École polytechnique — Majeure informatique
Promotion 2008

Cours *conception et analyse d'algorithmes*
TD 4 – Programmation linéaire

6 octobre 2010

1. Plus court chemin.

Etant donné un graphe orienté G avec arêtes valuées par une fonction positive c, et deux sommets s et t de G, on souhaite trouver un plus court chemin de s à t dans G.

On associe à chaque sommet v une valeur $d(v)$. On considère le problème linéaire suivant :

$$\max(d(t) - d(s))$$

sous la contrainte que pour toute arête $e = (u, v)$, $d(v) - d(u) \leq c(u, v)$.

i. Montrer que pour toute fonction d satisfaisant aux contraintes de ce PL, $d(t) - d(s)$ est une borne inférieure à la longueur du plus court chemin de s à t dans G.

ii. Montrer que la fonction d égale à la longueur d'un plus court chemin de s à v satisfait aux contraintes du PL.

iii. En déduire que la solution optimale du problème linéaire est exactement égale à la longueur du plus court chemin de s à t.

iv. Ecrire le dual du problème linéaire, et l'interpréter.

2. Adéquation linéaire

Un physicien prend des mesures d'une fonction $y(x)$ qu'il sait linéaire. Les résultats sont donnés sous la forme de couples (x_i, y_i), qui sont les paramètres d'entrée du problème. Il souhaite trouver la droite qui "correspond le mieux" à ces données, en ce sens que la distance *verticale* maximale entre un point (x_i, y_i) et la droite soit la plus petite possible.

i. Modéliser ce problème comme un problème d'optimisation de type "min max" avec contraintes linéaires.

ii. Le réduire à un problème de programmation linéaire. Ecrire le problème dual.

iii. Pourquoi peut-il être préférable d'en résoudre le dual ?

3. Ordonnancement préemptif sur des machines parallèles

Un ensemble de tâches $\{1, \cdots, n\}$ doit être exécuté sur m machines parallèles. Chaque tâche a une durée p_i et peut être exécutée de manière préemptive, c'est-à-dire avec interruptions : l'exécution de la tâche i peut commencer sur une machine, être interrompue et continuer ultérieurement sur une autre machine, sachant toutefois que deux parties d'une même tâche ne peuvent être exécutées en même temps sur des machines différentes.

i. Montrer que la durée totale minimale d'un tel ordonnancement est égale à

$$\max\left(\max_{1\leq i\leq n} p_i, \frac{\sum_{i=1}^{n} p_i}{m}\right)$$

et qu'un tel ordonnancement peut être déterminé par un algorithme ayant une complexité $O(n)$.

ii. Chaque tâche a maintenant une fenêtre horaire $[d_i, f_i]$ pendant laquelle elle doit être entièrement exécutée. Donner un système linéaire ayant une taille polynomiale en n indépendante de m qui permet de déterminer s'il est possible d'exécuter toutes les tâches.

iii. En déduire un algorithme de minimisation de la durée totale de l'ordonnancement.

4. Égalités forcées

Étant donné un système S d'inéquations

$$\begin{aligned} a_{11}x_1 + \ldots + a_{1n}x_n &\leq b_1 \\ &\vdots \\ a_{m1}x_1 + \ldots + a_{mn}x_n &\leq b_m \end{aligned}$$

on dit que la $j$ème inéquation est forcée à l'égalité si toute solution x du système est telle que $a_{j1}x_1 + \ldots + a_{jn}x_n = b_j$. Soit F_S l'ensemble des indices des inéquations forcées à l'égalité de S.

1. Montrer qu'il existe une solution x^* de S telle que :

$$a_{j1}x_1^* + \ldots + a_{jn}x_n^* = b_j \text{ si et seulement si } j \in F.$$

Indication : l'ensemble des solutions de S est convexe.

2. Donner un algorithme, basé sur la programmation linéaire, qui permet de déterminer quelles sont les équations forcées à l'égalité dans le système S et qui donne une solution x^* satisfaisant la condition de la question précédente.
Indications : Un problème de programmation linéaire est constitué d'un ensemble de contraintes et d'une fonction objectif. Il pourra être utile de résoudre plusieurs problèmes de programmation linéaire pour arriver au résultat.