UE Androide/MAOA : Novembre 2016.

Durée 2h00. Notes de cours et de TD autorisées.

Exercice 1 (7 points)

Formulation PLNE pour emploi du temps

On s'intéresse à la conception automatique d'emploi du temps pour une formation par alternance où enseignent des intervenants venant d'entreprises et ayant des obligations professionnelles extérieures à cette formation. Il s'agit de concevoir une semaine de cours adaptée aux horaires des intervenants (l'emploi du temps est ainsi reproduit chaque semaine).

On connaît la liste des matières $\{1, \ldots, m\}$, la durée hebdomadaire d_l, $l \in \{1, \ldots, m\}$, de chaque matière (en heures). Un unique intervenant intervient par matière : il fournit la liste des créneaux horaires où il est disponible pour son intervention. On note L_l la liste des créneaux de la matière $l \in \{1, \ldots, m\}$ et la liste de tous les créneaux est donc $L = \bigcup_{l=1}^{m} L_l$ avec $n = |L|$. Chacun des n créneaux $i \in \{1, \ldots, n\}$ est donné par le quadruplet $c_i = (M_i, J_i, S_i, T_i)$ indiquant la matière M_i du créneau, le jour $J_i \in \{1, \ldots, 5\}$, l'horaire du début du créneau S_i et l'horaire de fin du créneau T_i. Le tableau suivant donne une illustration des propositions de créneaux avec $m = 5$ et $n = 17$.

Matières	Durée	Créneaux possibles
1	4h	Lu 13h-15h / Ma 10h-12h / Ma 16h-18h / Me 11h-13h
2	2h	Lu 10h-12h / Ma 10h-12h / Je 10h-12h
3	4h	Lu 8h-10h / Lu 17h30-19h30 / Ma 14h-16h / Ve 8h-10h
4	2h	Me 8h-11h / Me 13h30-17h30 / Je 8h-11h
5	3h	Lu 9-10h30 / Lu 10h30-12h / Ve 11h-12h30

La conception de l'emploi du temps doit prendre en compte des contraintes fonctionnelles (disponibilité des intervenants, horaire maximal d'enseignement,...) tout en tenant compte des préférences des enseignants et des étudiants : pour chacun des n créneaux $i \in \{1, \ldots, n\}$, on connaît également une note p_i entre 1 et 10 qui est une moyenne (pondérée...) des préférences de l'intervenant et des étudiants pour le créneau i.

La liste des contraintes fonctionnelles à respecter peut s'écrire ainsi :

a) pas de chevauchement des créneaux (il s'agit d'une seule formation)
b) pas plus de 7 heures de cours par jour
c) une demi-journée doit être libre de cours pour permettre des travaux en groupe
d) une heure doit être libre sur le créneau 12h-14h pour le repas

Ce problème de conception d'emploi du temps consiste donc à fournir un emploi du temps vérifiant les contraintes et maximisant la satisfaction des participants à cette formation.

Pour résoudre ce problème, on désire le modéliser par un programme linéaire en nombres entiers. Pour cela, on considère les variables binaires x_i indiquant si un créneau est sélectionné ou non.

Attention : il s'agit de donner une formulation pour toute instance et non pour l'exemple ci-dessus. Vous devez utiliser les notations génériques de cet énoncé. Vous pouvez éventuellement utiliser les outils suivants :

- pour deux créneaux c_i et c_j sur la même journée, on sait ainsi tester en $\theta(1)$ si $c_i < c_j$, c'est-à-dire si le créneau c_i se termine avant le créneau c_j.
- pour deux créneaux c_i et c_j, la fonction $intersec(c_i, c_j)$ retourne vrai si les créneaux s'intersectent.

Question 1 (1/7) — Modéliserla contrainte a).
Question 2 (1/7) — Modéliser la contrainte b).
Question 3 (1/7) — Notons A_k, $k \in \{1, \ldots, 10\}$, le créneau correspondant à chacune des 10 demi-journées de la semaine. Modéliser la contrainte c).
Question 4 (2/7) — On suppose que chaque créneau laisse au moins 1 heure de libre sur le temps 12h-14h. Modéliser la contrainte d).
Question 5 (2/7) — On désirer maximiser, parmi toutes les matières, la préférence du créneau le moins préféré de cette matière. Modéliser ce critère.

Exercice 2 (6 points)

Problème du sous-graphe biparti induit

Soit un graphe $G = (V, E)$ où V est l'ensemble des sommets et E l'ensemble des arêtes. Un graphe est dit *biparti* lorsque son ensemble de sommets peut être partitionné en deux ensembles de sommets non vides V_1 et V_2 tels qu'aucun couple de sommets de V_1 (respectivement de V_2) ne soit adjacent.
Soit $W \subset V$ un sous-ensemble de sommets, on note $E(W)$ l'ensemble des arêtes ayant leurs deux extrémités dans W. On dit alors que le graphe $(W, E(W))$ est le graphe induit par W. Etant un poids $c(v)$ $v \in V$ associé aux sommets, le *problème du sous-graphe biparti induit* (PSBI) consiste à déterminer un sous-graphe biparti induit $(W, E(W))$ de G tel que $\sum_{v \in W} c(v)$ soit maximum.

Question 1 (2/6) —*Propriété sur les graphes bipartis*

Une *chaîne* P de G est une séquence d'arêtes $e_1, e_2, ..., e_k$ telles que $e_1 = v_1v_2, e_2 = v_2v_3, ..., e_k = v_kv_{k+1}$. Le nombre k d'arêtes de P est appelé la *longueur* de P. Une chaîne est dit *impaire* (resp. paire) si la longueur est impaire (resp. paire). Si $v_0 = v_{k+1}$, alors la chaîne P est dite un *cycle* de longueur k. On peut remarquer que l'ensemble vide est une chaîne de longueur nulle (0 est un nombre pair).
a) Montrez que si un graphe contient un cycle impair, alors il n'est pas biparti.
b) Soit $G = (V, E)$ un graphe sans cycle impair. On choisit arbitrairement un un sommet quelconque u dans V . On construit alors la partition (V_{pai}, V_{imp}) de l'ensemble V des sommets de G de la façon suivante

- V_{pai} contient les sommets v de V tels que le plus court chemin de u à v en nombre d'arêtes est pair,
- V_{imp} contient les sommets v de V tels que le plus court chemin de u à v en nombre d'arêtes est impair. On peut remarquer que $u \in V_{pai}$ et que (V_{pai}, V_{imp}) partitionne bien V en deux sous-ensembles distincts.

Montrez qu'il n'y a pas d'arête entre deux sommets de V_{pai}.
Montrez qu'il n'y a pas d'arête entre deux sommets de V_{imp}.
c) En déduire qu'un graphe est biparti si et seulement s'il ne contient pas de cycle impair.

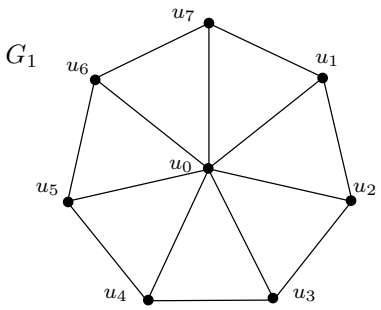
Question 2 (1/6) — *Formulation du problème*
Pour un sous-ensemble $F \subset E$ d'arêtes, on note $V(F)$ les sommets extrémités d'une arête de F.
a) Prouver que le problème du sous-graphe biparti induit est équivalent au programme en nombres entiers (P) suivant

$$(P)\begin{cases} Max \quad \sum_{u\in V} c(u)x(u) & \\ \sum_{u\in V(C)} x(u) \leq |C|-1, & \forall C \text{ cycle impair}, \quad (1) \\ 0 \leq x(u) \leq 1, & \forall u \in V \quad (2) \\ x(u) \text{ entier}, & \forall u \in V. \quad (3) \end{cases}$$

Les inégalités (1) sont appelées inégalités de cycle impair.
b) Quels sont les caractéristiques de ce programme en terme de nombres de contraintes et de variables ? Quelles sont les possibilités pour résoudre un tel programme ?

Question 3 (3/6) —*Point extrême sur une roue*

Considérons le graphe $G_1 = (V_1, E_1)$ de la figure suivante (appelée une roue). On note u_0 son sommet "central" qui est relié à tous les autres sommets. On note $u_1, ..., u_7$ les autres sommets de G_1 qui forment un cycle dit "cycle extérieur".
Considérons la solution y^* donnée par $y^*(u_0) = \frac{2}{7}$ et $y^*(u_i) = \frac{6}{7}$, $i = 1, ..., 7$.



Pour un cycle C donnée, on appelle *corde* une arête reliant 2 sommets non-consécutifs du cycle. On admet que toutes les inégalités de cycles impairs, correspondant à des cycles avec cordes, sont dominées par les inégalités de cycles sans corde (c'est-à-dire que les premières peuvent être obtenues par somme simple des secondes). Ainsin si un point vérifie les inégalités de cycle sans corde, alors elle vérifie les inégalités de cycle en général.
a) Donnez tous les cycles impairs sans corde du graphe G_1.
b) Montrez que y^* vérifie toutes les contraintes du programme (P) quand on relaxe la contrainte d'intégrité (3). Donnez n inégalités vérifiées par y^* à l'égalité. Indiquez, sans le prouver pour y^* une condition pour que y^* soit un point extrême de cette formulation pour la roue G_1.
c) Considérons l'inégalité suivante

$$\sum_{i=1}^{7} x(u_i) + 3x(u_0) \leq 6. \quad (4)$$

On veut prouver que cette contrainte est valide. Pour cela, prenons un ensemble de sommets $B \subset V_1$ induisant un graphe biparti dans G_1. En considérant les deux cas suivants : $u_0 \in B$ et $u_0 \notin B$, prouvez alors que le vecteur d'incidence de B vérifie l'inégalité (4).

d) Pourquoi est-il utile d'ajouter cette contrainte au programme (P) correspondant au graphe G_1 dans le cadre d'un algorithme de coupes ?

Exercice 3 (8 points)

Ordonnancement et apprentissage

On considère un problème d'ordonnancement à une machine qui doit exécuter un ensemble de n tâches. On s'intéresse au cas particulier où la durée opératoire d'une tâche peut diminuer si on suppose que des tâches ont déjà été exécutées par la machine. Cette situation illustre un effet d'apprentissage pour lequel plus la tâche apparaît tard dans la séquence plus vite elle sera exécutée.

On suppose dans la suite de l'exercice qu'une tâche J_i possède une durée *nominale* p_i. Si la tâche J_i est la première à être exécutée, sa durée sera égale à p_i, si elle est exécutée à une position r sa durée sera égale à $p_{ir} = p_i \times r^a$ avec $a \leq 0$. Le paramètre a est un indicateur d'apprentissage.
Pour illustrer cette notion, si on suppose que l'indicateur d'apprentissage $a = -\frac{1}{2}$, une tâche J_i de durée nominale 100 qui serait exécutée en deuxième position dans un ordonnancement aurait une durée $p_{i2} = 100 \times 2^{-\frac{1}{2}} = 70,71$.

Question 1 (1/8) — Définir les valeurs p_{ir} pour $i = 1, \cdots, n$ et $r = 1, \cdots, n$ en fonction des durées nominales p_i et du paramètre a.

Question 2 (2,5/8) — On considère le problème $1|p_{ir} = p_i r^a|C_{\max}$.

a On suppose dans cette question que la contrainte d'apprentissage est relâchée. On considère donc le problème $1|-|C_{\max}$. Comment résoudre optimalement ce problème ? Justifiez votre réponse.

b Est-ce que le même principe peut-être utilisé pour résoudre le problème $1|p_{ir} = p_i r^a|C_{\max}$? Justifiez votre réponse.

c Montrer que l'ordonnancement obtenu en séquençant les tâches dans l'ordre des durées opératoires nominales croissantes est optimal pour $1|p_{ir} = p_i r^a|C_{\max}$. La preuve pourra être établie par argument d'échange.

Question 3 (2,5/8) — On considère le problème $1|p_{ir} = p_i r^a|\sum_i C_i$.

a On suppose dans cette question que la contrainte d'apprentissage est relâchée. On considère donc le problème $1|-|\sum_i C_i$. Rappelez la propriété vue en cours pour résoudre ce problème ?

b Montrer que cette propriété permet de résoudre le problème $1|p_{ir} = p_i r^a|\sum_i C_i$. La preuve pourra être établie par argument d'échange.

Question 4 (2/8) — On considère le problème $1|p_{ir} = p_i r^a|L_{\max}$.

a On suppose dans cette question que la contrainte d'apprentissage est relâchée. On considère donc le problème $1|-|L_{\max}$. Rappelez la propriété vue en cours pour résoudre ce problème ?

b Cette propriété permet-elle de résoudre le problème $1|p_{ir} = p_i r^a|L_{\max}$. Si oui, le prouver, sinon fournir un contre-exemple.