**MASTER EN SCIENCES INFORMATIQUES**

# Méthodes de sélection de coils basées sur des techniques d'aide à la décision multicritère

Lamouline, Thierry

*Award date:*
1991

*Awarding institution:*
Universite de Namur

FACULTES
UNIVERSITAIRES
N.D. DE LA PAIX
**NAMUR**

INSTITUT D'INFORMATIQUE

# METHODES DE SELECTION DE COILS BASEES SUR DES TECHNIQUES D'AIDE A LA DECISION MULTICRITERE

Par Thierry Lamouline

Promoteur: Professeur J.P. Leclercq

Mémoire présenté en vue
de l'obtention du titre de
Licencié et Maître en Informatique

Année académique 1990-1991

## RESUME

Dans le souci constant de répondre aux exigences de la clientèle au meilleur coût, les entreprises demandent de plus en plus fréquemment à l'informatique de leur fournir une assistance dans les prises de décisions.

Dans le cadre de notre stage, nous avons participé au développement d'un logiciel destiné à aider les responsables d'une unité de production à sélectionner les produits à transformer et à déterminer l'ordre de leur traitement dans une installation donnée.

Notre travail concerne spécialement le problème de la sélection.

Nous décrivons l'analyse et les recherches qui ont orienté la construction de la méthode utilisée pour le résoudre, ainsi que les résultats jugés satisfaisants par les responsables de production qu'elle nous a permis d'obtenir.

Deux autres approches possibles, étudiées et développées par d'autres équipes, sont présentées.



## ABSTRACT

In recent years more than ever, decision aid systems appear to be powerful tools for decision takers in their task of evaluating alternative actions.

Industrial production is one of the fields where such systems can lead to significant improvements in quality and production delay.

This study concerns the design and development of a decision aid software for engineers working in steel industry.

This software determines the set of products that are to be transformed during the next 8 hours. Multiple competing criteria are taken into account using user defined priorities. Satisfying results are presented.

Alternative approaches for solving such a selection problem are also discussed.

Avant de présenter le travail qui clôture ma Licence et Maîtrise en informatique, je tiens à remercier chaleureusement tous mes professeurs et tous ceux qui ont contribué à ma formation.

J'adresse un merci tout spécial à ceux qui m'ont aidé au cours de cette année académique:

- Monsieur le Professeur J.P. Leclercq, promoteur de ce travail, pour ses conseils et les remarques constructives apportées aux premières versions de ce document.

- Monsieur F. Thill pour son accueil au sein du Service Informatique de l'ARBED dont il est responsable.

- Les membres de l'unité Modélisations et Optimisations et particulièrement Monsieur R. Bausch son responsable, ainsi que M. Lescrenier, chargé d'études qui m'a suivi tout au long de mon stage, et qui m'a offert six des mois les plus enrichissants de ma "carrière d'étudiant".

- Alain Gabriel chercheur au Centre de Recherche Public du Centre Universitaire de Luxembourg pour ses patientes explications relatives à PrologIII.

Je remercie enfin mes parents pour leur soutien constant tout au long de mes études.

# TABLE DES MATIERES

# Table des figures

# INTRODUCTION

Tous les jours dans les entreprises, les administrations, sont prises de nombreuses décisions qui consistent très souvent à choisir parmi les solutions possibles, celle qui satisfait le mieux à un ensemble de critères. Dans la plupart des cas, ces critères qu'il faut satisfaire au mieux sont contradictoires. En effet, lorsque qu'il s'agit de remplacer les pneus de la voiture, par exemple, certains désirent les plus résistants à l'usure, ceux qui assureront la meilleure sécurité et les moins chers.
Remarquons que l'existence de ce problème est généralisable à tout achat dans un système de concurrence.

Tous ces problèmes caractérisés par l'existence:

- d'au moins deux critères en conflit
- et d'au moins deux solutions alternatives possibles

constituent la classe des problèmes multicritères.
Ce travail va s'attacher à la résolution d'un problème multicritère caractérisé par un très grand nombre de solutions possibles.

Nous décrirons d'abord le problème qui a été à l'origine de ce travail tel qu'il se posait dans la société ARBED qui m'a accueilli pour un stage de cinq mois. La suite de ce travail sera composée de deux parties décrivant différentes approches utilisées pour résoudre ce problème.

La première présentera différentes manières de ramener ce problème multicritère à un problème monocritère par agrégation des différents critères, en insistant particulièrement sur celle qui a été retenue pour la

réalisation du logiciel répondant aux exigences de production.

Nous décrirons également la méthode utilisée pour obtenir la solution du problème monocritère.

Nous justifierons les décisions prises lors de l'implémentation du logiciel. Nous terminerons cette première partie par quelques résultats obtenus grâce à l'outil.

La deuxième partie introduira les travaux du projet d'Optimisation Dynamique de la Gestion de Production (mené par l'ARBED conjointement avec le centre de Recherche Public, Centre Universitaire à Luxembourg et le laboratoire LAMSADE, Paris-Dauphine). Nous y aborderons d'abord :

- des méthodes préconisées par les auteurs de l'école française. Nous montrerons, toujours sur le même problème, comment ces méthodes sont applicables aux problèmes multicritères de grande taille.
- ensuite une expérimentation basée sur un langage de programmation par contraintes de haut niveau: PROLOG III

# ENONCE DU PROBLEME DE SELECTION

## 2.1 L'ARBED un complexe sidérurgique.

Depuis plus d'un siècle, l'acier est le fondement des structures d'ARBED Luxembourg et intervient à l'heure actuelle pour quelque 53% du chiffre d'affaires du groupe ARBED, qui s'élève à 209 milliards de FLUX

Avec 400 sociétés, réunissant un effectif total de plus de 50.000 personnes, et une production annuelle, en 1990, de 7,7 millions de tonnes d'acier, le groupe ARBED est le cinquième groupe sidérurgique en Europe et le 13 ème producteur des pays à économie de marché.

Installé principalement au Grand-Duché du Luxembourg, l'ARBED y fournit des emplois à 10.000 personnes (soit 35% des salariés industriels), intervient pour près de 15% dans la formation du PIB et est de loin la plus importante entreprise du pays.

La production de l'acier sur le territoire luxembourgeois est assurée par cinq usines, spécialisées dans des gammes distinctes de produits:

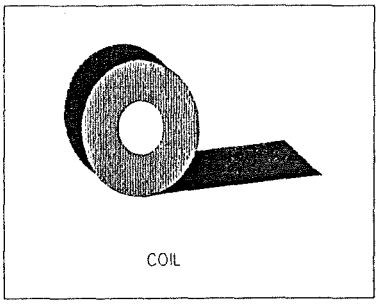
- L'usine d'Esch-Belval
- L'usine de Differdange
- L'usine d'Esch-Schifflange
- L'usine de Dudelange
- L'usine de Rodange

## 2.2 Le problème et son contexte d'apparition

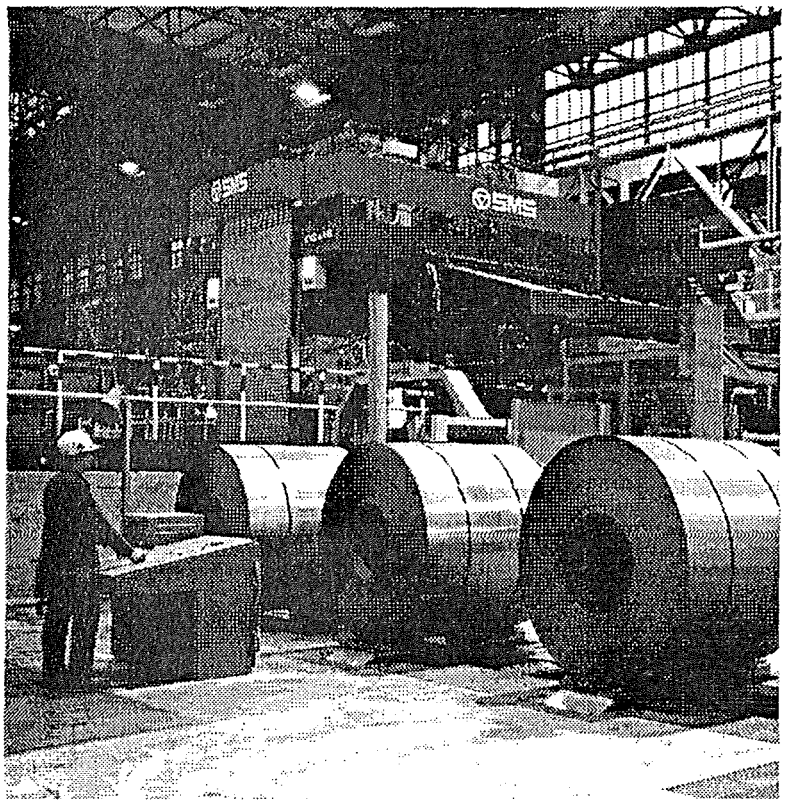
### 2.2.1 La production de coils à Arbed DUdelange (ADU)

L'usine de Dudelange est la plus petite des usines du ARBED. Elle occupe 850 personnes.

Elle réalise le **relaminage** (le laminage consiste à diminuer l'épaisseur et la rugosité) de coils (rouleaux de tôle) à froid laminés sur le train CVC de très haute performance, installé en 1988.


COIL

Il s'agit à l'heure actuelle de l'installation la plus moderne au monde.



Les installations de relaminage à Dudelange

Les largeurs des coils produits à ADU varient de 550 à 1600 mm et les épaisseurs de 0,25 à 2,99 mm. Un coil pèse au maximum 40 t et est laminé à une vitesse maximale de 1350 m/minute.

Les coils relaminés à ADU peuvent être répartis en 5 classes selon leur destination après laminage:

1) les coils pour la galvanisation à ADU,
2) les coils pour la société de galvanisation Galvalange,
3) les coils à découper en tôles(1),
4) les coils à feuillards(2),
5) les coils (3)(produits finis).

Les traitements des coils à ADU dépendent donc de la classe à laquelle ils appartiennent.

On distingue les traitements de:

L: relaminage,
R: rebobinage,
F: four,
S: skin pass,
C: cisaillage.

Chaque classe de coils nécessite des traitements propres. A titre indicatif:

| CLASSE | TRAITEMENTS |
|---|---|
| galvanisation | L, (R) |
| Galvalange | L, (R) |
| feuillards | L, R, (F, S) |
| tôles | L, R, F, S, C |
| coils | L, R, F, S |

Le plan de travail de la cage CVC qui réalise le relaminage est nommé programme de laminage. Il indique l'ordre de passage des coils à cette étape de la production. Il concerne habituellement une production de coils correspondant au temps d'usure d'un cylindre de laminage. Un programme de laminage peut être défini par un kilométrage de production (de l'ordre de 300 km par exemple) ou par un tonnage de production (de l'ordre de 1600 tonnes par exemple).

(1) La découpe s'opère perpendiculairement à la longueur du coil.

(2). La découpe, appelée refendage, produit de longues bandes de tôle de petite largeur

(3) Un coil peut ne subir qu'un traitement visant à améliorer sa rugosité ou à uniformiser son épaisseur. On obtient alors un produit fini qui a toujours l'aspect d'un rouleau de tôle.

### 2.2.2 La gestion de production informatisée

ADU va prochainement compléter le suivi automatique de la cage CVC par les modules d'ASIA (Application Suivi Intégré Arbed) s'occupant de la prise en charge des commandes

Ces modules permettront, par exemple, de générer la commande des matières premières et de déterminer l'ordre de passage à chaque étape de production (CVC, skin pass, four,...)

La suite de ce travail s'attachera au module réutilisable de sélection de coils à produire par une installation de production.

### 2.2.3 La sélection des coils à produire

Les quelque 40 coils qui constituent un programme de laminage sont sélectionnés chaque jour parmi les quelque 800 coils en stock. ( En provenance de producteurs du groupe ou étrangers)

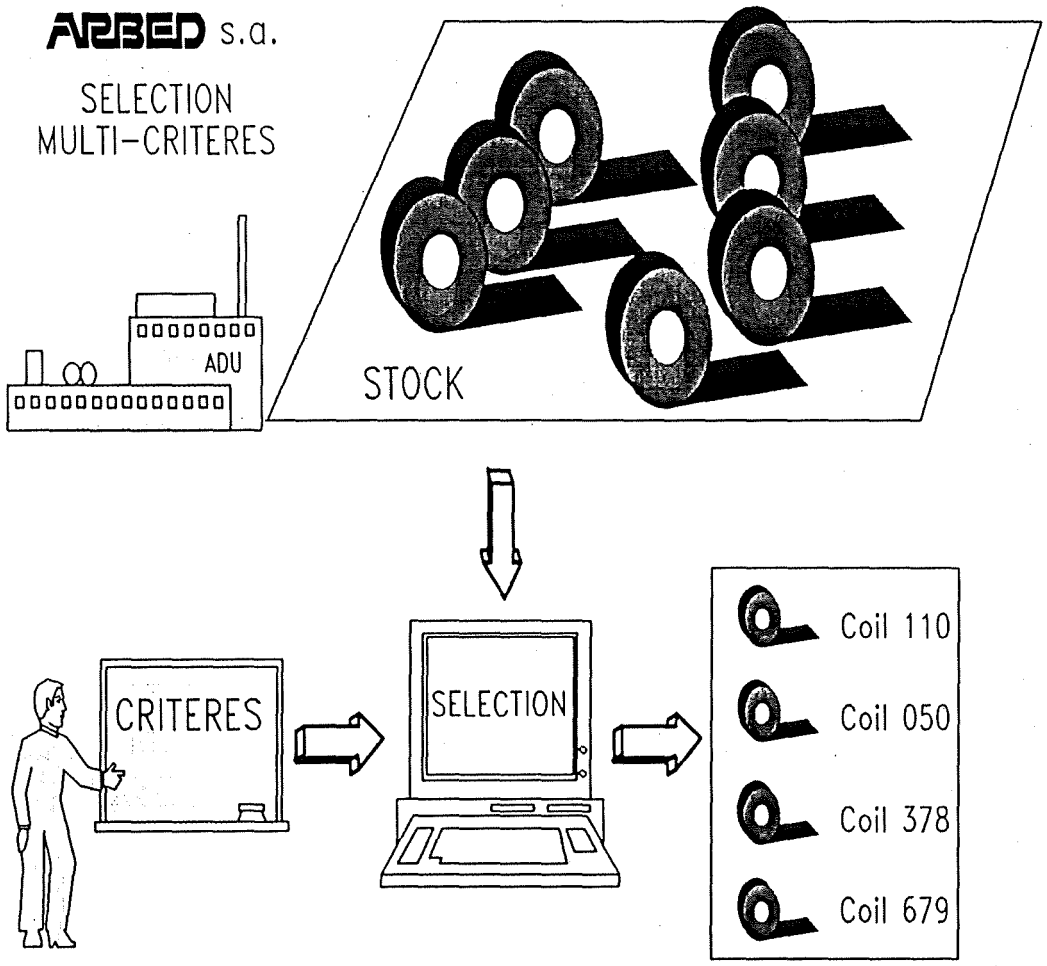
Cette sélection se fait selon certains critères (décrits à la section 2.3). Par exemple, ADU tente de produire un pourcentage adéquat de coils pour chacune des classes de destination après laminage.

Actuellement, cette sélection réalisée manuellement, ne permet pas d'introduire efficacement un critère de sélection important pour ADU: **choisir les coils les plus urgents.** De plus le temps-homme consacré à cette tâche est non négligeable (1 heure ou plus par jour).

## 2.3 Les requêtes des utilisateurs

Il a été décidé de rechercher un outil automatique de sélection des coils d'un programme de laminage (= relaminage L) permettant d'intégrer les critères essentiels de sélection.

Le schéma suivant donne un aperçu général du problème.




L'outil à concevoir doit être suffisamment souple de manière à permettre son utilisation pour la sélection des coils à chaque étape de la production (four et skin pass par exemple).

### 2.3.1 Fonctionnalités de la sélection automatique des coils

A) Caractéristiques des coils

La sélection des coils se fait sur base des valeurs de paramètres associés aux coils. Ainsi, un coil appartient à une certaine classe (un poste de commande(1) ou une classe de largeur(2) par exemple), possède certaines propriétés (être facile à laminer par exemple) et est associé à certaines mesures (sa longueur et son tonnage par exemple).

Les paramètres repris dans la table ci-dessous sont utilisés par les critères de sélection choisis actuellement à ADU. Leur signification apparaîtra clairement lorsque nous décrirons ces critères.

D'autres paramètres peuvent être définis pour les besoins de nouveaux critères de sélection introduits afin que l'outil de sélection réponde à d'autres attentes.

Pour chaque type de paramètres, nous indiquons le type et le nombre de valeurs qu'il peut prendre.

(1) Un poste de commande est défini comme l'ensemble des coils de caractéristiques identiques destinés à un même client.

(2) Une classe de largeur désigne l'ensemble des coils dont la largeur se situe dans un intervalle donné

| PROPRIETES | CLASSES | MESURES |
|---|---|---|
| facile à laminer<br>en attente<br>imposé | poste (de commande)<br>classe de largeur<br>classe de largeur-épaisseur<br>destination après laminage<br>diamètre intérieur | longueur<br>largeur<br>épaisseur<br>tonnage<br>productivité (h/t)<br>délai théorique de fin de laminage |
| 2 valeurs | n valeurs | un nombre illimité de valeurs |
| valeur = OUI ou NON | valeur = un identifiant alphanumérique | valeur réelle |
| | | |

Ainsi, un coil donné pourra appartenir à la classe DESTINATION de valeur "Galvalange", à la classe DIAMETRE INTERIEUR de valeur "610", avoir une valeur "NON" pour la propriété FACILE A LAMINER et une valeur "1200" pour la mesure LARGEUR.

B) Définition des critères

L'outil de sélection devra déterminer un ensemble de coils satisfaisant à un certain nombre de critères eux-mêmes répartis en deux catégories: les critères à respecter absolument ou à respecter au mieux. Pour chacun de ces critères nous donnerons un énoncé généralisé vu que, dès la définition du projet, les responsables d'ADU ont prévu la réutilisation du module à d'autres étapes de la production notamment au passage au four, au cisaillage,... De plus son utilisation pourrait être envisagée dans un contexte différent de celui des coils. Ce qui nous amène à substituer la notion d'objets à celle de coils et à introduire le concept de famille d'objets qui désigne l'ensemble des objets identiques (par exemple: tous les coils d'un même poste de commande constituent une famille).

| CRITERES A RESPECTER ABSOLUMENT |
|---|
| **A: "sélectionner uniquement des objets dans la classe C de valeur V". Par exemple:** |
| 1) afin que tous les coils sélectionnés aient un diamètre intérieur identique précisé par l'utilisateur (deux cas se présentent: 610 (90% des coils) ou 508 (10% des coils)) on utilisera la contrainte: sélectionner uniquement des coils dans la classe DIAMETRE INTERIEUR de valeur 610". |
| **B: "sélectionner uniquement des objets dont la mesure M a une valeur supérieure à la valeur MMIN et inférieure à la valeur MMAX". Par exemple:** |
| 2) sélectionner uniquement des coils dont la mesure LARGEUR a une valeur supérieure à la valeur 1000 et inférieure à la valeur 1200. |
| **C: "sélectionner absolument les objets de propriété P à valeur V". Par exemple:** |
| 3) l'utilisateur pourra forcer la sélection de certains coils s'il désire les voir obligatoirement présents dans le programme de laminage, en utilisant le critère: sélectionner absolument les coils de propriété IMPOSE à valeur OUI. |
| **D: "ne pas sélectionner les objets dont la propriété P a la valeur V". Par exemple:** |
| Ce critère ne semble pas utile à ce stade mais pourrait s'avérer utile ultérieurement. |

| CRITERES A RESPECTER AU MIEUX |
|---|
| **E: "choisir les objets de manière à minimiser la somme des mesures M". Par exemple:** |
| 5) les coils à produire sont ceux de DELAIS DE FIN DE LAMINAGE les plus courts |
| **F: "éviter la sélection des objets dont la propriété P a la valeur V". Par exemple:** |
| 6) Ce critère ressemble au critère D. Le critère D est toujours satisfait alors que le critère F peut ne l'être que partiellement. Afin de favoriser le laminage des coils par poste complet, les utilisateurs préciseront les coils dont les postes attendent encore des coils en provenance des fournisseurs grâce à la propriété EN ATTENTE. On utilisera la contrainte: éviter la sélection des coils dont la propriété EN ATTENTE a la valeur OUI. |
| **G: "le pourcentage des objets sélectionnés ayant une valeur V pour la propriété P est compris entre NMIN et NMAX". Par exemple:** |
| 7) le pourcentage des coils sélectionnés ayant une valeur OUI pour la propriété FACILE À LAMINER est compris entre 80 et 90 (la difficulté de laminage dépend de l'épaisseur du coil, de la qualité d'acier) |
| **H: "le pourcentage (en termes de la mesure M) des objets sélectionnés appartenant à la classe C de valeur V est compris entre NMIN et NMAX". Par exemple:** |
| $8_a$) le pourcentage des coils sélectionnés appartenant à la classe DESTINATION de valeur GALVALANGE est compris entre 15 et 25. |
| $8_b$) le pourcentage des coils sélectionnés appartenant à la classe DESTINATION de valeur GALVANISATION est compris entre 20 et 25. |
| $8_c$) le pourcentage des coils sélectionnés appartenant à la classe DESTINATION de valeur FEUILLARDS est compris entre 15 et 30. |

| |
|---|
| $8_d$) le pourcentage des coils sélectionnés appartenant à la classe DESTINATION de valeur TOLES est compris entre 20 et 30. |
| $8_e$) le pourcentage des coils sélectionnés appartenant à la classe DESTINATION de valeur COILS est compris entre 10 et 40. |
| **I: "si un objet est sélectionné dans une classe de type C, alors au moins n objets ayant même valeur de classe C seront sélectionnés. Par exemple:** |
| 9) pour que chaque classe de largeur prévue dans le programme de laminage soit représentée par au moins 5 coils, on utilisera le critère suivant: si un coil est sélectionné dans la classe LARGEUR, alors au moins 5 coils ayant même valeur de classe LARGEUR seront sélectionnés. |
| **J: "la somme des mesures M pour les objets sélectionnés est comprise entre les valeurs MMIN et MMAX". Par exemple:** |
| 10) les coils constituent une production d'un TONNAGE compris entre 1500 et 1700 tonnes |
| **K: "la moyenne des mesures M1 pondérées par les mesures M2 est comprise entre les valeurs MMIN et MMAX". Par exemple:** |
| 11) pour que la productivité (en h/t) obtenue pour le programme de laminage soit comprise entre une productivité minimale PRODMIN (h/t) et une productivité maximale PRODMAX (h/t), on utilisera la contrainte suivante: la moyenne des mesures PRODUCTIVITE pondérées par les mesures TONNAGE est comprise entre les valeurs PRODMAX et PRODMIN |

### 2.3.2 Résultats attendus de l'outil de sélection

L'outil de sélection fournira comme résultats:

1) le nombre d'objets sélectionnés dans chaque famille.
2) une indication de la manière dont les critères sont respectés.

### 2.3.3 Temps de réponse attendu de l'outil de sélection

La demande initiale prévoit que le calcul de la sélection des objets doit être réalisé en moins de 10 minutes.

# ELABORATION D'UNE METHODE DE RESOLUTION

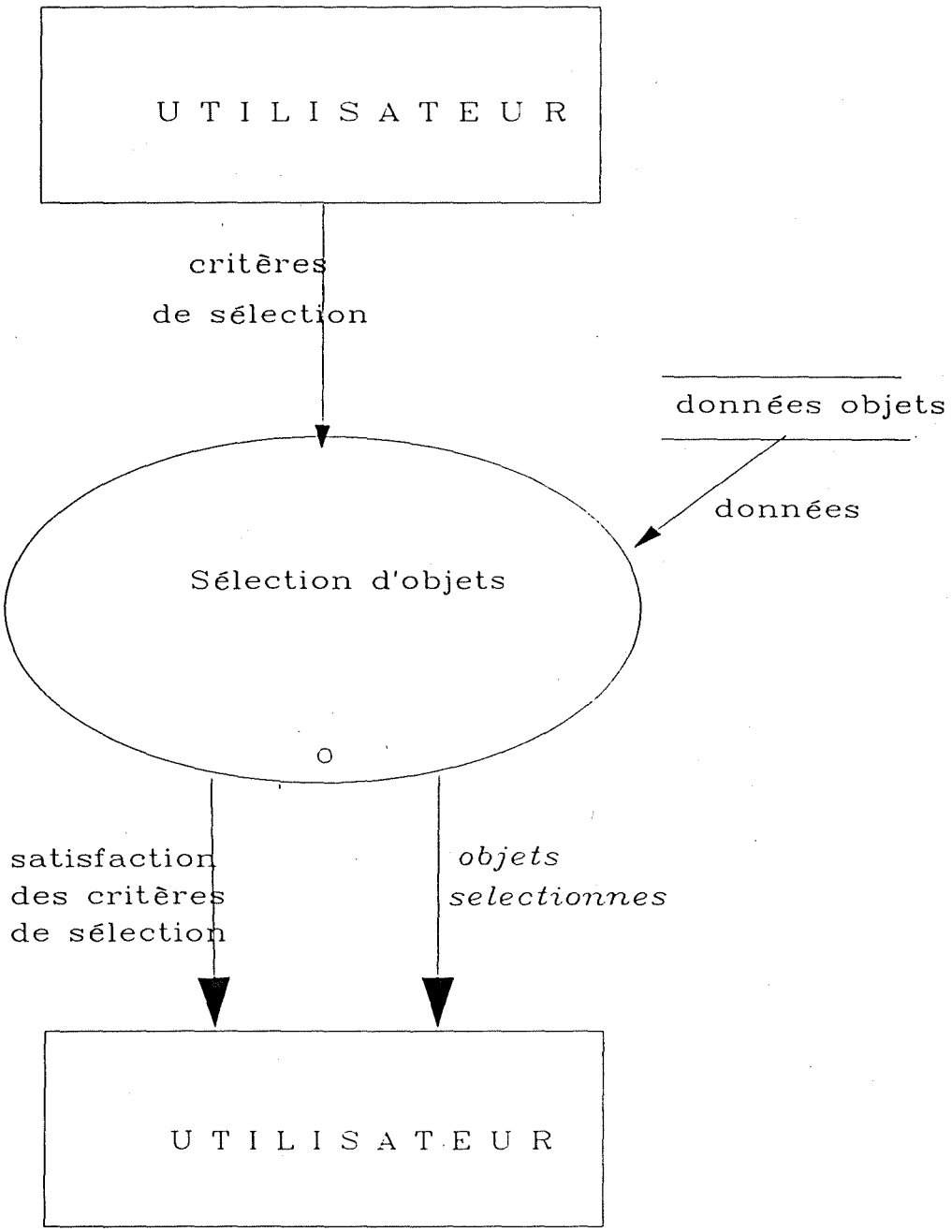
Après l'examen des fonctionnalités souhaitées par les utilisateurs nous choisissons une structure pour le logiciel à développer. La décomposition du problème ainsi réalisée permettra de répartir la réalisation des sous-problèmes entre les différentes équipes. Nous examinerons ensuite quelques techniques susceptibles d'apporter une solution au problème qui nous a été confié.

## 3.1 Structure du logiciel

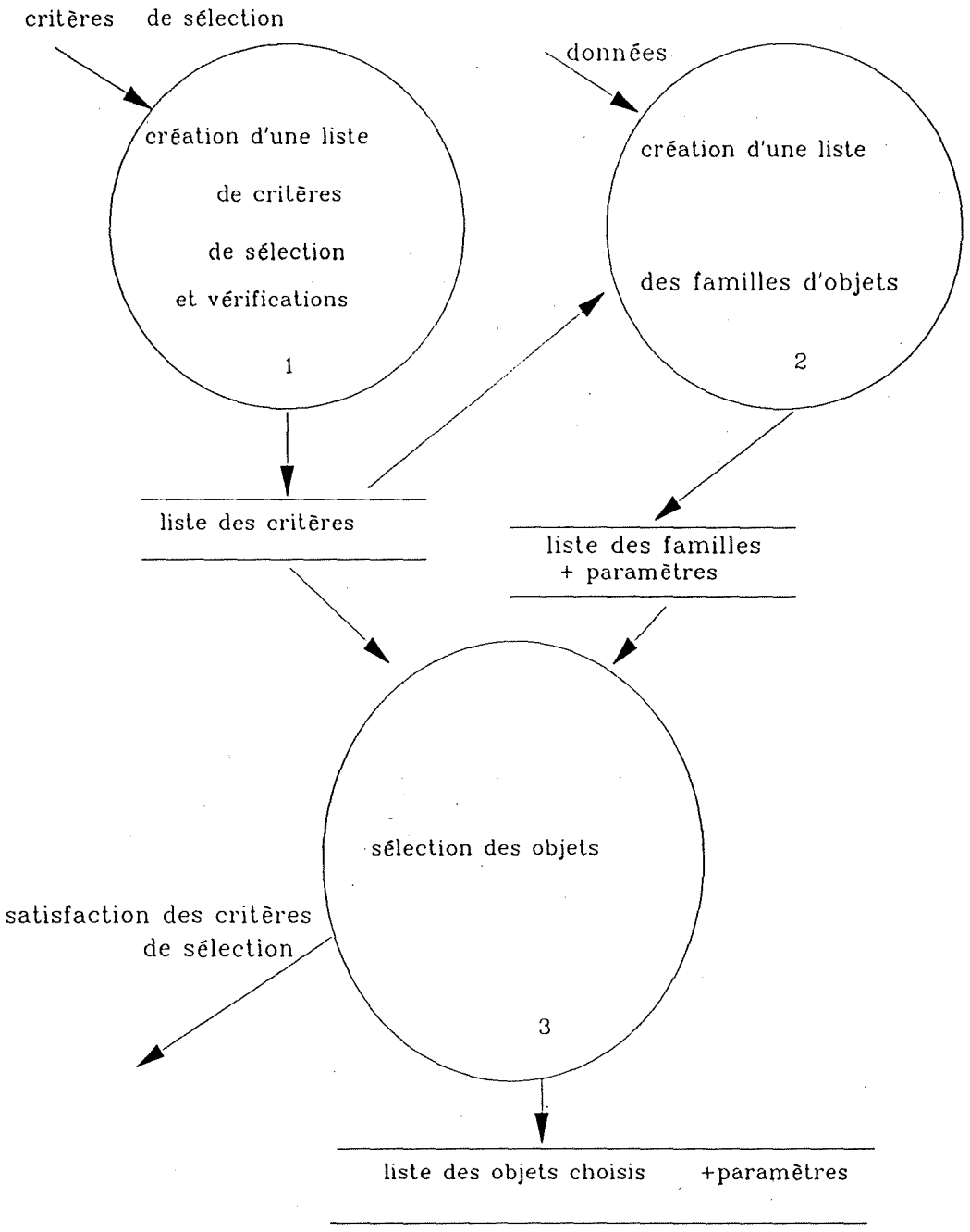
Le formalisme utilisé pour définir la structure du logiciel est celui proposé par Yourdon-Demarko. [Yourdon 89]

### 3.1.1 structure du logiciel de sélection

Le logiciel d'aide à la sélection d'objets nécessite, de la part de l'utilisateur, l'introduction de critères de sélection.
Il utilise également les données relatives aux objets à sélectionner, et transmet à l'utilisateur les objets sélectionnés ainsi qu'une évaluation du degré de satisfaction des critères pour la sélection effectuée.

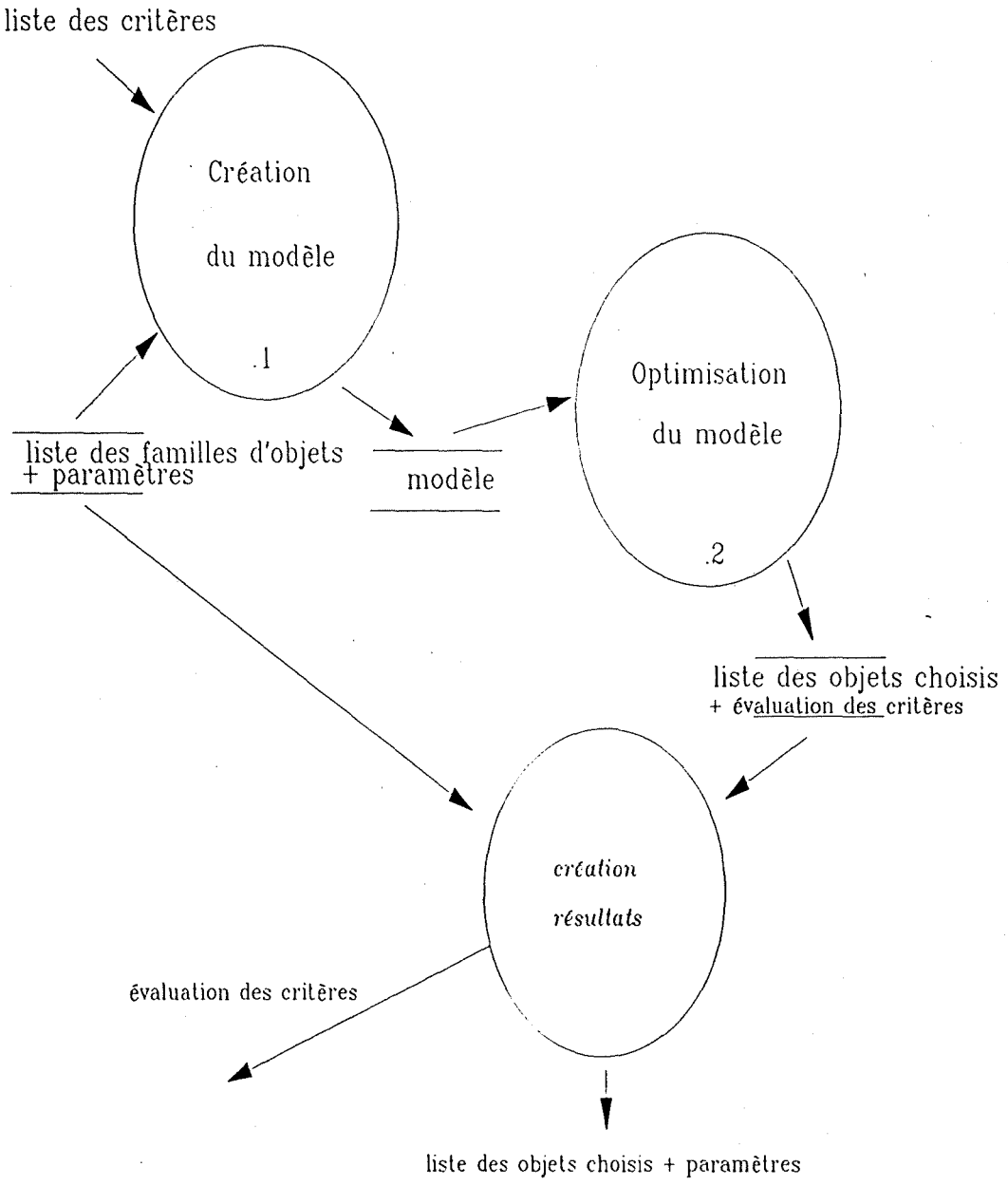



structure du logiciel de sélection




structure du logiciel de sélection

3.sélection des objets



### 3.1.2 Justification de la formalisation

Le traitement "sélection" se décompose en
1)La création d'une liste de critères de sélection fournis par l'utilisateur (avec vérification),
2)La création d'une liste des objets,
3)La sélection proprement dite.

Le traitement "sélection des objets" qui nous a été confié, nécessite la création d'un modèle qui pourra ensuite être optimisé. Pour réaliser le modèle, il faut disposer de:

la liste des critères avec leurs paramètres,

la liste des objets et leurs paramètres (mesures, classes et propriétés)

définissant la sélection.

## 3.2 Optimisation linéaire

Les premières recherches dans l'analyse du problème ont montré la possibilité d'utiliser les techniques de programmation linéaire.
Nous allons d'abord rappeler brièvement les principes généraux de ces techniques. On tentera ensuite de formuler le problème de sélection sous une forme linéaire.

### 3.2.1 Le problème linéaire

De nombreux problèmes de programmation mathématique se ramènent à déterminer le mieux possible l'allocation de ressources rares (le travail, le matériel, les machines et le capital) de façon à maximiser le bénéfice.

On tente de déterminer les valeurs des variables de décision (durée du travail sur une machine, nombre de pièces à produire,...) en respectant des contraintes ( les règles d'utilisation des ressources, par exemple: le temps de réalisation d'une pièce par la machine est de 5 minutes). Le but poursuivi est, quant à lui, une expression des variables de décision qu'il faut optimiser (communément appelée fonction objectif).

La programmation linéaire apporte une solution à une classe particulière des problèmes exposés ci-dessus.

A) Définition

Un problème relève de la programmation linéaire s'il vérifie les deux conditions suivantes:

1) La fonction objectif peut être exprimée sous forme d'une fonction linéaire des variables de décision (c.-à-d. une fonction mathématique ne faisant intervenir que la première puissance des variables de décision).

2) Les contraintes peuvent être exprimées par un ensemble d'équations et d'inéquations linéaires.

Mathématiquement ce problème peut toujours s'écrire de manière générale sous la forme suivante:

$$Minimiser \quad Z = \sum_{j=1}^{n} c_j x_j$$

sous les contraintes:

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i \qquad i = 1 \ldots m$$
$$x \in \Re$$

Remarquons, en effet, que tout problème de maximisation se ramène aisément en un problème minimisation.

B) exemple

En planification de production, un industriel dispose de deux machines $M_1$ et $M_2$ et fabrique deux biens A et B qui nécessitent chacun le recours aux deux machines.
Pour fabriquer une unité du bien A il faut utiliser $M_1$ durant 3 heures et $M_2$ durant 5 heures.
Pour fabriquer une unité du bien B il faut utiliser $M_1$ durant 5 heures et $M_2$ durant 2 heures. Compte tenu des contraintes techniques, la machine $M_1$ ne peut fonctionner plus de 15 heures par jour et la machine $M_2$ pas plus de 10 heures par jour. Sachant que A est vendu 5Frs l'unité et B 3Frs l'unité, déterminez les quantités respectives $x_1$ du bien A et $x_2$ de B que l'industriel doit fabriquer de manière à maximiser son chiffre d'affaires.
Formalisation:

Maximiser $Z = 5x_1 + 3x_2$ compte tenu du système de contraintes:

$$x_1 \geq 0 \wedge x_2 \geq 0$$
$$3x_1 + 5x_2 \leq 15$$
$$5x_1 + 2x_2 \leq 10$$

C) Succès de la programmation linéaire

La programmation linéaire est couramment utilisée pour résoudre de nombreux problèmes. Les principales raisons de cette large utilisation sont:

1) La variété des problèmes qui peuvent être modélisés sous forme linéaire (au moins en première approximation) est très grande.

2) Des techniques efficaces existent pour résoudre le problème linéaire. La plus connue étant très certainement la méthode du *simplexe* que nous introduisons à présent.

## D) Le SIMPLEXE

Due à George Dantzig en 1947, cette méthode n'a pas cessé d'être améliorée et étendue depuis.
Nous n'allons pas décrire ici les détails de l'implémentation de cet algorithme et encore moins la base théorique (présentée dans [Fichefet 82]) sur laquelle il prend appui. Nous nous limiterons à une présentation intuitive du procédé.

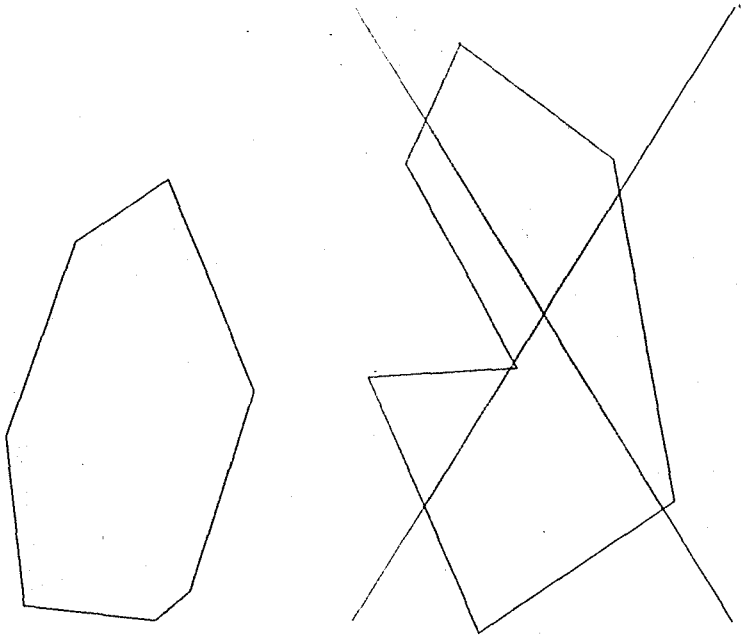
On peut démontrer que l'espace des contraintes (l'ensemble des solutions qui vérifient les contraintes) obtenu en programmation linéaire est obligatoirement un espace convexe (car l'intersection de convexes est un convexe).

### a) Les espaces convexes

Définition:

A est un espace convexe ssi le segment de droite joignant deux points quelconques de A est entièrement inclus dans A.
mathématiquement: $\forall a_1, a_2 \in A \subset \Re^n \forall \lambda \in [0,1] : \lambda a_1 + (1-\lambda) a_2 \in A$

Exemple dans $\Re^2$:

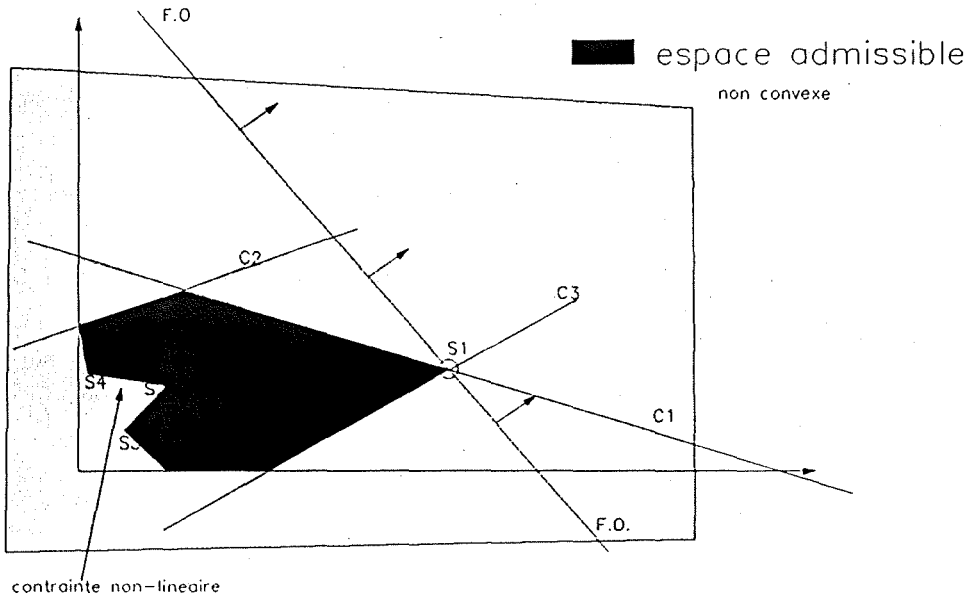



b) Idée intuitive de l'algorithme

L'algorithme détermine un sommet de départ (l'intersection de droites [hyperplans] de contraintes) et calcule pour chacun des sommets adjacents si la fonction objectif est supérieure, si on maximise, (ou inférieure si on minimise) à la valeur obtenue pour le sommet départ. Si tel est le cas, ce sommet devient le nouveau sommet de départ. Le processus s'arrête quand il n'existe plus de sommet adjacent améliorant la fonction objectif.

c) Justification intuitive

L'algorithme ci-dessus converge vers l'optimum (s1) recherché. S'il n'en était pas ainsi il devrait exister un sommet non optimum (s2) dont tous les voisins seraient "moins bons". Le morceau d'hyperplan joignant les voisins, (sur la figure: le segment joignant s3 et s4) ne serait pas entièrement dans l'espace des contraintes. Cet espace ne pourrait donc être convexe.




d) Complexité

Klee et Minty [Klee & Minty 72] ont démontré que l'algorithme du simplexe avait une complexité théorique exponentielle. Pourtant la complexité observée est souvent polynomiale.

D'autres méthodes ont été proposées pour résoudre le problème notamment:

-L'algorithme polynomial de Kachian(1) [Kachian 79]

-L'algorithme polynomial de Karmakar. [Karmakar 84]

-La méthode du gradient projeté de E.Jacquet-Lagrèze. [E.Jacquet-Lagrèze 87]

(1) cet algorithme d'intérêt pratique faible est le premier algorithme polynomial pour résoudre le problème linéaire.

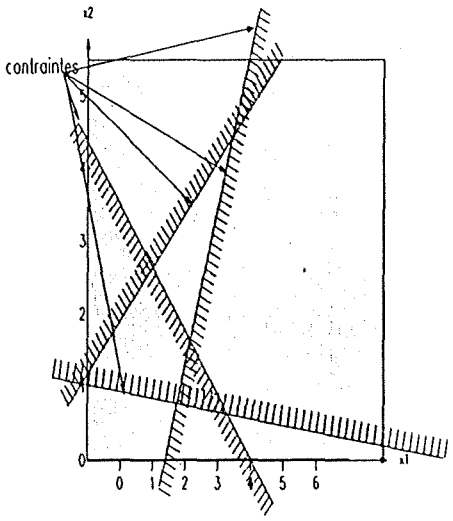
E) Contraintes incompatibles

Lors d'une modélisation en programmation linéaire, il peut arriver que le décideur formule des contraintes contradictoires ou incompatibles.

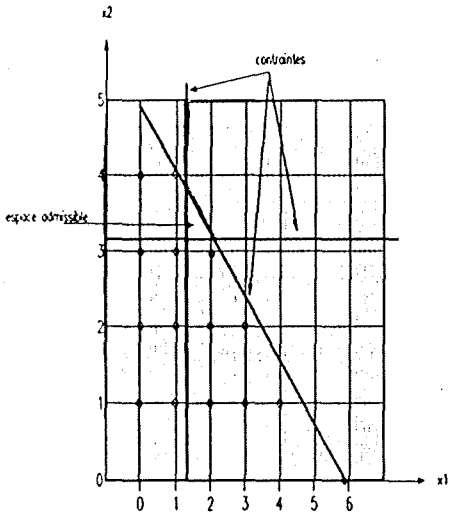
a) Définition

Un ensemble de contraintes sont dites incompatibles, s'il n'existe pas de solutions vérifiant simultanément ces contraintes.

Graphiquement:




espace des solutions = vide.




espace des solutions entières = vide.

b) Cas du problème de sélection

Le décideur de l'usine de Dudelange pourrait souhaiter une sélection qui comprenne entre 15% et 19% de coils de destination "Galvalange" et au

moins quatre coils par classe de largeur retenue. Il est possible qu'il n'y ait pas de solution c.-à-d., que pour atteindre le pourcentage désiré, il faille sélectionner un coil dans une classe de largeur mais que l'obligation d'en prendre quatre fasse dépasser le pourcentage souhaité.

c) Remède

Afin d'éviter d'obtenir un modèle dans lequel les contraintes seraient incompatibles on peut les rendre moins strictes. Cette approche n'est valable que si le décideur accepte une marge d'erreur sur le degré de satisfaction des contraintes. Pour ce faire on introduit dans chacune d'elles une variable de décision supplémentaire: $aX + bY + C > 6$ par exemple. Cette variable C (dite d'écart) est également introduite dans la fonction objectif de manière à être minimisée. De cette façon si les anciennes contraintes ne sont pas incompatibles, la variable introduite sera nulle. L'expression de la nouvelle contrainte est alors équivalente à l'ancienne puisque : $aX + bY > 6 \iff aX + bY + 0 > 6$. Par contre, si les anciennes contraintes sont incompatibles, la variable introduite prendra la valeur positive minimale rendant le nouveau système compatible. On obtient alors une solution qui tend à respecter les contraintes mais qui ne les respecte pas exactement!

F) Modélisation linéaire du problème de sélection

Avant de présenter la modélisation du problème de sélection nous définissons:

les variables de décisions:

$n_i$ = le nombre d'objets sélectionnés dans la famille[1] i.

les constantes suivantes:

(1) rappel: tous les objets d'une même famille sont identiques.

nobjets = nombre d'objets parmi lesquels la sélection peut avoir lieu.

nfamilles = nombre de familles parmi lesquelles la sélection peut avoir lieu.

nbof(i) = nombre d'objets dans la famille i

Cla(c,i,v) = 1 si la famille i appartient à la classe c de valeur v, 0 sinon.

mesure(m,i) = la valeur de la mesure m pour la famille i

prop(p,i,v) = 1 si la famille i a v pour valeur de la propriété p, 0 sinon.

L'optimisation du modèle aura pour but de déterminer les variables $n_i$ qui interviennent dans le modèle descriptif du problème de sélection.

CRITERES A RESPECTER ABSOLUMENT.

**A) Sélectionner uniquement des objets dans la classe C de valeur V.**

$n_i$ = 0 pour toute famille i tq Cla(C,i,V) = 0
$n_i$ indéterminé sinon.

**B) Sélectionner uniquement des objets dont la mesure M a une valeur supérieure à la valeur Mmin et inférieure à la valeur Mmax.**

$n_i$ = 0 si Mmin > mesure(M,i) ou si mesure(M,i) > Mmax
$n_i$ indéterminé sinon.

**C) Sélectionner absolument les objets dont la propriété P à la valeur V.**

$n_i$ = nbof(i) si prop(P,i,V)=1

$n_i$ indéterminé sinon

**D) Ne pas sélectionner les objets dont la propriété P à la valeur V.**

$n_i$ = 0 si prop(P,i,V)=1
$n_i$ indéterminé sinon

Remarque: afin d'éviter toute contradiction nous accorderons la priorité à la non sélection lorsqu'il y a conflit entre C) et D).

CRITERES A RESPECTER AU MIEUX .

**E) Choisir les objets de manière à minimiser la somme des mesures M.**

Il faut minimiser la fonction suivante:

$$\sum_{i=1}^{nfamilles} mesure(M,i).n_i$$

**F) Eviter la sélection des objets dont la propriété P à la valeur V.**

Il faut minimiser la fonction suivante:

$$\sum_{i=1}^{nfamilles} prop(P,i,V)n_i$$

Pour chaque contrainte, nous donnerons la modélisation linéaire classique, suivie de la modélisation dans laquelle on a introduit les écarts qui permettent de tenir compte d'éventuelles incompatibilités

Les critères sont modifiés légèrement en introduisant des variables d'écarts. Plus ces variables ( EXyz X = nom du critère, y,z = numérotation si nécessaire) seront petites et mieux les critères seront respectés. Ces écarts seront donc minimisés dans la fonction objectif et seront choisis positifs.

**G) Le pourcentage des objets sélectionnés en termes de la mesure M ayant une valeur V pour la propriété P est compris entre Nmin% et Nmax%.**

$$\sum_{i=1}^{nfamilles} mesure(M,i).n_i.Nmin \leq \sum_{i=1}^{nfamilles} 100.prop(P,i,V).n_i.mesure(M,i) \leq \ldots$$

$$\sum_{i=1}^{nfamilles} mesure(M,i).n_i.Nmax$$

$$\sum_{i=1}^{nfamilles} mesure(M,i).n_i.Nmin - EG_1 \leq \sum_{i=1}^{nfamilles} 100.prop(P,i,V).n_i.mesure(M,i) \leq \ldots$$

$$\sum_{i=1}^{nfamilles} mesure(M,i).n_i.Nmax + EG_2$$

**H) Le pourcentage des objets sélectionnés en termes de la mesure M appartenant à la classe C de valeur V est compris entre Nmin% et Nmax%.**

$$\sum_{i=1}^{nfamilles} mesure(M,i).ni.Nmin \leq \sum_{i=1}^{nfamilles} 100.Cla(C,i,V).ni.mesure(M,i) \ldots$$

$$\leq \sum_{i=1}^{nfamilles} mesure(M,i).ni.Nmax$$

$$\sum_{i=1}^{nfamilles} mesure(M,i).n_i.Nmin - EH_1 \leq \sum_{i=1}^{nfamilles} 100.Cla(C,i,V).n_i.mesure(M,i) \ldots$$

$$\leq \sum_{i=1}^{nfamilles} mesure(M,i).n_i.Nmax + EH_2$$

**I) "si un objet est sélectionné dans une classe de type C, alors au moins n objets ayant même valeur de classe C seront sélectionnés"**

Cette contrainte ne peut être modélisée à ce stade, il est impossible de l'exprimer sous forme linéaire. Nous y reviendrons dans la section suivante.

**J) La somme des mesures M pour les objets sélectionnés est comprise entre la valeur Mmin et Mmax.**

$$Mmin \leq \sum_{i=1}^{nfamilles} mesure(M,i)ni \leq Mmax$$

$$Mmin - EJ_1 \leq \sum_{i=1}^{nfamilles} mesure(M,i)ni \leq Mmax + EJ_2$$

**K) La moyenne des mesures M1 pondérées par les mesures M2 est comprise entre les valeurs Mmin et Mmax.**

$$Mmin \sum_{i=1}^{nfamilles} mesure(M2,i)n_i \leq \sum_{i=1}^{nfamilles} n_i.mesure(M1,i)mesure(M2,i) \leq \ldots$$

$$Mmax \sum_{i=1}^{nfamilles} mesure(M2,i)n_i$$

$$Mmin \sum_{i=1}^{nfamilles} mesure(M2,i)n_i - EK_1 \leq \sum_{i=1}^{nfamilles} n_i.mesure(M1,i)mesure(M2,i) \leq ..$$

$$Mmax \sum_{i=1}^{nfamilles} mesure(M2,i)n_i + EK_2$$

A ce stade nous voyons que nous pouvons, grâce à des relations linéaires, modéliser la majorité des contraintes du problème.
Cependant, deux problèmes au moins subsistent encore:

comment tenir compte de la contrainte I ?

plusieurs critères et non un seul (E et F) s'expriment sous forme de fonctions à minimiser.

Pour les résoudre nous allons recourir à des extensions de la programmation linéaire.
La première d'entre elles, la programmation en nombres entiers va rendre possible le traitement de la contrainte I, la seconde, l'optimisation multicritère nous permettra de tenir compte simultanément de plusieurs fonctions objectifs.

### 3.2.2 Le problème linéaire entier (PLE)

L'optimisation du modèle précédent aurait donné aux variables de décision $n_i$ des valeurs réelles or la production ne traite que des coils entiers. La technique du problème linéaire entier rencontrera cette préoccupation.

A) Définitions

Un problème est linéaire en nombres entiers s'il vérifie les conditions 1 et 2 de la définition d'un problème linéaire et la condition suivante:

Les variables utilisées dans la modélisation ne prennent que des valeurs entières.

Un problème est linéaire mixte si seulement certaines variables utilisées dans la modélisation doivent être entières.

B) Exemple

$$\text{maximiser} \quad z = 3x_1 + 2x_2$$

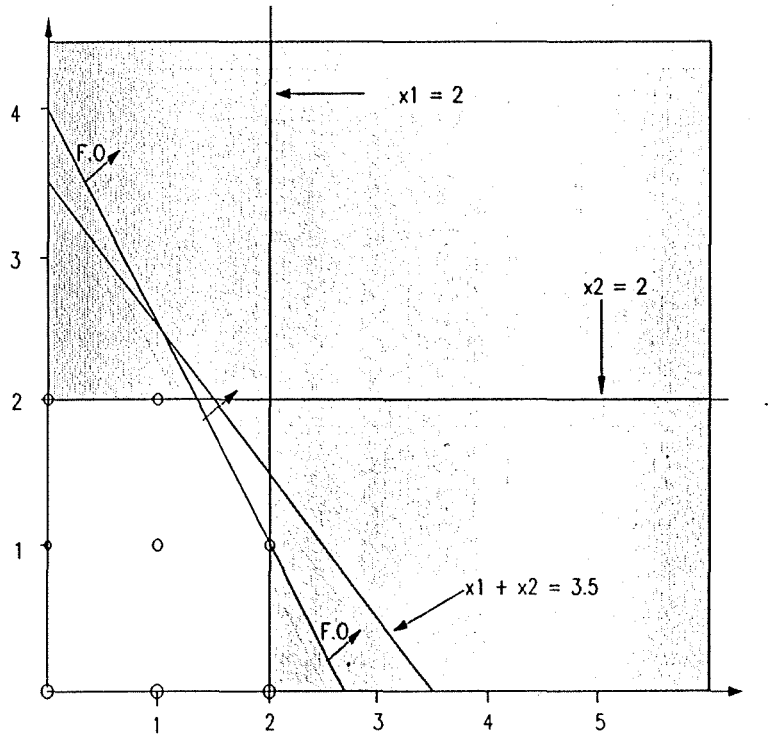
sc:

$x_1 <= 2$

$x_2 <= 2$

$x_1 + x_2 <= 3.5$

$x_1$ et $x_2 >= 0$ et entiers




C) Complément de modélisation du problème de sélection

La contrainte I du problème de sélection que nous avons ignoré jusqu'à présent, à savoir:

**"si un objet est sélectionné dans la classe C, alors au moins n objets ayant même valeur de classe C seront sélectionnés,**

peut s'exprimer grâce à la programmation linéaire en nombres entiers. Si nous considérons le cas général de la contrainte conditionnelle suivante:

$$g(x) > 0 \quad => \quad h(x) >= 0,$$

nous remarquons qu'elle est équivalente à:

$h(x) >= \delta \underline{h}$

et

$g(x) <= (1-\delta) g^{+}$

et

$\delta = 0$ ou 1

avec $g^{+}(x)$ borne supérieure de g(x)
et $\underline{h}$ borne inférieure de h(x)

ce qui donne donc pour la contrainte I du problème de sélection:

$\delta_i = 0$ ou 1

$$\sum_{i=1}^{nfamilles} Cla(C,i,V_j) n_i \leq (1-\delta_j).nobjets$$

$$\sum_{i=1}^{nfamilles} Cla(C,i,V_j) n_i - nb_j \geq -\delta_j.nb_j$$

$$nb_j = \min\left(n, \sum_{i=1}^{nfamilles} Cla(C,i,V_j)*nbof(i)\right)$$

(1)

j = 1,2,..,nbv(C) = nombre de valeurs distinctes pour la classe C.

Remarquons que

$\sum_{i=1}^{nfamilles} Cla(C,i,Vj) n_i$ représente le nombre d'objets sélectionnés dans la

classe C de valeur $V_j$

(1) Cette dernière équation résout le cas où n est supérieur au nombre d'objets présents dans le stock pour la classe donnée.

$\sum_{i=1}^{nfamilles} Cla(C,i,Vj)nbof(i)$ représente le nombre d'objets dans la classe C de valeur $V_j$

$\delta_i = 0$ si au moins un objet est sélectionné dans la classe C de valeur $V_i$

$\delta_i = 1$ sinon

D) Autres exemples de puissance de modélisation en PLE

La programmation linéaire en variables entières offre une puissance de modélisation nettement supérieure à la programmation linéaire. D'autres types de contraintes peuvent encore s'exprimer grâce à la PLE.
On peut par exemple rendre compte des volontés suivantes:

a) Dichotomies.

$$\min_{x \in X} f(x) \qquad X \subset \Re$$

s.c. $g_1 >= 0$ ou $g_2 >= 0$ ou les deux.

Equivalent à:

$g_1(x) >= \delta g_1$
et
$g2(x) >= \delta g2$
et
$\delta = 0$ ou 1

avec $g_i(x)$ la borne inférieure de $g_i(x)$

b) Contraintes mutuellement exclusives

$$\min_{x \in X} \quad f(x) \qquad X \subset \Re$$

$$\text{s.c. } g_1 >= 0 \text{ ou } g_2 >= 0 \text{ ou } \ldots g_m >= 0$$

Equivalent à:

$$g_i(x) >= (1 - \delta_i) g_i$$

et

$$\sum_{i=1}^{m} \delta_i = 1$$

et

$$\delta_i = 0 \text{ ou } 1$$

avec $g_i(x)$ la borne inférieure de $g_i(x)$

Les propriétés ci-dessus (ainsi que celle relative aux contraintes conditionnelles) sont justifiées dans [Fichefet 82].

E) Techniques de résolution

Le simplexe ne permet pas de résoudre le PLE vu qu'il fournit comme solution qu'un sommet du convexe des contraintes, ce qui ne garantit pas une solution entière.
Nous allons présenter trois techniques permettant de résoudre des problèmes linéaires entiers ou mixtes. Nous nous attarderons plus longuement sur la dernière de ces méthodes qui a été utilisée pour traiter le problème de sélection.

a) énumération

L'idée utilisée est certainement la plus simple pour résoudre le problème. Elle consiste à parcourir la liste de toutes les solutions entières possibles

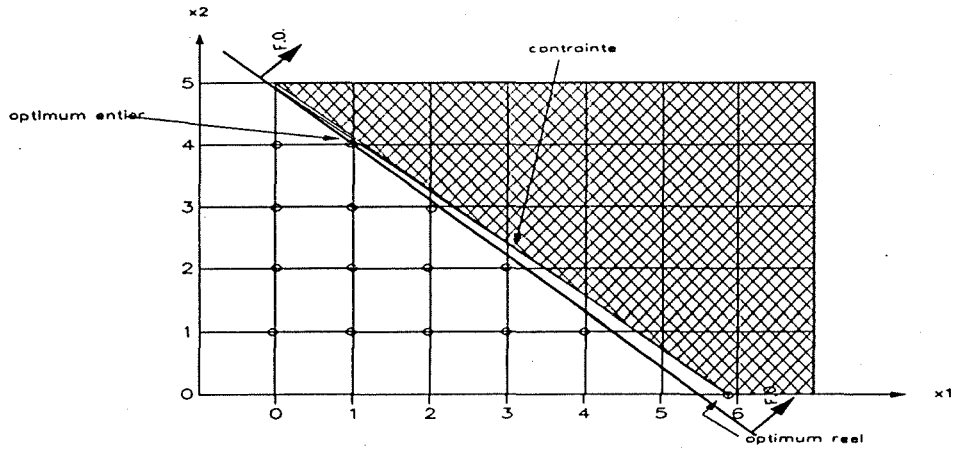
et à retenir, parmi celles qui vérifient les contraintes, celle qui donne la valeur extrémale pour la fonction objectif. Cette technique ne peut être utilisée que pour des problèmes dont l'espace des solutions est très petit. En effet, si l'on se limite à n variables booléennes, le nombre de solutions à envisager est de $2^n$ ce qui demanderait des siècles sur l'ordinateur le plus puissant et ce, déjà pour des valeurs de n inférieures à 70.

b) Les coupes

Nous nous limiterons à l'idée générale:

> si la solution continue du problème n'est pas entière, on ajoute une contrainte de façon à éliminer l'optimum réel sans supprimer de solutions entières. Une telle contrainte est appelée coupe et théoriquement, il en existe toujours une infinité. Après avoir ajouté une coupe, on recherche la solution continue du nouveau problème. On itère ainsi jusqu'à obtenir une solution entière. La principale difficulté réside bien évidemment dans le choix des coupes utilisées. Gomory en 1958 a mis en évidence des coupes d'un type particulier qui permettent d'assurer la convergence de la méthode.

Exemple de coupe: $x_1 <= 5$ pour le problème ci-dessous.




Cette méthode ne peut pratiquement s'appliquer qu'à des contraintes dont les coefficients des variables sont entiers.

### c) Le Branch and bound

L'algorithme de Branch and Bound est la méthode la plus courante utilisée pour résoudre les problèmes linéaires entiers (si toutes les variables utilisées sont entières) ou mixtes (si certaines variables peuvent être réelles). La plupart des codes commerciaux sont basés sur cette approche. Fondamentalement cet algorithme n'est rien d'autre qu'une énumération "intelligente" des solutions possibles.

Nous considérerons dans la suite que le problème d'optimisation est un problème de maximisation.

La première étape consiste à résoudre le problème linéaire mixte, noté PLE, en ignorant les restrictions imposant aux variables d'être entières, soit PL-1 ce problème et Z1 la fonction objectif à l'optimum.

Si toutes les variables de la solution optimale de PL-1 qui doivent être entières, le sont, le problème initial est bien évidemment résolu.

Sinon, nous n'avons pas la solution optimale du problème linéaire mixte mais Z1 est une borne supérieure de Z, la valeur maximale du problème PLE. En effet, l'ajout de contraintes ne peut que restreindre l'ensemble des solutions admissibles et faire décroître la fonction objectif.

L'étape suivante consiste à remplacer le problème initial par un ensemble équivalent de deux problèmes linéaires en découpant en deux, l'espace admissible de PL-1. Pour cela on choisit une variable contrainte à être entière. Nous reviendrons plus tard sur les différents choix possibles et leurs conséquences. Supposons $x_j$ la variable sélectionnée, $\beta_j$ sa valeur fractionnelle dans PL-1 et int(r) la partie entière du réel r (int(4.7)=4).
Les deux nouveaux problèmes PL-2 et PL-3 sont obtenus en introduisant dans PL-1 respectivement la contrainte $x_j \leq int(\beta_j)$ et $x_j > int(\beta_j)$.

Mathématiquement:

| PL-2 | PL-3 |
|---|---|
| Maximum $Z_2$=cx<br>sc:<br>Ax=b<br>$x_j \leq int(\beta_j)$<br>$x_j \geq 0 \quad \forall j$ | Maximum $Z_3$=cx<br>sc:<br>Ax=b<br>$x_j > int(\beta_j)$<br>$x_j \geq 0 \quad \forall j$ |

Une fois les nouvelles contraintes ajoutées, on utilise l'algorithme du simplexe pour déterminer les nouvelles solutions optimales de PL-2 et PL-3.

Supposons que ces solutions optimales soient toujours fractionnaires et donc non admissibles pour le problème PLE.
La prochaine étape sera de choisir un des sous-problèmes dans la liste des sous-problèmes non encore "découpés", ici: [ PL-2, PL-3 ], et de réappliquer la démarche comme on l'a fait pour PL-1 à partir du sous-problème choisi. On obtient deux nouveaux sous-problèmes PL-4 et PL-5 auxquels on applique à nouveau l'algorithme du simplexe.

Comme nous l'avons déjà signalé précédemment l'optimum réel obtenu par le simplexe est une borne supérieure pour la solution entière d'un problème linéaire mixte donné, d'autre part la solution optimale est obligatoirement à l'intérieur d'un des sous-problèmes issus du découpage. D'où, si toutes les contraintes entières sont respectées pour une des solutions optimales d'un sous-problème et, si la valeur Z de la fonction objectif de cette solution est supérieure à celle des PL- non encore "découpés", on peut arrêter le processus; la solution entière obtenue est une solution optimale du problème de départ: PLE.

D'autre part, si la valeur de la fonction objectif n'est pas supérieure à celle des PL- non encore "découpés", la solution entière obtenue constitue une borne inférieure pour la solution de PLE. On ne doit plus chercher que des solutions améliorant cette solution entière.

Ceci permet, dès lors, de ne plus considérer les sous-problèmes dont l'optimum réel est inférieur à la valeur Z obtenue pour la solution entière. Considérons, par exemple l'arborescence de la fig. 4, la solution de PL-4 constitue une borne inférieure pour Z la solution de PLE. Le découpage du noeud 4 n'est plus nécessaire, la solution optimale étant entière. Le noeud 5 n'admet pas de solution réelle, donc certainement pas de solution entière, il ne doit plus être décomposé. On choisit de décomposer LP-3 parce que Z3 est supérieure à Z4 et que Z6 ou Z7 pourraient améliorer Z4 et être entières. Mais si Z6 et Z7 sont inférieures à Z4 on est alors certain que Z4 est la solution optimale.

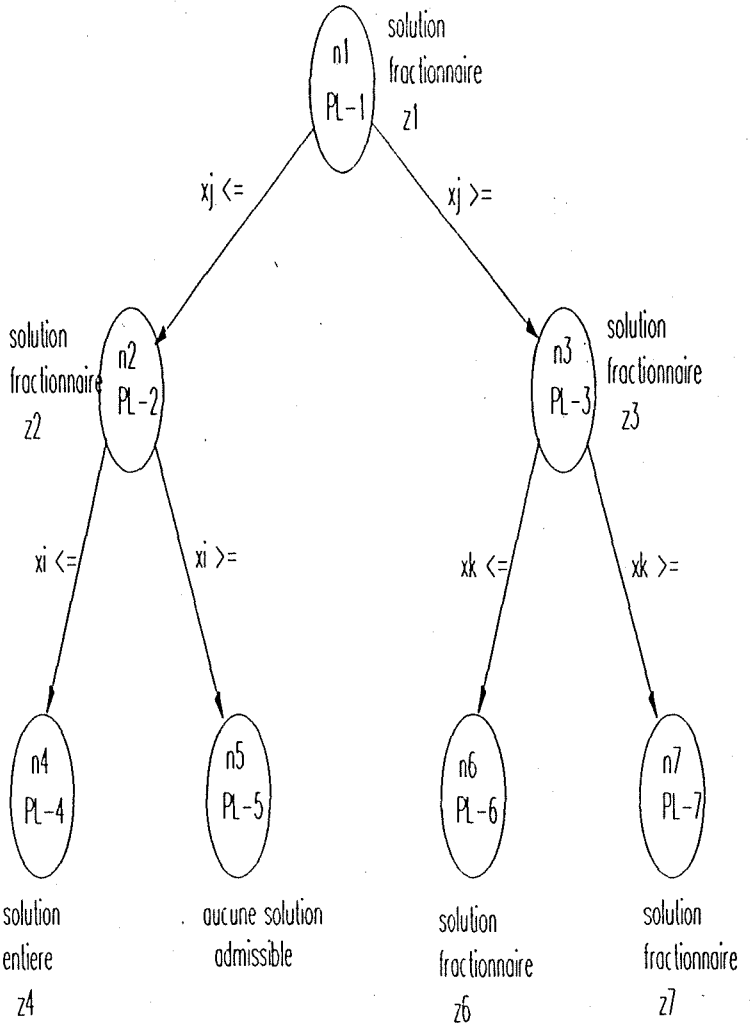



fig. 4

Pour ne pas compliquer d'avantage l'exposé des principes du branch and bound en PLE, nous avons volontairement ignoré jusqu'ici deux points fondamentaux: comment s'opère le choix de la variable qui permet la décomposition et comment détermine-t-on le prochain problème à décomposer ?

Nous allons présenter chacun de ces choix et décrire leurs implications.

- Choix de la variable de branchement $x_i$

Ce choix peut être déterminant: on trouve d'ailleurs dans [Minoux 88] cette affirmation qui le souligne: "... suivant l'ordre adopté, le temps de résolution peut varier de 1 à 10 ou de 1 à 100... "

*La première

La liste des variables entières est dans ce cas parcourue séquentiellement, la première de ces variables à ne pas être entière est sélectionnée.
C' est certainement la technique la plus simple mais pas nécessairement la plus efficace.

*La meilleure

L'idée est ici de réaliser une décomposition en deux problèmes dont l'un des deux a la valeur de la fonction objectif la plus proche possible de celle de départ. On espère trouver une solution sans avoir trop dégradé la fonction objectif et ainsi, en utilisant "la propriété de la borne inférieure", accélérer la recherche de la solution optimale.

*La moins entière

On choisit la variable dont l'écart au nombre entier le plus proche est maximum.

*La plus prioritaire

Dans certains problèmes le décideur peut vouloir imposer l'ordre dans lequel vont être fixées les variables. A titre d'exemple, on

voudrait fixer les delta du critère I avant les autres variables pour éviter de développer un grand nombre de sous-problèmes (ou noeuds) menant à une impasse.

*Autres

Les choix présentés ici ne sont évidemment pas les seuls possibles, on trouvera dans [Fichefet 82] une liste de choix et des explications plus complètes ainsi que de nombreuses références.

- Choix d'un noeud

Remarquons d'abord que la méthode utilisée pour choisir le prochain noeud à décomposer détermine le parcours d'arbre utilisé pour la recherche. Les principaux types de choix sont : lifo et priorité.

*LIFO (Last In First OUT)

On décompose le PL qui a été ajouté en dernier dans la liste. Ce qui revient à parcourir l'arbre en profondeur d'abord et donc de fournir (généralement) assez rapidement une première solution entière qui peut encore être très éloignée de la solution optimale.

*Priorité

On choisit dans la liste le PL dont la fonction objectif est la meilleure. On obtient un parcours proche de la largeur d'abord. La première solution entière obtenue est généralement bonne. En la comparant à celle du prochain noeud développé, on peut mesurer la distance maximale qui la sépare de l'optimum.

### 3.2.3 Optimisation multicritère

La progression de l'analyse nous amène à envisager maintenant la concurrence des différents objectifs:

**E) Choisir les objets de manière à minimiser la somme des mesures M.**

$$minimiser \quad \sum_{i=1}^{nfamilles} mesure(M,i).n_i$$

**F) Eviter la sélection des objets dont la propriété P à la valeur V.**

$$minimiser \quad \sum_{i=1}^{nfamilles} prop(P,i,V)n_i$$

On les traitera en se limitant aux problèmes linéaires, même si certains éléments seraient encore vérifiés dans le contexte non linéaire.

A) Evaluation d'une solution

a) Solution optimale

Définition:

Une solution optimale, au sens classique est une solution qui atteint la valeur maximale pour tous les objectifs simultanément.

Nous avons déjà vu que les problèmes multicritères, de par leur définition, mettaient en présence des objectifs contradictoires.
Le concept de solution optimale n'existe plus et doit être remplacé par celui de meilleur compromis entre les objectifs.
Un compromis correspond à une solution efficace.

b) Solution efficace

Définition:

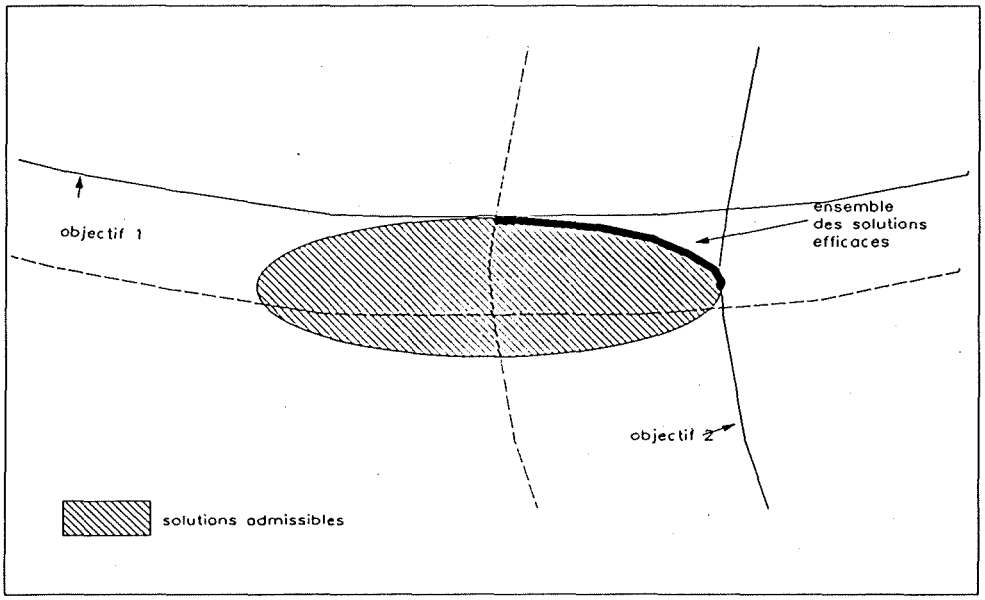
Une solution efficace (aussi appelée solution optimale au sens de Pareto) est une solution telle qu'aucun accroissement ne peut être obtenu sur un quelconque des objectifs, sans causer simultanément une diminution sur, au moins, un autre objectif.

Mathématiquement:

Soit $f_i$ la fonction exprimant le $^{ième}$ objectif à maximiser.
$x^*$ est solution efficace de l'ensemble des solutions admissibles S ssi:

$$\neg \exists (x \in S, i) : \forall i \, f_i(x) \geq f_i(x^*) \wedge f_i(x) > f_i(x^*)$$

Graphiquement:



Cette définition correspond bien à l'idée qu'un compromis ne peut plus être amélioré sur une composante sans perdre nécessairement sur une autre.
L'ensemble des solutions efficaces est souvent appelé "frontière efficace".

B) Quelques méthodes possibles.

Les techniques auxquelles on pense plus ou moins naturellement et que nous allons exposer sont basées sur la théorie de l'utilité multiattribut. De nombreux ouvrages sont consacrés à cette théorie et le livre de KEEYNEY et RAIFFA [Keeyney & Raiffa 76] est généralement considéré comme la référence en la matière.

L'idée fondamentale de cette théorie est que tout décideur essaie inconsciemment de maximiser une fonction:

$$U = U(g1, g2, ..., gn)$$

résultant de l'agrégation des différents objectifs pris en compte.

Nous allons montrer comment est construite la fonction U dans quelques méthodes non-interactives et nous verrons ensuite comment cette fonction évolue au cours de son utilisation dans les méthodes interactives.

a) Méthodes non interactives

- Combinaison linéaire des objectifs

On demande au décideur de fournir un vecteur $W = (w_1, w_2, w_3, .., w_n)$ dont chacune des composantes représente l'importance relative de l'objectif correspondant.
On recherche la solution optimale de:

$$\min_{x \in C} \sum_{i=1}^{n} w_i f^i(x)$$

où C est l'espace des solutions admissibles et $f^i$ le $i^{ème}$ objectif.

Même si, mathématiquement, cette formulation paraît tout à fait satisfaisante, en pratique elle pose plus de problèmes qu'elle n'en résout, surtout pour le décideur qui doit fournir le vecteur W.
Il s'avère en effet, dans la plupart des cas, que le décideur n'est capable de fournir ce vecteur que pour le seul problème donné.

- Enumération des solutions efficaces

Nous avons déjà dit que le problème multicritère revenait à choisir le meilleur compromis. On sait aussi que ce compromis doit faire partie de l'ensemble des solutions efficaces.
Des auteurs comme Yu et Zeleny [Yu & Zeleny 75] ont proposé une méthode de résolution basée sur l'idée suivante:

- déterminer l'ensemble de toutes les solutions extrémales efficaces (c.-à-d. non combinaison linéaire d'autres solutions efficaces). Cet ensemble est de dimension finie puisque l'espace des solutions admissibles est un polyèdre qui possède un nombre fini de faces.

- et ensuite, engendrer toutes les solutions efficaces par combinaisons linéaires des solutions extrémales.

Comme on obtient toutes les solutions efficaces, on aura certainement celle qui satisfera le plus le décideur. Malheureusement cette méthode fournit généralement trop de solutions et ne permet pas de guider le décideur vers la solution de compromis qu'il recherche.

- Goal programming

Cette méthode introduite par Charnes et Cooper [Charnes & Cooper 60], tente de trouver un compromis (une solution optimale qui soit aussi proche que possible des objectifs), en respectant l'importance relative des différents objectifs. Pour cela, en goal programming, on assigne à

chaque objectif un niveau à atteindre et une priorité pour atteindre ce niveau.

Dans cette section, nous allons analyser comment formuler un modèle de goal programming.

Avant d'entrer dans la méthode de formulation, il nous faut préciser la différence entre les termes *contraintes réelles* et *contraintes objectifs* (ou buts) utilisés en goal programming.

Les contraintes réelles sont des restrictions absolues sur les variables de décision, alors que les buts sont des conditions qu'il faut satisfaire au mieux, sans obligation. Par exemple une contrainte réelle est donnée par:

$$x_1 + x_2 = 3$$

qui impose que toutes les valeurs possibles de $x_1 + x_2$ soient égales à 3.

Contrairement au but:

$$x_1 + x_2 = 3$$

qui acceptera des valeurs de $x_1 + x_2$ aussi bien supérieures qu' inférieures à 3.

Dans les contraintes objectifs, les déviations positives et négatives sur les objectifs sont introduites de la façon suivante:

$$x_1 + x_2 + d^+ - d^- = 3$$

$$d^+, d^- \geq 0$$

On remarque sans peine que si $d^- > 0$ alors $x_1 + x_2 > 3$ et si $d^+ > 0$ alors $x_1 + x_2 < 3$

En assignant les poids $w^+$ et $w^-$ sur $d^+$ et $d^-$ dans la fonction objectif du modèle, la somme $x_1 + x_2$ va tendre aussi bien que possible vers 3. Si le but était de satisfaire $x_1 + x_2 >= 3$ alors $d^-$ est soumis à un poids positif dans la fonction objectif et $d^+$ est mis à zéro.

Le modèle général du goal programming peut être exprimé de la façon suivante:

$$\text{minimiser} \qquad Z = \sum_{i=1}^{m} (w_i^+ d_i^+ + w_i^- d_i^-) \qquad (1)$$

soumis aux contraintes:

$$\sum_{j} a_{ij} + d_i^+ - d_i^- = b_i \qquad \forall i \qquad (2)$$

$$x_j, d_i^+, d_i^- \geq 0 \qquad \forall i, j \qquad (3)$$

l'équation (1) représente la fonction objectif qui minimise la somme des variables d'écarts. Le système d'équations (2) renferme les buts à atteindre en se rapprochant du niveau fixé.

L'ensemble des inégalités (3) donne les restrictions standard sur toutes les variables. Si les poids relatifs ($w^+$ et $w^-$) peuvent être spécifiés par les décideurs, alors le goal programming se ramène à un seul problème monocritère. Malheureusement, il est très difficile et, dans beaucoup de cas, impossible de fournir l'approximation de poids qui conduira au compromis désiré.

Pour tenter de résoudre ce problème de pondération, de nombreux auteurs ont proposé des méthodes interactives.

b) Méthodes interactives

- STEM

Cette méthode fut proposée par Benayoun, de Montgolfier, Tergny et Larichev [Benayoun 70] au début des années 70 et a été améliorée notamment par Ph. Vincke [Vincke 76]. Le décideur interagit dans le processus en répondant aux questions que lui pose l'algorithme. Ces questions permettent de guider l'exploration des solutions efficaces.

L'algorithme est constitué d'une succession de phases de calcul et de prise de décision.

Le but de la phase de décision est de déterminer avec le décideur et sur base de la solution actuelle non satisfaisante, un objectif susceptible d'être dégradé. La phase de calcul recherche alors une nouvelle solution dans laquelle on espère que l'effort consenti sur l'objectif choisi permette d'améliorer les autres objectifs et ainsi de les rendre acceptables pour le décideur.

Détails de la méthode:

ETAPE 0:

déterminer la matrice de gains (payoff matrix).

| | $x_1^*$ | $x_2^*$ | $x_n^*$ |
|---|---|---|---|
| obj n°1 | $f_1(x^{1*})$ | $f_1(x^{2*})$ | $f_1(x^{n*})$ |
| obj n°2 | $f_2(x^{1*})$ | $f_2(x^{2*})$ | $f_2(x^{n*})$ |
| | ... | ... | ... |
| | ... | ... | ... |
| | ... | ... | ... |
| obj n°i | $f_i(x^{1*})$ | $f_i(x^{2*})$ | $f_i(x^{n*})$ |
| | ... | ... | ... |
| | ... | ... | ... |
| obj n°k | $f_k(x^{1*})$ | $f_k(x^{2*})$ | $f_k(x^{n*})$ |

où les $f_i$ sont les fonctions exprimant les différents objectifs à maximiser et $x^{i*}$ l'optimum pour la fonction $f_i$ considérée isolément.
La solution $y$ serait une solution optimale comme définie en a)
$\Leftrightarrow \quad f_i(y) = f_i(x_i^*) \qquad \forall i$

ETAPE 1 : CALCUL

$$\min z = y$$
$$\text{sc:}$$
$$y \geq (f_l(x^{l*}) - f_l(x))\pi_l \quad l = 1, \ldots, k$$
$$x \in X^m$$
$$y \geq 0$$

$X^m$ est l'espace défini par les contraintes du départ auxquelles s'ajoutent les contraintes des m-1 cycles précédents.
Les $\pi_l$ fournissent l'importance relative des distances vis-à-vis de l'optimum. Ils sont obtenus par les formules suivantes:

$$\pi_l = \frac{\alpha_l}{\sum_{i=1}^{k} \alpha_i} \qquad l = 1, \ldots, k$$

$$\alpha l = \left| \frac{M_l - m_l}{M_l} \right| \cdot \frac{1}{\sqrt{\sum_{j=1}^{n} c_{lj}^2}} \qquad l = 1, \ldots, k$$

où $M_l$ $m_l$ sont respectivement le maximum et le minimum des valeurs de la $l^{ème}$ ligne de la matrice des gains et le terme $c_{lj}$ donne le coefficient de la $j^{ème}$ variable de décision dans la $l^{ème}$ fonction objectif.

ETAPE 2: DECISION

On demande au décideur de comparer les composantes $f_i(x^m)$ (résultat de la $m^{ème}$ phase d'optimisation) avec la solution optimale ($f_i(x^{i*})$. Si certains objectifs ne sont pas satisfaits, le décideur fournit $\delta_j$ égal à ce qu'il consent à "perdre" sur un objectif j.
$X^{m+1}$ est obtenu en ajoutant à $X^m$ la contrainte suivante:

$$f_j \geq f_j(x^m) - \delta j$$

On recommence ensuite l'optimisation de la phase 1 après avoir mis $\pi_j$ à 0 et incrémenté l'indice du cycle.

La mise à la disposition du décideur de la matrice des gains, la construction progressive et interactive du compromis et l'absence de demande d'informations a priori sur l'importance relative des objectifs, constituent les points forts de cette méthode.
L'interdiction de relaxer plusieurs fois le même objectif(1), ce qui empêche le décideur d'introduire des seuils de satisfaction différents,

(1) Cette interdiction peut être levée si l'on consent à perdre la garantie de la convergence

assure la convergence en k itérations. Les auteurs proposent pour résoudre le problème du choix de l'objectif à relâcher une technique inspirée de l'analyse de sensibilité.

- GPSTEM

Cette méthode, due à J. Fichefet, en mariant le goal programming et la méthode STEM permet au décideur de définir des seuils de satisfaction et le guide à les adapter jusqu'à l'obtention d'une solution satisfaisante.
Nous ne donnerons ici que l'algorithme de la méthode (extrait du mémoire de J-L Borlon), nous remarquerons que le mécanisme de calcul des pondérations dispense le décideur de fournir explicitement ses poids.

Le lecteur désireux d'approfondir pourra trouver tous les détails de cette méthode ainsi que ses fondements théoriques dans [Fichefet 76] et [Borlon 77]

Algorithme:

#1 pour chaque i = 1, ..., k calculer $\hat{x}_i$ solution optimale de $\min_{x \in C} f^i(x)$

et construire la matrice des gains tq $P_i^j = f^j(\hat{x}_i)$

#2 calculer:

$$\alpha^i = \left| \frac{P_i^i}{P_i^i - m^i} \right| \cdot \sqrt{\sum_{j=1}^{n} (c_j^i)^2} \qquad i = 1, \ldots, k$$

avec $m^i$ = la plus grande valeur dans la ligne i de P

#3 demander les niveaux de satisfaction $M^i$ i=1,...,k

#4 k := 0; calculer $x_k$ solution optimale de

$$\min \; Z = \sum_{i=1}^{r} (y^{+i} + y^{-i})$$

sc:

$$f^i(x) + \alpha^i y^{+i} - \alpha^i y^{-i} = M^i$$
$$x \in C \qquad y^{+i} \geq 0 \quad y^{-i} \geq 0 \quad i = 1, \ldots, k$$

#5 tester si $y^{+i} = 0 \qquad \forall i = 1, \ldots k$

oui: passer à l'étape suivante
non: fournir $x_l$; stop

#6 donner les valeurs prises par les fonctions économiques

elles sont satisfaisantes : fournir $x_l$ ; stop
sinon : passer à l'étape suivante

#7 tester si $y^{-i} = 0 \qquad \forall i = 1, \ldots k$

oui : passer à l'étape suivante
non : le décideur désire une modification $R^l$ de son objectif:

$$M^l = M^l + R^l \; ; \quad l = l + 1$$

soit $x_l$ la solution après cette modification du second membre;
aller en #5

#8 prendre un vecteur de perturbation des niveaux de satisfaction:
$\Delta M$ tq $\Delta M^i \geq 0$, i=1,...,k

résoudre le problème paramétrique:

$$\min \quad Z = \sum_{i=1}^{r} (y^{+i} + y^{-i})$$

sc:

$$f^i(x) + \alpha^i y^{+i} - \alpha^i y^{-i} = M^i - \theta . \Delta . M^i$$

$$x \in C \qquad y^{+i} \geq 0 \quad y^{-i} \geq 0 \quad i = 1, \ldots, r$$

qui garde z=0

si Θ=0, il n'y a pas d'amélioration; retourner en #7, deuxième partie.
sinon $M^i := M^i - \theta . \Delta M^i$ $\Delta$ i= 1,...,k
soit $x_1$ la solution après la paramétrisation; aller en #6

- Autres

Beaucoup d'autres méthodes ont été proposées dans la littérature. On pourra trouver notamment dans [Vincke 88] une présentation d'une dizaine de ces méthodes.

c) Méthode et modèle retenus

Le modèle de programmation linéaire présenté au point 3.2.1.F en introduisant les écarts permettra la "satisfaction au mieux" des critères pour prévoir les cas où les contraintes sont incompatibles. La programmation linéaire entière apporte une solution au cas de la contrainte I. Si on y introduit des écarts comme pour les autres contraintes on obtient:

**I) Si un objet est sélectionné dans la classe C alors au moins n objets ayant même valeur de classe C seront sélectionnés.**

$\delta_i = 0$ ou 1

$$\sum_{i=1}^{nfamilles} Cla(C,i,Vj)n_i \leq (1-\delta_j).nobjets + EI_{(1j)}$$

$$\sum_{i=1}^{nfamilles} Cla(C,i,Vj)n_i - nb_j + EI_{(2j)} \geq -\delta_j.nb_j$$

$$nb_j = \min\left(n, \sum_{i=1}^{nfamilles} Cla(C,i,Vj)*nbof(i)\right)$$

j = 1,2,..,nbv(C) = nombre de valeurs distinctes pour la classe C.

Pour résoudre le problème posé par les différents objectifs nous avons retenu la technique du Goal Programming. On obtient alors le modèle suivant:

- Tous les critères à respecter absolument sont inchangés

- Tous les critères à respecter au mieux excepté E) et F) sont inchangés.

- Chaque occurrence du critère E) va introduire dans la fonction objectif l'écart entre la valeur de l'objectif E et sa valeur idéale. Comme cette dernière valeur est 0, on introduit directement l'objectif E dans la fonction objectif finale.

- Chaque occurrence du critère F) va introduire dans la fonction objectif l'écart entre la valeur de l'objectif F et sa valeur idéale. Comme cette dernière valeur est 0, on introduit directement l'objectif F dans la fonction objectif finale.

- Toutes les variables d'écarts introduites viennent s'ajouter à la fonction objectif finale.

La fonction objectif du problème linéaire (la fonction à minimiser) sera donc:

$$\sum_{critères\ E} \sum_{i=1}^{nfamilles} mesure(M,i)ni + \sum_{critères\ F} \sum_{i=1}^{nfamilles} prop(P,i,V)n_i + \sum_{(X,y,z)} EXyz$$

De manière à permettre à l'utilisateur de préciser l'importance qu'il accorde aux différents critères, on introduit des facteurs de pondération intervenant en coefficients des termes de la fonction objectif correspondant aux critères E,F..K. Différents coefficients de pondération peuvent être précisés pour différentes occurrences d'un même critère. Nous les notons $P_E$,$P_F$,...
Ces coefficients (appelés aussi poids, dans la suite) devront être introduits par les décideurs lors de la définition des critères de sélection.

De manière à donner, avant pondération, une importance relative identique à tous les termes de la fonction objectif (et en particulier pour supprimer l'influence des unités de mesure), on utilisera une forme NORMALISEE de la fonction objectif ( voir par exemple: [Tabucanon 89]).

La fonction objectif s'écrit alors:

$$\sum_{critères\ E} \sum_{i=1}^{nfamilles} \frac{mesure(M,i)\ \ n_i}{\sqrt{\sum_{i=1}^{nfamilles} mesure^2(M,i)}} \cdot P_E$$

$$+ \sum_{critères\ E} \sum_{i=1}^{nfamilles} \frac{prop(P,i,V)\ \ n_i}{\sqrt{\sum_{i=1}^{nfamilles} prop^2(P,i,V)}} \cdot P_F$$

$$+ \sum_{(X,y,z)} \frac{EXyz}{FXyz} \cdot P_X$$

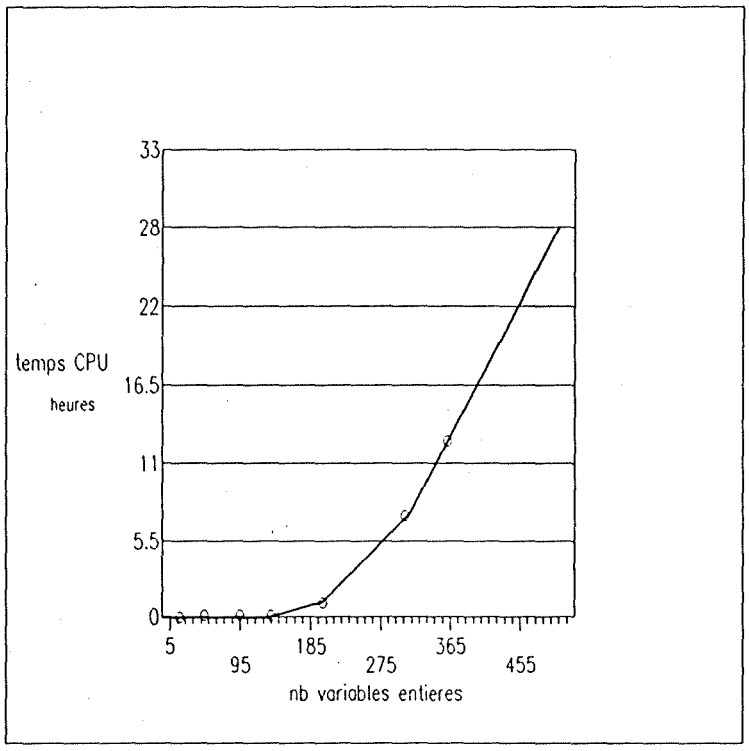
Le calcul du facteur de normalisation FXyz s'effectue de la manière suivante:
pour l'inéquation : a1 n1+ a2 n2 + EXyz <= B

$$FXyz = \sqrt{a_1^2 + a_2^2 + B^2}$$

## 3.3 Expérimentation

### 3.3.1 Premier prototype

La routine réalisée s'insère dans la structure vue à la section 3.1.1 et permet de générer le modèle de goal programming qui est résolu ensuite par une routine de programmation linéaire mixte.
Cette routine qui fait partie de la librairie NAG, utilise la technique du branch and bound décrite à la section 3.2.2.E.C avec le choix de la variable de branchement vu en *La première de la même section et avec le parcours d'arbre vu en *Priorité toujours de la même section.
On trouvera dans l'annexe 3 quelques détails sur la routine utilisée.
Les variables de décision les ($n_i$) sont considérées comme entières dans l'intervalle [1..nobf(i)].
Lors de la phase de développement nous avons effectué quelques tests, avec des versions en évolution et des données parfois aléatoires, qui permettent d'établir le graphique suivant:




remarque: la machine utilisée est un Vax 8530 d'une puissance de 3,5 VUPS = +- 4 MIPS.

La figure ci-dessus exprime bien l'impression d'impuissance que l'on ressent en résolution de PLE, même pour des nombres de variables qui semblent petits.

Une fois l'outil opérationnel, nous avons réalisé les premiers tests sur base de données réelles constituées à partir d'un stock de ± 400 coils de ADU (c.-à-d. 400 variables binaires(1)). Les résultats obtenus peuvent se résumer comme suit:

-Le temps nécessaire pour obtenir la solution entière optimale n'a pu être déterminé(2). Nous pouvons résumer nos tentatives par l'anecdote suivante : l'examen des résultats du batch lancé un vendredi vers 18H00 permet de constater

(1) On considère le cas particulier où il n'y a qu'un élément par famille voir aussi le dernier paragraphe de cette section.

(2) Le graphique n'en donne qu'une approximation.

que le job a pris fin après 48H00 d'exécution non pas après avoir trouvé la solution, mais après avoir rempli les 60 méga disponibles sur le disque. Le processus n'écrivait qu'une ligne pour chaque nouveau noeud de l'arbre de recherche.

-La mise en pratique d'un conseil de l'équipe de recherche en programmation linéaire d'IBM, indiquant que "Si la situation ne l'impose pas, on peut considérablement réduire le temps de calcul en se contentant de la première solution entière quand celle-ci n'est plus qu'à 3% de la solution optimale réelle" [Ravindran 87], a permis d'établir que le temps nécessaire pour obtenir une (1) solution entière se situe dans une fourchette de 5-10 minutes.
Les variations du temps de calcul provoquées par de légers changements des paramètres des critères sont totalement imprévisibles.

Nous avons également vérifié si le fait de considérer des familles d'objets, plutôt que les objets eux-mêmes(2), se traduisait par un gain de performances. C'est en effet le cas, le temps pour obtenir la première solution entière pour un problème avec 112 familles (c.-à-d. 112 variables comprises entre 1 et 3(3)) est trois fois plus petit que pour le même problème avec 333 objets (c.-à-d. 333 variables binaires).

(1) Il s'agit ici de la toute première solution entière qui peut être encore distante de plus de 3% de l'optimale réelle.

(2) Ce qui revient à utiliser pour un ensemble de coils indifférenciables, une variable entière $[0..n_i]$ indiquant le nombre de coils identiques sélectionnés, à la place d'une variable binaire [0..1] indiquant pour chaque coil si on le sélectionne ou non.

(3) En moyenne.

### 3.3.2 Deuxième prototype

Comme nous venons de le voir, avec le modèle précédent où chaque $n_i$ est déclaré variable entière, nous ne pourrons pas garantir au décideur un temps de réponse inférieur à dix minutes.
De plus, on a constaté dans les tests précédents la difficulté de fixer les poids du premier coup.

Il est intéressant cependant de noter que plus de 90% des variables de décision prennent spontanément des valeurs entières et ce déjà dans la solution réelle, résultat du premier simplexe.

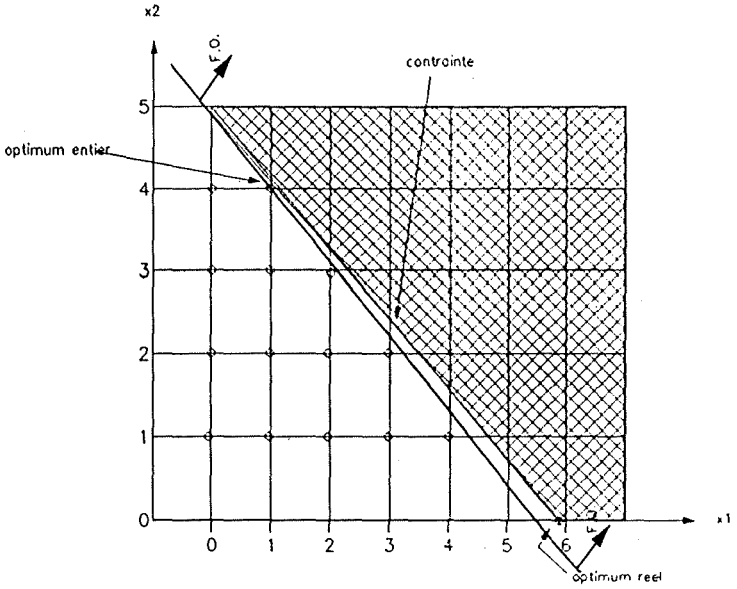
On penserait dès lors à arrêter l'optimisation à la solution réelle. Malheureusement, dans ce cas, les contraintes de type I ne seront plus vérifiées. On peut, en effet, montrer que les équations engendrées pour modéliser cette contrainte n'ont *aucune* efficacité, si les variables auxiliaires introduites ne sont pas strictement booléennes.

La seconde implémentation supprime l'obligation pour les variables $n_i$ d'être entières pour l'optimisation. Elles seront ensuite arrondies. Seules les variables Delta de la contrainte I subsistent en tant que variables entières dans le modèle.

Remarquons que les contre-exemples à cette technique des arrondis ne manquent pas, dans d'autres contextes, on trouve, notamment, dans Minoux[83] l'exemple suivant:

$$\text{Min } z = -10\,x_1 - 11\,x_2$$
$$\text{avec : } 10\,x_1 + 12\,x_2 <= 59$$
$$x_1 \text{ et } x_2 \text{ positifs et entiers.}$$

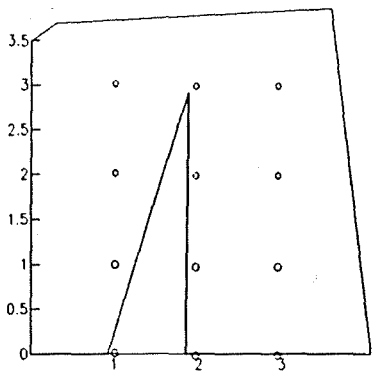
que l'on peut représenter dans le plan par la figure ci-dessous.




L'optimum réel est le point de coordonnées $x_1 = 5.9$ et $x_2 = 0$ pour lequel $z = -59$.
Une simple méthode d'arrondi conduirait à la solution $x_1 = 6$ et $x_2 = 0$ qui ne satisfait pas les contraintes.
L'optimum entier est donné par le point de coordonnées: $x_1 = 1$ et $x_2 = 4$

On imagine sans peine d'autres exemples où

l'espace des contraintes délimite un cône




ce qui permet de se convaincre que l'on peut construire des exemples pour lesquels l'optimum entier est aussi éloigné que l'on veut de la solution optimale réelle.

Même si théoriquement il est possible de construire autant de contre-exemples que l'on veut pour prouver la non-optimalité d'une procédure utilisant les arrondis, on peut penser que pour notre implémentation dans le contexte du goal programming, la procédure des arrondis ne sera pas gênante puisqu'au pire elle ne pourra que dégrader légèrement la satisfaction des contraintes. Ce qui s'avère être tout-à-fait le cas en pratique.

nb: Les chercheurs du LAMSADE (Université Paris Dauphine), consultés à ce sujet, avaient déjà rencontré d'autres cas pratiques où cette façon d'agir s'était révélée efficace.

### 3.3.3 Choix judicieux d'une mesure de délai

Les utilisateurs d'ADU désiraient choisir les coils les plus urgents, c'est ce que permet le critère E qui, appliqué à la mesure du délai, donne:

**Choisir les objets de manière à minimiser la somme des délais**

ce qui revient, comme nous l'avons vu, à minimiser la fonction suivante:

$$\sum_{i=1}^{nfamilles} delai_i . n_i$$

Il fallait donc définir avec eux une échelle pour mesurer ce délai.
La solution la plus simple techniquement, s'imposait: à savoir la date de fin de fabrication exprimée par une suite de cinq chiffres, les deux premiers indiquant l'année, les trois autres précisant le jour dans cette année (aajjj).
On ne travaille alors qu'avec des valeurs positives.

On aurait pu donner aux coils en retard des valeurs négatives, en donnant à la date du jour la valeur zéro.

Des essais comparatifs de ces possibilités ont mis en évidence un comportement surprenant bien que très facilement explicable une fois qu'on en a pris conscience.

A) Délais positifs

Examinons le cas où toutes les valeurs du délai sont positives du type aajjj. On constate que toutes ces valeurs sont du même ordre de grandeur (comprises entre 88000 et 91000). Comme il faut prendre un certain nombre d'objets pour atteindre le tonnage désiré, le modèle va préférer un objet fort lourd (n°1) à deux objets de faible délai (n° 2 et 3).

soit d1,d2,d3 les délais
et p1,p2,p3 les poids des objets n° 1,2,3

| | |
|---|---|
| d1=91000 | d2+d3 = 2*88000 = 176000 |
| p1=45 t | p2+p3 = 45 t |

vu l'objectif: somme des di minimum !

Les différences de masse entre coils ne sont pas aussi importantes en réalité mais le nombre d'objets sélectionnés est plus grand. On obtiendra donc une sélection comprenant un petit nombre d'objets lourds alors qu'une sélection constituée de plus d'objets (forcement plus légers) avec des délais individuels meilleurs, aurait pu être trouvée.

B) Délais positifs et négatifs

Si on peut atteindre la quantité souhaitée en tonnage, uniquement avec des objets de délai négatif, les objets sélectionnés sont alors les plus légers, puisque, dans ce cas, on ne prend évidemment que des négatifs et le plus nombre possible. Si on doit prendre des positifs pour atteindre le tonnage voulu, la sélection de ces positifs souffre du défaut du point A) .
Globalement c'est toutefois la moins mauvaise des deux solutions, car elle prend au moins les objets parmi les négatifs (coils en retard de production).

C) Autre solution

Comme on vient de le voir, même la solution du point B) est peu satisfaisante, c'est pourquoi nous avons envisagé une autre technique. Elle est très simple, comme souvent, il suffisait d'y penser. Elle consiste à pondérer la mesure du délai (exprimé sur une échelle positive) par la mesure du tonnage.

## 3.4 Conclusion

### 3.4.1 Aspect scientifique

L'utilisation du goal programming permet de tenir compte des multiples objectifs concurrents et d'éventuelles contraintes incompatibles.
Le modèle issu du goal programming ne peut fournir une solution dans un espace de temps acceptable pour la production parce qu'il nécessite une optimisation linéaire entière avec beaucoup de variables.
La technique des arrondis donne dans ce cas une bonne approximation de la solution optimale.
Elle permet d'obtenir une solution satisfaisante en une vingtaine de secondes.

### 3.4.2 Démonstration de l'outil

Une démonstration de l'outil de sélection à l'intention des utilisateurs et des décideurs a été effectuée afin de vérifier l'adéquation des résultats avec leurs espérances. Les tests réalisés lors de cette démonstration sont du type de ceux présentés dans l'annexe 2. Les données relatives aux coils étaient celles du stock du jour. Les critères traduisaient les souhaits exprimés lors de la confection manuelle de la sélection.
Contrairement à ce qu'on aurait pu penser, le concept de poids a très bien été compris par les utilisateurs. Cependant les résultats obtenus avec le premier jeu de poids ne satisfaisaient généralement pas. Comme le temps de réponse ne dépassait pas quinze secondes, les utilisateurs se sont orientés vers une utilisation répétitive en variant les valeurs données aux $P_E$ $P_F$...$P_K$. Au vu de la première solution, ils arrivaient en un ou deux essais maximum à déterminer les poids qui leur fournissaient des résultats jugés très satisfaisants en comparaison de la solution obtenue manuellement.
Il apparaît donc que l'aide à la décision multicritère peut être facilitée par l'utilisation d'un outil interactif, d'autant plus efficace qu'il fournira au décideur une évaluation précise et synthétique du respect de ses souhaits, dans chaque compromis proposé.
La procédure des arrondis s'avère très efficace dans le cadre de la sélection des coils de Dudelange. Mais l'usage de la routine dans un autre contexte (sur des

objets autres que les coils) pourrait ne pas être adéquat et demandera donc à être vérifié. De même, il faudra s'assurer que les échelles sur lesquelles seront mesurés les critères, n'introduiront pas d'effets indésirables.

# AUTRES APPROCHES ENVISAGEES

## 4.1 Méthode de surclassement

Dans de la première partie de ce mémoire nous avons constaté à plusieurs reprises les difficultés auxquelles se heurtent les techniques fondées sur des fonctions d'utilité pour résoudre des problèmes multicritères. Car, même si en pratique l'existence de la fonction d'utilité n'est pas mis en cause, il faut admettre qu'elle n'est pas toujours facile à déterminer. D'autre part, l'analyse théorique montre que ces méthodes impliquent en général des hypothèses restrictives. En effet, elles impliquent l'existence d'une relation de comparaison entre toutes les actions ( cette relation est donc **totale**) interdisant ainsi l'**incomparabilité**. De plus, il résulte de l'utilisation des fonctions d'agrégation que, si a est meilleur que b, et b meilleur que c, il n'est plus nécessaire de consulter le décideur pour comparer a à c, on peut le "calculer" et a devra être préféré à c. La relation entre les actions sera donc **transitive**.

Pour permettre de lever ces hypothèses, qui peuvent être jugées trop contraignantes B.Roy et son équipe ont proposé des méthodes basées sur l'idée du surclassement. [Roy 85]
Ce sont ces techniques que nous allons aborder maintenant.

### 4.1.1 Les méthodes "ELECTRE"

Définition:

> Une relation de surclassement est une relation binaire définie sur l'ensemble A des actions, dont la signification est la suivante: une action a surclasse une action b si l'on peut affirmer sans trop de risques d'erreurs que le décideur préfère a à b.

Une telle relation n'a aucune raison d'être complète ni transitive. Elle ne fournit en général pas immédiatement le meilleur compromis ou le classement des actions recherché. La plupart des méthodes de surclassement seront donc composées de deux parties: la première construira la relation de surclassement et la seconde l'exploitera en vue de produire le résultat souhaité (soit le meilleur compromis, soit le classement).

B Roy distingue trois classes de problèmes dans l'aide à la décision.

-la problématique de choix ou problématique *alpha*, qui consiste à rechercher la meilleure action. Le choix d'une configuration informatique relève de cette problématique.

- la problématique du tri, dite aussi problématique *bêta*, est celle qui cherche à associer à chaque action, une classe d'actions. C'est la problématique du banquier qui lors de l'attribution des prêts souhaite déterminer si un candidat est "bon" ou "mauvais" ou encore "à risques".

-la problématique de rangement ou problématique *gamma*, consiste à classer les actions de la meilleure à la moins bonne. Le résultat d'un concours relève de cette problématique.

Pour les problématiques alpha et gamma nous allons examiner une des méthodes de la famille ELECTRE.

L'idée de base d'Electre réside dans la façon d'établir la relation de surclassement. Partant d'une prise de position de Condorcet concernant la démocratie, on dira qu'une action en surclasse une autre si elle est au moins aussi bonne que l'autre, relativement à une majorité de critères, sans être trop mauvaise relativement aux autres critères.

## A) ELECTRE I

C'est la plus ancienne des méthodes de la famille, elle fut proposée par B. Roy et Bertier [Roy 66]. Nous avons choisi de la présenter parce que, même si elle n'est plus guère utilisée aujourd'hui et qu'on lui préfère généralement sa version Electre Is, elle contient déjà la plupart des idées utilisées dans les versions suivantes.

a) Construction de la relation de surclassement

Pour chaque critère i, indiquer un poids $P_i$ croissant avec son importance.
Pour chaque couple d'actions (a,b), calculer l'indice de concordance selon la formule suivante:

$$c(a,b) = \frac{1}{P} \sum_{j \in w^+} P_j$$

$$\text{où } P = \sum_{i=1}^{n} p_j$$

et:

$w^+$ = { i tq l'évaluation du critère i pour l'action a, est supérieure à celle de l'action b}

$w^-$ = { i tq l'évaluation du critère i pour l'action a, est inférieure à celle de l'action b}

$w^=$ = { i tq l'évaluation du critère i pour l'action a, est égale à celle de l'action b}

L'indice de concordance rend compte de l'importance des critères favorables à l'affirmation "a surclasse b".

Pour exprimer l'importance des critères opposés à cette thèse on définit l'indice de discordance de la manière suivante:

$$d(a,b) = \max_i \frac{f_i(a) - f_i(b)}{R_i^*}$$

$f_i(x)$ = mesure de l'action x selon le critère i
$R_i^*$ = la plus grande valeur possible sur l'échelle d'évaluation du critère i

On se donne ensuite $\hat{d}$ et $\hat{c}$ les seuils de discordance et de concordance.

La relation de surclassement est alors entièrement définie par:

a S b <=>

$$\begin{cases} c(a,b) \geq \hat{c} \\ d(a,b) \leq \hat{d} \end{cases}$$

ce qui revient à dire qu'il ya suffisamment d'arguments pour admettre que a est au moins aussi bonne que b sans qu'il y ait de raison importante de refuser cette affirmation

b) Exploitation de la relation de surclassement

Partant de la relation de surclassement, on va construire un sous-ensemble N d'actions tel que toute action est surclassée par au moins une action de N et les actions de N sont incomparables entre elles.

Le sous-ensemble ainsi défini est un noyau au sens de la théorie des graphes. On peut noter que si le graphe de surclassement est sans circuit, le noyau existe toujours et est unique.
Une technique pour se débarrasser des circuits consiste à considérer comme une seule action toutes les actions du circuit.

## B) ELECTRE II

Cette méthode, toujours proposée par B. Roy [Roy 73] et son équipe répond à la problématique gamma.

a) Construction de la relation de surclassement

Contrairement à Electre 1, les auteurs proposent non pas une, mais deux relations de surclassement: un surclassement fort et un faible.

On définit donc:

a $S_F$ b <=>

$$\begin{cases} c(a,b) \geq \hat{c}_1 \\ \sum_{j \in w^+} p_j > \sum_{j \in w^-} p_j \\ d(a,b) \leq \hat{d}_1 \end{cases}$$

a $S_f$ b <=>

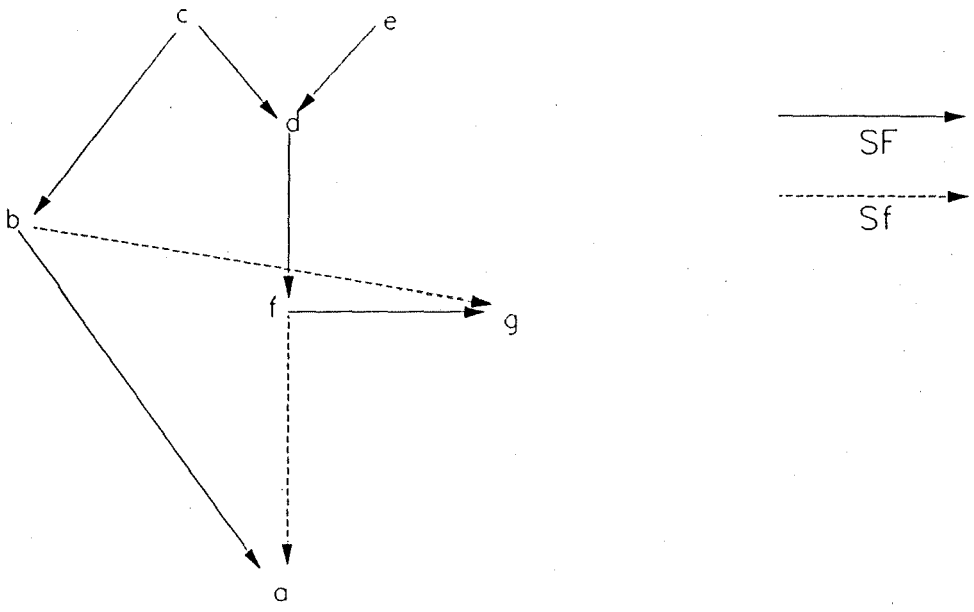
$$\begin{cases} c(a,b) \geq \hat{c}_2 \\ \sum_{j \in w^+} p_j > \sum_{j \in w^-} p_j \\ d(a,b) \leq \hat{d}_2 \end{cases}$$

avec:

$$\hat{c}_1 > \hat{c}_2 \text{et}$$
$$\hat{d}_1 > \hat{d}_2$$

b) Exploitation de la relation de surclassement

A partir de la relation de surclassement, on établit un graphe (comme celui de la figure ci-dessous) qui comporte deux types d'arcs: les traits pleins indiquent les surclassements forts et les traits discontinus, les surclassements faibles.




Il peut arriver que ce graphe contiennent des circuits, on les élimine comme pour la méthode Electre 1. On recherche dans la problématique gamma un classement; on va donc tenter de ranger toutes les actions par classe. Pour cela on considère la relation de surclassement fort et spécialement les chemins qu'elle forme dans le graphe. On s'intéresse, pour chaque action, à la longueur du chemin (nombre d'arcs qui constituent le chemin) qui y aboutit. La première classe est donnée par les actions non surclassées, c'est-à-dire celles auxquelles aboutit un chemin de longueur nulle. On définit les classes suivantes en fonction des longueurs croissantes. Comme le but est de départager toutes les actions, on utilise les renseignements issus de la relation de surclassement faible, de manière à départager les ex aequo au sein d'une même classe.

On réalise ensuite le classement inverse, c'est-à-dire qu'au lieu de considérer les longueurs des chemins qui aboutissent à une action, on s'intéresse aux longueurs des chemins qui en partent. On range, cette fois, les actions en fonction de la longueur décroissante des chemins et toujours en utilisant uniquement le surclassement fort. Le surclassement faible n'intervient que pour trancher entre les ex aequo

La dernière étape fait appel au jugement du décideur. Si les deux classements sont proches, les résultats de la procédure sont solides; il peut soit garder les deux classements soit calculer un classement médian où, les rangs sont la moyenne de ceux obtenus dans les deux premiers classements. Il est fortement conseillé de toujours analyser les deux classements obtenus et de regarder "ce qui bouge". En effet on n'est pas à l'abri de certaines surprises.

### 4.1.2 La méthode "ELECTRE" adaptée aux coils

A première vue, le problème des coils relève de la problématique alpha. La technique qui devrait être utilisée est donc ELECTRE 1. Mais on voit très vite que pour utiliser cette technique, il ne faut pas un espace d'actions trop grand, puisqu'il faut les comparer deux à deux. On ne peut donc utiliser l'ensemble des $2^{500}$ possibilités [1] de solutions au problème des coils. Il faut trouver un moyen de réduire cet ensemble, si l'on veut pouvoir utiliser la méthode. On pourrait, par exemple, se limiter à un sous- ensemble de solutions efficaces. C'est l'idée que des chercheurs du Lamsade (Centre de recherche dirigé par B. ROY) utilisent pour tester une méthode dérivée d' ELECTRE.
La démarche proposée présente les trois phases suivantes.
La première phase consiste à générer des solutions [2] efficaces. Elle serait fondée sur l'idée suivante:
déterminer les solutions du modèle ci-dessous basé sur les distances pondérées de Tchebycheff:

(1) qui consiste à dire, si pour chacun des 500 coils, on le prend ou si on ne le prend pas.

(2) Comme pour SELECTION une solution consiste à donner pour chaque famille le nombre d'objets (de coils) que l'on prend.

$$\text{Min} \quad \{\alpha\}$$

$$\alpha \geq \lambda_i (Z_i^{**} - Z_i) \quad \forall i = 1, \ldots, k$$

$$a - e_a \leq AX \leq b + e_b$$

$$e_a, e_b \geq 0$$

$$\alpha \geq 0 \; ; X \geq 0 \quad (X \in \Re)$$

où:

$\alpha$ est une variable réelle.
$Z_i^{**}$ l'optimum pour le $i^{ème}$ objectif
$\lambda_i$ la $i^{ème}$ composante du vecteur poids
et $Z_i$ l'expression du $i^{ème}$ objectif.

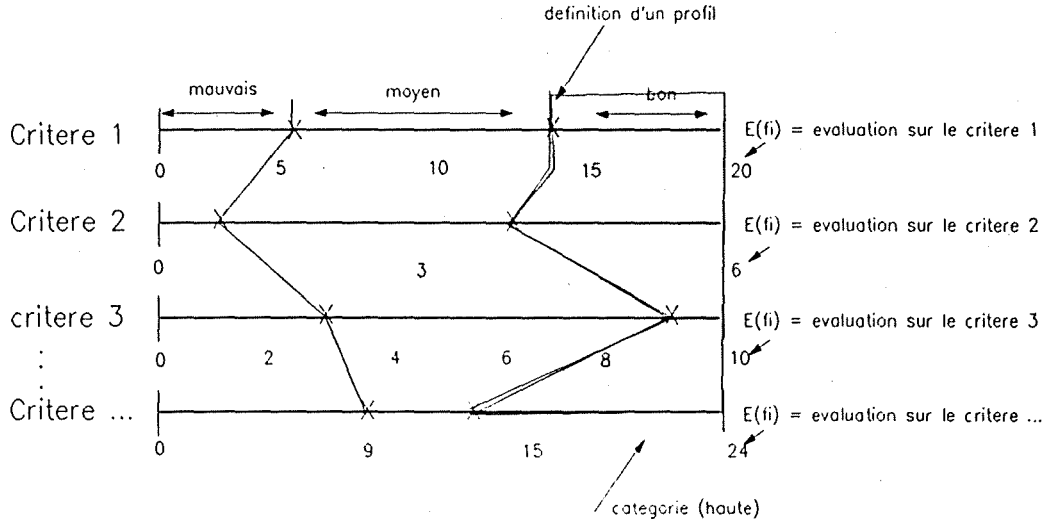
et ce pour différentes valeurs du paramètre $\lambda$

On peut noter que ce procédé génère bien toutes les solutions efficaces, cependant on trouve dans [Zeleny] et [Vincke 88] des théorèmes précisant que toutes les solutions obtenues de cette façon ne sont pas nécessairement efficaces.

Lors de cette première phase, on ne considérera pas le critère I pour éviter l'utilisation de la programmation linéaire entière.
Il faudra encore recourir à une procédure d'arrondi pour obtenir une *solution*

Vient ensuite une phase de tri; les solutions obtenues lors de la première phase seront réparties en plusieurs **catégories** (ex: bon, moyen, mauvais). Le but étant d'éliminer un grand nombre d'actions et de n'en garder qu'un nombre restreint de l'ordre d'une vingtaine. Pratiquement chaque action sera évaluée sur certains des critères(1) (intervenant dans la sélection). On définira pour cela des **profils** délimitant les différentes catégories d'actions.

(1) On pense spécialement à ceux qui ne sont pas intervenus dans la phase de génération des solutions efficaces. Ex: la contrainte (I) du nombre minimum par classe




exemple de définition de catégories.

Après avoir effectué le tri on ne retiendra que la catégorie la plus haute (ou éventuellement la réunion de deux ou trois catégories) déterminée au sens multicritère selon la problématique bêta, grâce à une méthode ELECTRE, en cours de développement au Lamsade. On ne gardera ainsi qu'une vingtaine d'actions qui seront traitées par une procédure de rangement. Les trois ou quatre premières seront présentées au décideur qui en choisira une, suivant les performances de chacune d'elles. Le rangement envisagé sera effectué sur une famille de critères, qui peut reprendre les critères utilisés pour SELECTION mais formalisés autrement, de même que certains critères de la procédure de tri.

Remarque:
Il est intéressant de noter que le problème des coils nécessite l'utilisation des deux techniques présentées (goal programming et ELECTRE) qui ne s'opposent pas mais au contraire se complètent.

## 4.2 Programmation par contraintes

Dans le cadre du projet d'Optimisation Dynamique de la Gestion de Production, parallèlement à la réalisation de l'outil de sélection à l'ARBED et aux recherches de l'équipe du Lamsade, le centre de Recherche Public, Centre Universitaire à Luxembourg évalue, toujours sur le même problème, l'efficacité d'un langage de programmation par contraintes. Notre but n'est pas bien sûr de présenter en détail les travaux du CRP sur le sujet, mais nous voudrions donner une idée de cette approche qui ne manque pas d'intérêt. Elle repose sur l'utilisation d'un langage spécialisé: PrologIII.
Après un bref rappel des caractéristiques du langage Prolog nous verrons les extensions que lui apporte PrologIII. Ensuite nous examinerons comment il peut être utilisé pour réaliser des sélections.

### 4.2.1 PROLOG

Dans un langage de programmation classique, comme Pascal, C, etc..., on attache une valeur à une variable. L'exécution d'un programme consiste à modifier la valeur de la variable par une instruction d'affectation. Il incombe au programmeur de veiller à ce que son programme puisse traiter convenablement toute possibilité pouvant survenir. Cette programmation est de type procédurale.
En Prolog, le déroulement d'un programme consiste à déterminer la (souvent les) valeurs des variables. C'est au système Prolog qu'est laissé la charge d'explorer toutes les possibilités. Une variable représente un objet pouvant prendre des valeurs dans un certain domaine, comme une inconnue dans une équation mathématique. Prolog est un langage déclaratif. Il est conçu pour représenter la connaissance que l'on a sur certains domaines. Un domaine est un ensemble d'objets et la connaissance est matérialisée par un ensemble de relations. Ces relations décrivent les propriétés de ces objets et leurs interactions. Un programme Prolog est constitué de règles décrivant ces objets et ces relations. Programmer en Prolog consiste donc à:

| | |
|---|---|
| - déclarer des faits ou assertions<br>- définir des règles<br>- poser des questions | sur des objets et leurs interactions. |

Ces trois types de clauses (faits, règles et questions) peuvent être illustrées par l'exemple suivant.

les faits:

```
client(cl1,"Durand","Mertert");
client(cl2,"Dupont","Paris");
client(cl3,"Buch","Mertert");
client(cl4,"Dupont","Metz");
client(cl5,"Dupont",'Mertert");

commande(cm1,cl1,2601);
commande(cm2,cl1,1201);
commande(cm3,cl1,0302);
commande(cm4,cl1,0502);
commande(cm5,cl1,1003);
commande(cm6,cl1,1201);
commande(cm7,cl1,1711);
```

une règle:

```
ensemble_de_commandes(N_umCmd,  N_omClmient, A_dresse)
->
        commande(N_umCmd,  N_umClient, D_ate)
        client(N_umClient,  N_omClient, A_dresse);
```

On voit qu'une variable, par exemple, N_omClient évoque un objet sans savoir ce qu'il représente. N_omClient ne représente pas un objet particulier, mais tout objet appartenant à l'ensemble (éventuellement vide) de ceux qui ont la propriété d'être le nom d'un client. Le domaine de la variable N_omClient est celui des noms de clients.

Il y a une grande analogie entre la programmation en Prolog et les bases de données relationnelles. L'ensemble des faits constitue le contenu (une instance) de la base de données, et les questions, interrogeant les relations, sont comparables aux consultations de la base de données. Par exemple, en SQL: SELECT Client WHERE Nom = "Durand"

Exemples de questions à poser à ce programme:

```
ensemble_de_commandes(N_cmde,   "Durand",  A_dresse);
```

*réponse:*

```
{Ncmde  =  cm1, A_dresse =  "Mertert"}
{Ncmde  =  cm2, A_dresse =  "Mertert"}
{Ncmde  =  cm3, A_dresse =  "Mertert"}
```

```
ensemble_de_commandes(N_cmde,   C_lient, "Mertert");
```

*réponse:*

```
{Ncmde  =  cm1, Client =  "Durand"}
{Ncmde  =  cm2, Client =  "Durand"}
{Ncmde  =  cm3, Client =  "Durand"}
{Ncmde  =  cm5, Client =  "Dupont"}
```

```
ensemble_de_commandes(N_cmde,   C_lient, A_dresse);
```

*réponse:*

```
{N_cmde  =  cm1, C_lient =  "Durand",  A_dresse =  "Mertert"}
{N_cmde  =  cm2, C_lient =  "Durand",  A_dresse =  "Mertert"}
{N_cmde  =  cm3, C_lient =  "Durand",  A_dresse =  "Mertert"}
{N_cmde  =  cm4, C_lient =  "Dupont",  A_dresse =  "Paris"}
{N_cmde  =  cm5, C_lient =  "Dupont",  A_dresse =  "Mertert"}
{N_cmde  =  cm6, C_lient =  "Dupont",  A_dresse =  "Metz"}
{N_cmde  =  cm7, C_lient =  "Bush",  A_dresse =  "Baghdad"}
```

### Mécanisme d'unification

La règle permettant de décider si deux termes S et T peuvent s'unifier est récursive et s'exprime comme ceci:

Si S et T sont des constantes, alors S et T s'unifient si et seulement si elles sont le même objet.

Si S est une variable et T n'importe quoi, alors ils s'unifient. S est instancié à T. De même, si T est la variable, T est instancié à S.

Si S et T sont des structures, alors ils s'unifient seulement si:

S et T ont le même foncteur principal et
toutes leurs composantes s'unifient deux à deux.

### Arbre ET/OU et backtracking

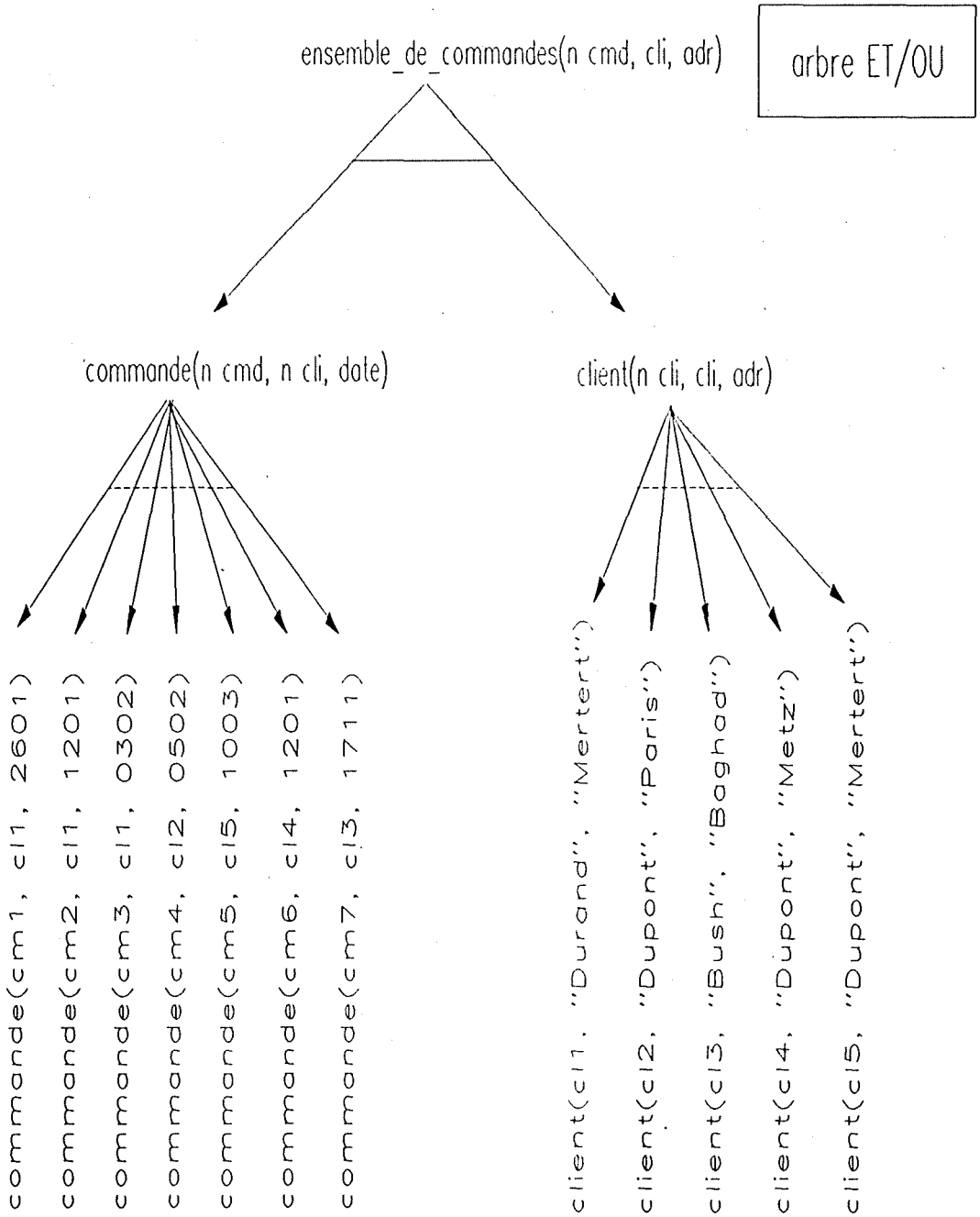
Le contrôle Prolog procède par essais successifs afin de déterminer l'ensemble des solutions d'un problème donné. Cette démarche se représente par le parcours en profondeur d'abord d'un arbre ET/OU. Les arbres ET/OU, utilisés en intelligence artificielle, permettent de représenter toute exécution d'un programme Prolog. Ce type d'arbre se décrit comme suit:

-les faits sont des feuilles et les feuilles sont des faits.

-Les noeuds ET correspondent à la situation suivante: "pour effacer un but B, on efface la conjonction (d'où le nom ET) de ses sous-buts B1,...,Bn. Ces sous-buts sont les fils du noeud.
Les branches sont reliées par un trait plein.

-Les noeuds OU correspondent à la situation suivante: "pour effacer un but B, il suffit d'effacer un des buts équivalents de la disjonction (d'où le nom OU) des buts B1,...,Bn. Ces buts sont les fils du noeud. Leurs branches sont reliées par un trait discontinu.

Le système Prolog parcourt cet arbre et le mécanisme de backtracking se produit lors d'un échec ou à l'arrivée à une feuille. Un backtracking est un retour au dernier point de choix.


ensemble_de_commandes(n cmd, cli, adr)
arbre ET/OU
commande(n cmd, n cli, date)
client(n cli, cli, adr)
commande(cm1, c11, 2601)
commande(cm2, c11, 1201)
commande(cm3, c11, 0302)
commande(cm4, c12, 0502)
commande(cm5, c15, 1003)
commande(cm6, c14, 1201)
commande(cm7, c13, 1711)
client(c11, "Durand", "Mertert")
client(c12, "Dupont", "Paris")
client(c13, "Bush", "Baghad")
client(c14, "Dupont", "Metz")
client(c15, "Dupont", "Mertert")

### 4.2.2 PROLOG III

Ce langage permet la résolution de contraintes dans un domaine muni d'opérations et de relations:

-traitement du numérique (programmation linéaire).
-traitement complet de l'algèbre de Boole (calcul propositionnel).

Prolog III est donc un langage de programmation par contraintes et aborde ainsi le domaine de la recherche opérationnelle. Prolog III a le même caractère formel et déclaratif que Prolog.
Voici un exemple simple de programme Prolog III. On établit la relation entre le nombre O d'oiseaux et le nombre C de chats qui ensemble, totalisent T têtes et P pattes:

```
ChatsOiseaux  (C,O,P,T)  ->
                {P = 4C + 2O, T = C + O
                C >=  0, O >=0,  P >=  0, T >=0  };
```

deux requêtes possibles seraient:

```
ChatOiseaux(C,  O, 14, 5);
ChatOiseaux(1,  O, P, 5);
```

et on obtient:

```
{C = 2, O = 3 }
{O = 4, P = 12}
```

On peut souligner, au passage, la possibilité de travailler en nombres entiers en Prolog III, alors que les équations, elles, sont résolues dans l'espace des rationnels.
En effet la requête:

ChatOiseaux(C, O, 7, 3);

donne:

{C = 1/2, O = 5/2}

Pour n'obtenir que des solutions entières, il faut utiliser le prédicat prédéfini enum(x), qui impose à la variable x d'être entière. Donc, en posant la requête à trois buts:

ChatOiseaux(C, O, 7, 3) enum(C) enum(O);

cette fois, on obtient comme résultat l'ensemble vide, parce qu'il n'existe pas de solution entière à cette requête.

## La machine Prolog III

L'exécution d'un programme Prolog III vise à résoudre le problème suivant: "étant donné une suite t0,...,tn de termes et S un système de contraintes, trouver les valeurs des variables qui transforment:

-les termes $t_i$ en des faits définis par le programme, et
-les contraintes du système S en des conditions vérifiées".

La suite des termes ou le système S peut, bien sûr, être vide.
Considérons une machine abstraite non déterministe dont l'unique instruction de base est décrite par les trois formules:

(1) (W, t0 t1...tN , S)
(2) s0 -> s1...Sm, R
(3) (W,s1...Sm t1...tN,S U R U {s0 = t0)

La formule (1) représente l'état de la machine à un instant donné. W est l'ensemble des variables auxquelles on s'intéresse. La requête courante est t0 t1...tn, S.
La formule (2) représente la règle du programme utilisée pour changer d'état. Au besoin, la machine a renommé certaines de ses variables pour qu'il n'y en ait aucune de commune avec (1). Les autres règles du programme auraient aussi bien pu être employées, mais utiliser celle-ci plutôt qu'une autre résulte d'un choix.

La formule (3) donne le nouvel état de la machine, après l'application de la règle (2). Le passage dans cet état n'est possible que si le système de contraintes S U R U {s0 = t0 } possède une solution pour laquelle chacun des termes de la suite s1...sm t1...tn représente un arbre bien défini.

Pour répondre à la requête t0 t1...tn,S, la machine partira de l'état initial (W,t0 t1...tn,S) où W est l'ensemble des variables figurant dans la requête. Elle passera ensuite dans tous les états qu'elle peut atteindre en répétant l'opération de base ci-dessus. Bien entendu, le choix évoqué plus haut exprimé par la formule (2) n'est pas laissé au hasard. Prolog III fait en sorte que toutes les règles soient successivement essayées dans l'ordre où elles sont écrites dans le programme. Du moins, s'agit-il de toutes les règles raisonnables, c'est-à-dire toutes, sauf celles qui, de manière évidente, rendraient le système S U R U {s0 = t0} insoluble. Ainsi, la machine Prolog III est la machine déterministe qui produit toutes les exécutions possibles de la machine non déterministe définie par les expressions (1) (2) (3).

Chaque fois qu'il parviendra à un état de la forme (W, ,S), c'est-à-dire dont la suite des buts est vide, Prolog III fournira comme réponse à l'utilisateur la solution du système S sur l'ensemble des variables W. Du moins, si elle existe. Si le système est soluble sans pour autant qu'il y ait assez de contraintes pour déterminer une solution, Prolog III fournit ce système entre accolades.

Il apparaît donc clairement que, partant de l'état initial (W, t0 t1...tn,S), la suite des termes diminue (ou plus généralement fluctue jusqu'à diminuer) pour se vider totalement. D'où le verbe effacer les buts. A l'opposé, le système S de contraintes ne cesse de s'agrandir, ce qui revient à prendre une direction de plus en plus précise vers une solution.

Exemple:

Contraintes numériques

Considérons le problème d'affectation suivant: on dispose de 5 postes de travail pour 5 ouvriers. A toute affectation $(x_i,y_i)$ correspond un coût $d_{ij}$. On veut affecter les 5 ouvriers aux 5 postes, de manière à ce que chaque ouvrier soit à un poste et à un seul. Ceci de telle sorte que le coût total des affectations soit minimal. La matrice des coûts est la suivante:

| X \ Y | | | | |
|---|---|---|---|---|
| 7 | 3 | 5 | 7 | 10 |
| 6 | 1000 | 1000 | 8 | 7 |
| 6 | 5 | 1 | 5 | 1000 |
| 11 | 4 | 1000 | 11 | 15 |
| 1000 | 4 | 5 | 2 | 10 |

La solution peut être obtenue par le programme suivant:

```
Affect(<<x11,  x12, x13, x14, x15>,
        <x21,  x22, x23, x24, x25>
        <x31,  x32, x33, x34, x35>
        <x41,  x42, x43, x44, x45>
        <x51,  x52, x53, x54, x55>,  c) ->
      Min_value(c,c),
        {x11 >=  0, x12 >=  0, x13 >=  0, x14 >=  0, x15 >=  0,
        x21 >=  0, x22 >=  0, x23 >=  0, x24 >=  0, x25 >=  0,
        x31 >=  0, x32 >=  0, x33 >=  0, x34 >=  0, x35 >=  0,
        x41 >=  0, x42 >=  0, x43 >=  0, x44 >=  0, x45 >=  0,
        x51 >=  0, x52 >=  0, x53 >=  0, x54 >=  0, x55 >=  0,

        x11 + x12 + x13 + x14 + x15 = 1,
        x21 + x22 + x23 + x24 + x25 = 1,
        x31 + x32 + x33 + x34 + x35 = 1,
        x41 + x42 + x43 + x44 + x45 = 1,
        x51 + x52 + x53 + x54 + x55 = 1,


        x11 + x21 + x31 + x41 + x51 = 1
        x12 + x22 + x32 + x42 + x52 = 1
        x13 + x23 + x33 + x43 + x53 = 1
        x14 + x24 + x34 + x44 + x54 = 1
        x15 + x25 + x35 + x45 + x55 = 1
```

```
c = 7x11 + 3x12 + 5x13 + 7x14 + 10x15 +
6x21 + 1000x22 + 1000x23 + 8x24 + 7x25 +
6x31 + 5x32 + 1x33 + 5x34 + 1000x35 +
11x41 + 4x42 + 1000x43 + 11x44 + 15x45 +
1000x51 + 4x52 + 5x53 + 2x54 + 10x55 };
```

La requête est évidente:

```
Affect(M,c);
```

La réponse suivante est alors obtenue:

{ M = <<1,0,0,0,0>, <0,0,0,0,1>, <0,0,1,0,0>, <0,1,0,0,0> <0,0,0,1,0>, c=21}

Cette façon de procéder fonctionne efficacement, mais présente un inconvénient majeur: ce programme est trop statique. C'est-à-dire que, si les données ou le nombre de postes et d'ouvriers venaient à changer, il faudrait réécrire une nouvelle version du programme. La deuxième solution, ne mentionnant plus explicitement les contraintes, montre que Prolog III construit lui-même le système de contraintes puis le résout. Donc, cette fois, la taille du système d'équations variera automatiquement en fonction de la taille du problème posé au sein de la requête. Voici cette solution:

```
Affectation(M,C,c) ->
        Cout(M,C,c)
        SommeChaqueLigne(M,1)
        Transpose(M,M')
        SommeChaqueLigne(M',1)
        Min_value(c,c);
Cout(<>,<>,0)      ->;

Cout(<L>.M,   <L'>.C,   s+c) ->
        ProdScal(L,L',s)
        Cout(M,C,c);
```

```
ProdScal(<>,<>,0)        ->;

ProdScal(<x>.L,<x'>.L,x*x'      + s) ->;
                ProdScal(L,L',s)
                {x >=   0 };

SommeChaqueLigne(<>,s)        ->;

SommeChaqueLigne(<L>.M,s)        ->;
                Somme(L,s)
                SommeChaqueLigne(M,s);

Somme(<>,0)      ->;

Somme(<e>.L,e+s)        ->
                Somme(L,s);
Transpose(<>,T)     -> {T,<<>>      =  <<>>,T};

Transpose(<L>.M,T)      ->
                Ajouter(L,M',T)
                Tranpose(M,M');

AjouterLigne(<>,<>,<>)          ->;

AjouterLigne(<e>.L,     <L'>.M,    <<e>.L'>.M')       ->
                Ajouter(L,M,M');
```

puisque la matrice de coût ne réside plus dans aucune règle, il faut la fournir lors de la requête:

```
Affectation(M,C,c),
{C  =  <<7,   3, 5 ,7, 10>,
<6  ,1000,  1000,  8,  7>,
```

```
<6,  5, 1, 5, 1000>,
<11,  4, 1000, 11, 15>,
<1000,  4, 5, 2, 10>>};
```

Solution obtenue:

```
{M =
<<1,0,0,0,0>,   <0,0,0,0,1>,  <0,0,1,0,0>,  <0,1,0,0,0>  <0,0,0,1,0>,C
= <<7,  3, 5 ,7, 10>,  <6  ,1000, 1000, 8, 7>,  <6,  5, 1, 5, 1000>,  <11,
4, 1000, 11, 15>,  <1000,  4, 5, 2, 10>>,   c =  21};
```

Une autre façon de poser la même requête est de fournir différemment cette matrice de coûts:

```
Affectation(M,
<<7,  3, 5 ,7, 10>,
<6  ,1000, 1000, 8, 7>,
<6,  5, 1, 5, 1000>,
<11,  4, 1000, 11, 15>,
<1000,  4, 5, 2, 10>>,   c);
```

On obtient forcément la même solution, sans la valeur de la matrice, puisque le deuxième argument de l'appel n'est plus une variable:

```
{M =
<<1,0,0,0,0>,   <0,0,0,0,1>,  <0,0,1,0,0>,  <0,1,0,0,0>  <0,0,0,1,0>,
c =  21};
```

### 4.2.3 RESULTATS

A la vue du deuxième exemple, on devine aisément la possibilité d'utiliser des clauses pour générer le système d'inéquations du problème linéaire associé au problème de sélection de coils. On pourrait reprendre le modèle de goal programming de la partie 1.
On utilise alors la puissance de Prolog pour "programmer" la transformation des critères et des données relatives aux objets, en modèle. Les extensions de programmation linéaire offertes par Prolog III résolvent ensuite ce modèle.
Cette technique peut paraître en tous points semblable à celle de la partie 1. C'est en effet le cas, les différences résident dans le langage de programmation utilisé: Fortran est remplacé par Prolog et la routine H02BBF par une boîte noire: les extensions constituant PrologIII.

Les avantages relevés en faveur de Prolog III sont, bien évidemment, ceux de Prolog en général: rapidité de développement, puissance du langage, lisibilité, et "simplicité" de la programmation.

Les points faibles résident essentiellement dans la relative lenteur de la génération du modèle mais, surtout dans l'absence de programmation entière qui a pour effet de provoquer l'énumération de toutes les possibilités lorsque des variables doivent prendre des valeurs entières. On se retrouve face à l'explosion combinatoire évoquée au point: 3.2.1.D.a. Mais un langage spécifiquement orienté vers des parcours d'arbres et capable de résoudre des problèmes linéaires doit permettre de programmer une recherche intelligente du type branch and bound.

L'utilisation de Prolog III en production semble cependant peu réaliste, vu l'absence de diagnostic en cas d'erreur au niveau du problème linéaire.

nb: Une sélection identique a été réalisée sur VAX (module de sélection) et sur Mac (Prolog III) les temps obtenus sont respectivement 15 secondes et 5 minutes. Ce qui, compte tenu de la différence des machines, semble une bonne performance.

## Bibliographie

[Goicoechea ] : Ambrose Goicoechea, Don R Hansen, Lucien Duckstem.
" Multiobjective decision analysis whith engineering and business applications."
(John Wiley)

[Charnes & Cooper 70] : A. Charnes W.W. Cooper
" Management models and industrial applications of linear programming."
Vol 1 (John Wiley, New-York (1971) )

[Geoffrion & Marsten 72] : A. M. Geoffrion et R. E. Marsten
" Integer programming algorithms: a framework and state-of-the-art survey."
(Management Science, Vol 18, N° 9, mai 1972)

[Keeney & Raiffa 76] Keeney R.L. Raiffa H.
" Decisions whith Multiple Objectives: Préference and Value Tradeoffs."
John Wiley New-York 1976.

[Carlier & Chrétienne] Carlier J. Chrétienne P.
" Problèmes d'ordonnancement modélisation/ complexité/ algorithmes."
Etude et Recherche en Informatique (Masson)

[Benayoun 70] Benayoun R. De Montgolfier J. Tergny j. Larichev O.
" Linear programming whith multiple objective functions: STEP Method (STEM)."
Mathématical programming 1 PP 366-375

[Borlon 77] : Jean-Luc Borlon
" Implémentation d'un langage interactif de programmation linéaire multicritère."
(Mémoire en Informatique FUNDP (1977) )

[Evans 84] : Gerald W. Evans
" An overview of techniques for solving multiobjective mathematical programs."
(Management Science Vol 30 N° 11 Novembre 1984)

[Fichefet 76] : J. Fichefet
" GPSTEM: An interactive multi-objective optimisation method."
(Progress in Operations Research, Vol 1, édité par A. Prékopa, North-Holland, Amsterdam 1976 )

[Fichefet 82] Fichefet J.
" Eléments de programmation linéaire."
Notes du cours Recherche Opérationnelle de Gestion FUNDP (Namur)

[Forrest 74] : J.J.H Forrest, J.P. Hirst et J.A. Tomlin
" Practical solution of large mixed integer programming problems whith Umpire."
(Management Science, Vol 20, N° 5, janvier 1974)

[Gautier & Ribière 77] : J. M. Gautier et G. Ribière
" Experiments in mixed-integer linear programming using pseudo-costs."
(Mathematical Programming 12 (1977))

[Ignizio 76] : James P.Ignizio
"Goal programming and extensions."
(Lexington Books)

[Ignizio 89] James P. Ignizio
" On the Merits and Demerits of Integer Goal programming."
J. Opl Res vol 40 n°8 pp 781-785 (1989)

[Jacquet-Lagrèze 87] E. Jacquet-Lagrèze
" A projected Gradient Method for Linear Programming."
Cahier du Lamsade n° 79 juin 1987 (Université de Paris-Dauphine)

[Kachian 79] Kachian L.G.
" A polynomial Algorithm in Linear Programming."
Soviet Math Doklady 1979 20, 191-194

[Karmakar 84] Karmakar N.
" A new polynomial Time Algorithm for linear Programming."
Combinatorica, 4, 373-395

[Klee & Minty 72] Klee Y. Minty G.J.
" How good is the Simplex algorithm ?"
In Shisha O. (ed) Inéqualities III, Académic press New-York pp 159-175.

[Minoux 88] Minoux M.
" Programmation mathématique Théorie et algorithmes."
(Collection technique et scientifique des télécommunication CNST)

[NAG] Numerical Algorithms Group
Numerical Algorithms Group Ltd.,
NAG Central Office, Mayfield House,
256 Banbury Road, Oxford 0X2 7DE
Tel (0865) 511245

[Nemhauser & Wolsey 89] : George L. Nemhauser et Laurence A. Wolsey
" Integer and Combinatorial Optimisation."
(Wiley Interscience)

[Ravindran 87] Ravindran Phillips Solberg
" Operations Research Principles and Practice ."
John Wiley & Sons New-York

[Roy 66] Benayoun R. Roy B. Sussman B.
" ELECTRE: une méthode pour guider le choix en présence de critères multiples."
Direction scientifique note de travail n°49 juin 1966. Société d'Economie et de Mathématiques Appliquées, Paris.

[Roy 73] Roy B. Bertier P.
" La méthode ELECTRE II."
Note de travail n°142 SEMA

[Roy 85] Roy B.
" Méthodologie multicritère d'aide à la décision."
Economica, Paris.

[Schärlig] Alain Schärlig
" Décider sur plusieurs critères panorama de l'aide à la décision multicritères."

[Steuer 79] Ralph E. Steuer
" Goal Programming Sensitivity Analysis Using Interval Penalty Weights."
Mathematical Programming 17 (1979) 16-31

[Tabucanon 89] Mario T. Tabucanon
" Multiple Criteria Decision Making in Industry."
Studies in Production and Engineering Economics 8 (Elsevier)

[Teghem & Kunsch ] : J. Teghem et P. Kunsch
" Panorama des méthodes multicritères de programmation linéaire à variables entières."

[Vincke 76] Vincke Ph.
" Une méthode interactive en programmation linéaire à plusieurs fonctions économiques."
Revue Française d'informatique et de Recherche Opérationnelle, 2, pp 5-20

[Vincke 88] : Philippe Vincke
" L'aide multicritère à la décision."
(Editions de l'Université de Bruxelles)

[Yourdon 89] : Edward Yourdon
" Modern structured Analysis."
(Yourdon Press computing series (1989) )

[Yu & Zeleny 75] Yu P.L. et Zeleny M.
"The set of all Nondominated Solutions in Linear Cases and a Multicriteria Simplex Method."
Journal of Mathematical and Applications 49, 430-468 (1975)

[Zeleny] Zeleny M.
" Linear Multiobjective Programming."
Lecture notes in Economics and Mathematical Systems vol 95 Springer Verlag.

## ANNEXE 1

Exemple de données de sélection.

1) Les critères:

7 nombre de critères utilisés pour la sélection

| | |
|---|---|
| 5<br>11 0.1<br>but: minimiser le délai | numéro du critère utilisé<br>paramètre du critère 5 (n°11) + POIDS (0.1) |
| 10<br>10 2100000 2220000 1E20<br>but: 210000 < tonnage < 2220000 | numéro du critère utilisé<br>paramètres du critère 10 (= n° 10, 2100000, 2220000) + POIDS (= 1E20) |
| 8<br>10 6 'FR-TOLES' 48 52 1<br>but: le % de FR-TOLES compris entre 48% et 52% | numéro du critère utilisé<br>paramètres du critère 8 + POIDS |
| 8<br>10 6 'FEUILLARDS' 6 7 1<br>but: le % de FEUILLARDS compris entre 6% et 7% | numéro du critère utilisé<br>paramètres du critère 8 + POIDS |
| 8<br>10 6 'GALVALANGE' 23 27 1<br>but: le % de GALVALANGE compris entre 23% et 27% | numéro du critère utilisé<br>paramètres du critère 8 + POIDS |
| 8<br>10 6 'GALVA-ADU' 17.5 19.5 1<br>but: le % de GALVA-ADU compris entre 17.5% et 19.5% | numéro du critère utilisé<br>paramètres du critère 8 + POIDS |

| | |
|---|---|
| 9<br>4 4 1<br>but: 4 coils au-moins par classe de largeur. | numéro du critère utilisé<br>paramètres du critère 4 + POIDS |

## 2) les familles:

pour chaque famille on donne: (sur une ligne)

- a le nombre de coils identiques qui la composent
- b le diamètre intérieur
- c laminage par poste complet (booléen)
- d coils difficiles à laminer (booléen)
- e la classe de largeur
- f le numéro identifiant du poste
- g la destination
- h productivité
- i la largeur
- j l'épaisseur
- k le poids
- l le délai
- m la longueur
- n le numéro identifiant de la famille

| a | b | c | d | e | f | g | h | i | j | k | l | m | n |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 610 | 1 | 1 | 1500 | 0063/01 | FR-TOLES | 1 | 1500 | 3431 | 18307 | 10 | 452 | 114138 |
| 1 | 610 | 1 | 1 | 0900 | 0100/05 | FR-TOLES | 1 | 903 | 2030 | 13920 | -369 | 967 | 114157 |
| 1 | 610 | 1 | 1 | 1100 | 0150/03 | FR-TOLES | 1 | 1120 | 1840 | 16563 | -271 | 1023 | 15413 |
| 2 | 610 | 1 | 1 | 1500 | 0190/54 | FR-TOLES | 1 | 1515 | 2545 | 29763 | -96 | 982 | 28488 |
| 3 | 610 | 1 | 1 | 1200 | 0190/55 | FR-TOLES | 1 | 1265 | 5040 | 25134 | -96 | 501 | 28738 |
| 1 | 610 | 1 | 1 | 1200 | 0190/58 | FR-TOLES | 1 | 1265 | 3440 | 25110 | -96 | 735 | 28521 |
| 1 | 610 | 1 | 1 | 1200 | 0190/60 | FR-TOLES | 1 | 1265 | 2550 | 25135 | -96 | 992 | 28501 |
| 1 | 610 | 1 | 1 | 1000 | 0190/61 | FR-TOLES | 1 | 1015 | 5030 | 12500 | -96 | 311 | 28829 |
| 1 | 610 | 1 | 1 | 1000 | 0190/62 | FR-TOLES | 1 | 1015 | 4550 | 24270 | -96 | 669 | 28566 |
| 2 | 610 | 1 | 1 | 1000 | 0190/65 | FR-TOLES | 1 | 1015 | 2835 | 24550 | -96 | 1086 | 28709 |
| 1 | 610 | 1 | 1 | 1000 | 0190/66 | FR-TOLES | 1 | 1015 | 2560 | 24745 | -96 | 1213 | 28714 |
| 2 | 610 | 1 | 1 | 1500 | 0190/68 | FR-TOLES | 1 | 1515 | 5000 | 26825 | -82 | 451 | 11713 |
| 3 | 610 | 1 | 1 | 1500 | 0190/69 | FR-TOLES | 1 | 1515 | 4000 | 25980 | -82 | 545 | 11654 |
| 3 | 610 | 1 | 1 | 1500 | 0190/70 | FR-TOLES | 1 | 1515 | 2500 | 24653 | -82 | 829 | 11625 |
| 2 | 610 | 1 | 1 | 1200 | 0190/71 | FR-TOLES | 1 | 1265 | 5000 | 24810 | -82 | 499 | 11897 |
| 1 | 610 | 1 | 1 | 1200 | 0190/73 | FR-TOLES | 1 | 1265 | 3400 | 13970 | -82 | 413 | 11826 |
| 2 | 610 | 1 | 1 | 1000 | 0190/74 | FR-TOLES | 1 | 1015 | 5000 | 19090 | -82 | 478 | 20469 |
| 2 | 610 | 1 | 1 | 1000 | 0190/76 | FR-TOLES | 1 | 1015 | 2500 | 22635 | -82 | 1136 | 11683 |
| 5 | 610 | 1 | 1 | 1200 | 0247/02 | FR-TOLES | 1 | 1230 | 2000 | 22302 | -12 | 1154 | 110694 |
| 1 | 610 | 1 | 1 | 1500 | 0263/01 | FR-TOLES | 1 | 1500 | 3440 | 31167 | -19 | 769 | 33797 |
| 1 | 610 | 1 | 1 | 1200 | 0264/05 | FR-TOLES | 1 | 1250 | 3450 | 18780 | 10 | 554 | 114130 |
| 2 | 610 | 1 | 0 | 0900 | 0400/36 | GALVALANGE1 | | 913 | 1750 | 17439 | -110 | 1390 | 28379 |
| 1 | 610 | 1 | 1 | 0700 | 0414/12 | GALVALANGE1 | | 740 | 2050 | 14360 | -47 | 1205 | 114005 |
| 1 | 610 | 1 | 1 | 1000 | 0419/10 | GALVALANGE1 | | 1003 | 2300 | 7530 | 0 | 415 | 11794 |
| 3 | 610 | 1 | 1 | 0700 | 0419/20 | GALVALANGE1 | | 752 | 2000 | 12386 | -40 | 1048 | 110692 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 610 | 1 | 1 | 0700 | 0421/16 | GALVALANGE1 | | 740 | 1850 | 14665 | -33 | 1364 | 114008 |
| 2 | 610 | 1 | 1 | 1000 | 0422/01 | GALVALANGE1 | | 1082 | 2000 | 16845 | -26 | 991 | 110572 |
| 1 | 610 | 1 | 1 | 1000 | 0422/02 | GALVALANGE1 | | 1020 | 2000 | 16450 | -26 | 1027 | 110691 |
| 3 | 610 | 1 | 0 | 0900 | 0422/09 | GALVALANGE1 | | 917 | 1600 | 14863 | -26 | 1290 | 110639 |
| 4 | 610 | 1 | 0 | 0900 | 0422/10 | GALVALANGE1 | | 917 | 1750 | 14757 | -26 | 1171 | 20546 |
| 5 | 610 | 1 | 1 | 1200 | 0423/02 | GALVALANGE1 | | 1283 | 2000 | 20686 | -26 | 1026 | 110725 |
| 1 | 610 | 1 | 0 | 1200 | 0423/04 | GALVALANGE1 | | 1223 | 1750 | 12890 | -26 | 767 | 110588 |
| 3 | 610 | 1 | 1 | 1000 | 0423/06 | GALVALANGE1 | | 1003 | 2000 | 20926 | -26 | 1328 | 110587 |
| 2 | 610 | 1 | 1 | 1200 | 0424/06 | GALVALANGE1 | | 1243 | 2000 | 19685 | -19 | 1008 | 110687 |
| 5 | 610 | 1 | 0 | 1200 | 0424/07 | GALVALANGE1 | | 1223 | 1800 | 19622 | -19 | 1135 | 110664 |
| 9 | 610 | 1 | 1 | 1200 | 0424/08 | GALVALANGE1 | | 1223 | 2000 | 23885 | -19 | 1243 | 110668 |
| 1 | 610 | 1 | 1 | 1000 | 0424/11 | GALVALANGE1 | | 1003 | 2000 | 15720 | -19 | 998 | 110601 |
| 2 | 610 | 1 | 0 | 0900 | 0424/12 | GALVALANGE1 | | 928 | 1800 | 15130 | -19 | 1153 | 110603 |
| 7 | 610 | 1 | 0 | 0900 | 0424/13 | GALVALANGE1 | | 928 | 1800 | 14298 | -19 | 1090 | 110688 |
| 1 | 610 | 1 | 1 | 1200 | 0425/01 | GALVALANGE1 | | 1253 | 2040 | 24867 | -19 | 1239 | 113804 |
| 6 | 610 | 1 | 1 | 0700 | 0425/14 | GALVALANGE1 | | 740 | 1850 | 13936 | -19 | 1296 | 114079 |
| 5 | 610 | 1 | 0 | 0900 | 0426/10 | GALVALANGE1 | | 917 | 1800 | 17151 | -19 | 1323 | 33624 |
| 1 | 610 | 1 | 1 | 1200 | 0428/02 | GALVALANGE1 | | 1253 | 2800 | 20330 | 3 | 738 | 110679 |
| 2 | 610 | 1 | 1 | 1200 | 0428/03 | GALVALANGE1 | | 1253 | 2000 | 18150 | 3 | 922 | 110714 |
| 0 | 610 | 1 | 1 | 1200 | 0428/04 | GALVALANGE1 | | 1253 | 2000 | 21202 | 3 | 1077 | 110719 |
| 7 | 610 | 1 | 1 | 1200 | 0428/05 | GALVALANGE1 | | 1253 | 2000 | 22721 | 3 | 1154 | 110722 |
| 1 | 610 | 1 | 1 | 1000 | 0428/15 | GALVALANGE1 | | 1059 | 2000 | 16510 | 3 | 993 | 110670 |
| 2 | 610 | 1 | 1 | 1000 | 0428/16 | GALVALANGE1 | | 1003 | 2000 | 12155 | 3 | 771 | 110697 |
| 3 | 610 | 1 | 1 | 1000 | 0428/17 | GALVALANGE1 | | 1003 | 2000 | 15430 | 3 | 979 | 110704 |
| 2 | 610 | 1 | 1 | 1000 | 0428/18 | GALVALANGE1 | | 1003 | 2000 | 15635 | 3 | 992 | 110724 |
| 2 | 610 | 1 | 1 | 1000 | 0428/19 | GALVALANGE1 | | 1003 | 2000 | 15730 | 3 | 998 | 110707 |
| 2 | 610 | 1 | 1 | 1000 | 0428/20 | GALVALANGE1 | | 1003 | 2000 | 15605 | 3 | 990 | 110700 |
| 5 | 610 | 1 | 1 | 1000 | 0428/21 | GALVALANGE1 | | 1003 | 2000 | 15586 | 3 | 989 | 110710 |
| 6 | 610 | 1 | 1 | 1000 | 0428/22 | GALVALANGE1 | | 1003 | 2000 | 18591 | 3 | 1180 | 110712 |
| 2 | 610 | 1 | 0 | 1000 | 0428/24 | GALVALANGE1 | | 1003 | 1750 | 15370 | 3 | 1115 | 110662 |
| 1 | 610 | 1 | 1 | 0800 | 0642/12 | FR-TOLES | 1 | 850 | 4060 | 15150 | -369 | 559 | 18648 |
| 1 | 610 | 1 | 1 | 1500 | 0690/03 | FR-TOLES | 1 | 1515 | 3460 | 28870 | 0 | 701 | 15540 |
| 2 | 610 | 1 | 1 | 1000 | 0761/05 | FR-TOLES | 1 | 1000 | 3145 | 24080 | 0 | 975 | 26320 |
| 1 | 610 | 1 | 1 | 1500 | 0765/03 | FR-TOLES | 1 | 1515 | 4040 | 29155 | 0 | 606 | 28196 |
| 1 | 610 | 1 | 1 | 1200 | 0767/24 | FR-TOLES | 1 | 1250 | 3430 | 12311 | 0 | 365 | 28190 |
| 1 | 610 | 1 | 1 | 1000 | 0782/18 | FR-TOLES | 1 | 1015 | 2850 | 13371 | 0 | 588 | 29608 |
| 1 | 610 | 1 | 1 | 1000 | 0787/09 | FR-TOLES | 1 | 1015 | 5040 | 16289 | 0 | 405 | 29133 |
| 1 | 610 | 1 | 1 | 1000 | 0789/14 | FR-TOLES | 1 | 1015 | 3500 | 25670 | -82 | 920 | 22212 |
| 1 | 610 | 1 | 1 | 1000 | 0789/15 | FR-TOLES | 1 | 1015 | 2850 | 19643 | 0 | 865 | 29248 |
| 1 | 610 | 1 | 1 | 1000 | 0789/16 | FR-TOLES | 1 | 1015 | 2550 | 20167 | 0 | 992 | 29254 |
| 1 | 610 | 1 | 1 | 1200 | 0791/04 | FR-TOLES | 1 | 1237 | 2750 | 29176 | -82 | 1092 | 29401 |
| 1 | 610 | 1 | 1 | 0800 | 0791/09 | FR-TOLES | 1 | 835 | 2740 | 10601 | -82 | 590 | 22278 |
| 1 | 610 | 1 | 1 | 1000 | 0799/30 | FR-TOLES | 1 | 1015 | 2870 | 13098 | -40 | 572 | 32846 |
| 1 | 610 | 1 | 1 | 1500 | 0823/04 | FR-TOLES | 1 | 1515 | 3140 | 29685 | -47 | 794 | 22632 |
| 1 | 610 | 1 | 1 | 1500 | 0823/06 | FR-TOLES | 1 | 1515 | 2550 | 29961 | -47 | 987 | 22736 |
| 1 | 610 | 1 | 1 | 1200 | 0828/09 | FR-TOLES | 1 | 1250 | 3840 | 19020 | -40 | 504 | 114129 |
| 1 | 610 | 1 | 1 | 1500 | 0830/04 | FR-TOLES | 1 | 1515 | 3450 | 29980 | -40 | 730 | 23232 |
| 2 | 610 | 1 | 1 | 1200 | 0830/10 | FR-TOLES | 1 | 1265 | 2550 | 16568 | -40 | 654 | 32986 |
| 1 | 610 | 1 | 1 | 1200 | 0831/07 | FR-TOLES | 1 | 1235 | 1860 | 20626 | -40 | 1143 | 114159 |
| 1 | 610 | 1 | 1 | 0800 | 0834/22 | FR-TOLES | 1 | 855 | 5330 | 10545 | -33 | 294 | 33435 |
| 2 | 610 | 1 | 1 | 1200 | 0838/03 | FR-TOLES | 1 | 1225 | 3855 | 23950 | -26 | 645 | 114126 |
| 4 | 610 | 1 | 1 | 0900 | 0838/14 | FR-TOLES | 1 | 925 | 2742 | 23140 | -26 | 1161 | 114031 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 610 | 1 | 1 | 1200 | 0844/06 | FR-TOLES | 1 | 1265 | 2550 | 10180 | -4 | 402 | 113954 |
| 1 | 610 | 1 | 1 | 1000 | 0844/14 | FR-TOLES | 1 | 1015 | 3410 | 25124 | -4 | 924 | 114158 |
| 1 | 610 | 1 | 1 | 1000 | 0844/15 | FR-TOLES | 1 | 1015 | 3630 | 19486 | -4 | 673 | 114164 |
| 1 | 610 | 1 | 1 | 1200 | 0845/06 | FR-TOLES | 1 | 1265 | 2550 | 24920 | -4 | 984 | 113976 |
| 1 | 610 | 1 | 1 | 1100 | 0845/08 | FR-TOLES | 1 | 1125 | 3150 | 14129 | -4 | 507 | 114161 |
| 1 | 610 | 1 | 1 | 0900 | 0845/15 | FR-TOLES | 1 | 915 | 2440 | 11360 | -4 | 648 | 114076 |
| 1 | 610 | 1 | 1 | 1200 | 0845/17 | FR-TOLES | 1 | 1265 | 2450 | 10030 | -4 | 412 | 113951 |
| 1 | 610 | 1 | 1 | 1300 | 0846/04 | FR-TOLES | 1 | 1305 | 3160 | 16224 | 3 | 501 | 114156 |
| 1 | 610 | 1 | 1 | 1000 | 0846/15 | FR-TOLES | 1 | 1015 | 3450 | 19500 | 3 | 709 | 114117 |
| 3 | 610 | 1 | 1 | 1500 | 0847/01 | FR-TOLES | 1 | 1520 | 4046 | 28483 | 3 | 589 | 114142 |
| 7 | 610 | 1 | 1 | 1200 | 0847/02 | FR-TOLES | 1 | 1270 | 4048 | 24652 | 3 | 610 | 114144 |
| 2 | 610 | 1 | 1 | 1200 | 0847/05 | FR-TOLES | 1 | 1233 | 3440 | 23625 | 3 | 709 | 114145 |
| 1 | 610 | 1 | 1 | 0900 | 0847/10 | FR-TOLES | 1 | 927 | 2560 | 13525 | 3 | 726 | 114150 |
| 2 | 610 | 1 | 1 | 1000 | 0849/21 | FR-TOLES | 1 | 1015 | 3050 | 15310 | 3 | 629 | 114110 |
| 1 | 610 | 1 | 1 | 1200 | 0849/26 | FR-TOLES | 1 | 1280 | 2040 | 25464 | 3 | 1242 | 114163 |
| 5 | 610 | 1 | 1 | 1500 | 0850/02 | FR-TOLES | 1 | 1500 | 3440 | 18815 | 3 | 464 | 114152 |
| 1 | 610 | 1 | 1 | 1500 | 0850/03 | FR-TOLES | 1 | 1500 | 2540 | 18898 | 3 | 631 | 114160 |
| 1 | 610 | 1 | 1 | 1500 | 0851/04 | FR-TOLES | 1 | 1515 | 2550 | 29583 | 3 | 975 | 114154 |
| 5 | 610 | 1 | 1 | 1500 | 0855/04 | FR-TOLES | 1 | 1512 | 2038 | 18370 | 10 | 758 | 114153 |
| 1 | 610 | 1 | 0 | 1000 | 1047/11 | GALVA-ADU | 1 | 1002 | 1790 | 9640 | -4 | 684 | 113942 |
| 1 | 610 | 1 | 1 | 1000 | 1367/05 | GALVA-ADU | 1 | 1001 | 2150 | 16873 | -33 | 998 | 114162 |
| 1 | 610 | 1 | 1 | 1000 | 1373/03 | GALVA-ADU | 1 | 1000 | 2050 | 7429 | -19 | 461 | 33488 |
| 3 | 610 | 1 | 1 | 1000 | 1376/01 | GALVA-ADU | 1 | 1001 | 2140 | 19653 | -4 | 1168 | 114016 |
| 2 | 610 | 1 | 1 | 1000 | 1376/02 | GALVA-ADU | 1 | 1001 | 2050 | 19335 | -4 | 1200 | 113968 |
| 1 | 610 | 1 | 1 | 1000 | 1376/04 | GALVA-ADU | 1 | 1001 | 2040 | 14870 | -4 | 927 | 113946 |
| 2 | 610 | 1 | 1 | 1000 | 1376/05 | GALVA-ADU | 1 | 1001 | 2040 | 25215 | -4 | 1572 | 113993 |
| 1 | 610 | 1 | 1 | 1000 | 1376/06 | GALVA-ADU | 1 | 1002 | 2050 | 15230 | -4 | 944 | 113961 |
| 1 | 610 | 1 | 1 | 1000 | 1376/07 | GALVA-ADU | 1 | 1002 | 2050 | 26210 | -4 | 1625 | 113927 |
| 1 | 610 | 1 | 1 | 1000 | 1376/08 | GALVA-ADU | 1 | 1002 | 2040 | 9610 | -4 | 598 | 113989 |
| 2 | 610 | 1 | 1 | 1000 | 1376/09 | GALVA-ADU | 1 | 1002 | 2045 | 12685 | -4 | 788 | 113964 |
| 1 | 610 | 1 | 1 | 0900 | 1376/10 | GALVA-ADU | 1 | 987 | 2550 | 18090 | -4 | 915 | 114029 |
| 2 | 610 | 1 | 1 | 0800 | 1377/05 | GALVA-ADU | 1 | 888 | 1840 | 16255 | -4 | 1267 | 113960 |
| 2 | 610 | 1 | 1 | 0800 | 1377/06 | GALVA-ADU | 1 | 888 | 2040 | 8015 | -4 | 563 | 113978 |
| 1 | 610 | 1 | 1 | 1000 | 1378/03 | GALVA-ADU | 1 | 1001 | 2050 | 19020 | 3 | 1180 | 114041 |
| 3 | 610 | 1 | 1 | 1000 | 1378/08 | GALVA-ADU | 1 | 1002 | 2043 | 18920 | 3 | 1176 | 114044 |
| 5 | 610 | 1 | 1 | 1100 | 2224/15 | FEUILLARDS1 | | 1135 | 3060 | 22312 | -4 | 818 | 114083 |
| 1 | 610 | 1 | 1 | 1200 | 2227/01 | FEUILLARDS1 | | 1250 | 2050 | 19160 | -124 | 952 | 17296 |
| 4 | 610 | 1 | 1 | 1200 | 2227/04 | FEUILLARDS1 | | 1250 | 2555 | 11509 | -117 | 458 | 29151 |
| 2 | 610 | 1 | 1 | 1200 | 2227/10 | FEUILLARDS1 | | 1230 | 2605 | 10468 | -82 | 416 | 29946 |
| 4 | 610 | 1 | 1 | 1200 | 2227/18 | FEUILLARDS1 | | 1250 | 2045 | 12218 | -54 | 608 | 22361 |
| 8 | 610 | 1 | 1 | 1200 | 2227/24 | FEUILLARDS1 | | 1270 | 2041 | 12505 | -47 | 614 | 32789 |
| 3 | 610 | 1 | 1 | 1200 | 2227/32 | FEUILLARDS1 | | 1220 | 3850 | 23944 | -19 | 649 | 114146 |
| 3 | 610 | 1 | 1 | 1200 | 2227/33 | FEUILLARDS1 | | 1230 | 2643 | 23633 | -19 | 925 | 114018 |
| 4 | 610 | 1 | 1 | 1200 | 2227/96 | FEUILLARDS1 | | 1230 | 2600 | 8292 | -166 | 329 | 10971 |
| 2 | 610 | 1 | 1 | 1100 | 3359/14 | FEUILLARDS1 | | 1135 | 2570 | 22320 | -26 | 974 | 114090 |
| 1 | 610 | 1 | 1 | 0900 | 3773/07 | FEUILLARDS1 | | 985 | 2250 | 14350 | -4 | 824 | 114067 |
| 1 | 610 | 1 | 1 | 0900 | 3773/09 | FEUILLARDS1 | | 980 | 3120 | 18440 | -4 | 768 | 114066 |
| 2 | 610 | 1 | 1 | 0900 | 3773/11 | FEUILLARDS1 | | 945 | 2450 | 17833 | -4 | 981 | 114155 |
| 5 | 610 | 1 | 1 | 0900 | 3773/12 | FEUILLARDS1 | | 945 | 2450 | 17846 | -4 | 981 | 114093 |
| 2 | 610 | 1 | 1 | 0900 | 3773/13 | FEUILLARDS1 | | 920 | 2550 | 16515 | -4 | 896 | 114094 |
| 2 | 610 | 1 | 1 | 1000 | 3777/01 | FEUILLARDS1 | | 1070 | 2745 | 20485 | -4 | 888 | 114082 |

## ANNEXE 2

Exemple de résultats de sélection:

le premier nombre de chaque ligne indique le nombre de coils sélectionnés pour chaque famille.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0/ 7 | 610 | 1 | 1 | 1500 | 0063/01 | FR-TOLES | 1 | 1500 | 3431 | 18307 | 10 | 452 | 114138 |
| 1/ 1 | 610 | 1 | 1 | 0900 | 0100/05 | FR-TOLES | 1 | 903 | 2030 | 13920 | - 69 | 967 | 114157 |
| 1/ 1 | 610 | 1 | 1 | 1100 | 0150/03 | FR-TOLES | 1 | 1120 | 1840 | 16563 | - 71 | 1023 | 15413 |
| 2/ 2 | 610 | 1 | 1 | 1500 | 0190/54 | FR-TOLES | 1 | 1515 | 2545 | 29763 | 96 | 982 | 28488 |
| 3/ 3 | 610 | 1 | 1 | 1200 | 0190/55 | FR-TOLES | 1 | 1265 | 5040 | 25134 | 96 | 501 | 28738 |
| 1/ 1 | 610 | 1 | 1 | 1200 | 0190/58 | FR-TOLES | 1 | 1265 | 3440 | 25110 | 96 | 735 | 28521 |
| 1/ 1 | 610 | 1 | 1 | 1200 | 0190/60 | FR-TOLES | 1 | 1265 | 2550 | 25135 | 96 | 992 | 28501 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 0190/61 | FR-TOLES | 1 | 1015 | 5030 | 12500 | 96 | 311 | 28829 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 0190/62 | FR-TOLES | 1 | 1015 | 4550 | 24270 | 96 | 669 | 28566 |
| 2/ 2 | 610 | 1 | 1 | 1000 | 0190/65 | FR-TOLES | 1 | 1015 | 2835 | 24550 | 96 | 1086 | 28709 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 0190/66 | FR-TOLES | 1 | 1015 | 2560 | 24745 | 96 | 1213 | 28714 |
| 2/ 2 | 610 | 1 | 1 | 1500 | 0190/68 | FR-TOLES | 1 | 1515 | 5000 | 26825 | 82 | 451 | 11713 |
| 3/ 3 | 610 | 1 | 1 | 1500 | 0190/69 | FR-TOLES | 1 | 1515 | 4000 | 25980 | 82 | 545 | 11654 |
| 3/ 3 | 610 | 1 | 1 | 1500 | 0190/70 | FR-TOLES | 1 | 1515 | 2500 | 24653 | 82 | 829 | 11625 |
| 2/ 2 | 610 | 1 | 1 | 1200 | 0190/71 | FR-TOLES | 1 | 1265 | 5000 | 24810 | 82 | 499 | 11897 |
| 1/ 1 | 610 | 1 | 1 | 1200 | 0190/73 | FR-TOLES | 1 | 1265 | 3400 | 13970 | 82 | 413 | 11826 |
| 2/ 2 | 610 | 1 | 1 | 1000 | 0190/74 | FR-TOLES | 1 | 1015 | 5000 | 19090 | 82 | 478 | 20469 |
| 2/ 2 | 610 | 1 | 1 | 1000 | 0190/76 | FR-TOLES | 1 | 1015 | 2500 | 22635 | 82 | 1136 | 11683 |
| 0/ 5 | 610 | 1 | 1 | 1200 | 0247/02 | FR-TOLES | 1 | 1230 | 2000 | 22302 | 12 | 1154 | 110694 |
| 0/ 1 | 610 | 1 | 1 | 1500 | 0263/01 | FR-TOLES | 1 | 1500 | 3440 | 31167 | 19 | 769 | 33797 |
| 0/ 1 | 610 | 1 | 1 | 1200 | 0264/05 | FR-TOLES | 1 | 1250 | 3450 | 18780 | 10 | 554 | 114130 |
| 2/ 2 | 610 | 1 | 0 | 0900 | 0400/36 | GALVALANG1 | | 913 | 1750 | 17439 | - 10 | 1390 | 28379 |
| 1/ 1 | 610 | 1 | 1 | 0700 | 0414/12 | GALVALANG1 | | 740 | 2050 | 14360 | 47 | 1205 | 114005 |
| 0/ 1 | 610 | 1 | 1 | 1000 | 0419/10 | GALVALANG1 | | 1003 | 2300 | 7530 | 0 | 415 | 11794 |
| 3/ 3 | 610 | 1 | 1 | 0700 | 0419/20 | GALVALANG1 | | 752 | 2000 | 12386 | 40 | 1048 | 110692 |
| 2/ 2 | 610 | 1 | 1 | 0700 | 0421/16 | GALVALANG1 | | 740 | 1850 | 14665 | 33 | 1364 | 114008 |
| 2/ 2 | 610 | 1 | 1 | 1000 | 0422/01 | GALVALANG1 | | 1082 | 2000 | 16845 | 26 | 991 | 110572 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 0422/02 | GALVALANG1 | | 1020 | 2000 | 16450 | 26 | 1027 | 110691 |
| 3/ 3 | 610 | 1 | 0 | 0900 | 0422/09 | GALVALANG1 | | 917 | 1600 | 14863 | 26 | 1290 | 110639 |
| 4/ 4 | 610 | 1 | 0 | 0900 | 0422/10 | GALVALAN | 1 | 917 | 1750 | 14757 | -26 | 1171 | 20546 |
| 3/ 5 | 610 | 1 | 1 | 1200 | 0423/02 | GALVALANG1 | | 1283 | 2000 | 20686 | 26 | 1026 | 110725 |
| 1/ 1 | 610 | 1 | 0 | 1200 | 0423/04 | GALVALANG1 | | 1223 | 1750 | 12890 | 26 | 767 | 110588 |
| 0/ 3 | 610 | 1 | 1 | 1000 | 0423/06 | GALVALANG1 | | 1003 | 2000 | 20926 | 26 | 1328 | 110587 |
| 0/ 2 | 610 | 1 | 1 | 1200 | 0424/06 | GALVALANG1 | | 1243 | 2000 | 19685 | 19 | 1008 | 110687 |
| 0/ 5 | 610 | 1 | 0 | 1200 | 0424/07 | GALVALANG1 | | 1223 | 1800 | 19622 | 19 | 1135 | 110664 |
| 0/ 9 | 610 | 1 | 1 | 1200 | 0424/08 | GALVALANG1 | | 1223 | 2000 | 23885 | 19 | 1243 | 110668 |
| 0/ 1 | 610 | 1 | 1 | 1000 | 0424/11 | GALVALANG1 | | 1003 | 2000 | 15720 | 19 | 998 | 110601 |
| 0/ 2 | 610 | 1 | 0 | 0900 | 0424/12 | GALVALANG1 | | 928 | 1800 | 15130 | 19 | 1153 | 110603 |
| 7/ 7 | 610 | 1 | 0 | 0900 | 0424/13 | GALVALANG1 | | 928 | 1800 | 14298 | 19 | 1090 | 110688 |
| 0/ 1 | 610 | 1 | 1 | 1200 | 0425/01 | GALVALANG1 | | 1253 | 2040 | 24867 | 19 | 1239 | 113804 |
| 6/ 6 | 610 | 1 | 1 | 0700 | 0425/14 | GALVALANG1 | | 740 | 1850 | 13936 | 19 | 1296 | 114079 |
| 0/ 5 | 610 | 1 | 0 | 0900 | 0426/10 | GALVALANG1 | | 917 | 1800 | 17151 | 19 | 1323 | 33624 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0/ 1 | 610 | 1 | 1 | 1200 | 0428/02 | GALVALANG1 | 1253 | 2800 | 20330 | 3 | 738 | 110679 |
| 0/ 2 | 610 | 1 | 1 | 1200 | 0428/03 | GALVALANG1 | 1253 | 2000 | 18150 | 3 | 922 | 110714 |
| 0/10 | 610 | 1 | 1 | 1200 | 0428/04 | GALVALANG1 | 1253 | 2000 | 21202 | 3 | 1077 | 110719 |
| 0/ 7 | 610 | 1 | 1 | 1200 | 0428/05 | GALVALANG1 | 1253 | 2000 | 22721 | 3 | 1154 | 110722 |
| 0/ 1 | 610 | 1 | 1 | 1000 | 0428/15 | GALVALANG1 | 1059 | 2000 | 16510 | 3 | 993 | 110670 |
| 0/ 2 | 610 | 1 | 1 | 1000 | 0428/16 | GALVALANG1 | 1003 | 2000 | 12155 | 3 | 771 | 110697 |
| 0/ 3 | 610 | 1 | 1 | 1000 | 0428/17 | GALVALANG1 | 1003 | 2000 | 15430 | 3 | 979 | 110704 |
| 0/ 2 | 610 | 1 | 1 | 1000 | 0428/18 | GALVALANG1 | 1003 | 2000 | 15635 | 3 | 992 | 110724 |
| 0/ 2 | 610 | 1 | 1 | 1000 | 0428/19 | GALVALANG1 | 1003 | 2000 | 15730 | 3 | 998 | 110707 |
| 0/ 2 | 610 | 1 | 1 | 1000 | 0428/20 | GALVALANG1 | 1003 | 2000 | 15605 | 3 | 990 | 110700 |
| 0/ 5 | 610 | 1 | 1 | 1000 | 0428/21 | GALVALANG1 | 1003 | 2000 | 15586 | 3 | 989 | 110710 |
| 0/ 6 | 610 | 1 | 1 | 1000 | 0428/22 | GALVALANG1 | 1003 | 2000 | 18591 | 3 | 1180 | 110712 |
| 0/ 2 | 610 | 1 | 0 | 1000 | 0428/24 | GALVALANG1 | 1003 | 1750 | 15370 | 3 | 1115 | 110662 |
| 1/ 1 | 610 | 1 | 1 | 0800 | 0642/12 | FR-TOLES 1 | 850 | 4060 | 15150 | - 69 | 559 | 18648 |
| 0/ 1 | 610 | 1 | 1 | 1500 | 0690/03 | FR-TOLES 1 | 1515 | 3460 | 28870 | 0 | 701 | 15540 |
| 0/ 2 | 610 | 1 | 1 | 1000 | 0761/05 | FR-TOLES 1 | 1000 | 3145 | 24080 | 0 | 975 | 26320 |
| 0/ 1 | 610 | 1 | 1 | 1500 | 0765/03 | FR-TOLES 1 | 1515 | 4040 | 29155 | 0 | 606 | 28196 |
| 0/ 1 | 610 | 1 | 1 | 1200 | 0767/24 | FR-TOLES 1 | 1250 | 3430 | 12311 | 0 | 365 | 28190 |
| 0/ 1 | 610 | 1 | 1 | 1000 | 0782/18 | FR-TOLES 1 | 1015 | 2850 | 13371 | 0 | 588 | 29608 |
| 0/ 1 | 610 | 1 | 1 | 1000 | 0787/09 | FR-TOLES 1 | 1015 | 5040 | 16289 | 0 | 405 | 29133 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 0789/14 | FR-TOLES 1 | 1015 | 3500 | 25670 | 82 | 920 | 22212 |
| 0/ 1 | 610 | 1 | 1 | 1000 | 0789/15 | FR-TOLES 1 | 1015 | 2850 | 19643 | 0 | 865 | 29248 |
| 0/ 1 | 610 | 1 | 1 | 1000 | 0789/16 | FR-TOLES 1 | 1015 | 2550 | 20167 | 0 | 992 | 29254 |
| 1/ 1 | 610 | 1 | 1 | 1200 | 0791/04 | FR-TOLES 1 | 1237 | 2750 | 29176 | 82 | 1092 | 29401 |
| 1/ 1 | 610 | 1 | 1 | 0800 | 0791/09 | FR-TOLES 1 | 835 | 2740 | 10601 | -82 | 590 | 22278 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 0799/30 | FR-TOLES 1 | 1015 | 2870 | 13098 | 40 | 572 | 32846 |
| 1/ 1 | 610 | 1 | 1 | 1500 | 0823/04 | FR-TOLES 1 | 1515 | 3140 | 29685 | 47 | 794 | 22632 |
| 1/ 1 | 610 | 1 | 1 | 1500 | 0823/06 | FR-TOLES 1 | 1515 | 2550 | 29961 | 47 | 987 | 22736 |
| 1/ 1 | 610 | 1 | 1 | 1200 | 0828/09 | FR-TOLES 1 | 1250 | 3840 | 19020 | 40 | 504 | 114129 |
| 1/ 1 | 610 | 1 | 1 | 1500 | 0830/04 | FR-TOLES 1 | 1515 | 3450 | 29980 | 40 | 730 | 23232 |
| 2/ 2 | 610 | 1 | 1 | 1200 | 0830/10 | FR-TOLES 1 | 1265 | 2550 | 16568 | 40 | 654 | 32986 |
| 1/ 1 | 610 | 1 | 1 | 1200 | 0831/07 | FR-TOLES 1 | 1235 | 1860 | 20626 | 40 | 1143 | 114159 |
| 1/ 1 | 610 | 1 | 1 | 0800 | 0834/22 | FR-TOLES 1 | 855 | 5330 | 10545 | 33 | 294 | 33435 |
| 1/ 2 | 610 | 1 | 1 | 1200 | 0838/03 | FR-TOLES 1 | 1225 | 3855 | 23950 | 26 | 645 | 114126 |
| 4/ 4 | 610 | 1 | 1 | 0900 | 0838/14 | FR-TOLES 1 | 925 | 2742 | 23140 | 26 | 1161 | 114031 |
| 0/ 1 | 610 | 1 | 1 | 1200 | 0844/06 | FR-TOLES 1 | 1265 | 2550 | 10180 | -4 | 402 | 113954 |
| 0/ 1 | 610 | 1 | 1 | 1000 | 0844/14 | FR-TOLES 1 | 1015 | 3410 | 25124 | -4 | 924 | 114158 |
| 0/ 1 | 610 | 1 | 1 | 1000 | 0844/15 | FR-TOLES 1 | 1015 | 3630 | 19486 | -4 | 673 | 114164 |
| 0/ 1 | 610 | 1 | 1 | 1200 | 0845/06 | FR-TOLES 1 | 1265 | 2550 | 24920 | -4 | 984 | 113976 |
| 0/ 1 | 610 | 1 | 1 | 1100 | 0845/08 | FR-TOLES 1 | 1125 | 3150 | 14129 | -4 | 507 | 114161 |
| 0/ 1 | 610 | 1 | 1 | 0900 | 0845/15 | FR-TOLES 1 | 915 | 2440 | 11360 | -4 | 648 | 114076 |
| 0/ 1 | 610 | 1 | 1 | 1200 | 0845/17 | FR-TOLES 1 | 1265 | 2450 | 10030 | -4 | 412 | 113951 |
| 0/ 1 | 610 | 1 | 1 | 1300 | 0846/04 | FR-TOLES 1 | 1305 | 3160 | 16224 | 3 | 501 | 114156 |
| 0/ 1 | 610 | 1 | 1 | 1000 | 0846/15 | FR-TOLES 1 | 1015 | 3450 | 19500 | 3 | 709 | 114117 |
| 0/ 3 | 610 | 1 | 1 | 1500 | 0847/01 | FR-TOLES 1 | 1520 | 4046 | 28483 | 3 | 589 | 114142 |
| 0/ 7 | 610 | 1 | 1 | 1200 | 0847/02 | FR-TOLES 1 | 1270 | 4048 | 24652 | 3 | 610 | 114144 |
| 0/ 2 | 610 | 1 | 1 | 1200 | 0847/05 | FR-TOLES 1 | 1233 | 3440 | 23625 | 3 | 709 | 114145 |
| 0/ 1 | 610 | 1 | 1 | 0900 | 0847/10 | FR-TOLES 1 | 927 | 2560 | 13525 | 3 | 726 | 114150 |
| 0/ 2 | 610 | 1 | 1 | 1000 | 0849/21 | FR-TOLES 1 | 1015 | 3050 | 15310 | 3 | 629 | 114110 |
| 0/ 1 | 610 | 1 | 1 | 1200 | 0849/26 | FR-TOLES 1 | 1280 | 2040 | 25464 | 3 | 1242 | 114163 |
| 0/ 5 | 610 | 1 | 1 | 1500 | 0850/02 | FR-TOLES 1 | 1500 | 3440 | 18815 | 3 | 464 | 114152 |
| 0/ 1 | 610 | 1 | 1 | 1500 | 0850/03 | FR-TOLES 1 | 1500 | 2540 | 18898 | 3 | 631 | 114160 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0/ 1 | 610 | 1 | 1 | 1500 | 0851/04 | FR-TOLES 1 | 1515 | 2550 | 29583 | 3 | 975 | 114154 |
| 0/ 5 | 610 | 1 | 1 | 1500 | 0855/04 | FR-TOLES 1 | 1512 | 2038 | 18370 | 10 | 758 | 114153 |
| 1/ 1 | 610 | 1 | 0 | 1000 | 1047/11 | GALVA-ADU1 | 1002 | 1790 | 9640 | -4 | 684 | 113942 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 1367/05 | GALVA-ADU1 | 1001 | 2150 | 16873 | 33 | 998 | 114162 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 1373/03 | GALVA-ADU1 | 1000 | 2050 | 7429 | 19 | 461 | 33488 |
| 3/ 3 | 610 | 1 | 1 | 1000 | 1376/01 | GALVA-ADU1 | 1001 | 2140 | 19653 | -4 | 1168 | 114016 |
| 2/ 2 | 610 | 1 | 1 | 1000 | 1376/02 | GALVA-ADU1 | 1001 | 2050 | 19335 | -4 | 1200 | 113968 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 1376/04 | GALVA-ADU1 | 1001 | 2040 | 14870 | -4 | 927 | 113946 |
| 2/ 2 | 610 | 1 | 1 | 1000 | 1376/05 | GALVA-ADU1 | 1001 | 2040 | 25215 | -4 | 1572 | 113993 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 1376/06 | GALVA-AD 1 | 1002 | 2050 | 15230 | -4 | 944 | 113961 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 1376/07 | GALVA-ADU1 | 1002 | 2050 | 26210 | -4 | 1625 | 113927 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 1376/08 | GALVA-ADU1 | 1002 | 2040 | 9610 | -4 | 598 | 113989 |
| 2/ 2 | 610 | 1 | 1 | 1000 | 1376/09 | GALVA-ADU1 | 1002 | 2045 | 12685 | -4 | 788 | 113964 |
| 1/ 1 | 610 | 1 | 1 | 0900 | 1376/10 | GALVA-ADU1 | 987 | 2550 | 18090 | -4 | 915 | 114029 |
| 2/ 2 | 610 | 1 | 1 | 0800 | 1377/05 | GALVA-ADU1 | 888 | 1840 | 16255 | -4 | 1267 | 113960 |
| 2/ 2 | 610 | 1 | 1 | 0800 | 1377/06 | GALVA-ADU1 | 888 | 2040 | 8015 | -4 | 563 | 113978 |
| 1/ 1 | 610 | 1 | 1 | 1000 | 1378/03 | GALVA-ADU1 | 1001 | 2050 | 19020 | 3 | 1180 | 114041 |
| 0/ 3 | 610 | 1 | 1 | 1000 | 1378/08 | GALVA-ADU1 | 1002 | 2043 | 18920 | 3 | 1176 | 114044 |
| 0/ 5 | 610 | 1 | 1 | 1100 | 2224/15 | FEUILLARD1 | 1135 | 3060 | 22312 | -4 | 818 | 114083 |
| 1/ 1 | 610 | 1 | 1 | 1200 | 2227/01 | FEUILLARD1 | 1250 | 2050 | 19160 | - 24 | 952 | 17296 |
| 4/ 4 | 610 | 1 | 1 | 1200 | 2227/04 | FEUILLARD1 | 1250 | 2555 | 11509 | - 17 | 458 | 29151 |
| 2/ 2 | 610 | 1 | 1 | 1200 | 2227/10 | FEUILLARD1 | 1230 | 2605 | 10468 | 82 | 416 | 29946 |
| 4/ 4 | 610 | 1 | 1 | 1200 | 2227/18 | FEUILLARD1 | 1250 | 2045 | 12218 | 54 | 608 | 22361 |
| 8/ 8 | 610 | 1 | 1 | 1200 | 2227/24 | FEUILLARD1 | 1270 | 2041 | 12505 | 47 | 614 | 32789 |
| 0/ 3 | 610 | 1 | 1 | 1200 | 2227/32 | FEUILLARD1 | 1220 | 3850 | 23944 | 19 | 649 | 114146 |
| 0/ 3 | 610 | 1 | 1 | 1200 | 2227/33 | FEUILLARD1 | 1230 | 2643 | 23633 | 19 | 925 | 114018 |
| 4/ 4 | 610 | 1 | 1 | 1200 | 2227/96 | FEUILLARD1 | 1230 | 2600 | 8292 | - 66 | 329 | 10971 |
| 0/ 2 | 610 | 1 | 1 | 1100 | 3359/14 | FEUILLARD1 | 1135 | 2570 | 22320 | 26 | 974 | 114090 |
| 0/ 1 | 610 | 1 | 1 | 0900 | 3773/07 | FEUILLARD1 | 985 | 2250 | 14350 | -4 | 824 | 114067 |
| 0/ 1 | 610 | 1 | 1 | 0900 | 3773/09 | FEUILLARD1 | 980 | 3120 | 18440 | -4 | 768 | 114066 |
| 0/ 2 | 610 | 1 | 1 | 0900 | 3773/11 | FEUILLARD1 | 945 | 2450 | 17833 | -4 | 981 | 114155 |
| 0/ 5 | 610 | 1 | 1 | 0900 | 3773/12 | FEUILLARD1 | 945 | 2450 | 17846 | -4 | 981 | 114093 |
| 0/ 2 | 610 | 1 | 1 | 0900 | 3773/13 | FEUILLARD1 | 920 | 2550 | 16515 | -4 | 896 | 114094 |
| 0/ 2 | 610 | 1 | 1 | 1000 | 3777/01 | FEUILLARD1 | 1070 | 2745 | 20485 | -4 | 888 | 114082 |

SOLUTION REELLE : % de DESTINATION

| | | | |
|---|---|---|---|
| 12.000 % de FEUILLARDS | = | 22 | 265560 |
| 47.550 % de FR-TOLES | = | 46 | 1052281 |
| 16.550 % de GALVA-ADU | = | 22 | 366251 |
| 23.900 % de GALVALANGE | = | 35 | 528907 |

SOLUTION REELLE : CLASSE DE LARGEUR

| | | | |
|---|---|---|---|
| 7.432 % de 0700 | = | 12 | 164464 |
| 3.834 % de 0800 | = | 7 | 84836 |
| 16.410 % de 0900 | = | 22 | 363151 |
| 26.326 % de 1000 | = | 31 | 582594 |
| 0.748 % de 1100 | = | 1 | 16563 |
| 29.222 % de 1200 | = | 40 | 646690 |
| 0.000 % de 1300 | = | 0 | 0 |
| 16.028 % de 1500 | = | 13 | 354701 |

nombre de coils = 126.8052

longueur totale des coils = 2213000.

solution arrondie
==========

nombre de coils = 127.0000

longueur totale des coils = 2217304.

| DESTINATION | | nb coils | tonnage |
|---|---|---|---|
| 12.096 % de FEUILLARDS | = | 23 | 268212 |
| 47.897 % de FR-TOLES | = | 47 | 1062018 |
| 16.188 % de GALVA-ADU | = | 22 | 358941 |
| 23.819 % de GALVALANGE | = | 35 | 528133 |

| CLASSE DE LARGEUR | | nb coils | tonnage |
|---|---|---|---|
| 7.417 % de 0700 | = | 12 | 164464 |
| 3.826 % de 0800 | = | 7 | 84836 |
| 16.378 % de 0900 | = | 22 | 363151 |
| 25.945 % de 1000 | = | 31 | 575284 |
| 0.747 % de 1100 | = | 1 | 16563 |
| 29.689 % de 1200 | = | 41 | 658305 |
| 0.000 % de 1300 | = | 0 | 0 |
| 15.997 % de 1500 | = | 13 | 354701 |

| DELAI | | | | nb coils | tonnage |
|---|---|---|---|---|---|
| 1.311052 | % de | -369.0000 | = | 2.000000 | 29070.00 |
| 0.7469882 | % de | -271.0000 | = | 1.000000 | 16563.00 |
| 1.495871 | % de | -166.0000 | = | 4.000000 | 33168.00 |
| 0.8641125 | % de | -124.0000 | = | 1.000000 | 19160.00 |
| 2.076215 | % de | -117.0000 | = | 4.000000 | 46036.00 |
| 1.572991 | % de | -110.0000 | = | 2.000000 | 34878.00 |
| 13.33998 | % de | -96.00000 | = | 12.00000 | 295788.0 |
| 19.79756 | % de | -82.00000 | = | 20.00000 | 438972.0 |
| 2.204118 | % de | -54.00000 | = | 4.000000 | 48872.00 |
| 7.849442 | % de | -47.00000 | = | 11.00000 | 174046.0 |
| 6.901083 | % de | -40.00000 | = | 9.000000 | 153018.0 |
| 2.559324 | % de | -33.00000 | = | 4.000000 | 56748.00 |
| 15.56913 | % de | -26.00000 | = | 19.00000 | 345215.0 |
| 8.619973 | % de | -19.00000 | = | 14.00000 | 191131.0 |
| 0.0000000 | % de | -12.00000 | = | 0.000000 | 0.0000000 |
| 14.23436 | % de | -4.000000 | = | 19.00000 | 315619.0 |
| 0.0000000 | % de | 0.0000000 | = | 0.0000000 | 0.0000000 |
| 0.8577985 | % de | 3.000000 | = | 1.000000 | 19020.00 |
| 0.0000000 | % de | 10.00000 | = | 0.0000000 | 0.0000000 |

STOCK RANGE SUIVANT LE DELAI

| | | | | | |
|---|---|---|---|---|---|
| 0.5270005 | % de | -369.0000 | = | 2.000000 | 29070.00 |
| 0.3002652 | % de | -271.0000 | = | 1.000000 | 16563.00 |
| 0.6012918 | % de | -166.0000 | = | 4.000000 | 33168.00 |
| 0.3473454 | % de | -124.0000 | = | 1.000000 | 19160.00 |
| 0.8345715 | % de | -117.0000 | = | 4.000000 | 46036.00 |
| 0.6322918 | % de | -110.0000 | = | 2.000000 | 34878.00 |
| 5.362244 | % de | -96.00000 | = | 12.00000 | 295788.0 |
| 7.957979 | % de | -82.00000 | = | 20.00000 | 438972.0 |
| 0.8859844 | % de | -54.00000 | = | 4.000000 | 48872.00 |
| 3.155223 | % de | -47.00000 | = | 11.00000 | 174046.0 |
| 2.774013 | % de | -40.00000 | = | 9.000000 | 153018.0 |
| 1.028766 | % de | -33.00000 | = | 4.000000 | 56748.00 |
| 9.389836 | % de | -26.00000 | = | 27.00000 | 517955.0 |
| 15.84584 | % de | -19.00000 | = | 46.00000 | 874076.0 |
| 2.021528 | % de | -12.00000 | = | 5.000000 | 111510.0 |
| 14.03330 | % de | -4.000000 | = | 44.00000 | 774094.0 |
| 3.544083 | % de | 0.0000000 | = | 10.00000 | 195496.0 |
| 26.42970 | % de | 3.000000 | = | 74.00000 | 1457895. |
| 4.328746 | % de | 10.00000 | = | 13.00000 | 238779.0 |

## ANNEXE 3

Pour réaliser la sélection, le module SELECTION fait appel à une routine mathématique d'optimisation.

Cette routine (H02BBF) est extraite de la libraire NAG.

Elle fournit la solution d'un programme linéaire mixte.

Ce programme doit avoir la forme suivante:

min $CVEC^tX$ sous les contraintes $B1<=AX<=B2$, $L<=X<=U$

La routine H02BBF impose d'introduire notamment:

N = nombre de variables
M = nombre de contraintes
A = matrice des coefficients des contraintes
CVEC = coefficients des variables dans la F.O.
BL et BU respectivement les bornes inférieure et supérieure des variables (L et U) et des contraintes (B1 et B2).

L'en-tête FORTRAN de la routine H02BBF est le suivant:

```
H02BBF(ITMAX,MSGLVL,N,M,A,LDA,BL,BU,INTVAR,CVEC,MAXNOD,
*      INTFST,MAXDPT,TOLIV,TOLFES,BIGBND,X,OBJMIP,
*      IWORK,LIWORK,RWORK,LRWORK,IFAIL)
```

L'ensemble de ces paramètres sont décrits dans: H02BBF _ NAG Fortran Library Manual, MARK 14, volume 8. [NAG]

Les résultats de H02BBF sont:

X: qui contient la valeur des variables après l'optimisation.

IFAIL: qui contient le code d'erreur éventuelle.