CONTRÔLE DE RECHERCHE OPÉRATIONNELLE

FÉVRIER 2015

1. PLUS COURT CHEMIN – 2 PTS

Le résultat s'obtient par une simple application de l'algorithme de Dijkstra. Au moins quelques étapes de l'application de l'algorithme étaient attendues dans les copies.

La s-t chaîne la plus courte est de longueur 15 et est indiquée sur la Figure 1.

2. AFFECTATION DE TÂCHES, COUPLAGE ET MAKESPAN – 3 PTS

1. On considère P l'ensemble des poids apparaissant sur les arêtes : $P = \{w(e) : e \in E\}$. On numérote ces poids du plus grand au plus petit $P = \{w_1, \dots, w_k\}$, avec $w_1 > \cdots > w_k$.

On définit une suite de graphe $G^0, G^1, \dots$ de la manière suivante. $G^0 = G$. Le graphe G^1 s'obtient en retirant de G^0 toutes les arêtes de poids w_1. Si G^1 ne possède pas de couplage de cardinalité $\nu(G)$, on s'arrête, et tout couplage de cardinalité $\nu(G)$ de G est solution optimale. Si non, on répète l'opération en retirant de G^1 toutes les arêtes de poids w_2, etc.

Plus généralement, le graphe G^i s'obtient à partir de G^{i-1} en retirant toutes les arêtes de poids w_i. Si G^i ne possède pas de couplage de cardinalité $\nu(G)$, alors tout couplage de cardinalité $\nu(G)$ de G^{i-1} est solution optimale (de valeur w_{i-1}) et on s'arrête. Si non, on continue.

On trouve bien ainsi le couplage M de cardinalité $\nu(G)$ minimisant $\max_{e\in M} w(e)$. Soit w^* la valeur minimale que peut avoir $\max_{e\in M} w(e)$. Tant que $w_i < w^*$, le graphe G^i possède un couplage de cardinalité $\nu(G)$. Quand $w_i = w^*$, par définition de w^*, le graphe G^i ne possède pas de couplage de cardinalité $\nu(G)$.

2. Considérons le graphe biparti $G = (V, E)$ dont l'ensemble des sommets est $W \cup T$ et dont les arêtes sont formées par toutes les paires wt avec $w \in W$ et $t \in T$. On veut trouver un couplage M couvrant W tel que $\max_{wt\in E} d(w,t)$ soit le plus petit possible. En effet, cette quantité est le temps total requis pour la réalisation de l'ensemble des tâches dans l'affectation encodé par le couplage M. On est exactement dans le cas traité en 1. avec $\nu(G) = |Y|$. Comme le graphe G est biparti complet et que $|T| \le |W|$, on sait qu'un couplage de cardinalité $|W|$ existe.

3. RELAXATION LAGRANGIENNE POUR UN PROBLÈME D'ENTREPÔTS – 11 PTS

2. On a

$$\begin{array}{rcll} \mathscr{G}(\boldsymbol{\lambda}) & = & \min & \displaystyle\sum_{i\in I, j\in J} (a_{ij}+\lambda_i)x_{ij} + \sum_{j\in J} c_j y_j - \sum_{i\in I}\lambda_i \\ & & \text{s.c.} & \displaystyle\sum_{i\in I} d_i x_{ij} \le k_j y_j & \forall j \in J \\ & & & x_{ij} \in [0,1] & \forall i \in I, j \in J \\ & & & y_j \in \{0,1\} & \forall j \in J. \end{array}$$

Ce programme d'optimisation est séparable en j. Il suffit donc de vérifier que tout programme de la forme suivante se résout en temps polynomial :

$$\begin{array}{lll} \min & \sum_{i\in I}(a_i+\lambda_i)x_i + cy & \\ \text{s.c.} & \sum_{i\in I} d_i x_i \le ky & \\ & x_i \in [0,1] & \forall i \in I \\ & y \in \{0,1\}. & \end{array}$$

C'est bien le cas : la solution optimale pour $y = 1$ se calcule en temps polynomial car cela devient alors un programme linéaire (par les points intérieurs, ou même par une méthode directe) ; la valeur optimale pour $y = 0$, c'est 0.

3 .On a

$$\begin{array}{rcll} \mathcal{H}(\boldsymbol{\mu}) & = & \min & \sum_{i\in I, j\in J}(a_{ij}+d_i\mu_j)x_{ij} + \sum_{j\in J}(c_j - k_j\mu_j)y_j \\ & & \text{s.c.} & \sum_{j\in J} x_{ij} = 1 \qquad \forall i \in I \\ & & & x_{ij} \in [0,1] \qquad \forall i \in I, j \in J \\ & & & y_j \in \{0,1\} \qquad \forall j \in J. \end{array}$$

Ce programme se résout bien en temps polynomial. Pour chaque i, on définit $j(i)$ comme étant un indice j minimisant $a_{ij} + d_i\mu_j$. On a alors

$$\mathcal{H}(\boldsymbol{\mu}) = \sum_{i\in I}(a_{ij(i)} + d_i\mu_{j(i)}) + \sum_{j\in J}\min(0, c_j - k_j\mu_j).$$

4. On a

$$\begin{array}{rcll} \mathcal{G}(\lambda) & = & \min & (a+\lambda)x + cy - \lambda \\ & & \text{s.c.} & dx \le ky \\ & & & x \in [0,1] \\ & & & y \in \{0,1\}. \end{array}$$

Si $\lambda \le -a$, on a $\mathcal{G}(\lambda) = \min(-\lambda, (a+\lambda) + c - \lambda)$. Sinon, $\mathcal{G}(\lambda) = -\lambda$. Donc, on a $\max_{\lambda\in\mathbb{R}}\mathcal{G}(\lambda) = a + c$.

On a

$$\begin{array}{rcll} \mathcal{H}(\boldsymbol{\mu}) & = & \min & (a+d\mu)x + (c - k\mu)y \\ & & \text{s.c.} & x = 1 \\ & & & y \in \{0,1\}. \end{array}$$

Donc $\mathcal{H}(\mu) = a + d\mu + \min(0, c - k\mu)$, et $\max_{\mu\in\mathbb{R}_+}\mathcal{H}(\mu) = a + \frac{cd}{k}$.

La relaxation donnée par $\mathcal{G}$ est donc meilleure que celle donnée par $\mathcal{H}$ (car $d \le k$).

5. Elle est strictement meilleure si et seulement si $d < k$ et $c \neq 0$ simultanément.

4. REPOSITIONNEMENT DANS UN SYSTÈME DE VÉLOS PARTAGÉS – 5 PTS

1. Sur un arc $(o, s^{(1)})$, on met comme capacité i_s, et sur un arc $(s^{(n_s)}, d)$, on met comme capacité f_s. Sur un arc $(s^{(\alpha)}, t^{(\beta)})$, on met une capacité q. Sur un arc $(s^{(k)}, s^{(k+1)})$, on met une capacité k_s.

Si la tournée est équilibrante, on obtient naturellement un flot dont la valeur est exactement $\sum_{s\in S} i_s$ de la manière suivante. En effet, sur un arc $(o, s^{(1)})$, on met un flot égal à i_s et sur un arc $(s^{(n_s)}, d)$, on met un flot égal à f_s. Sur un arc $(s^{(\alpha)}, t^{(\beta)})$, on met comme flot le nombre de vélos transportés par le camion lors de son passage de s à t. Sur un arc $(s^{(k)}, s^{(k+1)})$, on met le nombre de vélos présents sur la station s juste après le $k$ème passage. Ces quantités respectent les capacités et la loi de Kirchoff.

Réciproquement, si on a un flot de valeur $\sum_{s\in S} i_s$, alors il existe un s-t flot entier de même valeur (toutes les capacités étant entières - théorème du cours). Sur un arc $(o, s^{(1)})$, le flot s'interprète comme le nombre de vélos présents initialement en s, et sur un arc $(s^{(n_s)}, d)$, il s'interprète comme le nombre de vélos en s à la fin de la tournée. Sur un arc $(s^{(\alpha)}, t^{(\beta)})$, le flot s'interprète comme le nombre de vélos transportés par le camion lorsqu'il va de s à t. Enfin, sur un arc $(s^{(k)}, s^{(k+1)})$, le flot s'interprète comme le nombre de vélos présent sur la station s après le $k$ème passage. La loi de Kirchoff et le respect des capacités fait que cette solution est réalisable en pratique.

2. Le calcul du flot maximum se faisant de manière rapide à l'aide d'algorithmes ad hoc, on peut construire une recherche locale n'explorant que les séquences de stations, sans avoir à préciser les opérations réalisées par le camion. Cela réduit considérablement la taille de l'espace des solutions et la gestion de son exploration. On peut de plus s'autoriser des tournées non équilibrantes, l'algorithme de flot maximum permettant également de quantifier le "degré de non-réalisabilité" d'une tournée ($\sum_s i_s$ – valeur du flot).

5. TOURNÉE LE LONG D'UNE VOIE FERRÉE – 9 PTS

1. Il suffit de parcourir tous les points dans l'ordre de leurs indices. Le durée totale est $p_n - p_1$.

2. On montre d'abord que la machine n'a jamais un intérêt strict à revenir sur ses pas. En effet, supposons que la machine visite un point p_j, puis revient sur p_{j-1}, puis retourne sur p_j ultérieurement. La première visite de p_j pourrait être évitée, quitte à attendre en p_{j-1}. On peut donc à nouveau parcourir tous les points dans l'ordre de leurs indices. En revanche, si on arrive avant r_i en un point p_i, il faut attendre jusqu'à l'instant r_i.

3. En notant h_j l'instant au plus tôt auquel on peut traiter la position p_j en ayant traité tous les points p_i avec $i < j$, on a la relation $h_{j+1} = \max(r_j, h_j + p_{j+1} - p_j)$. La durée totale est donc donnée par h_n et une récurrence immédiate fournit :

$$h_n = \max_{j=1,\dots,n} (r_j + p_n - p_j).$$

4. $p_j - p_{j-1} + \pi(j-1, k)$ est l'instant au plus tôt ayant permis à la machine d'être en p_j, en ayant traité l'ensemble des points de $\{p_1, \dots, p_{j-1}\} \cup \{p_k, \dots, p_n\}$ (mais pas nécessairement le point p_j lui-même), et en arrivant en p_j depuis p_{j-1}. La quantité $\max(r_j, p_j - p_{j-1} + \pi(j-1, k)$ est donc l'instant au plus tôt ayant permis à la machine d'être en p_j, en ayant traité l'ensemble des points de $\{p_1, \dots, p_j\} \cup \{p_k, \dots, p_n\}$ (point p_j inclus) en arrivant depuis p_{j-1}. La quantité $\pi(j, k)$ est définie ainsi, sans la contrainte d'arriver de p_{j-1}. D'où l'inégalité.

Le raisonnement est en tout point semblable pour l'autre inégalité, à ceci près que cette fois on regarde la solution obtenue en se contraingnant à arriver par p_k.

5. Considérons la solution optimale permettant de réaliser $\pi(j,k)$. Elle arrive forcément en p_j depuis p_{j-1} ou de p_k, en ayant visité tous les points de $\{p_1,\dots,p_{j-1}\}\cup\{p_k,\dots,p_n\}$. Si elle vient de p_{j-1}, on obtient que $\pi(j,k)$ est plus grand que le terme de gauche dans le min ; si elle vient de p_k, on obtient le terme de droite. Combiné avec la question précédente, on obtient l'inégalité.

6. On calcule progressivement toutes les valeurs $\pi(j,k)$ (en initialisant avec les valeurs pour $j=0$) et $\lambda(j,k)$ (en initialisant avec les valeurs pour $k=n+1$) : quand on connaît les valeurs prises par $\pi(j-1,\cdot)$ et par $\lambda(\cdot,k+1)$ pour j et k donnés, on peut déterminer tous les $\pi(j,k)$ et tous les $\lambda(j,k)$, chacun d'eux calculé en temps constant ; on passe en revue $j=0$ et $k=n+1$, puis $j=1$ et $k=n$, puis $j=2$ et $k=n-1$,... $j=x$ et $k=n+1-x$, et ce pour x allant de 0 à n. En procédant ainsi, chaque $\pi(j,k)$ et chaque $\lambda(j,k)$ n'est calculé qu'une seule fois. La complexité totale est bien $O(n^2)$.

7. La méthode s'appuie sur le même principe que la programmation dynamique (à ceci près que l'on a deux équations de Bellman corrélées).

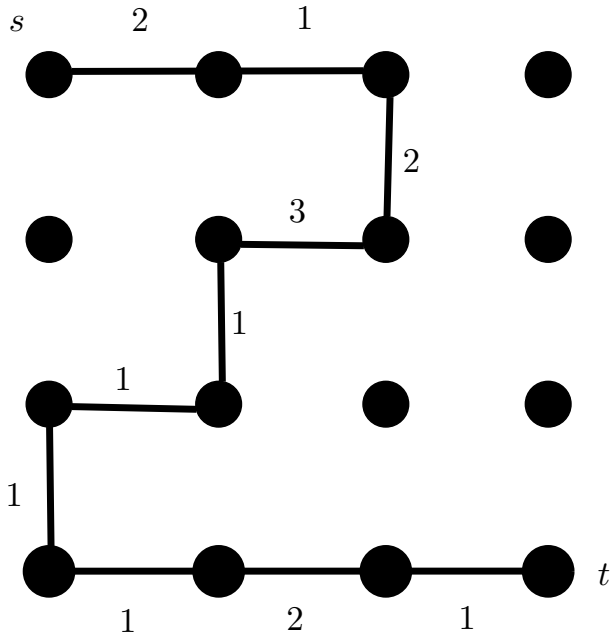

FIGURE 1. Le réseau de l'exercice 1