# GIPAD – GS2
# Recherche Opérationnelle - APP2
# Correction devoir final

30 novembre 2010

---

**durée :** 3 heures

**documents autorisés :** tous documents de cours hors livres et photocopies de livre.

**barème :** note sur 24 = nombre de points * 20 / 30

1. Modélisation et décomposition (12 points)
2. Lot-Sizing Problems :
   (a) Modélisation linéaire (7 points)
   (b) Programmation dynamique (9 points)
   (c) Relaxation lagrangienne (8 points)

---

## Notations et définitions

| | |
|---|---|
| $\mathbb{Z}$, $\mathbb{Z}_+$, $\mathbb{Z}_{+*}$ | les ensembles d'entiers, d'entiers positifs, d'entiers positifs non-nuls |
| $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{R}_{+*}$ | les ensembles de réels, de réels positifs, de réels positifs non-nuls |

**Définition 1** *Une instance de* ***set-packing*** *est définie par la donnée (1) d'un ensemble de base* $A$*, (2) d'une collection* $B$ *de sous-ensembles de* $A$*, (3) d'un coût* $c_b \geqslant 0$ *associé à chaque élément* $b \in B$*. Le problème consiste à sélectionner des éléments de* $B$ *deux-à-deux disjoints et dont la somme des coûts est maximale.*

# 1 Modélisation et décomposition

**Problème 1** Graph Clustering.
*Soit un graphe complet non-orienté* $G = (V, E)$*, une constante entière* $K \in \mathbb{Z}_{+*}$*, un coût* $c_e > 0$ *associé à chaque arête* $e \in E$*, un poids* $d_i \geqslant 0$ *associé à chaque sommet* $i \in V$*, et une capacité* $C \in \mathbb{Z}_{+*}$ *telle que* $\min_{i \in V} d_i \leqslant C < \sum_{i \in V} d_i$.
*Un* ***cluster*** *de* $G$ *est un sous-ensemble (éventuellement vide) de sommets de* $G$ *dont la somme des poids n'excède pas la capacité* $C$*. Les arêtes du sous-graphe engendré par un cluster sont les arêtes de* $G$ *dont les deux extrémités appartiennent au cluster.*
*Le problème consiste à séparer l'ensemble* $V$ *des sommets en* $K$ *clusters de sorte que chaque sommet appartient à au plus un cluster et la somme des coûts des arêtes des sous-graphes engendrés par chaque cluster est maximale.*

**Question 1** ***(12 points).***

**Q1.1. (1)** Montrer que ce problème possède toujours une solution optimale.

**Q1.2. (2)** Soient les variables binaires $x_i^k = 1$ si un sommet $i$ appartient à un cluster $k$ et $y_e^k = 1$ si une arête $e$ appartient au sous-graphe engendré par un cluster $k$, modéliser au moyen de contraintes linéaires sur ces variables la condition logique suivante :

$$y_e^k = 1 \quad \Longleftrightarrow \quad x_i^k = 1 \text{ et } x_j^k = 1, \qquad \text{avec } e = (i, j).$$

**Q1.3. (2)** Modéliser le problème de graph-clustering en un programme linéaire en variables binaires (PLVB) et identifier dans cette formulation les contraintes et variables redondantes (si elles existent).

**Q1.4. (3)** Reformuler le problème de graph-clustering en une variante du set-packing, en définissant précisément les données d'une telle instance de set-packing, et une manière de construire ces données à partir d'une instance de graph clustering.

**Q1.5. (2)** En déduire un second modèle de PLVB pour le problème de graph clustering.

**Q1.6. (2)** Proposer une méthode de construction heuristique d'une solution réalisable de bonne qualité pour le problème de graph clustering, basée sur la résolution de la relaxation continue de ce PLVB.

**Correction Question 1.**

1. L'ensemble des solutions réalisables du problème est non-vide (la solution triviale avec K clusters vides), discret et fini (car il existe un nombre fini de clusters de G) : il possède donc un élément optimal pour toute fonction objectif.
2. $y_e^k \leqslant x_i^k, y_e^k \leqslant x_j^k, y_e^k \geqslant x_i^k + x_j^k - 1$
3. Soit $y_e^k = 1$ si $e$ est une arête du k-ème cluster et $x_i^k = 1$ si $i$ appartient au k-ème cluster :

$$
\begin{aligned}
(GC) : z \;=\; \max & \sum_{k=1}^{K} \sum_{e \in E} c_e y_e^k & \\
\text{s.t.} & \sum_{k=1}^{K} x_i^k \leqslant 1 & \forall i \in V, \\
& \sum_{i \in V} d_i x_i^k \leqslant C & \forall k = 1, \ldots, K, \\
& y_e^k \leqslant x_i^k & \forall e \in E, i \in e, k = 1, \ldots, K, \\
& y_e^k \geqslant x_i^k + x_j^k - 1 & \forall e = (i,j) \in E, k = 1, \ldots, K, \\
& x_i^k \in \{0,1\} & \forall i \in V, k = 1, \ldots, K, \\
& y_e^k \in \{0,1\} & \forall e \in E, k = 1, \ldots, K.
\end{aligned}
$$

   Les contraintes $y_e^k \geqslant x_i^k + x_j^k - 1$ sont redondantes, car elles sont satisfaites par toute solution optimale $(x^*, y^*)$ du programme $(GC')$ obtenu en relaxant ces contraintes. En effet, dans le cas contraire, il existe k et $e$ tels que $y_e^{k*} = 0$ et $x_i^{k*} = x_j^{k*} = 1$, hors remplacer $y_e^{k*}$ par 1 dans la solution $(x^*, y^*)$ mène à une solution réalisable et de coût strictement meilleur ($+c_e > 0$).
4. On considère l'ensemble des sommets de G comme ensemble de base et la collection des clusters non-vides de G à chacun desquels est associé un coût défini comme la somme des coûts des arêtes du sous-graphe engendré par le cluster : $A = V$, $B = \{b \subset V \mid b \neq \emptyset, \sum_{i \in b} d_i \leqslant C\}$ et $c_b = \sum_{e \subseteq b} c_e$ pour tout $b \in B$. Le problème de graph-clustering consiste à résoudre cette instance de set-packing avec la contrainte supplémentaire : on ne peut sélectionner que K éléments au plus de B. L'ensemble des clusters de G coincide avec l'ensemble des solutions réalisables d'un problème de Knapsack 0-1 de capacité C avec un item par sommet $i \in V$ de poids $d_i$. On peut construire cette collection au moyen d'un arbre de recherche binaire (ajoute ou non le sommet $i$ au cluster père), chaque noeud correspondant à un cluster, en coupant un noeud dès que le poids de celui-ci excède la capacité. On accélère la recherche en classant initialement les sommets du graphe par ordre de poids croissant (si le sommet $i$ ne peut être ajouté au cluster, alors les sommets $i+1, i+2, \ldots$ non plus). On peut facilement calculer le coût d'un cluster, quand celui-ci est construit, en parcourant les arêtes du graphe engendré. De plus, ce calcul peut se faire de manière incrémental dans l'arbre de recherche ci-dessus.
5. Pour tous $i \in V$ et $b \in B$, on note $\delta_{ib}$ l'indicateur booléen d'appartenance de $i$ à $b$ : $\delta_{ib} = 1$ si $i \in s$, et $\delta_{ib} = 0$ sinon.

$$\max\{\sum_{b \in B} c_b x_b \mid \sum_{b \in B} x_b \leqslant K,\ \sum_{b \in B} \delta_{ib} x_b \leqslant 1 (\forall i \in V),\ x_b \in \{0,1\} (\forall b \in B)\}.$$

6. Étant donnée une solution fractionnaire $\bar{x}$ de ce programme, on sélectionne l'ensemble des éléments $b \in B$ tels que $x_b > 0$. Il y en a éventuellement plus que K et ne sont pas nécessairement disjoints. Exemple pour se ramener à une solution réalisable : on considère chaque sommet appartiennant à plusieurs élements sélectionnées et on le supprime de tous les éléments sauf de l'élément b qui maximise $\sum_{j \in b} c_{ij}$. À chaque fois, si le coût d'un élément devient nul, on le supprime de la sélection. Enfin, on conserve les K éléments de coût maximal.

# 2 Étude : Lot-Sizing Problems

## 2.1 Modélisation linéaire.

**Problème 2** Uncapacited Lot-Sizing Problem (ULS).
*Le problème de production par lot consiste à planifier la fabrication d'un type de produit dans une usine sur une période de $n$ jours consécutifs, de manière à satisfaire la demande journalière $d_t$ des clients, à chaque jour $t$, tout en minimisant les coûts de production. Sachant que :*

1. *les coûts unitaires de production $p_t$ sont variables, fonctions du jour $t$,*
2. *le coût fixe de démarrage des machines $f_t$ n'est pas comptabilisé si aucun produit n'est fabriqué un jour $t$ donné,*
3. *les produits fabriqués non vendus le jour même peuvent être stockés pour être vendus un jour futur, sans limite de temps ni d'espace, mais avec un coût unitaire de stockage $h_t$, lui aussi variable et fonction du jour $t$,*

*alors il peut être avantageux de produire plus que la demande, certains jours (où les coûts de fabrication sont moins élevés que les coûts de stockage), de façon à produire moins, ou pas du tout, les jours suivants (où les coûts de fabrication sont plus élevés).*
*Le problème consiste donc à déterminer le nombre $x_t$ de produits fabriqués à chaque jour $t$ dans un plan de production de moindre coût, compte-tenu des coûts donnés suivants :*
– $f_t \in \mathbb{Z}_{+*}$ *le coût fixe de fabrication le jour* $t$
– $p_t \in \mathbb{Z}_{+*}$ *le coût unitaire (par unité de produit) de fabrication le jour* $t$
– $h_t \in \mathbb{Z}_{+*}$ *le coût unitaire (par unité de produit) de stockage le jour* $t$
*Le problème ULS se modélise par le Programme Linéaire Mixte suivant :*

$$(P): \min \sum_{t=1}^{n} f_t y_t + \sum_{t=1}^{n} p_t x_t + \sum_{t=1}^{n} h_t s_t \tag{1}$$

$$\text{s.t. } s_{t-1} + x_t = d_t + s_t \qquad t = 1, \dots, n \tag{2}$$

$$x_t \leqslant M_t y_t \qquad t = 1, \dots, n \tag{3}$$

$$y_t \in \{0, 1\} \qquad t = 1, \dots, n \tag{4}$$

$$s_t, x_t \geqslant 0 \qquad t = 1, \dots, n \tag{5}$$

$$s_0 = 0 \tag{6}$$

*si $M_t$ sont des valeurs constantes « suffisamment » grandes.*

**Question 2** ***Modèles linéaires et inégalités valides (7 points).***

**Q2.1. (1)** Définir les variables et expliquer les contraintes du modèle $(P)$.

**Q2.2. (1)** Déterminer une valeur de $M_t$ la plus serrée possible, pour chaque jour $t = 1, \dots, n$.

**Q2.3. (1)** Déterminer la valeur de $y_1$ et de $s_n$ dans toute solution optimale.

**Q2.4. (2)** Quelle est la valeur minimale de $s_{t-1}$ dans toute solution réalisable dans laquelle $y_t = 0$ ? En déduire une inégalité valide pour $(P)$ de la forme $s_{t-1} \geqslant \alpha y_t + \beta$.

**Q2.5. (2)** Exprimer, en le prouvant, chaque variable $s_t$ par une fonction linéaire en les variables $x$. En déduire un second modèle linéaire $(P')$ de ULS basé uniquement sur les variables $x$ et $y$.

**Correction Question 2.**

1. $x_t$ est le nombre de produits fabriqués le jour t, $s_t$ le nombre de produits fabriqués le jour t ou avant mais non vendus ni au jour t ni avant, $y_t = 1$ ssi l'usine fabrique le jour t. Les contraintes (9) spécifient la condition logique $y_t = 0 \Rightarrow x_t = 0$ (si l'usine ne fonctionne pas alors aucun produit n'est fabriqué) à chaque jour t. Les contraintes (2) spécifient qu'à chaque jour t, la quantité de produits disponible au jour t pour satisfaire la demande $d_t$ est égale au stock du jour précédent $s_{t-1}$ plus la quantité $x_t$ fabriquée au jour t, et la quantité restante $s_t$ est stockée pour les jours suivants.
2. $M_t = \sum_{k=t}^{n} d_t$ : au maximum, on produit un jour t toute la demande des jours suivants (produire plus est possible mais coûtera nécessairement plus cher).
3. $y_1 \geqslant x_1/M_1 = (s_1 + d_1)/M_1 \geqslant d_1/M_1 > 0$ et $y_1 \in \{0,1\}$ donc $y_1 = 1$. (Plus précisément, comme la demande au temps 1 doit être satisfaite et qu'aucun stock n'est disponible $s_0 = 0$, alors on a $x_1 \geqslant d_1$.) Le stock $s_n$ à l'issue de la période de planification est naturellement vide dans toute solution optimale.
4. si $y_t = 0$ alors $x_t = 0$ donc $s_{t-1} = d_t + s_t \geqslant d_t$. La condition $y_t = 0 \Rightarrow s_{t-1} \geqslant d_t$ est donc satisfaite par toute solution réalisable. Elle se linéarise par $s_{t-1} \geqslant d_t(1 - y_t)$ qui est donc une inégalité valide.
5. $s_t = \sum_{k=1}^{t}(x_k - d_k)$. Preuve par récurrence : $s_1 = x_1 - d_1 + s_0 = x_1 - d_1$ OK pour $t = 1$. On suppose que c'est vrai pour $1 \geqslant t < n$ alors $s_{t+1} = x_{t+1} - d_{t+1} + s_t = x_{t+1} - d_{t+1} + \sum_{k=1}^{t}(x_k - d_k) = \sum_{k=1}^{t+1}(x_k - d_k)$, donc OK pour $t + 1$. cqfd.

$$(P') : \min \sum_{t=1}^{n} f_t y_t + \sum_{t=1}^{n} p_t x_t + \sum_{t=1}^{n} h_t \sum_{k=1}^{t}(x_k - d_k) \tag{7}$$

$$\text{s.t.} \sum_{k=1}^{t} x_k \geqslant \sum_{k=1}^{t} d_k \qquad t = 2, \ldots, n \tag{8}$$

$$x_t \leqslant M_t y_t \qquad t = 1, \ldots, n \tag{9}$$

$$y_t \in \{0,1\} \qquad t = 1, \ldots, n \tag{10}$$

$$x_t \geqslant 0 \qquad t = 1, \ldots, n \tag{11}$$

## 2.2 Programmation dynamique

*On utilisera, sans la prouver la proposition suivante :*

**Proposition 1** *ULS possède une solution optimale* $(x, y, s)$ *vérifiant la condition suivante :*

$$s_{t-1}.x_t = 0, \qquad \forall t = 1, \ldots, n \tag{12}$$

**Question 3** ***Programmation dynamique (9 points).***

**Q3.1. (1)** Exprimer la condition (12) en langage naturel

**Q3.2. (2)** Prouver que la solution optimale de la proposition vérifie la condition suivante :

$$x_t > 0 \quad \Rightarrow \quad \exists t' \geqslant t,\ x_t = \sum_{k=t}^{t'} d_k, \qquad \forall t = 1, \ldots, n \tag{13}$$

Autrement dit, soit la production du jour $t$ est nulle, soit elle coincide exactement avec la demande de ce jour et d'un certain nombre de jours consécutifs suivant.

*Soit* $(x, y, s)$ *une solution optimale de ULS vérifiant la condition* (12)*. On note, pour tout* $t = 0, 1, \ldots, n$, $G(t)$ *le coût de la solution pendant les* $t$ *premiers jours, autrement dit :*

$$G(t) = \sum_{k=1}^{t} f_k y_k + \sum_{k=1}^{t} a_k x_k + b,$$

*où* $a_t = p_t + \sum_{k=t}^{n} h_k$ *et* $b = -\sum_{t=1}^{n} h_t \sum_{k=1}^{t} d_k$ *sont les constantes obtenues par substitution des variables* $s$*. On utilisera, sans la prouver, la proposition suivante*

**Proposition 2** $G(n) = H(n) + b$ *avec*

$$H(t') = \begin{cases} \min_{1 \leqslant t \leqslant t'} (H(t-1) + f_t + a_t \sum_{k=t}^{t'} d_k) & \text{si } 1 \leqslant t' \leqslant n \\ 0 & \text{si } t' = 0 \end{cases}$$

**Q3.3. (1)** *Écrire un algorithme de programmation dynamique calculant la valeur optimale du problème ULS.*

**Q3.4. (2)** *Modifier et compléter cet algorithme afin qu'il calcule également une solution optimale.*

**Q3.5. (2)** *Résoudre l'instance de ULS suivante :* $n = 4$, $d = (2, 4, 5, 1)$, $p = (3, 3, 3, 3)$, $h = (1, 2, 1, 1)$ *et* $f = (12, 20, 16, 8)$.

**Q3.6. (1)** *Donner la classe de complexité de ULS.*

**Correction Question 3.**

1. Il y a fabrication uniquement les jours où le stock est vide.
2. La condition est vrai si $t = n$ avec $t' = n$. Soit un jour $t < n$ tel que $x_t > 0$, on choisit la période précédant la prochaine période de production i.e. la période $t' > t$ minimale telle que $t' = n$ ou bien $x_{t'+1} > 0$. On a alors $s_k = 0$ pour $k = t$ et $k = t'$ et $x_k = 0$ pour tout $t < k \leqslant t'$. En sommant la contrainte (2), on obtient
$$\sum_{k=t}^{t'} d_k = \sum_{k=t}^{t'} (s_{k-1} - s_k + x_k) = s_{t-1} - s_{t'} + \sum_{k=t}^{t'} x_k = x_t.$$
3. FOR $t = 1..n$ DO $H[t] = MAX\_INT$ ; END FOR
   $H[0] = 0$ ;
   FOR $t = 1..n$ DO
   $d = d_t$ ;
   FOR $k = t..0$ DO
   IF $H[t] > H[k-1] + f_k + a_k d$ THEN
   $H[t] = H[k-1] + f_k + a_k d$ ;
   $L[t] = k$ ;
   END IF
   $d += d_k$ ;
   END FOR
   END FOR
   L'algorithme tourne en $O(n^2)$.
4. $L[t]$ est l'indice du précédent jour où l'usine fabrique. Il est calculé en même temps que $H$. On peut ensuite calculer $X[t]$ en $O(n)$ par l'algorithme suivant :
   FOR $t = 1..n$ DO $X[t] = 0$ ; END FOR
   $t = n$ ;
   WHILE $t \geqslant 1$ DO
   $d = d_t$ ;
   FOR $k = t-1..L[t]$ DO $d += d_k$ ; END FOR
   $X[L[t]] = d$ ;
   $t = L[t] - 1$ ;
   END WHILE
5. On commence par calculer $a_t = p_t + \sum_{k=t}^{n} h_k$ : $a = (8, 7, 5, 4)$. $H(1) = f_1 + a_1 d_1 = 28$ et $L[1] = 1$. $H(2) = \min(28 + f_2 + a_2 d_2, 28 + a_1 d_2) = \min(76, 60) = 60$ et $L[2] = 1$. $H(3) = \min(60 + f_3 + a_3 d_3, 76 + a_2 d_3, 60 + a_1 d_3) = \min(101, 111, 100) = 100$ et $L[3] = 1$. $H(4) = \min(100 + f_4 + a_4 d_4, 101 + a_3 d_4, 111 + a_2 d_4, 100 + a_1 d_4) = \min(112, 106, 118, 108) = 106$ et $L[4] = 3$. Le coût de la solution est $G(4) = H(4) + b = 106 - 37 = 69$. Pour calculer la solution, $L[4] = 3$ et $L[3] = 1$ donc les jours de production sont $t = 1$ et $t = 3$, avec : $X[4] = 0$, $X[3] = d_3 + d_4 = 6$, $[2] = 0$ et $X[1] = d_1 + d_2 = 6$.
6. il existe un algorithme de résolution polynomial, donc ULS appartient à la classe $\mathcal{P}$.

## 2.3 Relaxation Lagrangienne

**Problème 3** Multi-Item Lot-Sizing Problem (MLS).
*Dans cette variante usuelle dans l'industrie, le problème de production s'applique à plusieurs types de produits à la fois, avec une capacité globale de production limitée. On suppose donc, que l'usine produit* $m$ *types de produits différents. Pour chaque type de produit* $i = 1, \dots, m$ *et chaque jour de planification* $t = 1, \dots, n$*, on dispose des données suivantes :*
– $d_t^i \in \mathbb{Z}_{+*}$ *la quantité de produits de type* $i$ *vendus le jour* $t$
– $f_t^i \in \mathbb{Z}_{+*}$ *le coût fixe de fabrication des produits de type* $i$ *le jour* $t$
– $p_t^i \in \mathbb{Z}_{+*}$ *le coût unitaire, par unité de produit de type* $i$*, fabriqué le jour* $t$
– $h_t^i \in \mathbb{Z}_{+*}$ *le coût unitaire, par unité de produit de type* $i$*, stocké entre le jour* $t$ *et le jour* $t+1$
*Enfin, il y a au plus* $C_t$ *types de produits distincts fabriqués chaque jour* $t = 1, \dots, n$*.*
*Le problème consiste à déterminer un plan de production de l'usine sur la période de* $n$ *jours le moins coûteux.*

**Question 4** ***Relaxation Larangienne (8 points).***

**Q4.1. (2)** Modéliser par un programme linéaire en variables mixtes.

**Q4.2. (3)** Appliquer une relaxation lagrangienne à ce programme (exprimer le dual lagrangien et décomposer le modèle du sous-problème).

**Q4.3. (3)** Proposer et détailler une méthode de résolution pour ce problème (réponse libre).

**Correction Question 4.**

$$
\begin{aligned}
(M): \ \min \ & \sum_{i=1}^{m}\sum_{t=1}^{n}(f_t^i y_t^i + p_t^i x_t^i + h_t^i s_t^i) & & \\
\text{s.t. } & s_{t-1}^i + x_t^i = d_t^i + s_t^i & & t = 1,\dots,n;\ i = 1,\dots,m \\
& x_t^i \leqslant M_t^i y_t^i & & t = 1,\dots,n;\ i = 1,\dots,m \\
& \sum_{i=1}^{m} y_t^i \leqslant C_t & & t = 1,\dots,n \\
& y_t^i \in \{0,1\}, s_t^i \geqslant 0, x_t^i \geqslant 0 & & t = 1,\dots,n;\ i = 1,\dots,m \qquad s_0 = 0
\end{aligned}
$$

Les contraintes de knapsack sont des contraintes à la fois complicantes et couplantes car les supprimer permet de décomposer le problème en $m$ problèmes ULS donc faciles, résolvables en $O(n^2)$. La dualisation de ces contraintes consiste à résoudre le dual lagrangian $(D) : \max_{\lambda \in \mathbb{R}_+^n} z_\lambda$ où $z_\lambda$ est la valeur optimale du sous-problème lagrangien $(L_\lambda)$ associé aux multiplicateurs $\lambda \in \mathbb{R}_+^n$. Celui-ci se décompose en $m$ sous-problèmes ULS distincts $(U_\lambda^i)$ pour tout $i = 1, \dots, m$ de la manière suivante :

$$
z_\lambda = \sum_{i=1}^{m} z_\lambda^i - \sum_{t=1}^{n} \lambda_t C_t, \qquad \text{avec}
$$

$$
\begin{aligned}
(U_\lambda^i): \ z_\lambda^i = \min \ & \sum_{t=1}^{n}((f_t^i + \lambda_t) y_t^i + p_t^i x_t^i + h_t^i s_t^i) & & \\
\text{s.t. } & s_{t-1}^i + x_t^i = d_t^i + s_t^i & & t = 1,\dots,n; \\
& x_t^i \leqslant M_t^i y_t^i & & t = 1,\dots,n; \\
& y_t^i \in \{0,1\}, s_t^i \geqslant 0, x_t^i \geqslant 0 & & t = 1,\dots,n; \\
& s_0 = 0
\end{aligned}
$$