École polytechnique Majeure informatique

Cours *conception et analyse d'algorithmes*
Correction du TD 4

1. Plus court chemin

i. Considérons un chemin $\gamma = (x_i)_{1 \leq i \leq n}$ avec $x_1 = s$, $x_n = t$ et $(x_i, x_{i+1}) \in E$ pour tout $i < n$. Si $d_0(v)$ est une solution aux contraintes du problème linéaire ci-dessus, alors on a

$$d_0(t) - d_0(s) = \sum_{i=1}^{n-1} d_0(x_{i+1}) - d_0(x_i) \leq \sum_{i=1}^{n-1} c(x_i, x_{i+1}),$$

qui est exactement la longueur du chemin γ. Il s'ensuit que $d_0(t) - d_0(s)$ minore en particulier la longueur du plus court chemin de s à t dans G.

ii. Inversement, si l'on prend comme valeurs des variables $d(v)$ les longueurs $\delta(v)$ des plus courts chemins de s à v, on a manifestement $d(v) - d(u) \leq c(u, v)$, sans quoi on obtient un chemin plus court pour aller en v en suivant un plus court chemin vers u puis en prenant l'arête (u, v) ; les valeurs $d(v) = \delta(v)$ vérifient donc le système de contraintes.

iii. De la question (i) on déduit $\max(d(t) - d(s)) \geq \delta(t) - \delta(s) = \delta(t)$. De plus, d'après la question (ii) on a $\max(d(t) - d(s)) \leq \delta(t)$. Il y a donc égalité (dans les deux dernières inégalités, le maximum est bien entendu pris sur l'ensemble des solutions au système de contraintes).

iv. Le dual du problème linéaire s'écrit, quant à lui, en associant à chaque arête une variable $f(u, v)$,

$$\min \sum_{(u,v) \in E} c(u, v) f(u, v)$$

sous les contraintes :

$$\begin{aligned}
\sum_{(v,u) \in E} f(u, v) - \sum_{(u,v) \in E} f(v, u) &= 0, \forall u \in V - \{s, t\} \\
\sum_{(v,s) \in E} f(v, s) - \sum_{(s,v) \in E} f(s, v) &= -1 \\
\sum_{(v,t) \in E} f(v, t) - \sum_{(t,v) \in E} f(t, v) &= 1 \\
f(u, v) &\geq 0.
\end{aligned}$$

Les contraintes sont des contraintes de type flot, hormis le fait qu'il peut exister un flot sortant de t ou entrant en s, et que toutes les arêtes sont de capacité infinie.

La première contrainte traduit la conservation du flot, les deux suivantes indiquant que le flot total sortant de s vaut 1 et le flot total entrant en t vaut 1. Il ne s'agit donc pas d'un problème de flot maximal (voir exercice 4), mais d'un problème ou la quantité de flot est

fixée (ici unitaire), et où le flot est pondéré par des coefficients $c(u, v)$, et on veut que son coût soit minimal ; ce type de problème est appelé *min-cost flow*. L'exercice montre que ce problème se réduit, par dualité, à un problème de plus court chemin. Noter le parallèle avec Ford-Fulkerson, où l'augmentation du flot se réduisait à un problème d'atteignabilité.

2. Adéquation linéaire

i. On doit minimiser la valeur de $\max\{y_i - ax_i - b,\ i = 1 \dots n\}$ sur toutes les valeurs possibles des coefficients a et b définissant la droite $y = ax + b$.

ii. On introduit trois variables a, b, m, où m est le maximum des distances entre les points et la droite $y = ax + b$. On obtient alors le système de contraintes $-ax_i - b + m \geq -y_i$ et $ax_i + b + m \geq y_i$. La fonctionnelle linéaire à minimiser est tout simplement m. La seule contrainte de signe est $m \geq 0$. On notera C_i la i-ème contrainte du premier type, C_{i+n} la i-ème contrainte du second type.

iii. Notons que ce problème linéaire a peu de variables mais énormément de contraintes. Le dual, quant à lui, s'écrit

$$\begin{cases} \max \sum y_i(z_{i+n} - z_i), \\ \sum x_i(z_{i+n} - z_i) = 0, \\ \sum z_{i+n} - z_i = 0, \\ \sum z_{i+n} + z_i \leq 1. \end{cases}$$

Il est possible, quand le nombre de points est grand, que sa géométrie soit plus simple, et qu'il soit par conséquent plus facile à résoudre par la méthode du simplexe.

3. Ordonnancement préemptif sur des machines parallèles

i. Soit $D = \max\left(\max p_i, \frac{\sum p_i}{m}\right)$. Il est clair que D est une borne inférieure de la date de fin. Nous décrivons maintenant un algorithme qui renvoie un ordonnancement réalisable de durée égale à D.

On note $M_1, \dots, M_m$ les machines. À l'initialisation, la machine courante M_c est M_1 et la date courante d est 0. Pour $i = 1$ à n, on calcule la date $f = d + p_i$. Si $f \leq D$, alors on prévoit d'exécuter la tâche i sur la machine courante M_c entre les dates d et f. On met à jour la date courante qui devient f. Si au contraire $f > D$, on prévoit d'exécuter la tâche i en deux partie : une partie sur la machine courante M_c entre les dates d et D et une partie sur la machine M_{c+1} entre les dates 0 et $f - D = p_i - (D - d)$. Cette affectation est valide car la condition $D \geq pi$ assure que les parties à effectuer sur les machines M_c et M_{c+1} ne se chevauchent pas. On met à jour la machine courante qui devient M_{c+1} et la date courante qui devient $f - D$. Clairement, le numéro de la machine courante après avoir affecté la tâche i est $\lceil \sum_{k=1}^{i} p_k / D \rceil$. Donc le numéro de la machine courante ne dépasse pas m par définition de la durée D.

ii. On classe les dates d_i et f_i de manière croissante, c'est-à-dire qu'on détermine $\theta_1 \leq \theta_2 \leq \dots \leq \theta_{2n}$ tels que chaque θ_k correspond à un d_j ou à un f_j. Soit $x_{i,k}$ ($1 \leq i \leq n$ et $1 \leq k < 2n$) la longueur de la tâche i exécutée entre les dates θ_k et θ_{k+1}.

Les contraintes évidentes pour une exécution sont les suivantes.
Les contraintes de durée totale des tâches s'écrivent :

$$\forall i \in \{1, \dots, n\}, \quad \sum_{1 \leq k < 2n} x_{i,k} = p_i.$$

Les contraintes de fenêtre de temps s'écrivent :

$$\forall i \in \{1,\dots,n\},\ \forall k \in \{1,\dots,2n\},\ x_{i,k} = 0 \text{ si } \theta_{k+1} \leq d_i \text{ ou } \theta_k \geq f_i.$$

De plus, sur chaque intervalle, la durée maximale d'une tâche est limitée par la longueur de l'intervalle :

$$\forall i \in \{1,\dots,n\},\ \forall k \in \{1,\dots,2n-1\},\ x_{i,k} \leq \theta_{k+1} - \theta_k,$$

et la charge totale est limitée par la charge disponible :

$$\forall i \in \{1,\dots,n\},\ \forall k \in \{1,\dots,2n\},\ \sum_{1\leq i\leq n} x_{ik} \leq m(\theta_{k+1} - \theta_k).$$

Réciproquement, si on trouve des $x_{i,k}$ satisfaisant toutes ces inégalités, il est possible, d'après la première question, de construire un ordonnancement réalisable. On utilise en effet l'algorithme décrit en (i) sur chaque intervalle $[\theta_k, \theta_{k+1}]$.

iii. On obtient un algorithme d'optimisation en remplaçant la contrainte fixant la valeur de θ_{2n} par l'inégalité $\theta_{2n-1} \leq \theta_{2n}$. La valeur de $F = \theta_{2n}$ n'est plus alors une date fixe et représente la date de fin de l'ordonnancement. On minimise $F = \theta_{2n}$ sous les contraintes ci-dessus. Si ce problème linéaire admet une solution telle que $\theta_{2n-1} = \theta_{2n}$, on fait la même chose en imposant $\theta_{2n-2} \leq \theta_{2n-1} = \theta_{2n} = F$ et ainsi de suite jusqu'à ce qu'on arrive a un problème sans solution. La date de fin minimale correspond à la dernière solution obtenue.

4. Égalités forcées

1. Pour tout $j \notin F$ il existe une solution x^j telle que $a_{j1}x_1^j + \ldots + a_{jn}x_n^j < b_j$. Soit $k = m - |F|$. Le barycentre $x^* = \frac{1}{k}\sum_{j\notin F} x^j$ est un point admissible par convexité et il satisfait la condition requise.

2. Pour chaque $j \in \{1,\dots,m\}$ on considère le problème de programmation linéaire donné par les contraintes S et la fonction d'objectif $\min_x a_{j1}x_1 + \ldots + a_{jn}x_n$. Si la solution x^j renvoyée vérifie l'égalité l'équation est forcée à l'égalité, sinon on la garde pour obtenir à la fin x^* en prenant un barycentre.