CONTRÔLE DE RECHERCHE OPÉRATIONNELLE

1. Creuser une rivière artificielle à moindre coût – 3 pts

C'est un problème de plus court chemin. Il peut être résolu par l'algorithme de Dijkstra, mais cela risque d'être long. Il vaut mieux utiliser l'équation de programmation dynamique, le graphe étant acircuitique. On pouvait aussi le résoudre en supprimant progressivement les arcs "dominés" par des chemins dans le graphe.

Si on note $\pi(x)$ le coût minimal d'un chemin de s à x, on a

$$\pi(x) = \min_{y \in N^-(x)} (\pi(y) + c(yx)).$$

En appliquant cette formule de proche en proche, on obtient en mettant en indice le sommet précédent qui minimise : $\pi(s) = 0$, $\pi(a) = 3_s$, $\pi(b) = 8_a$, $\pi(c) = 9_s$, $\pi(d) = 21_b$, $\pi(f) = 20_b$, $\pi(k) = 24_f$, $\pi(g) = 26_d$, $\pi(e) = 32_d$, $\pi(i) = 44_e$, $\pi(h) = 29_g$, $\pi(l) = 37_h$, $\pi(m) = 51_l$, $\pi(n) = 54_h$, $\pi(o) = 59_n$, $\pi(q) = 38_k$, $\pi(j) = 42_e$, $\pi(r) = 55_q$, $\pi(p) = 58_j$, $\pi(t) = 64_p$.

Le chemin optimal est $s - a - b - d - e - j - p - t$.

2. Identification des tâches critiques et calcul des marges d'un projet de construction d'un pavillon – 4 pts

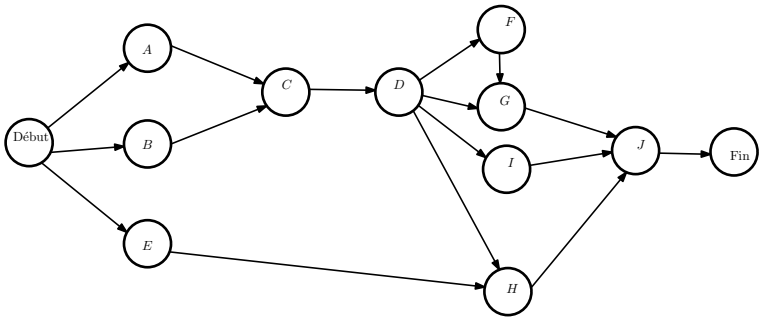
On utilise la méthode vue en cours (équation de programmation dynamique).

1. $\eta_A = 0$, $\eta_B = 0$, $\eta_C = \max(0+5, 0+4) = 5$, $\eta_D = 5+10 = 15$, $\eta_E = 0$, $\eta_F = 15+22 = 37$, $\eta_G = \max(15+22, 37+5) = 42$, $\eta_H = \max(15+22, 0+3) = 37$, $\eta_I = 15+3 = 18$, $\eta_J = \max(42+6, 37+4, 18+3) = 48$.

2. La durée minimal du projet est $\eta_J + 3 = 51$.

3. On calcule π'_X la longueur du plus long chemin de X à Fin. On aura alors $\pi_X = 51 - \pi'_X$ (cours).
   $\pi'_J = 3$, $\pi'_G =$
4. Les marges sont ($m_X = \pi_X - \eta_X$)
   $m_A = 0$, $m_B =$

Date: 22 février 2018.



3. Voyageur de commerce et programmation dynamique – 6 pts

1. Pour (X, v) tels que $v \in X \subseteq V \setminus \{s\}$, on note $\pi(X, v)$ le coût minimum d'une chaîne élémentaire dont les sommets sont $\{s\} \cup X$ et dont les extrémités sont s et v. On a alors

$$\pi(X, v) = \begin{cases} c(sv) & \text{si } |X| = 1 \\ \min_{u \in X \setminus \{v\}} (\pi(X \setminus \{v\}, u) + c(uv)) & \text{sinon.} \end{cases}$$

Pour trouver la meilleure chaîne hamiltonienne d'extrémité s, il suffit alors de comparer les valeurs de $\pi(V, v)$ pour tout $v \in V$ (les coûts étant positifs, on n'a pas intérêt à revenir en s).

2. C'est le nombre de couples (X, v) comme dans l'énoncé. Il y en a

$$\sum_{k=1}^{n-1} \binom{n-1}{k} k = (n-1)2^{n-2}.$$

3. Pour chaque état (X, v), le nombre d'additions est $|X| - 1$ (nombre de u possibles dans l'équation de programmation dynamique). D'où un nombre d'opérations

$$\sum_{k=1}^{n-1} \binom{n-1}{k} k(k-1) = (n-1)(n-2)2^{n-3}.$$

Enumérer toutes les solutions donne un nombre d'additions $(n-1)!(n-1) \sim n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$, ce qui est incomparablement plus grand.

4. Pour $n = 15$, on compte $745'472$ opérations, ce qui se fait en moins d'une seconde.

Pour $n = 30$, on compte 1.09×10^{11} opérations, ce qui se fait en 1.09×10^5 secondes, soit environ 30 heures. On ne peut résoudre cette instance en 1 jour, mais en une semaine oui.

Pour $n = 45$, on compte 8.32×10^{15} opérations, ce qui fait 8.32×10^9 secondes, soit environ 263 années. Même en un siècle on ne parvient à résoudre cette instance.

5. On sait calculer la plus courte chaîne hamiltonienne de s à v, pour tout $v \notin V$. Il suffit alors de comparer les valeurs de $\pi(V, v) + c(vs)$ pour tout v.

4. Ordonnancement de voitures sur une chaîne de montage – 10 pts

4.1. **Cas général – 6 pts.**

1. La fonction objectif est $\sum_{i=1}^{n-1} \rho u_i + \sum_{o \in \mathcal{O}} \sum_{i=1}^{n-q_o} \gamma_o v_{i,o}$.

2. Pour tout k, on doit avoir $\sum_{i=1}^{n} x_{i,k} = n_k$, et pour tout $i \in \{1, \ldots, n\}$, on doit avoir $\sum_{k \in \mathcal{K}} x_{i,k} = 1$.

3. On doit avoir $y_{i,c} = \sum_{k \in \mathcal{K}} \epsilon_{k,c} x_{i,k}$ pour tout $i \in \{1, \ldots, n\}$ et tout $c \in \mathcal{C}$. De même, on doit avoir $z_{i,o} = \sum_{k \in \mathcal{K}} \delta_{k,o} x_{i,k}$ pour tout $i \in \{1, \ldots, n\}$ et tout $o \in \mathcal{O}$.

4. Il y a deux solutions. La première consiste à écrire $\sum_{j=i}^{i+r-1} u_j \geq 1$ pour tout $i \in \{1, \ldots, n-r+1\}$. C'est celle-là que l'on retiendra. L'autre solution consiste à écrire $\sum_{j=i}^{i+r} y_{j,c} \leq r$ pour tout $i \in \{1, \ldots, n-r\}$ et tout $c \in \mathcal{C}$.

5. On a pour tout i et tout c les inégalités $u_i \geq y_{i,c} - y_{i+1,c}$ et $u_i \geq y_{i+1,c} - y_{i,c}$. Par ailleurs, pour tout i et o, on a $v_{i,o} \geq \sum_{j=i}^{i+q_o-1} z_{j,o} - p_o$.

6. Le programme linéaire complet est alors

$$\begin{array}{lrll}
\min & \sum_{i=1}^{n-1} \rho u_i + \sum_{o\in\mathcal{O}} \sum_{i=1}^{n-q_o} \gamma_o v_{i,o} & & \\
\text{s.c.} & \sum_{i=1}^{n} x_{i,k} = n_k & k \in \mathcal{K} \\
& \sum_{k\in\mathcal{K}} x_{i,k} = 1 & i \in \{1,\dots,n\} \\
& y_{i,c} = \sum_{k\in\mathcal{K}} \epsilon_{k,c} x_{i,k} & i \in \{1,\dots,n\}, c \in \mathcal{C} \\
& z_{i,o} = \sum_{k\in\mathcal{K}} \delta_{k,o} x_{i,k} & i \in \{1,\dots,n\}, k \in \mathcal{K} \\
& \sum_{j=i}^{i+r-1} u_j \geq 1 & i \in \{1,\dots,n-r+1\} \\
& u_i \geq y_{i,c} - y_{i+1,c} & i \in \{1,\dots,n-1\}, c \in \mathcal{C} \\
& u_i \geq y_{i+1,c} - y_{i,c} & i \in \{1,\dots,n-1\}, c \in \mathcal{C} \\
& v_{i,o} \geq \sum_{j=i}^{i+q_o-1} z_{j,o} - p_o & o \in \mathcal{O}, i \in \{1,\dots,n-q_o+1\} \\
& u_i, y_{i,c}, z_{i,o}, x_{i,k} \in \{0,1\} & \\
& v_{i,o} \in \mathbb{Z}_+. &
\end{array}$$

7. On vérifie aisément que $x_{i,k} = n_k/n$ pour tout i et k est solution réalisable de ce programme. Montrons que cette solution est également optimale. La condition sur les options assure que le terme "des options" de la fonction objectif est nul. L'autre terme de la fonction objectif est alors égal à $\lfloor n/r \rfloor \rho$. Or cette quantité est aussi une borne inférieure car c'est la valeur du programme suivant obtenu en supprimant des contraintes :

$$\begin{array}{lrl}
\min & \sum_{i=1}^{n-1} \rho u_i & \\
\text{s.c.} & \sum_{j=i}^{i+r-1} u_j \geq 1 & i \in \{1,\dots,n-r+1\} \\
& u_i \in \{0,1\}. &
\end{array}$$

La valeur optimale du relâché continu est donc bien $\lfloor n/r \rfloor \rho$. Cette borne est probablement de piètre qualité car elle ne tient absolument pas compte des contraintes sur les options.

4.2. **Cas particuliers – 4 pts.**

4.2.1. *Si $r = +\infty$ et $q_o = 2$ pour tout $o \in \mathcal{O}$.*

8. Chaque sommet correspond a une voiture distincte. Sur chaque arête ij, on met le coût qu'aurait l'enchaînement ij dans un ordonnancement des voitures. Comme il n'y a pas de contraintes sur les couleurs et $q_o = 2$, le coût total d'un ordonnancement ne dépend que des paires de voitures consécutives. Ce coût se calcule de la manière suivante :

$$c_{ij} = \rho \times 1_{\text{voitures } i \text{ et } j \text{ de couleurs différentes}} + \sum_{o\in\mathcal{O}} \gamma_o \times 1_{\text{voitures } i \text{ et } j \text{ ont toutes deux l'option } o \text{ et } p_o = 1}$$

Une chaîne ouverte de plus petit coût correspond à un ordonnancement réalisable de plus petit coût.

9. Dans ce cas particulier, tous les sommets sont incidents à une arête de coût 1, sauf le sommet 3 qui n'est incident qu'à des arêtes de coût 2. La quantité $5 \times 1 + 2 = 7$ est donc une borne inférieure sur le coût optimal. La suite réalisable $1, 4, 6, 2, 7, 5, 3$ est de coût 7 et est donc optimal.

4.2.2. *Si $\mathcal{O} = \emptyset$.*

10*. Ecrivons d'abord une condition nécessaire. m_{c^*} est le nombre minimum de blocs de voitures consécutives de couleur c^* que l'on pourra trouver dans la séquence. Ces blocs doivent être séparés les uns des autres d'au moins une voiture de couleur $c \neq c^*$. Une condition nécessaire est donc

$$m_{c^*} - 1 \leq \sum_{c\in\mathcal{C}\setminus\{c^*\}} N_c.$$

Montrons qu'elle est également suffisante. Pour cela on considère que l'on dispose pour chaque couleur c de m_c blocs de voitures consécutives de couleur c. Par définition de m_c, on peut choisir ces blocs de longueur $\leq r$ et bien avoir toutes les voitures. On va essayer de construire une solution en posant les blocs les uns derrière les autres. Deux cas sont à distinguer :

— $m_{c^*} - 1 \leq \sum_{c \in \mathcal{C} \setminus \{c^*\}} m_c$: on peut utiliser un bloc monocouleur de couleur $c \neq c^*$ pour séparer deux blocs consécutifs de couleur c^*. Les blocs en trop peuvent être placés entre deux blocs de couleurs différentes. On peut donc construire une solution avec m_c bloc de couleur c, pour tout $c \in \mathcal{C}$. Noter que le nombre de changements de couleur dans ce cas est $\sum_{c \in \mathcal{C}} m_c - 1$.

— $m_{c^*} - 1 > \sum_{c \in \mathcal{C} \setminus \{c^*\}} m_c$: dans ce cas, on a $m_c < m_{c^*}$ pour tout c. On 'casse' autant de blocs de couleurs $c \neq c^*$ qu'il faut afin d'arriver à un nombre de blocs $= m_{c^*} - 1$, puis on applique la méthode décrite au point précédent. Noter que le nombre de changements de couleur dans ce cas est $2m_{c^*} - 2$.

Enfin, montrons l'optimalité de ces solutions. Pour cela, il suffit de noter que $\sum_{c \in \mathcal{C}} m_c$ est le nombre minimum de blocs monocouleurs que l'on peut avoir. $\sum_{c \in \mathcal{C}} m_c - 1$ est donc une borne inférieure. D'autre part, entre deux blocs consécutifs de couleur c^*, on a forcément 2 changements de couleurs, donc $2m_{c^*} - 2$ est aussi une borne inférieure.