UNIVERSITE DE NANCY I — U.E.R SCIENCES MATHEMATIQUES

PROGRAMMATION LINEAIRE A VARIABLES BOOLEENNES PAR LA METHODE S.E.P.

THESE

pour l'obtention du

Doctorat de spécialité d'Informatique

soutenue le 5 décembre 1975

par

Olivier BEZAUT

Membres du Jury:

président: M. J. LEGRAS

examinateurs: M^me C. ROLLAND

M. P. BOYER

UNIVERSITE DE NANCY I U.E.R SCIENCES MATHEMATIQUES

PROGRAMMATION LINEAIRE A VARIABLES BOOLEENNES PAR LA METHODE S.E.P.

THESE

pour l'obtention du

Doctorat de spécialité d'Informatique

soutenue le 5 décembre 1975

par

Olivier BEZAUT

Membres du Jury:

président: M. J. LEGRAS

examinateurs: M^me C. ROLLAND

M. P. BOYER

Qu'il me soit permis de remercier Monsieur LEGRAS pour les conseils et encouragements qu'il n'a cessé de me prodiguer durant la poursuite de ce travail, Madame ROLLAND et Monsieur BOYER qui m'ont fait l'honneur de faire partie du jury ainsi que Monsieur STROSSER qui s'est intéressé avec sympathie à mon travail.

TABLE DES MATIERES

Introduction.

Chapitre I — Principe fondamental de la méthode des séparations et des évaluations progressives.

A. Introduction. I.1

B. Propriété séparatrice. I.2

C. Fonction d'évaluation et borne. I.4

D. Critère de progression. I.6

Chapitre II — Application de la méthode S.E.P. au cas d'un problème de programmation linéaire à variables booléennes.

A. Principe fondamental. II.2

B. Etude détaillée d'un exemple. II.8

C. Enoncé de l'algorithme. II.25

D. Programmation. II.32

Chapitre III — Algorithme de Balas.

A. Introduction. III.1

B. Terminologie. III.2

C. Enoncé de l'algorithme. III.3

D. Etude d'un exemple simple. III.13

E. Programme. III.18

Conclusion.

Bibliographie.

INTRODUCTION

Le présent travail a pour objet l'étude d'une méthode de résolution de programmes linéaires à variables booléennes.

De nombreuses recherches ont conduit à des algorithmes spécifiques, capables de résoudre certains types, plutôt que l'ensemble des problèmes de programmation linéaire, en tenant compte de la structure particulière de chaque classe de problèmes.

Un problème de programmation linéaire à variables booléennes s'énoncera sous une forme standard de la façon suivante:

$$A\,X \geqslant D$$

$$\min z = C\,X$$

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

X est un vecteur à n composantes. Ce vecteur peut prendre 2^n états appelés dans la littérature "stratégies" représentant chacun une combinaison de n valeurs 0 ou 1.

Pour résoudre de tels problèmes, on examinera une suite de stratégies. Pour chacune d'elles, on déterminera si son domaine des solutions est vide ou non. Dans ce dernier cas on déterminera la valeur de la fonction attachée à cette stratégie.

Les deux algorithmes dont nous allons faire l'étude différent par la méthode de choix des stratégies à examiner successivement, ainsi que par le critère de retour-arrière permettant de remonter dans l'arborescence pour déterminer le point de départ d'une nouvelle descendance.

Les deux premiers chapitres seront consacrés à l'étude de la méthode des séparations et évaluations progressives appelée aussi méthode "Branch and Bund" et le troisième chapitre sera l'énoncé de l'algorithme de Balas, qui présente des améliorations par rapport à l'algorithme classique.

Dans le premier chapitre nous analysons le principe fondamental de la méthode S.E.P.. Le deuxième chapitre consiste en l'application de cette méthode à des programmes linéaires spécifiques n'acceptant que des variables booléennes. Nous reprenons le principe fonda-

mental en le limitant à cette classe de problèmes. Ce paragraphe sera suivi du déroulement d'un exemple simple permettant de mettre en évidence les points essentiels de l'algorithme. Nous donnons ensuite l'énoncé de l'algorithme ainsi que l'organigramme et le programme correspondant.

Dans le troisième chapitre nous étudions l'algorithme de Balas qui diffère du précédent par un critère de choix de stratégies plus élaboré tenant compte de la fonction économique à optimiser et de la nature des différentes contraintes, ainsi que par un critère de retour-arrière permettant de limiter le nombre des calculs nécessaires à l'obtention du point de départ d'une nouvelle descendance. Nous énoncerons l'algorithme et présenterons le programme correspondant.

CHAPITRE I.

PRINCIPE FONDAMENTAL DE LA METHODE DES SEPARATIONS ET DES EVALUATIONS PROGRESSIVES

A. INTRODUCTION

Considérons le problème à résoudre. Soit Π ce problème et soit $E = \{S_1, S_2, \ldots, S_n\}$ l'ensemble dénombrable de ses solutions.

Suivant la nature du problème, on attache une fonction $f(S_j)$ à chacune de ces solutions. Nous chercherons par exemple, à minimiser la durée d'éxécution d'un ensemble de tâches dans des problèmes d'ordonnancement d'ateliers ou bien à minimiser la durée d'une tournée dans le problème du voyageur de commerce.

Ainsi le but de la résolution de Π est d'obtenir le sous-ensemble E_m des solutions correspondant au minimum de la fonction $f(S_j)$ pour un problème de minimisation et au maximum de la fonction $f(S_j)$ dans le cas d'un problème de maximisation en supposant évidemment qu'il existe un tel sous-ensemble des solutions.

Le principe de séparation dans cette méthode doit conduire à isoler des sous-ensembles de E de plus en plus réduits desquels on doit pouvoir déduire simplement une solution optimale ou l'ensemble des solutions optimales.

Les règles d'examen et de sélection ou "d'évaluations progressives" doivent permettre, de plus, après chaque séparation, de réduire le champ des investigations en éliminant le plus grand nombre possible de sous-ensembles de E inintéressants pour la suite du problème, et de choisir parmi les sous-ensembles restants celui auquel le pricipe de séparation doit être appliqué.

B. PROPRIETE SEPARATRICE

Connaissant la nature du problème soulevé, on choisit une propriété séparatrice $\mathcal{P}_A$ permettant de faire une partition de l'ensemble E de ses solutions en k sous-ensembles disjoints $A_1, A_2, \ldots, A_k$ tels que:

$$\bigcup_{i=1,k} A_i = E$$

L'idéal serait que cette propriété séparatrice puisse faire en sorte d'isoler dans un même sous-ensemble les décisions les plus satisfaisantes du sous-ensemble séparé, c'est à dire de disséminer le moins possible parmi les divers sous-ensembles engendrés, les décisions proches de l'optimum.

Ainsi pour les problèmes de programmation linéaire que l'on résolvera par la mèthode S.E.P., un sous-ensemble A_i obtenu par séparation pourra être de trois sortes:

- soit vide, s'il ne contiend aucun élément, c'est à dire aucune solution;
- soit contenant un seul élément S_i . Une solution du problème, optimale ou non, aura alors été isolée;
- soit contenant plus d'un élément.

Par conséquent, à chaque étape, si un sous-ensemble A_p est choisi pour être séparé, on obtiendra:

- soit une suite finie, non vide et évidemment non réduite à un seul élément, de sous-ensembles de A_p, ceux-ci étant notés A_{q+1}, A_{q+2},...,A_{q+j}
- soit A_p lui même, si le principe de séparation conduit à ne pas le séparer, soit parce que A_p est vide, soit parce qu'il se réduit à un seul élément.

Terminologie.

°) Tout sous-ensemble qui peut être choisi pour être séparé est appelé bout pendant.

°°) Si le critère de séparation conduit à ne pas séparer ce bout pendant, celui-ci est dit terminal vide s'il ne contiend aucun élément, ou bien terminal solution s'il contiend un seul élément solution.

Il est à noter que les sous-ensembles engendrés par séparation sont notés A_q , l'indice q indiquant l'ordre dans lequel ces sous-ensembles apparaissent au cours des séparations successives.

C. FONCTION D'EVALUATION ET BORNE

On définit à présent sur chacun des sous-ensembles obtenus une fonction d'évaluation que l'on choisit d'après la nature du problème et plus précisément suivant la fonction objectif à optimiser.

Cette fonction devra permettre de borner inférieurement la valeur des éléments de chacun des sous-

ensembles obtenus par séparation dans le cas d'un problème de minimisation et de les borner supérieurement dans le cas d'un problème de maximisation. Elle devra être choisie de plus, de telle sorte que pour une solution S_k contenue dans un bout pendant terminal solution A_q, la borne b_k évaluée soit telle que:

$$\boxed{b_k = f(S_k)}$$

Supposons à présent que nous ayons à résoudre un problème de minimisation. Nous choisissons alors une fonction d'évaluation qui nous donne lors de la première séparation:

- une borne inférieure b_0 de l'ensemble E des solutions du problème telle que, quelle que soit la solution $S_j \in E$ pour $j=1,\ldots,n$, b_0 soit une minorante de $f(S_j)$.

$$b_0 \leqslant \text{Min } f(S_j) \qquad \forall j = 1,\ldots,n$$

- des bornes inférieures b_i aux k sous-ensembles A_i obtenus précédemment telles que chaque b_i soit respectivement minorante de chacun des A_i.

Donc : $$b_i \leq \min_{S_j \in A_i} f(S_j) \quad \text{pour} \quad i = 1,\ldots,k$$

De plus, puisque par définition $A_i \subset E$, nous avons alors:

$$b_0 \leq b_i \qquad \forall\, i = 1,\ldots,k$$

Il est à noter qu'à chacun des sous-ensembles est associé une borne et une seule.

D. CRITERE DE PROGRESSION

Une fois toutes les bornes $b_1, b_2, \ldots, b_k$ évaluées, on choisit en fonction d'un critère lié à celles ci le nouveau sous-ensemble auquel sera appliqué le principe de séparation.

Ce critère de choix doit permettre d'éviter l'examen des sous-ensembles ne renfermant pas de solutions optimales ou satisfaisantes, soit parce qu'ils sont vides ou soit encore parce que leurs meilleures solutions sont moins bonnes que d'autres déjà obtenues. Il doit per-

mettre de plus, si possible, de choisir parmi les sous-ensembles restants celui ayant le plus de chance de renfermer une solution optimale.

Règle de progression.

On choisira le sous-ensemble auquel correspond le ou un plus petit minorant de la fonction objectif dans le cas d'un problème de minimisation et le ou un plus grand majorant dans le cas d'un problème de maximisation.

Pour poursuivre la résolution de ce problème, on considère une nouvelle propriété séparatrice $\mathcal{P}_B$ distincte de la précédente et permettant de séparer le sous-ensemble sélectionné précédemment.

Soit A_j ce sous-ensemble et soit b_j sa borne. A_j est alors séparé en k' sous-ensembles $A_{k+1},\ldots,A_{k+k'}$ auxquels à l'aide de la fonction d'évaluation on fait correspondre respectivement les bornes $b_{k+1},\ldots,b_{k+k'}$ telles que :

- b_{k+l} soit minorante du sous-ensemble A_{k+l}

pour $l = 1,\ldots,k'$.

$$b_{k+1} \leq \min_{S_j \in A_{k+1}} f(S_j) \quad \text{pour} \quad l = 1,\ldots,k'$$

et puisque $A_{k+1} \subset A_j$ pour $l = 1,\ldots,k'$, on a:

$$b_j \leq b_{k+1} \quad \text{pour } l = 1,\ldots,k'$$

En continuant ainsi le processus et en introduisant à chaque étape une nouvelle propriété séparatrice distincte des précédentes, on engendre une famille de sous-ensembles $A_1, A_2, \ldots, A_q, \ldots$ que l'on peut représenter sous forme d'arborescence.

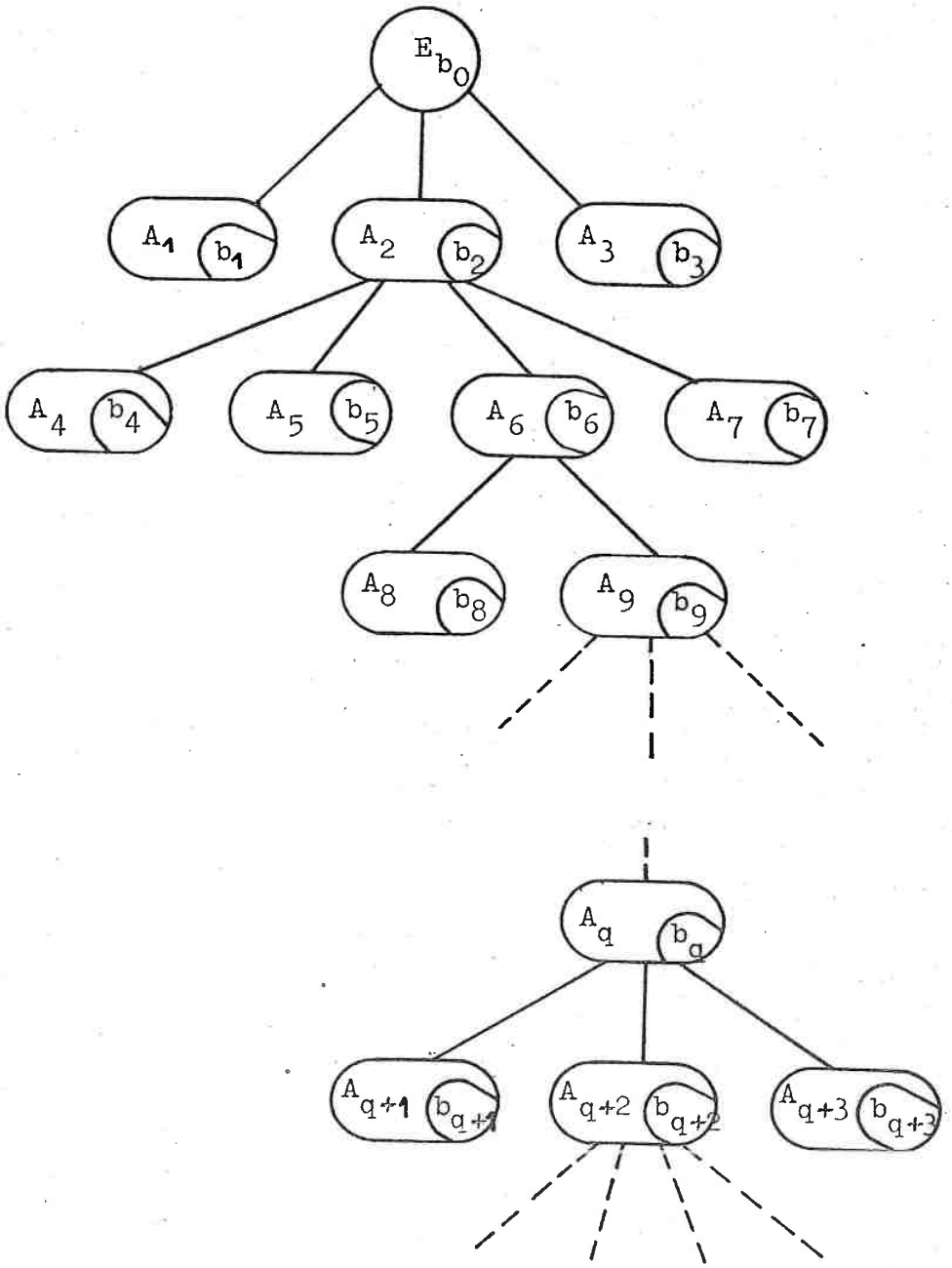
Dans l'exemple d'arborescence de la page suivante nous aurons les inégalités:

$$b_0 \leq b_1,\ b_2,\ b_3$$

$$b_2 \leq b_4,\ b_5,\ b_6,\ b_7$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$b_9 \leq \ldots \leq b_q \leq b_{q+1},\ b_{q+2},\ b_{q+3}$$


E_{b_0}
A_1 b_1
A_2 b_2
A_3 b_3
A_4 b_4
A_5 b_5
A_6 b_6
A_7 b_7
A_8 b_8
A_9 b_9
A_q b_q
A_{q+1} b_{q+1}
A_{q+2} b_{q+2}
A_{q+3} b_{q+3}

De par la nature du problème et du choix des propriétés séparatrices, la suite maximum de parties $A_1, A_2, \ldots, A_q, \ldots$ qu'il est possible d'engendrer par séparation est finie. De plus il est évident que, quel que soit le bout pendant terminal A_q on sait:

- soit prouver que A_q est bout pendant terminal vide, c'est à dire ne donnant pas lieu à une solution;
- soit déduire de A_q une solution optimale ou non.

Le processus prend fin lorsque tous les bouts pendants sont terminaux. Il suffit alors de considérer les différentes solutions associées aux bouts pendants terminaux non vide pour déterminer l'ensemble des solutions optimales du problème.

Démonstration

Soit E^p l'ensemble désignant ce qui reste de l'ensemble E à l'itération p après avoir éliminé les sous-ensembles reconnus sans intérêt lors des itérations antérieures ainsi que tous les bouts pendants terminaux vides ou solutions.

Nous allons montrer tout d'abord dans cette démonstration, que la réunion de tous les bouts pendants à chaque itération n'est autre que l'ensemble E^p.

Nous allons faire la démonstration par récurrence.

°) Initialement nous avons l'ensemble E lui même.

°°) Par définition, nous avons à l'itération p-1 l'égalité suivante:

$$\bigcup_{A_i \in \mathcal{S}^{p-1}} A_i = E^{p-1}$$

$\mathcal{S}^{p-1}$ désigne le groupe des sous-ensembles bouts pendants succeptibles d'être séparés.

°°°) Etablissons la formule pour l'itération p.
Soit A_k le sous-ensemble choisi pour être séparé, $A_k \in \mathcal{S}^{p-1}$, par définition n'est pas terminal. Nous avons alors l'égalité suivante.

$$\bigcup_{A_i \in \mathcal{S}^{p}} A_i = \bigcup_{A_i \in \mathcal{S}^{p-1}} A_i - \bigcup_{A_i \in (suc(A_k) - sûc(A_k))} A_i$$

$suc(A_k)$ désigne le groupe des sous-ensembles obtenus par séparation de A_k.
$sûc(A_k)$ désigne le groupe des sous-ensembles obtenus par séparation de A_k qui sont succeptibles d'être séparés.

d'où nous en déduisons que:

$$\bigcup_{A_i \in \mathcal{S}^p} A_i = E^{p-1} - (E^{p-1} - E^p) = E^p$$

c.q.f.d.

Montrons à présent que, lorsque tous les bouts pendants sont terminaux, nous sommes à l'optimum.

Soit A_p le sous-ensemble choisi au début de l'itération p, en raison du critère de progression, c'est à dire en raison de la règle de choix du sous-ensemble à séparer, nous avons:

$$b_{A_p} \leq b_{A_i} \qquad \forall A_i \in \mathcal{S}^p$$

Or, d'après le critère de minoration, on a:

$$b_{A_i} \leq \operatorname{Min}_{S_i \in A_i} f(S_i) \qquad \forall A_i \in \mathcal{S}^p$$

d'où : $$b_{A_p} \leq \operatorname{Min}_{S_i \in A_i} f(S_i) \qquad \forall A_i \in \mathcal{S}^p$$

or, comme nous l'avons vu ci-dessus:

$$\bigcup_{A_i \in \mathcal{S}^p} A_i = E^p$$

par conséquent, nous en déduisons l'inégalité suivante:

$$b_{A_p} \leqslant \operatorname*{Min}_{S_i \in E^p} f(S_i)$$

De plus, puisque la solution S_p appartient au sous-ensemble A_p et par conséquent à E_p, on a bien:

$$\operatorname*{Min}_{S_i \in E^p} f(S_i) \leqslant f(S_p)$$

Et le sous-ensemble A_p étant bout pendant terminal lorsque le processus s'arrête, on a l'égalité:

$$b_{A_p} = f(S_p)$$

on en déduit que:

$$\boxed{f(S_p) = \operatorname*{Min}_{S_i \in E^p} f(S_i)}$$

La solution S_p est donc optimale par rapport au sous-ensemble E^p et par extension du raisonnement par rapport à l'ensemble E tout entier. En effet, il est clair que toute solution optimale pour le problème déduit du problème initial par réduction de E à E^p est également optimale pour ce problème initial. On obtient ainsi l'ensemble des solutions optimales.

En conclusion, il apparait que le problème le plus délicat lors de l'utilisation de la méthode S.E.P. réside dans le choix des propriétés séparatrices et dans le choix de la fonction d'évaluation permettant de borner pas à pas les sous-ensembles obtenus par séparation. Ces choix sont fonction essentiellement de la nature du problème à résoudre.

Théoriquement cette méthode nous permet d'obtenir l'ensemble de toutes les solutions optimales possibles. Mais pour des raisons liées à la taille mémoire des ordinateurs utilisés, nous nous limiterons à la recherche d'une seule solution optimale dans l'étude pratique du problème.

CHAPITRE II.

APPLICATION DE LA METHODE S.E.P. AU CAS D'UN PROBLEME DE PROGRAMMATION LINEAIRE A VARIABLES BOOLEENNES

Soit à résoudre le problème. P suivant:

Recherche du minimum de la fonction: $\sum_{j=1}^{n} c_j x_j$

avec pour contraintes: $\sum_{j=1}^{n} a_{ij} x_j \geqslant d_i \quad i = 1,\ldots,m$

$x_j = 0$ ou 1 pour $j = 1,\ldots,n$

Les coefficients c_j, a_{ij} et d_i sont des réels de signe quelconque.

Cette deuxième partie reprend tout d'abord le principe fondamental de la méthode S.E.P. en fonction du problème de programmation linéaire à variables booléennes. Nous donnerons ensuite le développement d'un exemple simple. Cet exemple sera suivi d'une description des schémas logiques de calcul pour la program-

mation sur calculateur électronique, subdivisée en deux parties suivant que la valeur de tous les coefficients c_j de la fonction économique à optimiser est positive ou quelconque.

A. PRINCIPE FONDAMENTAL DE LA METHODE S.E.P. POUR UN PROBLEME DE PROGRAMMATION LINEAIRE A VARIABLES BOOLEENNES

Soit P le problème énoncé précédemment et E l'ensemble dénombrable de ses solutions. On cherche à déterminer le sous-ensemble E_m des solutions correspondant au minimum de la fonction citée.

Les variables x_j ne pouvant être affectées que des valeurs 0 ou 1, on prend comme principe de séparation une partition dichotomique fondée sur l'alternative $x_j=0$ ou $x_j=1$. Ce principe conduit à isoler des sous-ensembles de solutions du système:

$$(S) \begin{cases} \sum_{j=1}^{n} a_{ij} x_j \geq d_i & \forall i = 1,\ldots,m \\ x_j = 0 \text{ ou } 1 & \forall j = 1,\ldots,n \end{cases}$$

Tous les sous-ensembles séparés pourront être identifiés par une partition de l'ensemble des indices des variables de la fonction économique à optimiser en trois sous-ensembles:

J_0 : ensemble des indices j tels que $x_j=0$

J_1 : ensemble des indices j tels que $x_j=1$

$\bar{J}$: ensemble des indices j des x_j non arbitrées

La première propriété séparatrice $\mathcal{P}_A$ que l'on choisit, permet de faire une bipartition de l'ensemble E en deux sous-ensembles A et $\bar{A}=\complement_E A$.

A l'ensemble A va correspondre:

$$J_0 = \{j = 1\}$$
$$J_1 = \varnothing$$
$$\bar{J} = \{j \mid j = 2,\ldots,n\}$$

et à $\bar{A}$: $$J_0 = \varnothing$$
$$J_1 = \{j \mid j = 1\}$$
$$\bar{J} = \{j \mid j = 2,\ldots,n\}$$

A chacun de ces sous-ensembles est affectée une borne b_1 et b_2. Ces bornes sont égales respectivement à la valeur de la fonction économique à cette étape suivant la valeur des variables arbitrées.

Le nouveau sous-ensemble choisi pour être séparé sera celui auquel correspond la borne b_i obtenue qui est minimum, puisque nous résolvons un problème de minimisation. Ce sous-ensemble trouvé, on choisira une nouvelle propriété séparatrice $\mathcal{P}_B$ pour séparer ce sous-ensemble. Cette nouvelle propriété sera la mise à 0 ou à 1 d'une nouvelle variable non encore considérée.

Nous obtenons ainsi les quatre sous-ensembles $A \cap B$, $A \cap \bar{B}$, $\bar{A} \cap B$ et $\bar{A} \cap \bar{B}$.

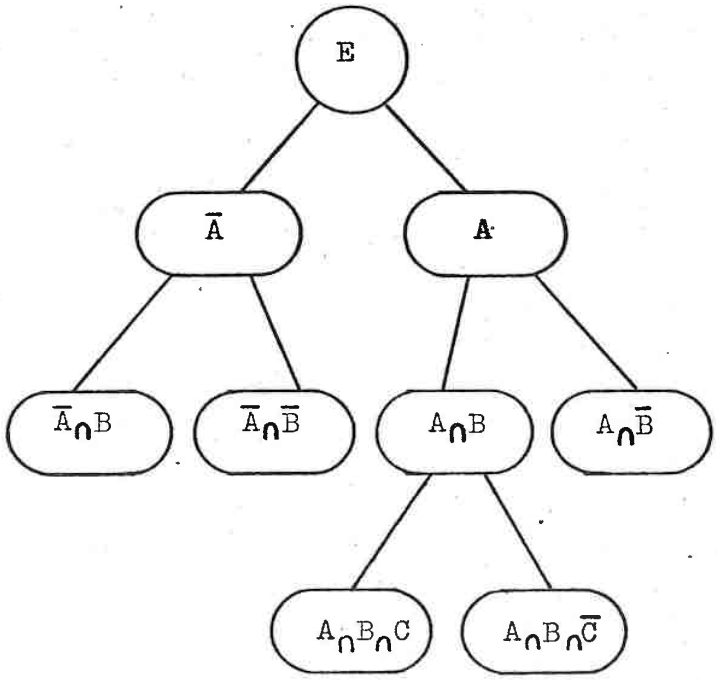
A $A \cap B$ va correspondre: $J_0 = \{j = 1,2\}$
$J_1 = \varnothing$
$\bar{J} = \{j | j = 3,\ldots,n\}$

à $A \cap \bar{B}$ va correspondre: $J_0 = \{j = 1\}$
$J = \{j = 2\}$
$J = \{j | j = 3,\ldots,n\}$

à $\bar{A} \cap B$ va correspondre: $J_0 = \{j = 2\}$
$J_1 = \{j = 1\}$
$\bar{J} = \{j | j = 3,\ldots,n\}$

à $\bar{A} \cap \bar{B}$ va correspondre: $J_0 = \varnothing$
$J_1 = \{j = 1,2\}$
$\bar{J} = \{j | j = 3,\ldots,n\}$

A chacun des sous-ensembles est affecté une borne ayant les propriétés vues au chapitre précédent et en continuant ainsi le processus, on construit une arborescence ayant la structure suivante:



Pour éviter de décrire l'arborescence complète, c'est à dire pour éviter de décrire au $n^{ième}$ pas 2^n sommets pendants (n étant le nombre de variables de la fonction économique à optimiser), on choisit une règle de progression qui permet de n'en décrire à chaque pas que 2 seulement.

Cette règle de progression sera de séparer au pas n, le ou un des bouts pendants correspondant à la borne inférieure des bornes de tous les bouts pendants de l'arborescence à ce pas. S_i nous avions traité un problème de maximisation nous aurions choisi la borne supérieure des bornes de tous les bouts pendants.

L'ensemble E_m des solutions est borné inférieurement par cette valeur, car l'union de tous les sous-ensembles bouts pendants donnent le référentiel E lui-même.

En effet, si l'on se reporte à la figure précédente, les bouts pendants sont $\bar{A} \cap B$, $\bar{A} \cap \bar{B}$, $A \cap B \cap \bar{C}$, $A \cap B \cap C$ et $A \cap \bar{B}$.

On vérifie facilement que:

$$(\bar{A} \cap B) \cup (\bar{A} \cap \bar{B}) \cup (A \cap B \cap \bar{C}) \cup (A \cap B \cap C) \cup (A \cap \bar{B}) = E \quad (1)$$

nous avons: $\bar{C} = \int_B C$ d'où: $(A \cap B \cap \bar{C}) \cup (A \cap B \cap C) = (A \cap B)$

donc $(1) \Longleftrightarrow (\bar{A} \cap B) \cup (\bar{A} \cap \bar{B}) \cup (A \cap B) \cup (A \cap \bar{B})$

de même: $\bar{B} = \int_A B$ et $\bar{A} = \int_E A$

d'où: $(1) \Longleftrightarrow (\bar{A} \cap B) \cup (\bar{A} \cap \bar{B}) \cup A$

or: $(\bar{A} \cap B) \cup (\bar{A} \cap \bar{B}) = \bar{A}$

par conséquent: $(1) \Longleftrightarrow \bar{A} \cup A = E$ c.q.f.d.

Le processus prend fin lorsque tous les bouts pendants sont terminaux. On a alors l'ensemble E_m des solutions optimales du problème.

1ère ETAPE:

En arbitrant la première variable de la fonction économique à 0 ou à 1, on décompose le problème P énoncé ci-dessus en deux sous-problèmes A_1 et A_2.

A_1 : Soit $x_1 = 0$

Le problème s'énonce alors de la façon suivante:

$$A_1 : \begin{cases} \text{Min } z_1 = 3x_2 + 5x_3 + x_4 \\ 3x_2 - x_3 + 2x_4 \geq 3 \\ x_j = 0 \text{ ou } 1 \quad \text{pour } j = 2,3,4 \end{cases}$$

Nous prenons pour fonction d'évaluation la fonction économique à minimiser.

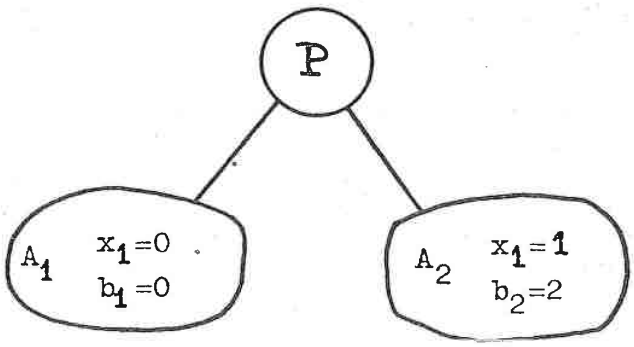
La minorante du sous-problème A_1 que nous en déduisons est $b_1 = 0$, par conséquent dans toute la descendance de A nous aurons: Min $z_1 \geq 0$

$\underline{A_2}$: Soit $x_1 = 1$

$$A_2 : \begin{cases} \text{Min } z_2 = 2 + 3x_2 + 5x_3 + x_4 \\ 1 + 3x_2 - x_3 + 2x_4 \geq 3 \\ x_j = 0 \text{ ou } 1 \quad \text{pour } j = 2,3,4 \end{cases}$$

Dans ce sous-problème, la borne calculée à l'aide de la fonction d'évaluation est $b_2 = 2$, d'où nous en déduisons que $\text{Min } z_2 \geq 2$ dans toute la succession de A_2.

La représentation du problème sous forme d'arborescence à ce point de la résolution est la suivante:



REGLE DE PROGRESSION :

Lors de la résolution d'un problème de minimisation, on s'intéressera à chaque étape aux descendances du sous-problème auquel correspond la borne la plus petite de la fonction objectif.

Par conséquent, dans une deuxième étape, nous allons étudier la descendance du sous-problème A_1 auquel correspond la borne la plus petite $b_1 = 0$.

2ème ETAPE :

On décompose le sous-problème $A_1(x_1=0,\ b_1=0)$ en deux sous-problème A_3 et A_4 en arbitrant la deuxième variable de la fonction économique à 0 puis à 1.

A_3 : Soit $x_2 = 0$

$$A_3 : \begin{cases} \text{Min } z_3 = 5x_3 + x_4 \\ -x_3 + 2x_4 \geq 3 \qquad (c_3) \\ x_j = 0 \text{ ou } 1 \quad \text{pour} \quad j = 3,4 \end{cases}$$

Considérons la contrainte (C_3) du sous-problème A_3: $-x_3 + 2x_4 \geqslant 3$. Sachant que les variables x_j ne peuvent prendre que les valeurs 0 ou 1, le premier membre de la contrainte sera maximum si on égale:

- à 0 la variable x_j correspondant à un coefficient négatif dans cette contrainte;
- à 1 la variable x_j correspondant à un coefficient positif ou nul.

Dans ce cas, la valeur maximum du premier membre de la contrainte (C_3) est 2 et puisque la valeur du second membre est 3, la contrainte n'est pas vérifiée et ne pourra plus l'être dans la suite du problème.

On se trouve donc dans un cas d'arrêt de la progression car il est initéressant d'étudier la descendance de A_3 qui est bout pendant terminal vide.

Cas général:

Supposons être arrivé à la résolution du sous-problème A_k.

Dans A_k soit J l'ensemble des indices j tel que x_j ait une valeur déterminée et $\bar{J}$ l'ensemble des indices j tel que x_j soit indéterminée.

Si pour un indice i on a:

$$\sum_{j \in J} a_{ij} x_j + \sum_{j \in \bar{J}} \max (0, a_{ij}) < d_i$$

on est alors dans un cas d'arrêt de la progression.

En effet, dans la suite de la résolution, même en fixant à 0 les variables auxquelles correspondent un coefficient $a_{ij} \leqslant 0$, et à 1 celles auxquelles correspondent un coefficient $a_{ij} > 0$, l'inégalité:

$$\sum_{j \in J} a_{ij} x_j < d_i$$ resterait vérifiée.

Par conséquent, il est inutile d'explorer les descendances du sous-problème A_k qui est bout pendant terminal vide, puisque ne donnant pas lieu à une solution.

Remarque:

Pour repérer ce bout pendant terminal vide dans l'arborecence, nous pouvons, soit l'indiquer explicitement en précisant la mention "impossible" dans la feuille correspondante, soit égaler la borne de ce sous-problème à l'infini, $b_k = \infty$.

$\underline{A_4 : \text{Soit } x_2 = 1}$

$$A_4 : \begin{cases} \text{Min } z_4 = 3 + 5x_3 + x_4 \\ 3 - x_3 + 2x_4 \geqslant 3 \qquad (C_4) \\ x_j = 0 \text{ ou } 1 \quad \text{pour } j = 3,4 \end{cases}$$

La minorante de la fonction objectif du sous-problème A_4 est $b_4=3$. Cette minorante est obtenue en supposant $x_3=x_4=0$ dans la fonction à optimiser, puisque par hypothèse tous les coefficients de cette fonction sont positifs.

Or pour $x_3=x_4=0$, nous constatons que la contrainte (C_4) est vérifiée. En effet le premier et le second membre sont alors tous deux égaux à 3.

Donc à ce niveau de la résolution nous pouvons dire que le sous-problème A_4 donne lieu à une solution implicitement évaluée qui est $z_4 = 3$ obtenue pour $x_2 = 1$ et $x_1=x_3=x_4=0$.

Remarque:

Si d'un sous-problème A_k nous pouvons déduire une solution que l'on dira alors "implicitement évaluée" (puisque les variables non arbitrées sont mises implici-

tement à zéro), alors cette solution est la meilleure que l'on puisse obtenir des descendances de A_k, c'est à dire la plus petite dans le cas de ce problème de minimisation et de plus:

- elle est unique si les coefficients c_j sont strictement positifs;
- mais non obligatoirement unique si il existe des coefficients nuls dans la fonction économique.

Démonstration:

Supposons que nous résolvions le sous-problème A_1. A_1 s'énonce de la façon suivante:

$$A_1 : \begin{cases} x_1,\ldots,x_k \text{ déteminées (égaux à 0 ou à 1)} \\ x_{k+1},\ldots,x_n \text{ indéterminées} \\ \text{Min } z_1 = \sum_{j=1}^{n} c_j x_j \\ \text{avec: } \sum_{j=1}^{n} a_{ij} x_j \geq d_i \quad \forall i = 1,\ldots,m \end{cases}$$

Si quelque soit $i=1,\ldots,m$ nous avons $\sum_{j=1}^{n} a_{ij} x_j \geq d_i$ alors toutes les contraintes du problème sont vérifiées et par conséquent nous pouvons déduire de A_1 une solution S que l'on dira implicitement évaluée en supposant:

$x_{k+1}=x_{k+2}=\ldots=x_n=0$.

Soit $z = \sum_{j=1}^{k} c_j x_j$ la valeur de la fonction économique dans cette solution S. ($\sum_{j=k+1}^{n} c_j x_j$ étant nulle)

A) Montrons que S est la meilleure, c'est à dire la plus petite solution que l'on puisse obtenir des descendances de A_1.

Pour cela nous allons raisonner par l'absurde.

Supposons qu'il existe une autre solution S' que l'on obtient à partir des descendances de A_1 telle que z' la valeur de la fonction économique en S' soit strictement inférieure à z c'est à dire: $z' < z$.

Alors il existe au moins un indice q tel que:

$n \geq q > k$

z' s'écrive $z' = \sum_{j=1}^{k} c_j x_j + c_q x_q$

Or puisque $z' < z$ et que la variable x_q ne peut prendre que la valeur 0 ou 1, on en déduit que le coefficient c_q est strictement négatif, ce qui est contraire aux hypothèses.

Par conséquent S est la plus petite solution que l'on puisse obtenir des descendances de A_1. c.q.f.d.

B) Montrons les conditions d'unicité et de non unicité de la solution S.

La solution S obtenue implicitement est:

$$(S): \begin{cases} z = \sum_{j=1}^{k} c_j x_j \\ x_1, \ldots, x_k \text{ égaux à 0 ou 1} \\ x_{k+1} = x_{k+2} = \ldots = x_n = 0 \end{cases}$$

Supposons que nous obtenions une solution (S_1) distincte de (S) à partir des descendances du sous-problème A_1. Comme nous l'avons vu précédemment la valeur z_1 de la fonction économique en (S_1) est égale à celle en (S), et z_1 s'écrira:

$$z_1 = \sum_{j=1}^{k} c_j x_j + \sum_{j=k+1}^{n} c_j x_j$$

or pour que $z_1 = z$, il faut que $\sum_{j=k+1}^{n} c_j x_j = 0$

Deux cas peuvent se présenter:

- les coefficients c_j sont tous strictement positifs.

Alors pour que $\sum_{j=k+1}^{n} c_j x_j = 0$, il faudrait que $x_j=0$ pour $j=k+1,\ldots,n$ ce qui donne la solution (S), d'ou l'unicité de la solution.

- les coefficients c_j sont positifs ou nuls.

Alors il peut exister un indice q tel que :

$$c_q = 0 \quad \text{avec} \quad k < q \leq n$$

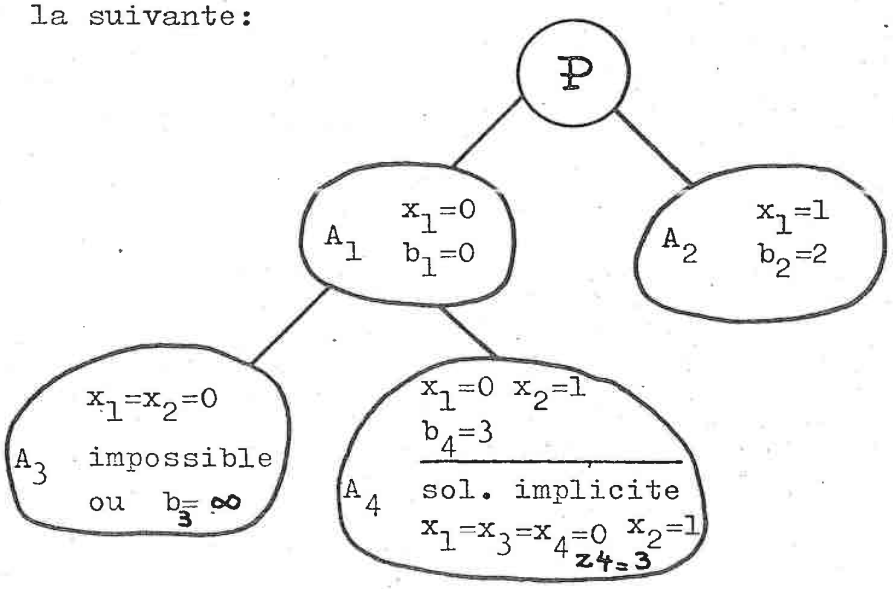
$$z_1 = \sum_{j=1}^{k} c_j x_j + c_q x_q = z$$

avec $x_q = 1$, ce qui donne une solution (S_1) distincte de (S) d'ou la non unicité de la solution. c.q.f.d.

Par conséquent, pour reprendre notre problème, il s'avère totalement inutile d'étudier toutes les descendances de A_4 qui ne pouront donner lieu qu'à des solutions plus grandes puisque tous les coefficients de la fonction économique sont strictement positifs.

Ainsi A_4 peut être considéré comme un bout pendant terminal donnant lieu à une solution implicitement évaluée.

La représentation sous forme d'arborescence est la suivante:



Remarque: Le sous-problème A_2 est le seul à être bout pendant non terminal.

3ème ETAPE:

Nous allons étudier les descendances du sous-problème A_2 ($x_1=1$, $b_2=2$).

A_5: Soit $x_2=0$

$$A_5: \begin{cases} \text{Min } z_5 = 2 + 5x_3 + x_4 \\ 1 - x_3 + 2x_4 \geq 3 \\ x_j = 0 \text{ ou } 1 \quad \text{pour } j = 3,4 \end{cases}$$

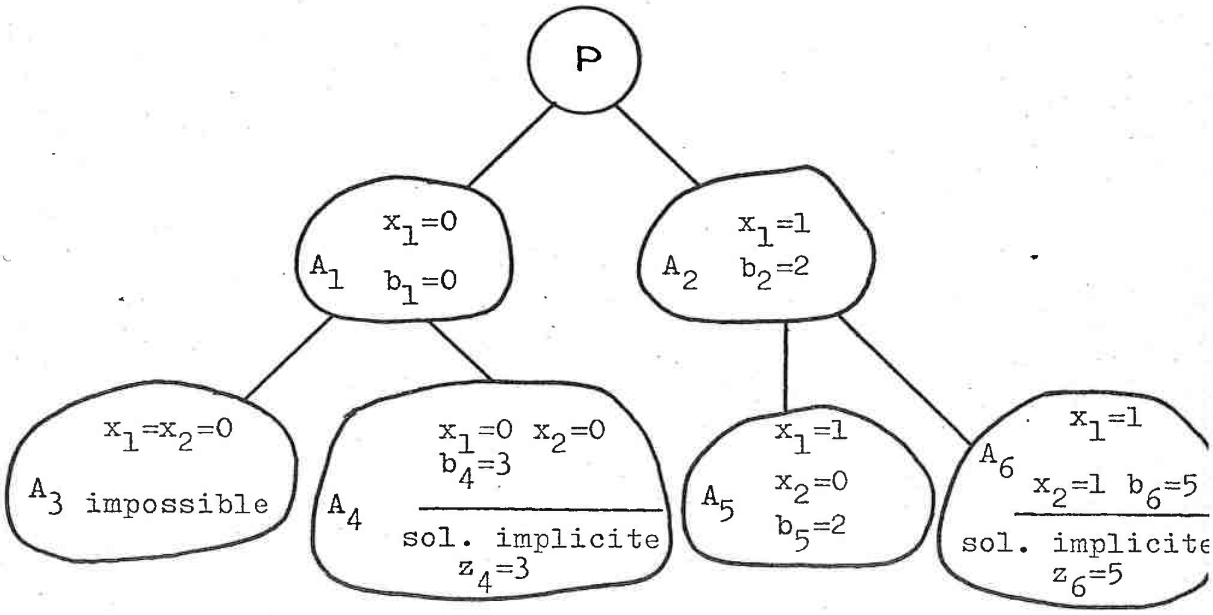
La borne du sous-problème A_5 est $b_5=2$, donc dans toute la succession de A_5, min $z_5 \geq 2$.

A_6: Soit $x_2=1$

$$A_6: \begin{cases} \text{Min } z_6 = 2 + 3 + 5x_3 + x_4 \\ 1 + 3 - x_3 + 2x_4 \geq 3 \qquad (C_6) \\ x_j = 0 \text{ ou } 1 \quad \text{pour } j = 3,4 \end{cases}$$

La borne du sous-problème A_6 est $b_6=5$. Pour

$x_3=x_4=0$ la contrainte (C_6) est vérifiée, donc comme dans le cas du sous-problème A_4, le sous-problème A_6 donne lieu à une solution implicitement évalué ($z_6=5$ pour $x_1=1, x_2=1, x_3=0, x_4=0$). Par conséquent nous ne nous intéressons plus aux descendances de A_6.



4ème ETAPE:

Les sous-problèmes A_3, A_4 et A_6 sont bouts pendants terminaux, nous nous intéressons aux descendances de A_5 ($x_1=x_2=0$, $b_5=2$) seul bout pendant non terminal, puisque sa borne b_5 est inférieure à la meilleure solution déjà obtenue.

$\underline{A_7}$: $\underline{\text{Soit } x_3 = 0}$

$$A_7: \begin{cases} \text{Min } z_7 = 2 + x_4 \\ 1 + 2x_4 \geq 3 \\ x_j = 0 \text{ ou } 1 \quad \text{pour } j=4 \end{cases}$$

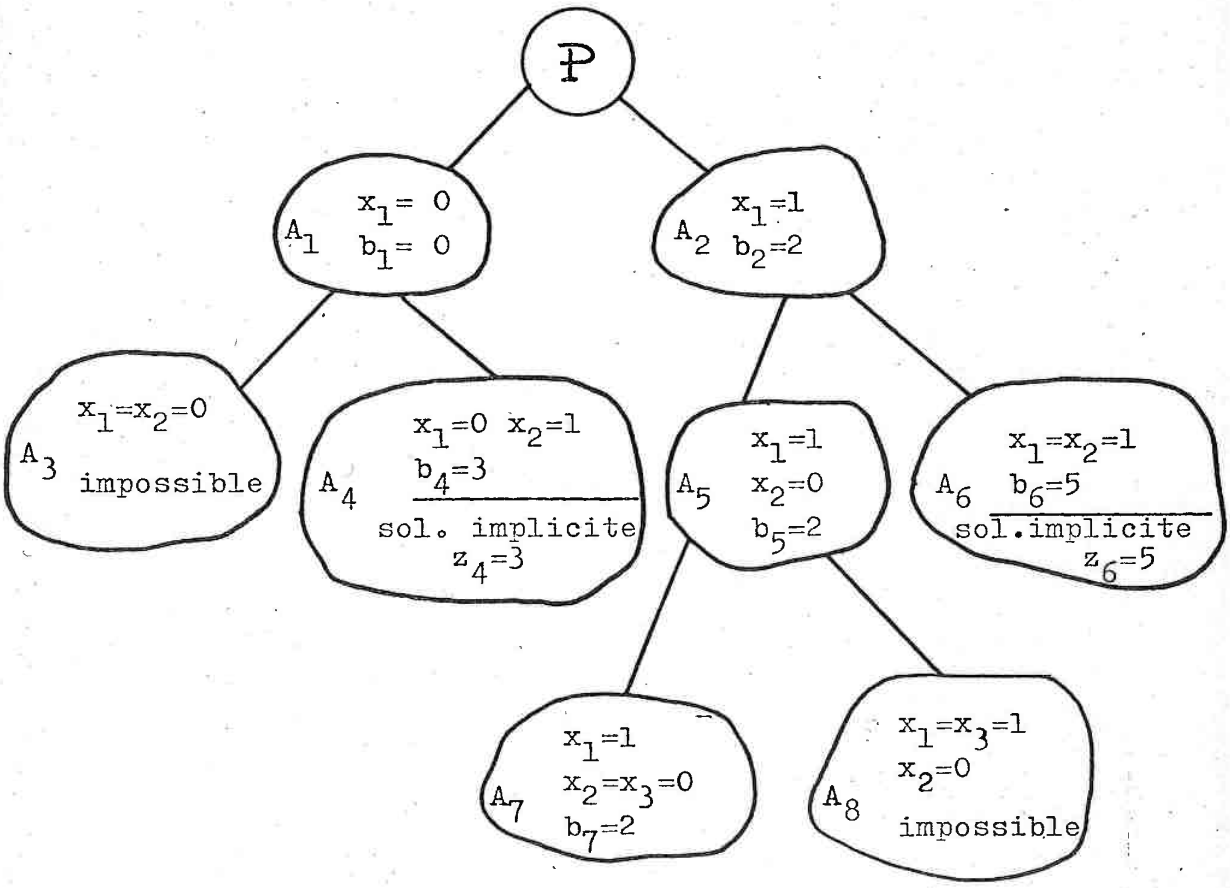
La borne du sous-problème A_7 est $b_7=2$, donc dans toute la succession de A_7, $\text{Min } z_7 \geq 2$

$\underline{A_8}$: $\underline{\text{Soit } x_3 = 1}$

$$A_8: \begin{cases} \text{Min } z_8 = 2 + 5 + x_4 \\ 1 - 1 + 2x_4 \geq 3 \qquad (C_8) \\ x_j = 0 \text{ ou } 1 \quad \text{pour } j=4 \end{cases}$$

Comme dans le cas du sous-problème A_3, la contrainte (C_8) n'est plus vérifiée et ne pourra plus l'être dans la suite de la résolution. Par conséquent nous sommes dans un cas d'arrêt du problème et A_8 peut être considéré comme un bout pendant terminal vide, puisqu'il ne donne pas lieu à une solution.

La représentation sous forme d'arborescence à ce niveau de la résolution est la suivante:



5ème ETAPE:

Nous nous intéressons à présent aux descendances du sous-problème A_7 ($x_1=1$, $x_2=x_3=0$, $b_7=2$)

A_9: Soit $x_4 = 0$

$$A_9: \begin{cases} \text{Min } z_9 = 2 \\ 1 \geqslant 3 \end{cases}$$

Ce cas est impossible, A_9 est bout pendant terminal vide, sans solution.

$\underline{A_{10}: \text{ Soit } x_4 = 1}$

$$A_{10}: \begin{cases} \text{Min } z_{10} = 2+1 \\ 1 + 2 \geq 3 \end{cases}$$

Nous obtenons une solution $z_{10} = 3$.

Conclusion :

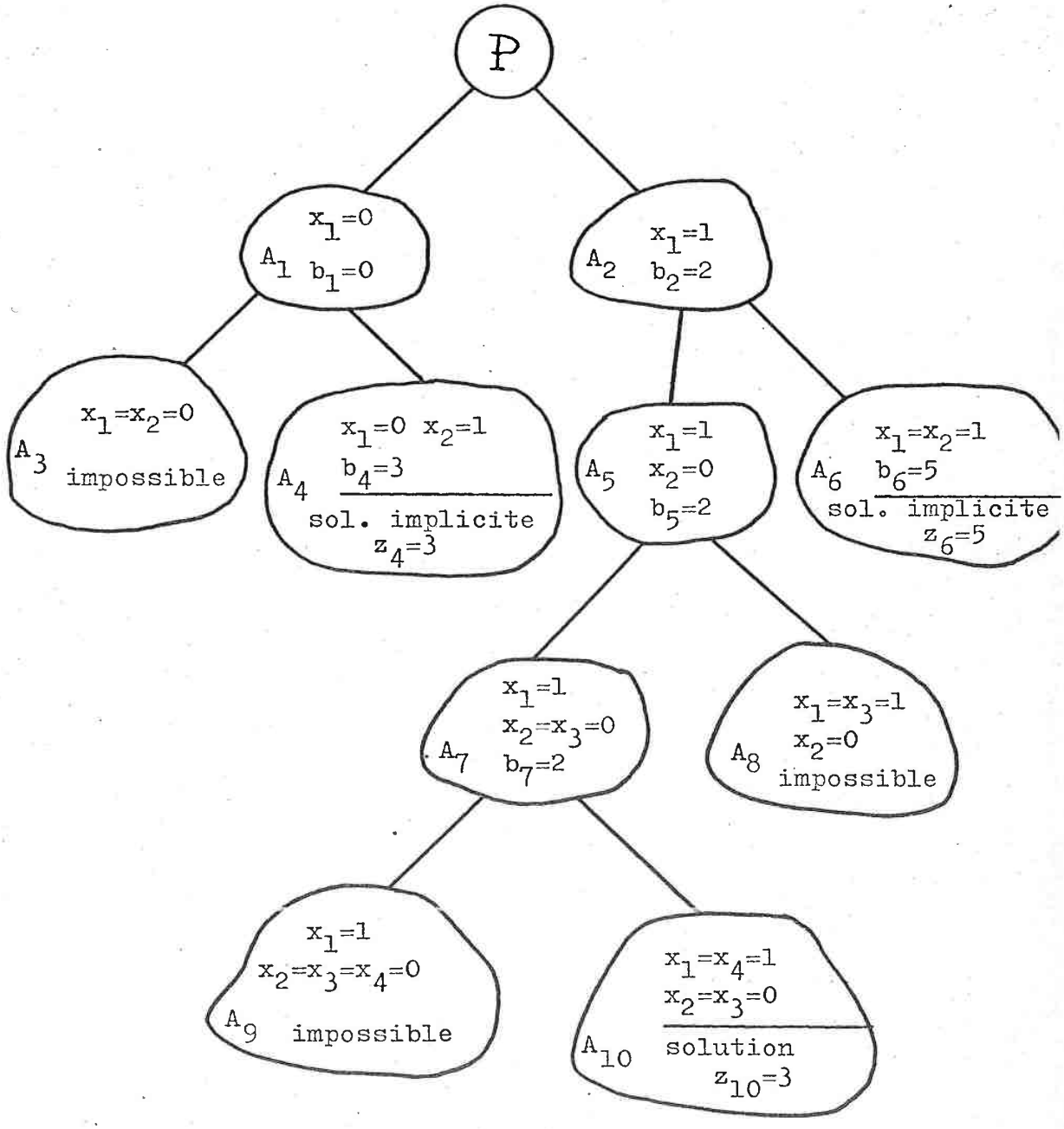
Tous les bouts pendants étant terminaux, le processus de séparation prend fin et le problème est résolu. Puisque tous les coefficients de la fonction économique sont strictement positifs, nous avons obtenu l'ensemble des solutions optimales.

$z_4 = 3$ pour $x_1=x_3=x_4=0$, $x_2=1$

$z_{10}= 3$ pour $x_1=x_4=1$, $x_2=x_3=0$

Pour les raisons citées précédemment, lors de la programmation du problème nous ne rechercherons qu'une seule solution optimale.

Représentation finale du problème sous forme d'arborescence.



C. ENONCE DE L'ALGORITHME

Différents algorithmes ont été proposés pour la résolution de problèmes de programmation linéaire à variables booléennes par la méthode S.E.P.. Ceux-ci se différencient les uns des autres par certains critères:

- critère de choix de la variable à arbitrer à 0 ou à 1;
- critère de choix du point de départ d'une nouvelle branche de l'arborescence;
- critère d'arrêt de la progression.

L'algorithme que nous allons décrire permet d'explorer efficacement l'ensemble des solutions en examinant chacune au plus une fois. Il conduit à un programme optimal s'il en existe un, ou permet de reconnaître qu'il n'en existe pas dans le cas contraire.

Dans une première partie nous étudirons le problème dans le cas où les coefficients a_{ij} et d_i des con-

traintes sont des nombres réels de signe quelconque et les coefficients c_j de la fonction économique des réels positifs ou nuls. Dans une seconde partie nous généraliserons l'étude du problème en supposant tous les coefficients a_{ij}, d_i, c_j réels de signe quelconque. Puis la programmation, l'organigramme et le programme FORTRAN de cette méthode constitueront la dernière partie de ce chapitre.

A) Cas où les coefficients c_j sont des réels positifs ou nuls.

L'exploration des solutions du problème, faite à partir d'une solution initiale telle que toutes les variables soient nulles, repose sur un ensemble de règles.

- une règle de progression;
- des règles d'arrêt de l'algorithme;
- une règle de retour arrière permettant de faire le choix du point de départ d'une nouvelle ligne de progression.

a) Règle de progression.

Cette règle de progression doit permettre de choisir la variable à arbitrer, c'est à dire de diriger le choix de la variable de la fonction économique que l'on fixe à 0 ou à 1.

Dans ce chapitre, nous prendrons pour règle générale une méthode simple ne nécessitant aucun calcul préalable. Ce choix de la variable à arbitrer se fera de façon séquentielle de la première jusqu'à la dernière.

Nous verons dans le chapitre suivant qu'il existe des possibilités d'amélioration quant au choix de cette variable. Mais ces méthodes, qui permettent d'arbitrer une variable en fonction de la nature des contraintes et des coefficients de la fonction économique, nécessitent un certain nombre de calculs préalables.

b) Règle d'arrêt de la progression dans l'arborescence.

Ces règles doivent permettre de reconnaître qu'un bout pendant de l'arborescence est terminal.

1. Terminal vide.

Soit J l'ensemble des indices j tel que x_j aient une valeur fixée à 0 ou à 1. Et $\bar{J} = \{1,\ldots,n\} - J$. Alors si pour un indice $i \in \{1,\ldots,m\}$ on a:

$$\sum_{j\in J} a_{ij}x_j + \sum_{j\in \bar{J}} \max(0,a_{ij}) < d_i$$

c'est à dire, si une des contraintes n'est plus vérifiée, le bout pendant correspondant est terminal pour cas d'impossibilité de vérification de la contrainte.

En effet, dans la suite de la résolution, même en fixant à 1 les variables ayant un coefficient a_{ij} strictement positif et à 0 les variables ayant un coefficient a_{ij} négatif ou nul, on aura toujours l'inégalité:

$$\sum_{j=1}^{n} a_{ij}x_j < d_i$$

c'est à dire que la contrainte resterait non vérifiée dans toute la descendance, donc que l'étude de celle-ci est inintéressante.

2. Si dans un bout pendant toutes les contraintes sont vérifiées, même si toutes les variables ne sont

pas arbitrées, alors conformément à la démonstration faite dans le chapitre II, paragraphe II, 2ème étape, il est inutile de continuer à progresser. Ce bout pendant est terminal, donnant lieu à une solution "implicitement évaluée".

3. Si la borne b_i associée à un bout pendant est supérieure ou égale à la valeur de la meilleure solution déjà obtenue, il est inutile de continuer la progression dans cette direction puisque toutes les solutions descendantes que l'on obtiendrait alors ne seraient que moins bonnes, les coefficients c_j étant tous positifs ou nuls. En effet, à chaque étape de la résolution la valeur de la meilleure solution déjà obtenue permet de borner supérieurement la valeur de la fonction économique.

4. Si toutes les variables sont arbitrées, alors le bout pendant correspondant, qu'il soit solution ou non, est évidemment terminal.

c) Règle de retour-arrière.

Lorsqu'on a atteint un bout pendant auquel l'une des quatre règles précédentes s'applique, ce bout pen-

dant est terminal et on revient en arrière de la façon suivante:

On considère tous les bouts pendants déjà obtenus en excluant tous ceux examinés antérieurement et tous ceux dont on peut montrer par l'emploi d'un critère d'arrêt qu'ils sont inintéressants pour la suite. Parmi ces bouts pendants non terminaux, on choisit comme point de départ d'une nouvelle descendance celui dont la borne correspondante est la plus petite puisque nous résolvons un problème de minimisation. On lui applique alors les règles de progression et d'arrêt énoncées ci-dessus.

d) Règle d'arrêt final.

Le développement de l'algorithme prend fin, lorsque tous les bouts pendants sont terminaux. Nous avons alors isolé une solution optimale du problème.

B) Cas où les coefficients c_j sont de signe quelconque.

La solution au problème posé par la présence de coefficients négatifs dans la fonction économique réside dans l'exécution d'un changement de variable approprié qui permet de rendre tous les coefficients de la fonction économique positifs sans modifier le problème et ainsi de pouvoir utiliser la méthode précédente dans son intégralité pour résoudre des problèmes dont la fonction économique a des coefficients de signe quelconque.

Changement de variable. Puisque nous voulons faire en sorte que tous les coefficients de la fonction économique soient positifs, nous allons éffectuer le changement de variable $x' = 1-x$ sur toutes les variables auxquelles correspond un coefficient négatif dans la fonction économique.

D. PROGRAMMATION

a) Les données du programme à introduire sont les suivantes:

- N le nombre de variables.
- les N coefficients des variables de la fonction économique qui seront rangés dans le vecteur Z(i).
- NBRE le nombre de contraintes du problème.
- les différents coefficients des variables des contraintes qui seront rangés dans la matrice CONTR(i,j), l'indice i indiquant le numéro de la contrainte et l'indice j l'indice du coefficient de la contrainte i. L'élément CONTR(i,N+1) est le second membre de chaque contrainte, c'est à dire le coefficient d_i de l'énoncé classique du problème.

n.b. : les différents coefficients nuls de la fonction économique ou des contraintes sont introduits au même titre que les autres.

b) Les différents vecteurs, matrices ou identificateurs utilisés sont les suivants:

ITER(i) : vecteur donnant le numéro de l'itération. Les itérations sont numérotées 1,2,...,k,... dans l'ordre ou elles apparaissent au cours des calculs et ne changent plus de numéro dans la suite des opérations.

PREDES(i) : vecteur dont les éléments indiquent de quelle itération, l'itération i est descendante.

BOUPEN(i) : vecteur dont les éléments indiquent si l'itération i est bout pendant ou non.
BOUPEN(i) = 0 si l'itération i est bout non pendant.
BOUPEN(i) = 1 si l'itération i est bout pendant.

VER(i) : vecteur dont les éléments prennent soit la valeur 0 soit la valeur 1.
VER(i) = 1 si à l'itération i les contraintes sont ou peuvent encore être vérifiées.
VER(i) = 0 si à l'itération i les contraintes ne sont pas ou ne peuvent plus être vérifiées. Dans ce cas, l'itération i est bout pendant terminal vide puisque ne donnant pas lieu à une solution.

SOL(i) : vecteur dont les éléments indiquent si à l'itération i nous avons obtenu une solution ou non.
SOL(i) = 1 si à l'itération i nous obtenons une solution.
SOL(i) = 0 si à l'itération i nous n'obtenons pas de solution.

IND(i,j) : matrice dont les éléments indiquent la valeur prise par les différentes variables de la fonction économique dans l'itération i.

NBIND(i) : vecteur dont les éléments indiquent le nombre de variables de la fonction économique ayant une valeur fixée dans l'itération i.

VALZ(i) : vecteur dont les éléments indiquent la valeur de la borne de la fonction ou la valeur de la solution suivant le cas.

INDSOL(j) : vecteur dont les éléments sont les valeurs prises par les variables de la fonction économique de la meilleure solution connue.

CHGT(i) : vecteur dont les éléments indiquent si un changement de variable a été effectué sur certaines variables.

SOLL : identificateur représentant la valeur de la meilleure solution connue. Initialement SOLL = ∞ ou pour des raisons de programmation SOLL = $2^{31}-1$.

ZMIN : identificateur servant dans la recherche du point de départ d'une nouvelle descendance. Il permet de trouver la borne inférieure de tous les bouts pendants non terminaux connus.

CONST : identificateur représentant la somme des coefficient des variables sur lesquelles a été effectué un changement de variable.

c) Schéma logique de calcul.

Ce schéma se décompose en trois phases.

1. Phase de calcul d'une itération.

Puisque le programme a été écrit en FORTRAN, tous les vecteurs ou matrices utilisés sont initialisés à 0. Cette phase se déroule de la façon suivante.

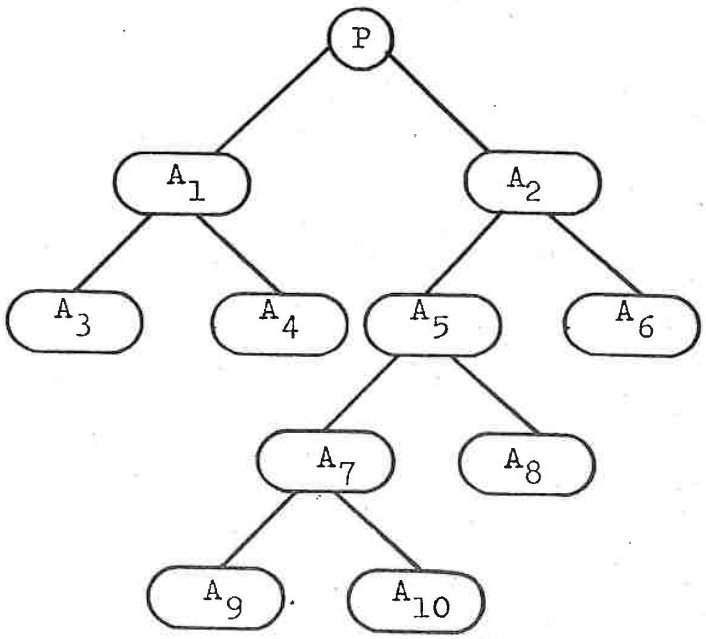
1.1. On considère tout d'abord le signe des coefficients de la fonction économique. On effectue le changement de variable de la manière énoncée dans le paragraphe précédent pour tous ceux qui sont négatifs. Le vecteur CHGT(i) est construit et permettra ainsi de savoir sur quelles variables a été effectué un changement. Une fois ces changements de variable effectués le problème est uniformalisé et nous pouvons passer à la phase de calcul proprement dite.

1.2. On charge dans le vecteur ITER le numéro de l'itération en cours. Ce vecteur ne sert en aucun cas dans le déroulement des calculs, mais il permet de connaître le nombre d'itérations nécessaires à l'obtention d'une solution optimale et de donner les renseignements nécessaires à la construction de l'arborescence obtenue.

De même, on mettra dans le vecteur PREDES le numéro, dont l'itération en cours est la descendante. Ce vecteur, comme le vecteur ITER est là à titre purement indicatif.

Exemple:

Dans l'exemple de déroulement de la méthode donné précédemment nous avions à la fin de la résolution du problème obtenu l'arborescence suivante:



Cette arborescence est construite à partir des vecteurs ITER et PREDES suivants:

ITER	1	2	3	4	5	6	7	8	9	10
PREDES	0	0	1	1	2	2	5	5	7	7

1.3. Une variable de la fonction économique est alors arbitrée à 0 ou à 1. Le choix de cette variable dans ce cas est purement arbitraire. Comme nous l'avons vu elles sont choisies les unes après les autres dans l'ordre ou elles apparaissent. Un index AIG permet suivant sa valeur de nous renseigner sur l'arbitrage à effectuer. On positionne alors l'élément correspondant de la matrice IND(i,j) à 0 ou à 1 suivant le cas, i étant le numéro de l'itération et j l'indice de la variable sélectionnée.

Cette itération étant un bout pendant de l'arborescence, on met à 1 l'élément correspondant du vecteur BOUPEN pour le signaler. Puis la valeur de l'indice j de la variable arbitrée est chargée dans le vecteur NBIND afin de nous permettre de savoir à tout moment combien de variables, dans une itération, ont une valeur déterminée.

Une fois tous ces vecteurs positionnés, on passe à la partie vérification des contraites et calcul de la borne de la fonction économique ou calcul de la solution.

1.4. Vérification des contraintes. Le raisonnement suivant est fait sur chacune des contraintes du problème. Supposons que nous soyons à l'itération p. Pour cette itération, soit k l'élément NBIND(i) indiquant le

nombre de variables ayant une valeur fixée. Nous effectuons alors la somme des k premiers coefficients de la contrainte considérée qui correspondent à un élément non nul de IND(p,j).

$$\text{Soit} \quad SOM = \sum_{j=1}^{k} CONTR(i,j) \quad \text{pour } j \mid IND(i,j) \neq 0$$

Si cette somme SOM est supérieure ou égale à CONTR(i,N+1), c'est à dire au second membre de la contrainte, alors nous indiquons que cette contrainte est vérifiée en positionnant l'élément correspondant du vecteur SOL à 1, dans le cas contraire ce vecteur est positionné à 0.

Dans la mesure ou dans un problème nous avons plusieurs contraintes, il suffit qu'une et une seule ne soit pas vérifiée pour que l'élément correspondant du vecteur SOL soit positionné à 0.

Si les contraintes ne sont pas vérifiées, c'est à dire si nous ne sommes pas à une solution, nous allons nous assurer que ces contraintes peuvent être vérifiées dans la suite de la résolution. Dans le cas contraire, nous mentionerons que cette itération est bout pendant puisque sa descendance est inintéressante.

Pour ce faire, à la somme précédemment obtenue (SOM) nous allons ajouter tous les coefficients positifs de la contrainte que nous n'avons pas encore considéré. Nous obtenons une nouvelle somme:

$$SOM1 = SOM + \sum_{j=k+1}^{n} CONTR(i,j) \quad \text{pour } j \mid CONTR(i,j) > 0$$

Si cette nouvelle somme SOM1 est supérieure ou égale à CONTR(i,N+1), c'est à dire au second membre de la contrainte alors cette contrainte pourra encore être vérifiée dans la suite du problème, dans le cas contraire cette itération est bout pendant terminal vide puisque ne donnant pas lieu à une solution optimale ou non.

Si toutes les contraintes du problème peuvent encore être vérifiées dans la suite de la résolution alors l'élément correspondant du vecteur VER est positionné à 1, dans le cas contraire si une seule des contraintes ne peut plus être vérifiées VER est positionné à 0.

1.5. Calcul de la borne de la fonction. Pour évaluer cette borne, nous additionnons les différents coefficients de la fonction économique qui correspondent à une variable positionnée à 1. La valeur de cette borne est placée

dans l'élément correspondant du vecteur VALZ.

Ainsi, si NBIND(i) = k

$$VALZ(i) = \sum_{j=1}^{k} Z(j) \quad \text{pour } j \mid IND(i,j) = 1$$

Exemple:

Soit à chercher:

$$\begin{cases} \text{Min } z = 8x_1 + 12x_2 + 15x_3 \\ \text{avec : } 12x_1 + 7x_2 - 4x_3 \geqslant 7 \\ x_i = 0 \text{ ou } 1 \quad \text{pour } i = 1,2,3 \end{cases}$$

Dans cet exemple simple les différents vecteurs ou matrice contiendront:

ITER	SUCCES	IND	NBIND	VALZ	VER	SOL
1	0	1 0 0	1	8	1	1
2	0	0 0 0	1	0	1	0
3	2	0 1 0	2	12	1	1
4	2	0 0 0	2	0	0	0

1.5. Conservation d'une meilleure solution. Lorsque nous avons obtenu une solution, c'est à dire lorsque SOL=1 nous la comparons à la meilleure solution déjà obtenue

et la meilleure des deux est conservée dans SOLL. La valeur des variables de cette solution est recopiée dans le vecteur INDSOL.

2. Recherche du "père" d'une nouvelle itération.

Puisque nous traitons un problème de minimisation nous allons prendre comme "père" d'une nouvelle itération un des bouts pendants non terminaux qui correspond à la valeur minimale des différentes bornes.

Pour obtenir ce "père", nous considérons successivement toutes les itérations. Parmi celles-ci nous ne retenons pas:

- celles qui ne sont pas bout pendant c'est à dire qui correspondent à un élément BOUPEN(i) = 0;
- celles qui sont bout pendant terminal vide ou non, c'est à dire celles pour lesquelles VER(i) = 0 ou SOL(i) = 1.
- celles qui sont bout pendant mais dont la borne correspondante est supérieure ou égale à la meilleure solution déjà obtenue.

Parmi celles restantes nous prenons une des itérations ayant la borne inférieure.

Une fois ce "père" trouvé, son numéro d'itération ITER(i) est recopié dans l'élément du vecteur SUCCES correspondant à la nouvelle itération que l'on va étudier. Nous retranscrivons ensuite l'ensemble des valeurs de ses variables dans la ligne de la matrice IND correspondant à la nouvelle itération envisagée. En fait nous recopions cet ensemble deux fois de suite, puisque successivement nous allons étudier les deux itérations descendantes de ce "père", qui correspondent aux valeurs 0 et 1 que va pouvoir prendre la variable à arbitrer. A présent nous indiquons que ce "père" est devenu bout non pendant en positionnant l'élément correspondant du vecteur BOUPEN à 0.

Nous reprenons alors l'algorithme au niveau 1.2.

3. Compactage des différents tableaux.

Lors de la résolution d'un problème nous ne connaissons pas le nombre d'itérations nécessaires à l'obtention d'une solution optimale. Aussi sommes nous contraint de donner une dimension à priori aux différents vecteurs et matrice utilisés.

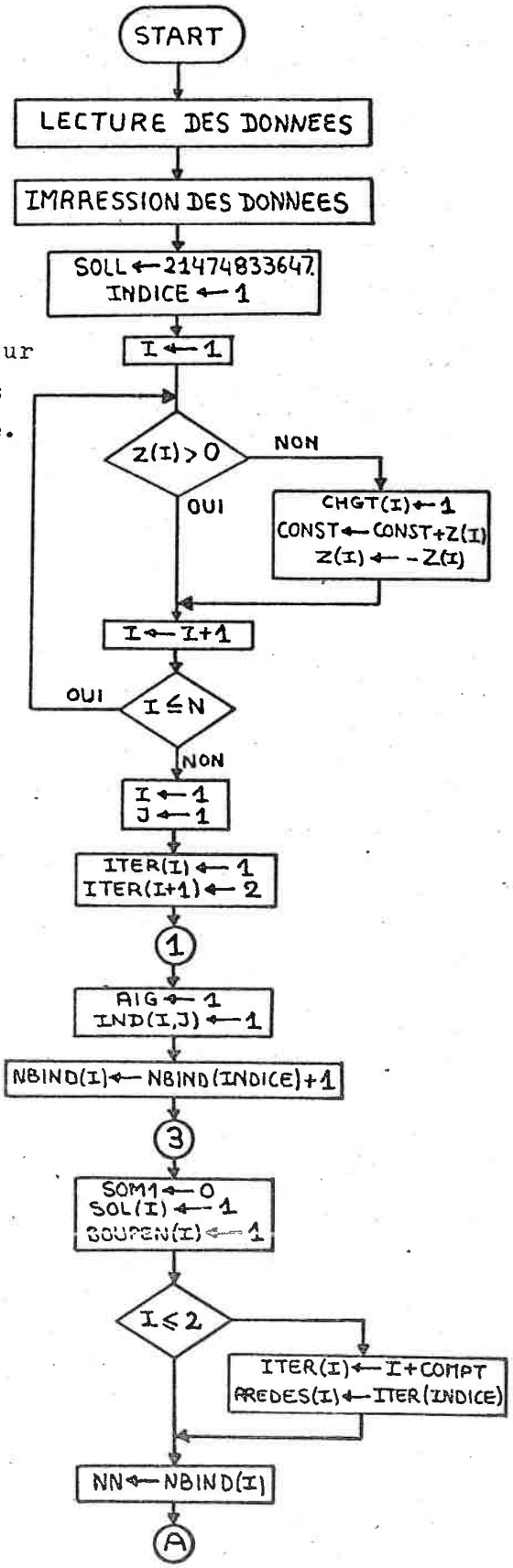
Cette dimension sera la plus grande possible en fonction de la taille mémoire de l'ordinateur utilisé et du nombre de variables du problème à résoudre.

Lors du déroulement de l'algorithme nous remplissons les différents vecteurs et matrice. Si nous atteignons la taille maximum de ces tableaux, nous les compactons en ne conservant que les bouts pendants non terminaux et intéressants pour la suite de la résolution. Toutes les autres itérations peuvent être détruites, même les solutions puisque la meilleure est conservée dans le vecteur INDSOL.

Cette technique de compactage est intéressante, car elle permet d'atténuer le problème posé par l'occupation mémoire de telle méthode de résolution de programme linéaire. En effet, pour qu'il y ait saturation de la mémoire, il faut que le nombre des bouts pendants non terminaux soit égal à la taille des différents tableaux.

En effet, nous avons pu constater, lors du traitement de certains exemples qu'une taille 300 des tableaux était suffisante pour résoudre des problèmes ayant une vingtaine de variables et nécessitant l'étude d'environ 6000 itérations.

d) Organigramme.

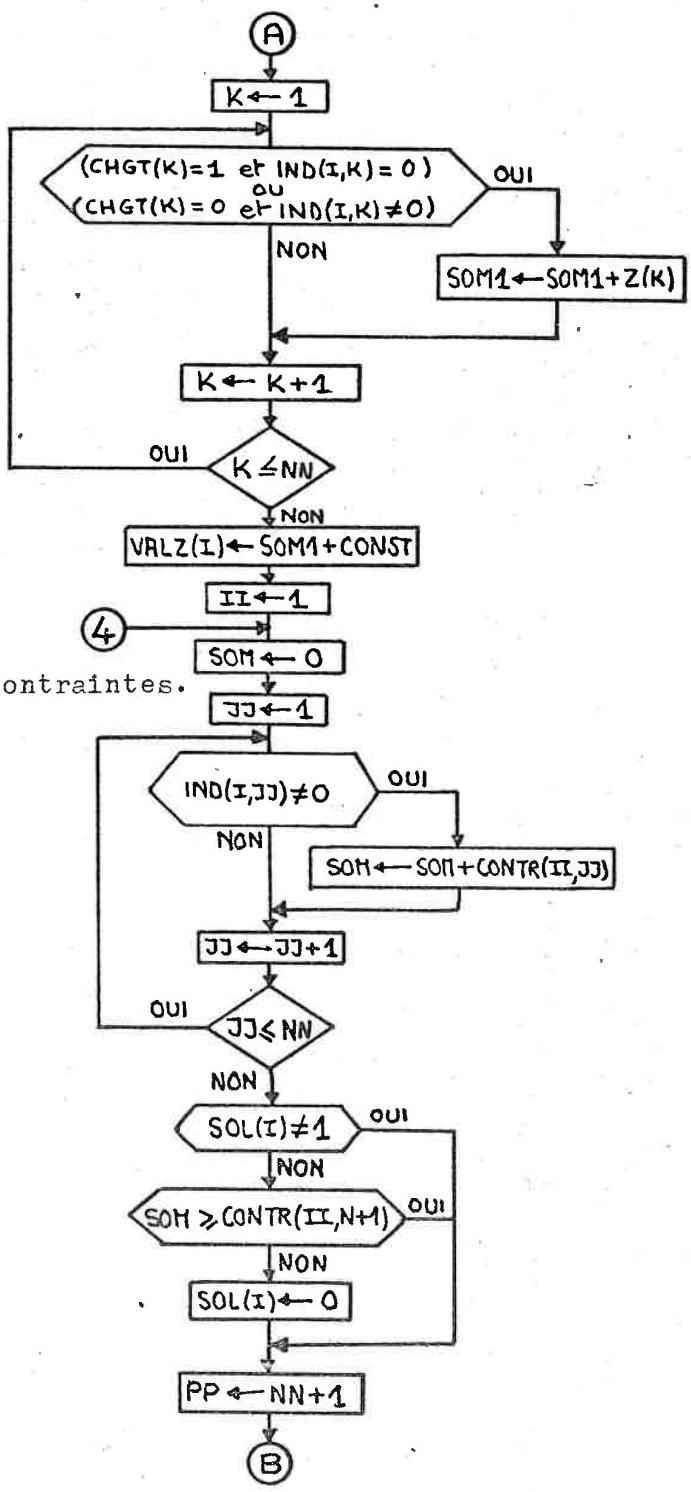


Changement de variable pour les coefficients négatifs de la fonction économique.

Phase d'initialisation.

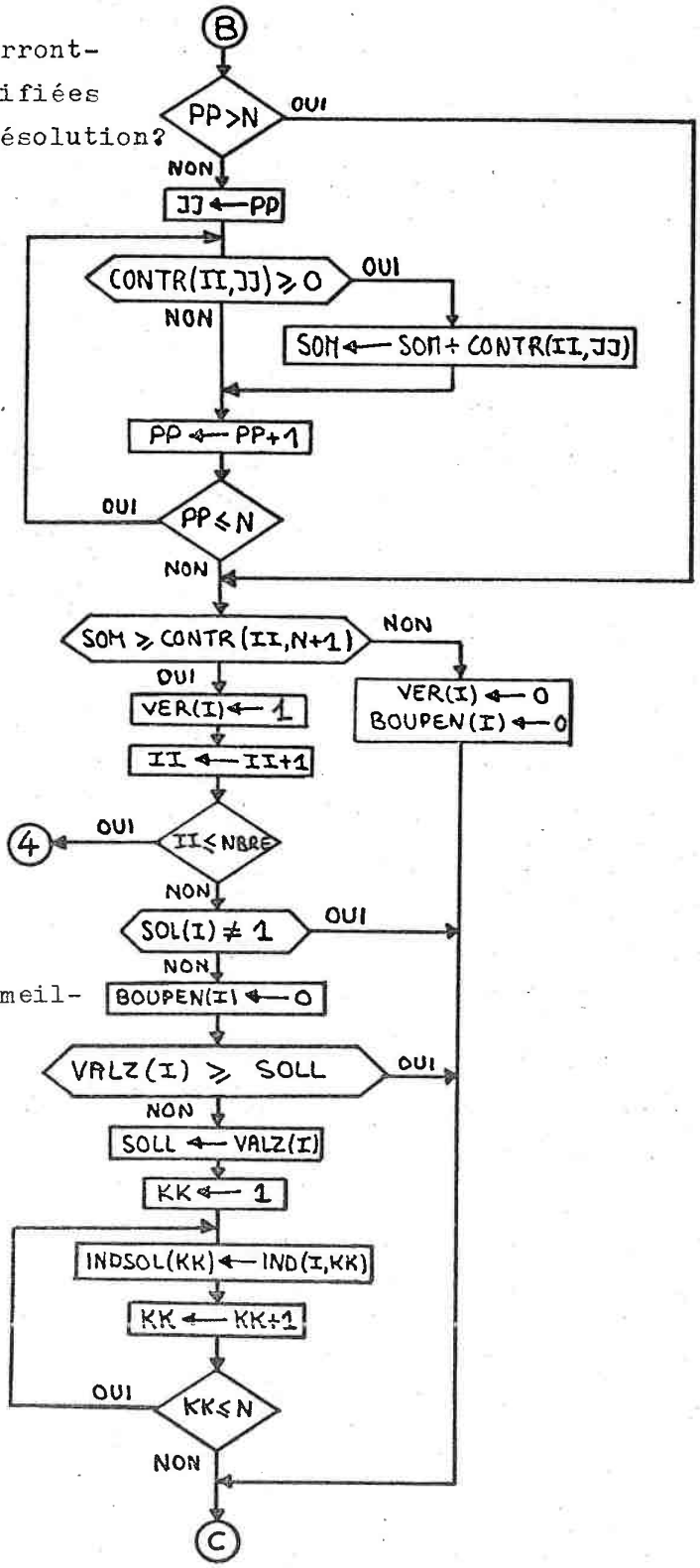
Phase de calcul.

-calcul de la borne.



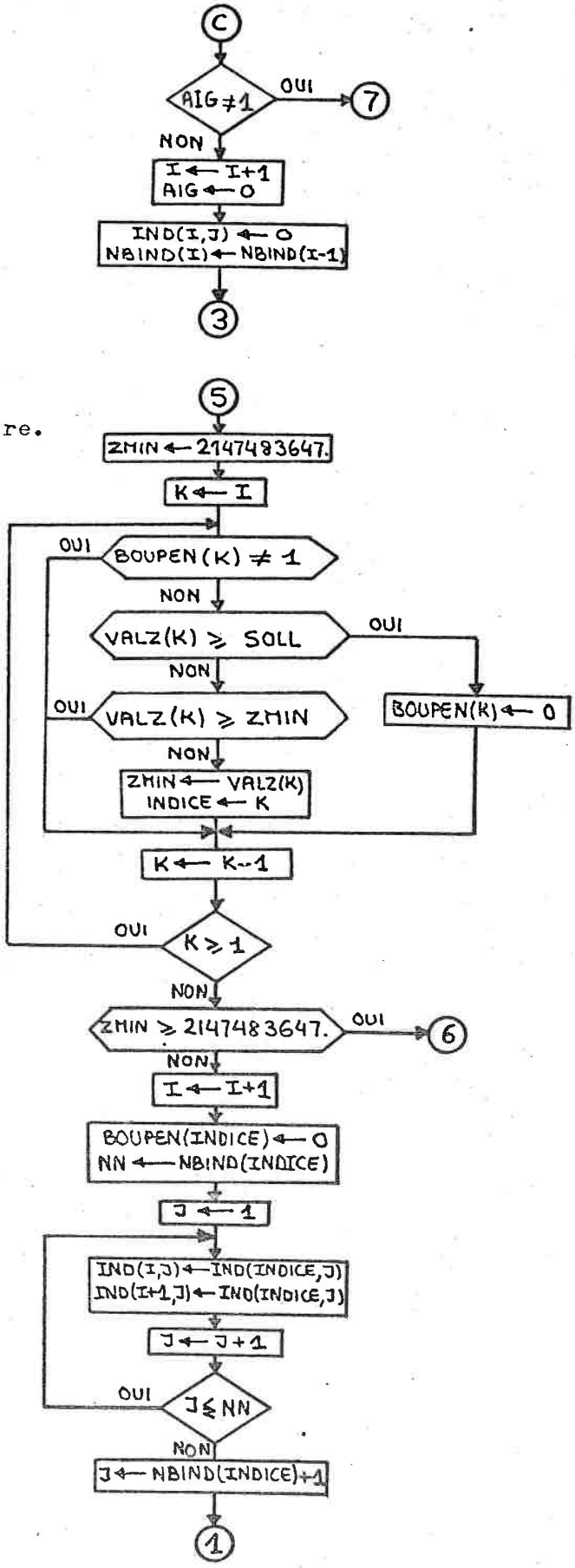
- vérification des contraintes.

- les contraintes pourront-elles encore être vérifiées dans la suite de la résolution?

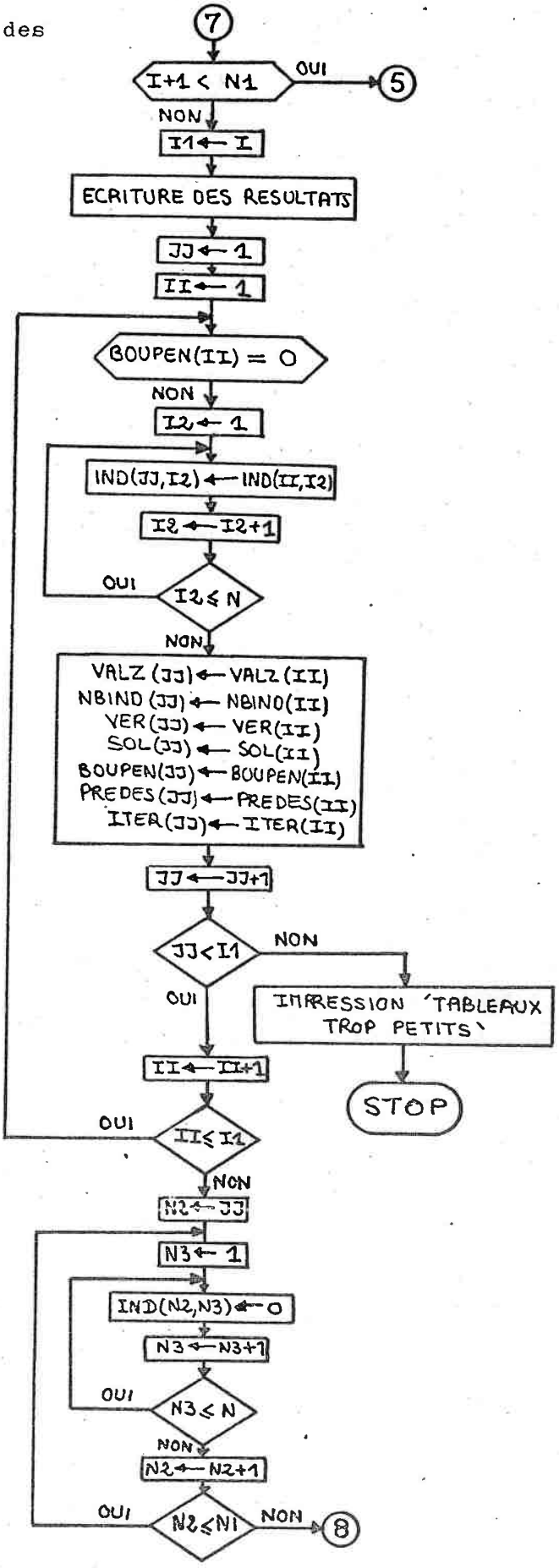


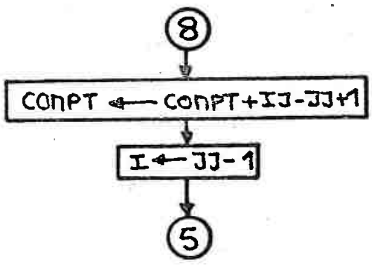
- conservation de la meilleure solution.

Phase de retour-arrière.

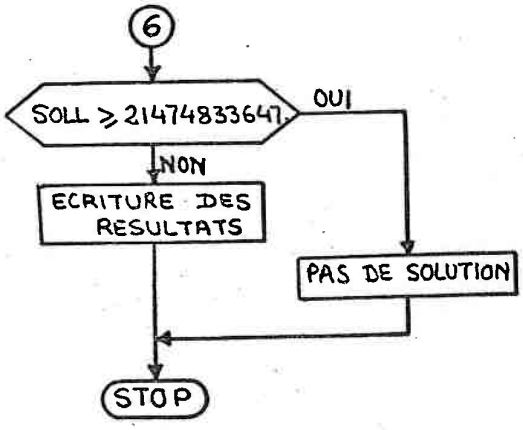


Phase de compactage des différents tableaux.





Test d'arrêt du problème.



e) Programme.

Le programme que nous présentons a été écrit en FORTRAN et mis au point sur un ordinateur UNIVAC 1110 (time sharing exec, multi-processor system).

L'énoncé du problème que nous résolvons est le suivant:

Minimiser $z = 2x_1-4x_2+6x_3+5x_4-3x_5+2x_6-4x_7-3x_8+9x_9+4x_{10}+3x_{11}$

sous les contraintes:

$3x_1+4x_2-3x_3+5x_4+8x_5+9x_6-4x_7+3x_8-2x_9+7x_{10}+6x_{11} \geq 27$

$-x_1+2x_2+4x_3+3x_4+7x_5-6x_6+3x_7-x_8+8x_9+3x_{10}+x_{11} \geq 18$

$x_1+x_2-x_3+x_4-3x_5+5x_6+2x_7-2x_8+x_9+3x_{10}+2x_{11} \geq 8$

pour $x_j = 0$ ou $1 \quad j = 1,\ldots,11$

Le temps total de traitement de cet exemple est de 15.568 sec., le temps de calcul étant de 4.574 sec..

```
@FOR,IS
FOR S11A-10/14/75-11:11:10
FORTRAN 11A INTER-PROJETS 1100 (ORSAY)

   MAIN PROGRAM

   STORAGE USED: CODE(1) 001122; DATA(0) 010377; BLANK COMMON(2) 000000


   EXTERNAL REFERENCES (BLOCK, NAME)

    0003   NINTR$
    0004   NRDU$
    0005   NTO2$
    0006   NTO1$
    0007   NWDU$
    0010   NSTOP$


   STORAGE ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0001   000204 1L        0001   000536 10L       0000   010216 100F      0000   010217 101F      0000   010244 102F
    0000   010254 103F      0000   010276 104F      0000   010311 105F      0000   010320 106F      0000   010330 107F
    0000   010341 108F      0000   010343 109F      0001   000027 112G      0001   000571 12L       0001   000063 123G
    0001   000070 127G      0001   000106 136G      0001   000123 145G      0001   000130 151G      0001   000405 15L
    0001   000150 162G      0001   000237 18L       0001   000264 213G      0001   000325 223G      0001   000335 227G
    0001   000772 23L       0001   000375 246G      0001   000631 26L       0001   000445 273G      0001   001116 33L
    0001   000216 3L        0001   001023 30L       0001   000466 303G      0001   000544 322G      0001   000520 345G
    0001   000570 35L       0001   001111 36L       0001   000663 363G      0001   000572 371G      0001   000714 404G
    0001   000733 411G      0001   001003 435G      0001   001004 440G      0001   000173 45L       0001   001051 455G
    0001   001060 463G      0001   001104 477G      0001   000421 5L        0001   000515 6L        0001   000350 9L
    0000 I 000000 AIG       0000 I 000456 BOUPEN     0000 I 001133 CHGT      0000 I 001132 COMPT      0000 I 001157 CONST
    0000 R 007550 CONTR     0000 I 010175 I          0000 I 010205 II        0000 I 001160 IND        0000 I 010202 INDICE
    0000 I 010147 INDSOL    0000 I 007721 ITER       0000 I 010212 I1        0000 I 010213 I2         0000 I 010200 J
    0000 I 010207 JJ        0000 I 010204 K          0000 I 010210 KK        0000 I 010174 N          0000 I 007276 NBIND
    0000 I 010176 NBRE      0000 I 010177 NN         0000 I 010173 N1        0000 I 010214 N2         0000 I 010215 N3
    0000 I 000455 PP        0000 I 000704 PREDES     0000 I 000227 SOL       0000 R 010201 SOLL       0000 R 010206 SOM
    0000 R 010203 SOM1      0000 R 007050 VALZ       0000 I 000001 VER       0000 R 007524 Z          0000 R 010211 ZMIN
```

```
00100      1*     C
00100      2*     C          PROGRAMMATION LINEAIRE EN VARIABLES BOOLEENNES
00100      3*     C   PAR LA METHODE DES SEPARATIONS ET DES EVALUATIONS PROGRESSIVES
00100      4*     C
00101      5*            INTEGER AIG,VER,SOL,PP,BOUPEN,PREDES,COMPT,CHGT,CONST
00103      6*            DIMENSION IND(150,20),VALZ(150),NBIND(150),VER(150),SOL(150),
00103      7*           *Z(20),CONTR(5,21),BOUPEN(150),PREDES(150),ITER(150),INDSOL(20),
00103      8*           *CHGT(20)
00104      9*            N1=150
00105     10*            READ(5,100) N
00110     11*            READ(5,108)(Z(I),I=1,N)
00116     12*            READ(5,100) NBRE
00121     13*            NN=N+1
00122     14*            DO 2 I=1,NBRE
00125     15*          2 READ(5,108)(CONTR(I,J),J=1,NN)
```

```
00134    16*          WRITE(6,104)(Z(I),I=1,N)
00142    17*          WRITE(6,105)
00144    18*          DO 22 I=1,NBRE
00147    19*       22 WRITE(6,106)((CONTR(I,J),J=1,N),CONTR(I,NN))
00157    20*          SOLL=21474833647.
00160    21*          INDICE=1
00160    22*    C
00160    23*    C      CHANGEMENT DE VARIABLE POUR LES COEFFICIENTS NEGATIFS DE LA
00160    24*    C                    FONCTION ECONOMIQUE
00160    25*    C
00161    26*          DO 45 I=1,N
00164    27*          IF(Z(I).GT.0) GO TO 45
00166    28*          CHGT(I)=1
00167    29*          CONST=CONST+Z(I)
00170    30*          Z(I)=-Z(I)
00171    31*       45 CONTINUE
00173    32*          I=1
00174    33*          J=1
00175    34*          ITER(I)=1
00176    35*          ITER(I+1)=2
00176    36*    C
00176    37*    C               PHASE DE CALCUL
00176    38*    C
00177    39*        1 IND(I,J)=1
00200    40*          AIG=1
00201    41*          NBIND(I)=NBIND(INDICE)+1
00202    42*        3 SOM1=0
00203    43*          BOUPEN(I)=1
00204    44*          SOL(I)=1
00205    45*          IF (I.LE.2) GO TO 18
00207    46*          PREDES(I)=ITER(INDICE)
00210    47*          ITER(I)=I+COMPT
00211    48*       18 NN=NBIND(I)
00212    49*          DO 7 K=1,NN
00215    50*          IF((CHGT(K).EQ.1.AND.IND(I,K).EQ.0).OR.(CHGT(K).EQ.0.AND.
00215    51*         *IND(I,K).NE.0)) SOM1=SOM1+Z(K)
00217    52*        7 CONTINUE
00221    53*          VALZ(I)=SOM1+CONST
00222    54*          DO 19 II=1,NBRE
00225    55*          SOM=0
00226    56*          DO 20 JJ=1,NN
00231    57*          IF (IND(I,JJ).NE.0) SOM=SOM+CONTR(II,JJ)
00233    58*       20 CONTINUE
00235    59*          IF (SOL(I).NE.1) GO TO 9
00237    60*          IF (SOM.GE.CONTR(II,N+1)) GO TO 9
00241    61*          SOL(I)=0
00242    62*        9 PP=NN+1
00243    63*          IF (PP.GT.N) GO TO 16
00245    64*          DO 4 JJ=PP,N
00250    65*          IF(CONTR(II,JJ).GE.0) SOM=SOM+CONTR(II,JJ)
00252    66*        4 CONTINUE
00254    67*       16 IF (SOM.GE.CONTR(II,N+1)) GO TO 5
00256    68*          VER(I)=0
00257    69*          BOUPEN(I)=0
00260    70*          GO TO 6
00261    71*        5 VER(I)=1
00262    72*       19 CONTINUE
```

```
00264    73*          IF (SOL(I).NE.1) GO TO 6
00266    74*          BOUPEN(I)=0
00267    75*          IF (VALZ(I).GE.SOLL) GO TO 6
00271    76*          SOLL=VALZ(I)
00272    77*          DO 31 KK=1,NN
00275    78*       31 INDSOL(KK)=IND(I,KK)
00277    79*          IF(NN.EQ.N) GO TO 6
00301    80*          NN=NN+1
00302    81*          DO 32 KK=NN,N
00305    82*          IF((CHGT(KK).EQ.1.AND.IND(I,KK).EQ.0).OR.(CHGT(KK).EQ.0.AND.
                     *IND(I,KK).NE.0)) INDSOL(KK)=1
00307    84*       32 CONTINUE
00311    85*        6 IF (AIG.NE.1) GO TO 26
00313    86*          I=I+1
00314    87*          AIG=0
00315    88*          IND(I,J)=0
00316    89*          NBIND(I)=NBIND(I-1)
00317    90*          GO TO 3
00317    91*    C
00317    92*    C   RECHERCHE DU POINT DE DEPART D'UNE NOUVELLE ITERATION (BORNE INF)
00317    93*    C
00320    94*       10 ZMIN=2147483647.
00321    95*          DO 12 K=I,1,-1
00324    96*          IF (BOUPEN(K).NE.1) GO TO 12
00326    97*          IF (VALZ(K).GE.SOLL) GO TO 35
00330    98*          IF (VALZ(K).GE.ZMIN) GO TO 12
00332    99*          ZMIN=VALZ(K)
00333   100*          INDICE=K
00334   101*          GO TO 12
00335   102*       35 BOUPEN(K)=0
00336   103*       12 CONTINUE
00340   104*          IF (ZMIN.GE.214783647.) GO TO 30
00342   105*          BOUPEN(INDICE)=0
00343   106*          I=I+1
00344   107*          NN=NBIND(INDICE)
00345   108*          DO 13 J=1,NN
00350   109*          IND(I,J)=IND(INDICE,J)
00351   110*       13 IND(I+1,J)=IND(INDICE,J)
00353   111*          J=NBIND(INDICE)+1
00354   112*          GO TO 1
00354   113*    C
00354   114*    C        COMPACTAGE DES DIFFERENTS TABLEAUX DU PROGRAMME
00354   115*    C      SEUL LES BOUTS PENDANTS NON TERMINAUX SONT CONSERVES
00354   116*    C
00355   117*       26 IF(I+1.LT.N1) GO TO 10
00357   118*          I1=I
00360   119*          WRITE(6,101)
00362   120*          DO 25 I=1,I1
00365   121*       25 WRITE(6,102)(ITER(I),PREDES(I),(IND(I,J),J=1,N),VALZ(I),NBIND(I),
                     *VER(I),SOL(I))
00402   123*          JJ=1
00403   124*          DO 23 II=1,I1
00406   125*          IF (BOUPEN(II).EQ.0) GO TO 23
00410   126*          DO 24 I2=1,N
00413   127*       24 IND(JJ,I2)=IND(II,I2)
00415   128*          VALZ(JJ)=VALZ(II)
00416   129*          NBIND(JJ)=NBIND(II)
```

```
00417    130*            VER(JJ)=VER(II)
00420    131*            SOL(JJ)=SOL(II)
00421    132*            BOUPEN(JJ)=BOUPEN(II)
00422    133*            PREDES(JJ)=PREDES(II)
00423    134*            ITER(JJ)=ITER(II)
00424    135*            JJ=JJ+1
00425    136*            IF (JJ.LT.I1) GO TO 23
00427    137*            WRITE(6,107)
00431    138*            GO TO 28
00432    139*         23 CONTINUE
00434    140*            DO 27 N2=JJ,N1
00437    141*            DO 27 N3=1,N
00442    142*         27 IND(N2,N3)=0
00445    143*            COMPT=COMPT+I1-JJ+1
00446    144*            I=JJ-1
00447    145*            GO TO 10
00450    146*         30 IF (SOLL.GE.21474833647.) GO TO 36
00452    147*            WRITE(6,101)
00454    148*            DO 14 K=1,I
00457    149*         14 WRITE(6,102)(ITER(K),PREDES(K),(IND(K,J),J=1,N),VAL7(K),NBIND(K),
00457    150*           *VER(K),SOL(K))
00474    151*            WRITE(6,103)(SOLL,(INDSOL(I),I=1,N))
00503    152*            GO TO 28
00504    153*         36 WRITE(6,109)
00506    154*         28 STOP
00507    155*        100 FORMAT(I2)
00510    156*        101 FORMAT('0',//,10X,'ITERATION',3X,'SUCCESSEUR DE',8X,'VALEUR DES VA
00510    157*           *RIABLES',8X,'VAL7',9X,'NBIND',6X,'VER',6X,'SOL')
00511    158*        102 FORMAT(14X,I3,11X,I3,5X,11(I2,1X),2X,E7.2,8X,4(I2,7X))
00512    159*        103 FORMAT('0',//,10X,'L''OPTIMUM DE LA FONCTION EST Z = ',E7.2,
00512    160*           *'   OBTENU POUR LES VALEURS  ',20(I1,','))
00513    161*        104 FORMAT('1',//,10X,'FONCTION A MINIMISER  MIN Z =  ',11(E7.2,'X',))
00514    162*        105 FORMAT('0',//,20X,' SOUS LES CONTRAINTES')
00515    163*        106 FORMAT('0',/,10X,11(E7.2,'X'),' > OU = A ',E7.2)
00516    164*        107 FORMAT('0',//,10X,'DIMENSION DES TABLEAUX TROP PETITES')
00517    165*        108 FORMAT(25E5.0)
00520    166*        109 FORMAT('0',//,10X,'LE PROBLEME N''ADMET PAS DE SOLUTION')
00521    167*            END

      END OF COMPILATION:        NO  DIAGNOSTICS.

@XQT
MAP 26.2-10/14-11:11


ADDRESS LIMITS    001000 013425      5398 IBANK WORDS DECIMAL
                  040000 055175      6782 DBANK WORDS DECIMAL
STARTING ADDRESS  012304



          SEGMENT $MAIN$          001000 013425     040000 055175

NS*TCS/FOR69              $(1)  001000 001024
```

FONCTION A MINIMISER MIN Z = .20+01,-.40+01, .60+01, .50+01,-.30+01, .20+01,-.40+01,-.30+01, .90+01, .40+01, .30+01,

SOUS LES CONTRAINTES

.30+01, .40+01,-.30+01, .50+01, .60+01, .90+01,-.40+01, .30+01,-.20+01, .70+01, .60+01, > OU = A .27+02

-.10+01, .20+01, .40+01, .30+01, .70+01,-.60+01, .30+01,-.10+01, .80+01, .30+01, .10+01, > OU = A .16+02

.10+01, .10+01,-.10+01, .10+01,-.30+01, .50+01, .20+01,-.20+01, .10+01, .30+01, .20+01, > OU = A .80+01

ITERATION	SUCCESSEUR DE	VALEUR DES VARIABLES										VALZ	NBIND	VER	SOL
1	0	1	0	0	0	0	0	0	0	0	0	-.12+02	1	1	0
2	0	0	0	0	0	0	0	0	0	0	0	-.14+02	1	1	0
3	2	0	1	0	0	0	0	0	0	0	0	-.14+02	2	1	0
4	2	0	0	0	0	0	0	0	0	0	0	-.10+02	2	1	0
5	3	0	1	1	0	0	0	0	0	0	0	-.80+01	3	1	0
6	3	0	1	0	0	0	0	0	0	0	0	-.14+02	3	1	0
7	6	0	1	0	1	0	0	0	0	0	0	-.90+01	4	1	0
8	6	0	1	0	0	0	0	0	0	0	0	-.14+02	4	1	0
9	8	0	1	0	0	1	0	0	0	0	0	-.14+02	5	1	0
10	8	0	1	0	0	0	0	0	0	0	0	-.11+02	5	0	0
11	9	0	1	0	0	1	1	0	0	0	0	-.12+02	6	1	0
12	9	0	1	0	0	1	0	0	0	0	0	-.14+02	6	0	0
13	11	0	1	0	0	1	1	1	0	0	0	-.12+02	7	1	0
14	11	0	1	0	0	1	1	0	0	0	0	-.80+01	7	0	0
15	13	0	1	0	0	1	1	1	1	0	0	-.12+02	8	0	0
16	13	0	1	0	0	1	1	1	0	0	0	-.90+01	8	1	0
17	1	1	1	0	0	0	0	0	0	0	0	-.12+02	2	1	0
18	1	1	0	0	0	0	0	0	0	0	0	-.80+01	2	1	0
19	17	1	1	1	0	0	0	0	0	0	0	-.60+01	3	1	0
20	17	1	1	0	0	0	0	0	0	0	0	-.12+02	3	1	0
21	20	1	1	0	1	0	0	0	0	0	0	-.70+01	4	1	0
22	20	1	1	0	0	0	0	0	0	0	0	-.12+02	4	1	0
23	22	1	1	0	0	1	0	0	0	0	0	-.12+02	5	1	0
24	22	1	1	0	0	0	0	0	0	0	0	-.90+01	5	0	0
25	23	1	1	0	0	1	1	0	0	0	0	-.10+02	6	0	0
26	23	1	1	0	0	1	0	0	0	0	0	-.12+02	6	0	0
27	4	0	0	1	0	0	0	0	0	0	0	-.40+01	3	1	0
28	4	0	0	0	0	0	0	0	0	0	0	-.10+02	3	1	0
29	28	0	0	0	1	0	0	0	0	0	0	-.50+01	4	1	0
30	28	0	0	0	0	0	0	0	0	0	0	-.10+02	4	1	0
31	30	0	0	0	0	1	0	0	0	0	0	-.10+02	5	1	0
32	30	0	0	0	0	0	0	0	0	0	0	-.70+01	5	0	0
33	31	0	0	0	0	1	1	0	0	0	0	-.80+01	6	0	0
34	31	0	0	0	0	1	0	0	0	0	0	-.10+02	6	0	0
35	16	0	1	0	0	1	1	1	0	1	0	.00	9	1	0
36	16	0	1	0	0	1	1	1	0	0	0	-.90+01	9	0	0
37	7	0	1	0	1	1	0	0	0	0	0	-.90+01	5	1	0

38	7	0	1	0	1	0	0	0	0	0	0	0	-.60+01	5	1	0
39	37	0	1	0	1	1	1	0	0	0	0	0	-.70+01	6	1	0
40	37	0	1	0	1	1	0	0	0	0	0	0	-.90+01	6	0	0
41	19	1	0	1	0	0	0	0	0	0	0	0	-.20+01	3	1	0
42	19	1	0	0	0	0	0	0	0	0	0	0	-.80+01	3	1	0
43	42	1	0	0	1	0	0	0	0	0	0	0	-.30+01	4	1	0
44	42	1	0	0	0	0	0	0	0	0	0	0	-.80+01	4	1	0
45	44	1	0	0	0	1	0	0	0	0	0	0	-.80+01	5	1	0
46	44	1	0	0	0	0	0	0	0	0	0	0	-.50+01	5	0	0
47	45	1	0	0	0	1	1	0	0	0	0	0	-.60+01	6	0	0
48	45	1	0	0	0	1	0	0	0	0	0	0	-.80+01	6	0	0
49	5	0	1	1	1	0	0	0	0	0	0	0	-.30+01	4	1	0
50	5	0	1	1	0	0	0	0	0	0	0	0	-.80+01	4	1	0
51	50	0	1	1	0	1	0	0	0	0	0	0	-.80+01	5	1	0
52	50	0	1	1	0	0	0	0	0	0	0	0	-.50+01	5	0	0
53	51	0	1	1	0	1	1	0	0	0	0	0	-.60+01	6	1	0
54	51	0	1	1	0	1	0	0	0	0	0	0	-.80+01	6	0	0
55	39	0	1	0	1	1	1	1	0	0	0	0	-.70+01	7	1	0
56	39	0	1	0	1	1	1	0	0	0	0	0	-.30+01	7	1	0
57	55	0	1	0	1	1	1	1	1	0	0	0	-.70+01	8	1	0
58	55	0	1	0	1	1	1	1	0	0	0	0	-.40+01	8	1	0
59	57	0	1	0	1	1	1	1	1	1	0	0	.20+01	9	1	0
60	57	0	1	0	1	1	1	1	1	0	0	0	-.70+01	9	0	0
61	21	1	1	0	1	1	0	0	0	0	0	0	-.70+01	5	1	0
62	21	1	1	0	1	0	0	0	0	0	0	0	-.40+01	5	1	0
63	61	1	1	0	1	1	1	0	0	0	0	0	-.50+01	6	1	0
64	61	1	1	0	1	1	0	0	0	0	0	0	-.70+01	6	1	0
65	64	1	1	0	1	1	0	1	0	0	0	0	-.70+01	7	1	0
66	64	1	1	0	1	1	0	0	0	0	0	0	-.30+01	7	0	0
67	65	1	1	0	1	1	0	1	1	0	0	0	-.70+01	8	0	0
68	65	1	1	0	1	1	0	1	0	0	0	0	-.40+01	8	1	0
69	53	0	1	1	0	1	1	1	0	0	0	0	-.60+01	7	1	0
70	53	0	1	1	0	1	1	0	0	0	0	0	-.20+01	7	1	0
71	69	0	1	1	0	1	1	1	1	0	0	0	-.60+01	8	1	0
72	69	0	1	1	0	1	1	1	0	0	0	0	-.30+01	8	1	0
73	71	0	1	1	0	1	1	1	1	1	0	0	.30+01	9	1	0
74	71	0	1	1	0	1	1	1	1	0	0	0	-.60+01	9	0	0
75	38	0	1	0	1	0	1	0	0	0	0	0	-.40+01	6	0	0
76	38	0	1	0	1	0	0	0	0	0	0	0	-.60+01	6	0	0
77	19	1	1	1	1	0	0	0	0	0	0	0	-.10+01	4	1	0
78	19	1	1	1	0	0	0	0	0	0	0	0	-.60+01	4	1	0
79	78	1	1	1	0	1	0	0	0	0	0	0	-.60+01	5	1	0
80	78	1	1	1	0	0	0	0	0	0	0	0	-.30+01	5	1	0
81	79	1	1	1	0	1	1	0	0	0	0	0	-.40+01	6	1	0
82	79	1	1	1	0	1	0	0	0	0	0	0	-.60+01	6	0	0
83	63	1	1	0	1	1	1	0	0	0	0	0	-.50+01	7	1	0
84	63	1	1	0	1	1	1	0	0	0	0	0	-.10+01	7	0	0
85	83	1	1	0	1	1	1	1	1	0	0	0	-.50+01	8	1	0
86	83	1	1	0	1	1	1	1	0	0	0	0	-.20+01	8	1	0
87	85	1	1	0	1	1	1	1	1	1	0	0	.40+01	9	1	0
88	85	1	1	0	1	1	1	1	1	0	0	0	-.50+01	9	0	0
89	29	0	0	0	1	1	0	0	0	0	0	0	-.50+01	5	1	0
90	29	0	0	0	1	0	0	0	0	0	0	0	-.20+01	5	1	0
91	89	0	0	0	1	1	1	0	0	0	0	0	-.30+01	6	1	0
92	89	0	0	0	1	1	0	0	0	0	0	0	-.50+01	6	0	0
93	81	1	1	1	0	1	1	1	0	0	0	0	-.40+01	7	1	0
94	81	1	1	1	0	1	1	0	0	0	0	0	.00	7	1	0

95	93	1	1	1	0	1	1	1	1	0	0	0	-.40+01	8	1	0
96	93	1	1	1	0	1	1	1	0	0	0	0	-.10+01	8	1	0
97	95	1	1	1	0	1	1	1	1	1	0	0	.50+01	9	1	0
98	95	1	1	1	0	1	1	1	1	0	0	0	-.40+01	9	0	0
99	68	1	1	0	1	1	0	1	0	1	0	0	.50+01	9	1	0
100	68	1	1	0	1	1	0	1	0	0	0	0	-.40+01	9	0	0
101	62	1	1	0	1	0	1	0	0	0	0	0	-.20+01	6	0	0
102	62	1	1	0	1	0	0	0	0	0	0	0	-.40+01	6	1	0
103	102	1	1	0	1	0	0	1	0	0	0	0	-.40+01	7	0	0
104	102	1	1	0	1	0	0	0	0	0	0	0	.00	7	0	0
105	58	0	1	0	1	1	1	1	0	1	0	0	.50+01	9	1	0
106	58	0	1	0	1	1	1	1	0	0	0	0	-.40+01	9	0	0
107	27	0	0	1	1	0	0	0	0	0	0	0	.10+01	4	1	0
108	27	0	0	1	0	0	0	0	0	0	0	0	-.40+01	4	1	0
109	108	0	0	1	0	1	0	0	0	0	0	0	-.40+01	5	1	0
110	108	0	0	1	0	0	0	0	0	0	0	0	-.10+01	5	0	0
111	109	0	0	1	0	1	1	0	0	0	0	0	-.20+01	6	1	0
112	109	0	0	1	0	1	0	0	0	0	0	0	-.40+01	6	0	0
113	91	0	0	0	1	1	1	1	0	0	0	0	-.30+01	7	1	0
114	91	0	0	0	1	1	1	0	0	0	0	0	.10+01	7	0	0
115	113	0	0	0	1	1	1	1	1	0	0	0	-.30+01	8	1	0
116	113	0	0	0	1	1	1	1	0	0	0	0	.00	8	1	0
117	115	0	0	0	1	1	1	1	1	1	0	0	.60+01	9	1	0
118	115	0	0	0	1	1	1	1	1	0	0	0	-.30+01	9	0	0
119	80	1	1	1	0	0	1	0	0	0	0	0	-.10+01	6	0	0
120	80	1	1	1	0	0	0	0	0	0	0	0	-.30+01	6	0	0
121	72	0	1	1	0	1	1	1	0	1	0	0	.60+01	9	0	0
122	72	0	1	1	0	1	1	1	0	0	0	0	-.30+01	9	0	0
123	56	0	1	0	1	1	1	0	1	0	0	0	-.30+01	8	0	0
124	56	0	1	0	1	1	1	0	0	0	0	0	.00	8	1	0
125	49	0	1	1	1	1	0	0	0	0	0	0	-.30+01	5	1	0
126	49	0	1	1	1	0	0	0	0	0	0	0	.00	5	1	0
127	125	0	1	1	1	1	1	0	0	0	0	0	-.10+01	6	1	0
128	125	0	1	1	1	1	0	0	0	0	0	0	-.30+01	6	0	0
129	43	1	0	0	1	1	0	0	0	0	0	0	-.30+01	5	1	0
130	43	1	0	0	1	0	0	0	0	0	0	0	.00	5	0	0
131	129	1	0	0	1	1	1	0	0	0	0	0	-.10+01	6	1	0
132	129	1	0	0	1	1	0	0	0	0	0	0	-.30+01	6	0	0
133	111	0	0	1	0	1	1	1	0	0	0	0	-.20+01	7	0	0
134	111	0	0	1	0	1	1	0	0	0	0	0	.20+01	7	0	0
135	90	0	0	0	1	0	1	0	0	0	0	0	.00	6	0	0
136	90	0	0	0	1	0	0	0	0	0	0	0	-.20+01	6	0	0
137	86	1	1	0	1	1	1	1	0	1	0	0	.70+01	9	1	0
138	86	1	1	0	1	1	1	1	0	0	0	0	-.20+01	9	0	0
139	70	0	1	1	0	1	1	0	1	0	0	0	-.20+01	8	0	0
140	70	0	1	1	0	1	1	0	0	0	0	0	.10+01	8	1	0
141	41	1	0	1	1	0	0	0	0	0	0	0	.30+01	4	1	0
142	41	1	0	1	0	0	0	0	0	0	0	0	-.20+01	4	1	0
143	142	1	0	1	0	1	0	0	0	0	0	0	-.20+01	5	1	0
144	142	1	0	1	0	0	0	0	0	0	0	0	.10+01	5	0	0
145	143	1	0	1	0	1	1	0	0	0	0	0	.00	6	1	0
146	143	1	0	1	0	1	0	0	0	0	0	0	-.20+01	6	0	0
147	131	1	0	0	1	1	1	1	0	0	0	0	-.10+01	7	1	0
148	131	1	0	0	1	1	1	0	0	0	0	0	.30+01	7	0	0
149	147	1	0	0	1	1	1	1	1	0	0	0	-.10+01	8	0	0
150	147	1	0	0	1	1	1	1	0	0	0	0	.20+01	8	1	0

ITERATION	SUCCESSEUR DE	VALEUR DES VARIABLES											VALZ	NBIND	VER	SOL
35	16	0	1	0	0	1	1	1	0	1	0	0	.00	9	1	0
59	57	0	1	0	1	1	1	1	1	1	0	0	.20+01	9	1	0
73	71	0	1	1	0	1	1	1	1	1	0	0	.30+01	9	1	0
77	19	1	1	1	1	0	0	0	0	0	0	0	-.10+01	4	1	0
87	85	1	1	0	1	1	1	1	1	1	0	0	.40+01	9	1	0
94	81	1	1	1	0	1	1	0	0	0	0	0	.00	7	1	0
96	93	1	1	1	0	1	1	1	0	0	0	0	-.10+01	8	1	0
97	95	1	1	1	0	1	1	1	0	1	0	0	.50+01	9	1	0
99	68	1	1	0	1	1	0	1	0	1	0	0	.50+01	9	1	0
105	58	0	1	0	1	1	1	1	0	1	0	0	.50+01	9	1	0
107	27	0	0	1	1	0	0	0	0	0	0	0	.10+01	4	1	0
116	113	0	0	0	1	1	1	1	0	0	0	0	.00	8	1	0
117	115	0	0	0	1	1	1	1	1	1	0	0	.60+01	9	1	0
124	56	0	1	0	1	1	1	0	0	0	0	0	.00	8	1	0
126	49	0	1	1	1	0	0	0	0	0	0	0	.00	5	1	0
127	125	0	1	1	1	1	1	0	0	0	0	0	-.10+01	6	1	0
137	84	1	1	0	1	1	1	1	0	1	0	0	.70+01	9	1	0
140	70	0	1	1	0	1	1	0	0	0	0	0	.10+01	8	1	0
141	41	1	0	1	1	0	0	0	0	0	0	0	.30+01	4	1	0
145	143	1	0	1	0	1	1	0	0	0	0	0	.00	6	1	0
150	147	1	0	0	1	1	1	1	0	0	0	0	.20+01	8	1	0
151	127	0	1	1	1	1	1	1	0	0	0	0	-.10+01	7	1	0
152	127	0	1	1	1	1	1	0	0	0	0	0	.30+01	7	1	0
153	151	0	1	1	1	1	1	1	1	0	0	0	-.10+01	8	1	0
154	151	0	1	1	1	1	1	1	0	0	0	0	.20+01	8	1	0
155	153	0	1	1	1	1	1	1	1	1	0	0	.80+01	9	1	0
156	153	0	1	1	1	1	1	1	1	0	0	0	-.10+01	9	0	0
157	96	1	1	1	0	1	1	1	0	1	0	0	.80+01	9	1	0
158	96	1	1	1	0	1	1	1	0	0	0	0	-.10+01	9	0	0
159	77	1	1	1	1	1	0	0	0	0	0	0	-.10+01	5	1	0
160	77	1	1	1	1	0	0	0	0	0	0	0	.20+01	5	1	0
161	159	1	1	1	1	1	1	0	0	0	0	0	.10+01	6	1	0
162	159	1	1	1	1	1	0	0	0	0	0	0	-.10+01	6	0	0
163	145	1	0	1	0	1	1	1	0	0	0	0	.00	7	1	0
164	145	1	0	1	0	1	1	0	0	0	0	0	.40+01	7	0	0
165	163	1	0	1	0	1	1	1	1	0	0	0	.00	8	1	0
166	163	1	0	1	0	1	1	1	0	0	0	0	.30+01	8	0	0
167	165	1	0	1	0	1	1	1	1	1	0	0	.90+01	9	1	0
168	165	1	0	1	0	1	1	1	1	0	0	0	.00	9	0	0
169	126	0	1	1	1	0	1	0	0	0	0	0	.20+01	6	1	0
170	126	0	1	1	1	0	0	0	0	0	0	0	.00	6	0	0
171	124	0	1	0	1	1	1	0	0	1	0	0	.90+01	9	1	0
172	124	0	1	0	1	1	1	0	0	0	0	0	.00	9	0	0
173	116	0	0	0	1	1	1	1	0	1	0	0	.90+01	9	1	0
174	116	0	0	0	1	1	1	1	0	0	0	0	.00	9	0	0
175	94	1	1	1	0	1	1	0	1	0	0	0	.00	8	0	0
176	94	1	1	1	0	1	1	0	0	0	0	0	.30+01	8	1	0
177	35	0	1	0	0	1	1	1	0	1	1	0	.40+01	10	1	0
178	35	0	1	0	0	1	1	1	0	1	0	0	.00	10	0	0
179	161	1	1	1	1	1	1	1	0	0	0	0	.10+01	7	1	0
180	161	1	1	1	1	1	1	0	0	0	0	0	.50+01	7	1	0
181	179	1	1	1	1	1	1	1	1	0	0	0	.10+01	8	1	0
182	179	1	1	1	1	1	1	1	0	0	0	0	.40+01	8	1	0
183	181	1	1	1	1	1	1	1	1	1	0	0	.10+02	9	1	0

184	181	1	1	1	1	1	1	1	1	0	0	0	.10+01	9	0	0
185	140	0	1	1	0	1	1	0	0	1	0	0	.10+02	9	1	0
186	140	0	1	1	0	1	1	0	0	0	0	0	.10+01	9	0	0
187	107	0	0	1	1	1	0	0	0	0	0	0	.10+01	5	1	0
188	107	0	0	1	1	0	0	0	0	0	0	0	.40+01	5	1	0
189	187	0	0	1	1	1	1	0	0	0	0	0	.30+01	6	1	0
190	187	0	0	1	1	1	0	0	0	0	0	0	.10+01	6	0	0
191	169	0	1	1	1	0	1	1	0	0	0	0	.20+01	7	1	0
192	169	0	1	1	1	0	1	0	0	0	0	0	.60+01	7	0	0
193	191	0	1	1	1	0	1	1	1	0	0	0	.20+01	8	0	0
194	191	0	1	1	1	0	1	1	0	0	0	0	.60+01	8	0	0
195	160	1	1	1	1	0	1	0	0	0	0	0	.40+01	6	0	0
196	160	1	1	1	1	0	0	0	0	0	0	0	.20+01	6	0	0
197	154	0	1	1	1	1	1	1	0	1	0	0	.11+02	9	1	0
198	154	0	1	1	1	1	1	1	0	0	0	0	.20+01	9	0	0
199	150	1	0	0	1	1	1	1	0	1	0	0	.11+02	9	1	0
200	150	1	0	0	1	1	1	1	0	0	0	0	.20+01	9	0	0
201	59	0	1	0	1	1	1	1	1	1	1	0	.60+01	10	1	1
202	59	0	1	0	1	1	1	1	1	1	0	0	.20+01	10	0	0
203	189	0	0	1	1	1	1	1	0	0	0	0	.30+01	7	1	0
204	189	0	0	1	1	1	1	0	0	0	0	0	.70+01	7	1	0
205	203	0	0	1	1	1	1	1	1	0	0	0	.30+01	8	1	0
206	203	0	0	1	1	1	1	1	0	0	0	0	.60+01	8	1	0
207	205	0	0	1	1	1	1	1	1	1	0	0	.12+02	9	1	0
208	205	0	0	1	1	1	1	1	1	0	0	0	.30+01	9	0	0
209	176	1	1	1	0	1	1	0	0	1	0	0	.12+02	9	1	0
210	176	1	1	1	0	1	1	0	0	0	0	0	.30+01	9	0	0
211	152	0	1	1	1	1	1	0	1	0	0	0	.30+01	8	0	0
212	152	0	1	1	1	1	1	0	0	0	0	0	.60+01	8	1	0
213	141	1	0	1	1	1	0	0	0	0	0	0	.30+01	5	1	0
214	141	1	0	1	1	0	0	0	0	0	0	0	.60+01	5	1	0
215	213	1	0	1	1	1	1	0	0	0	0	0	.50+01	6	1	0
216	213	1	0	1	1	1	0	0	0	0	0	0	.30+01	6	0	0
217	73	0	1	1	0	1	1	1	1	1	1	0	.70+01	10	1	0
218	73	0	1	1	0	1	1	1	1	1	0	0	.30+01	10	0	0
219	188	0	0	1	1	0	1	0	0	0	0	0	.60+01	6	0	0
220	188	0	0	1	1	0	0	0	0	0	0	0	.40+01	6	0	0
221	182	1	1	1	1	1	1	1	0	1	0	0	.13+02	9	1	0
222	182	1	1	1	1	1	1	1	0	0	0	0	.40+01	9	0	0
223	177	0	1	0	0	1	1	1	0	1	1	1	.70+01	11	1	1
224	177	0	1	0	0	1	1	1	0	1	1	0	.40+01	11	0	0
225	87	1	1	0	1	1	1	1	1	1	1	0	.80+01	10	1	1
226	87	1	1	0	1	1	1	1	1	1	0	0	.40+01	10	0	0
227	215	1	0	1	1	1	1	1	0	0	0	0	.50+01	7	1	0
228	215	1	0	1	1	1	1	0	0	0	0	0	.90+01	7	1	0
229	227	1	0	1	1	1	1	1	1	0	0	0	.50+01	8	1	0
230	227	1	0	1	1	1	1	1	0	0	0	0	.80+01	8	1	0
231	229	1	0	1	1	1	1	1	1	1	0	0	.14+02	9	1	0
232	229	1	0	1	1	1	1	1	1	0	0	0	.50+01	9	0	0
233	180	1	1	1	1	1	1	0	1	0	0	0	.50+01	8	1	0
234	180	1	1	1	1	1	1	0	0	0	0	0	.80+01	8	1	0
235	233	1	1	1	1	1	1	0	1	1	0	0	.14+02	9	1	0
236	233	1	1	1	1	1	1	0	1	0	0	0	.50+01	9	0	0
237	105	0	1	0	1	1	1	1	0	1	1	0	.90+01	10	1	1
238	105	0	1	0	1	1	1	1	0	1	0	0	.50+01	10	0	0
239	99	1	1	0	1	1	0	1	0	1	1	0	.90+01	10	1	0
240	99	1	1	0	1	1	0	1	0	1	0	0	.50+01	10	0	0

```
 241          97     1  1  1  0  1  1  1  1  1  1  0     .90+01          10          1          0
 242          97     1  1  1  0  1  1  1  1  1  0  0     .50+01          10          0          0

 L'OPTIMUM DE LA FONCTION EST Z =  .60+01   OBTENU POUR LES VALEURS  0,1,0,1,1,1,1,1,1,1,0,

$FIN

 RUNID: BEZA01   ACCT: A18001          PROJECT: A18001
 TIME:    TOTAL: 00:00:15.568
          CAU:   00:00:04.574      I/O:  00:00:03.961
          CC/ER: 00:00:07.033      WAIT: 00:00:00.050
 SRC:     PS --  4464588           ES -- 2854161
 IMAGES  READ:   177       PAGES:  13
 START:    11:10:59 OCT 14,1975    FIN:  11:11:42 OCT 14,1975
```

CHAPITRE III.

ALGORITHME DE BALAS

A. INTRODUCTION

Comme nous venons de le voir, l'algorithme proposé précédemment présente l'inconvénient de conduire à de nombreux changements de chemin dans la recherche du programme optimal. En effet, le sommet choisi pour poursuivre la résolution ne figure généralement pas parmi les sommets créés à l'occasion de la dernière exploration. On est ainsi amené à développer de nombreux arcs sans aboutir à une solution optimale ou non. Cet inconvénient est dû au choix purement arbitraire de la variable à arbitrer.

Aussi il paraît intéressant de faire l'étude de l'algorithme de Balas. Dans cet algorithme, le choix de la variable à arbitrer dépend de la nature des différentes contraintes ainsi que des coefficients de la fonction économique, et la règle de retour-arrière permet de choisir le sommet origine d'une nouvelle progression en limitant les recherches.

Sous sa forme canonique le problème que nous allons étudier se formule de la façon suivante:

$$(P)\quad \begin{cases} \sum_{j=1}^{n} a_{ij}x_j \quad d_i & i = 1,\ldots,m \quad (S.1) \\ x_j = 0 \text{ ou } 1 & j = 1,\ldots,n \quad (S.2) \\ \min z = \sum_{j=1}^{n} c_j x_j & (S.3) \end{cases}$$

Nous limiterons l'étude du problème au cas où tous les coefficients c_j de la fonction économique sont positifs ou nuls. (Nous rappelons que si certains coefficients c_j sont négatifs, nous nous renvoyons facilement à cette étude en appliquant le même changement de variable que dans l'algorithme précédent).

B. TERMINOLOGIE

On appellera solution du problème toute combinaison de n valeurs 0 ou 1 associées au vecteur $X=(x_1,\ldots,x_n)$

Une solution sera dite réalisable quand elle satisfera aux contraintes (S.1) et (S.2).

Les solutions optimales sont celles qui minimisent la fonction économique.

C. ENONCE DE L'ALGORITHME.

L'algorithme de Balas permet d'explorer efficacement l'ensemble des solutions d'un programme linéaire à variables booléennes en les examinant chacune au plus une fois. Il permet d'obtenir le programme optimal dans la mesure où il en existe un et de reconnaître qu'il n'en existe pas dans le cas contraire.

L'exploration de l'ensemble des solutions se fait à partir d'une solution initiale S_0 pour laquelle l'ensemble des variables est nul. Elle se fera le long d'un chemin suivant un critère de progression permettant de choisir parmi l'ensemble des variables pouvant être arbitrées à 1 celle permettant de se rapprocher le plus rapidement d'une solution réalisable. Des critères d'arrêt vont permettre d'interrompre cette progression quand celle-ci s'avèrera inutile et une règle de retour arrière permettra de trouver la solution point de départ d'une nouvelle progression.

Pour exposer le déroulement de l'algorithme nous allons nous placer à la solution S_p.

a) Construction de l'ensemble des indices intéressants à être considérer pour passer d'une solution S_p à une solution descendante.

Soit J_p l'ensemble des indices des variables égales à 1 dans S_p.

$$J_p = \left\{ j \mid x_j = 1 \quad \text{dans } S_p \right\}$$

Nous allons déterminer l'ensemble N_p des indices n'appartenants pas à J_p et qui ajoutés à J_p peuvent permettre d'obtenir une solution meilleure que celles déjà obtenues antérieurement.

Pour l'obtention de N_p nous allons exclure de l'ensemble $1,\ldots,n$ des indices succeptibles d'être considérés, l'ensemble J_p ainsi que les trois ensembles suivants:

- l'ensemble E_p des indices j construit de la façon suivante:

Pour chacune des contraintes nous calculons la différence $dif_i = d_i - \sum_{j=1}^{n} a_{ij}x_j$. Soit I l'ensemble des indices i pour lesquels $dif_i > 0$. Si $a_{ij} \leq 0$, $\forall i \in I$ alors $j \in E_p$.

En effet, I contient l'ensemble des indices des contraintes non vérifiées dans S_p. Nous excluons alors de l'ensemble des indices j succeptibles d'être choisis

pour progresser ceux pour lesquels toutes ces contraintes ne seraient toujours pas vérifiées dans la solution descendante de S_p, c'est à dire nous éliminons les indices j pour lesquels $a_{ij} \leq 0 \quad \forall i \in I$.

$$E_p = \left\{ j \mid a_{ij} \leq 0 \text{ pour tout } i \mid d_i - \sum_{j=1}^{n} a_{ij} x_j > 0 \right\}$$

- l'ensemble D_p des indices j tel que j ajouté à J_p donne une solution S_{p+1} pour laquelle la valeur de la fonction économique est supérieure à la meilleure solution réalisable $\bar{z}$ obtenue jusqu'à là.

$$D_p = \left\{ j \mid (z_p + c_j) \geq \bar{z} \right\}$$

z_p est la valeur de la fonction économique en S_p.

- l'ensemble K_p des indices j qui ajoutés à J_p donne une solution déjà examinée antérieurement.

$$K_p = \left\{ j \mid J_p \cup \{j\} \quad \text{déjà examinée} \right\}$$

L'ensemble N_p est alors défini de la façon suivante:

$$N_p = \{1, \ldots, n\} - (J_p \cup D_p \cup E_p \cup K_p)$$

b) Règles d'arrêt.

Avant de passer à l'étude de l'ensemble N_p permettant de déterminer quel est l'indice fournissant la meilleure progression dans la résolution du problème, nous allons énoncer les différentes règles d'arrêt de la progression.

La solution S_p sera considérée comme bout pendant terminal de l'arborescence dans les trois cas suivants:

- si la solution S_p est solution réalisable. En effet, comme pour l'algorithme précédent, dans ce cas il est inutile de continuer à progresser puisque les coefficients c_j de la fonction économique étants positifs, les solutions descendantes ne pourraient pas être meilleures que la solution S_p elle même.

- si l'ensemble N_p déterminé précédemment est vide. Cela signifie qu'il n'existe pas d'indice qui ajouté à l'ensemble J_p permettrait d'obtenir une meilleure solution. La solution S_p est alors considéré comme bout pendant terminal vide puisque ne donnant pas lieu à une solution réalisable. En effet, si S_p était solution réalisable nous serions dans le premier cas d'arrêt et nous n'aurions pas déterminé l'ensemble N_p associé à S_p.

- si l'ensemble N_p n'étant pas vide, il existe un indice i pour lequel on a dans la solution S_p:

$$d_i - \sum_{j=1}^{n} a_{ij}x_j > 0$$

et : $$(d_i - \sum_{j=1}^{n} a_{ij}x_j) - \sum_{j \in N_p} \max \left[0, a_{ij}\right] > 0$$

Comme dans l'énoncé de l'algorithme précédent cela signifie qu'il existe au moins une contrainte qui dans la solution S_p n'est pas vérifiée et qu'elle ne pourra plus l'être dans la descendance de S_p. En effet, si l'inégalité précédente est vérifiée, alors même en fixant à 1 les variables ayant un coefficient $a_{ij}>0$ et à 0 les variables ayant un coefficient $a_{ij}<0$, la différence $d_i - \sum_{j=1}^{n} a_{ij}x_j$ restera positive dans la solution descendante. Donc, dans ce cas il s'avère inutile de continuer la progression. S_p sera bout pendant terminal vide ne donnant pas lieu à une solution réalisable.

En conclusion, nous pouvons constater que le premier et le troisième cas d'arrêt sont communs aux deux algorithmes. Le deuxième cas d'arrêt va permettre une nette amélioration par rapport à l'algorithme précédent car il limitera le champ d'investigations.

c) Recherche d'une nouvelle solution.

Dans la mesure où aucune des trois règles d'arrêt énoncées précédemment n'est vérifiée, nous pouvons continuer à progresser en recherchant la descendance de S_p. Pour déterminer cette nouvelle solution nous devons prolonger le chemin issu de S_p en ajoutant à l'ensemble J_p des indices des variables égales à 1 dans S_p un indice j qui est choisi dans l'ensemble N_p dont nous venons de voir la construction.

Certains tests vont nous permettre de déterminer qu'elle est l'indice $j \in N_p$ qui doit être choisi de préférence aux autres.

Si pour $i \in \{1,\ldots,m\}$ $d_i - \sum_{j=1}^{n} a_{ij}x_j > 0$, alors on a nécessairement pour cet indice i:

$$\left(d_i - \sum_{j=1}^{n} a_{ij}x_j \right) - \sum_{j \in N_p} \max\left[0, a_{ij}\right] \leqslant 0$$

sinon nous serions dans le dernier cas d'arrêt énoncé précédemment.

Par conséquent, si il existe un indice $k \in N_p$ tel que:

$$\left(d_i - \sum_{j=1}^{n} a_{ij} x_j \right) - \sum_{j \in N_p} \max \left[0, a_{ij} \right] - a_{ik} > 0$$

avec: $a_{ik} < 0$ alors: $d_i - \sum_{j=1}^{n} a_{ij} x_j \leqslant 0$ que si $x_k = 0$.

Donc on évitera de choisir l'indice k autant que possible.

On en déduit une règle de choix de l'indice permettant de progresser le long d'un chemin.

Règle de progression.

Pour chacun des indices $j \in N_p$, on calcule l'expression:

$$\sum_{i=1}^{m} \max \left[d_i - \sum_{j=1}^{n} a_{ij} x_j - a_{ij}, \; 0 \right]$$

Si il existe un indice j pour lequel cette expression est nulle, cela signifie que pour la nouvelle solution que nous allons obtenir, toutes les contraintes sont vérifiées. On choisira donc cet indice pour obtenir une solution réalisable S_{p+1} descendante de S_p.

Dans le cas contraire, c'est à dire si aucun indice $j \in N_p$ n'annule cette expression on choisira l'indice correspondant à:

$$\min_{j \in N_p} \left[\sum_{i=1}^{m} \max \left(d_i - \sum_{j=1}^{n} a_{ij} x_j - a_{ij}, \; 0 \right) \right]$$

Sachant que les contraintes sont vérifiées lorsque $d_i - \sum_{j=1}^{n} a_{ij}x_j \leqslant 0$ pour i=1,...,m , et qu'ainsi nous obtenons une solution réalisable, cette dernière règle de choix va permettre de déterminer quel est l'indice j fera diminuer le plus possible la valeur des nouvelles différences $d_i - \sum_{j=1}^{n} a_{ij}x_j$ que l'on obtiendra dans la solution descendante. Nous pouvons ainsi dire que le choix de cet indice permettra d'obtenir une solution descendante S_{p+1} qui s'approche le plus possible d'une solution réalisable.

Si plusieurs indices peuvent être choisis, on considérera celui pour lequel l'accroissement de la fonction économique est le plus petit. Si il y a encore ambiguïté à ce niveau du choix, on prendra l'un quelconque des indices donnant le plus faible accroissement.

En résumé la règle de progression permet:
- de choisir l'indice, si il en existe un, qui permet d'obtenir une solution descendante réalisable.
- entre plusieurs indices donnant une solution réalisable de choisir celui qui donne la plus petite valeur de la fonction économique.
- d'obtenir une solution "s'approchant le plus possible" d'une solution réalisable dans le cas où on ne peut pas en obtenir une à ce niveau de la résolution.

d) Règle de retour-arrière.

Dans le cas où l'une des trois règles d'arrêt est vérifiée, la solution S_p atteinte est bout pendant terminal.

La poursuite de la résolution se fera alors à partir d'une solution S_q appartenant au chemin allant de S_0 à S_p. S_q sera la solution la plus proche de S_p sur ce chemin pour laquelle l'ensemble N_q correspondant comporte au moins deux éléments. En effet, si N_q ne contient qu'un seul élément il est inutile d'examiner plus loin les descendances de S_q puisqu'on en vient. Une fois cette solution obtenue on poursuit la résolution du problème en reprenant l'algorithme à la construction du nouvel ensemble N'_q qui est évidemment différent de N_q.

Le choix de cette solution, point de départ d'une nouvelle descendance, n'est plus, comme dans l'algorithme précédent dépendant de la valeur de la fonction d'évaluation. Sous cette forme, la règle de retour-arrière permet de remédier à l'inconvénient présenté par le premier algorithme qui conduisait à de nombreux changements de chemin dans la recherche d'une solution optimale.

Les calculs prendront fin lorsque pour toute solution S_q du chemin que l'on vient de parcourir l'ensemble N_q correspondant est vide.

D. ETUDE D'UN EXEMPLE SIMPLE

Enoncé du problème:

Recherche de: Min $z = 5x_1+7x_2+10x_3+3x_4+x_5$

avec pour contraintes:

$$x_1-3x_2+5x_3+x_4-4x_5 \geq 2$$
$$-2x_1+6x_2-3x_3-2x_4+2x_5 \geq 0$$
$$-x_2+2x_3-x_4-x_5 \geq 1$$

avec $x_j=0$ ou 1 pour j=1,2,3,4

I^{ère} ETAPE:

La solution initiale est $S_0=(0,0,0,0)$, la valeur de la fonction économique associée est $z_0=0$. Pour cette solution, nous calculons les différences:

$dif_i = d_i - \sum_{j=1}^{n} a_{ij}x_j$ pour i=1,2,3.

$dif_1=2$ $dif_2=0$ $dif_3=1$

Nous déterminons à présent l'ensemble N_0 des indices j intéressants à considérer pour progresser. Pour la construction de N_0 nous nous référons à la définition de cet ensemble énoncée précédemment.

- $J_0 = \{ j \mid x_j=1 \text{ dans } S_0\} \Rightarrow J_0 = \emptyset$
- $D_0 = \{ j \mid (z_0+c_j) \geq \bar{z}_0\}$

  $\bar{z}_0$ est la valeur de la meilleure solution réalisable rencontrée jusqu'à cette étape, par convention initialement $\bar{z}_0=+\infty$, par conséquent: $D_0 = \emptyset$
- $K_0 = \{ j \mid J_0 \cup \{j\}$ déjà examiné $\} \Rightarrow K_0 = \emptyset$
- $E_0 = \{j \mid a_{ij} \leq 0$ pour tout $i \mid dif_i > 0\}$

  dif_i est strictement positif pour i=1 et pour i=3, de plus pour j=2 nous avons $a_{12}=-3$ et $a_{32}=-1$ de même pour j=5, $a_{15}=-4$ et $a_{35}=-1$, par conséquent:

  $E_0 = \{2,5\}$

  Nous en déduisons l'ensemble N_0:

  $N_0 = \{1,2,3,4,5\} - (J_0 \cup D_0 \cup K_0 \cup E_0) = \{1,3,4\}$

Tests d'arrêt.

Nous allons appliquer les tests d'arrêt à la solution S_0 pour déterminer si S_0 est bout pendant terminal ou non.

- S_0 n'est pas solution réalisable puique les contraintes 1 et 3 ne sont pas vérifiées à ce niveau de la résolution.

- l'ensemble No=$\{1,3,4\}$ n'est pas vide, de plus pour les contraintes 1 et 3 pour lesquelles nous avons $dif_1>0$ et $dif_3>0$ nous avons:

$$dif_1 - \sum_{j \in N_0} \max [0, a_{1j}] = 2-(1+5+1) = -5 < 0$$

$$dif_3 - \sum_{j \in N_0} \max [0, a_{3j}] = 1-(0+2+0) = 0$$

Par conséquent, nous en déduisons que la solution S_0 n'est pas bout pendant terminal.

Choix de l'indice j pour passer à la solution descendante.

Pour chacun des indices $j \in N_0$, nous calculons l'expression:

$$\sum_{i=1}^{m} \max \left[d_i - \sum_{j=1}^{n} a_{ij} x_j - a_{ij},\ 0 \right] \qquad (1)$$

pour j=1 (1) $\Leftrightarrow$ 1+2+1 = 4

j=3 (1) $\Leftrightarrow$ 0+3+0 = 3

j=4 (1) $\Leftrightarrow$ 1+2+2 = 5

L'indice j qui est choisi pour être ajouté à J_0 pour passer à la solution descendante est l'indice qui correspond à la valeur minimale de l'expression ci-dessus. Nous choisirons donc l'indice j=3.

2ème ETAPE

La deuxième solution considérée est $S_1=(0,0,1,0,0)$. Pour cette solution $z_1=10$. Les nouvelles valeurs des différences dif_i pour i=1,2,3 sont:

$dif_1=-3$ $dif_2=3$ $dif_3=-1$

Détermination de N_1

$J_1 = \{ j \mid x_j=1 \text{ dans } S_1 \} \Rightarrow J_1 = \{3\}$

$D_1 = \varnothing$ puisque $z_1 = +\infty$

$K_1 = \varnothing$

$E_1 = \{1,3,4\}$

Nous en déduisons que: $N_1 = \{2,5\}$

Conformément aux différentes remarques faites au courant de la I^{ère} étape sur les tests d'arrêt, nous constatons que la solution S_1 est bout pendant non terminal.

Choix de l'indice.

pour j=2 $\sum_{i=1}^{3} \max\left[d_i - \sum_{j=1}^{5} a_{ij} - a_{i2}, 0\right] = 0+0+0 = 0$

pour j=5 $\sum_{i=1}^{3} \max\left[d_i - \sum_{j=1}^{5} a_{ij} - a_{i5}, 0\right] = 1+1+0 = 2$

L'indice choisi pour être ajouté à J_1 pour passer à la solution descendante est j=2.

3ème ETAPE

La nouvelle solution est S_2=(0,1,1,0,0). Pour cette solution z_2=17. Pour cette solution, nous pouvons constater que les différences dif_1, dif_2 et dif_3 sont toutes trois négatives ou nulles, par conséquent la solution S_2 est réalisable, donc bout pendant terminal. Nous en déduisons que: $\bar{z}_2$=17.

On applique alors la règle de retour-arrière. On retourne à la solution S_1 qui est la première solution appartenant au chemin allant de S_2 à S_0 pour laquelle l'ensemble N_1 associé contient au moins deux éléments.

On détermine le nouvel ensemble N_1' associé à S_1 qui est une mise à jour de N_1. En effet, par rapport à la deuxième étape de la résolution l'ensemble K_1 a été modifié puisque nous venons de considérer la solution S_2, par conséquent N_1 lui aussi a été modifié.

$$K_1' = \{j \mid J_1 \cup \{j\} \text{ déjà examiné}\} = \{2\}$$

d'où: $N_1' = \{1,2,3,4,5\} - (J_1 \cup E_1 \cup K_1' \cup D_1)$

$= \{1,2,3,4,5\} - (\{3\} \cup \{1,3,4\} \cup \{2\})$

$= \{5\}$

Test d'arrêt.

Pour $N_1' = \{5\}$, nous avons $dif_2 - \sum_{j \in N_i} \max(o, a_{2j})$ $= 3-2 > 0$. Par conséquent, la deuxième contrainte n'est pas vérifiée et ne pourra plus l'être dans la nouvelle descendance de S_1. La dernière règle d'arrêt étant vérifiée, il ne sert à rien d'examiner la solution (0,0,1,0,1).

Nous retournons à la solution S_0 en appliquant à nouveau la règle de retour-arrière. Comme précédemment nous mettons à jour l'ensemble N_0, nous obtenons le nouvel ensemble N_0' associé à S_0. $N_0' = \{1,4\}$

Pour $N'_0 = 1,4$, nous constatons que la troisième contrainte ne pourra plus être vérifiée. En effet, $dif_3 - \sum_{j \in N'_0} \max(o, a_{3j}) = 1+0+0 = 1 > 0$. Puisque la troisième règle d'arrêt est vérifiée, il ne sert à rien d'examiner plus avant les descendances de la solution (0,0,0,0,0)

Nous ne pouvons plus appliquer la règle de retour-arrière puisque nous sommes à S_0. Par conséquent l'étude prend fin.

Le programme optimal est (0,1,1,0,0) et z = 17.

E. PROGRAMME

Le problème que nous résolvons à l'aide du programme proposé est le même que lors de l'étude du programme précédent.

Le temps total de traitement de l'exemple par ce deuxième programme est de 14.694 sec., le temps de calcul étant de 3.650 sec.. Nous constatons que ces deux temps sont inférieures à ceux du premier programme.

```
*FOR,IS
FOR 5114-10/27/75-11:59:09
FORTRAN 11A INTER-PROJETS 1100 (ORSAY)

   MAIN PROGRAM

   STORAGE USED: CODE(1) 001457; DATA(0) 003523; BLANK COMMON(2) 000000


   EXTERNAL REFERENCES (BLOCK, NAME)

    0003   NINTRS
    0004   NRDBS
    0005   NIOCS
    0006   NIOIS
    0007   NWDUS
    0010   NSTOPS


   STORAGE ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0000    003352 100F      0000    003353 101F      0000    003375 102F      0000    003405 103F      0000    003425 104F
    0000    003440 105F      0000    003447 106F      0000    003457 109F      0000    003461 109F      0001    000025 111G
    0001    000051 122G      0001    000066 126G      0001    000350 13L       0001    000104 135G      0001    001030 14L
    0001    000121 144G      0001    001007 15L       0001    000126 150G      0001    001031 16L       0001    000145 162G
    0001    001020 17L       0001    000201 177G      0001    000211 204G      0001    001303 21L       0001    000214 210G
    0001    000311 22L       0001    000361 23L       0001    000300 233G      0001    000303 237G      0001    000547 24L
    0001    000620 25L       0001    000324 251G      0001    000616 26L       0001    000341 261G      0001    000370 275G
    0001    001320 30L       0001    000414 307G      0001    000445 323G      0001    000474 334G      0001    000523 347G
    0001    000536 352G      0001    000577 373G      0001    000167 4L        0001    000646 40L       0001    000526 41L
    0001    000670 42L       0001    000550 43L       0001    000726 435G      0001    000712 44L       0001    000751 447G
    0001    000672 45L       0001    000760 452G      0001    000735 46L       0001    000775 49L       0001    000373 5L
    0001    001072 51L       0001    001131 517G      0001    001153 52L       0001    001134 522G      0001    001175 540G
    0001    001201 544G      0001    001234 557G      0001    001252 567G      0001    001332 61L       0001    001366 617G
    0001    001405 627G      0001    001447 642G      0001    000424 70L       0001    001375 71L       0001    000223 8L
    0001    000251 80L       0001    000454 81L       0000 I  001046 CHGT      0000 R  003316 CONS      0000 R  003323 CONSDI
    0000 R  003315 CONST     0000 R  001762 CONTR     0000 I  000050 D         0000 R  002203 DIF       0000 R  002373 DMEIL
    0000 R  003347 DMINME    0000 I  000074 E         0000 I  003310 I         0000 I  003321 IAIG      0000 I  003322 ID
    0000 I  003334 IE        0000 I  003340 IJ        0000 I  003346 IM        0000 I  001072 IND       0000 I  003337 INOD
    0000 I  003335 INDE      0000 I  003332 INDJ      0000 I  002347 INON      0000 I  003263 INOSOL    0000 I  003342 INOTRA
    0000 I  003331 IRET      0000 I  003326 IS        0000 I  002157 ITER      0000 I  003317 I1        0000 I  003314 I2
    0000 I  003341 I3        0000 I  003345 I4        0000 I  003313 J         0000 I  003336 JD        0000 I  003333 JE
    0000 I  003324 JI        0000 I  003330 JJ        0000 I  002133 JP        0000 I  002417 KK        0000 I  003344 MM
    0000 I  003351 MI        0000 I  003307 N         0000 I  003311 NBRE      0000 I  003312 NN        0000 R  003237 PRED
    0000 I  000024 PREDES    0000 I  000000 SOL       0000 R  003320 SOLL      0000 R  003327 SOL1      0000 R  003343 SOM
    0000 R  003325 SOM1      0000 I  000740 TRAVAI    0000 R  001712 VALZ      0000 I  001022 VAR       0000 I  000120 VECN
    0000 R  001736 Z         0000 R  003350 ZMIN


00100      1*      C
00100      2*      C     PROGRAMMATION LINEAIRE A VARIABLES BOOLEENNES PAR LA METHODE
00100      3*      C                        ARBORESCENTE DE BALAS
00100      4*      C
00101      5*            INTEGER SOL,PREDES,D,E,VECN,TRAVAI,VAR,CHGT
00103      6*            DIMENSION IND(20,20),VALZ(20),SOL(20),Z(20),CONTR(5,21),JP(20),
```

```
00103      7*          *PREDES(20),ITER(20),L(20),D(20),VECN(20,20),DIF(20,5),CHGT(20),
00103      8*          *INON(20),DMEIL(20),TRAVAI(50),KK(20,20),PRED(20),INDSOL(20),
00103      9*          *VAR(20)
00104     10*           READ(5,100) N
00107     11*           READ(5,108)(Z(I),I=1,N)
00115     12*           READ(5,100) NBRE
00120     13*           NN=N+1
00121     14*           DO 2 I=1,NBRE
00124     15*         2 READ(5,106)(CONTR(I,J),J=1,NN)
00133     16*           WRITE(6,104)(Z(I),I=1,N)
00141     17*           WRITE(6,105)
00143     18*           DO 3 I=1,NBRE
00146     19*         3 WRITE(6,106)((CONTR(I,J),J=1,N),CONTR(I,NN))
00155     20*           I2=0
00157     21*           CONST=0
00160     22*           CONS=0
00161     23*           DO 4 I1=1,N
00164     24*           IF(Z(I1).GT.0) GO TO 4
00166     25*           I2=I2+1
00167     26*           VAR(I2)=I1
00170     27*           CHGT(I1)=1
00171     28*           CONST=CONST+Z(I1)
00172     29*           Z(I1)=-Z(I1)
00173     30*         4 CONTINUE
00175     31*           WRITE(6,103)(VAR(I1),I1=1,I2)
00203     32*           DO 7 I1=1,NBRE
00206     33*           CONS=0
00207     34*           DO 8 I2=1,N
00212     35*           IF (CHGT(I2).EQ.0) GO TO 8
00214     36*           CONS=CONS+CONTR(I1,I2)
00215     37*           CONTR(I1,I2)=-CONTR(I1,I2)
00216     38*         8 CONTINUE
00220     39*         7 CONTR(I1,NN)=CONTR(I1,NN)-CONS
00222     40*           WRITE(6,101)
00224     41*           SOLL=2147483647.
00225     42*           IAIG=1
00226     43*           I=1
00227     44*           ITER(I)=I
00230     45*           PREDES(I)=0
00231     46*           PRED(I)=0
00232     47*          80 DO 11 ID=1,NBRE
00235     48*           CONSDI =0
00235     49*           DO 10 JI=1,N
00241     50*           IF(IND(I,JI).NE.0) CONSDI =CONSDI +CONTR(ID,JI)
00243     51*          10 CONTINUE
00245     52*          11 DIF(I,ID)=CONTR(ID,NN)-CONSDI
00247     53*           SOM1=0
00250     54*           DO 12 J=1,N
00253     55*           IF(IND(I,J).NE.0) SOM1=SOM1+Z(J)
00255     56*          12 CONTINUE
00257     57*           VALZ(I)=SOM1
00260     58*           DO 13 IS=1,NBRE
00263     59*           IF(DIF(I,IS).LE.0) GO TO 13
00265     60*           SOL(I)=0
00266     61*           GO TO 70
00267     62*          13 CONTINUE
00271     63*           SOL(I)=1
```

```
00272     64*          IF(VALZ(I).GE.SOLL) GOTO 5
00274     65*          SOLL=VALZ(I)
00275     66*          DO 6 I1=1,N
00300     67*        6 INDSOL(I1)=IND(I,I1)
00302     68*        5 SOL1=VALZ(I)+CONST
00303     69*          WRITE(6,102)(ITER(I),PREDES(I),(IND(I,J),J=1,N),SOL1,SOL(I))
00315     70*          GO TO 81
00316     71*       70 SOL1=VALZ(I)+CONST
00317     72*          WRITE(6,102)(ITER(I),PREDES(I),(IND(I,J),J=1,N),SOL1,SOL(I))
00331     73*       81 JJ=1
00332     74*          IPET=I
00333     75*          DO 22 J=1,N
00336     76*          IF(IND(I,J).NE.1) GO TO 22
00340     77*          JP(JJ)=J
00341     78*          JJ=JJ+1
00342     79*       22 CONTINUE
00344     80*          INDJ=JJ-1
00345     81*          I1=1
00346     82*          DO 23 JE=1,N
00351     83*          DO 24 IE=1,NBRE
00354     84*          IF(DIF(I,IE).LE.0) GO TO 24
00356     85*          IF(CONTR(IE,JE).GT.0) GO TO 23
00360     86*       24 CONTINUE
00362     87*          E(I1)=JE
00363     88*          I1=I1+1
00364     89*       23 CONTINUE
00366     90*          INDE=I1-1
00367     91*          I1=1
00370     92*          IF(SOLL.GE.2147483647.) GO TO 25
00372     93*          DO 26 JD=1,N
00375     94*          IF(VALZ(I)+Z(JD).LT.SOLL) GO TO 26
00377     95*          D(I1)=JD
00400     96*          I1=I1+1
00401     97*       26 CONTINUE
00403     98*       25 INDD=I1-1
00404     99*          I1=1
00405    100*          IJ=1
00406    101*       41 IF(IJ.GT.INDJ) GO TO 40
00410    102*          TRAVAI (I1)=JP(IJ)
00411    103*          I1=I1+1
00412    104*          IJ=IJ+1
00413    105*          GO TO 41
00414    106*       40 IE=1
00415    107*       43 IF(IE.GT.INDE) GO TO 42
00417    108*          TRAVAI (I1)=E(IE)
00420    109*          I1=I1+1
00421    110*          IE=IE+1
00422    111*          GO TO 43
00423    112*       42 ID=1
00424    113*       45 IF(ID.GT.INDD) GO TO 44
00426    114*          TRAVAI (I1)=D(ID)
00427    115*          I1=I1+1
00430    116*          ID=ID+1
00431    117*          GO TO 45
00432    118*       44 IF(KK(I,1).EQ.0) GO TO 46
00434    119*          I3=KK(I,1)+1
00435    120*          DO 48 I2=2,I3
```

```
00440    121*          TRAVAI(I1)=KK(I,I2)
00441    122*          I1=I1+1
00442    123*       48 CONTINUE
00444    124*       46 INDTRA=I1-1
00445    125*          I3=1
00446    126*          DO 49 I2=1,N
00451    127*          DO 60 I1=1,INDTRA
00454    128*          IF(TRAVAI (I1).EQ.I2) GO TO 49
00456    129*       60 CONTINUE
00460    130*          VECN(I,I3)=I2
00461    131*          I3=I3+1
00462    132*       49 CONTINUE
00464    133*          INDN(I)=I3-1
00465    134*          I3=I3-1
00466    135*       50 IF(INDN(1).EQ.0)  GO TO 51
00470    136*          IS=1
00471    137*       15 I1=1
00472    138*          IF(DIF(I,IS).GT.0) GO TO 14
00474    139*       17 IS=IS+1
00475    140*          IF(IS.LE.NBRE) GO TO 15
00477    141*          GO TO 52
00500    142*       14 SOM=0
00501    143*       16 MM=VECN(I,I1)
00502    144*          IF(CONTR(IS,MM).GT.0) SOM=SOM+CONTR(IS,MM)
00504    145*          I1=I1+1
00505    146*          IF(I1.LE.INDN(I)) GO TO 16
00507    147*          IF(DIF(I,IS)-SOM.GT.0) GO TO 51
00511    148*          GO TO 17
00512    149*       51 KK(1,1)=0
00513    150*          I=PRED(I)
00514    151*          I4=I+1
00515    152*          I3=KK(I,1) +1
00516    153*          DO 88 I1=I4,N
00521    154*          DO 88 I2=1,I3
00524    155*       88 KK(I1,I2)=KK(I,I2)
00527    156*          IF(INDN(I).GE.2) GO TO 81
00531    157*          IF(I.EQ.0) GO TO 71
00533    158*          GO TO 51
00534    159*       52 IF(INDN(I).EQ.1) GO TO 30
00536    160*          IM=INDN(I)
00537    161*          DO 18 I1=1,IM
00542    162*          SOM=0
00543    163*          DO 19 ID=1,NBRE
00546    164*          MM=VECN(I,I1)
00547    165*          IF(DIF(I,ID)-CONTR(ID,MM).GT.0) SOM=SOM+DIF(I,ID)-CONTR(ID,MM)
00551    166*       19 CONTINUE
00553    167*       18 DMEIL(I1)=SOM
00555    168*          DMINME  =21474833647.
00556    169*          DO 20 I1=1,IM
00561    170*          IF(DMEIL(I1).LT.DMINME  ) DMINME  =DMEIL(I1)
00563    171*       20 CONTINUE
00565    172*          ZMIN=21474833647.
00566    173*          DO 21 I1=1,IM
00571    174*          IF(DMEIL(I1).LT.DMINME.OR.DMEIL(I1).GT.DMINME) GO TO 21
00573    175*          MM=VECN(I,I1)
00574    176*          IF(Z(MM).GE.ZMIN) GO TO 21
00576    177*          ZMIN=Z(MM)
```

```
00577    178*          M1=MM
00600    179*       21 CONTINUE
00602    180*          KK(I,1)=KK(I,1)+1
00603    181*          I1=KK(I,1)+1
00604    182*          KK(I,I1)=MM
00605    183*          GO TO 61
00606    184*       30 KK(I,1)=KK(I,1)+1
00607    185*          I1=KK(I,1)+1
00610    186*          KK(I,I1)=VECN(I,1)
00611    187*       61 I=I+1
00612    188*          IAIG=IAIG+1
00613    189*          ITER(I)=IAIG
00614    190*          PRED(I)=IRET
00615    191*          PREDES(I)=ITER(IRET)
00616    192*          DO 62 I1=1,N
00621    193*       62 IND(I,I1)=IND(IRET,I1)
00623    194*          IND(I,MM)=1
00624    195*          GO TO 80
00625    196*       71 SOLL=SOLL+CONST
00626    197*          DO 9 I1=1,N
00631    198*          IF(CHGT(I1).EQ.1.AND.INDSOL(I1).EQ.1) INDSOL(I1)=0
00633    199*          IF(CHGT(I1).EQ.1.AND.INDSOL(I1).EQ.0) INDSOL(I1)=1
00635    200*        9 CONTINUE
00637    201*          WRITE(6,109)(SOLL,(INDSOL(I1),I1=1,N))
00646    202*          STOP
00647    203*      100 FORMAT(I2)
00650    204*      101 FORMAT('0',//,10X,'ITERATION',3X,'SUCCESSEUR DE',8X,'VALEUR DES VA
00650    205*         *RIABLES',8X,'VALZ',9X,'SOLUTION')
00651    206*      102 FORMAT(14X,I3,11X,I3,5X,11(I2,1X),2X,E7.2,10X,I2)
00652    207*      103 FORMAT('0',/,10X,'LE CHANGEMENT DE VARIABLE  X''=1-X  A ETE EFFECT
00652    208*         *UE SUR LES VARIABLES ',4(2X,I2))
00653    209*      104 FORMAT('1',//,10X,'FONCTION A MINIMISER   MIN Z =  ',11(E7.2,'+'))
00654    210*      105 FORMAT('0',//,20X,' SOUS LES CONTRAINTES')
00655    211*      106 FORMAT('0',/,10X,11(E7.2,'+'),' > OU = A ',E7.2)
00656    212*      108 FORMAT(25E5.0)
00657    213*      109 FORMAT('0',//,10X,'L''OPTIMUM DE LA FONCTION EST Z = ',E7.2,
00657    214*         *'   OBTENU POUR LES VALEURS   ',20(I1,','))
00660    215*          END

      END OF COMPILATION:        NO  DIAGNOSTICS.

%XQT
MAP 26.2-10/27-11:59


ADDRESS LIMITS    001000 013762      5619 IBANK WORDS DECIMAL
                  040000 050321      4306 DBANK WORDS DECIMAL
STARTING ADDRESS  012304



          SEGMENT $MAIN$          001000 013762     040000 050321

NSWTC$/FOR69          S(1)  001000 001024
```

FONCTION A MINIMISER MIN Z = .20+01,-.40+01, .60+01, .50+01,-.30+01, .20+01,-.40+01,-.30+01, .90+01, .40+01, .30+01,

SOUS LES CONTRAINTES

.30+01, .40+01,-.30+01, .50+01, .80+01, .90+01,-.40+01, .30+01,-.20+01, .70+01, .60+01, > OU = A .27+02

-.10+01, .20+01, .40+01, .30+01, .70+01,-.60+01, .30+01,-.10+01, .80+01, .30+01, .10+01, > OU = A .18+02

.10+01, .10+01,-.10+01, .10+01,-.30+01, .50+01, .20+01,-.20+01, .10+01, .30+01, .20+01, > OU = A .80+01

LE CHANGEMENT DE VARIABLE X'=1-X A ETE EFFECTUE SUR LES VARIABLES 2 5 7 8

ITERATION	SUCCESSEUR DE	VALEUR DES VARIABLES											VALZ	SOLUTION
1	0	0	0	0	0	0	0	0	0	0	0	0	-.14+02	0
2	1	0	0	0	0	0	0	0	0	0	1	0	-.10+02	0
3	2	0	0	0	0	0	0	0	0	0	1	1	-.70+01	0
4	3	0	0	0	1	0	0	0	0	0	1	1	-.20+01	0
5	4	0	0	0	1	0	0	0	0	1	1	1	.70+01	0
6	5	0	0	0	1	0	1	0	0	1	1	1	.90+01	1
7	4	0	0	0	1	0	0	0	1	0	1	1	.10+01	0
8	7	1	0	0	1	0	0	0	1	0	1	1	.30+01	0
9	8	1	0	0	1	0	1	0	1	0	1	1	.50+01	0
10	8	1	0	0	1	1	0	0	1	0	1	1	.60+01	0
11	7	0	0	0	1	0	1	0	1	0	1	1	.30+01	0
12	7	0	0	0	1	0	0	1	1	0	1	1	.50+01	0
13	4	1	0	0	1	0	0	0	0	0	1	1	.00	0
14	13	1	0	1	1	0	0	0	0	0	1	1	.60+01	0
15	14	1	0	1	1	0	1	0	0	0	1	1	.80+01	0
16	4	0	0	0	1	0	1	0	0	0	1	1	.00	0
17	3	1	0	0	0	0	0	0	0	0	1	1	-.50+01	0
18	17	1	0	0	0	0	0	0	0	1	1	1	.40+01	0
19	18	1	0	0	0	0	1	0	0	1	1	1	.60+01	0
20	18	1	0	0	0	0	0	1	0	1	1	1	.80+01	0
21	17	1	0	0	0	0	0	0	1	0	1	1	-.20+01	0
22	21	1	0	0	0	0	1	0	1	0	1	1	.00	0
23	21	1	0	1	0	0	0	0	1	0	1	1	.40+01	0
24	17	1	0	1	0	0	0	0	0	0	1	1	.10+01	0
25	24	1	0	1	0	0	1	0	0	0	1	1	.30+01	0
26	3	0	0	0	0	0	0	0	0	1	1	1	.20+01	0
27	26	0	0	0	0	0	1	0	0	1	1	1	.40+01	0
28	27	0	0	0	0	0	1	0	1	1	1	1	.70+01	1
29	3	0	0	0	0	0	1	0	0	0	1	1	-.50+01	0
30	3	0	0	0	0	0	0	0	1	0	1	1	-.40+01	0
31	2	0	0	0	1	0	0	0	0	0	1	0	-.50+01	0
32	31	0	0	0	1	0	1	0	0	0	1	0	-.30+01	0
33	32	0	0	0	1	0	1	0	0	1	1	0	.60+01	1
34	31	1	0	0	1	0	0	0	0	0	1	0	-.30+01	0

35	34	1	0	0	1	0	0	0	1	0	1	0	.00	0
36	31	0	0	0	1	0	0	0	0	1	1	0	.40+01	0
37	2	0	0	0	0	0	1	0	0	0	1	0	-.80+01	0
38	37	0	0	0	0	0	1	0	0	1	1	0	.10+01	0
39	1	0	0	0	0	0	0	0	0	0	0	1	-.11+02	0
40	39	0	0	0	1	0	0	0	0	0	0	1	-.60+01	0
41	40	0	0	0	1	0	1	0	0	0	0	1	-.40+01	0
42	41	0	0	0	1	0	1	0	0	1	0	1	.50+01	0
43	40	1	0	0	1	0	0	0	0	0	0	1	-.40+01	0
44	43	1	0	0	1	0	0	0	0	1	0	1	.50+01	0
45	39	0	0	0	0	0	1	0	0	0	0	1	-.90+01	0
46	45	0	0	0	0	0	1	0	0	1	0	1	.00	0
47	1	0	0	0	1	0	0	0	0	0	0	0	-.90+01	0
48	47	0	0	0	1	0	1	0	0	0	0	0	-.70+01	0
49	48	0	0	0	1	0	1	0	0	1	0	0	.20+01	0
50	1	0	0	0	0	0	1	0	0	0	0	0	-.12+02	0
51	50	0	0	0	0	0	1	0	0	1	0	0	-.30+01	0

L'OPTIMUM DE LA FONCTION EST Z = .60+01 OBTENU POUR LES VALEURS 0,1,0,1,1,1,1,1,1,1,0,

%FIN

RUNID: BEZA01 ACCT: A18001 PROJECT: A18001

TIME: TOTAL: 00:00:14.694

CAU: 00:00:03.650 I/O: 00:00:04.129

CC/ER: 00:00:06.914 WAIT: 00:00:00.050

SRC: PS -- 4066945 ES -- 1773375

IMAGES READ: 225 PAGES: 10

START: 11:59:09 OCT 27,1975 FIN: 11:59:19 OCT 27,1975

CONCLUSION.

Ce travail nous a permis de mettre en évidence les possibilités d'application de la méthode S.E.P.. Nous avons démontré que cette méthode détermine l'existence ou la non existence d'un programme optimal pour tous programmes linéaires et ceci en un nombre fini d'opérations. Son utilisation repose sur le choix de propriétés séparatrices et sur le choix d'une fonction d'évaluation; ces choix s'appuient sur la nature du problème étudié.

L'analyse de cette méthode nous a permis de montrer qu'elle est applicable à la résolution de programmes linéaires à variables booléennes ayant une fonction à optimiser et des contraintes comportant des coefficients de signe et de valeur absolument quelconque. La programmation de l'algorithme déduit de cette méthode est trés simple. Le programme ne comporte que des opérations d'additions, de soustractions et de comparaisons, ce qui permet d'éliminer les problèmes d'erreurs d'arrondi et d'obtenir un programme performant par ses temps d'exécution. Les limites de cette méthode repose

sur la nécessité d'utiliser des calculateurs à capacité de mémoire importante pour le traitement de problème de grande taille, ainsi que sur le fait que le choix de la variable que l'on fixe à 1 est purement arbitraire. La place nécessaire au traitement d'un problème ne peut pas être calculée a priori puisqu'elle ne dépend pas simplement du nombre de variables mais dépend essentiellement de la nature particulière de chaque problème.

L'algorithme de Balas présente l'avantage de déterminer la variable à arbitrer en fonction de la nature des coefficients des contraintes et de ceux de la fonction économique. Le but de cet algorithme est d'étudier chaque contrainte séparément de façon à déterminer quelle est la variable, s'il en existe une, qui permette de diminuer la somme totale des impossibilités. Elle permet ainsi d'atteindre rapidement une solution réalisable et de pouvoir borner la fonction économique. Sa règle de retour-arrière, de plus, permet de limiter les recherches pour trouver une solution point de départ d'une nouvelle descendance et ainsi de diminuer le nombre des calculs. Sa programmation nécessite peu de place mémoire et cette place ne dépend que du nombre de variables et du nombre de contraintes du problème traité. Cet algorithme, par conséquent, pourrait permettre l'étude de programmes

linéaires trés importants et d'obtenir, même si le temps de calculateur est limité, une "bonne" solution. Nous pouvons qualifier la solution de "bonne" en raison du critère de choix des variables arbitrées.

En conclusion et au vu des résultats de la résolution de l'exemple proposé suivant les deux algorithmes, nous pouvons dire que l'algorithme de Balas est moins coûteux aussi bien en temps de calcul qu'en place de mémoire et qu'il mérite par conséquent d'être préféré à l'algorithme classique pour la résolution de tels problèmes.

BIBLIOGRAPHIE

1. E. BALAS — Un algorithme additif pour la résolution des programmes linéaires en variables bivalentes. C.R. Académie des Sciences de Paris, 13 avril 1964.

2. C. BERGE — Théorie des graphes et ses applications. Dunod, Paris, 1958.

3. G. ESCHER — Einfuhrung in die Methode "Branch and Bund"

4. FORD L.R. et FULKERSON D.R. Flots dans un graphe. Gauthier-Villards, Paris, 1967.

5. A. KAUFMANN — Introduction à la combinatorique. Dunod, Paris, 1959.

6. B. ROY — Algèbre moderne et théorie des graphes. Dunod, Paris, 1969.

7. B. ROY et M. SIMMONARD "Nouvelle méthode permettant d'explorer un ensemble de possibilités et de déterminer un optimum" Rev. Fr. Rech. Op., 18, 1961.

8. M. SIMMONARD Programmation linéaire, technique de calcul économique. Tome 2, Dunod, Paris, 1973.

9. NOTE d'ETUDE AN-N°36 Méthode arborescente pour programmes linéaires mixtes. Centre de Calcul Scientifique de l'Armement, déc. 1972.

NOM DE L'ETUDIANT : BEZAUT Olivier

NATURE DE LA THESE : DOCTORAT DE SPECIALITE EN MATHEMATIQUES

VU, APPROUVE

& PERMIS D'IMPRIMER

NANCY, le 7 novembre 1975

LE PRESIDENT DE L'UNIVERSITE DE NANCY I