I l'ensemble des ordinateurs, J l'ensemble des applications ; p_i le nombre de processeurs et c_i la capacité mémoire de l'ordinateur i ; w_j la consommation mémoire et r_j la priorité de l'application j.

1. Problème de sac-à-dos 0-1 (avec poids unitaires) :

$$\max\{ \sum_{j\in J} r_j x_j \mid \sum_{j\in J} x_j \leqslant |I|, x \in \{0,1\}^J \}$$

Comme les poids sont tous unitaires, il s'agit également d'un problème de couplage de poids maximal dans le graphe biparti complet $(J \times I, r)$:

$$\max\{ \sum_{j\in J}\sum_{i\in I} r_j x_{ij} \mid \sum_{j\in J} x_{ij} \leqslant 1\ (\forall i \in I),\ \sum_{i\in I} x_{ij} \leqslant 1\ (\forall j \in J),\ x \in \{0,1\}^{I\times J} \}$$

Il se résoud à l'optimum en temps $O(|J| \ln |J|)$ en triant les applications par priorité décroissante et en sélectionnant les $|I|$ premières.

2. Le problème se décompose en 2 selon que $\sum_{i\in I} p_i \geqslant |J|$ (toutes les applications sont exécutables simutanément) ou non. Dans le premier cas, il s'agit d'un problème de Bin-Packing :

$$\min\{ \sum_{i\in I} y_i \mid \sum_{j\in J} x_{ij} \leqslant p_i y_i\ (\forall i \in I),\ \sum_{i\in I} x_{ij} = 1\ (\forall j \in J),\ x \in \{0,1\}^{I\times J},\ y \in \{0,1\}^I \}$$

Dans le second cas, il s'agit d'un Problème de sac-à-dos multidimensionnel 0-1 (0-1 Multi-Knapsack) :

$$\max\{ \sum_{j\in J}\sum_{i\in I} r_j x_{ij} \mid \sum_{j\in J} x_{ij} \leqslant p_i\ (\forall i \in I),\ \sum_{i\in I} x_{ij} \leqslant 1\ (\forall j \in J),\ x \in \{0,1\}^{I\times J} \}$$

3. Problème de Bin-Packing 2D :

$$\min\{ \sum_{i\in I} y_i \mid \sum_{j\in J} x_{ij} \leqslant p_i y_i\ (\forall i \in I),\ \sum_{j\in J} w_j x_{ij} \leqslant c_i y_i\ (\forall i \in I),\ \sum_{i\in I} x_{ij} = 1\ (\forall j \in J),\ x \in \{0,1\}^{I\times J},\ y \in \{0,1\}^I \}$$

4. Il s'agit d'un problème bi-objectif que l'on va simplement linéarisé ici comme une somme pondérée des deux coûts. Dans le premier objectif, on considère une variante du Bin-Packing, dans laquelle les containers (les ordinateurs) possèdent deux dimensions (nombre de processeurs et capacité mémoire) ainsi qu'un coût fixe d'utilisation. Dans le second objectif, on considère une variante du 0-1 multi-knapsack à deux dimensions, dans laquelle les items (les composants) sont liés entre eux : soit tous les composants d'une même application sont exécutés, soit aucun. On note K_j l'ensemble des composants d'une application j ; w_k^j la consommation de mémoire du composant $k \in K_j$; b_i le coût de fonctionnement de l'ordinateur i.

$$\begin{aligned}
\max\ & \sum_{j\in J} r_j z_j + \sum_{i\in I} b_i y_i \\
\text{s.t.}\ & \sum_{(j,k)\in J\times K_j} x_{ik}^j \leqslant p_i y_i && \forall i \in I, \\
& \sum_{(j,k)\in J\times K_j} w_k^j x_{ik}^j \leqslant c_i y_i && \forall i \in I, \\
& \sum_{i\in I} x_{ik}^j = z_j && \forall (j,k) \in J \times K_j, \\
& x_{ik}^j \in \{0,1\}, y_i \in \{0,1\} && \forall i \in I, \forall (j,k) \in J \times K_j, \\
& z_j \in \{0,1\} && \forall j \in J
\end{aligned}$$

Correction Problème 2.

1. On note $I = \{1, \ldots, n\}$ l'ensemble des clients et $J = \{1, \ldots, m\}$ l'ensemble des sites. On définit pour chaque site $j \in J$, une variable booléenne $y_j = 1$ si le site j est ouvert, $y_j = 0$ sinon. La variable x_{ij} indique la quantité de produit livrée au client $i \in I$ depuis l'entrepôt $j \in J$.

$$
\begin{aligned}
(CFL): z \; = \; & \min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} f_j y_j \\
\text{s.t. } & \sum_{j \in J} x_{ij} \geqslant a_i && \forall i \in I, \\
& \sum_{i \in I} x_{ij} \leqslant b_j y_j && \forall j \in J, \\
& x_{ij} \in \mathbb{R}_+ && \forall i \in I, \forall j \in J, \\
& y_j \in \{0, 1\} && \forall j \in J.
\end{aligned}
$$

2. Pour tout multiplicateur $\mu \in \mathbb{R}_+^I$:

$$
\begin{aligned}
L(CFL, \mu): z_\mu \; = \; & \min \sum_{i \in I} \sum_{j \in J} (c_{ij} - \mu_i) x_{ij} + \sum_{j \in J} f_j y_j + \sum_{i \in I} \mu_i a_i \\
\text{s.t. } & \sum_{i \in I} x_{ij} \leqslant b_j y_j && \forall j \in J, \\
& 0 \leqslant x_{ij} \leqslant a_i && \forall i \in I, \forall j \in J, \\
& y_j \in \{0, 1\} && \forall j \in J.
\end{aligned}
$$

   On note que la contrainte $x_{ij} \leqslant a_i$ n'est naturellement pas présente dans la relaxation lagrangienne de (CFL). Cependant celle-ci permet de renforcer la relaxation sans la complexifier.

3. Le dual lagrangien s'écrit $D = \max\{z_\mu \mid \mu \in \mathbb{R}_+^I\}$.

4. $L(CFL, \mu)$ se décompose en m sous-problèmes, un pour chaque site : $z_\mu = \sum_{i \in I} \mu_i a_i + \sum_{j \in J} u_\mu^j$ où u_μ^j est défini pour tout $j \in J$ par :

$$
L_j(CFL, \mu): u_\mu^j = \min\{ \sum_{i \in I} (c_{ij} - \mu_i) x_{ij} + f_j y_j \mid \sum_{i \in I} x_{ij} \leqslant b_j y_j, 0 \leqslant x_i \leqslant a_i (\forall i \in I), y_j \in \{0, 1\} \}
$$

5. Soit $(x_j, y_j) \in \mathbb{R}^n \times \{0, 1\}$ une solution optimale de $L_j(CFL, \mu)$. Si $y_j = 0$ (le site n'est pas ouvert) alors $x_{ij} = 0 \; \forall i \in I$. Sinon, $y_j = 1$ et x est la solution optimale du problème de sac-à-dos continu borné suivant, où tous les poids sont unitaires :

$$
max\{ \sum_{i \in I} (\mu_i - c_{ij}) x_i \mid \sum_{i \in I} x_i \leqslant b_j, 0 \leqslant x_i \leqslant a_i (\forall i \in I) \}
$$

   Ce problème se résoud à l'optimum en triant les clients par valeurs $(\mu_i - c_{ij})$ décroissantes. On sélectionne alors les $k + 1$ premiers clients i tels que : $\mu_i - c_{ij} > 0$ et $\sum_{i=1}^{k} a_i \leqslant b_j < \sum_{i=1}^{k+1} a_i$. On pose alors $x_i = a_i$ pour tout $i = 1, \ldots, k$, $x_{k+1} = b_j - \sum_{i=1}^{k} a_i$, et $x_i = 0$ pour tout $i > k + 1$. On note I_μ^j l'ensemble des k premiers clients et i_μ^j le $k + 1$-ème client. Ainsi, la valeur optimale de $L_j(CFL, \mu)$ est égale à $u_\mu^j = \min\{0, \sum_{i \in I_\mu^j} (c_{ij} - \mu_i) a_i + (c_{i_\mu^j j} - \mu_{i_\mu^j})(b_j - \sum_{i \in I_\mu^j} a_i) + f_j\}$ et se calcule en temps $O(n \ln n)$. Par conséquent, chaque sous-problème $L(CFL, \mu)$ peut être résolu optimalement en temps $O(mn \ln n)$.

Correction Problème 3.

1. les sous-suites communes de X_i et Y_{j-1} sont des sous-suites communes de X_i et Y_j, donc $c_{ij} \geqslant c_{i,j-1}$. Par symétrie, on a $c_{ij} \geqslant \max(c_{i,j-1}, c_{i-1,j})$. Par ailleurs, la condition 2 de la proposition 1 indique que si $x_i \neq y_j$ alors $c_{i,j} = c_{i,j-1}$ ou bien $c_{i,j} = c_{i-1,j}$. Donc, $c_{i,j} = \max(c_{i,j-1}, c_{i-1,j})$.
2. **Ne pas oublier les conditions initiales :**

$$c_{i,j} = \begin{cases} 0 & \text{si } i = 0 \text{ ou } j = 0 \\ c_{i-1,j-1} + 1 & \text{si } 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant n \text{ et } x_i = y_j \\ \max(c_{i-1,j}, c_{i,j-1}) & \text{si } 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant n \text{ et } x_i \neq y_j \end{cases}$$

   Par définition, $0 \leqslant c_{i,j} \leqslant \min(i, j)$ donc $c_{i,j} = 0$ si $i = 0$ ou $j = 0$. La condition 1 de la proposition 1 se traduit par : si $x_i = y_j$ alors $c_{i-1,j-1} = c_{i,j} - 1$.
3. le calcul de $c_{m,n}$ par cette formule s'effectue trivialement par une double-boucle sur i et j en $O(mn)$ (step 1 de l'algorithme). Chaque couple (i, j) correspond à un état de programmation dynamique.

   step 0. initialisation :
   pour $i = 0, \ldots, m$ faire $c_{i,0} = 0$
   pour $j = 1, \ldots, n$ faire $c_{0,j} = 0$

   step 1. boucle de récursion :
   pour $i = 1, \ldots, m$, $j = 1, \ldots, n$ faire
   si $x_i = y_j$ alors $c_{i,j} = c_{i-1,j-1} + 1$; $u_{i,j}$=0 ;
   sinon si $c_{i-1,j} \geqslant c_{i,j-1}$ alors $c_{i,j} = c_{i-1,j}$; $u_{i,j}$=1 ;
   sinon $c_{i,j} = c_{i,j-1}$; $u_{i,j}$=2 ;
   fin pour.

   step 2. calcul de Z, X^Z, Y^Z :
   $i = m$; $j = n$; $Z = \emptyset$; $X^Z = \emptyset$; $Y^Z = \emptyset$
   tant que $i > 0$ ou $j > 0$ faire
   si j=0 ou $u_{i,j}$=1 alors $X^Z = x_i X^Z$; $Y^Z = *Y^Z$; i- - ;
   si i=0 ou $u_{i,j}$=2 alors $X^Z = *X^Z$; $Y^Z = y_j Y^Z$; j- - ;
   si $u_{i,j}$=0 alors $Z = x_i Z$; $X^Z = x_i X^Z$; $Y^Z = y_j Y^Z$; i- - ; j- - ;
   fin tant que

   step 3. retourne $c_{m,n}$, X^Z, Y^Z, et Z.
4. Une plus longue sous-suite Z et l'alignement (X^Z, Y^Z) peuvent être construits, à l'envers, en parcourant les états depuis (m, n) vers $(0, 0)$ (step 2 de l'algorithme). Cet étape possède un nombre fini d'itérations car la somme $i + j$ décroît à chaque itération. La condition d'arrêt $i = j = 0$ est nécessairement rencontrée car $i + j$ décroît de 2 seulement si $i + j \geqslant 2$ et de 1 autrement. À l'itération initiale $l = 0$, on a $i = m$, $j = n$, $Z = \emptyset$, $X^Z = \emptyset$, $Y^Z = \emptyset$. On montre aisément par récurrence sur $l \geqslant 0$, qu'au début de chaque itération, on a :
   - $X'_i = x_{i+1} \ldots x_m$ est une sous-suite de X^Z,
   - $Y'_j = y_{j+1} \ldots x_n$ est une sous-suite de Y^Z,
   - Z est une plus longue sous-suite commune de X'_i et Y'_j (en appliquant la proposition à (X'_i, Y'_j) au lieu de (X_i, Y_j)).

   À l'itération finale, Z est une plus longue sous-suite de $X'_0 = X$ et $Y'_0 = Y$.