Université de Nantes - UFR Sciences

Année 2010-2011 - M1 ORO

TP Recherche opérationnelle 1

Xavier Gandibleux

Organisation : TP1 - Préparation + 4 séances de 1h20

1. Problème de production de type ULS

Soit un problème de planification de la production d'un produit unique comprenant une demande variant dans le temps, des coûts de démarrage fixes et des coûts unitaires de productions. Le problème ne comporte aucune contrainte de capacité de production.

• Ce problème est connu sous le nom de *Uncapacitated Lot-Sizing Problem (ULS)* qui s'énonce de la façon suivante (voir figure 1). Soit un horizon de production composé de n périodes. La demande pour le produit en période t est $d_t \geq 0$ pour $t = 1, \ldots, T$. Pour chaque période t, on connait un coût de production p_t, un coût de stockage h_t pour les produits restants en fin de période, et un coût fixe f_t de démarrage de la production en période t.

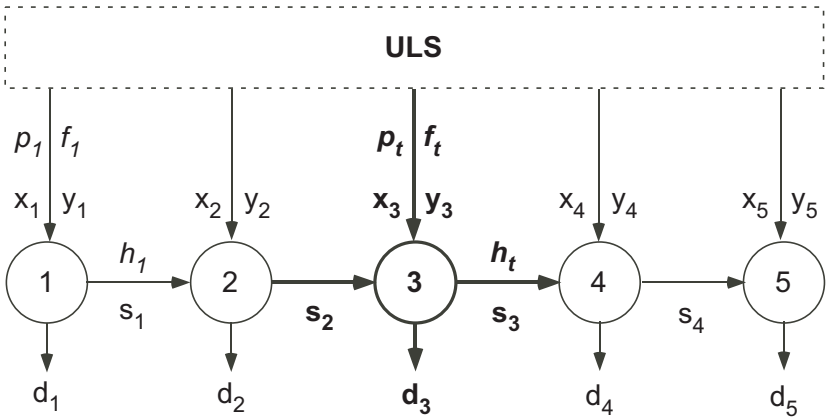

FIGURE 1 – Exemple d'un ULS sur 5 périodes. Pour la période 3, x_3 représente la quantité produite (produit divisible) sur la période, s_3 est le stock en fin de période et y_3 est une variable bivalente qui indique si l'outil de production est en activité sur cette période.

QUESTION : Ecrire un programme linéaire qui formalise le problème ULS. Faites valider votre modèle avant de passer à la question suivante.

• Si on considère l'instance suivante pour le problème de ULS :

t	1	2	3	4	5
d_t	3	5	6	3	8
p_t	2	4	6	8	10
h_t	3	2	3	2	-
f_t	10	8	6	4	2

TABLE 1 – Une instance de ULS (T=5)

QUESTION : Ecrire le modèle instancié. Proposer un algorithme de branch and bound qui sépare un sommet sur la variable y_t. Donner l'arborescence construite en vous aidant d'un solveur LP pour les besoins de résolution. Quelle est la solution optimale ?

• Soit le langage de modélisation GNU MathProg qui permet de modéliser un ULS facilement et de résoudre l'instance proposée via l'appel au solveur GLPK en mode MIP.

QUESTION : Implémenter le modèle sous GNU MathProg sous la forme la plus générique possible et rapporter la solution optimale trouvée par GLPK. Observer et relever l'activité du solveur.

• Analyser finement les résultats récupérés. Que pouvez-vous déduire sur la valeur des variables ? Que pouvez-vous proposer sur les valeurs des paramètres de votre modèle ?

QUESTION : A l'aide de votre algorithme de branch & bound, rapporter la solution optimale trouvée avec les propositions apportées à l'issue de votre analyse. Donner l'arborescence construite pour votre algorithme de branch & bound en vous aidant d'un solveur LP pour les besoins de résolution. Comparer avec la résolution précédente.

• Analyser finement votre formulation. Proposez un ensemble complémentaire de contraintes agissant comme des coupes (inégalités valides).

QUESTION : A l'aide de votre algorithme de branch & bound, rapporter la solution optimale trouvée avec ces coupes ajoutées à l'issue de votre analyse. Donner l'arborescence construite pour votre algorithme de branch & bound en vous aidant d'un solveur LP pour les besoins de résolution. Comparer avec la résolution précédente.

• Analyser finement les résultats récupérés. Que pouvez-vous dire sur la structure des solutions ?

QUESTION : Donner une représentation visuelle de la solution.

• Pour le problème de ULS, il apparaît très souvent en pratique une structure particulière sur les coûts. On parle de lot-sizing avec les coûts de Wagner-Whitin si $p_t + h_t \geq p_{t+1}$ sur toutes les périodes t.

QUESTION : Construire une instance sur 5 périodes répondant aux coûts de Wagner-Whitin. Quels sont les conséquences immédiates des coûts de Wagner-Whitin ? Peut-on exploiter cette particularité ? Argumentez votre réponse.

• Soit la formulation suivante :

$$
\begin{aligned}
\min \quad z = \quad & 2x_1 + 4x_2 + 6x_3 + 8x_4 + 10x_5 \\
+ \quad & 3s_1 + 2s_2 + 3s_3 + 2s_4 \\
+ \quad & 10y_1 + 8y_2 + 6y_3 + 4y_4 + 2y_5
\end{aligned}
$$

$$
\begin{aligned}
w_{11} &= 3 \\
w_{12} + w_{22} &= 5 \\
w_{13} + w_{23} + w_{33} &= 6 \\
w_{14} + w_{24} + w_{34} + w_{44} &= 3 \\
w_{15} + w_{25} + w_{35} + w_{45} + w_{55} &= 8
\end{aligned}
$$

$$
\begin{aligned}
w_{11} &\leq 3y_1 \\
w_{12} &\leq 5y_1 \\
w_{13} &\leq 6y_1 \\
w_{14} &\leq 3y_1 \\
w_{15} &\leq 8y_1
\end{aligned}
$$

$$
\begin{aligned}
w_{22} &\leq 5y_2 \\
w_{23} &\leq 6y_2 \\
w_{24} &\leq 3y_2 \\
w_{25} &\leq 8y_2
\end{aligned}
$$

$$
\begin{aligned}
w_{33} &\leq 6y_3 \\
w_{34} &\leq 3y_3 \\
w_{35} &\leq 8y_3
\end{aligned}
$$

$$
\begin{aligned}
w_{44} &\leq 3y_4 \\
w_{45} &\leq 8y_4
\end{aligned}
$$

$$
w_{55} \leq 8y_5
$$

$$
\begin{aligned}
x_1 &= w_{11} + w_{12} + w_{13} + w_{14} + w_{15} \\
x_2 &= w_{22} + w_{23} + w_{24} + w_{25} \\
x_3 &= w_{33} + w_{34} + w_{35} \\
x_4 &= w_{44} + w_{45} \\
x_5 &= w_{55}
\end{aligned}
$$

$$
\begin{aligned}
s_1 &= x_1 - 3 \\
s_2 &= x_1 + x_2 - 8 \\
s_3 &= x_1 + x_2 + x_3 - 14 \\
s_4 &= x_1 + x_2 + x_3 + x_4 - 17
\end{aligned}
$$

$$
w_{ij} \geq 0, \quad 0 \leq y_t \leq 1 \quad i \leq j \quad i = 1, \ldots, 5 \quad j = 1, \ldots, 5
$$

$$
s_t \geq 0, \quad x_t \geq 0, \quad y_t \in \{0, 1\} \quad t = 1, \ldots, 5
$$

QUESTION : A l'aide de votre algorithme de branch & bound, rapporter la solution optimale trouvée le modèle ci-dessous. Donner l'arborescence construite pour votre algorithme de branch & bound en vous aidant d'un solveur LP pour les besoins de résolution. Que constatez-vous ? Quelle conclusion tirez-vous ?

Livrable

- un rapport (pdf rédigé sous latex) discutant chacun des points demandés
- date de remise : 26/10/2010 sur madoc

Commentaires

Le problème ULS connaît depuis les années 50 un nombre conséquent de contributions scientifiques. Cela s'explique par l'importance que revêt ce problème en gestion de production, où il se retrouve souvent dans des versions plus complexes. Le lecteur intéressé trouvera notamment dans le livre de Laurence Wolsey (1998) un ensemble de résultats sur ce problème. Par exemple, un algorithme polynomial fondé sur le principe de la programmation dynamique y est présenté.

DEVOIR LIBRE : lire l'article de L. Wolsey "Progress with single-item lot-sizing", European Journal of Operational Research 86 (1995) 395-401, disponible sur madoc.

Concernant le solveur

Le package GLPK (GNU Linear Programming Kit) est un solveur de programmes linéaires sous licence GNU. Il peut traiter des programmes linéaires en variables continues (LP) de grande taille et des programmes linéaires en variables mixtes (MIP). C'est une collection de routines écrites en C ANSI et organisées sous la forme de librairie appelable via API. Il est disponible (version 4.39) à l'adresse `http://www.gnu.org/software/glpk/`. D'autres solveurs non-commerciaux sont disponibles comme

- LPSOLVE (`http://lpsolve.sourceforge.net/5.5/`)
- CLP sous Coin-OR (`http://www.coin-or.org/projects/`), etc.

Vous êtes invités à prendre connaissance de ces trois solveurs en parcourant les sites web respectifs. GLPK sera recommandé pour la réalisation des TP mais vous êtes libre de retenir le solveur LP de votre choix. Pour les besoins des TP il sera attendu de vous les points suivants :

1. Lire la section 1.3 du manuel de référence de GLPK. Des exemples y sont présentés et conditionnés pour être traité par GLPK (procéder de la même façon si vous utilisez un autre solveur).
2. Résoudre un exemple avec GLPK (mode 'usage en boîte noire" et mode "application propriétaire faisant usage des API")
3. Rapporter la solution optimale trouvée.
4. Parcourir les API de GLPK de façon a vous imprégner de ses principes et principales fonctionnalités avec l'objectif de résoudre des LP au sein d'un branch & bound.