RAIRO. RECHERCHE OPÉRATIONNELLE

C. COSTA

Formalisation et résolution des problèmes de découpes linéaires

RAIRO. Recherche opérationnelle, tome 16, n° 1 (1982), p. 65-82

<http://www.numdam.org/item?id=RO_1982__16_1_65_0>

R.A.I.R.O. Recherche opérationnelle/Operations Research
(vol. 16, n° 1, février 1982, p. 65 à 82)

FORMALISATION ET RÉSOLUTION DES PROBLÈMES DE DÉCOUPES LINÉAIRES (*)

par M. C. COSTA (1)

Résumé. — *Nombreuses sont les entreprises ayant à résoudre des problèmes de découpes de verre, de papier, de tôle..., qui peuvent être effectuées dans une ou plusieurs dimensions (longueurs, rectangles ou volumes).*

Nous nous intéressons seulement ici, aux problèmes de découpes linéaires et présentons tout d'abord les matériaux divers dans lesquelles elles sont effectuées.

Cet article propose une formalisation très générale de ces problèmes, puis une formalisation plus spécifique de certains d'entre eux.

Les différentes méthodes classiques utilisées pour les résoudre, fondées par exemple sur la programmation linéaire ou le calcul statistique, sont ensuite rapidement passées en revue.

Nous proposons enfin deux méthodes originales, arborescente ou aléatoire, qui se sont avérées efficaces.

Un tableau récapitulatif des problèmes et des méthodes de résolution est présenté dans la conclusion.

Mots clés : Découpes linéaires, programmation mathématique, heuristique.

Abstract. — *Cutting stock problems often occur in industries such as glass, paper, sheet metal, etc. The cuts can be performed in one or several dimensions (lengths, rectangles, volumes).*

In this paper only one-dimensional cutting is dealt with ("linear" problems): first the classical cases of linear cutting and the materials involved are recalled; next a very general formalisation of the problems is given, followed by a more specific one for a simpler (but frequent) case. Then the usual approaches based upon linear programming (cutting patterns) or upon statistics, are briefly reviewed. Finally two original methods are proposed: a boolean depthfirst treesearch and a random method; they both proved efficient in real life cases. In the conclusion, a table summarizing the various cases and adapted method is given.

Keywords: One-dimensional cutting problem, mathematical programming, heuristic.

INTRODUCTION

Définissons tout d'abord les problèmes de découpes : comment, dans un « matériau » de dimensions données, découper (ou disposer) un nombre maximal d'éléments de dimensions plus petites ? Ou encore, comment découper (ou disposer) un nombre donné d'éléments de manière à utiliser une quantité minimale du « matériau » de base ?

(*) Reçu mars 1981.
(1) Conservatoire national des Arts et Métiers.

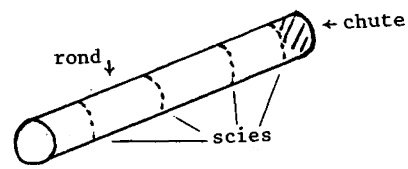
Étant donnés la variété et le nombre de problèmes de ce type, nous ne nous intéressons ici qu'à ceux d'entre eux qui revêtent un caractère unidimensionnel (découpes linéaires).

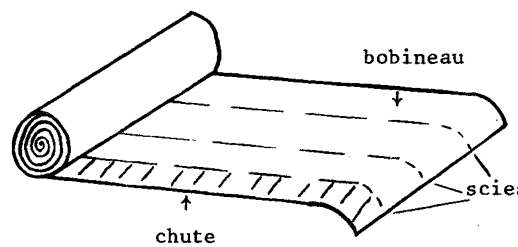
Après avoir présenté les divers problèmes de « découpes » couramment rencontrés nous en proposons une formalisation très générale. Nous passons ensuite rapidement en revue les méthodes classiques permettant de les résoudre, et proposons enfin des méthodes originales.

Divers problèmes concrets de découpes linéaires

Il peut s'agir de découpes réellement effectuées :

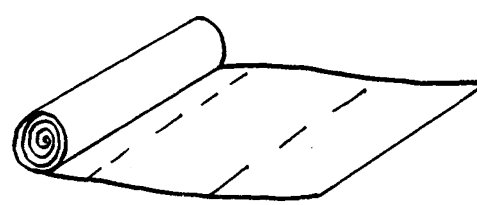
– découper des « ronds » de longueur et nombre commandés dans des barres d'acier en stock;





– où des bobineaux de petites largeurs dans des bobines de papier ou de tôle (la longueur d'un bobineau est celle de la bobine dans laquelle on le découpe);

– ou encore des rectangles dans des rouleaux de moquette (la largeur de ces rectangles est celle du rouleau). On obtient alors une chute au bout du rouleau.

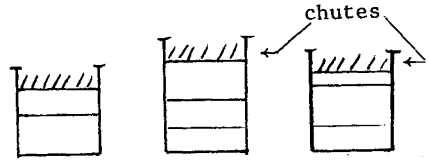


On peut également procéder à des partages :

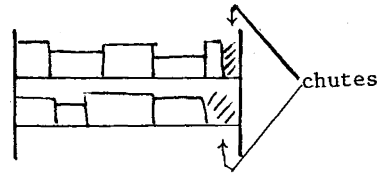
(*a*) « découpes » de volumes ou de poids :

– emballer des articles de poids ou volumes donnés dans des boîtes de capacités connues;

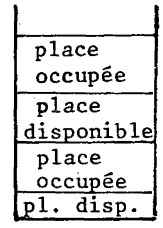
(*a*)



(*b*)



(*c*)



(*b*) « découpes » d'espaces :

– ranger des objets de largeurs données sur des rayons d'entrepôt (c'est le « problèmes du bibliothécaire ») [3];

– répartir des programmes d'ordinateur entre différentes zones de mémoire qui sont disponibles dans la machine (gestion de mémoire par zones);

(*c*) « découpes » de temps :

– affecter des tâches, qui ne peuvent être morcelées et dont on connaît le temps d'accomplissement, à divers processeurs disponibles durant un temps donné chaque jour.

Des caractéristiques particulières différencient, ou au contraire regroupent, ces problèmes : peut-on avoir une surproduction ? Les commandes (pour une période de temps) sont-elles connues ou arrivent-elles aléatoirement ? La quantité de matériaux de base disponibles est-elle fixée ou limitée ? Et enfin, et surtout, quelles sont les dimensions des problèmes et des variables, en général inhérentes au type du problème ?

Ce sont les réponses à ces différentes questions qui permettront de choisir l'une ou l'autre des méthodes proposées plus loin (*cf.* tableau final). Cependant, nous montrerons tout d'abord qu'il est possible de donner une formalisation très générale de ces problèmes.

1. FORMALISATION

1.1. Problème général (1)

Nous allons formaliser le problème concret des découpes de bobines de tôle, comme programme mathématique en nombres entiers.

Notons les *données* :

L_j et Λ_j, $j=1, \ldots, n$: largeurs et longueurs des n bobines en stock;

l_i et λ_i, $i=1, \ldots, m$: largeurs et longueurs des m bobineaux commandés [avec $\forall(i, i') : l_i \neq l_{i'}$].

Précisons qu'un bobineau de largeur l_i et longueur λ_i peut être obtenu en plusieurs morceaux, ou sous-bobineaux de même largeur l_i. De plus, il est tout à fait possible d'admettre une surproduction, c'est-à-dire des « surlongueurs » de bobineaux qui seront remises en stock.

Les *inconnues* seront :

$$x_{ij}, \qquad i=1, \ldots, m \quad j=1, \ldots, n,$$

nombre de sous-bobineaux de largeurs l_i à découper dans la j-ième bobine;

$$r_j=1 \text{ ou } 0 : \; r_j=1 \text{ si la bobine } j \text{ est utilisée}$$
$$r_j=0 \text{ sinon.}$$

La modélisation prend en compte les deux principaux types de *contraintes* usuellement rencontrées, contraintes de commande et contraintes physiques :

– la somme des longueurs des sous-bobineaux de largeur l_i découpés doit être supérieure à la longueur commandée dans cette largeur : contraintes (III);

– la somme des largeurs des sous-bobineaux découpés dans une bobine doit être inférieure à la largeur de la bobine : contraintes (IV).

Étudions enfin la *fonction économique* : les deux critères d'optimisation intéressant les praticiens sont, d'une part, la *surface totale de chute obtenue* : Z_1, que l'on cherchera à minimiser, et d'autre part, la *surface totale des bobines utilisées* : Z_2, que l'on cherchera également à minimiser. Ces deux critères étant,

hélas, souvent contradictoires, nous avons introduit un coefficient β, permettant de pondérer l'un d'eux par rapport à l'autre :

$$Z^* = MIN\,[Z_1 + \beta Z_2],$$

soit :

$$Z^* = MIN\,[\sum_j ((r_j . L_j - \sum_i x_{ij} . l_i) . \Lambda_j) + \beta . \sum_j r_j . l_j . \Lambda_j].$$

En posant $\alpha = \beta + 1$ et en regroupant les termes $\sum_j r_j L_j$ nous obtenons la formalisation complète suivante :

(1)

$$x_{ij} \in \mathbb{N}, \qquad \forall i, \forall j \quad \text{(I)},$$

$$r_j \in \{0, 1\}, \quad r_j = 1 \quad \text{si} \quad \sum_i x_{ij} \neq 0, \quad r_j = 0 \quad \text{sinon} \quad \text{(II)},$$

$$\sum_{j=1}^{n} x_{ij} . \Lambda j \geqq \lambda_i, \qquad \forall i \in \{1, \ldots, m\} \quad \text{(III)},$$

$$\sum_{i=1}^{m} x_{ij} . l_i \leqq L_j, \qquad \forall j \in \{1, \ldots, n\} \quad \text{(IV)},$$

$$Z^* = MIN\,[\sum_j [(\alpha . r_j . L_j - \sum_i x_{ij} . l_i) . \Lambda_j]].$$

1.2. Étude du coefficient β ($\alpha = \beta + 1$)

Pour certaines valeurs de β, la résolution du programme ci-dessus conduira à minimiser soit la quantité de chute, soit la quantité du matériau de base utilisé.

PROPRIÉTÉ 1 : *Pour* $0 < \beta < 1/\sum_j L_j . \Lambda_j$, *si* $Z = Z_1 + \beta Z_2$ *est minimal, alors* Z_1 *l'est aussi.*

Démonstration : Remarquons tout d'abord que ni Z_1, ni Z_2 ne peuvent excéder la surface totale disponible en stock : $\sum_j L_j . \Lambda_j$.

Soient deux solutions admissibles de valeurs :

$$z = z_1 + \beta z_2 \qquad \text{et} \qquad z' = z'_1 + \beta z'_2.$$

Montrons que, pour :

$$0 < \beta < \frac{1}{\sum_j L_j . \Lambda_j},$$

$$(\forall (z_1, z_2) \text{ et } (z'_1, z'_2)) : \quad z'_1 < z_1 \quad \Rightarrow \quad z'_1 + \beta z'_2 < z_1 + \beta z_2 \quad \Leftrightarrow \quad z' < z.$$

c'est-à-dire que, si Z_1 n'est pas minimal, alors Z ne l'est pas non plus.

- Si $z'_2 \leqq z_2$ l'implication est évidemment toujours vérifiée :
- Soit $z_2 < z'_2$:

$$[0 \leqq z_2] \qquad \text{et} \qquad [z'_2 \leqq \sum_j L_j . \Lambda_j]$$

entraînent :

$$[0 < z'_2 - z_2 \leqq \sum_j L_j . \Lambda_j] \qquad \text{d'où} \quad \frac{1}{z'_2 - z_2} \geqq \frac{1}{\sum_j L_j . \Lambda_j} > \beta.$$

Donc : $\beta(z'_2 - z_2) < 1$.

Or, $[z'_1 < z_1]$ et $[z_1$ et z'_1 entiers$]$ entraînent $[1 \leqq z_1 - z'_1]$. Donc : $[z'_1 < z_1]$ entraîne :

$$[\beta(z'_2 - z_2) < z_1 - z'_1] \qquad \text{d'où} \quad z'_1 + \beta z'_2 < z_1 + \beta z_2.$$

C.Q.F.D.

PROPRIÉTÉ 2 : *Pour* $\beta > \sum_j L_j . \Lambda_j$, *si* $Z = Z_1 + \beta Z_2$ *est minimal, alors* Z_2 *l'est aussi.*

La démonstration sera analogue à la précédente, en posant :

$$\beta' = \frac{1}{\beta}.$$

Enfin, pour éviter des découpes de chute nulle, mais inutiles, nous devons poser $\beta \neq 0$, excepté dans le cas ci-après (1.3).

1.3. Problème annexe fréquent (2)

Il est souvent possible de déterminer à l'avance la liste exacte des bobines en stock qui seront découpées (par simulation ou d'après des expériences antérieures). Les r_j et $Z_2 = \sum_j r_j . L_j . \Lambda_j$ sont alors fixés et nous cherchons à minimiser les chutes, ce qui revient à *maximiser la surface totale découpée.* Le programme ainsi défini est linéaire en nombres entiers :

(2)

$$\begin{aligned} & x_{ij} \in \mathbb{N}, \qquad \forall i, \ \forall j \quad \text{(I)}, \\ & \sum_j x_{ij} . \Lambda_j \geqq \lambda_i, \qquad \forall i \in \{1, \ldots, m\} \quad \text{(III)}, \\ & \sum_i x_{ij} . l_i \leqq L_j, \qquad \forall j \in \{1, \ldots, n\} \quad \text{(IV)}, \\ & MAX [\sum_i \sum_j x_{ij} . l_i . \Lambda_j]. \end{aligned}$$

Exemple : Si l'on dispose de deux bobines : $L_1 = 12$, $\Lambda_1 = 20$ et $L_2 = 11$, $\Lambda_2 = 30$ pour découper 2 bobineaux : $l_1 = 3$, $\lambda_1 = 70$, $l_2 = 5$, $\lambda_2 = 40$ on écrira : $x_{ij} \in \mathbb{N}$, i et $j \in \{1, 2\}$:

$$\text{(III)} \qquad \begin{cases} 20x_{11} + 30x_{12} \geqq 70, \\ 20x_{21} + 30x_{22} \geqq 40, \end{cases}$$

et :

$$\text{(IV)} \qquad \begin{cases} 3x_{11} + 5x_{21} \leqq 12, \\ 3x_{12} + 5x_{22} \leqq 11, \end{cases}$$

$$[MAX]\, Z = 150x_{22} + 100x_{21} + 90x_{12} + 60x_{11}.$$

1.4. Conclusion

Les problèmes présentés se formalisent tous ainsi : L_j et Λ_j sont les dimensions des matériaux de base, l_i et λ_i celles des éléments commandés; cependant, *dans la plupart des cas, les matériaux n'ont qu'une dimension : alors,* $\Lambda_j = 1$ *pour tout* j *et* λ_i *est le nombre d'éléments de dimension* l_i *commandés*, pour $i = 1, \ldots, m$. On pourra se reporter au tableau présenté dans la conclusion générale pour obtenir les correspondances concrètes entre les divers problèmes.

2. MÉTHODES DE RÉSOLUTION

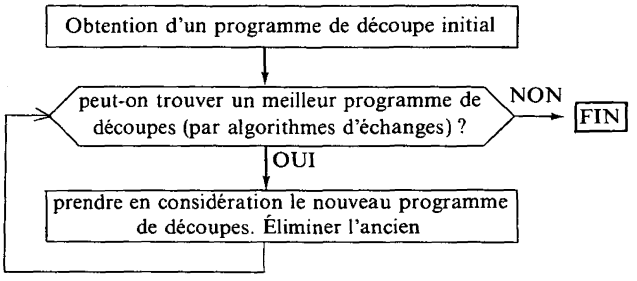
Nous faisons ici une revue des méthodes variées utilisées pour résoudre les problèmes de découpes linéaires, en insistant un peu plus sur les méthodes nouvelles que nous proposons (2.2). Rappelons que la méthode choisie devra tenir compte de la nature du problème à résoudre : les barres d'acier sont découpées par centaines : la découpe de quelques barres « de trop » n'a alors pas grande importance; on se contentera de chercher des solutions approchées. En revanche, il est impensable de scier inutilement une bobine de sept ou huit tonnes de tôle; de plus, dans ce dernier cas, seule la chute intervient généralement dans la fonction économique, et la taille de ces problèmes est suffisamment réduite pour espérer aboutir à une solution exacte.

Nous appellerons « programme de découpes » toute solution admissible du système des contraintes (I), (III), (IV) [et (II) dans (1)].

2.1. Quelques méthodes classiques

a) Méthodes heuristiques

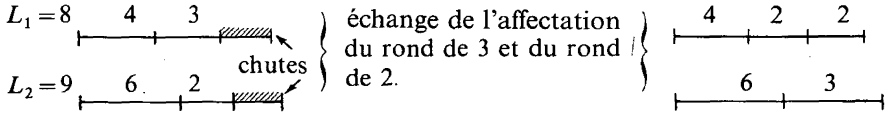
Elles permettent souvent d'obtenir rapidement une bonne solution du problème (1). Les diverses méthodes que nous avons pu rencontrer fonctionnent toutes selon ce simple schéma général :



– Obtention d'un programme de découpes initial : il peut être déterminé par une méthode gourmande (par exemple : affecter un à un les éléments commandés aux premiers matériaux de base de la liste qui conviennent [3, 6]) ou par une méthode plus judicieuse tenant compte par exemple de certains critères permettant de donner la priorité à tel ou tel matériau de base [10].

– Recherche d'un meilleur programme de découpes : algorithmes d'échanges.

Un exemple : Soit la découpe initiale donnée par :



La nouvelle découpe est de chute nulle et fournit un rond de 2 supplémentaire.

Ces échanges peuvent être effectués de proche en proche (ce qui ne peut convenir qu'aux petits problèmes), en considérant deux découpes successives de la liste proposée [3] ou (pour les autres problèmes) après une recherche rapide des seuls échanges intéressants [10].

Soulignons enfin que ces méthodes permettent de considérer des critères secondaires grâce au choix des matériaux de base utilisés (par exemple : taille du stock), et de traiter des problèmes importants (centaines de découpes).

b) *La méthode classique des plans de coupe* [5] (fondée sur la programmation linéaire)

Comme les précédentes, elle ne fournit qu'une solution approchée du problème (1).

La résolution se décompose en deux phases que nous illustrerons par un

petit exemple : On veut obtenir 4 ronds de $3(l_1)$ et 5 ronds de $4(l_2)$ dans des barres en stock de longueurs $9(L_1)$ et $10(L_2)$, le coût d'une découpe étant ici proportionnel à la longueur de la barre du stock utilisée.

Phase 1 : *Détermination des « plans de coupe »* ou façon de découper les matériaux de base selon les dimensions des éléments commandés : ce sont *les solutions admissibles des contraintes* (I) *et* (IV) *de* (1).

Soient A_k; $k=1, \ldots, K$; les K solutions trouvées : $A_k=(a_{ik},\ i=1, \ldots, m)$ et $a_{ik}=$nombre d'éléments de dimension l_i obtenus pour le plan de coupe A_k; alors, $C_k=(\alpha L_k-\sum_i a_{ik}.l_i).\Lambda_k$ est le coût associé à A_k.

Exemple (suite) : $x_{ij}\in\mathbb{N}$;

$$3x_{11}+4x_{21}\leqq 9 \qquad \text{et} \qquad 3x_{12}+4\,x_{22}\leqq 10.$$

On trouve, dans L_1 :

$$A_1=\begin{bmatrix}3\\0\end{bmatrix}, \qquad A_2=\begin{bmatrix}1\\1\end{bmatrix}, \qquad A_3=\begin{bmatrix}0\\2\end{bmatrix},$$

de coûts $9M$, et dans L_2 :

$$A_4=\begin{bmatrix}3\\0\end{bmatrix}, \qquad A_5=\begin{bmatrix}2\\1\end{bmatrix} \qquad \text{et} \qquad A_6=\begin{bmatrix}0\\2\end{bmatrix}$$

de coûts $10M$ ($M=$Cte).

Phase 2 : *Recherche de la solution du programme* $\{(II),(III),Z^*\}$ *de* (1), c'est-à-dire recherche des valeurs à donner aux variables x_k de sorte que si x_k est le nombre d'exécutions du plan de coupe A_k,

$$\text{(PL)} \qquad \begin{cases} x_k\geqq 0 \quad (\forall k), \\ \sum_{k=1}^{K} a_{ik}.\Lambda_k.x_k\geqq\lambda_i \quad (\forall i), \\ [MIN]\sum_k C_k.x_k. \end{cases}$$

Notons que les contraintes d'intégrité sur x_k ont été supprimées du (PL) qui sera résolu par l'algorithme du simplexe : chaque colonne du tableau représente un plan de coupe. Une solution entière, mais non exacte, sera obtenue en arrondissant la solution continue trouvée (rappelons que les valeurs des variables sont de l'ordre des centaines).

Exemple (suite) :

$$3x_1 + x_2 + 3x_4 + 2x_5 \geqq 4,$$

$$x_2 + 2x_3 + x_5 + 2x_6 \geqq 5,$$

$$[MIN]\, Z = 9x_1 + 9x_2 + 9x_3 + 10x_4 + 10x_5 + 10x_6.$$

Le simplexe donne $x_3 = 3/2$, $x_5 = 2$, $Z = 67/2$, $x_i = 0$ si $i \neq 3$ ou 5.

Soit, en arrondissant : 2 fois le plan de coupe A_3, c'est-à-dire 2 fois 2 ronds de 4, et 2 fois A_5, soit 2 fois 2 ronds de 3 et 1 rond de 4.

Enfin, nous nous intéressons ici à des problèmes de taille telle qu'il est impossible de résoudre globalement la phase 1. Il faudra donc, selon le principe de décomposition de Dantzig et Wolfe, utiliser en phase 2 la méthode révisée du simplexe [9] : on déterminera (phase 1) les *seules* colonnes (ou plans de coupe) permettant d'obtenir une solution de base (départ de la phase 2) (en résolvant un problème auxiliaire de sac à dos), puis, pour chaque itération du simplexe, on déterminera, de même, LA colonne (s'il en existe une) qui permettra, en entrant dans la base, d'améliorer la solution en cours.

c) *Méthodes heuristiques statistiques*

Aucune des méthodes précédentes ne peut être appliquée lorsque les commandes arrivent une à une aléatoirement et que la découpe doit être effectuée immédiatement, ce qui est fréquent. Dans ce cas, il faudra étudier statistiquement la distribution des commandes, puis calculer, ensuite, la probabilité d'avoir une chute supérieure à un seuil fixé si la découpe est effectuée dans tel ou tel matériau de base [7]. Des méthodes de simulation peuvent également être utilisées.

2.2. Méthodes nouvelles

a) *Une méthode arborescente pour le problème* (2)

Si le problème (1) ne peut être traité que par de bonnes heuristiques, il est en revanche possible de donner une solution exacte du problème (2) pour un nombre réduit de dimensions de base (on découpe rarement plus de 10 bobines de tôle à la fois).

• Pour cela nous avons tout d'abord tenu compte de *caractéristiques spécifiques* de ces problèmes de découpes [1] :

Le temps de réglage des scies sur les machines est important, il est donc préférable de répartir les sous-bobineaux de même largeur sur différentes bobines. Pour cela, on posera $x_{ij} \leqq 3$ (x_{ij}=nombre de sous-bobineaux de largeur l_i coupés dans la bobine j), ce qui permet de décomposer x_{ij} en deux variables bivalentes :

$$\forall i, \quad \forall j, \qquad x_{ij}=2\,y_{ij}+z_{ij} \quad \text{avec } y_{ij} \text{ et } z_{ij} \in \{0, 1\}$$

(Si l'on veut être moins strict et pour d'autres types de problèmes concrets on pourra toujours poser $x_{ij} \leqq 7$ et décomposer x_{ij} en 3 variables booléennes, le problème restant le même.)

• Nous avons ensuite ajouté des contraintes obtenues par déduction du problème initial :

La surface totale des bobineaux obtenus doit être supérieure à celle des bobineaux commandés :

$$\boxed{Z \geqq \sum_i l_i . \lambda_i} \qquad \text{(V)}$$

[(V) est déduite de (III) et Z].

D'autre part, elle sera toujours inférieure à la surface totale des bobines disponibles :

$$\boxed{Z \leqq \sum_j L_j . \Lambda_j} \qquad \text{(VI)}$$

[(VI) est déduite de (IV) et Z].

Ces deux contraintes seront ajoutées à (III) et (IV) dans (2).

De plus, les remarques suivantes nous ont permis d'accélérer sensiblement la recherche de la solution :

Notons e_i l'expression de gauche des inéquations (III) et f_j celle des inéquations (IV) :

$$e_i=\sum_j x_{ij} . \Lambda_j, \qquad f_j=\sum_i x_{ij} . l_i, \qquad \forall i, \forall j.$$

Alors : $Z=\sum_i l_i . e_i=\sum_j \Lambda_j . f_j$. Autrement dit, Z est une combinaison linéaire des expressions des contraintes. Deux nouvelles séries de contraintes (III') et (IV')

vont pouvoir être ajoutées au problème (2). Soient A et B les bornes inférieure et supérieure de Z [(V) et (VI)] : $A \leqq Z \leqq B$:

$$Z \leqq B \quad \Rightarrow \quad e_k \leqq \frac{B - \sum_{i \neq k} l_i . \lambda_i}{l_k}, \qquad (\forall k \in \{1, \ldots, m\}). \tag{III'}$$

Démonstration (B, l_i, λ_i et l_k sont de valeurs connues). Quel que soit k, on a :

$$Z \leqq B \Leftrightarrow \sum_i l_i . e_i \leqq B \quad \Rightarrow \quad l_k . e_k \leqq B - \sum_{i \neq k} l_i . e_i,$$

or, $(\forall i) : e_i \geqq \lambda_i$. Donc :

$$B - \sum_{i \neq k} l_i . e_i \leqq B - \sum_{i \neq k} l_i . \lambda_i, \qquad \text{et} \qquad l_k . e_k \leqq B - \sum_{i \neq k} l_i . \lambda_i. \quad \blacksquare$$

$$Z \geqq A \quad \Rightarrow \quad f_{k'} \geqq \frac{A - \sum_{j \neq k'} \Lambda_j . L_j}{\Lambda_{k'}}, \qquad (\forall k' \in \{1, \ldots, n\}). \tag{IV'}$$

Démonstration (A, Λ_j, L_j et $\Lambda_{k'}$ sont de valeurs données). Quel que soit k', on a :

$$Z \geqq A \quad \Leftrightarrow \quad \sum_j \Lambda_j . f_j \geqq A \quad \Rightarrow \quad \Lambda_{k'} . f_{k'} \geqq A - \sum_{j \neq k'} \Lambda_j . f_j$$

or :

$$(\forall j) : \quad f_j \leqq L_j \quad \Rightarrow \quad A - \sum_{j \neq k'} \Lambda_j . f_j \geqq A - \sum_{j \neq k'} \Lambda_j . L_j$$

et :

$$\Lambda_{k'} . f_{k'} \geqq A - \sum_{j \neq k'} \Lambda_j . L_j. \quad \blacksquare$$

Si $B = \sum_j \Lambda_j . L_j$, les contraintes (III') expriment le fait que la surface totale *découpée d'un* bobineau k doit être inférieure à la surface totale disponible moins la surface totale *commandée* des autres bobineaux.

Si $A = \sum_i l_i . \lambda_i$, les contraintes (IV') expriment le fait que la surface totale *découpée* dans *une* bobine k' doit être supérieure à la surface totale des bobineaux *commandés* moins la surface totale des autres bobines disponibles.

Nous remplacerons la fonction économique par la contrainte (V) : $A \leqq Z$, où A vaut au départ $\sum_i l_i . \lambda_i$. Puis, à chaque itération, A sera remplacé par la valeur de la nouvelle solution trouvée [+ PGCD $(l_i . \Lambda_j)$, si l'on ne cherche qu'une solution

optimale]. L'itération consistera à chercher une solution admissible de l'ensemble des contraintes ci-dessous, l'optimum ([*MAX*] Z) étant atteint lorsque le système n'admet plus de solution, ce qui revient à chercher les valeurs de y_{ij} et z_{ij}, avec $x_{ij}=y_{ij}+2\,z_{ij}$, telles que :

$$y_{ij}\in\{0,1\},\quad z_{ij}\in\{0,1\},\quad (\forall i\in\{1,\ldots,m\}),\quad (\forall j\in\{1,\ldots,n\}),$$

$$\lambda_i \leqq \sum_{j=1}^{n}(2\Lambda_j.y_{ij}+\Lambda_j.z_{ij}) \leqq \frac{\sum_j \Lambda_j.L_j-\sum_{k\neq i} l_k.\lambda_k}{l_i}, \qquad (\forall i)\quad \text{(III)-(III')},$$

$$\frac{A-\sum_{k\neq j}\Lambda_k.L_k}{\Lambda_j} \leqq \sum_{i=1}^{m}(2\,l_i.y_{ij}+l_i.z_{ij}) \leqq L_j, \qquad (\forall j)\quad \text{(IV)-(IV')},$$

$$A \leqq Z=\sum_i\sum_j(2\,l_i.\Lambda_j.y_{ij}+l_i.\Lambda_j.z_{ij}) \leqq \sum_j L_j.\Lambda_j.$$

Après avoir transformé l'écriture du système pour n'avoir que des inégalités de même sens, nous utiliserons, pour en trouver une solution, une procédure arborescente par séparation et évaluation séquentielle (PSES) les variables étant classées par ordre décroissant de leurs coefficients dans la fonction économique [4].

On prendra soin, après chaque choix fait en un sommet de l'arborescence (variable *choisie* égale à 0 ou à 1) de noter toutes les implications (variables à *fixer* à 0 ou à 1) dues à l'influence du choix sur les contraintes (on voit ici le rôle des contraintes (III') et (IV') qui fournissent de nombreuses implications).

Reprenons l'*exemple* donné au 1.3 :

— bornage de Z : $\sum_{i=1}^{2} l_i.\lambda_i \leqq Z \leqq \sum_{j=1}^{2}\Lambda_j.L_j$: $410 \leqq Z \leqq 570$;

— nouvelles contraintes :

$$Z=30\,e_1+50\,e_2=20\,f_1+30\,f_2$$

et :

$$e_1 \geqq 7,\qquad e_2 \geqq 4,\qquad f_1 \leqq 12,\qquad f_2 \leqq 10$$

$$Z \leqq 570 \;\Rightarrow\; e_1 \leqq \frac{570-200}{3}=12{,}3 \;\Rightarrow\; e_1 \leqq 12,$$

de même : $e_2 \leqq 7$.

$$Z \geqq 410 \;\Rightarrow\; f_1 \geqq 4 \quad\text{et}\quad f_2 \geqq 6;$$

– le système s'écrit alors avec :

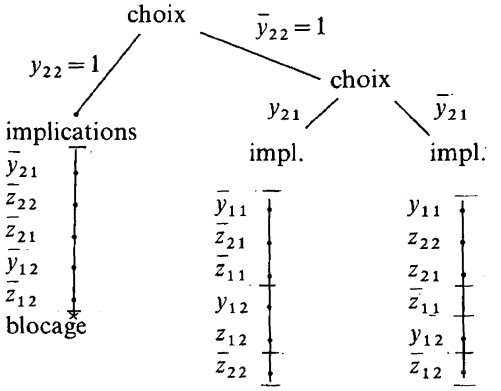
$$x_{ij}=y_{ij}+2\,z_{ij}, \qquad y_{ij}\in\{0,\,1\}, \quad z_{ij}\in\{0,\,1\},$$

$$\begin{cases} 7 \leqq 4\,y_{11}+2\,z_{11}+6\,y_{12}+3\,z_{12} \leqq 12, \\ 4 \leqq 4\,y_{21}+2\,z_{21}+6\,y_{22}+3\,z_{22} \leqq 7, \end{cases}$$

$$\begin{cases} 4 \leqq 6\,y_{11}+3\,z_{11}+10\,y_{21}+5\,z_{21} \leqq 12, \\ 6 \leqq 6\,y_{12}+3\,z_{12}+10\,y_{22}+5\,z_{22} \leqq 11, \end{cases}$$

$$41 \leqq 30\,y_{22}+20\,y_{21}+18\,y_{12}+15\,z_{22}+12\,y_{11}+10\,z_{21}+9\,z_{12}+6\,z_{11} \leqq 57;$$

– arborescence obtenue (on ne donne pas ici le tableau associé des coefficients).



SOLUTION 1 SOLUTION 2

Solution 1 :

$$y_{21}=y_{12}=z_{12}=1, \qquad y_{22}=y_{11}=z_{21}=z_{11}=z_{22}=0, \qquad \underline{V=47}.$$

Soit en remplaçant A par 47 :

$$f_1 \geqq \frac{480-330}{20}=7{,}5 \quad \Rightarrow \quad f_1 \geqq 8$$

et de même $f_2 \geqq 8$ et enfin $47 \leqq Z$ au lieu de $41 \leqq Z$.

Solution 2 :

$$y_{11}=z_{22}=z_{21}=y_{12}=1, \qquad y_{22}=y_{21}=z_{11}=z_{12}=0, \qquad \boxed{V=55}$$

Solution optimale :

$$x_{11}=2, \qquad x_{12}=2, \qquad x_{21}=1, \qquad x_{22}=1$$

— on découpera donc dans chacune des bobines 2 sous-bobineaux de largeur 3 et 1 sous-bobineau de largeur 5.

b) *Une méthode de cheminement aléatoire*

Une méthode purement aléatoire consisterait à créer des solutions possibles grâce à une loi (par exemple tirage de séries de nombres au hasard), puis à calculer la valeur économique de ces solutions avant, éventuellement, de les sélectionner. Hélas, le nombre de solutions théoriques du problème reste beaucoup trop grand pour permettre d'envisager une méthode aussi simple pour le résoudre : la probabilité de « tirer » LA meilleure solution est trop faible. Il va donc falloir combiner la logique à l'aléatoire. Comment choisir la méthode logique et quand l'employer? Cela dépendra des cas particuliers à traiter. Nous présentons ici ([2]) une résolution du problème (2) pour la découpe de bobines de tôle.

L'aléatoire

Deux choses sont possibles : considérer successivement les bobineaux et tirer au sort les bobines auxquelles les affecter, ou le contraire. Pour cela, on cheminera dans une table de nombres aléatoires introduite en mémoire; une table de correspondance permettra d'associer une bobine (ou un bobineau) à chaque nombre tiré. Les considérations exposées ci-dessous ont conduit à préférer la première solution (tirage des bobines).

Les bobineaux les plus larges sont les plus difficiles à placer; on aura intérêt à les placer en premier pour diminuer la probabilité d'aboutir à une solution impossible. Pour cela, on classera d'abord les bobineaux par largeurs décroissantes : cela serait impossible s'ils étaient tirés au hasard. Le classement préalable des bobineaux évitera de plus toutes les « variantes » résultant des permutations possibles des bobineaux sur chaque bobine et diminuera donc le nombre de solutions équivalentes que l'on pourra obtenir par des tirages différents. On ne pourra éviter celles résultant des permutations entre bobines pour chaque bobineau mais elles seront beaucoup moins nombreuses car le nombre de bobines est toujours très nettement inférieur au nombre de bobineaux sur une bobine.

La logique

Les bobineaux sont donc classés par ordre de largeurs décroissantes. Le processus aléatoire est abandonné lorsqu'il n'est plus avantageux : pour un

([2]) Avec l'accord de M. Billard qui l'a imaginée et en a vérifié les résultats.

bobineau donné de largeur k on effectue les tests suivants après chaque tirage (rappelons qu'un bobineau est obtenu en plusieurs morceaux) :

– s'il reste une largeur suffisante sur la première bobine tirée le bobineau est affecté à cette bobine.

Sinon :

– si aucune bobine n'a une largeur restante supérieure à k, la solution en cours est à rejeter et on recommence le tirage pour le premier bobineau;

– s'il reste exactement une largeur k sur une bobine ou si une seule bobine dispose d'une longueur supérieure à k, le bobineau est affecté à cette bobine;

– si plusieurs bobines ont une largeur supérieure à k, on tire au hasard parmi elles celle qui sera retenue;

– si, après obtention d'une solution, il reste des largeurs disponibles sur certaines bobines, on les « complètera » en y plaçant des bobineaux supplémentaires de petites largeurs.

Cette méthode est très simple à programmer et la solution optimale est très rapidement « tirée ». Cependant, il faut continuer la recherche encore assez longtemps pour pouvoir accepter cette solution comme étant la meilleure. Bien que le résultat reste malgré tout « aléatoire », utilisée parallèlement à la précédente, cette méthode a toujours fourni LA meilleure solution !

CONCLUSION

Toutes les méthodes présentées, classiques ou non, ont été testées avec succès par diverses entreprises (IBM, Simca, Jones Reinforcement LTD, etc.) ou par nous-mêmes. Remarquons toutefois que le problème est toujours traité pour des dimensions données des matériaux de base. Il pourrait être intéressant de commencer par rechercher les meilleures dimensions des matériaux à tenir en stock connaissant la répartition de celles des commandes.

Bien qu'adaptées aux problèmes de découpes linéaires, certaines méthodes présentées peuvent parfois être utilisées pour résoudre des problèmes à deux dimensions. Nous avons ainsi pu, à l'aide de plans de coupe et de la PL 01, proposer une bonne solution d'un problème de découpes de panneaux de bois qui nous avait été soumis par les responsables d'une entreprise de meubles [2].

Nous donnons enfin, un tableau récapitulatif des différents problèmes linéaires rencontrés.

Types de découpes	L_j	l_i	Λ_j	Problème à résoudre	Solutions proposées
Barres	*Longueurs* en stock	Longeurs des commandes	1	(1) Cas général (2) (1) Avec $\beta > \sum_j L_j$	Solution approchée par la programmation linéaire (P.L.) ou heuristique
Bobines (découpes longitudinales)	*Largeurs* en stock	Largeurs des bobineaux à découper	Longueurs des bobines en stock	(1) (2)	P.L. Méthode aléatoire ou P.L. 0.1.
Rouleaux (découpes transversales)	*Longueurs* en stock	Longueurs des commandes	1	(1) Mais ensemble des l_i et λ_i non connu au départ	Statistique
Boîtes	*Poids* acceptables	Poids des articles à placer ($l_i = p_i$)	1	(1) Avec $\beta > \sum_j L_j$	Heuristique
Rayonnages	*Longueurs* des rayons	Longueurs ou largeurs des articles à placer	1	(2)	P.L. 0.1. ou heuristique
Processus	*Temps* de disponibilité des processeurs	Temps demandés par les tâches à effectuer ($l_i = t_i$)	1	(2) Sauf (III)	P.L. 0.1. ou heuristique
Gestion de mémoire d'ordinateur par zones	*Places mémoire disponibles*, dans chaque zone	Places mémoire demandées par chaque programme ($l_i = m_i$)	1	(1) Mais ensemble des l_i non connu	Statistique

REMERCIEMENTS

Je tiens à remercier M. G. Billard pour les précieux et nombreux renseignements qu'il m'a fournis.

BIBLIOGRAPHIE

1. G. BILLARD, *Problème de découpes des tôles*, Communications non publiées, 1975, 30 p.
2. M. C. COSTA, *Problèmes de découpes linéaires-formalisation et solutions économiques*, Thèse de 3e cycle, Paris-VI, 1980, 228 p.
3. P. DUMONT, *Le problème du bibliothécaire*, Revue belge de statistique, vol. 12, n° 3, 1972, p. 5-33.
4. R. FAURE et Y. MALGRANGE, *Une méthode pour résoudre les programmes linéaires en nombres entiers*, Gestion 3, avril 1963, p. 48-56.
5. P. C. GILMORE et R. E. GOMORY, *A Linear Programming Approach to the Cutting Stock Problem*, Operations Research, vol. 9, 1961, p. 849-859.
6. R. L. GRAHAM, *The Combinatorial Mathematics of Scheduling*, Scientific american, vol. 238, n° 3, mars 1978, p. 124-132.
7. C. D. LITTON, *A Frequency Approach to the One Dimensional Cutting Problem*, Opérational Research Quart., vol. 28, n° 4, 1977, p. 927-938.
8. G. MOREAU, *Méthodes pour la résolution des problèmes d'optimisation de découpe*, Thèse de docteur ingénieur, Lyon, 1973, 180 p.
9. M. SIMMONARD, *Programmation linéaire-technique du calcul économique*, tomes 1 et 2, Dunod, Paris, 1973.
10. R. S. STAINTON, *The Cutting Stock Problem for the Stockholder of Steel Reinforcement Bars*, Operational Research Quart., vol. 28, n° 1, 1977, p. 139-149.
11. P. C. GILMORE, *Cutting Stock, Linear Programming, Knapsacking, Dynamic Programming, Some Interconnections*, Annals of Discrete Mathematics 4, Discrete optimization, tome 1, Ed. Hammer, Johnson et Korte, North Holland, 1979, p. 217-235.