**MASTER EN SCIENCES MATHÉMATIQUES**

# Techniques de décomposition de systèmes d'équations NXN à grande échelle par la théorie des graphes

Haustrate, Gertrude; Dontaine, Philippe

*Award date:*
1976

*Awarding institution:*
Universite de Namur

# TECHNIQUES DE DECOMPOSITION DE SYSTEMES D'EQUATIONS ("N X N") A GRANDE ECHELLE PAR LA THEORIE DES GRAPHES

TOME I

Gertrude HAUSTRATE

et

Philippe DONTAINE

TABLE DES MATIERES.

CHAPITRE II. ETUDE D'UN SOUS SYSTEME IRREDUCTIBLE.



CHAPITRE III. AU POINT DE VUE PRATIQUE

## CHAPITRE IV: REALISATION D'UN PROGRAMME POUR ORDINATEURS

## Introduction

L'avènement des ordinateurs a permis à l'homme de s'attaquer à des problèmes de grande complexité. Parmi ces problèmes, il est rare de rencontrer des approches de solutions qui ne fassent pas appel à des résolutions de systèmes d'équations. La complexité des problèmes est liée à la grande taille, du moins dans certains cas, des processus physiques, chimiques, biologiques, économiques...,à résoudre. Trouver rapidement une solution de systèmes d'équations à grande échelle rend réalisable la recherche de solutions numériques des problèmes.

Il s'avère que, pratiquement, décomposer les systèmes à grande échelle en sous-systèmes facilement résolvables isolément, est indispensable si l'on cherche l'efficacité dans la recherche de solutions aux problèmes posés.

Nous comptons proposer un tel algorithme de décomposition de systèmes à grande échelle.

La littérature a proposé,depuis quelques années des méthodes dont l'inspiration se trouve en théorie des graphes ( Kevorkian, Ledet et Himmelblau, Steward ). A ces techniques de décomposition se joignent des études plus spécialisées , telle celle de Cheung et Kuh ( voir bibliographie ) qui suggèrent des raffinements aux méthodes existantes ( Kevorkian ). La théorie des graphes est utilisée dans le but d'indiquer des ordres de résolution d'équations optimaux. Ce but est-il atteint?

Nous voyons au travail que nous proposons deux parties

1. Dans un premier temps, nous comptons définir un algorithme de décomposition qui soit une synthèse des différents algorithmes portés à notre connaissance

- Dans le chapitre 1, nous exposerons comment obtenir les plus petits sous-systèmes d'équations à résoudre simultanément. Nous indiquerons comment découvrir un certain ordre de résolution de ces sous-systèmes.
- Dans le chapitre 2, nous montrons que décomposer un sous-système est possible et que nous pouvons donner un ordre de résolution des équations du

sous-système qui nous permette de ne résoudre simultanément qu'une petite partie des équations.

- Dans le chapitre 3 nous donnons des exemples et remarques relatives aux chapitres I et II.

Ces 3 chapitres font l'objet du premier tome du travail.

2. Un but du travail est de construire un programme qui permettrait de constater si la méthode de décomposition proposée est réalisable pratiquement, c.à.d. apporte une aide "efficace" à la résolution de système d'équations.

- Dans le chapitre 4, nous exposons la réalisation pratique d'une partie de l'algorithme, qui exige auparavant une analyse de la représentation des données en mémoire centrale d'ordinateurs. Nous y donnons brièvement les renseignements permettant la compréhension du programme réalisé.
Nous apportons aussi succintement une conclusion à ce travail en montrant que les algorithmes existant ne sont pas tous réalisables pour des décompositions effectives de systèmes à grande échelle.

Ce chapitre fait l'objet du second tome.

# Chapitre I : A LA RECHERCHE DE SOUS SYSTEMES FACILEMENT RESOLVABLES.

## I.1) Représentation du système d'équations.

Nous nous trouvons en présence d'un ensemble de n équations linéaires ou non à "n" inconnues avec "n" très grand. Le premier problème qui se pose est de découvrir une représentation de ce système, représentation qui, dans la suite, va nous permettre un certain ordre pour la résolution des équations du système et sur laquelle il est assez facile de travailler.

Nous considérons donc un ensemble de n équations non linéaires ou linéaires à n inconnues :

$$f_i(x) = 0 \qquad i \in \bar{n}$$

avec x : un vecteur à n inconnues

La forme sous laquelle se trouvent les équations est assez gênante parce qu'elle ne nous permet en aucune façon de déterminer l'ordre de résolution des équations de telle sorte que toutes les équations seront résolues.

Dans ce but, nous allons définir plusieurs notions :

### I.1.A) . Définition d'une matrice d'occurrence.

Chacune des équations $f_i$ dépend d'un certain nombre d'inconnues $x_i$. Nous définissons, pour chaque équation du système, un ensemble $C_i$ d'éléments $C_{ij}$ qui prendront la valeur "1" si l'inconnue $x_j$ apparait dans la fonction $f_i$ et la valeur "0" si non.

Nous aurons par conséquent :

$$C_i = \left\{ C_{ij} \text{ tel que } \begin{array}{l} C_{ij} = 0 \text{ si } x_j \notin f_i \\ C_{ij} = 1 \text{ si } x_j \in f_i \end{array} \quad j \in \bar{n} \right\} \quad i \in \bar{n}$$

Les ensembles vont nous permettre de définir une matrice booléenne appelée "matrice d'occurrence". A chaque ligne, nous faisons correspondre une équation $f_i$ et à chaque colonne une variable $x_i$. Nous indiquerons un "1" dans la ligne i et dans la colonne j si l'élément $C_{ij} = 1$ dans $C_i$ c'est-à-dire si $x_j$ apparait dans l'équation $f_i$, tandis que nous noterons un "0" dans la ligne i et dans la colonne k si l'élément $C_{ik} = 0$ dans $C_i$ c'est-à-dire $x_k$ ne se trouve pas dans l'équation $f_i$.

Par conséquent, pour tout système d'équations, nous avons une matrice du type :

$$\begin{array}{c} \\ f_1 \\ \vdots \\ f_i \\ \vdots \\ f_n \end{array} \begin{array}{c} \begin{array}{ccc} x_1 \cdots\cdots & x_2 \cdots\cdots & x_n \end{array} \\ \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \end{array} = C$$

. Définition de l'ensemble de sortie.

Dans un second temps, nous allons assigner à chaque équation $f_i$ une certaine variable appelée "variable de sortie" par rapport à laquelle l'équation $f_i$ est résolue.

Cette détermination est faite dans un but bien précis : nous voulons déterminer par quelle équation une variable aura sa solution ainsi que de quoi dépendra cette variable.

Cette détermination des variables de sortie est sujette à la loi suivante :

> A chaque équation $f_i$ correspond une seule variable de sortie.
> A chaque variable de sortie $x_j$ est assignée une et une seule équation $f_i$ avec $x_j \in f_i$ obligatoirement.

Nous pouvons alors définir :

- des paires de sortie : $(x_j, f_i)$
- un ensemble de paires de sortie :

$$(x,f) = \{(x_j,f_i),\ i \in \bar{n}\ ,\ j \in \bar{n}\}$$

Pour un système de n équations à n inconnues, il est possible de trouver plusieurs ensembles de paires de sortie. Le problème est donc de déterminer un ensemble de sortie qui optimise les résultats c'est-à-dire non pas la réponse finale mais bien le temps d'exécutions.

Dans les problèmes physiques, l'ensemble de sortie est généralement défini au moyen de la configuration du système. Dans ces cas-là, l'ensemble de sortie est implicitement déterminé par le problème lui-même.

Dans les cas où la détermination d'un ensemble de sortie n'est pas immédiate, à cette détermination se superpose une analyse de sensibilité qui est basée sur le principe suivant :

Nous considérons une équation :

$$f_i (x_1, x_2, x_k, x_\ell, x_n) = 0$$

Supposons à présent qu'une petite variation relative dans les variables $x_1$, $x_k$, $x_\ell$, $x_n$ dans $f_i$ produit une grande variation relative dans $x_2$.

Dans un tel cas, la variable $x_2$ ne pourra être utilisée comme variable de sortie pour $f_i$.

Dans un chapitre suivant, nous verrons un algorithme de résolution basé sur un schéma itératif. L'emploi de l'analyse de sensibilité dans la détermination de l'ensemble de sortie a comme effet une convergence plus rapide de la méthode itérative.

Après avoir défini l'ensemble de sortie, nous pouvons opérer une modification de la matrice d'occurrence. Nous effectuons des permutations de colonne de façon à ce que les paires de sortie se situent sur la diagonale (c'est-à-dire soient indiquées par les 1 diagonaux).

Par conséquent, afin d'avoir les paires de sortie toujours sur la diagonale, toute permutation de lignes devra être suivie de la même permutation des colonnes.

Dès que nous avons mis la matrice d'occurrence sous cette forme-là, nous sommes certains d'avoir des "1" sur la diagonale principale.

. Définition du graphe associé.

Dès que pour une équation $f_i(x_1 \cdots x_i \cdots x_n) = 0$ on a défini une paire de sortie $(x_j, f_i)$, nous pouvons écrire l'équation sous la forme :

$$x_j = f'_i(x_1 \cdots x_{j-1}, x_{j+1}, \cdots x_n)$$

Nous constatons donc qu'il existe une certaine dépendance entre la variable $x_j$ et les variables $x_1 \cdots x_{j-1}$, $x_{j+1}, \cdots x_n$. Il apparait par conséquent un flux d'information de $x_1 \cdots x_{j-1}$, $x_{j+1} \cdots x_n$ vers $x_j$.

Si nous représentons les variables par les sommets d'un graphe orienté, les arcs indiqueront les influences directes entre les différentes variables.

Nous définirons alors l'existence d'un arc du graphe par la méthode suivante :

$\exists$ l'arc $(x_j, x_i) \Leftrightarrow$ $x_j$ influence directement $x_i$ c'est-à-dire $x_j$ apparait dans l'équation ayant $x_i$ comme variable de sortie.

Il est à remarquer qu'un tel graphe n'aura aucune boucle : en effet, pour que l'arc $(x_i, x_i)$ existe, il faut que $x_i$ influence directement $x_i$, ce qui est impossible par définition.

Une influence indirecte entre $x_i$ et $x_j$ sera indiquée par un chemin orienté de longueur $\geqslant 2$ dans le graphe.

Par conséquent, nous voyons que le système de n équations à n inconnues que nous avions au départ, a pu être mis sous la forme d'un 1— Graphe; étant donné que le flux d'information est déterminé par l'ensemble de sortie que nous avons assigné au système, il s'en suit que si nous considérons un autre ensemble de sortie, nous nous trouverons en présence d'un nouveau graphe.

I.1.B) Exemple.

A présent, nous allons rapidement voir ce que cela nous donne sur un exemple. Nous verrons également que si nous considérons deux ensembles de sortie différents, nous obtenons deux graphes associés différents.

Considérons un système de 13 équations à 13 inconnues :

| équations | 1er ens. de sortie | 2ème ens. de sortie |
|---|---|---|
| $f_1(x_4, x_9) = 0$ | $(x_4, f_1)$ | $(x_9, f_1)$ |

| | | |
|---|---|---|
| $f_2(x_1,x_5,x_{10}) = 0$ | $(x_5,f_2)$ | $(x_1,f_2)$ |
| $f_3(x_{11},x_{12}) = 0$ | $(x_{11},f_3)$ | $(x_{11},f_3)$ |
| $f_4(x_3,x_7) = 0$ | $(x_7,f_4)$ | $(x_3,f_4)$ |
| $f_5(x_2,x_{12},x_{13}) = 0$ | $(x_2,f_5)$ | $(x_2,f_5)$ |
| $f_6(x_4,x_9) = 0$ | $(x_9,f_6)$ | $(x_4,f_6)$ |
| $f_7(x_1,x_5,x_7) = 0$ | $(x_1,f_7)$ | $(x_5,f_7)$ |
| $f_8(x_2,x_5,x_6) = 0$ | $(x_6,f_8)$ | $(x_6,f_8)$ |
| $f_9(x_7,x_{10}) = 0$ | $(x_{10},f_9)$ | $(x_7,f_9)$ |
| $f_{10}(x_8,x_{13}) = 0$ | $(x_8,f_{10})$ | $(x_{13},f_{10})$ |
| $f_{11}(x_4,x_{12}) = 0$ | $(x_{12},f_{11})$ | $(x_{12},f_{11})$ |
| $f_{12}(x_2,x_4,x_8,x_{13}) = 0$ | $(x_{13},f_{12})$ | $(x_8,f_{12})$ |
| $f_{13}(x_3,x_{10}) = 0$ | $(x_3,f_{13})$ | $(x_{10},f_{13})$ |

Nous avons vu que dans un premier temps, nous définissons les ensembles :

$C_1 = \{0,0,0,1,0,0,0,0,1,0,0,0,0\}$
$C_2 = \{1,0,0,0,1,0,0,0,0,1,0,0,0\}$
$C_3 = \{0,0,0,0,0,0,0,0,0,0,1,1,0\}$
$C_4 = \{0,0,1,0,0,0,1,0,0,0,0,0,0\}$
$C_5 = \{0,1,0,0,0,0,0,0,0,0,0,1,1\}$
$C_6 = \{0,0,0,1,0,0,0,0,1,0,0,0,0\}$
$C_7 = \{1,0,0,0,1,0,1,0,0,0,0,0,0\}$
$C_8 = \{0,1,0,0,1,1,0,0,0,0,0,0,0\}$
$C_9 = \{0,0,0,0,0,0,1,0,0,1,0,0,0\}$
$C_{10} = \{0,0,0,0,0,0,0,1,0,0,0,0,1\}$
$C_{11} = \{0,0,0,1,0,0,0,0,0,0,0,1,0\}$
$C_{12} = \{0,1,0,1,0,0,0,1,0,0,0,0,1\}$
$C_{13} = \{0,0,1,0,0,0,0,0,0,1,0,0,0\}$

Alors en mettant les fonctions fi en ligne et les variables $x_j$ en colonne, nous obtenons la matrice booléenne suivante :

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $x_{11}$ | $x_{12}$ | $x_{13}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $f_1$ | | | | 1 | | | | | 1 | | | | |
| $f_2$ | 1 | | | | 1 | | | | | 1 | | | |
| $f_3$ | | | | | | | | | | | 1 | 1 | |
| $f_4$ | | | 1 | | | | 1 | | | | | | |
| $f_5$ | | 1 | | | | | | | | | | 1 | 1 |
| $f_6$ | | | | 1 | | | | | 1 | | | | |
| $f_7$ | 1 | | | | 1 | | 1 | | | | | | |
| $f_8$ | | 1 | | | 1 | 1 | | | | | | | |
| $f_9$ | | | | | | | 1 | | 1 | | | | |
| $f_{10}$ | | | | | | | | 1 | | | | | 1 |
| $f_{11}$ | | | | 1 | | | | | | | 1 | | |
| $f_{12}$ | | 1 | | 1 | | | | 1 | | | | | 1 |
| $f_{13}$ | | | 1 | | | | | | | 1 | | | |

Par des permutations des colonnes, nous allons réarranger cette matrice de telle sorte que les paires de sortie se situent sur la diagonale

Pour le premier ensemble de sortie, nous aurons donc :

| $f_i$ \ $x_i$ | 4 | 5 | 11 | 7 | 2 | 9 | 1 | 6 | 10 | 8 | 12 | 13 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | | | | 1 | | | | | | | |
| 2 | | 1 | | | | | | | 1 | | | | |
| 3 | | | 1 | | | | | | | | 1 | | |
| 4 | | | | 1 | | | | | | | | | 1 |
| 5 | | | | | 1 | | | | | | 1 | 1 | |
| 6 | 1 | | | | | 1 | | | | | | | |
| 7 | | 1 | | 1 | | | 1 | | | | | | |
| 8 | | 1 | | | 1 | | | 1 | | | | | |
| 9 | | | | 1 | | | | | 1 | | | | |
| 10 | | | | | | | | | | 1 | | 1 | |
| 11 | 1 | | | | | | | | | | 1 | | |
| 12 | 1 | | | | 1 | | | | 1 | | | 1 | |
| 13 | | | | | | | | | 1 | | | | 1 |

Pour le second ensemble de sortie, nous obtenons donc :

| $f_i$ \ $x_i$ | 9 | 1 | 11 | 3 | 2 | 4 | 5 | 6 | 7 | 13 | 12 | 8 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | | | | 1 | | | | | | | |
| 2 | | 1 | | | | | 1 | | | | | | 1 |
| 3 | | | 1 | | | | | | | | 1 | | |
| 4 | | | | 1 | | | | | 1 | | | | |
| 5 | | | | | 1 | | | | | 1 | 1 | | |
| 6 | 1 | | | | | 1 | | | | | | | |
| 7 | | 1 | | | | | 1 | | 1 | | | | |
| 8 | | | | | 1 | | 1 | 1 | | | | | |
| 9 | | | | | | | | | 1 | | | | 1 |
| 10 | | | | | | | | | | 1 | | 1 | |
| 11 | | | | | 1 | | | | | | 1 | | |
| 12 | | | | | 1 | 1 | | | 1 | | | 1 | |
| 13 | | | | 1 | | | | | | | | | 1 |

Dans un dernier temps, nous associons un graphe au système. Pour le premier ensemble de sortie, nous allons obtenir :

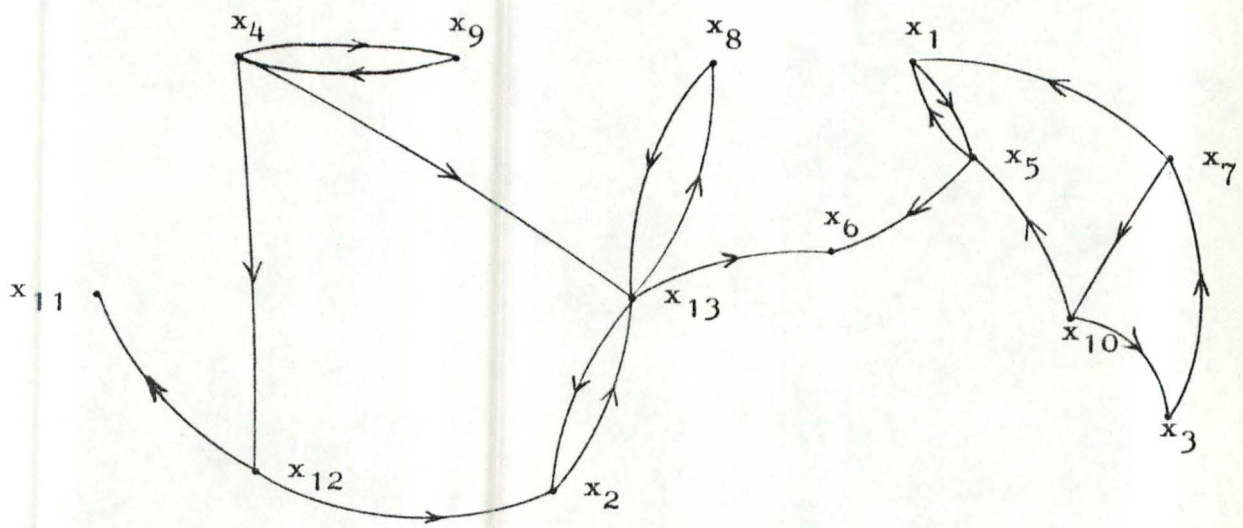



FIG. 1 : GRAPHE ASSOCIE EN FONCTION DU 1ER ENS. DE SORTIE

Pour le deuxième ensemble de sortie, cela va donc nous donner :

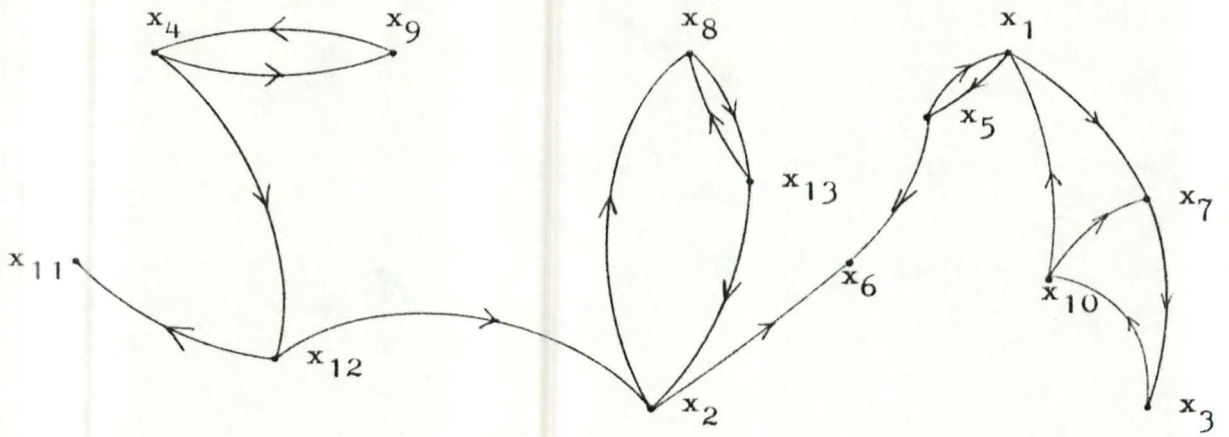



FIG. 2 : GRAPHE ASSOCIE EN FONCTION DU 2ème ENS. DE SORTIE

Nous constatons très facilement que nous sommes bien en présence de deux graphes différents.

I.1.C) Conclusion.

Sans perdre aucune généralité, nous pouvons considérer une rénumérotation des paires de sortie de telle façon qu'elles se présentent sous la forme :

$$(x_i, f_i)$$

La matrice d'occurrence du 1 - graphe (Fichefet chapitre 2 page 7) associé aura des éléments diagonaux égaux à 1 dès qu'à la ligne i, nous assignerons l'équation i.

Dans ces hypothèses, nous pouvons alors écrire :

$$f_i(x_1 \cdots x_n) = 0 \qquad x_i = f'_i(x_1 \cdots x_{i-1}, x_{i+1}, \cdots x_n)$$

Pour le 1 - graphe associé, nous aurons que l'existence de l'arc $(x_j, x_i)$ équivaut au fait que $x_j$ influence $x_i$ c'est-à-dire que $x_j \in f_i$. Il est alors à remarquer que par cette définition, aucune boucle $(x_i, x_i)$ n'apparaîtra dans le graphe.

Considérons la matrice M associée au 1 - graphe ( Berge page 123).

Nous avons vu précédemment que nous étions en présence d'un graphe sans boucle, cela a comme conséquence :

$$\forall\ i \quad M_{ii} = 0$$

Dire que $M_{ji} = 1$ équivaut à dire qu'il existe un arc allant de $x_j$ à $x_i$. L'existence de l'arc $(x_j, x_i)$ est synonyme du fait que $x_j \in f_i$. Par définition de C, nous obtenons $C_{ij} = 1$. Si $M_{ji} = 0$, nous n'avons aucun arc allant de $x_j$ à $x_i$. Par définition, nous avons que $x_j \notin f_i$ c'est-à-dire $C_{ij} = 0$.

La matrice M associée au 1 - graphe peut être définie au moyen de :

$$\left\| \begin{array}{lll} j \neq i & M_{ji} = 1 & \text{ssi } x_j \in f_i \text{ c'est-à-dire } C_{ij} = 1 \\ & M_{ji} = 0 & \text{ssi } x_j \in f_i \text{ c'est-à-dire } C_{ij} = 0 \\ \forall\ i & M_{ii} = 0 & \end{array} \right.$$

Donc à l'aide de la i ème ligne de la matrice M, nous déterminons les suivants du sommet $x_i$ tandis que dans la i ème colonne, nous allons voir apparaître ses précédents.

Si maintenant nous regardons la matrice C, nous voyons que la i ème ligne c'est-à-dire $C_i$, indique $\{x_i\} \cup \{$à ses précédents notés $\Gamma^{-1} x_i\}$ alors que la ième colonne, indique $\{x_i\} \cup \{$à ses suivants.$\}$

Si nous comparons ces deux matrices, nous voyons que pour passer de M à C, il faut faire réapparaître des "1" pour les $x_i$ diagonaux et ensuite il faut transposer les lignes et les colonnes de façon à obtenir les précédents en ligne et les suivants en colonne.

Cela nous donne donc :

$$C = (I \oplus M)^{\text{transposé}}$$

avec $\oplus$ = somme booléenne (Berge page 133)

en effet : Par $(I \oplus M)$, et à cause de la définition de la somme booléenne de deux matrices, nous obtenons pour les éléments non diagonaux ceux de M et pour les diagonaux ceux de I.
Donc pour la i ème colonne de $(I \oplus M)$, nous avons $\{x_i\} \cup \{$à ses précédents.$\}$

Donc la transposé va nous donner, sur la i ème ligne, $\{x_i\} \cup \{$les précédents$\}$ ce qui par définition est la matrice C.

Remarque

. Dans la suite du travail, dans un seul but de facilité, nous emploierons la terminologie de "matrice d'occurrence associée à un graphe" —— de matrice associée M. Il s'agit simplement de la matrice $(\perp \oplus M)^T$

. Pour vérifier s'il existe un flux d'information "directe" de $x_j$ à $x_i$, il suffit de considérer la j ème colonne de la matrice C et de vérifier qu'à la i ème ligne nous avons bien un "1". ( $i \neq j$ )

1.2) Bloc triangularisation d'une matrice d'occurrence.

Nous travaillons avec une matrice d'occurrence C définie précédemment.

1.2.A) Définition:

1) Nous appelons permutation symétrique d'une matrice E carrée, une permutation des lignes de E suivie d'une permutation analogue des colonnes.

2) Nous dirons qu'une matrice d'occurrence C n x n, associée à la résolution d'un système de n équations à n inconnues, peut être mise sous la forme bloc - triangulaire si, par une permutation symétrique de C, nous pouvons amener C sous la forme C' - fig. 1

$$C' = \begin{bmatrix} C_{1,1} & & & \\ C_{2,1} & C_{2,2} & & 0 \\ \vdots & \vdots & & \\ C_{k-1,1} & C_{k-1,k-1} & & \\ C_{k,1} & C_{k,k-1} & C_{kk} & \end{bmatrix}$$

fig. 1

où C ii, i = 1, ... k) est une sous matrice carrée d'ordre $\theta_i$ (avec des 1 sur la diagonale puisque C, qui a des 1 sur la diagonale, est permutée symétriquement) aux lignes correspondant à $C_{ii}$ sont associées $\theta_i$ équations que l'on regroupe dans le sous-ensemble $g_i$ de f.

Aux colonnes correspondant à $C_{ii}$ sont associées $\theta_i$ variables que l'on regroupe dans le sous-ensemble $z_i$ de x de telle sorte que les égalités suivantes sont vérifiées :

$$\bigcup_{i=1}^{k} g_i = f \qquad \bigcup_{j=1}^{k} z_i = x$$

$$g_i \cap g_j = \emptyset \quad ; \quad z_i \cap z_j = \emptyset \qquad i,j = 1, \ldots k$$
$$i \neq j \qquad\qquad i \neq j$$

$$\sum_{i=1}^{k} \theta_i = n$$

3) Le couple $(z_i, g_i)$ associé à $C_{ii}$ sera appelé <u>sous-système</u> et noté $S_i$.

I.2.B) <u>Intérêt de la bloc triangularisation</u>:

La résolution du système S, auquel est associée la matrice C peut être effectuée plus aisément.

En effet, considérons les sous-ensembles d'équations associés aux sous-matrices $C_{ii}$ de C; on peut écrire :

$$g_i(z_1, z_2, \ldots z_i) = 0 \quad i = 1, \ldots k$$

<u>Procédons comme suit</u> :

* Nous pouvons résoudre les $\theta_1$ premières équations par rapport aux $\theta_1$ premières variables ($\theta_1$ équations à $\theta_1$ inconnues).

* Les $\theta_2$ équations suivantes peuvent être résolues par rapport aux $\theta_2$ variables suivantes; les $\theta_1$ premières variables, pour lesquelles nous possédons une solution, apparaissent comme paramètres.

* Nous résolvons les sous-ensembles d'équations suivants pour les sous-ensembles de variables suivants de la même manière.

<u>Note</u> :

Dans le schéma de résolution que nous venons de décrire nous omettons, pour la clarté de l'exposé, les redondances éventuelles apparaissant dans les systèmes d'équations à résoudre.

Il faut bien sûr espérer que les blocs $C_{11}$, ... $C_{kk}$ sont de taille nettement plus petite que le système tout entier.

Appliquer des résolutions de systèmes linéaires, des méthodes "Newton - Raphson" sur chacun des petits blocs est "moins coûteux" qu'appliquer ces méthodes au système complet. En effet, le nombre d'opérations nécessitées pour la résolution d'un système n x n d'équations est souvent proportionnel à $n^3$ au moins.

Si nous avons k systèmes d'équations de taille respective $\theta_1$, ... $\theta_k$ à résoudre avec $\theta_1$, ... $\theta_k \neq 0$

$$\theta_1 + \theta_2 + \ldots \theta_k = n$$

nous avons que $\theta_1^3 + \theta_2^3 + \ldots + \theta_k^3$ = nombre d'opérations requis pour résoudre k systèmes $\leqslant n^3$ = nombre d'opérations requis pour résoudre le système globalement. (= uniquement si k=1)

Exemple :

$3 + 2 = 5$

$3^3 + 2^3 = 35 < 5^3 = 125$

I.2.C) Nous vous proposons d'examiner comment nous pouvons mettre sous forme bloc triangulaire une matrice d'occurrence C :

Dans cette section nous comptons :

a) Définir et dégager l'intérêt de découvrir les sous-systèmes irréductibles associés au système S.

En fait les équations qui leur sont associées doivent être résolues ensemble; et ils peuvent être résolus successivement.

b) Montrer comment on peut effectivement découvrir les sous-systèmes irréductibles.

c) Donner une méthode de bloc triangularisation. C'est-à-dire découvrir un ordre de résolution des sous-systèmes.

I.2.C.a) Le but fixé dans I.2.C) est la bloc triangularisation de la matrice d'occurrence C.

Définition :

1) Graphe_condensé : $G_c = (X_c, E_c)$ (Cfr. Roy IV.C.2.a)

Soit $G = (X, E)$, un 1 - graphe.

Construisons le 1 - graphe $G_c$ de la manière suivante :

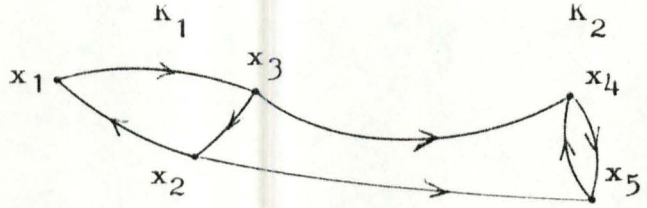
A chaque composante fortement connexe (cfr Roy IV.C.1.a)

$K_i$ de G correspond de façon bi-univoque un et un seul élément de $X_c$, soit $C_i$.

$\exists$ un arc $U_i = (t_l, t_j) \in E_c \Leftrightarrow ]x_r \in K_l, x_s \in K_j$ tq $V_m = (x_r, x_s) \in E$

Exemple : Soit le graphe G fig. 2 qui a deux composantes fortement connexes $K_1$ et $K_2$

Fig. 2




On lui associe le graphe condensé $G_c$ de la Fig. 3

Fig. 3


Supposition : Nous supposons que $G_c$ est simplement connexe (Deo p.221)

2) Un sous-système $S_i = (z_i, g_i)$ sera dit irréductible si les variables de $z_i$ sont celles correspondant à une composante fortement connexe du graphe de flux d'information associé au système S.

Notation : Nous écrivons désormais en abrégé fortement connexe : F.C., Composante F.C. : C.F.C., Sous-système irréductible : S.S.I.

Fait : Le graphe condensé associé à un graphe G est sans circuit.

Sinon soient deux C.F.C. $K_i$ et $K_j$, $i \neq j$ "sur" un même circuit. $x_k \in K_i, x_l \in K_j$. $x_k$ et $x_l$ sont mutuellement accessibles par définition de C.F.C. et de circuit. $K_i$ et $K_j$ ne peuvent pas être deux C.F.C. différents par définition de C.F.C.

Par le but de bloc triangularisation que nous nous fixons, nous aimerions que la matrice d'occurrence associée au graphe $G_c$ soit triangulaire.

3) Définition : Une matrice L est dite irréductible si on ne peut pas, par une permutation symétrique, l'amener sous la forme L' fig. 4.

$$\begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}$$ Fig. 4

où $L_{11}$ et $L_{22}$ sont des sous-matrices carrées.

Théorème 1 (cfr. Varga p.20)

Une matrice D est irréductible

$\Longleftrightarrow$

Son graphe orienté G (D) est F.C.

Conséquence de ce théorème :

Par contra position, le graphe $G_c$ étant sans circuit sa matrice d'occurrence associée $D(G_c)$ est réductible. Bien plus, elle peut être mise sous forme triangulaire.

En effet : Supposons qu'on ne puisse pas le faire. Soit la matrice $D'(G_c)$ une permutation de $D(G_c)$ constituée de i blocs irréductibles $T_i$ avec i éventuellement = 1.

Si un bloc $T_k$ n'a pas la taille 1, le sous-graphe de $G_c$ qu'on lui associe est F.C. Sinon, par le théorème 1 de cette section, on pourrait de nouveau décomposer, par permutation symétrique, ce bloc $T_k$ en 2 blocs $T_{k1}$ et $T_{k2}$ plus petits.

Mais $G_c$ est sans circuit; donc ses sous-graphes fortement connexes comportent 1 seul sommet.

Résultat :

Soit C la matrice d'occurrence du système d'équations. Une manière de la bloc triangularisiser est celle-ci :

- rechercher les C.F.C. du graphe G de flux d'information, graphe qui a pour matrice associée $C^T$ dont on a ôté les 1 diagonaux. Ces C.F.C. sont les mêmes que celles du graphe G' associé à la matrice C dont on a enlevé les 1 diagonaux. Ce graphe G' est en effet le graphe G où le sens des arcs est inversé. Ce qui ne change en rien les circuits et C.F.C.
- Former le graphe condensé $G_c$ associé au graphe G.
- Chercher une triangularisation de la matrice d'occurrence $C_{Gc}$ associée au graphe $G_c$.
- "Dé-condenser" la matrice $C_{Gc}$ pour trouver C' (proposée à la figure 1 section I.2.A).

Pour cela :

- Remarquer que la ligne i, colonne i de $C_{Gc}$ correspond à 1 C.F.C. $k_1$ (par construction de $C_{Gc}$) et un sous-système irréductible $S_1$.
- On forme la matrice C' de la manière suivante : on remplace la matrice $C_{Gc}$ par une matrice C' tq fig. 1 de I.2.A. au bloc $C_{ii}$ de C' correspond la C.F.C. $K_1$ assignée à la i ème ligne de $C_{Gc}$.
  Aux lignes du bloc $C_{ii}$ on fait correspondre les équations qui sont associées au sous-système irréductible associé à $K_1$.
  Aux colonnes $j_1$, ... $j_r$ du bloc $C_{ii}$ on fait correspondre les variables de sortie des équations qui correspondent maintenant aux lignes $j_1$, ... $j_r$.

Remarque :

1) La définition 2 donnée en I.2.C. de sous-système $S_i$ irréductible correspond à la notion d'irréductibilité de sous-matrice introduite dans la définition 3 en I.2.C.

Au sous-système $S_i$ correspond une C.F.C. $K_i$, du graphe $G_C$, de graphe $G_{Ki}$ auquel est associé une matrice $M_{Ki}$.

Cette matrice est irréductible; en effet, par le théorème 1 de cette section, comme $G_{Ki}$ est F.C...

2) Signification de l'irréductibilité. Si nous reprenons la signification du graphe de flux d'information exposée dans la section I.1. nous remarquons que les sommets d'une C.F.C. de $G_C$ correspondant à des variables dont il faut connaître, en cours de résolution du système d'équations, simultanément les valeurs respectives, autrement dit, qu'on ne peut résoudre successivement. (Les sommets d'une C.F.C. sont mutuellement accessibles).

3) On ne prend pas des "morceaux" $K_{li}$ de C.F.C. pour leur faire correspondre un sous-système. Des sommets hors de $K_{li}$ et ceux de $K_{li}$ étant mutuellement accessibles, ils correspondraient à des variables devant être résolues simultanément.

Considérer pour des sous-systèmes des "morceaux" de C.F.C. impliquerait que ces sous-systèmes devraient être résolus avec d'autres sous-systèmes.

4) Le graphe de flux d'information condensé étant sans circuit,

le flux d'information entre les sous-systèmes irréductibles va dans un seul sens. Bien plus, la triangularisation de la matrice $D(G_c)$ implique, si on se rappelle qu'elle est une matrice de flux d'information, que les sous-systèmes peuvent être résolus successivement. Ce qui était espéré.

I.2.C.b) Le problème de chercher des sous-systèmes irréductibles se ramène à proposer un algorithme de recherche des C.F.C. d'un 1 - graphe orienté.

Algorithme de Tarjan : Parmi les algorithmes actuellement existants, celui de Tarjan (cfr. Tarjan) parait le plus efficace, en vue d'une propriété qu'il possède :

. Quand on exécute l'algorithme à l'aide d'un ordinateur, le temps de calcul et la place de mémoire qu'il nécessite sont liés au nombre de sommets et d'arcs du graphe par une fonction linéaire.

Cette propriété est particulièrement intéressante quand on compte décomposer un système de grande taille.

Note : Dans ce qui suit, traverser un arc orienté signifie se déplacer le long de l'arc orienté en respectant son sens.

Soit G un 1 - graphe orienté

Technique utilisée par Tarjan :

depth - first - search ou backtracking.

search : exploration du graphe G. Consiste à procéder de telle sorte que l'on traverse tous les arcs de G chacun une et une seule fois.

depth - first - search : exploration pendant laquelle l'arc à traverser à chaque pas est toujours un arc non exploré dont l'origine est le sommet le plus récemment atteint.

L'algorithme de Tarjan

- néglige les arcs qui sont superflus pour la recherche des C.F.C. Il construit des "arbres" (voir définition dans Roy V.D.1) et classifie les arcs qui ne sont pas superflus à la recherche des C.F.C. en
  - arcs d'un arbre
  - arcs qui ne sont pas d'un arbre
- utilise un "stack", liste linéaire qui se caractérise par le fait que, lors de l'insertion ou de la destruction d'éléments, le dernier élément introduit dans la liste en est le premier

sorti. (technique "LIFO" : last - in - first - out)

- numérote les sommets consécutivement dans l'ordre suivant lequel on les atteint pendant l'exploration.

Cette numérotation sert pendant l'algorithme.

Enoncé de l'algorithme

Etape 1 Nous appellerons sommet actif de graphe, un sommet que l'on fixe comme origine du prochain arc à explorer.

Si il existe un sommet non encore numéroté, le prendre comme sommet actif; le numéroter; le mettre dans le stack.

Sinon arrêter.

Etape 2

- Choisir un arc non explré ayant pour origine le sommet actif $\gamma$.

Etape 3

Cas a : L'arc choisi nous mène à un sommet $\pi$ non encore numéroté. $(\gamma, \pi)$ est un arc d'un arbre; numéroter $\pi$ ; le mettre dans le stack prendre $\pi$ comme sommet actif. Aller à l'étape 2.

Cas b : L'arc choisi nous mène à un sommet $\pi$ déjà numéroté et on ne peut pas atteindre $\pi$ à partir de $\gamma$ en traversant un nombre quelconque ( $\geqslant 0$) d'arcs d'un arbre.

$(\gamma, \pi)$ est un arc qui n'est pas d'un arbre. Aller à l'étape 2.

Cas c : L'arc choisi mène à un sommet $\pi$ déjà numéroté et on peut atteindre $\pi$ à partir de $\gamma$ en traversant un certain nombre ( $\geqslant 0$) d'arc d'un arbre.

Alors, on néglige l'arc $(\gamma, \pi)$ . Aller à l'étape 2.

Cas d : Il n'y a plus d'arcs non explorés d'origine $\gamma$ .

Alors :

i évaluer L O W L I N K $(\gamma) \triangleq$ min {numéro du sommet $\gamma$, numéro du (des) sommet (s) $\pi$ tq $\pi$ est dans le stack et tq on peut atteindre $\pi$ à partir de $\gamma$ en traversant un certain nombre ( $\geqslant 0$) d'arcs d'un arbre et ensuite exactement 1 arc qui n'est pas d'un arbre.}

ii Si L O W L I N K $(\gamma)$ = numéro du sommet $\gamma$ , alors, détruire du stack $\gamma$ et tous les sommets qui viennent après $\gamma$ dans le stack parce qu'ils appartiennent tous à la même C.F.C.

Si le stack est, après cela, vide, aller à l'étape 1.

Sinon aller à l'étape 4

Si L O W L I N K $(\gamma) \neq$ numéro du sommet $\gamma$ , alors,

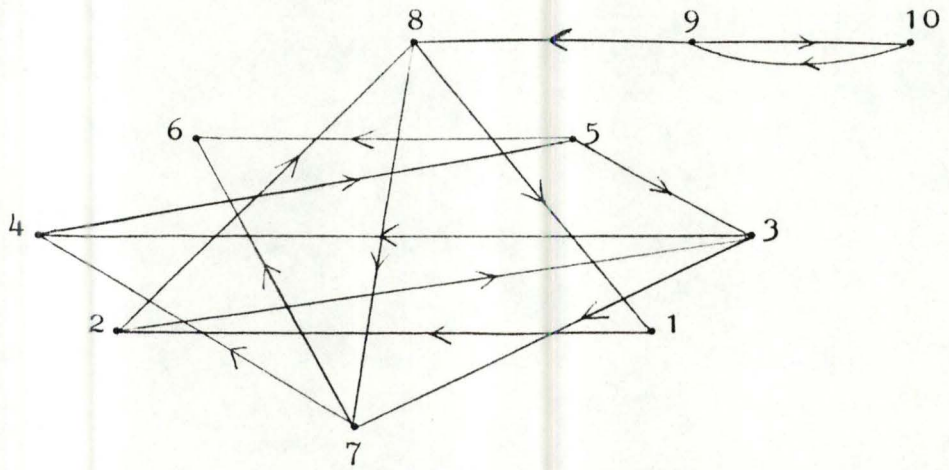
aller à l'étape 4.

Etape 4

Prendre le sommet actif précédent $\nu$ comme sommet actif.

Aller à l'étape 2.

Nous ne démontrons pas que l'algorithme donne effectivement des C.F.C. Nous montrons son fonctionnement à l'aide d'un exemple.




Dictionnaire du graphe (voir déf. dans Roy III.A.2.C.)

| | |
|---|---|
| 1 | 2 |
| 2 | 3,8 |
| 3 | 4,7 |
| 4 | 5 |
| 5 | 3,6 |
| 6 | |
| 7 | 4,6 |
| 8 | 1,7 |
| 9 | 8,10 |
| 10 | 9 |

Enumération des opérations effectuées en suivant l'algorithme.

Notation : - i (j) signifie i est affecté du numéro j

- LL (i) = j signifie : on affecte le sommet i d'un L Ø W L I N K = j

- ST (i) = j signifie : le ième sommet du stack est le sommet j

(i,j) = A.A. signifie = arc (i,j) = arc d'un arbre

(i,j) = N.A.A. signifie = arc (i,j) n'est pas un arc d'un arbre

(i,j) = NEGL. signifie = on néglige l'arc (i,j)

On aura successivement :

1 ( 1 ) ; ST (1) = 1

2 ( 2 ) ; ST (2) = 2; (1,2) = A.A.

3 ( 3 ) ; ST (3) = 3; (2,3) = A.A.

4 ( 4 ) ; ST (4) = 4; (3,4) = A.A.

5 ( 5 ) ; ST (5) = 5; (4,5) = A.A.; (5,3) = N.A.A.

6 ( 6 ) ; ST (6) = 6; (5,6) = A.A.

Etat du stack : { 1,2,3,4,5,6

LL (6) = 6

{6} est une C.F.C. que l'on retire du stack.

Etat du stack : { 1,2,3,4,5

LL (5) = 3

LL (4) = 3

7 ( 7 ) ; ST (6) = 7; (3,7) = A.A.; (7,4) = N.A.A.; (7,6) = N.A.A.

LL (7) = 4

LL (3) = 3

Etat du stack : { 1,2,3,4,5,7

{3,4,5,7} est une C.F.C. que l'on retire du stack

Etat du stack : { 1,2

8 ( 8 ) ; ST (3) = 8; (2,8) = A.A.; (8,1) = N.A.A.; (8,7) = N.A.A.

LL (8) = 1

LL (2) = 1; LL (1) = 1

Etat du stack : { 1,2,8

{1,2,8} est une C.F.C. que l'on retire du stack

Etat du stack : {

9 ( 9 ) ; ST (1) = 9

(9,8) N.A.A.

10( 10) ; ST (2) = 10; (9,10) = A.A.; (10,9) = N.A.A.

LL (10) = 9

LL (9) = 9

Etat du stack : { 9,10

{9,10} est une C.F.C. que l'on retire du stack

Etat du stack : {

Stop.

Dans ce qui suit, nous confondons volontairement une variable $x_i$ du système d'équations à décomposer et le sommet $x_i$ du graphe de flux d'information correspondant. Le contexte permet de faire la différence.

Remarque : Il existe dans la littérature d'autres algorithmes de recherche de C.F.C. tel que celui que nous allons décrire (Deo p.300)

Nous considérons un système de n équations $f_i = o, i = 1, \ldots n$ à n inconnues $x_i$, $i = 1, \ldots n$ avec n paires de sorties $(x_i, f_i)$.

Nous avons défini dans la section I.1. le graphe de flux d'information d'un tel système. C' est un 1 - graphe orienté.

La matrice booléenne M associée à un 1 - graphe orienté se définit comme suit : (cfr. Roy III.A.2.C.). $M = [m_{ij}]$ avec $m_{ji} = 1$ si $\exists$ un arc $(x_j, x_i)$. $m_{ji} = 0$ sinon. C'est un matrice carrée.

Si on se rappelle (cfr. section I.1.) la définition de matrice d'occurrence C d'un système d'équations, on remarque que

$C = I \oplus M^T = (I \oplus M)^T$ (1).

En effet : $C_{ij} = 1$ si il existe un arc $(x_j, x_i)$ dans le graphe de flux d'information.

$= 0$ sinon

et $\forall\, i,\ C_{ii} = 1$

Soit $f_i\,(x_1, \ldots x_i, \ldots x_n) = 0$, avec variable de sortie $x_i$, une équation du système.

$$x_i = f'i\,(x_1, \ldots x_{i-1}, x_{i+1}, \ldots x_n)$$

Définition : 1) $\Lambda_i = \{ x_j \text{ tq } x_j \in f'_i \}\ i \in \bar{n}$ = des sommets qui influencent directement $x_i$

Si on emprunte les notations de définitions de $\Gamma$ dans (Berge p.5) et si on se réfère au graphe de flux d'information, $\Lambda_i = \Gamma x_i^{-1}$

$\Lambda_i$ correspond à la i ème colonne de M, ou bien, suivant l'équation (1), la i ème ligne de C dont on a annulé l'élément diagonal.

2) $\hat{\Lambda}_i = \{x_i\} \cup \Gamma^{-1} x_i \cup \ldots \cup \Gamma^{-(n-1)} x_i$

$= \hat{\Gamma}^{-1} x_i$

= ensemble des sommets qui influencent $x_i$, directement ou indirectement

3) $\bar{C} = C^{n-1}$

$= (I \oplus M^T)^{n-1}$

$= I \oplus M^T \oplus M^{T2} \oplus \ldots \oplus M^{T\,n-1}$

$= (I \oplus M^T)^{n-1}$

Un théorème de théorie des graphes (cfr. Deo p. 300) nous affirme que $\bar{C}_i = 1$ ssi il existe un chemin allant de $x_j$ en $x_i$ c'est-à-dire ssi $x_j \in \hat{\Lambda}_i$.
La i ème ligne de $\bar{C}$ permet d'identifier $\hat{\Lambda}_i$

Note : $\bar{C}$ est appelée matrice booléenne associée à la fermeture transitive du graphe de matrice booléenne associée $M^T$.
Connaissant une méthode pour obtenir $\hat{\Lambda}_i$, i = 1, ... n, nous pouvons, en tenant compte du lemme suivant, identifier les C.F.C. du graphe de flux d'information.

Lemme 1 $\left.\begin{array}{l} x_i \in \hat{\Lambda}_j \\ x_j \in \hat{\Lambda}_i \end{array}\right\} \Longleftrightarrow x_i, x_j, \in$ à la même C.F.C.

En effet :

$x_i \in \hat{\Lambda}_j$ veut dire, $\exists$ un chemin allant de $x_i$ en $x_j$
$x_j \in \hat{\Lambda}_i$ veut dire, $\exists$ un chemin allant de $x_j$ en $x_i$
donc $x_i$ et $x_j$ $\in$ à la même C.F.C.

Cependant, calculer la matrice $\bar{C}$, quand on compte décomposer des systèmes à grande échelle, est peu sensé. Le nombre de multiplications booléennes qui exige ce calcul est $O(n^3)$ où n est la taille de C.
Kevorkian (Kev I.) propose une méthode d'indentification de sous-système, assez analogue à celle que nous venons de présenter : $\hat{\Lambda}_i$ est identifié par la ligne i de la matrice $\bar{C}'$ calculée, à l'aide de C de la manière suivante :

$$\bar{C}' = \bar{c}'_{ij} \quad \text{avec}$$

(2) $\bar{C}'_{ij} = C_{ij} \oplus C_{qj}$ si $\exists\, q,\ C_{iq} = 1$ , $q \in \{1,\ldots,n\}$

ce qui peut s'écrire :

$$\bar{C}_{ij} = C_{ij} \oplus \bigoplus_{q=1}^{n} (C_{iq} \otimes C_{qj}) \quad q = 1$$

Signification de la formule (2) :

Si $\exists$ un arc $(x_q, x_i)$ (c'est-à-dire si $x_q \in \Gamma^{-1} x_i$)
et si $\exists$ un arc $(x_j, x_q)$ (c'est-à-dire si $x_j \in \Gamma^{-1} x_q$)
ou si $\exists$ un arc $(x_j, x_i)$ ( c'est-à-dire $x_j \in \Gamma^{-1} x_i$)
ou si i = j
Alors $\exists$ un chemin de longueur $\leqslant 2$ qui mène de $x_j$ à $x_i$.

On remarque donc que $\hat{\Lambda}_i = \{x_i\} \cup \Gamma^{-1} x_i \cup \Gamma^{-2} x_i$ (3)
Si on définit $\hat{\Lambda}_i$ à l'aide de $C'$ défini par la formule (2).
Si $C = (I \oplus M^T)$ (formule 1), la formule 2 exprime que

$$\begin{aligned} C' &= C \oplus C^2 \\ &= (I \oplus M^T) (I \oplus M^T)^2 \\ &= (I \oplus M^T)^2 \\ &= (I \oplus M^T \oplus M^T)^2 \end{aligned}$$

Le lemme 1 ne permettra pas d'indentifier les C.F.C. si on prend $\hat{\Lambda}_i$ tq formule 3

$$\left.\begin{array}{l} x_j \in \{x_i\} \cup \Gamma^{-1} x_i \cup \Gamma^{-2} x_i \\ x_i \in \{x_j\} \cup \Gamma^{-1} x_j \cup \Gamma^{-2} x_j \end{array}\right\} \Rightarrow$$

$x_i$ et $x_j$ dans la même C.F.C.; la réciproque n'est pas vraie!
Si $x_k$ et $x_l \in$ à la même C.F.C. et sont mutuellement accessibles par des chemins de longueur $> 2$, il ne serait pas, suivant cette méthode, identifés dans la même C.F.C.
On ne permet, par cette méthode, que l'identification de sous-ensembles de C.F.C.
Une bloc triangularisation de la matrice d'occurrence n'est plus possible; deux sous-ensembles de la même C.F.C. sont reliés par un circuit dans le graphe condensé.
La prudence est donc de rigueur!
Kevorkian après avoir avoir répété l'erreur en 1975 l'a cependant corrigée.

Note importante :

Steward (Stew, I - 1962) a prouvé que les sous-systèmes irréductibles ne dépendent pas de l'ensemble de sortie que l'on se fixe.

I.2.C.c) Bloc triangularisation de la matrice d'occurrence d'un système d'équations.

Nous avons remarqué précédemment (voir I.2.C.a) que le graphe condensé $G_c$ est sans circuit.
Nous pouvons alors associer à chaque sommet un nombre appelé son rang (cfr. Roy - V.C.2.a)
Algorithme de détermination du rang d'un sommet dans un graphe = $G(X,E)$ supposé sans circuit.

a) Sélectionner tous les sommets de G sans précédent; ces sommets forment une classe $C_o$ et on leur assigne un rang = 0.

b) Former le sous-graphe de G ayant pour sommets $X / C_o$. Sélectionner dans ce sous-graphe (Cfr. Berge p. 6 ) tous les sommets sans précédents. Ils forment un ensemble $C_1$ de sommets auxquels on attribue le rang 1.

c) Si $C_i$ est le dernier ensemble obtenu, on considère le sous-graphe de G ayant pour sommets $X / C_o / C_1 / \ldots / C_i$. De ce sous-graphe, on sélectionne les sommets sans précédents; ils forment un ensemble $C_i + 1$ de sommets auxquels on attribue le rang $i + 1$.

d) Dès que l'ensemble des sommets à sélectionner est vide, le processus s'interrompt naturellement. Si il reste des sommets dont le rang n'a pas été calculé, le sous-graphe qu'ils engendrent contient un ou plusieurs circuits.

Pour preuve , voir (Roy V.C.2.a).

L'algorithme qui détermine ce rang est d'ailleurs (cours de Mr Fichefet - Algo III.3 - p.III 10) l'algorithme qui détermine des niveaux dans un graphe (voir définition p.III.9 du cours de Mr Fichefet).

Définition : Nous appelons niveau hiérarchique i, $i \geqslant 0$ du graphe $G_c$ tout sous-ensemble des sommets du graphe $G_c$, composé de tous les sommets de rang i, $i \geqslant 0$.

Propriétés des éléments d'un niveau : Par construction (voir Roy - V.C.2.a et Mr Fichefet p.III.9 - III.10) on a entre autres que :

1) Il n'existe aucun arc entre deux éléments d'un même niveau.

2) Tout élément d'un niveau i, $i \neq 0$, est suivant d'au moins un élément du niveau $i - 1$.

3) Si le graphe comporte r niveaux : 0,1, ... r-1, pour $0 \leqslant i < r-1$ tout élément du niveau i n'est suivant d'aucun élément d'un niveau j, $i < j \leqslant r-1$.

La hiérarchie a, ici le même sens que par exemple, hiérarchie militaire : Officiers supérieurs → officiers inférieurs → sous-officiers → soldats.

Nous pouvons maintenant proposer une construction de la matrice d'occurrence triangulaire $D'(G_c)$ associée au graphe $G_c$ (voir section I.2.C.a).

Considérons $G_c$ décomposé en niveaux hiérarchiques

- Si le niveau 0 de $G_c$ comprend $l$ sommets, on fait correspondre biunivoquement à chacune des $l$ premières lignes (et des $l$ premières colonnes) de $D'(G_c)$ un sommet du niveau 0 de $G_c$.
- Si le niveau 1 de $G_c$ comprend $l_1$ sommets, on fait correspondre à chacune des $l_1$ lignes (et $l_1$ colonnes suivantes de $D'(G_c)$ un sommet du niveau 1 de $G_c$.
- Ainsi de suite jusqu'à épuisement des niveaux.
- L'élément (i,j) de la matrice $D'(G_c)$ sera = 1 booléen .

$\Leftrightarrow$ . en notant $x_l$ et $x_k$ les sommets qui correspondent respectivement à la ligne i et à la colonne l de la matrice $D'(G_c)$, $\exists$ dans le graphe $G_c$, un arc $(x_k, x_l)$ ou bien ssi i = j

Fait : La matrice $D'(G_c)$ obtenue de cette manière est triangulaire.

Effectivement, par contraposition :

Soit l'élément (i,j) de $D'(G_c)$ = 1 avec $j > i$ supposons qu'à i et j correspondent respectivement les sommets $x_l$ et $x_k$.

Soit $x_l$ et $x_k$ dans le même niveau hiérarchique; par la propriété 1 des niveaux hiérarchiques il ne peut y avoir d'arc $(x_k, x_l)$, donc l'élément (i,j) de $D'(G_c)$ devrait être nul.

Soit $x_l$ et $x_k$ dans des niveaux hiérarchiques différents. Alors, par construction de $D'(G_c)$, rang $(x_k)$ > rang $(x_l)$ car $j > i$. Mais c'est exclus par la propriété 3 des niveaux hiérarchiques : il ne peut exister d'arc $(x_k, x_l)$; donc l'élément (i,j) de $D'(G_c)$ devrait être nul.

Remarque :

- Au niveau i de $G_c$, on peut faire correspondre un bloc diagonal de la matrice $D'(G_c)$ (celui qui a pour lignes et colonnes les $l_i$ lignes et colonnes correspondant au niveau hiérarchique i). En vue de la propriété 1 des niveaux hiérarchiques, chaque bloc diagonal de $D'(G_c)$ est une sous matrice diagonale.
- En vue de la propriété 2 des niveaux hiérarchiques, chaque ligne correspondant à un élément d'un niveau k, $k > 0$, contient au moins 1 élément sous diagonal non nul.

Note : Si nous considérons un algorithme de "tri topologique" (cfr. Knuth 2.2.3.p. 258), nous pouvons établir, à partir du graphe $G_c$ sans circuit (qui correspond, rappelons-le (cfr. remarque 4 finale de I.2.C.a), à un graphe de flux entre les sous-systèmes à résoudre) un ordre de résolution par substitution successive des sous-systèmes.

L'avantage qu'a la décomposition en niveaux hiérarchiques sur le tri topologique est de clarifier l'ordre de résolution des sous-systèmes et leurs relations entre eux (indépendance dans un même niveau par exemple). Dans notre cas, le tri topologique n'étant pas sensiblement "moins coûteux" qu'une décomposition en niveaux, cette dernière solution est adoptée.

La décondensation exposée en I.C.2.a. nous donne maintenant à partir de la matrice $D'(G_c)$ triangulaire, une matrice $C'$ bloc triangulaire.

Remarque :

La supposition de simple connexité de $G_c$ (cfr.I.C.2.a) propose que (Deo p. 221), que $C'$ ne peut être mise par permutation, sous forme $C' = \begin{bmatrix} C_{11} & 0 \\ 0 & C_{22} \end{bmatrix}$. Cette simplification permet de clarifier l'exposé et n'est pas contraignante (la théorie des graphes fourni des algorithmes de recherche de composantes simplement connexes (Deo p.274).

Résultat final :

0) Nous nous sommes donné la matrice d'occurrence C d'un système à grande échelle S.

1) Nous avons recherché les sous-systèmes irréductible deS.

2) Nous avons condensé les sous-systèmes irréductibles et le système S.

3) Nous avons pu découvrir une hiérarchie dans le système condensé.

4) Nous avons "dé-condensé" pour obtenir un réarrangement espéré commode de la matrice C.

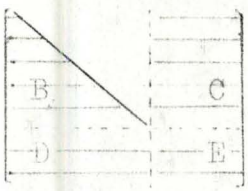
Les sous-systèmes irréductibles peuvent être de taille encore très grande. Le but du chapitre suivant est de voir si une décomposition plus élaborée de chaque sous-système irréductible est possible.

# CHAPITRE II. ETUDE D'UN SOUS SYSTEME IRREDUCTIBLE.

## II.1 Définitions et notations

. Forme triangulaire bordée

Considérons une matrice A. Cette matrice sera dite être mise sous la " forme triangulaire bordée " si par des permutations symétriques des colonnes et des lignes, nous arrivons à mettre cette matrice sous la forme:




La détermination d'une telle matrice correspond au fait que nous tâchons de résoudre d'une manière éfficace un système irréductible d' équations dont le graphe associé est fortement connexe(voir plus haut).Il s'avérera qu'un nombre minimum de colonnes du "bord" correspond à la manière la plus efficace de résoudre le système d' équa--tions.

. Ensemble essentiel

Considérons un graphe $G=(X,E)$.
L' ensemble $A \subset X$ est dit "ensemble essentiel" si le sous-graphe(Berge p.6) engendré par $X \setminus A$ est non cyclique ( Fichefet chap.2 page 23).
Pour un graphe G, il existe plusieurs ensembles essentiels On appelle alors "ensemble essentiel minimum" celui de tous ces ensembles qui contient un minimum d' éléments.

. Graphe complet

Un graphe $G=(X,U)$ est dit "complet" si

$$(x,y) \notin U \Rightarrow (y,x) \in U$$

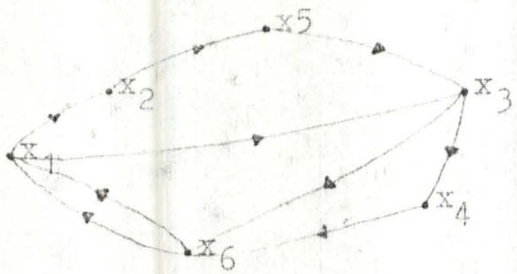
cad tout couple de sommets est relié dans au moins une des deux directions.

. Doublet

Considérons un graphe G=(X,E) . Soient deux sommets $x_i$, $x_j \in X$.
Ces deux sommets sont dit reliés par un "doublet" si les arcs$(x_i,x_j)$ et$(x_j,x_i)$ appartiennent à E.
Nous avons la situation suivante:


. Circuits dominants

Soit un graphe G=(X,E) et soient deux circuits $u_1$ et $u_2$ (Berge page 7).
Le circuit $u_2$sera appelé "circuit dominant" par rapport à $u_1$ si le circuit $u_2$ est inclus dans le circuit $u_1$.




$u_1=(x_1,x_2,x_5,x_3,x_4,x_6,x_1)$

$u_2=(x_1,x_3,x_6,x_1)$

Le circuit $u_2$est inclus dans le circuit $u_1$.
$u_2$ est le circuit dominant.

. Principe du circuit dominant

Si on casse un circuit dominant ,tous les circuits qui sont dominés seront aussi cassés.

REMARQUE

Dans la suite, "circuit brisé" =circuit qui a été cassé cad auquel on a retiré un sommet.

## II.2 Intérêt de la forme triangulaire bordée

Dans ce deuxieme chapitre, nous ne travaillons plus sur le système de n équations à n inconnues que nous avions au debut mais sur un sous-systeme irreductible. Nous avons vu dans le chapitre I que chacun de ses sous-systèmes correspond à une composante fortement connexe.

Considérons la matrice d' occurence associée à l' ensemble irréductible d' équations:

$$f_i(x_1,\ldots\ldots,x_\theta)=0 \qquad i\in I_\theta \qquad (1)$$

avec comme ensemble de sortie $(x,f)=\{(x_1,f_1),\ldots\ldots,(x_\theta,f_\theta)\}$ (2)

Dans ce chapitre , nous allons chercher une manière de la mettre sous une autre forme. Nous effectuons des réarrangements symétriques des lignes et des colonnes sur la matrice d' occurence de façon à obtenir un nombre minimum de colonnes $\beta$ ayant des éléments non nuls au-dessus de la diagonale principale avec la restriction que $(x,f)$ correspond à la diagonale principale. Un diagramme schématique de la matrice résultante est donné par la figure 1

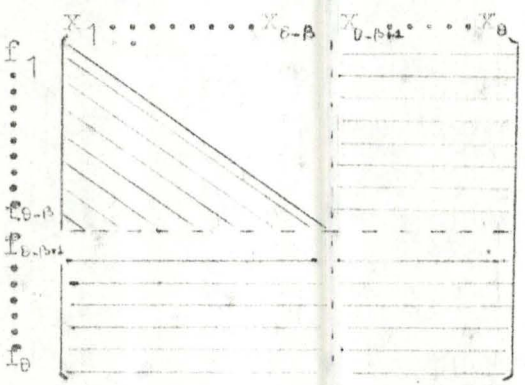



figure 1

Les équations associées à la matrice de la figure 1 peuvent être écrites sous la forme:

$$f_i(x_1,\ldots\ldots,x_i,x_{\theta-\beta+1}\ldots\ldots,x_\theta)=0 \qquad i=1,\ldots\ldots,\theta-\beta$$

$$f_j(x_1,\ldots\ldots,x_j,\ldots\ldots\ldots,x_\theta)=0 \qquad j=\theta-\beta+1,\ldots,\theta$$

avec $1\leq\beta\leq\theta-1$

$\theta=2,3,\ldots\ldots$

Il est à noter que le nombre $\beta$ ne peut jamais avoir la valeur

zéro à cause de la définition d' un ensemble irréductible d' équations.

On pose alors $v=\{x_{\theta-\beta+1},\ldots\ldots\ldots,x_\theta\}$

La permutation symétrique des lignes et des colonnes de la matrice d' occurence va nous permettre d' introduire une méthode itérative de solutions(figure 2).

en effet:

Nous fixons les $\beta$ variables $x_{\theta-\beta+1},\ldots\ldots,x_\theta$ et nous les introduisons dans les équations $f_1,\ldots\ldots\ldots,f_{\theta-\beta}$.Nous avons alors un système de $\theta-\beta$ équations à $\theta-\beta$ inconnues De plus, nous sommes en présence d' une matrice triangulaire. Nous pouvons par conséquent résoudre successivement les équations et nous obtenons les valeurs de $x_1,\ldots\ldots\ldots,x_{\theta-\beta}$.Ces valeurs sont alors introduites dans les fonctions $f_{\theta-\beta+1},\ldots\ldots,f_\theta$ ;nous obtenons un système de $\beta$ équations à $\beta$ inconnues.Nous le résolvons et donc donnons des valeurs à $x_{\theta-\beta+1},\ldots\ldots\ldots,x_\theta$ .Ce sont ces valeurs que nous réintroduisons dans l' algorithme. Il faut également déterminer un test d' arrêt:à la fin du processus, nous calculons la différence entre les valeurs que nous avons introduites et les valeurs que nous obtenons à la fin. Dés que toutes ces différences sont égales à une certaine précission,nous arrêtons le processus et considérons que les valeurs des variables à ce moment donnent la solution du système .

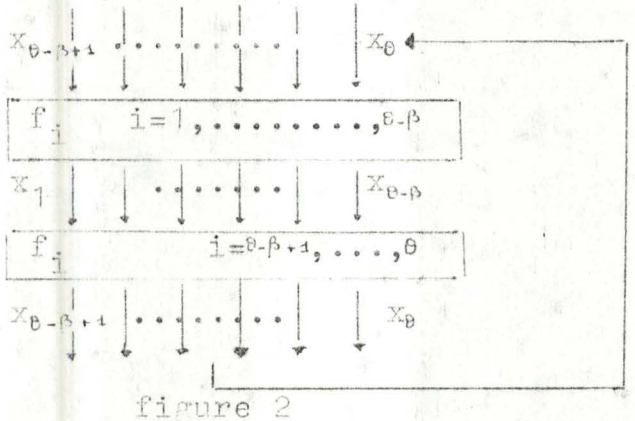
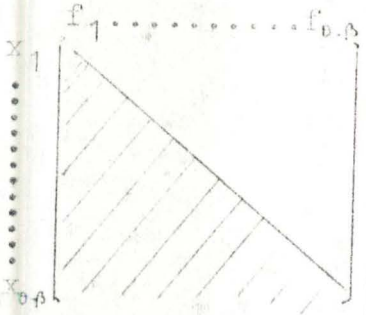



figure 2

Nous ne donnerons aucune notion de convergence de l' algorithme Il faut pourtant dire qu'en gros on cherche un point fixe d'une équation x=f(x).

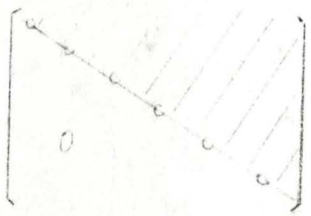
Considérons la matrice mise sous la forme triangulaire bordée. Nous avions alors vu que nous avions une sous matrice sous triangulaire. Nous allons établir sa signification.

Nous avons donc:




Par définition, il s' agit d' une matrice d' occurence C .Or nous avons vu qu'on pouvait y associer un graphe pour lequel il nous était facile de définir une matrice associée L.

Par la propriété $C=(I \oplus L)^T$ ,la matrice L sera de la forme:




Remarquons: 1)nous n' avons aucune boucle car on a des zéros sur la diagonale.

2)a-t-on des circuits?

- On constate que si l' on part d' un sommet $x_i$, on ne peut joindre que des sommets $x_j$ avec $j>i$. On ne pourra pas former de circuits.
- En calculant les puissances successi-ves, on voit que l' on garde des"0" sur la diagonale donc on n' a pas de circuits.

Si on pose $A=\{x_{\beta+1},\ldots\ldots,x_0\}$, on voit que le sous-graphe obtenu en supprimant ces sommets est non cyclique. Par définition, A est alors un ensemble essentiel.
Pour que le système résolvant de $\beta$ équations à $\beta$ inconnues ait une dimension minimale, on tâche à ce que la matrice triangulaire soit la plus grande possible.
Donc, pour le graphe, A est "1' ensemble minimum essentiel" .

Par conséquent, en terme de théorie des graphes, ceci revient à déterminer un ensemble minimum essentiel d' un graphe dirigé défini par A.

## II.2 Vocabulaire et définitions

Nous allons introduire quelques mots de vocabulaire qui apparaîtront dans la recherche de l' ensemble minimum essentiel.

- Information locale à un sommet x d'un graphe G=(X,E)

  Il s' agit de la connaissance topologique complète du sous-graphe(Berge page 6) $G(\{x\}\cup adj\{x\})$ où $adj(x)=\{y\in X$ tel que $(x,y)$ ou $(y,x)\in E\}$

- Graphe étiqueté

  Soit un graphe G=(X,E) .
  Le "graphe étiqueté" associé $\tilde{G}=(X,\tilde{E})$ est topologiquement identique à G excepté que chaque arc $(x_i,x_j)\in\tilde{E}$ porte une étiquette $\{x_i,x_j\}$ qui représente un chemin de longueur "1" de $x_i$ à $x_j$ dans $\tilde{E}$.

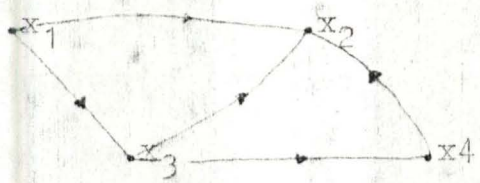
- Table de couverture

  La "table de couverture" est un tableau ayant comme indice de lignes les sommets du graphe et de colonnes les différents circuits.
  Ce tableau a des entrées booléennes. Chacun des circuits est noté en plaçant "1" aux sommets intervenant dans le circuit .

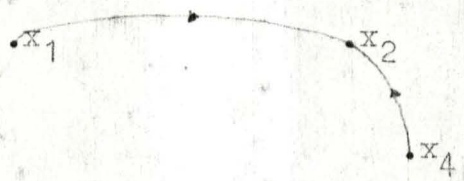
Lors de la recherche de cet ensemble minimum essentiel , nous faisons appel aux quatre transformations suivantes:

. transformation T 1:Suppression d'un sommet x

Nous enlevons dans ce cas le sommet x ainsi que ces arcs incidents(Berge page ).Il en résultera un sous-graphe G(X-{x}).



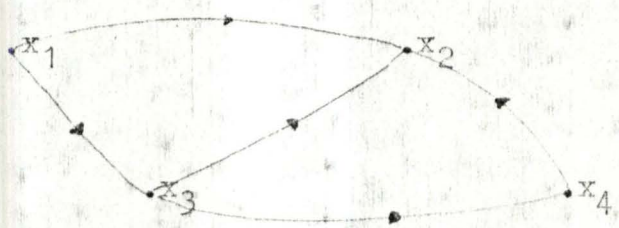

Si nous supprimons le sommet $x_3$,nous aurons en fin d' opérations le graphe:



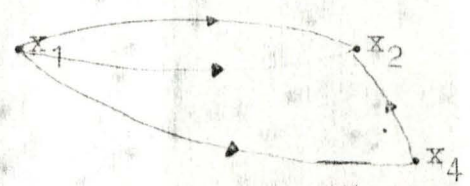

. Transformation T2:Elimination d'un sommet x

Nous éliminons le sommet x et nous ajoutons de nouveaux arcs au graphe G(X-{x}) utilisant la règle suivante:

on ajoute l' arc (z,y)à G(X-{x})
ssi (z,x) et (x,y)$\in$E

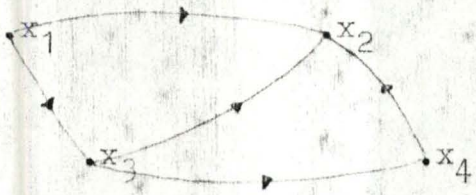



Nous allons éliminer le sommet $x_3$mais nous allons alors voir apparaitre deux nouveaux arcs $(x_1,x_2)$ et $(x_1,x_4)$



. transformation T3: Suppression d'un arc incident à x

Nous enlevons $(x,y)\in E$ ou $(y,x)\in E$ de $G(X,E)$ et nous for--mons un nouveau graphe $G(X, E\setminus\{(x,y) \text{ ou } (y,x)\})$.

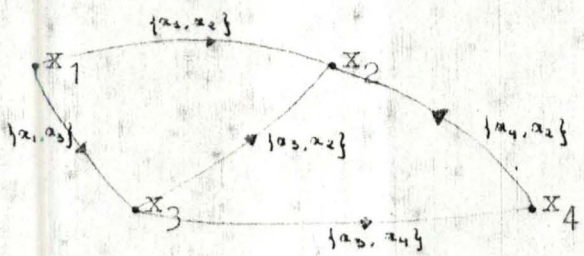



Nous allons supprimer l'arc $(x_3,x_4)$ incident au sommet $x_4$, nous obtenons par conséquent:


. transformation T4: Elimination d'un sommet x dans un graphe étiqueté

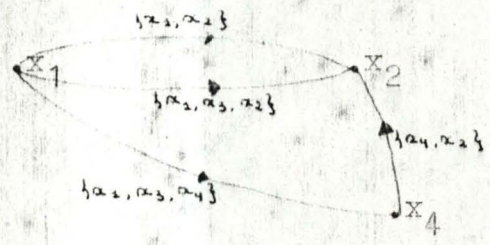
Nous supprimons un sommet x et nous ajoutons de nouveaux arcs à $G(X-\{x\})$ en utilisant la règle suivante:

Si $(z,x)$ et $(x,y)\in\tilde{E}$ alors on ajoute $(z,y)$ à $G(X-\{x\})$. Une adresse $\{z.....x.....y\}$ est assignée à ce nouvel arc où $\{z.....x\}$ et $\{x.....y\}$ sont les adresses de $(z,x)$ et $(x,y)$.




Nous allons éliminer le sommet $x_3$, nous aurons donc deux nouveaux arcs

$(x_1,x_2)$ avec l'adresse $\{x_1,x_3,x_2\}$
et $(x_1,x_4)$ avec l'adresse $\{x_1,x_3,x_4\}$




Un dernier point dans ce paragraphe va nous donner succinctement l'algorithme qui nous permet de decouvrir l' ensemble essentiel minimum.

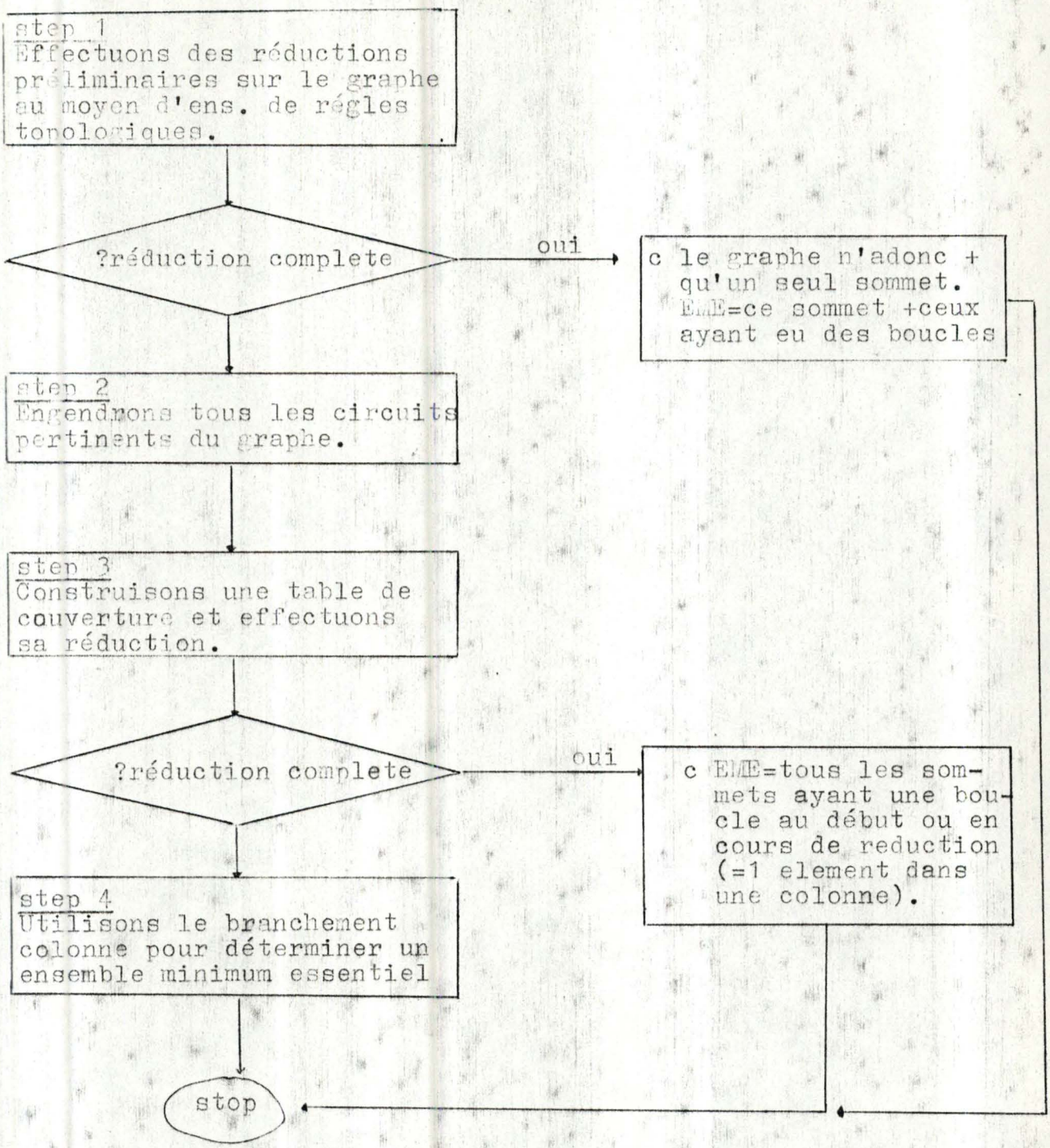
Note

1)Réduction: c'est la simplification du graphe au moyen de plusieurs transformations.

2)Réduction complète:(du graphe) au moyen de transformations diverses ,nous sommes arri--vés à éliminer tous les arcs et à ne plus avoir qu'un seul sommet.

3)Réduction complète de la table de couverture: au moyen de transformations, nous sommes arrivés à ne plus avoir qu'une seule colonne dans la table de couverture.

4)Branchement colonne:nous considérons une colonne de la table de couverture cad un circuit.Nous allons considérer un élément du circuit à partir duquel on effectue les transformations de réduction.Si nous arrivons à une réduction complète,nous aurons atteint notre but.Dans l'autre cas,nous passons àl' élèment suivant.




Nous allons dans les paragraphes suivants expliciter chacune de ces étapes.

## II.4 Réduction du graphe

Définissons et commentons les cinq règles de simplification. Rappelons le but que nous poursuivons: obtenir un sous-graphe sans circuit à partir du graphe associé à un sous-système irréductible d'équations.(ce graphe est, par définition de sous système irréductible, fortement connexe).
Nous nous servirons d' un lemme dans le paragraphe se rapportant à la regle 3(R3) de simplification.

Lemme 1

SI UN SOUS GRAPHE G' EST COMPLET ET COMPREND L SOMMETS IL FAUT ET IL SUFFIT D'EN DETRUIRE (T1) L-1 SOMMETS POUR EN BRISER TOUS LES CIRCUITS.

démonstration:par récurrence.

a)Si L=1,le lemme est vrai (le graphe étant supposé sans boucle)

b)Si L=2

C.N.

Il faut au moins détruire un sommet du graphe pour briser le circuit $\{x_1,x_2\}$


C.S.

IL est certain que détruire un sommet du graphe brise le seul circuit du graphe.

c)Supposons que le lemme soit vrai pour $L=i\geq 2$.

Démontrons qu'il est vrai pour L=i+1

Soit le graphe G ayant pour ensemble de sommets $\{x_1,x_2,\ldots\ldots\ldots\ldots,x_{i+1}\}$. Une conséquence de la définition de graphe complet est que tout sous-graphe d'un graphe complet est complet.

Soit le sous-graphe de G ayant pour sommets $\{x_1,..,x_i\}$ Il est complet.

Par l'hypothèse de récurrence, il faut et il suffit de détruire i-1 sommets pour briser tous ses cir-

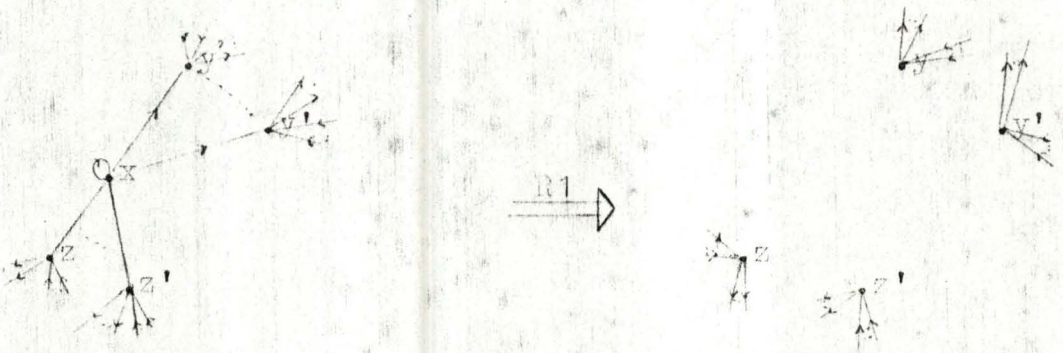
-cuits, soit par exemple les sommets $x_1, \ldots\ldots, x_{i-1}$
Il reste du graphe G les sommets $x_i$ et $x_{i+1}$ formant doublet (G etait initialement complet).
Le lemme etant prouvé pour L=2, nous avons détruit du graphe G (i-1)+1 sommets.

cqfd.

II.4.1 Enoncé des régles de simplification

. régle 1

Détruire le sommet x (T1(x)) quand il y a une boucle au sommet x. On diminue l' index de une unité.




commentaire

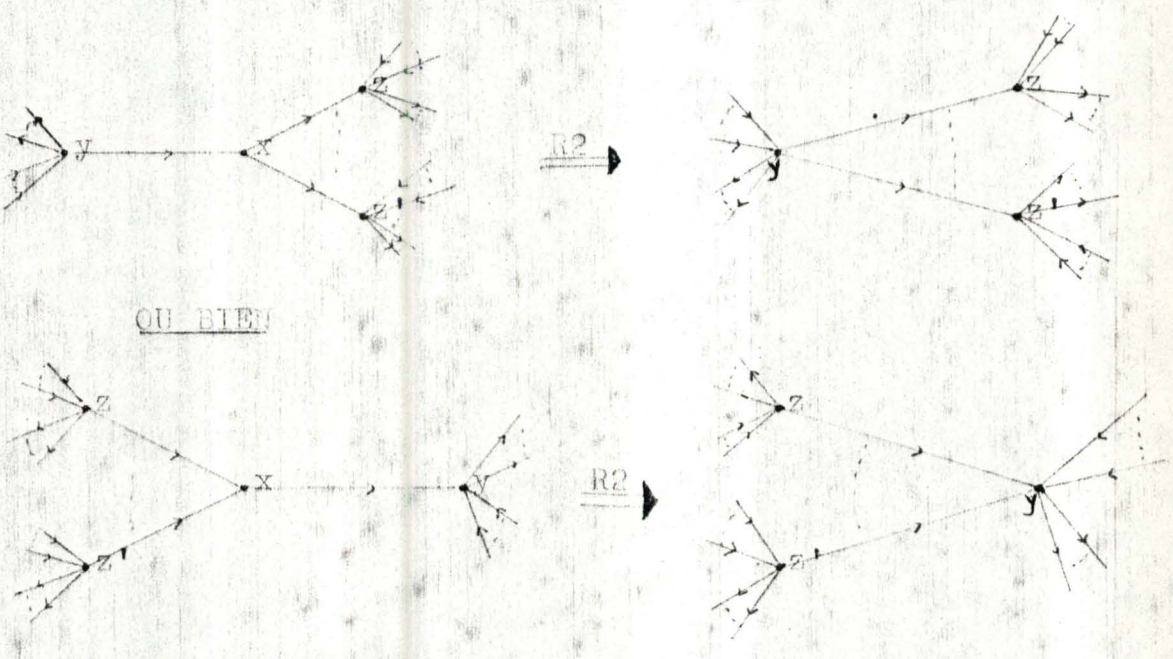
On est certain, si on a une boucle au sommet x, que la boucle provient d'élimination(T2) de sommets provoquée par d'autres régles car:

1) Au départ, le graphe est sans boucle(sous graphe du graphe de flux d' information).
2) T2 est la seule transformation appliquée au graphe, qui crée des arcs. T2 a pour effet de diminuer la longueur(Fichefet page II 12) d'un chemin entre deux sommets sans en changer le sens. Toute boucle provient d' un circuit du graphe original dont on a diminué la longueur par élimination (T2) de sommets.
Puisque ce circuit n' a pas été brisé (s'il l' avait été on n' aurait pas la possibilité d' avoir une boucle) la seule possibilité de le

briser est de détruire(T1)x et de le mettre dans l'ensemble minimum essentiel cad d'appli-quer R1.

. règle 2

Eliminer le sommet x(T2) quand le minimum de { degré intérieur de x, degré extérieur de x } est ≤ 1




commentaire

Si un circuit C1 passe par x, il passe certainement par y.

On est donc certain que détruire y brise au moins tous les circuits passant par x plus éventuellement d'autres circuits (ne comprenant pas l'arc(y,x) ou (x,y) mais passant par y).

On peut donc convenir que x n' interviendr pas dans le choix d'un ensemble minimum essentiel.

. règle 3

Eliminer le sommet x quand G({x} ∪ adj(x)) est un graphe orienté complet.

commentaire

Si le sous graphe G'=G({x} ∪ adj(x)) complet comprend

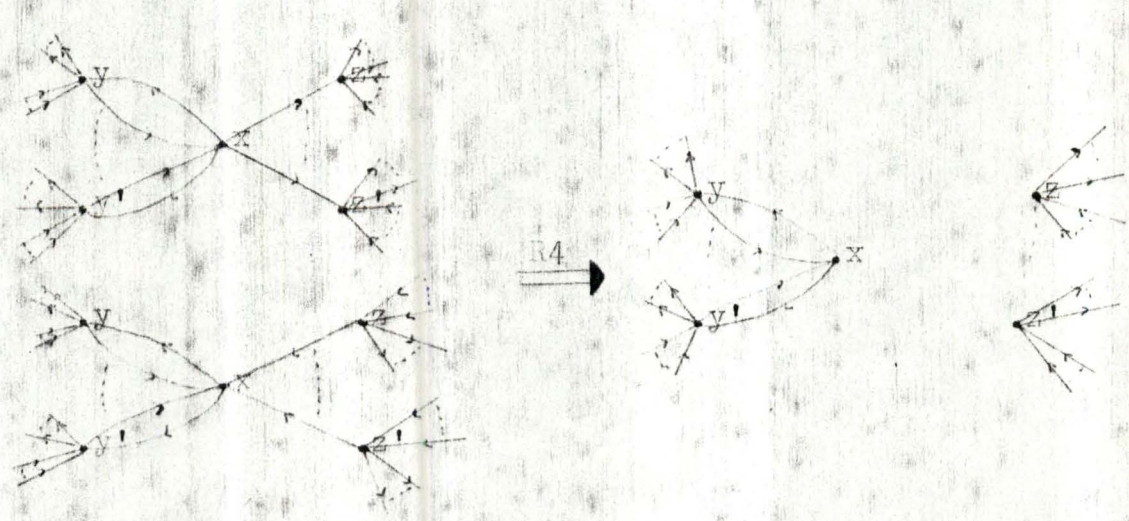
L sommets,le lemme nous dit qu'il faut et il suffit d'en détruire L-1 pour en briser tous les circuits.
Détruire(T1) le sommet x ne brisera jamais un cir--cuit dont les arcs et,à l'exception d'un sommet, les sommets ne sont pas des arcs et sommets du sous graphe G'.
Détruire(T1) les sommets de adj(x),en plus de briser tous les circuits du sous graphe(des circuits dont tous les arcs et sommets sont du sous-graphe),peut aussi briser d'autres circuits.
Un choix optimal sera donc posé si l'on choisit de détruire les L-1 sommets de adj(x).

Remarque:

En fait, si l'on applique T2à x ,cela coincidera avec notre but puisque des boucles apparaitront à chacun des sommets de adj(x) auxquels on appli--quera successivement R1.

• règle 4

Détruire(T3) tous les arcs incidents à x, excépté ceux formant des doublets si en retirant les arcs formant doublet, le minimum de {degré interieur de x ,degré ex--terieur de x} =o



commentaire

Tous les doublets $\{x,y\}$ ..... $\{x,y'\}$ doivent être brisés (ce sont des circuits) en détruisant:

-soit le sommet x

-soient les sommets $\{y,.....,y'\}$

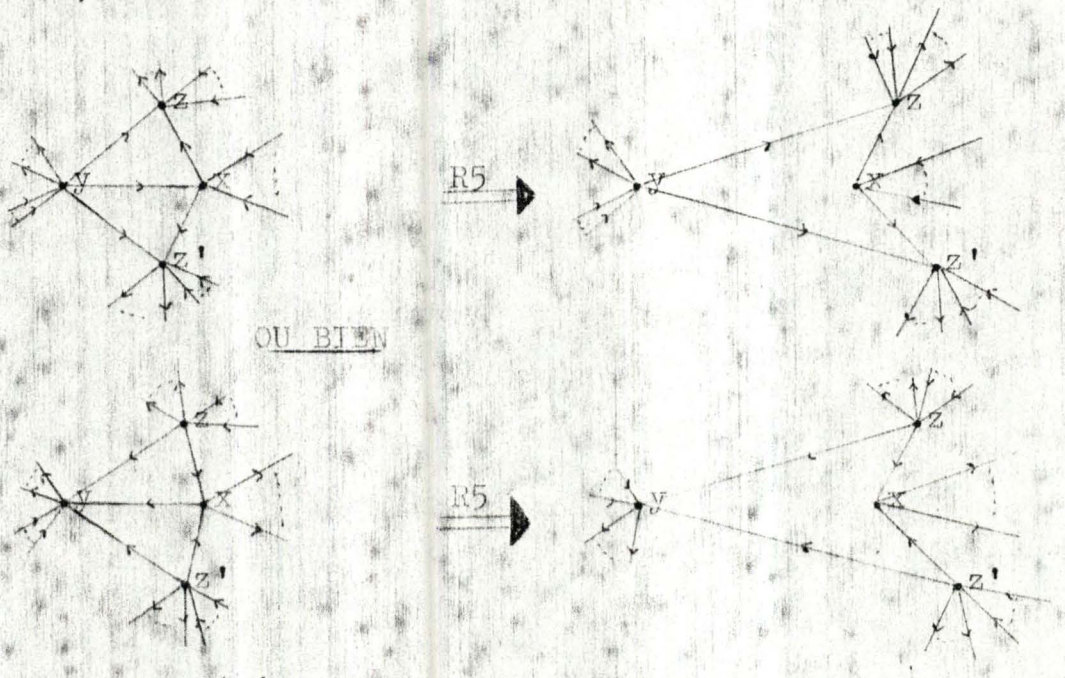
Tout circuit $C_i$ passant par (z,x) ou (x,z) doit passer par un des arcs (x,y) ou (y,x) puisque ce sont les seuls arcs sortant (entrant) en x; et $C_i$ sera donc ,dans les deux cas ,brisé.

Si on enlève les arcs (z,x) ,.....,(z',x) (ou(x,z) .....,(x,z')) du graphe ,on ne détériore pas la recherche d'ensembles minimaux essentiels.

. règle 5

Détruire(T3) l'arc $(y,x)\in E$ si $(x,y)\notin E$ et $(y,z)\in E$ quand $(x,z)\in E$.

De même, détruire l'arc (x,y) si $(y,x)\notin E$ et $(z,y)\in E$ quand $(z,x)\in E$




commentaire

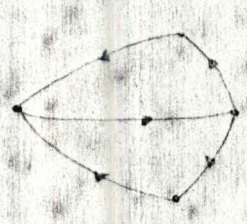
A tout circuit $C_i$ passant par (y,x) et (x,z) correspond un circuit passant par (y,z).Si on brise le circuit $C'_i$ ,on brise du même fait le cicuit $C_i$.

Si on brise le circuit $C_i$ en x, on doit encore briser le circuit $C'_i$ .Si on enlève du graphe l' arc (y,x), on ne fait que simplifier le problème de recherche d'un ensemble minimum essentiel,sans le fausser; on s'occupera de briser le circuit $C'_i$ plutôt que de briser à la fois $C_i$ et $C'_i$ .

. remarques

1)Par ces simplifications ,soit on réduit complètement le graphe; soit il reste des arcs. On est de toute façon certain que le caractère "cardinalité minimum" d'un ensemble essentiel que l'on cherche ou que l'on a trouvé est conservé.

2)Appliquer les cinq simplifications élimine des possi-bilités d' ensembles minimaux essentiels.Soit par exemple ci-dessous (fig 1) un graphe fortement connexe à simplifier.



On pourrait choisir soit x ,soity comme ensemble minimum essentiel.Cependant,appliquer R2 à y supprime définitivement la possibilité d'avoir y dans un ensem-ble minimum essentiel.De même pour x.

3)L'ordre d'application de ces régles n' a aucune importance,si ce n'est que,suivant leur ordre d' application,on peut trouver l' un oul'autre ensemble essentiel,comme le montre l'exemple (fig2) d'un graphe fortement connexe.


figure 2

Si on essaie d'appliquer R3 à un des sommets du graphe on remarque que $x_1$ est le seul sommet qui se prête à l'application de cette régle.On trouve $\{x_2,x_4\}$ comme ensemble essentiel minimum.Maintenant,essayons d' appli-quer plutôt R2 au même graphe.On peut éliminer(T2) $x_3$. Suivant le sommet auquel on appliquera R3,ontrouve alors un des ensembles minimaux suivants $\{x_1,x_2\}$ , $\{x_2,x_4\}$ ,$\{x_1,x_4\}$ .

4)On peut ,lors du processus de simplification du graphe avoir le choix entre différentes régles à appliquer au même sommet.Par exemple,au graphe da la figure 2 ,au sommet $x_3$,on peut appliquer soit R2,soit R5.

L'étape 1 de l'algorithme de Cheung et Kuh se résume à ceci: appliquer,tant que cela est possible,les régles R1 à R5 à chaque sommet du graphe àréduire ,et dans un ordre quelcon-que.En fait,nous proposons dans le chapitre IV section 2 un organigramme qui propose un ordre déterminé d'application de ces cinq régles,dans le but de ne pas essayer,inutilement d'appliquer une régle à un sommet.

## II.4.2 Algorithme de Kevorkian

Avant Cheung et Kuh ,Kevorkian(Kev 11) a proposé un algorit-hme dont la démarche est très proche de l'étape 1 de Cheung ET Kuh (C.K) .Aussi,plutôt que de décrire l' algorithme,nous préfèrons,sans citer les énoncés, montrer comment appliquer les régles R1 à R5 de C.K pour obtenir le même résultat que Kevorkian.

Cependant,nous citerons ,sans les commenter (ces commentai-res étant du même type que ceux proposés dans l'exposé des régles R1 à R5) deux corollaires de Kevorkian qui simplifient des situations où les régles R1 àR5 de C.K ne s'appliquent pas.

Nous montrerons ,sur un exemple,que Kevorkian introduit une simplification erronée.

Nous suggererons comment fonctionne l' algorithme de Kevorkian et poserons à propos d'un exemple ,la question de savoir si l'algorithme permet la réduction de tout graphe.
Nous choisirons enfin un algorithme.
La demarche de Kevorkian consiste:

1)Détruire(T1) dans le graphe, un sommet dont on sait,après examen du graphe,qu'il appartient à un ensemble minimum essentiel dont on a commencé la recherche

2)Opérer également des simplifications ,soit suppri-mer(T3) d'arcs,soit éliminer(T2) des sommets sur ce graphe.

Voici ces simplifications énoncées sous forme d'applications des régles R1à R5 de C.K.Nous gardons les mêmes notations que(Kev II) pour que le lecteur intéressé puisse vérifier que l' énoncé se transforme bien en application des régles R1 à R5 de C.K.

.partie 1

Théorème 1

Appliquer R4 à $x_r$,R2 à $x_r$,R1 à $x_p$.

Note:Les arcs incidents à $x_r$ sont détruits si on applique C.K.Ils le sont aussi chez Kevorkian(lemme 1 plus loin)

corollaireI

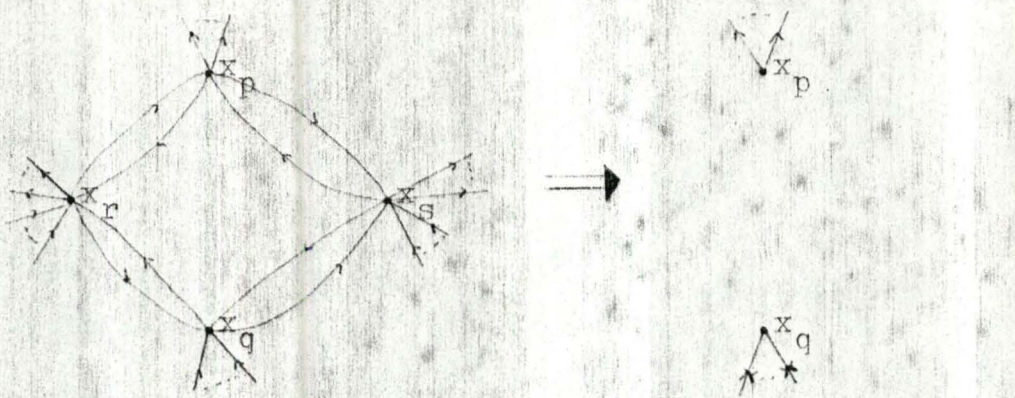
Appliquer R4 à $x_r$,R3 à$x_r$ ,R1 à $x_p$ et $x_s$.

Note:même note que pour le th 1

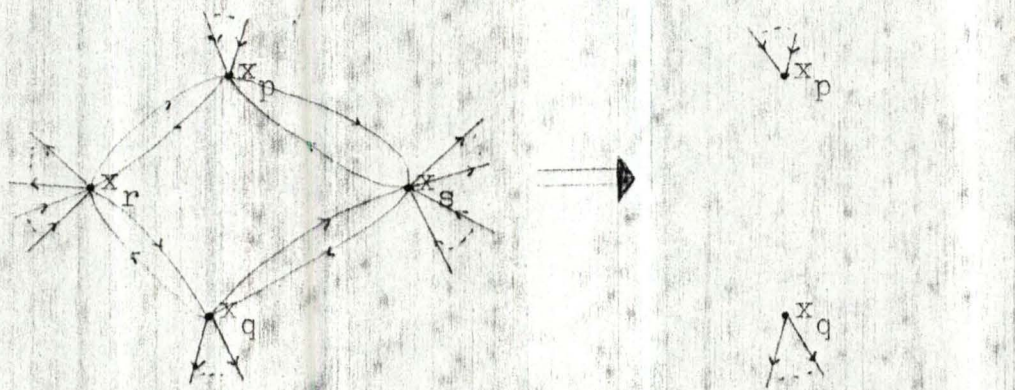
corollaireII

L'existence des doublets$\{x_p,x_r\}$ , $\{x_p,x_s\}$ , $\{x_q,x_r\}$ , $\{x_q,x_s\}$ où $\{x_p,x_q\}$ n'est pas un dou-blet et avec { soit aucun arx $(x_p,x_j)$ / soit aucun arc $(x_j,x_p)$ } et avec { soit aucun arc $(x_q,x_j)$ / soit aucun arc $(x_j,x_q)$ } avec $x_j \neq x_r$ et $x_j \neq x_s$

$\Longrightarrow$ $x_r, x_s \in F$




OU BIEN



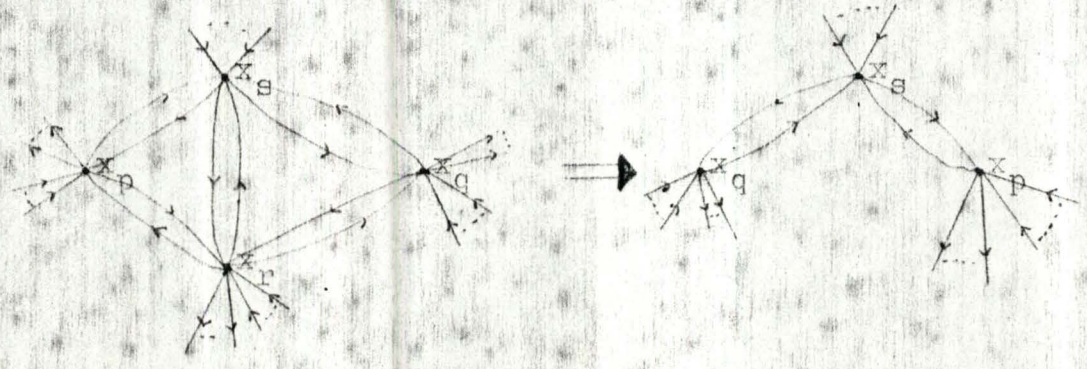

L'étape 1 de C.K permet tout au plus d'appliquer R4 à $x_p$ et $x_q$.

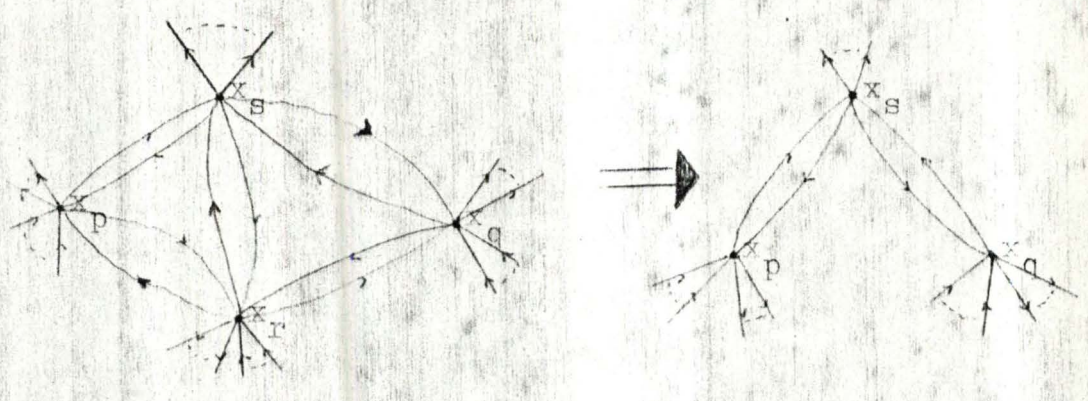
corollaireIII

L'existence des doublets $\{x_p,x_r\}$ , $\{x_p,x_s\}$ , $\{x_q,x_r\}$ , $\{x_q,x_s\}$ et $\{x_r,x_s\}$ avec

soit aucunarc $(x_s,x_j)$ ou soit aucun arc $(x_j,x_s)$ avec $x_j \neq x_p$, $x_j \neq x_q$, $x_j \neq x_r$

$\Longrightarrow x_r \in F$






Ici aussi, l'étape 1 de C.K ne peut qu'opérer au plus la simplification "R4" ausommet $x_s$.

.partie 2

lemmeI

Il s'agit de R4 de C.K.

lemmeII.a et II.b

On a la même simplification que pour R2 de C.K.

Note:Si on a un doublet $\{x_r,x_s\}$ ,on applique directement le théorème 1 de Kevorkian au lieu de R2 de C.K au sommet $x_r$ suivie de R1 de C.K au sommet $x_s$.

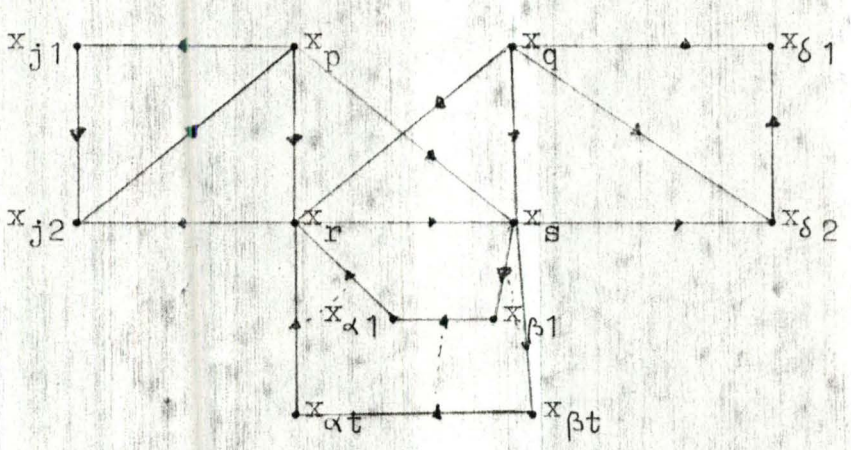
lemmeIII.a et III.b

a)Si il existe des arcs $(x_{p1},x_r),\ldots\ldots(x_{pj},x_r)$ et $(x_r,x_s)$ avec aucun doublet $\{x_r,x_{p1}\},\ldots\{x_{pj},x_r\}$ et avec des arcs $(x_{p1},x_s),\ldots\ldots,(x_{pj},x_s)$ ALORS si on supprime du graphe l' arc $(x_r,x_s)$, on ne diminue pas la taille d'un ensemble essentiel dont la recherche est en cours.

b)analogue au lemme III.a en changeant le sens des arcs.

Ces lemmes sont analogues à R5 de C.K.Cependant,R5 impose qu'il n'y ait aucun autre arc incident à $x_r$ vers l'interieur(vers l'exterieur)que ceux ayant

respectivement pour extrémité initiale(terminale) $x_{p1},.....,x_{pj}$.On s'aperçoit que cette précaution est indispensable.




posons $x_{p1}=x_p$; $x_{p2}=x_q$; $x_r=x_r$; $x_s=x_s$ pour rejoindre les notations de Kevorkian.

Prenons l'exemple du graphe fortement connexe ci-dessus ,dans les conditions du lemme III.a .

Ce graphe comporte entre'autres t circuits $x_r \longrightarrow x_s \longrightarrow x_{\beta i} \longrightarrow x_{\alpha i} \longrightarrow x_r$ avec i=1,.....,t qui doivent être brisés.

Appliquons le lemme III.a à l'arc $(x_r,x_s)$ ,qui peut donc être retiré du graphe.Si nous procédons ensuite comme suit:appliquons le lemme II au sommet $x_{j2}$;nous obtenons un doublet $\{x_p,x_r\}$ qui nous met dans la situation décrite dans le théorème 1;ce qui nous permet de détruire $x_p$ en le considèrant dans F.Si nous appliquons alors le lemme 2 au sommet $x_2$ ,nous pouvons appliquer le théorème 1 au sommet $x_q$ qui ,détruit , sera placé dans F.

Appliquer de cette manière les lois que donne Kevorkian ne permet de placer ni $x_r$,ni $x_s$ dans l'ensemble F alors que l'un des deux sommets au moins doit s'y trouver.

Pour terminer l'exposé de l'algorithme de Kevorkian:

Quand le graphe que nous tentons de réduire n'est pas complètement réduit après application des simplifications on se voit dans l'obligation d'appliquer le lemme suivant:

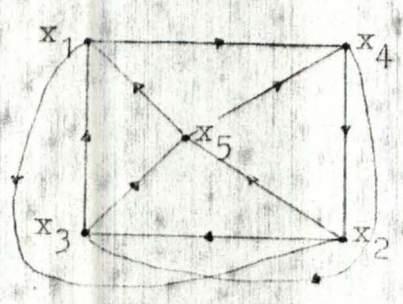
lemme 4

L'existence du doublet $\{x_p, x_q\}$ où $x_q \notin F$ implique que $x_p \notin F$

Ce qui signifie que l'on "éprouve " les deux possibilités 1) $x_p \in F$ et que l'on essaie 2) $x_q \in F$ d'appliquer de nouveau ,autant que possible; à chacune des possibilités les simplifica--tions énoncées.On compare alors la cardina--lité des ensembles essentiels obtenus.
Il se peut que l' on doive répéter l'utili--sation du lemme 4.Il s'agit d'une procédure par séparation qui n'aura son équivalent , chez C.K,qu 'en toute dernière extremité (etape 4).

Remarque

Nous n'avons pu répondre à la question de savoir comment l' algorithme de Kevorkian permet de simpli--fier le graphe fortement connexe suivant:




Nous suggérons que la solution non proposée par l' algorithme est de générer un circuit du graphe et de procèder par séparation en supposant successivement que chaque sommet du circuit appartient à F.

II.4.3 Commentaires comparatifs des deux algorithmes

Les lemmes 1,2,3 de Kevorkian ,après correction du lemme 3, ont leur équivalent dans C.K.
Le théorème 1 et le corollaire 1 de Kevorkian ont leur équivalent dans C.K mais sont appliqués "plus tot" chez Kevorkian:la réduction est un peu moins progressive (appli--quer,pratiquement,l'algorithme,pourrait nous faire préférer l'approche de C.K. qui nécessite,à un moment donné,moins d' analyse(elle en nécessite deux plus faciles successives))
Les corollaires 2 et 3 de Kevorkian apportent des simplifi--cations absentes de C.K. R3 de C.K n'a pas d'équivalent chez Kevorkian.

Remarque

On pourrait envisager des simplifications plus éla--borées du type corollaires 2 et 3,à appliquer aux graphes dans le cadre de l'etape 1 de C.K et de l' algorithme de Kevorkian.Cependant,ces simplifications très élaborées ne s'appliqueraient que trop rarement aux graphes.

II.4.4 Choix d'un algorithme

NOUS CHOISISSONS L'APPROCHE DE C.K A LAQUELLE NOUS INTEGRONS LES COROLLAIRES 2 ET 3 DE KEVORKIAN.

-étape 1 de C.K.

Cette étape diffère peu de l'algorithme deKevorkian. Rajouter les simplifications apportées par les corollai--res 2 et 3 de Kevorkian complète un premier jeu de simplifications du graphe.

-Une procédure de séparation préconisée par Kevorkian nécessite souvent un temps de calcul et une capacité de stockage des résultats intermédiaires élevés.
Nous préfèrons retarder l'application de cette procédure et donc favoriser les étapes 2 et 3 de C.K .Ces deux étapes paraissent "réalisables" en terme de temps de

calcul et de stockage des résultats intermédiaires et devraient permettre de simplifier notablement le problème (on envisage que les circuits pertinents dans la recher--che d'ensemble minimum essentiel)

-L'étape 4 de C.K présentera les inconvenients d'une procédure par séparation.Il est à espérer que,les étapes 2 et 3 y compris,la méthode est plus avantageuse qu'une procédure de séparation appliquée directement à la fin de l'étape 1.

Nous avons donc au moins un algorithme qui permet de trouver un ensemble minimum essentiel d'un graphe.Cet algorithme se présente comme suit:

.Etape 1

Réduction préliminaire du graphe

moyen ? R1 à R5 ainsi que les corollaires 2 et 3 de Kevorkian.

Graphe totalement réduit ?

oui:allez en étape 5

non:allez en étape 2

.Etape 2

On considère le graphe réduit et on engendre les circuits pertinents.

.Etape 3

On construit une table de couverture et on réduit cette table.

Table totalement réduite ?

oui:allez en étape 5

non:allez en étape 4

.Etape 4

On utilise une procédure par séparation en vue de déterminer un ensemble minimum essentiel.

.Etape 5

On aura ,aux étapes 1, 3, 4 ,découvert les sommets d'un même ensemble minimum essentiel. Tous les circuits sont brisés. Le but est atteint.

.Stop

## II.5 Les circuits pertinents.

Dans ce cas, "pertinent" prend le sens "devant être considéré pour casser tous les circuits" . Nous avons vu que notre but était de casser tous les circuits existant dans le graphe. Mais au lieu de prendre en considération tous les circuits,nous allons seulement tenir compte des circuits dominants car si nous détrui--sons celui-là ,nous sommes certains de casser aussi ceux qu'ils dominent.

### II.5.1 Régles appliquées

Nous nous situons au sommet $x_i$.

#### II.5.1.1 Régle 6

S'IL EXISTE UN DOUBLET SIMPLE ENTRE y ET x(CAD QUE L'ADRESSE DE CHAQUE ARC LE CONSTITUANT CONTIENT SEU--LEMENT DEUX SOMMETS )

ALORS .ON SUPPRIME TOUS LES ARCS ENTRE y ET $x_i$
.ON INDIQUE L'ADRESSE $\{y,x_i\}$ DANS LA TABLE DE COUVERTURE.

Ayant un doublet entre y et $x_i$,nous sommes surs d'avoir un circuit.De plus,s'il existe d'autres circuits passant par un arc $(y,x_i)$ ou $(x_i,y)$,nous les détruirons en enlevant soit $x_i$,soit y.L'adres--se $\{y,x_i\}$ est mise dans la table de couverture car nous avons au moins un circuit et donc un des sommets au moins appartiendra à l'ensemble mini--mal essentiel.

Au départ ,nous avons donc la situation figure 3

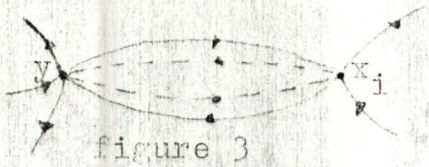



figure 3

Nous avons un doublet simple entre y et $x_i$, on supprime tous les arcs entre y et $x_i$ pour alors obtenir la figure 4:

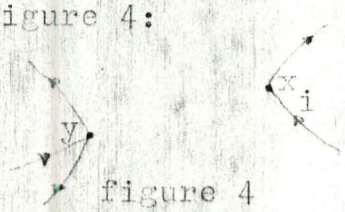



figure 4

→ $\{y, x_i\}$ est mis dans la table de couverture.

II.5.1.2 Régle 7

SI DANS $E_i$, IL EXISTE UN ARC $(x_i, y)$ (CAD ARC LE PLUS COURT ALLANT DE $x_i$ A y) AVEC UNE ADRESSE $\{x_i, y\}$ ALORS ON SUPPRIME TOUS LES ARCS $(x_i, y)$ SAUF $(x_i, y)$ QUI EST LE PLUS COURT CHEMIN DE $x_i$ A y.

Considérons la situation figure 5.

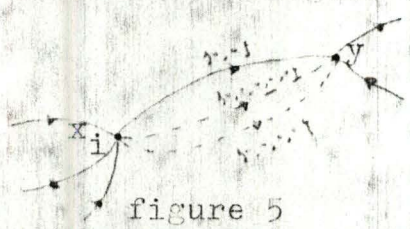



figure 5

Nous allons pouvoir supprimer les deux arcs "--" entre $x_i$ et y puisque si par les deux arcs passe un circuit, dés que l'on cassera le circuit passant par l'arc "——", nous détruirons en même temps les premiers. C'est le principe du circuit dominant.

Donc aprés l'élimination des deux arcs, nous aurons la situation donnée par la figure 6.

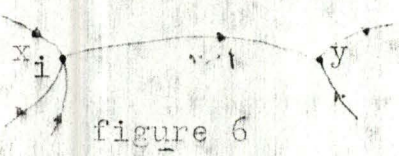



figure 6

Voyons que si nous ne considérons que le circuit dominant,cela est bien satisfaisant:


figure 7

Nous pouvons constater que si nous supprimons un sommet du circuit le plus court,nous cassons au--tomatiquement les deux circuits.

II.5.1.3 Régle 8

S'IL EXISTE $(x_i,y)$ ET $(x_i,y) \in \tilde{E}_i$ TEL QUE L'ADRESSE DE $(x_i,y)$ EST INCLUE DANS L'ADRESSE DE $(x_i,y)$ ,NOUS SUPPRIMONS $(x_i,y)$.
S'IL EXISTE $(y,x_i)$ ET $(y,x_i) \in \tilde{E}_i$ TEL QUE L'ADRESSE DE $(y,x_i)$ EST INCLUE DANS L'ADRESSE DE $(y,x_i)$,NOUS SUPPRIMONS $(y,x_i)$.

Soit donné la situation suivante représentée par la figure 8.

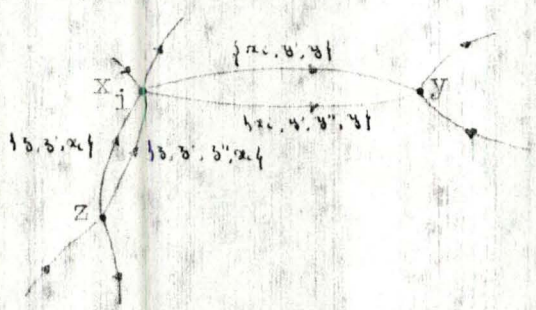



figure 8

Nous avons bien $\{x_i,y',y\} \subset \{x_i,y',y'',y\}$ .Par con--séquent,s'il y passe un circuit,il sera dominant par rapport à celui qui passera par l'arc ayant $\{x_i,y',y'',y\}$ comme adresse.
Donc nous pouvons supprimer l'arc ayant $\{x_i,y',y'',y\}$ comme adresse.Nous avons la même chose

pour les arcs allant de z à $x_i$.Le graphe résultant sera donné par la figure 9:

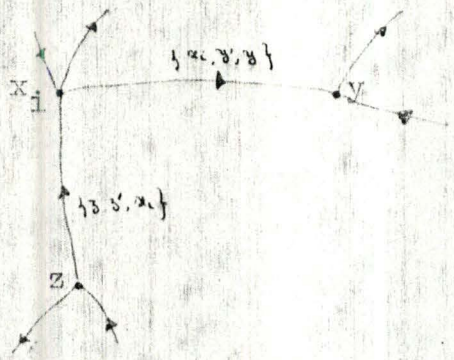



figure 9

II.5.1.4 Réle 9

SI DANS $\tilde{E}_i$,IL EXISTE $(x_i,y)$ AVEC $\{x_i,y\}$ COMME ADRES-
-SE

ALORS: .ON SUPPRIME TOUS LES ARCS $(y,x_i)$

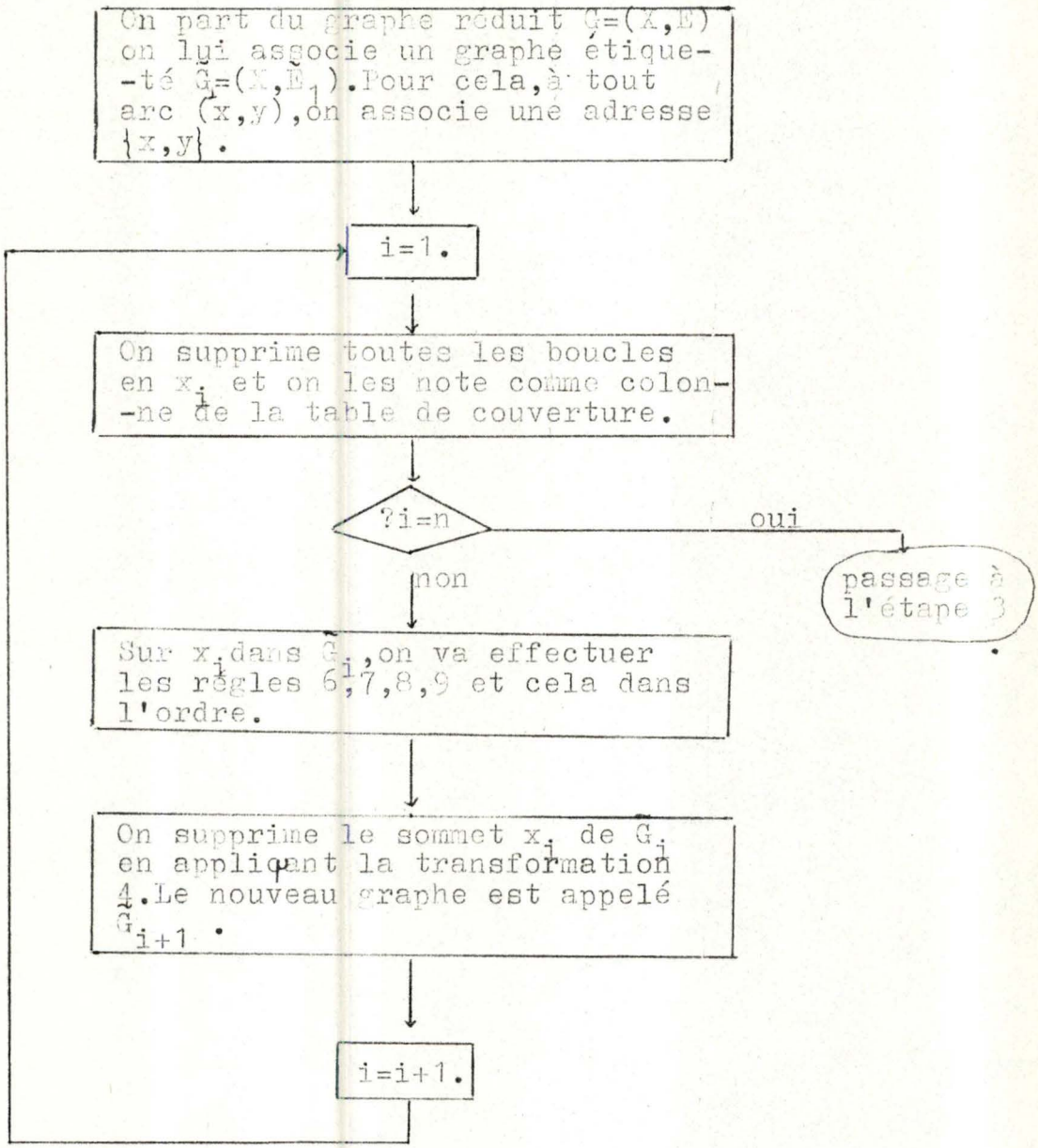
.ON OBTIENT PAR LES ARCS $(y,x_i)$ DES CIRCUITS ALLANT DE $x_i$ A $x_i$ .ON NOTE TOUTES LES ADRES-
-SES DE $(y,x_i)$ COMME COLONNE DE LA TABLE DE COUVERTURE.

Nous avons la situation de la figure 10.


figure 10

Nous sommes en présence de circuits allant de $x_i$ à $x_i$.Etant donné que ces circuits doivent tous être cassés,nous les repectorions tous dans la table de couverture.

## II.5.2 Algorithme des circuits pertinents




## II.5.3 Exemple

Dans le but d'étudier plus pratiquement ce qu'il se passe, nous allons illustrer cet algorithme au moyen d'un exemple.

Considérons le graphe G donné à la figure 11a. On lui associe son graphe étiqueté (voir figure 11b ).

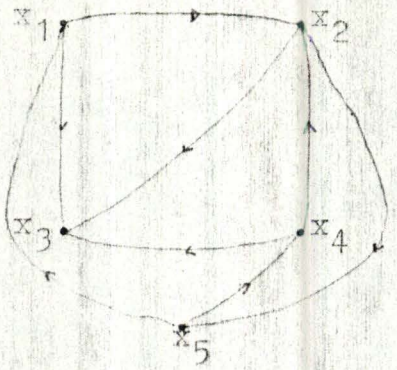



figure 11a

Alors $\tilde{G}$ =

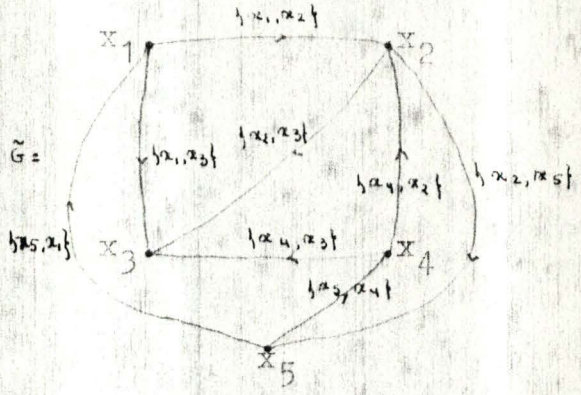


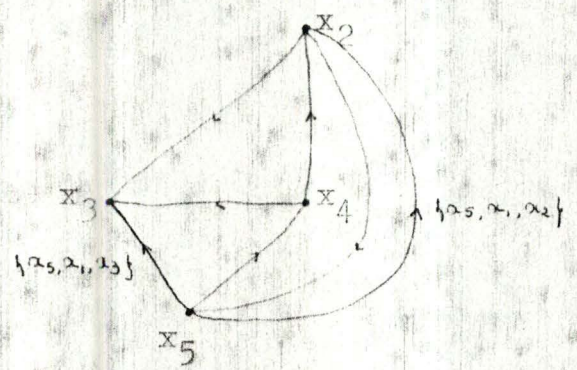

figure 11b

Dans la suite, les arcs qui n'auront pas d'adresse seront des arcs simples.

.On se situe en $x_1$.

Il n'y a pas de boucles et il n'y a pas moyen d'appliquer les régles R6-R9.

Nous supprimons le sommet $x_1$ par T4 et obtenons:




.On se place en $x_2$.

Nous pouvons appliquer la régle R9 puisque $(x_2, x_5)$ est un arc simple. Nous supprimons tous les arcs $(x_5, x_2)$.

<u>Par conséquent, $\{x_5, x_1, x_2\}$ sera la première colonne de la table de couverture.</u>

Après cette transformation, le graphe est représenté par la figure 13.

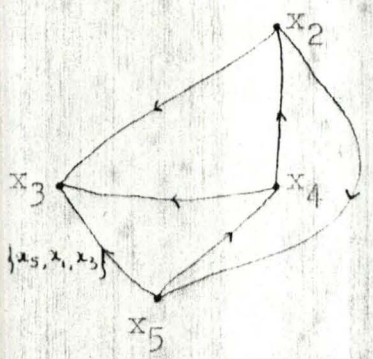



figure 13

Par T4, nous supprimons le sommet $x_2$ et obtenons le graphe de la figure 14.

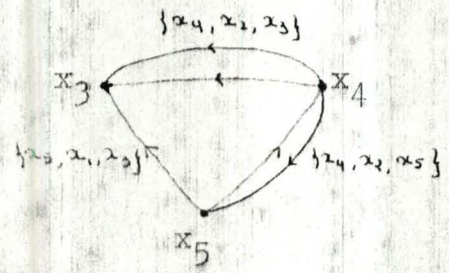



figure 14

.Considérons le sommet $x_3$.
Nous appliquons la régle R8 car $\{x_4, x_3\} \subset \{x_4, x_2, x_3\}$.

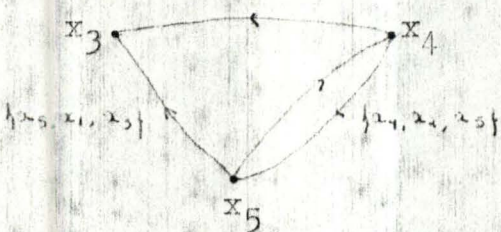



figure 15

Nous supprimons le sommet $x_3$ par T4.(figure 16)

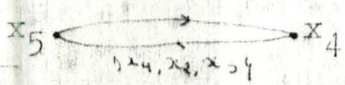



figure 16

.Considérons le sommet $x_4$.
Aucune régle n'est applicable. Quand on supprime $x_4$, on obtient une boucle en $x_5$.

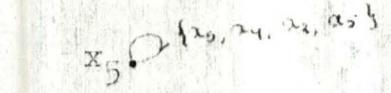


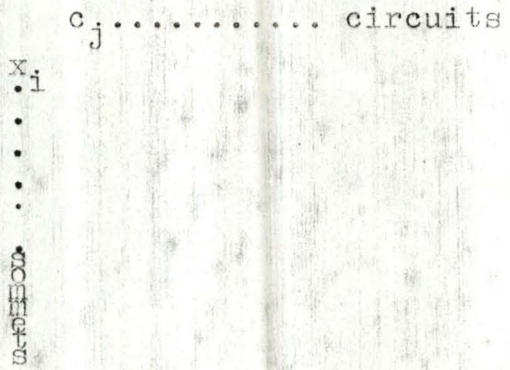

figure 17

Par conséquent, $\{x_5, x_4, x_2, x_5\}$ sera la deuxiéme colonne de la table de couverture.
La table de couverture est alors donnée par:

| | $n_1$ | $n_2$ | ..... |
|---|---|---|---|
| $x_1$ | x | | |
| $x_2$ | x | x | |
| $x_3$ | | | |
| $x_4$ | | x | |
| $x_5$ | x | X | |

## II.6 Réduction de la table de couverture.

La table de couverture se présente comme en figure 1.




Un élèment de la table $T_{ij}$ sera non nul lorsque le sommet $x_i$ apparaitra dans le circuit $c_j$.

Rappelons avant d'expliciter la réduction de la table de couverture, que l'ensemble essentiel minimum a comme effet de casser tous les circuits existants dans le graphe. De ce fait, il faut

que dans l'ensemble essentiel, nous retrouvons au moins un élément de chaque circuit.

## II.6.1 Définitions

### II.6.1.1 Notion de ligne essentielle

Si $x_i$ a une boucle, $x_i$ va devoir appartenir à chaque ensemble minimum essentiel puisqu'on ne peut casser cette boucle qu'en considérant le sommet $x_j$. (Dans la table de couverture, cette situation est représentée par le fait que dans la colonne nous n'avons qu'un seul élément.) Nous mettons $x_i$ dans l'ensemble minimum essentiel et nous supprimons tous les circuits contenant $x_i$ puisque nous venons de les casser.
La régle peut alors s'énoncer:

POUR TOUTE COLONNE DE LA TABLE DE COUVERTURE NE CONTENANT QU'UNE SEULE CROIX, NOUS ALLONS:

1. METTRE L'ELEMENT CORRESPONDANT A LA LIGNE MARQUEE DE LA CROIX DANS L'ENSEMBLE MINIMUM ESSENTIEL.
2. SUPPRIMER TOUS LES CIRCUITS CONTENANT CET ELEMENT CAD ON SUPPRIME LES COLONNES POUR LESQUELLES CET ELEMENT EST $\neq$ DE 0.
3. SUPPRIMER LA LIGNE.

### II.6.1.2 Notion de la colonne dominante

(=circuit dominant )

Si nous avons une colonne j inclue dans la colonne k, nous supprimons la colonne k car dés que nous considé-rons un sommet du circuit j, nous tiendrons en même temps compte du circuit k.
Nous avons la régle:

SI $\forall$ i L'ELEMENT $T_{ij} \neq 0$ ALORS L'ELEMENT $T_{ik} \neq 0$ ALORS NOUS POUVONS SUPPRIMER LES ELEMENTS DE LA COLONNE k.

II.6.1.3 Notion de la ligne dominante.

Si la ligne i est incluse dans la ligne j alors la li-
-gne j est appelée "dominante par rapport à la ligne i".
Nous supprimons alors la ligne i.De cette façon nous
sommes certains que lorsque nous aurons considérés les
circuits contenant $x_j$,nous aurons pris automatiquement
en compte les circuits contenant $x_i$.
La régle s'énonce alors:

SI SUR TOUTES LES COLONNES k, L ELEMENT $T_{ik} \neq 0$ ENTRAI-
-NE QUE $T_{jk}$ L EST AUSSI,NOUS SUPPRIMONS LA LIGNE i.

Ces régles sont appliquées dans n'importe quel ordre.

II.6.2 Test d'arrêt

Dans le paragraphe précédent,nous avons vu de quelle ma-
-ière nous réduisons la table de couverture.Une dernière
question va se poser,c'est:

"QUAND VA-T-ON S ARRETER "

A cette question ,nous allons pouvoir répondre de deux
façons:

1)La table de couverture a été complètement réduite
(cad que progressivement par applications successi-
-ves des régles de réduction,toutes les lignes et
toutes les colonnes ont été supprimées).Et nous
avons obtenu l'ensemble minimum essentiel que nous
recherchions .Il est constitué lors de l'applica-
-tion de la régle de la ligne essentielle.D'un au-
-tre coté,nous sommes surs de l'avoir déterminer
puisque nous avons casser tous les circuits.

2)La table de couverture n'est pas complètement ré-
-duite.Nous allons dans le paragraphe suivant ex-
-pliciter les opérations qui vont nous permettre
de compléter l'ensemble essentiel minimum que nous
sommes occupés à construire.

## II.7 Le branchement colonne.

Lorsqu'à la fin de l'étape 3,nous arrivons avec une table de couverture non complètement réduite,nous pouvons résoudre le problème de trois façons différentes.

### II.7.1.1 Methode de programmation linéaire en nombre entier

Voir Garfinkel et Nemhauser page 79.

### II.7.1.2 Méthode du branchement colonne

Nous considérons arbitrairement une colonne,soit la colonne j.En cette colonne,nous effectuons un branchement en un élèment non nul cad que nous supposons que le premier élèment non nul de la colonne appartient à l'ensemble minimum essentiel.Nous pouvons alors supprimer tous les circuits comprenant cet élèment puisque nous venons de les casser ainsi que la ligne.Nous effectuons alors la réduction de la table de couverture restante au moyen des régles vues au paragraphe précédent.Nous pouvons nous trouver en face de deux situations:

1)La table est cette fois complètement réduite et on stoppe.L'ensemble essentiel minimum est constitué lors de l'application de la régle 6 lors de la première réduction ainsi que par l'élément où on a branché et des nouveaux sommets détectés lors de la deuxième réduction.

2)La table n'est pas complètement réduite.Cela signifie que le sommet que l'on a supposé être dans l'ensemble essentiel minimum n'y est pas réellement. Nous considérons la table de couverture du début de cette phase et nous branchons àl'élément non nul suivant.En travaillant de cette manière ,nous sommes surs qu'à la fin de la colonne ,nous aurons terminé.En effet,comme de chaque circuit,il y a un élément dans l'ensemble minimum essentiel,un branchement conduira à une réduction totale.

II.7.1.3 Méthode booléenne

Nous savons que pour chaque ensemble essentiel R d'un graphe G,R contient au moins un élément de chaque circuit. Pour chaque circuit $c_j$,on peut écrire:

$$c_j=\{ x_{j1},x_{j2},.....,x_{jm_j} \}$$

Pour chaque sommet $x_{jk}$,on définit une fonction:

$\varsigma_{jk}=1$ si l'élément $x_{jk}$ à un ensemble essentiel

$=0$ si non

Dans un ensemble essentiel,nous avons au moins un sommet de chaque circuit.Cette condition soulignée peut être exprimée par la somme booléenne suivante:

$$(\varsigma_{j1}+\varsigma_{j2}+ .....)=1$$

$$\text{cad } \sum_{k=1}^{m_j} \varsigma_{jk}=1$$

Nous avons cette relation pour chaque circuit $c_j$.La solution doit satisfaire la condition booléenne:

$$(\varsigma_{11}+ \varsigma_{12}+ .....+\varsigma_{1m_1}).....(\varsigma_{j1}+ \varsigma_{j2}+ ...)=1$$

C'est un produit booléen de sommes booléennes

$$1= \prod_j \sum_{k=1}^{m_j} \varsigma_{jk}$$

Mais nous avons affaire à un produit de sommes booléennes. Nous aurons par conséquent comme résultat une somme de produits.Dans cette somme ,chaque terme réprésente un ensemble essentiel;pour obtenir l'ensemble essentiel minimum,il suffit de chercher le terme ayant le plus petit nombre d'éléments cad de lettres.Cette méthode n'est pas très difficile lorsqu'on travaille àla maïn mais dés qu'il s'agit d'un travail sur ordinateur,cela est plus ardu.

# CHAPITRE III. AU POINT DE VUE PRATIQUE.

## III.1 Ensemble essentiel minimum et ensemble de sortie

Dans le premier chapitre, nous avons vu que par l'intermédiaire d'un ensemble de sortie, nous pouvions associer à tout système de n équations à n inconnues un graphe de flux d'information. Nous allons voir sur un exemple que deux ensembles de sortie différents peuvent donner des ensembles minimaux essentiels de tailles différentes.

Considérons le système d'équations suivant:

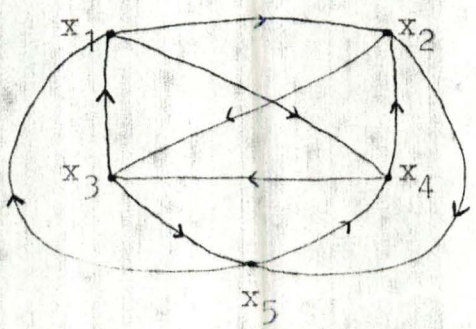
$$f_1(x_1,x_3,x_5)=0$$
$$f_2(x_1,x_2,x_4)=0$$
$$f_3(x_2,x_3,x_4)=0$$
$$f_4(x_1,x_4,x_5)=0$$
$$f_5(x_2,x_3,x_5)=0$$

Nous définissons alors deux ensembles de sortie:

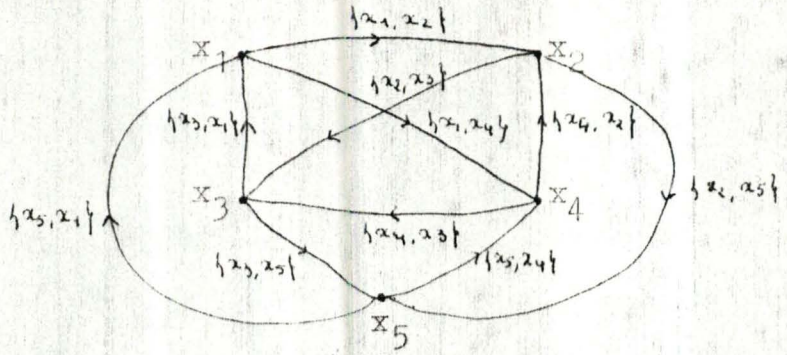
| 1) | 2) |
|---|---|
| $(x_1,f_1)$ | $(x_5,f_1)$ |
| $(x_2,f_2)$ | $(x_4,f_2)$ |
| $(x_3,f_3)$ | $(x_2,f_3)$ |
| $(x_4,f_4)$ | $(x_1,f_4)$ |
| $(x_5,f_5)$ | $(x_3,f_5)$ |

Cas du premier ensemble de sortie

Le graphe associé est donné par:

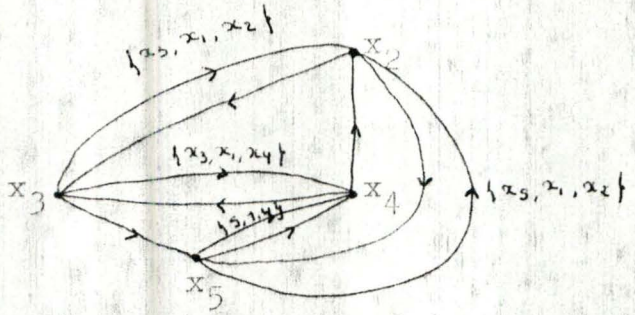


Pour rechercher l'ensemble minimum essentiel, nous allons appliquer l'algorithme des circuits pertinents de C et A. Nous définissons donc le graphe étiqueté:




Soit le sommet $x_1$.

Aucune des quatre régles n'est applicable. Nous appliquons T4 à $x_1$, nous obtenons:

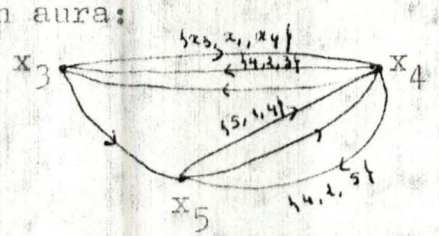



Soit le sommet $x_2$.

Les arcs $(x_2,x_3)$ et $(x_2,x_5)$ ont des étiquettes simples donc nous supprimons les arcs allant de $x_2$ à $x_3$ et de $x_2$ à $x_5$ et nous plaçons leurs adresses dans la table de couverture.

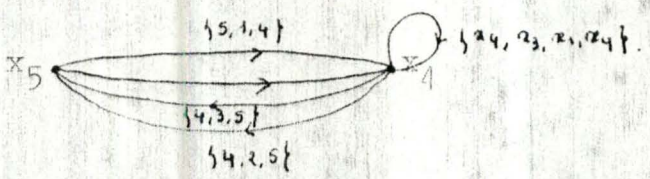
$\{x_3,x_1,x_2\}$ et $\{x_5,x_1,x_2\}$

Après T4 , on aura:



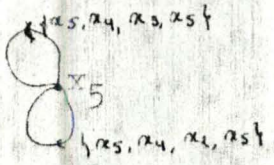
Soit le sommet $x_3$.

Nous pouvons appliquer R8 à l'arc $(x_4, x_3)$. Nous supprimons $x_3$ par T4:




Soit le sommet $x_4$.

Il y a une boucle donc nous mettons $\{x_4, x_3, x_1, x_4\}$ dans la table. Nous appliquons R8 et T4




Soit le sommet $x_5$.

$\{x_5, x_4, x_3, x_5\}$ et $\{x_5, x_4, x_2, x_5\}$ sont mis dans la table.

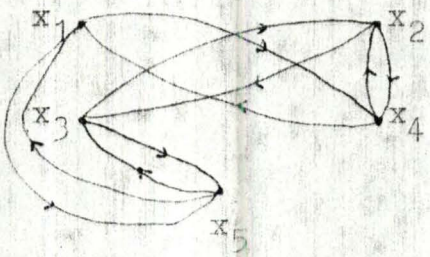
La table de couverture se présente sous la forme:

$$\begin{bmatrix} x & x & x & & \\ x & x & & & x \\ x & & x & x & \\ & & x & x & x \\ & x & & x & x \end{bmatrix}$$

Après réduction, nous pouvons constater que l'ensemble minimum essentiel est de taille 2.

Cas du second ensemble de sortie

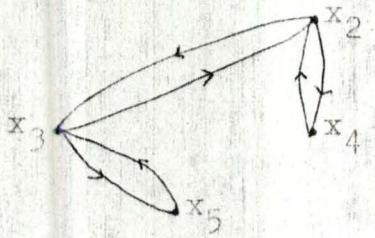
Le graphe associé est donné par:




A chaque arc, nous associons une adresse.

Soit le sommet $x_1$.

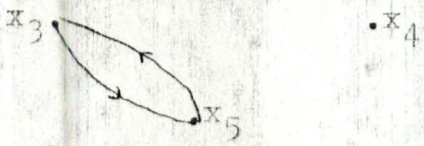
Nous pouvons appliquer R6. Après T4, on obtient:




$\{x_4, x_1\}$ et $\{x_5, x_1\}$ sont mises dans la table.

Soit le sommet $x_2$.

On applique R6 et T4:




$\{x_3, x_2\}$ et $\{x_4, x_2\}$ sont mises dans la table.

Soit le sommet $x_3$.

$\{x_5, x_3\}$ va dans la table.

Il ne reste plus d'arcs, on s'arrête.

La table de couverture est :

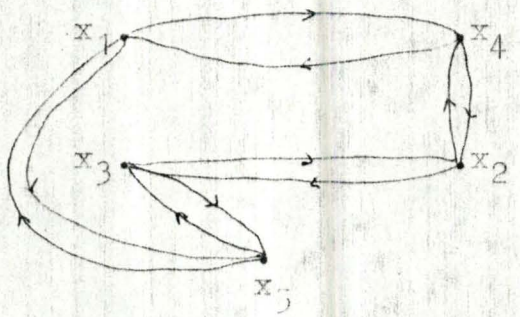
$$\begin{bmatrix} x & x & & & \\ & & x & x & \\ & & x & & x \\ x & & & x & \\ & x & & & x \end{bmatrix}$$

Après réduction, nous constatons que l'ensemble minimum essentiel est de taille 3.

Nous avons donc montrer sur cet exemple que l'ensemble minimum essentiel n'avait pas la même taille dans les deux cas.

## III.2 Exemple de branchement colonne

Lors de la recherche de l'ensemble minimum essentiel,nous avons vu que il y avait la construction d'une table de couverture. Dans l'étape 3 de Cheung et Kuh ,nous effectuons la réduction de cette table de couverture;dans ce paragraphe ,nous allons examiner ce qu'il se passe sur un exemple.Pour ne pas refaire de nouveau toute la recherche des circuits pertinents,nous ne donnerons uniquement le graphe de départ et la table de couverture définie à la fin de l'étape 2 de Cheung et Kuh.
Considérons le graphe représenté par la figure suivante:




Après l'application des régles R6,R7,R8,R9 et T4,la table de couverture de ce graphe peut s'écrire:

| | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ |
|---|---|---|---|---|---|
| $x_1$ | x | x | | | |
| $x_2$ | | | x | x | |
| $x_3$ | | | x | | x |
| $x_4$ | x | | | x | |
| $x_5$ | | x | | | x |

### Réduction de la table

Nous pouvons remarquer que aucune régle de réduction ne peut être appliquée dans ce cas-ci;nous allons par conséquent devoir employer la méthode du bran--chement colonne.

Branchement colonne

Nous considérons la deuxième colonne de la table de couverture et dans cette colonne ,nous branchons au premier élément non nul cad en $x_1$.Cela signifie que nous supposons que cet élément appartient à l'ensem--ble minimum essentiel.Par définition de cet ensem--ble ,nous supprimons les circuits contenant cet élément ainsi que la première ligne de la table.
On obtient par conséquent:

| | $n_3$ | $n_4$ | $n_5$ |
|---|---|---|---|
| $x_2$ | x | x | |
| $x_3$ | x | | x |
| $x_4$ | | x | |
| $x_5$ | | | x |

La trisième ligne est incluse dans la première,nous supprimons alors la troisième ligne.Suite à cette transformation,la seconde colonne ne contient qu'un seul élément;nous avons une boucle en $x_2$.Nous plaçons $x_2$ dans l'ensemble essentiel minimum et supprimons tous les circuits contenant $x_2$ ainsi que la première ligne.Il reste alors la colonne correspondant au circuit numéro 5.Nous pouvons mettre soit $x_3$ soit $x_5$ dans l'ensemble essentiel minimum.
De cette manière,la table a été complètement réduite et l'ensemble minimum essentiel est donne par $\{x_1,x_2,x_3\}$ ou $\{x_1,x_2,x_5\}$.

## III.3 La table de couverture.

### III.3.1 Représentation de la table

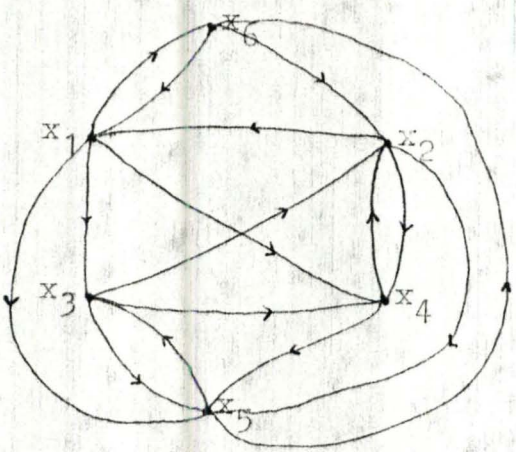
Nous représentons la table de couverture sous la forme d'un tableau ayant comme indice de lignes les différents sommets du graphe et comme indice de colonne les circuits Nous avons employé une table booléenne .L'emploi de "lists

enchainés" a été écarté par le fait que la table de cou-
-verture n'est pas creuse (creuse = contenant un nombre important d'éléments égaux à zéro).

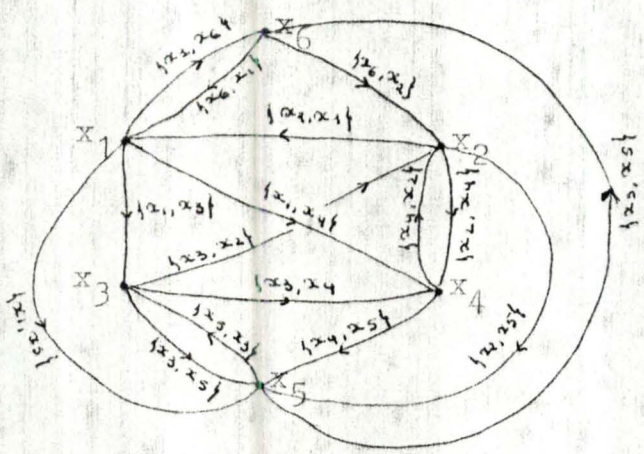
III.3.2 Inclusion de différentes colonnes.

Dans le chapitre II, nous avons vu que les régles 7 et 8 nous permettaient de supprimer des circuits dominés. Dans ce paragraphe, nous allons montrer à travers un exemple que la table de couverture définie à la fin de l'étape 2 de Cheung et Kuh peut encore contenir des colonnes inclu-
-ses.

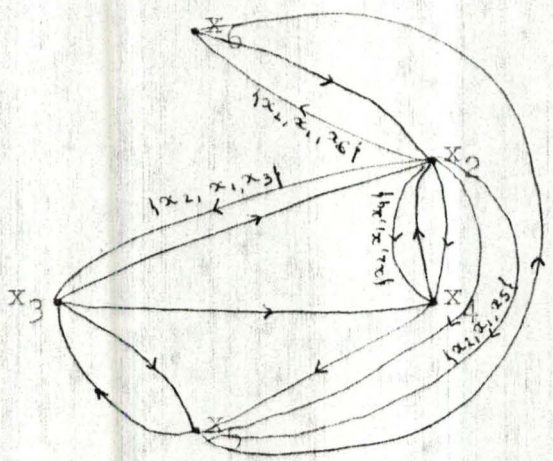
Soit le graphe suivant:



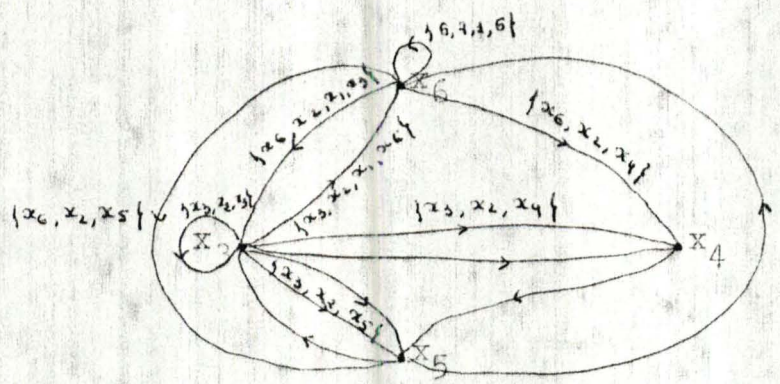

Son graph. étiqueté associé est donné par:



Nous nous plaçons au smmet $x_1$.Il existe un doublet simple entre $x_1$ et $x_6$.Nous supprimons l'arc $(x_6,x_1)$ et écrivons son adresse $x_6,x_1$ dans la table de couverture.Par T4,nous éliminons le sommet $x_1$:

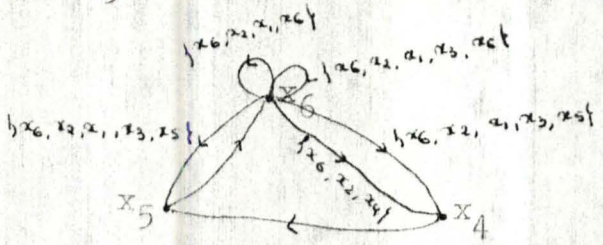



Nous considérons le sommet $x_2$.Il existe un doublet simple entre $x_2$ et $x_4$.Nous supprimons tous les arcs allant de $x_4$ à $x_2$ et inscrivons $\{x_4,x_2\}$ dans la table L'adresse $\{x_2,x_5\}$ est incluse dans $\{x_2,x_1,x_5\}$ ,nous pouvons alors éliminer l'arc associée à cette seconde adresse.Appliquons la transformation T4:

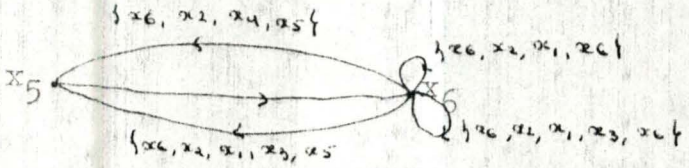



Nous nous plaçons au sommet $x_3$.Il existe une boucle en $x_3$;nous la supprimons et inscrivons son adresse

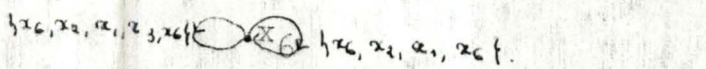
dans la table de couverture. Nous avons un doublet simple entre $x_3$ et $x_5$. Nous supprimons l'arc $(x_5, x_3)$ et indiquons l'adresse dans la table. Par la régle R7, nous pouvons éliminer les arcs d'adresse $x_3, x_2, x_4$ et $x_3, x_2, x_5$. Le graphe résultant est:




En $x_4$, aucune régle n'est applicable. Nous éliminons $x_4$ par T4:




Nous nous plaçons au sommet $x_5$. Par l'application de la régle R9, nous écrivons $x_6, x_2, x_1, x_3, x_5$ ainsi que $x_6, x_2, x_4, x_5$ dans la table de couverture.




En $x_6$, nous avons deux boucles dont nous inscrivons les adresses dans la table.
La table de couverture s'ecrit:

| | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | $n_6$ | $n_7$ | $n_8$ |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | x | | x | | x | | x | x |
| $x_2$ | | x | x | | x | x | x | x |
| $x_3$ | | | x | x | x | | | x |
| $x_4$ | | x | | | | x | | |
| $x_5$ | | | | x | x | x | | |
| $x_6$ | x | | | | x | x | x | x |

Nous constatons que le circuit $n_1$ est inclus dans les circuits $n_5, n_7, n_8$ et que le circuit $n_2$ est inclus dans le circuit $n_6$.

## III.4 Remarques

1) Emploi des "lists enchainés"

Pour la représentation du graphe, nous employons les "lists enchainés ". Cet emploi est justifié par le fait que la matri-ce à laquelle nous avons affaire est bien une matrice creu-se (définition page III.3.2).

2) Problème de dimension

Nous avons deux problèmes de dimension:

1. Au sujet de la dimension de la table de couvertu-re, nous ne pouvons absolument rien en dire avant d'avoir complètement terminer l'étape 2 de l'algo-rithme de Cheung et Kuh. Le dimensionnement doit alors se faire sans aucune certitude.
2. Lors de l'exécution de l'étape 2 , nous avons vu qu'à chaque arc, nous avions associé une adresse. Le problème est dû au fait qu'en cours d'étape, nous ajoutons et éliminons des arcs mais que nous ne connaissons pas leur nombre.

# TECHNIQUES DE DECOMPOSITION DE SYSTEMES D'EQUATIONS ("N X N") A GRANDE ECHELLE PAR LA THEORIE DES GRAPHES

## TOME II

Gertrude HAUSTRATE

et

Philippe DONTAINE

# CHAPITRE IV : Réalisation d'un programme pour ordinateurs.

( langage utilisé : FORTRAN IV )

Nous avons présenté dans les chapitres I et II un algorithme permettant :

- de décomposer en r sous-système $S_i$ irréductibles un système à grande échelle S
- de découvrir un ordre de résolution des sous-systèmes irréductibles $S_i$
- de décomposer plus finement chaque sous-système irréductible dans le but de tirer parti de la structure du sous-système en vue d'une résolution de celui-ci.

Cet algorithme est-il "réalisable" ?

Pour tenter de répondre à cette question, nous posons et résolvons partiellement le problème de la programmation d'un tel algorithme.

Quelles parties de l'algorithme sont-elles réalisées ?

- La recherche des sous-systèmes irréductibles et la découverte d'un ordre de résolution des sous-systèmes ( décomposition en niveaux hiérarchiques ). (cfr. chapitre I)
- Les étapes 1 et 2 de l'algorithme de C.K. dans la recherche des ensembles minimaux essentiels d'un graphe.

Ces deux parties fontionnent effectivement

- L'analyse de l'étape 3 de l'algorithme de C.K. est effectuée.

Deux parties importantes de tout l'algorithme manquent à ce programme

1. la recherche des paires de sortie d'un système d'équations.
   L'analyse de cette partie doit comporter la recherche d'un algorithme efficace qui effectue cette tâche ( se référer à Steward-1 ).
   Nous avons momentanément contourné ce problème en fournissant comme donnée du programme le graphe de flux d'information ( que l'on aurait construit à l'aide de l'ensemble des paires de sortie)
2. l'étape 4 de l'algorithme de C.K. Cette étape est (cfr. chapitre II ) une procédure de séparation du problème; elle pose un

intéressant problème de réservation de place en mémoire centrale de l'ordinateur.

La nécessité de limiter ce travail par manque de temps nous à amené à laisser ces deux problème de côté.

Une partie utile peut être ajoutée à cet ensemble : un générateur aléatoire de 1-graphes orientés qui permettrait de tester l'efficacité à espérer de cet algorithme appliqué à des graphes de flux d'information "quelconques".( cette façon de tester l'algorithme ne tient pas compte de la recherche de paires de sortie de systèmes d'équations, dont le "coût" est négligeable par rapport à celui du reste du travail ( voir Kev.II P.510 et Led.et Himm. P224 )

---

## IV.A. Description de la représentation de l'information. Difficultés, avantages et inconvénients que procurent ces représentations.

---

### IV.A.1. Introduction

Le but proposé était de construire un algorithme "réalisable" de décomposition de systèmes à grande échelle.
Les résultats d'une programmation d'algorithmes de décomposition cités dans la littérature( Kev.II P.510 et Led. et Himm.P.225) suggèrent que la place occupée en mémoire centrale de l'ordinateur par les variables du problème est une fonction quadratique $(n^2 + an + b)$ de la taille n du système à décomposer .
Cette exigence tient à l'introduction en mémoire de la matrice (nXn) C d'occurence du système, sous sa forme"complète", c.à.d. tous les 1 et 0 booléens de la matrice C sont stockés. Il est établi (Led. et Himm.P.187), que pour qu'une décomposition de système à grande échelle aie des chances de fonctionner, il faut pratiquement qu'une "petite"partie des n variables du problème apparaisse dans les équations.
La matrice d'occurence du système d'équations est alors "creuse", c.à.d. elle comprend une "petite proportion"d'éléments non nuls.
Lors de manipulations (p.ex. inversions ) de matrices scalaires

creuses, il est conseillé (cfr. Tew.P.527) d'utiliser en ordinateur des représentations de matrices où on ne considère que les éléments non nuls .

Plutôt que de réaliser "rapidement" ( c.à.d. de façon non élaborée) un programme pouvant effectuer l'algorithme proposé dans les chapitres I et II , un problème intéressant était de

1. Voir dans quelles mesures on pouvait "condenser" la matrice d'occurence

2. Montrer si nous pouvions proposer un programme dont les exigences en place en mémoire centrale d'ordinateur seraient fontion linéaire de la taille n du système d'équations et du nombre e d'arcs du graphe de flux d'information.

3. Remarquer si nous pouvions, en "condensant" la matrice d'occurence, permettre à l'algorithme de garder ou d'améliorer son efficacité pratique ( améliorer par rapport à une façon "classique" d'envisager le problème ) .

Nous montrons (cfr. IV.A.2 ) :

a. qu'il est possible de concevoir pour la recherche de sous-systèmes irréductibles, pour la décomposition en niveaux hiérarchiques (ch.I ), et pour l'étape 1 de l'algorithme de C.K. (ch.II ) une rèprésentation de l'information qui demande, en mémoire de l'ordinateur une place fonction linéaire de la taille n du système à résoudre et du nombre d'arcs du graphe de flux.

b. que l'utilisation d'une forme condensée de la table de couverture (cfr.ch.II. étape 2-3-4 de l'algorithme C.K.)nous paraît discutable, ce qui pose des problème d'efficacité pratique, et rend certainement non linéaire en les paramètres n et e ( resp. nombre de sommets et d'arcs du graphe de flux) l'occupation de mémoire centrale.

c. que cette remarque se répète quand on aborde une méthode de "séparation" conformément à l'étape 4 de l'algorithme C.K.

Peut-être ces remarques suggèreraient-elles une délimitation "meilleure" des notions de système à grande échelle et de matrice d'occurence creuse d'un système d'équations.

## IV.A.2. Représentation des données ou de l'information

Les données du programme dans sa formule actuelle sont :

1. Un graphe de flux d'information
2. Le nombre de sommets n de ce graphe.

Dans tout ce qui suit, soit e le nombre d'arcs du graphe.

### IV.A.2.a. Première partie de l'algorithme

1. Trouver les sous-systèmes irréductibles
2. Construire le graphe condensé
3. Décomposer en niveaux hiérarchiques le graphe condensé
4. "Dé-condenser" - n'est nécessaire que pour la présentation des résultats. Nous n'effectuons pas cette décondensation.

Pour cette partie de l'algorithme, le graphe de flux d'information se représente à l'aide de 2 vecteurs :

1. Sommet : dimension n+1
2. Arcs : dimension e

A chaque sommet i, le vecteur Sommet fait correspondre un nombre qui est la place où, dans le vecteur Arcs, on peut trouver le premier suivant du sommet i.

Dans le vecteur Arcs, les suivants d'un sommet sont placés les uns à la suite des autres.

La liste des suivants du sommet i se trouve comme suit :

soit LIM1= Sommet ( i )
LIM2= Sommet ( i + 1 ) - 1 } ( 1 )

Les suivants du sommet i sont les variables Arcs ( j ) telles que $LIM1 \leqslant j \leqslant LIM2$

Convention qui facilite la programmation : Si un sommet i n'a pas de suivants, Sommet ( i ) = Sommet ( i + 1 )

Remarque :

La dimension du vecteur Sommet est n+1 et Sommet ( n + 1 ) = e + 1

En effet : - cela permet de ne pas différencier dans le traitement le dernier sommet ( les formules ( 1 ) exigent que Sommet ( i + 1 ) soit déterminé
- nous ne comptons pas les arcs du graphe de flux. Le

vecteur Arcs a une dimension f , où f est une borne supérieure du nombre d'arcs du graphe. Il faut connaître où se termine la liste des suivants du dernier sommet dans le vecteur Arcs.

Exemple : soit à représenter le graphe fig.1

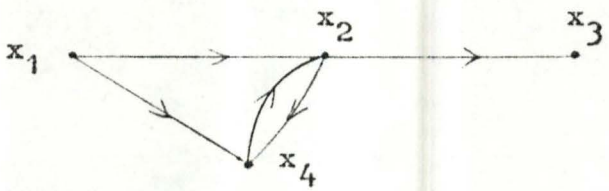



fig.1

Représentation : fig.2

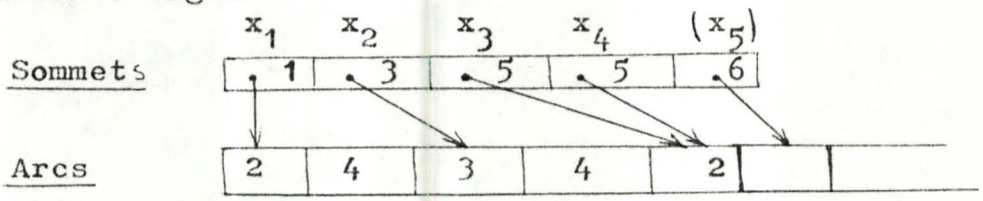



fig.2

Cette représentation est adéquate pour la réalisation de la première partie de l'algorithme.

1. Rechercher les composantes fortement connexes ( C.F.C. ) du graphe par l'algorithme de Tarjan (cfr.ch.I).
La technique d'exploration utilisée nécessite de traverser les arcs orientés exactement 1 fois ; traverser implique respecter le sens de l'arc. Trouver "rapidement" les suivants d'un sommet est nécessaire.

2. Connaître "rapidement" les suivants d'un sommet rend plus efficace la construction du graphe condensé. La représentation de celui-ci sera construite sous la même forme que celle du graphe de flux.

3. Cette forme compacte du graphe condensé s'adapte assez peu à la décomposition en niveaux qui nécessite la connaissance de sommets sans précédents alors que nous travaillons avec des listes de suivants. Cependant, construire des listes de précédents semble nécessiter un nombre d'opérations du même ordre que la décomposition en niveaux à l'aide de listes de suivants.
Cette décomposition en niveaux est de toute façon "peu couteuse"

au vu du reste du travail.
Remarque : a. Construire la représentation décrite ne pose pas de problèmes. Le graphe de flux est donné en indiquant pour chaque sommet ses suivants. La phase de construction du graphe de flux après la recherche d'un ensemble de paires de sortie, à ajouter à ce programme, peut tenir compte facilement de la représentation du graphe décrite.

b. Après avoir utilisé la représentation décrite, nous avons vérifié dans Rose et Will. P.44 qu'elle est reconnue comme adaptée aux problèmes de théorie des graphes où, essentiellement, la connaissance de suivants d'un sommet est nécessaire.

## IV.A.2.b. Deuxième partie de l'algorithme

Il s'agit de l'algorithme de C.K. ( voir chapitre II )- 4 étapes.
L'information utilisée pour effectuer cette partie de l'algorithme de décomposition est le sous-graphe de flux d'information associé à une C.F.C. du graphe de flux.
L'expérience nous a montré que la représentation de graphe décrite en IV.A.2.a. n'est pas adaptée à l'algorithme de C.K. même si on lui adjoint une représentation de graphe qui à chaque sommet associe la liste de ses précédents. Ce fait est causé par le besoin d'ajouter et de retirer fréquemment des arcs; de détruire des sommets; d'où il faut remettre les listes de précédents et de suivants en ordre, ce qui est couteux et difficile.
Pour ne pas avoir à réordonner les listes, l'emploi de la matrice au sous-graphe paraît inévitable.
Cependant, rappelons-nous ( IV.A.1 ) que nous ne pouvons traiter pratiquement que des matrices creuses.
Utiliser des listes enchainées ( Knuth. CH.2) permet de représenter ces matrices en ne tenant compte que des éléments non nuls et en facilitant l'insertion et la destruction d'éléments de la liste.
La manière de construire ces listes dépend de l'utilisation que l'on veut en faire.

IV.A.2.b.1. Représentation de matrice choisie.

Nous envisageons une C.F.C. $K_1$ du graphe de flux G et la matrice $M'^T$ transposée de la matrice associée au sous-graphe $K_1$ de G. Soit n' et e' le nombre de sommets et d'arcs de $K_1$ (de lignes et de "1" de $M'^T$ ).

Nous représentons $M'^T$ à l'aide de 6 vecteurs dont la dimension est e'.

Numérotons les arcs de $K_1$ de 1 à e'.
A chaque arc i, chaque vecteur fait correspondre respectivement :
1- ICOL : le numéro de colonne de l'arc i dans $M'^T$ (extrémité initiale de l'arc i )
2- LIGN : le numéro de ligne de l'arc i dans $M'^T$ .
3- ISUICO : le numéro de l'arc j qui se trouve juste sous l'arc i dans la colonne de l'arc i ( soit la colonne k ) dans la matrice $M'^T$ si l'arc en question existe .Voir fig.3.
Sinon le premier arc d'une colonne suivante (en commençant par le haut de la colonne), en envisageant successivement comme colonne suivante : a. les colonnes à droite de la colonne k en partant de la colonne k. Voir fig.4.
b. les colonnes à gauche de la colonne k en partant de la première colonne de $M'^T$ . Voir fig.5.
c. éventuellement en dernier lieu la colonne k . Voir fig.6.
Les figures indiquent comment s'effectue le "balayage" des arcs à partir de l'arc i ( X )pour la construction des vecteurs 3 et ensuite 4 , 5 et 6

fig.3

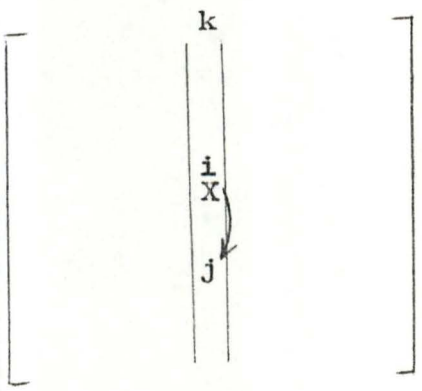



fig.4

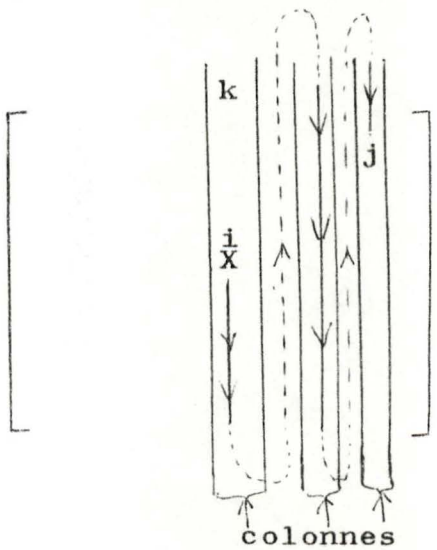


fig.5

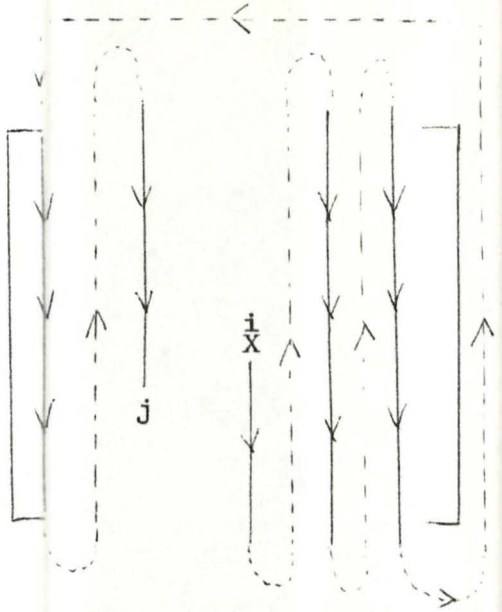
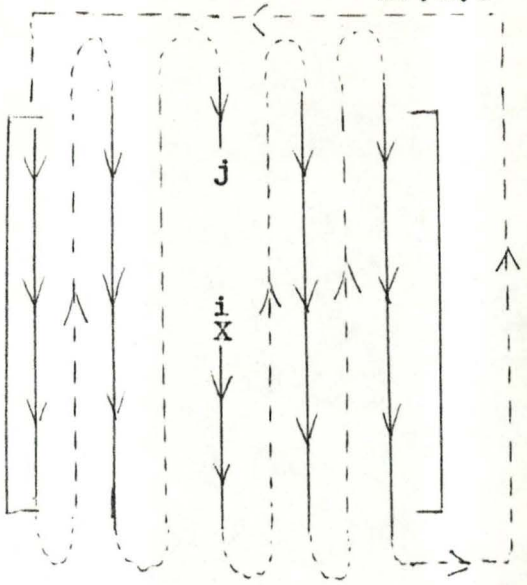



fig.6




4- IPRECO : le numéro de l'arc qui se trouve juste au dessus de l'arc i dans la colonne k de la matrice $M'^{T}$ si l'arc en question existe. Sinon, suivre un règle analogue à celle décrite en 3. Voir fig.7

5- ISUILI : le numéro de l'arc qui se trouve à droite de l'arc i dans la ligne de l'arc i ( soit la ligne l ) dans la matrice $M'^{T}$ si cet arc existe. Sinon suivre une règle analogue à celle décrite dans 3. Voir fig.8

fig.7

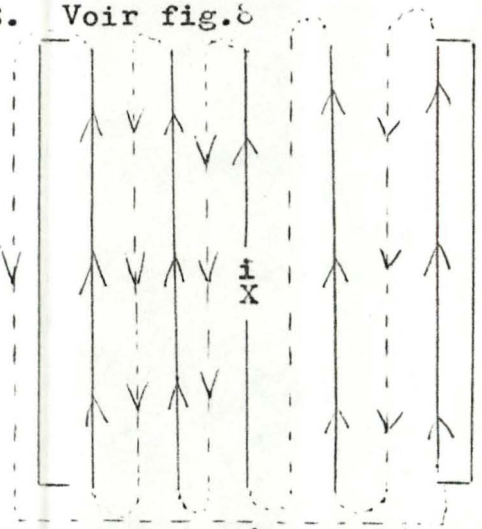
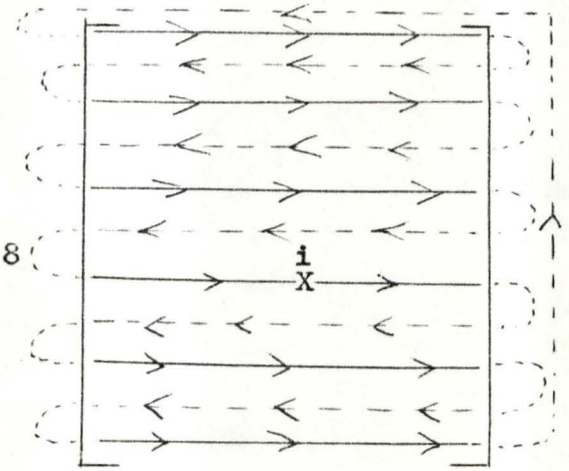


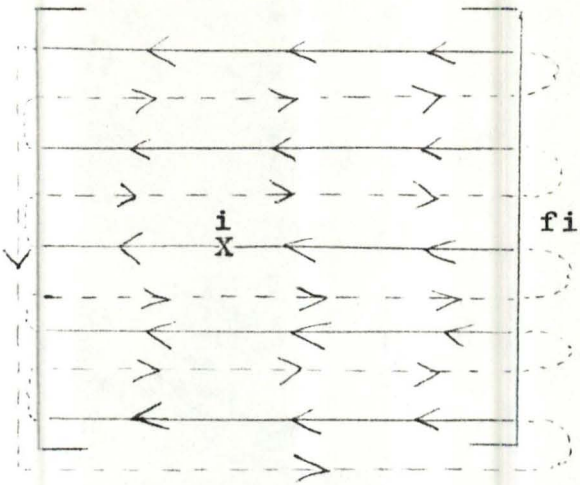

fig.8




6- IPRELI : le numéro de l'arc qui se trouve à gauche de l'arc i dans la ligne l de la matrice $M'^{T}$ si cet arc existe. Sinon suivre une règle analogue à celle décrite en 3. Voir fig.9

Les figures 10 et 11 résument la manière dont se font les balayages des arcs.

fig.9




balayage tout d'abord vert.
ensuite latéral.

fig.10

| vert. / lat. | ↑ | ↓ |
|---|---|---|
| ← | IPRECO | |
| → | | ISUICO |

balayage tout d'abord latéral
ensuite vertical

fig.11

| vert. / lat. | ↑ | ↓ |
|---|---|---|
| ← | IPRELI | |
| → | | ISUILI |

Nous ajoutons à cette représentation, 4 vecteurs donnant une information pour minimiser les opérations et localiser l'information utile. Ils indiquent respectivement :

7- NCLIG : le nombre d'éléments d'une ligne de $M'^{T}$ .

8- NOCOL : le nombre d'éléments d'une colonne de $M'^{T}$ .

9- INDLIG : le numéro du premier arc d'une ligne de $M'^{T}$ .

10- INDCOL : le numéro du premier arc d'une colonne de $M'^{T}$ .

Exemple : Soit à représenter le graphe de fig.12 de matrice $M'^{T}$ fig 13.

fig.12

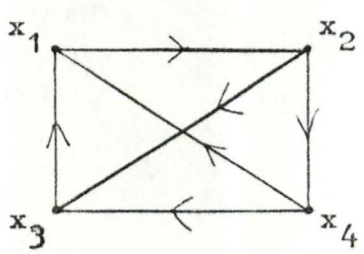



fig.13

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | | 1 | 1 |
| 2 | 1 | | | |
| 3 | | 1 | | 1 |
| 4 | | 1 | | |

Nous obtenons la représentation de la figure14 qui correspond à une numérotation des arcs de $M'^{T}$ - fig.15.

1- ICOL
2- LIGN
3- ISUICO(suiv.colonne)
4- IPRECO(préc.colonne)
5- ISUILI(suiv.ligne)
6- IPRELI(préc.ligne)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 4 | 4 | 0 |
| 2 | 3 | 4 | 1 | 1 | 3 | 0 |
| 2 | 3 | 4 | 5 | 6 | 1 | 0 |
| 6 | 1 | 2 | 3 | 4 | 5 | 0 |
| 2 | 6 | 4 | 5 | 1 | 3 | 0 |
| 5 | 1 | 6 | 3 | 4 | 2 | 0 |

fig.14.

7- NOLIG(nombre d'arcs ligne)
8- NOCOL(nombre d'arcs colonne)
9- INDLIG(début ligne)
10- INDCOL(début colonne)

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 2 | 1 | 2 | 1 |
| 1 | 2 | 1 | 2 |
| 4 | 1 | 2 | 3 |
| 1 | 2 | 4 | 5 |

fig.15

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | | | 4 | 5 |
| 2 | 1 | | | |
| 3 | | 2 | | 6 |
| 4 | | 3 | X | |

Ajouter un arc n° 7 en colonne 3 ligne 4 de $M'^{T}$ correspond à écrire :

- ajouter une colonne au tableau fig.14 ; lier l'arc 7 aux autres arcs du tableau ;

  1(7) ⟵ 3
  2(7) ⟵ 4
  3(7) ⟵ 5
  4(7) ⟵ 4
  5(7) ⟵ 4
  6(7) ⟵ 3

- ajuster les vecteurs de comptage d'arcs ; lier les arcs du tableau à l'arc 7

  7(4) ⟵ 7(4) + 1
  8(3) ⟵ 8(3) + 1
  3(4) ⟵ 7
  4(5) ⟵ 7
  5(3) ⟵ 7
  6(4) ⟵ 7

Détruire l'arc n° 2 du tableau $M'^{T}$ fig 15 correspond à écrire :

- détruire les liens de l'arc 2 avec les autres arcs

  3(1) ⟵ 3

4(3) ⟵ 1
5(1) ⟵ 6
6(6) ⟵ 1

- mettre à jour le nombre d'arcs en ligne et en colonne et le début de ligne et de colonne.

7(3) ⟵ 7(3) - 1
8(2) ⟵ 8(2) - 1
9(3) ⟵ 6
10(2) ⟵ 3

Cette représentation nécessite en mémoire une place proportionnelle à 6e' + 4n' ( parce que nous employons 6 vecteurs"arcs" et 4 vecteurs "sommets"

Note : l'étape 1 de l'algorithme de C.K. n'augmente à aucun pas le nombre e' d'arcs.

Remarque :

- Il est difficile de connaître au départ n' et e'
- La représentation décrite n'est pas utilisée dès la recherche des sous-systèmes irréductibles .
  a. On espère que n' et e' sont respectivement plus petits que n et e ( d'où cette représentation ne prendrait pas"beaucoup" de place ).
  b. On veut dans l'algorithme C.K. (cfr.ch.II) analyser 1 C.F.C. indépendamment des autres; on ne peut pas représenter la matrice M associée au graphe de flux par des"listes enchainées": isoler des sous-matrices ( associées à des sous-graphes F.C.) de la matrice M enchainée deviendrait difficile et couteux. La représentation décrite se construit facilement et rapidement ( Voir IV.B. Routine SPACFC )

IV.A.2.b.2. Représentation de table de couverture.

Les étapes 2-3-4 de l'algorithme de C.K. exigent l'utilisation d'un graphe étiqueté et ensuite d'une table de couverture .

Les opérations de génération des circuits pertinents

(ch.II. étape 2 de C.K.) exigent une représentation de p-graphe (Fichefet. P.II.7). Nous ne pourrions plus utiliser une représentation de matrice booléenne (celle-ci ne peut pas représenter un p-graphe!)

Cependant : Nous garderons la représentation de 1-graphe ( de matrice booléenne ) décrite plus-haut ( IV.A.2.b.1.)
Il suffit,à chaque arc, d'associer une liste des étiquettes ( voir déf. ch.II.étape 2 de C.K. ), qui simulera l'existence de plusieurs arcs entre deux sommets.

Comment effectuons-nous cela ?

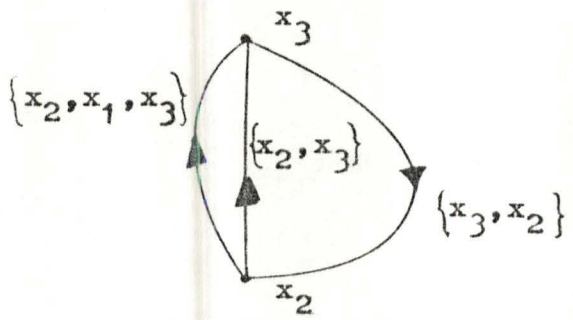
Supposons qu'à l'arc t correspondent les étiquettes $b_1,\ldots,b_1$.

1. Le vecteur IGRNTR fait correspondre à l'arc t l'étiquette $b_1$.
2. Le vecteur IPLUS fait correspondre :
   à l'étiquette $b_1$ , l'étiquette $b_2$ .
   à l'étiquette $b_2$ , l'étiquette $b_3$ .
   etc...
   à l'étiquette $b_1$ , un numéro d'étiquette non attribué ( 0 ).
3. Précisons que les étiquettes sont des colonnes d'une matrice booléenne COV. Les lignes de cette matrice correspondent chacune respectivement à un sommet du graphe que l'on étiquète. Si la colonne i de COV correspond à l'étiquette $b_i$, la valeur 1 à la ligne j de la colonne i signifiera que le sommet qui correspond à la ligne j appartient à l'étiquette $b_i$
4. En cours de l'étape 2 de l'algorithme C.K., les circuits pertinents détectés, seront des colonnes"particulières" de la matrice COV; les numéros de ces colonnes n'apparaîtront pas dans les vecteurs IPLUS et IGRNTR.
5. Une liste linéaire (stack - cfr. Knuth. P.234.)contiendra les numéros de colonnes de COV correspondant aux circuits pertinents. Ce vecteur sera appelé ICIPER .
6. Le vecteur NOCROS indiquera pour chaque colonne de la table de couverture et des étiquettes, le nombre d'éléments non nuls que contient cette colonne.

Exemple :

Soit à représenter le graphe étiqueté fig.16. qui était avant réduction partielle un graphe de sommets $\{x_1, x_2, x_3\}$

fig.16.

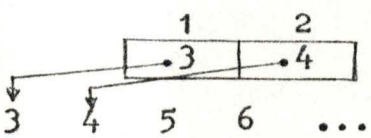



Il lui correspond la représentation de la fig.17.

L'arc ( $x_2$ , $x_3$ ) est numéroté 1

L'arc ( $x_3$ , $x_2$ ) est numéroté 2

IGRNTR : correspondance arc-étiquette :




| 1 | 2 | 3 | 4 | 5 | 6 | ... |
|---|---|---|---|---|---|---|

IPLUS : "enchainement" des étiquettes.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0 | 0 | 2 | 0 | 0 | 0 | |

COV : table des étiquettes et des circuits.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $x_1$ | 0 | 1 | 0 | 0 | 1 | 0 | |
| $x_2$ | 0 | 1 | 1 | 1 | 1 | 0 | |
| $x_3$ | 0 | 1 | 1 | 1 | 0 | 0 | |

fig.17

NOCROS : nombre d'éléments non nuls dans une colonne.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 3 | 2 | 2 | 2 | 0 | |

ICIPER : circuits enregistrés :

| | |
|---|---|
| 5 | |

( en supposant qu'1 circuit soit détecté)

Note : Les colonnes correspondant aux circuits pertinents formeront la table de couverture à réduire dans l'étape 3 de l'algorithme de C.K. Cette table est une sous-matrice de COV.

Le problème de représentation des étiquettes pose des difficultés :
Comment dimensionner la table de couverture et des étiquettes COV ?
Nous ne connaissons pas au départ :

1. le nombre de lignes de la table (nombre de sommets du graphe à étiqueter )
2. le nombre de colonnes de cette table qui seront nécessaires au cours du déroulement de l'algorithme.

A cause de l'étape 2 de l'algorithme de C.K. ( ch.II étape 2 ), nous faisons remarquer que , à chaque application de la règle de construction d'arcs T4 , le nombre de colonnes utilisées de COV peut croître énormément.
Soit un sommet x qui a 4 arcs incidents vers l'intérieur et 4 arcs incidents vers l'extérieur. Supposons les étiquettes de ces arcs simples(voir déf. dans ch.II). 8 étiquettes sont alors occupées par ces arcs. Si aucune simplification n'intervient, après application de T4 à x, 4 X 4 = 16 étiquettes seront occupées. Remplaçons les 4 arcs incidents vers l'intérieur à x par 8 arcs incidents vers l'intérieur à x ( 12 étiquettes occupées ). Après application de T4, 8 X 4 = 32 étiquettes peuvent être occupées.

Il faut malheureusement connaître une estimation acceptable d'une borne supérieure du nombre de colonnes occupées au cours de l'algorithme, dans la matrice booléenne COV.
Ce problème n'est pas résolu !
Est-il avantageux de construire une représentation de la matrice COV à l'aide de listes enchainées ?
Oui si les deux conditions ci-dessous sont satisfaites :

1. La table de couverture est"creuse".
2. On peut trouver une forme de listes enchainées qui se prête aux réductions ultérieures de table de couverture ( voir ch.II étapes 3 et 4 ).
   Ce second point nous paraît résolu si nous tenons compte de la construction de la matrice $M'^{T}$ dans IV.A.2.b.1. Une représentation similaire se prêteraitaux opérations de réduction de table.
   Remarquer que le balayage discuté en IV.A.2.b.1. est une technique de repérage employée pour construire des listes à deux

dimensions donc en particulier la table de couverture.

Le premier point nous paraît discutable : nous pensons qu'il peut apparaître comme assez courant que des étiquettes contiennent un "grand nombre" de sommets ( l'application de T4 combine des étiquettes ) sans que l'on puisse appliquer ensuite un règle de suppression des étiquettes à grand nombre de sommets ( P.Ex. R6 ). La matrice COV ne serait alors plus creuse.
Cependant l'expérience devrait déterminer si, en fait, notre supposition 1 est vérifiée. L'applicabilité de l'algorithme aux graphes de grande taille en dépend.

Nous sommes maintenant prêts à aborder l'exposé de la programmation de l'algorithme.

## IV.B. Description du programme HAUDON

Nous comptons dans cette partie du ch.IV donner quelques informations relatives à la compréhension du programme. Cela se limitera souvent à donner la signification des variables-clés et/ou un organigramme non détaillé. Parfois, nous donnerons des éléments de compréhension supplémentaires.
Nous invitons le lecteur à ne pas s'attarder en première lecture aux organigrammes marqués d'une * . Ceux-ci servent plus notre soucis d'être complet et de permettre un poursuite du travail que la compréhension globale du programme.
Le programme se découpe en routines.

### 1- Lecture des données et création de la représentation du graphe décrite en IV.A.2.a.

Routine CODE -
description des variables :
IS : vecteur Sommets de IV.A.2.a
IED : vecteur Arcs de IV.A.2.a

N : nombre de sommets du graphe.(taille du graphe)

IVECT : vecteur qui contient les données lues d'une carte.

IDEB : sommet dont on est occupé à traiter les suivants .

INDS : dernier sommet dont on a traité les suivants .

INDEDG : "indice de remplissage" du vecteur Arcs .

Note : les données sur une carte se présentent comme suit :

1- dix données par carte.

2- 1ère donnée de la carte : un sommet dont on va indiquer les suivants

3- Les 9 données suivantes : les suivants du sommet indiqué en première colonne de la carte.

4- dès qu'une donnée est nulle :

   A- si c'est la première donnée , on a terminé la lecture du graphe.

   B- si c'est une des 9 données suivantes, il n'y a plus de données sur la carte.

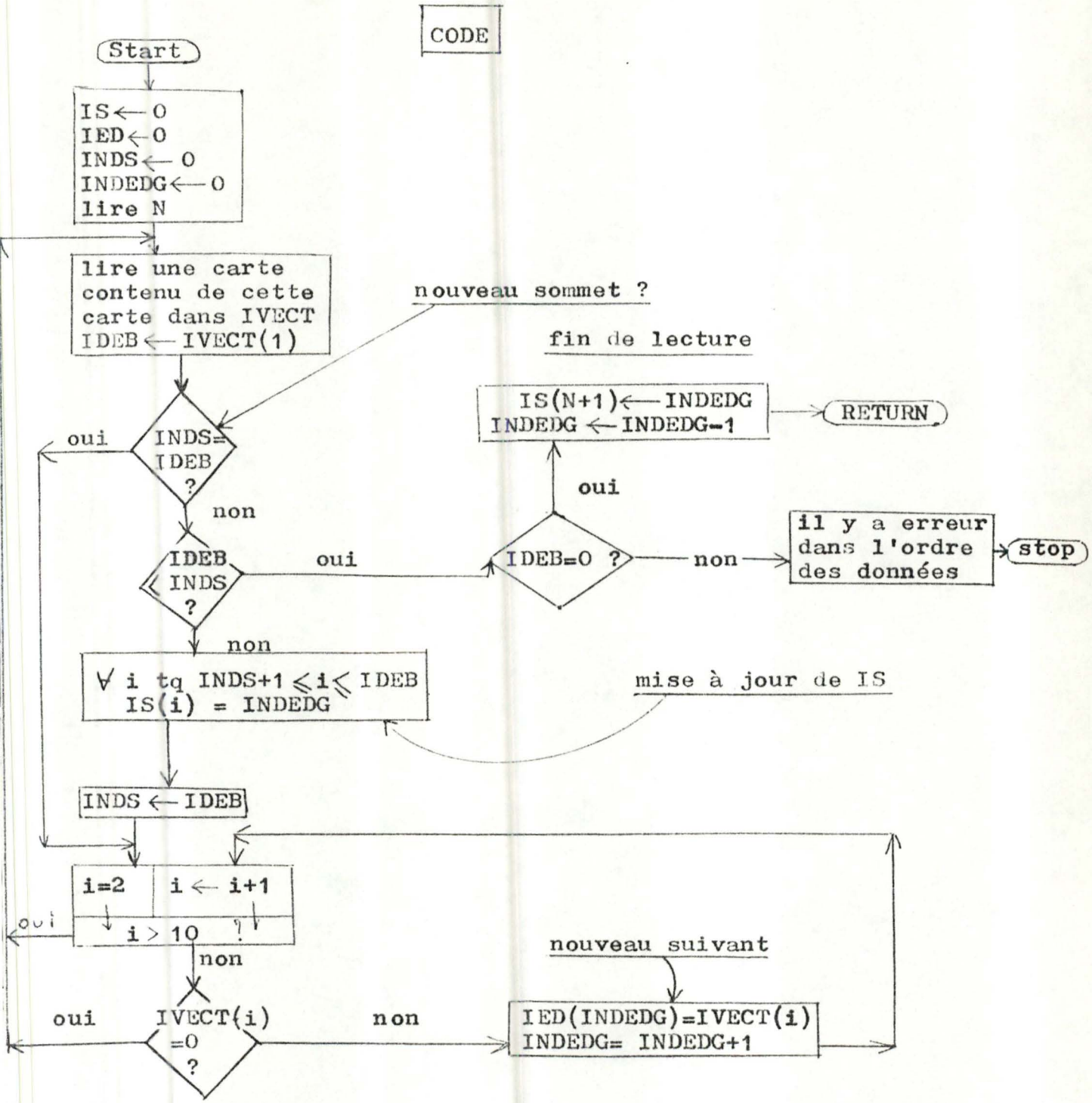
5- Si l'on a plus de 9 suivants, on utilise une autre carte que l'on commence de nouveau par le sommet dont on va continuer la liste des suivants.

+ Il est permis d'utiliser autant de cartes que l'on veut pour indiquer les suivants d'un sommet même si certaines de ces cartes ne comportent pas 9 suivants ou même aucun suivant.

+ Les sommets dont on indique les suivants sont classés par ordre croissant. Cela permet, dans la représentation décrite, d'éviter l'emploi d'un vecteur qui indique la fin de liste des suivants pour un sommet.

La toute première carte comporte la taille du graphe ( N ).

Voir organigramme page suivante .


CODE
Start
IS ← 0
IED ← 0
INDS ← 0
INDEDG ← 0
lire N
lire une carte
contenu de cette
carte dans IVECT
IDEB ← IVECT(1)
nouveau sommet ?
fin de lecture
IS(N+1) ← INDEDG
INDEDG ← INDEDG-1
RETURN
oui
INDS= IDEB ?
non
IDEB ≤ INDS ?
oui
IDEB=0 ?
non
il y a erreur dans l'ordre des données
stop
oui
non
∀ i tq INDS+1 ≤ i ≤ IDEB
IS(i) = INDEDG
mise à jour de IS
INDS ← IDEB
i=2
i ← i+1
oui
i > 10 ?
non
nouveau suivant
oui
IVECT(i) =0 ?
non
IED(INDEDG)=IVECT(i)
INDEDG= INDEDG+1

```
1         SUBROUTINE CODE(IS,IED,N,INDEDG,NN,MM)
2  C
3  C      CONSTRUCTION DE LA REPRESENTATION DU GRAPHE UTILISEE DANS LA PREMIERE PART
4  C      DE L"ALGORITHME
5  C
6  C      CODE LIT LES DONNEES, A SAVOIR LA TAILLE DU GRAPHE DE FLUX ET LE GRAPHE
7  C      DE FLUX LUI-MEME
8  C      VEUILLEZ VOUS REFERER AU TOME 2 MENTIONNE DANS LE PROGRAMME PRINCIPAL
9  C      POUR CONNAITRE LA DISPOSITION DES DONNEES SUR LES CARTES
10 C
11        IMPLICIT INTEGER*2(I-N)
12        DIMENSION IS(NN),IED(MM),IVECT(10)
13 C      INDEDG=NOMB.D'ARCS ET INDS=NOMB.DE SOMMETS
14        INDEDG=1
15        INDS=0
16 C      LECTURE DE LA TAILLE DU GRAPHE DE FLUX
17        READ 50,N
18 C      LECTURE D"UNE CARTE
19      6 READ 50,IVECT
20 C      ON PREND LES SUIVANTS D'UN SOMMET
21        IDEB=IVECT(1)
22 C      COMPARAISON DU PRECEDENT SOMMET TRAITE AVEC LE SOMMET A TRAITER
23        IF(INDS-IDEB)1,2,3
24      1 IA=INDS+1
25 C      ON MET A JOUR IS
26        DO 4 I=IA,IDEB
27 C      COMPLETER LE VECTEUR IS
28      4 IS(I)=INDEDG
29      2 INDS=IDEB
30 C      PAS SUPERFLU QUAND ON SORT DE BOUCLE 4
31        DO 5 I=2,10
32 C      FIN DES DONNEES SUR LA CARTE ?
33        IF(IVECT(I).EQ.0)GOTO 6
34        IVE=IVECT(I)
35        IED(INDEDG)=IVE
36 C      ON ENREGISTRE IVE COMME SUIVANT
37      5 INDEDG=INDEDG+1
38        GOTO 6
39      3 IF(IDEB.EQ.0)GOTO 7
40 C      ERREUR DANS DES DONNEES OU DANS LEUR ORDRE
41        PRINT 60
42        STOP
43 C      FIN DE LECTURE DU GRAPHE ET FIN DE CONSTRUCTION DE LA REPRESENTATION
44 C      DU GRAPHE
45      7 NP1=N+1
46        INDSP1=INDS+1
47        DO 8 I=INDSP1,NP1
48      8 IS(I)=INDEDG
49 C      ON ACHEVE LA CONSTRUCTION DU GRAPHE
50        IS(NP1)=INDEDG


51        INDEDG=INDEDG-1
52        RETURN
53     50 FORMAT(1018)
54     60 FORMAT(1H1,'ERREUR DANS ORDRE DES DONNEES')
55        END
```

## 2- "DEPTH-FIRST-SEARCH" utilisé dans l'algorithme de Tarjan

(voir I.2.C.b. )

Routine IDFS ( I- Depth-First-Search )

Il s'agit d'explorer un arc non encore exploré à partir d'un sommet actif ( cfr. I.2.C.b. ) . La fonction IDFS a pour valeur le sommet que l'on atteint si ce sommet existe ( si il existe un arc non exploré à partir du sommet actif) ; 0 sinon.

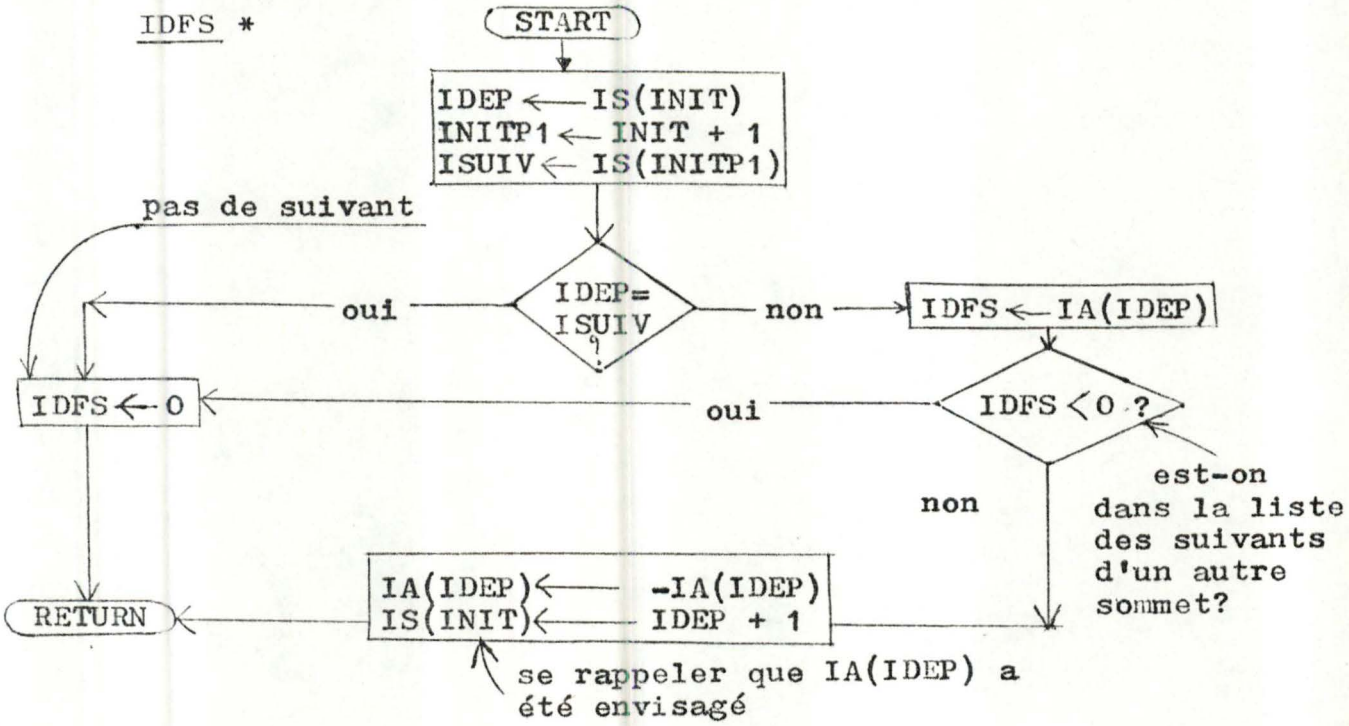
Noter que cette fonction transforme la représentation du graphe décrite en IV.A.2.a. en changeant les entrées du pointeur IS et en changeant le signe des entrées du vecteur IA. Après la recherche des C.F.C. qui utilise cette fonction, il suffit de retrouver la représentation décrite du graphe, ce qui est possible à partir de la valeur des pointeurs en fin d'opérations.

INIT : sommet actif

IS et IA : respectivement vecteur Sommet et vecteur Arc de la représentation du graphe.

IDEP : où trouver, dans IA, un suivant de INIT.

Organigramme :



```
      FUNCTION IDFS(IS,IA,INIT,NN,MM)
C
C     DEPTH FIRST SEARCH
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IS(NN),IA(MM)
      IDEP=IS(INIT)
      INITP1=INIT+1
C     INIT EST LE SOMMET DONT ON CHERCHE UN SUIVANT
      ISUIV=IS(INITP1)
C     TOUS LES ARCS D"ORIGINE INIT SONT EXPLORES?
      IF(IDEP.EQ.ISUIV)GOTO 1
      IDFS=IA(IDEP)
C     (INIT,IDFS)=ARC DEJA EXPLORE ?
      IF(IDFS.LE.0)GOTO1
C     ON VIENT D"EXPLORER L"ARC (INIT,IDFS)
      IA(IDEP)=-IA(IDEP)
      IS(INIT)=IDEP+1
      RETURN
C     PAS D"ARCS INEXPLORES A PARTIR DE INIT
    1 IDFS=0
      RETURN
C     VEILLER A REMETTRE LE GRAPHE EN ORDRE DANS LE PROGRAMME APPELANT IDFS APRE
C     S UTILISATION DE CELUI-CI
      END
```

3- Algorithme de Tarjan - Recherche de composantes fortement connexes . ( voir I.2.C.b. )

Routine STCNCT ("strong-connect"algorithm )

Cette routine recherche les composantes fortement connexes d'un graphe dont la représentation est décrite en IV.A.2.a. Elle utilise des vecteurs auxiliaires qui permettent que le temps de calcul soit fonction linéaire de n et e. ( voir I.2.C.b. ) en stockant des informations intermédiaires utiles.

ISOM et IARC : Représentation du graphe décrite en IV.A.2.a.

N et NEDG : respectivement le nombre de sommets et d'arcs

NUMBER : vecteur à deux utilisations

- numéroter les sommets lors du depth-first-search
- quand on a trouvé une C.F.C., il associe à chaque sommet de la C.F.C. le numéro de la C.F.C.

IGFC et NUMINV : permettent de stocker les C.F.C. d'une manière analogue à celle du stockage du graphe. IGFC(i) indique où dans NUMINV commence la liste des sommets de la C.F.C. numérotée i à l'instar de Sommets dans le par.IV.A.2.a.

NUMINV : vecteur qui, outre,l'utilisation que nous venons de décrire, contient à la place i + 1 un sommet prêt à devenir actif quand le sommet actuellement actif porte le numéro i ( ce qui évite dans le déroulement de l'algorithme de devoir éventuellement chercher parmi N sommets un nouveau sommet actif )

IPL : vecteur qui permet de connaître la place d'un sommet j dans le vecteur NUMINV ( même remarque que pour NUMINV )

ISTACK : vecteur qui sert de stack utilisé par l'algorithme.

IACDST : vecteur qui permet de connaître la place d'un sommet actif dans le stack.

IACTPR : vecteur qui donne pour un sommet, le sommet actif précédent

LL : vecteur qui,à un sommet, associe son "LOWLINK".

NOCFCN : nombre de C.F.C.

IFICFC : "indice de remplissage"de NUMINV ( doit être égal à N après utilisation de la routine.

IACT : sommet actif

I : numéro du sommet actif actuel

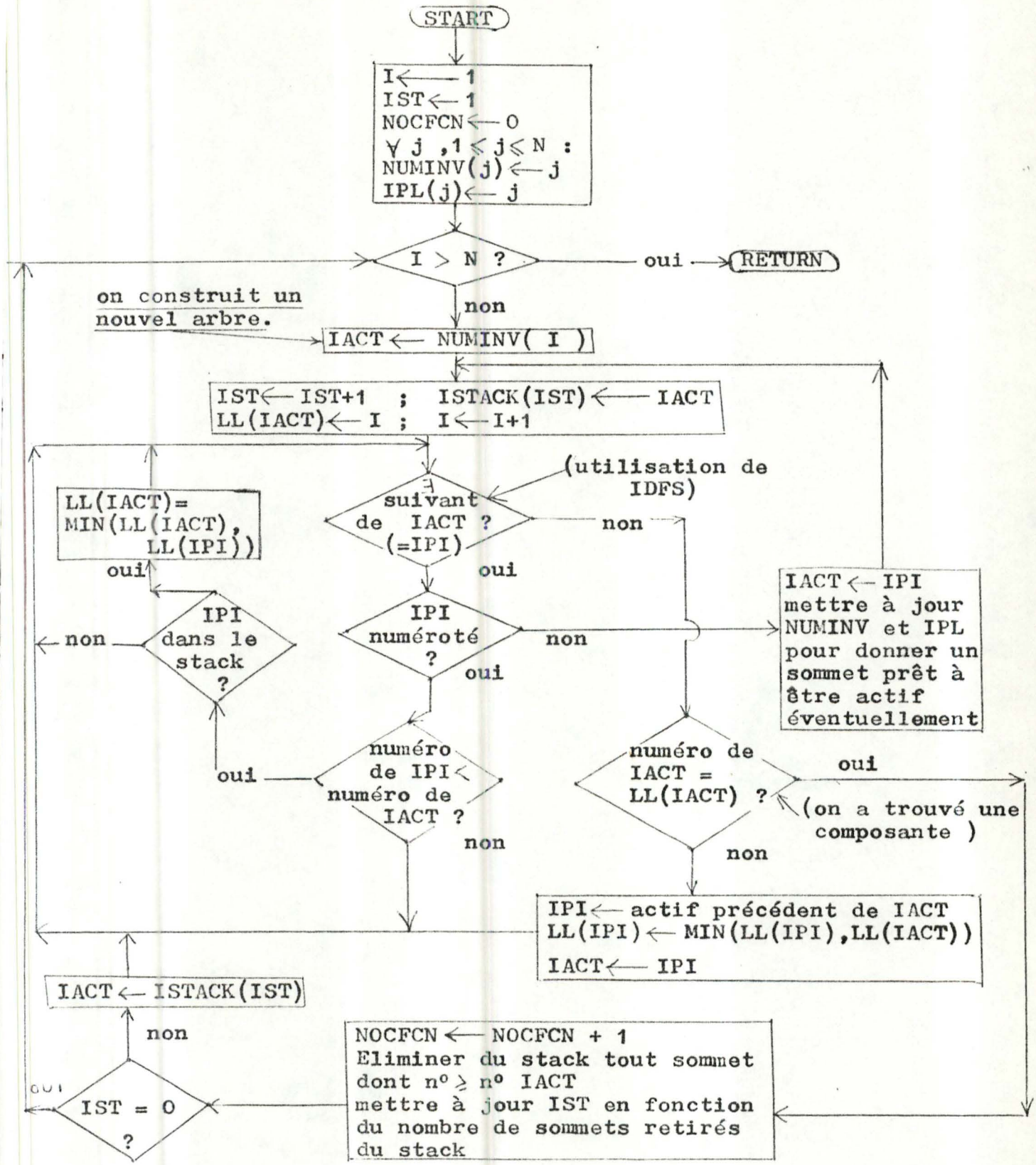
IST : "indice de remplissage"du stack

IPI : suivant envisagé du sommet actif

Voir organigramme à la page suivante

Note : STCNCT utilise la fonction IDFS

STCNCT *


START
I ← 1
IST ← 1
NOCFCN ← 0
∀ j , 1 ≤ j ≤ N :
NUMINV(j) ← j
IPL(j) ← j
I > N ?
oui
RETURN
non
on construit un nouvel arbre.
IACT ← NUMINV( I )
IST ← IST+1 ; ISTACK(IST) ← IACT
LL(IACT) ← I ; I ← I+1
(utilisation de IDFS)
∃ suivant de IACT ? (=IPI)
non
oui
LL(IACT) = MIN(LL(IACT), LL(IPI))
oui
IPI dans le stack ?
non
oui
IPI numéroté ?
non
oui
IACT ← IPI
mettre à jour NUMINV et IPL pour donner un sommet prêt à être actif éventuellement
numéro de IPI < numéro de IACT ?
oui
non
numéro de IACT = LL(IACT) ?
oui
(on a trouvé une composante )
non
IPI ← actif précédent de IACT
LL(IPI) ← MIN(LL(IPI),LL(IACT))
IACT ← IPI
IACT ← ISTACK(IST)
non
IST = 0 ?
oui
NOCFCN ← NOCFCN + 1
Eliminer du stack tout sommet dont n° ≥ n° IACT
mettre à jour IST en fonction du nombre de sommets retirés du stack

```
      SUBROUTINE STCNCT(NUMBER,N,ISOM,IARC,NEDG,IGFC,NOCFCN,IFICFC,NUMIN
     1V,IPL,IACDST,IACTPR,LL,MM,NN,ISTACK)
C
C     RECHERCHE DE COMPOSANTES FORTEMENT CONNEXES
C
      IMPLICIT INTEGER*2(I-N)
      INTEGER RA,RB
      DIMENSION NUMBER(NN),ISOM(NN),IARC(MM),IGFC(NN),NUMINV(NN),IPL(NN)
     1,IACDST(NN),IACTPR(NN),LL(NN),ISTACK(NN)
C     INITIALISATION DE NUMINV ET IPL
      DO 1 I=1,N
      NUMINV(I)=I
    1 IPL(I)=I
      I=1
      IST=0
      NOCFCN=0
      IFICFC=1
C     CONSTRUIRE UN NOUVEL ARBRE ?
    8 IF(I.GT.N)GOTO 2
      IACT=NUMINV(I)
C     NUMINV CONTIENT TOUJOURS A LA PLACE I UN SOMMET QUI DEVIENDRA ACTIF SI I=I
    9 IST=IST+1
C     ON TIENT UN SOMMET ACTIF IACT
C     IACT EST -IL UN SOMMET A CONSIDERER ?
      NUMBER(IACT)=-I
C     -I VEUT DIRE QUE IACT EST DANS LE STACK
      LL(IACT)=I
      ISTACK(IST)=IACT
      IACDST(IACT)=IST
      I=I+1
C     DEPTH FIRST SEARCH
    5 IPI=IDFS(ISOM,IARC,IACT,NN,MM)
      IF(IPI.EQ.0)GOTO 3
C     IPI  EST -IL UN SOMMET A CONSIDERER ?
      IF(NUMBER(IPI).EQ.0)GOTO 4
C     (IACT,IPI) ARC QUI N"EST PAS DE L"ARBRE
      RA=NUMBER(IPI)
      RB=NUMBER(IACT)
      IF((IABS(RA).GE.IABS(RB)).OR.(RA.GT.0))GOTO 5
      RA=LL(IACT)
      RB=LL(IPI)
      LL(IACT)=MIN0(RA,RB)
      GOTO 5
    3 RA=NUMBER(IACT)
      RB=LL(IACT)
C     REVENIR VERS LA RACINE DE L"ARBRE ?
      IF(IABS(RA).EQ.RB)GOTO 6
      IPI=IACTPR(IACT)
      RA=LL(IPI)
      RB=LL(IACT)
```

```
      LL(IPI)=MINO(RA,RB)
      IACT=IPI
      GOTO 5
C     SORTIE DE COMPOSANTE FORTEMENT CONNEXE
    6 NOCFCN=NOCFCN+1
      IGFC(NOCFCN)=IFICFC
      IBORN=IACDST(IACT)
      DO 7 L=IBORN,IST
      ITRANS=ISTACK(L)
      NUMINV(IFICFC)=ITRANS
      NUMBER(ITRANS)=NOCFCN
    7 IFICFC=IFICFC+1
      IST=IACDST(IACT)-1
C     CONSTRUIRE UN NOUVEL ARBRE ?
      IF(IST.EQ.0)GOTO 8
      IACT=ISTACK(IST)
      GOTO 5
    4 IACTPR(IPI)=IACT
C     (IACT,IPI) ARC DE L'ARBRE
C     NUMEROTATION DE IPI ET MISE A JOUR DE NUMINV
      IACT=IPI
      KTR=NUMINV(I)
      IPLACE=IPL(IACT)
      NUMINV(IPLACE)=KTR
      IPL(KTR)=IPLACE
      IPL(IACT)=I
      NUMINV(I)=IACT
      GOTO 9
    2 IND=NOCFCN+1
      IGFC(IND)=N+1
C     REMISE EN ORDRE DU GRAPHE APRES PASSAGE DANS IDFS
      IB=N+1
      IC=N
      DO 10 I=1,N
      ISOM(IB)=ISOM(IC)
      IB=IC
   10 IC=IC-1
      ISOM(1)=1
      DO 11 I=1,NEDG
   11 IARC(I)=-IARC(I)
      RETURN
      END
```

4- Construction du graphe condensé.

La routine GRACFC ( graphe des composantes fortement connexes) construit le graphe condensé.

ISOM et ISGR : représentation du graphe décrite en IV.A.2.a.

IGFC et NUMINV : vecteurs qui indiquent les sommets appartenant à une C.F.C. suivant une représentation analogue à celle décrite en IV.A.2.a.

IGCOND et ISUGL : représentation du graphe condensé analogue à celle décrite en IV.A.2.a.

IACTPR : vecteur qui permet de savoir si une C.F.C. est déjà indiquée, dans le graphe condensé, comme suivante d'une autre C.F.C.

NMPR : vecteur qui retient le nombre de précédents d'une C.F.C. dans le graphe condensé. Ce vecteur nous sera utile pour la décomposition en niveaux.

NUMBER : donne le numéro de C.F.C. d'un sommet du graphe de flux original.

NOCFCN : nombre de C.F.C. ( de sommets du graphe condensé )

IDEBUT : "indice de remplissage"du vecteur ISUGL

Voir organigramme à la page suivante.

5- Décomposition en niveaux.

La routine TOPSOR ( topological sort ) décompose en niveaux hiérarchiques le graphe condensé. En fait, le résultat est "plus" qu"un tri topologique.(voir I.2.C.c.)

NMPR : nombre de précédents d'un sommet du graphe condensé.

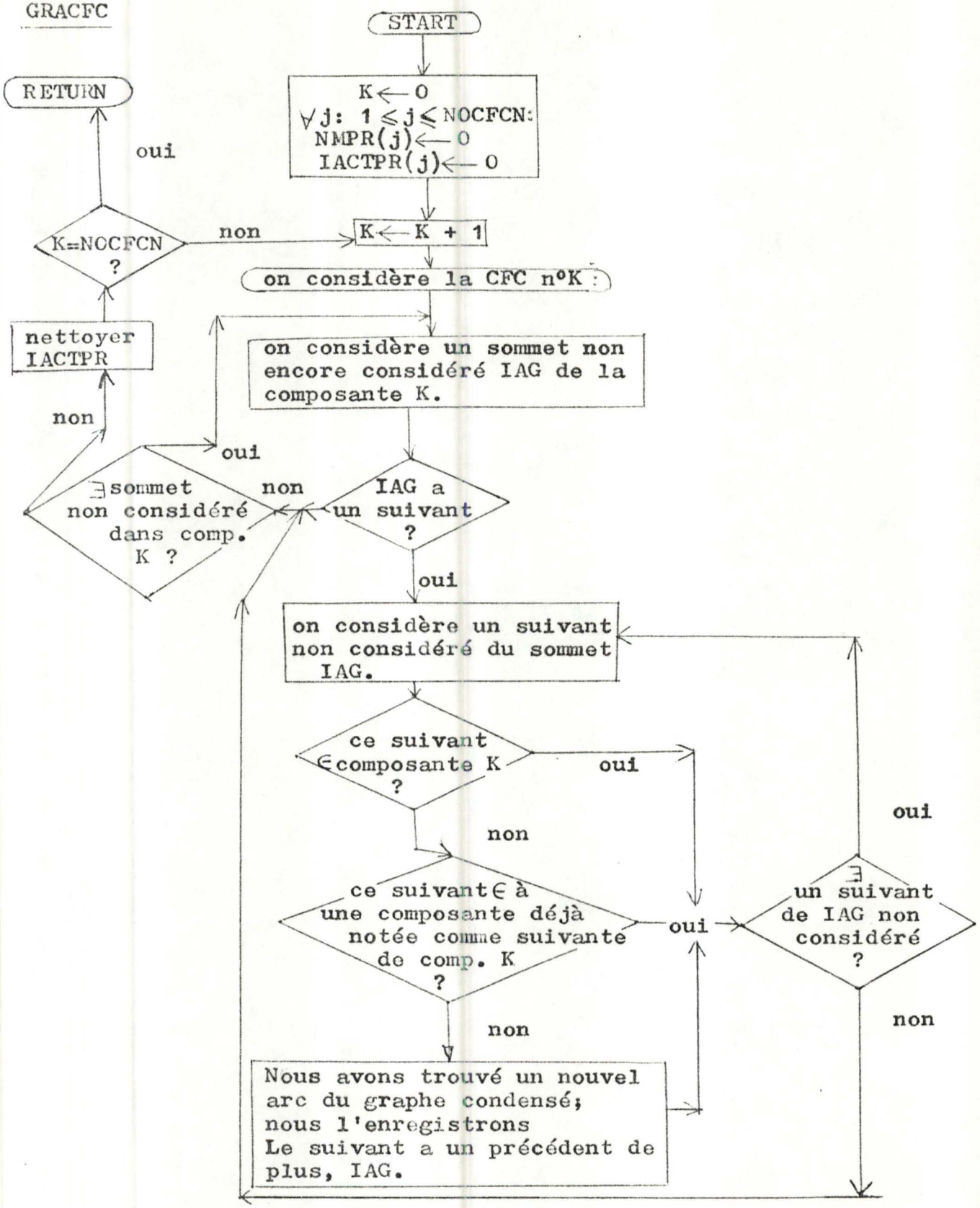
ISOMSU et IVECSO : représentation du graphe condensé

NIVEAU : vecteur qui indique où l'on peut trouver dans le vecteur ISUIVA les sommets du graphe condensé d'un certain niveau.

ISUIVA : vecteur qui "contient" les sommets du graphe condensé classés par niveaux.

Pour cette routine, nous ne donnons pas d'organigramme. Le"listing" se trouve après l'organigramme de GRACFC et le"listing" de GRACFC.

GRACFC


START
K ← 0
∀j: 1 ≤ j ≤ NOCFCN:
NMPR(j) ← 0
IACTPR(j) ← 0
K ← K + 1
on considère la CFC n°K :
on considère un sommet non encore considéré IAG de la composante K.
IAG a un suivant ?
oui
non
on considère un suivant non considéré du sommet IAG.
ce suivant ∈ composante K ?
oui
non
ce suivant ∈ à une composante déjà notée comme suivante de comp. K ?
oui
non
∃ un suivant de IAG non considéré ?
oui
non
Nous avons trouvé un nouvel arc du graphe condensé; nous l'enregistrons
Le suivant a un précédent de plus, IAG.
∃ sommet non considéré dans comp. K ?
non
oui
nettoyer IACTPR
non
K=NOCFCN ?
non
oui
RETURN

```
      SUBROUTINE GRACFC(NOCFCN,IACTPP,IGCOND,IGFC,ISUGL,NUMBER,NUMINV,IS
     1OM,ISGR,NN,MM,NMPR)
C
C     CONSTRUCTION DU GRAPHE CONDENSE
C
      INTEGER RR
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IACTPR(NN),IGCOND(NN),IGFC(NN),ISUGL(MM),NUMBER(NN),NUMI
     1NV(NN),ISOM(NN),ISGR(MM),NMPR(NN)
      IDEBUT=1
      DO 1 L=1,NOCFCN
      NMPR(L)=0
    1 IACTPR(L)=0
      DO 2 K=1,NOCFCN
      IGCOND(K)=IDEBUT
      IACTPR(K)=1
C     ON REGARDE OU CHERCHER LES SOMMETS DE LA COMPOSANTE K
      KP1=K+1
      I1=IGFC(K)
      I2=IGFC(KP1)-1
      DO 3 M=I1,I2
C     ON PREND UN SOMMET DE LA COMPOSANTE K
      IAG=NUMINV(M)
C     ON REGARDE LES SUIVANTS DE CE SOMMET
      J1=ISOM(IAG)
      IAGP1=IAG+1
      J2=ISOM(IAGP1)
C     SI PAS DE SUIVANTS CONTINUER AVEC UN AUTRE SOMMET
      IF(J1.EQ.J2)GOTO 3
      J2=J2-1
      DO 4 I4=J1,J2
      KSO=ISGR(I4)
C     ON A UN SUIVANT KSO
      NC=NUMBER(KSO)
C     A QUELLE COMPOSANTE APPARTIENT KSO
      IF(IACTPR(NC).EQ.1)GOTO 4
C     SI KSO DANS MEME COMPOSANTE ON PREND UN AUTRE SOMMET
      IACTPR(NC)=1
C     NC=COMPOSANTE UTILISEE
      ISUGL(IDEBUT)=NC
C     ON INDIQUE NC COMME SUIVANT(K)
C     ON INDIQUE LE NOMBRE DE PRECEDENTS D'UNE COMPOSANTE
      NMPR(NC)=NMPR(NC)+1
    4 IDEBUT=IDEBUT+1
    3 CONTINUE
C     ON REMET IACTPR A '0'AVANT DE REPARTIR
      K1=IGCOND(K)
      RR=IDEBUT-1
      K2=MAXO(RR,1)
      DO 5 IZ=K1,K2

      ITR=ISUGL(IZ)
    5 IACTPR(ITR)=0
      IACTPR(K)=0
    2 CONTINUE
      NOP1=NOCFCN+1
      IGCOND(NOP1)=IDEBUT
      RETURN
      END
```

```
      SUBROUTINE TOPSOR(N,IVECSO,ISOMSU,ISUIVA,NIVSOM,NIVEAU,NN,MM,I)
C
C     DECOMPOSITION EN NIVEAUX
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IVECSO(NN),ISOMSU(NN),ISUIVA(MM),NIVSOM(NN),NIVEAU(NN)
C     DETERMINATION DES SOMMETS SANS PRECEDENTS
      IM=1
      I=0
      J=0
    7 M=0
      I=I+1
      DO 1 K=1,N
      IF(IVECSO(K).NE.0) GOTO1
      J=J+1
      NIVSOM(J)=K
      IVECSO(K)=-1
      M=M+1
    1 CONTINUE
C     DETERMINATION DU VECTEUR NIVEAU
      NIVEAU(I)=IM
      IN=IM
      IM=IM+M
      IF(M.EQ.0) RETURN
C     MOCIFICATION DES VECTEURS
      IMM=IM-1
      DO 6 K=IN,IMM
      IFF=NIVSOM(K)
      IC=IFF+1
      ID=ISOMSU(IFF)
      IE=ISOMSU(IC)-1
      DO 6 IK=ID,IE
      IPS=ISUIVA(IK)
      IVECSO(IPS)=IVECSO(IPS)-1
    6 CONTINUE
      GOTO 7
      END
```

6- La routine SPACFC ( Sparse = creux et C.F.C.) construit la représentation de C.F.C. utilisée dans l'algorithme C.K. décrit dans le chapitre II.

NOCFCN : numéro de composante fortement connexe dont on construit la représentation.

ISOM et IARC : graphe de flux original sous la forme décrite en IV.A.2.a.

IGFC et NUMINV : vecteurs qui permettent de connaître les sommets appartenant à une C.F.C. Pour la description des 10 vecteurs suivants, se rappeler IV.A.2.b. Les 10 vecteurs décrits en IV.A.2.b.1. portent les noms respectifs :

1 ⟵ ICOL
2 ⟵ LIGN
3 ⟵ ISUICO
4 ⟵ IPRECO
5 ⟵ ISUILI
6 ⟵ IPRELI
7 ⟵ NOLIG
8 ⟵ NOCOL
9 ⟵ INDLIG
10 ⟵ INDCOL

Ces noms de vecteurs garderons la même signification dans toute la suite du travail.

NSCOMP : variable qui indique le nombre de sommets de la composante NOCFCN

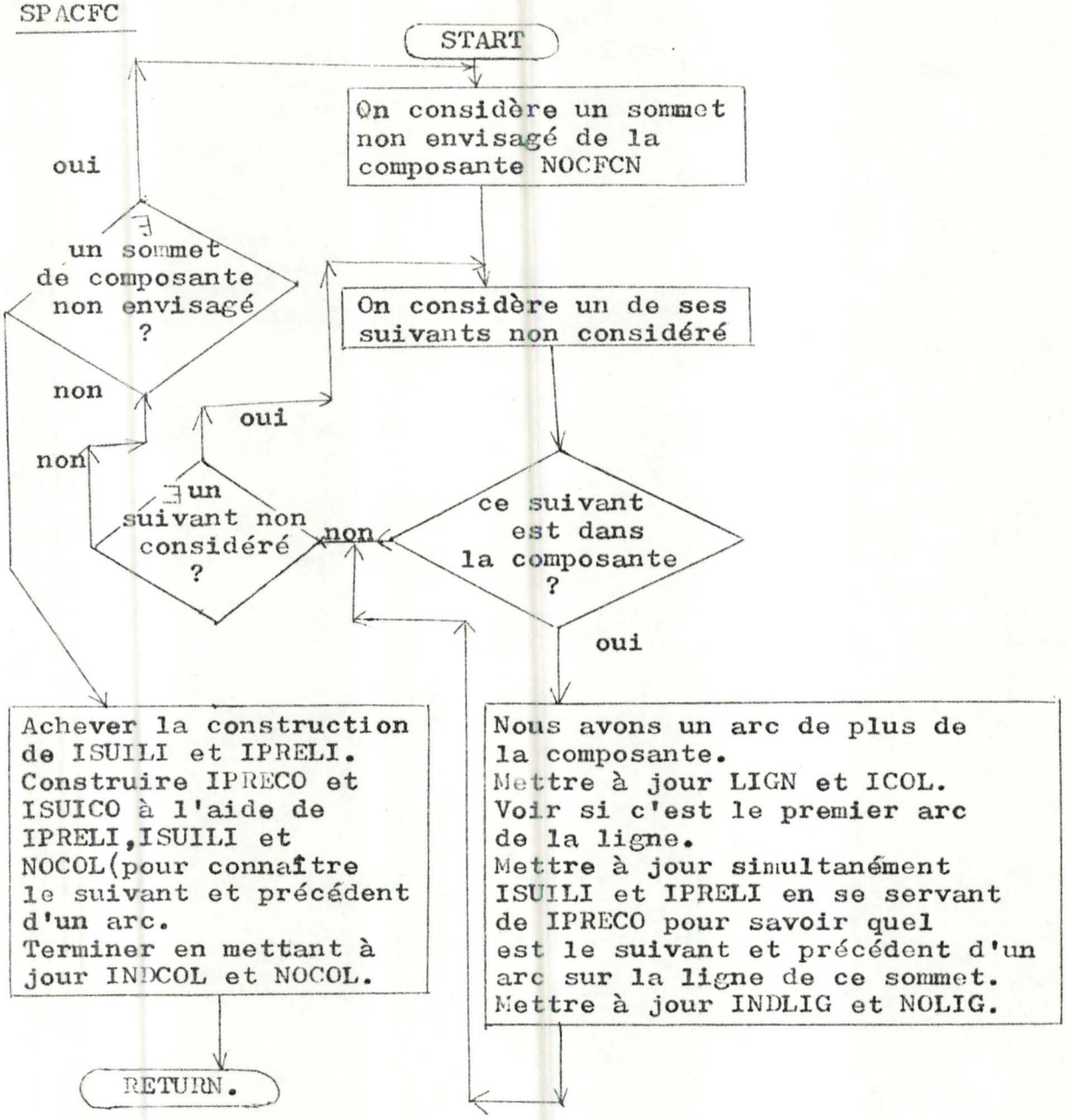
NOMBAR : variable qui indique le nombre d'arcs de la composante NOCFCN.

IOETNO et NOETSO : vecteurs qui permettent une numérotation des sommets de la composante de 1 à NOCFCN.

IOETNO : fait correspondre à un sommet son numéro

NOETSO : fait correspondre à un numéro le sommet qui le porte.

SPACFC


START
On considère un sommet non envisagé de la composante NOCFCN
oui
∃ un sommet de composante non envisagé ?
non
On considère un de ses suivants non considéré
oui
non
∃ un suivant non considéré ?
non
ce suivant est dans la composante ?
oui
Achever la construction de ISUILI et IPRELI. Construire IPRECO et ISUICO à l'aide de IPRELI, ISUILI et NOCOL (pour connaître le suivant et précédent d'un arc. Terminer en mettant à jour INDCOL et NOCOL.
Nous avons un arc de plus de la composante. Mettre à jour LIGN et ICOL. Voir si c'est le premier arc de la ligne. Mettre à jour simultanément ISUILI et IPRELI en se servant de IPRECO pour savoir quel est le suivant et précédent d'un arc sur la ligne de ce sommet. Mettre à jour INDLIG et NOLIG.
RETURN.

```
      SUBROUTINE SPACFC(NOCFCN,IGFC,NUMINV,ISOM,IARC,INDCOL,INDLIG,NOCOL
     1,NOLIG,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,NN,MM,NSCOMP,NOMBAR,I
     2OETNO,NOETSO)
C
C     CONSTRUCTION D"UNE FORME ENCHAINEE DE LA MATRICE ASSOCIEE A UNE COMPOSANTE
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IGFC(MM),NUMINV(NN),ISOM(NN),IARC(MM),INDCOL(NN),INDLIG(
     1NN),NOCOL(NN),NOLIG(NN),IPRECO(MM),ISUICO(MM),IPRELI(MM),ISUILI(MM
     2),ICOL(MM),LIGN(MM),IOETNO(NN),NOETSO(NN)
C     RECHERCHE DES SOMMETS D"UNE COMPOSANTE
      NOP1=NOCFCN+1
      L1=IGFC(NOCFCN)
      L2=IGFC(NOP1)-1
      NSCOMP=0
      DO 1 L=L1,L2
      NSCOMP=NSCOMP+1
      K=NUMINV(L)
C     NUMEROTATION DES SOMMETS D"UNE COMPOSANTE
      NOETSO(NSCOMP)=K
    1 IOETNO(K)=NSCOMP
      IF(NSCOMP.EQ.1)RETURN
      NOMBAR=0
C     ON ENREGISTRE LES SOMMETS DE LA COMPOSANTE
      DO 2 I1=1,NSCOMP
C     ON PREND LES SOMMETS DE LA COMPOSANTE UN A UN
      ISCOR=NOETSO(I1)
      ISP1=ISCOR+1
      K1=ISOM(ISCOR)
      K2=ISOM(ISP1)-1
      DO 3 K=K1,K2
C     ON PREND LES SUIVANTS DU SOMMET DE NUMERO I1
      KSOC=IARC(K)
C     KSOC DANS COMPOSANTE NOCFCN
      KTR=IOETNO(KSOC)
      IF(KTR.EQ.0)GOTO 3
      NOMBAR=NOMBAR+1
      ICOL(NOMBAR)=I1
      LIGN(NOMBAR)=KTR
      IF(INDLIG(KTR).NE.0)GOTO 4
C     RETENIR LE PREMIER ELEMENT D"UNE LIGNE
      INDLIG(KTR)=NOMBAR
      GOTO 5
C     CONSTRUCTION DU LIEN LIGNE
    4 MISO=IPRECO(KTR)
      IPRELI(NOMBAR)=MISO
      ISUILI(MISO)=NOMBAR
C     RETENIR LE PRECEDENT LIGNE D"UN SOMMET
    5 IPRECO(KTR)=NOMBAR
      NOLIG(KTR)=NOLIG(KTR)+1
```

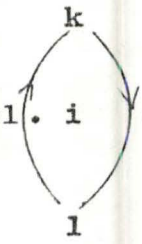
```
      3 CONTINUE
      2 CONTINUE
        M2=NSCOMP-1
        IND=1
C       CONSTRUCTION DES LIENS COLONNES
C       POUR CELA UTILISER UN BALAYAGE PAR LES LIENS LIGNES
        DO 6 M=1,M2
        IND=IND+1
        LK=INDLIG(IND)
        MK=IPRECO(M)
        IPRELI(LK)=MK
      6 ISUILI(MK)=LK
        LK=INDLIG(1)
        MK=IPRECO(NSCOMP)
        IPRELI(LK)=MK
        ISUILI(MK)=LK
        DO 7 I=1,NSCOMP
        IAR=INDLIG(I)
        LIMITE=NOLIG(I)
        DO 8 LI=1,LIMITE
        IT=ICOL(IAR)
        IF(INDCOL(IT).NE.0)GOTO 9
        INDCOL(IT)=IAR
        GOTO 10
C       UTILISER COMME VECTEUR AUXILIAIRE :NOCOL
      9 IV=NOCOL(IT)
        ISUICO(IV)=IAR
        IPRECO(IAR)=IV
     10 NOCOL(IT)=IAR
        IAR=ISUILI(IAR)
      8 CONTINUE
      7 CONTINUE
        IND=1
C       ACHEVER CONSTRUCTION DES LIENS COLONNES
        DO 11 M=1,M2
        IND=IND+1
        LK=INDCOL(IND)
        MK=NOCOL(M)
        IPRECO(LK)=MK
        ISUICO(MK)=LK
     11 NOCOL(M)=0
        LK=INDCOL(1)
        MK=NOCOL(NSCOMP)
        IPRECO(LK)=MK
        ISUICO(MK)=LK
        NOCOL(NSCOMP)=0
        IAR=INDLIG(1)
        DO 12 L=1,NOMBAR
        ITT=ICOL(IAR)
        NOCOL(ITT)=NOCOL(ITT)+1

     12 IAR=ISUILI(IAR)
        RETURN
C       VEILLER A NETTOYER SOETNO,INDLIG,INDCOL,NOLIG,NOCOL AVANT UTILISATION DE
C       CETTE SUBROUTINE
        END
```

7- La routine SAUCOL ( saut colonne): utilisée chaque fois que l'on considère qu'un arc n'appartient plus à une colonne de $M'^{T}$. Elle "ré-enchaine" les listes pour que l'arc ne soit plus considéré dans une colonne.

Soit un arc i. Les vecteurs IPRECO et ISUICO indiquent que l'arc i est"enchainé" respectivement à l'arc k qui le précède dans la colonne et à l'arc l qui le suit en colonne.(voir IV.A.2.b.1.) Considérer que l'arc i n'est plus dans une colonne de la matrice $M'^{T}$ , c'est indiquer que l'arc l a pour précédent en colonne l'arc k et que l'arc k a pour suivant en colonne l'arc l . voir fig.18

fig.18




ATTENTION : Par symétrie de la représentation décrite en IV.A.2.b. il suffit de remplacer les variables de SAUCOL se rapportant aux colonnes ( ICOL,NOCOL,ISUICO,IPRECO) par les variables analogues se rapportant aux lignes ( LIGN, NOLIG,ISUILI,IPRELI ) pour que la routine remplisse la même fonction par rapport aux lignes.
Cette note sera souvent valable dans la suite !

IPL : arc auquel nous appliquons SAUCOL.

ICOLPL : numéro de colonne de IPL

ISUCPL : arc suivant en colonne IPL.

Pour cette routine, pas d'organigramme

```
      SUBROUTINE SAUCOL(IPL,IPRECO,ISUICO,ICOL,NOCOL,ICOLPL,ISUCPL,NN,MM
     1)
C
C     DESTRUCTION DES LIENS COLONNES D"UNARC ET RACCORDS DES LIENS
C
C     DES ARCS PRECEDENTS ET SUIVANTS
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IPRECO(MM),ISUICO(MM),ICOL(MM),NOCOL(NN)
      IPRCPL=IPRECO(IPL)
      ISUCPL=ISUICO(IPL)
      ISUICO(IPRCPL)=ISUCPL
      IPRECO(ISUCPL)=IPRCPL
      ICOLPL=ICOL(IPL)
      NOCOL(ICOLPL)=NOCOL(ICOLPL)-1
      RETURN
      END
```

8- La routine ORDICO met éventuellement à jour le vecteur INDCOL après suppression d'un arc par la routine SAUCOL. En effet, si l'arc i était le premier arc de la colonne j, sa suppression doit entrainer une mise à jour du premier arc de la colonne j.
IPL, ICOLPL, ISUCPL : ont la même signification que les variables du même nom apparaissant dans SAUCOL.

```
      SUBROUTINE ORDICO(IPL,NOCOL,ICOLPL,INDCOL,ISUCPL,NN)
C
C     EVENTUELLEMENT ,APRES DESTRUCTION DE LIENS, MISE A JOUR DE DEBUT LIGNE
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION NOCOL(NN),INDCOL(NN)
      IF(NOCOL(ICOLPL).EQ.0)RETURN
      IF(IPL.EQ.INDCOL(ICOLPL))INDCOL(ICOLPL)=ISUCPL
      RETURN
      END
```

9- La routine DELEAR : détruit un arc ( T3 ) de la matrice $M'^{T}$.
Cette routine consiste en fait à "appeler" successivement SAUCOL et ORDICO avec arguments se rapportant aux colonnes et SAUCOL et ORDICO avec arguments se rapportant aux lignes. Il faut, en effet, briser successivement les "liens-ligne" et les "liens-colonnes".
Pour cette routine, pas d'organigramme.
DELEAR=delete arc
IPL : arc auquel on applique DELEAR.

```
       SUBROUTINE DELEAR(IPL,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN,
      1NOLIG,INDCOL,INDLIG,NN,MM)
C
C     ON DETRUIT UN ARC EN DETRUISANT SES LIENS LIGNE ET COLONNE
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IPRECO(MM),ISUICO(MM),IPRELI(MM),ISUILI(MM),ICOL(MM),NOC
     1OL(NN),LIGN(MM),NOLIG(NN),INDCOL(NN),INDLIG(NN)
      CALL SAUCOL(IPL,IPRECO,ISUICO,ICOL,NOCOL,ICOLPL,ISUCPL,NN,MM)
      CALL ORDICO(IPL,NOCOL,ICOLPL,INDCOL,ISUCPL,NN)
      CALL SAUCOL(IPL,IPRELI,ISUILI,LIGN,NOLIG,ILIGPL,ISULPL,NN,MM)
      CALL ORDICO(IPL,NOLIG,ILIGPL,INDLIG,ISULPL,NN)
      RETURN
      END
```

10- La routine DELESO effectue la destruction ( T1 ) d'un sommet. Cette destruction entraine des destructions d'arcs répertoriés dans un stack d'arcs libres.

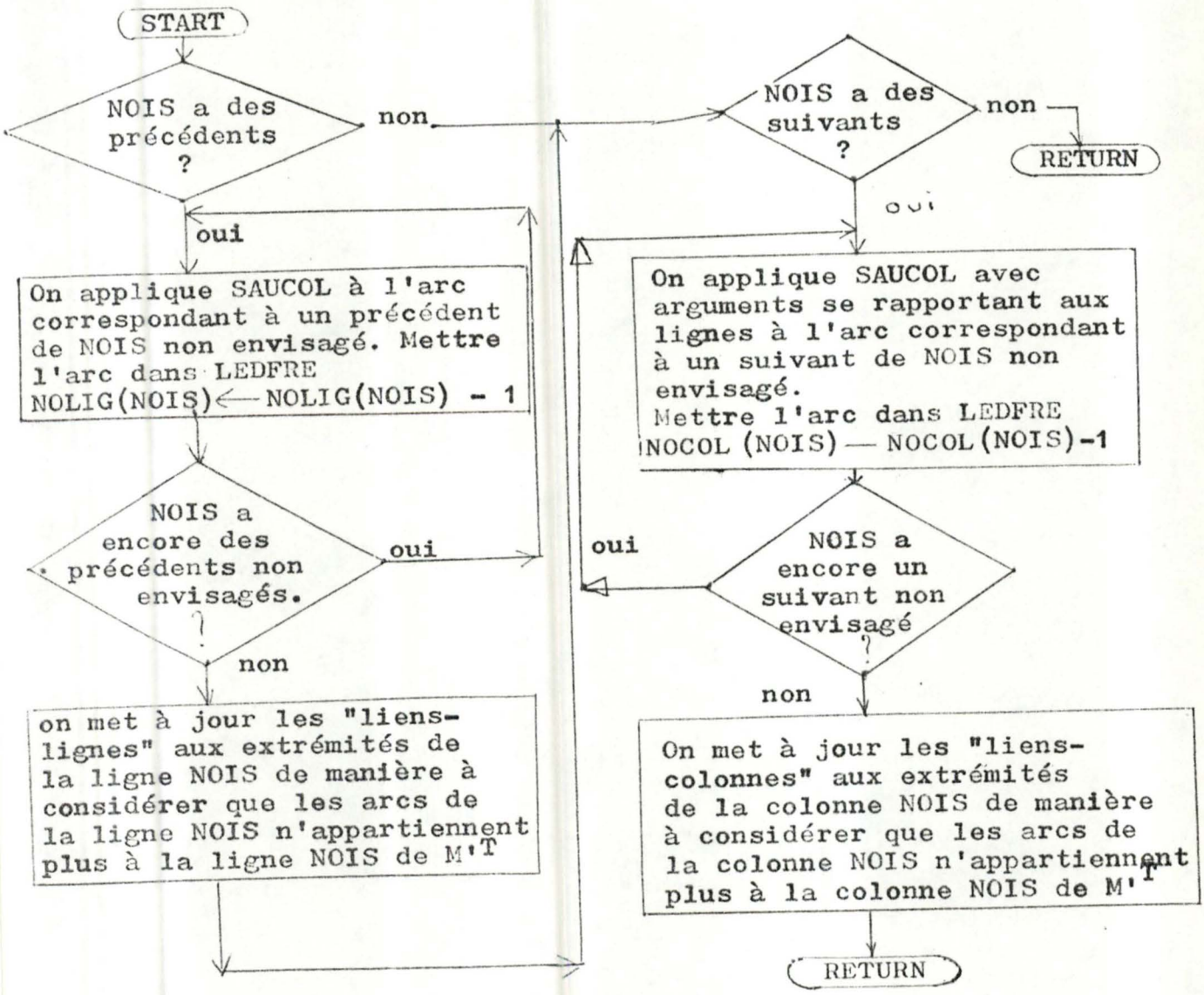
NOIS : sommet que l'on veut détruire.

LEDFRE : vecteur-stack d'arcs libres.

LNEDFR : "indice de remplissage" du stack des arcs libres.

DELESO = delete sommet.

DELESO



```
      SUBROUTINE DELESO(NOIS,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCO
     1L,INDLIG,NOCOL,NOLIG,NN,MM,LEDFRE)
C
C     DESTRUCTION D"UN SOMMET
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IPRECO(MM),ISUICO(MM),IPRELI(MM),ISUILI(MM),ICOL(MM),LIG
     1N(MM),INDCOL(NN),INDLIG(NN),NOCOL(NN),NOLIG(NN)
      DIMENSION LEDFRE(MM)
      LNEDFR=LEDFRE(MM)
      IF(NOLIG(NOIS).EQ.0)GOTO 1
      LIGIS=INDLIG(NOIS)
      KTRAV=LIGIS
C     POUR CHAQUE ELEMENT D"UNE LIGNE ,DETRUIRE LES LIENS COLONNES
    3 CALL SAUCOL(KTRAV,IPRECO,ISUICO,ICOL,NOCOL,INCIS,ISUCIS,NN,MM)
      CALL ORDICO(KTRAV,NOCOL,INCIS,INDCOL,ISUCIS,NN)
      LNEDFR=LNEDFR+1
      LEDFRE(LNEDFR)=KTRAV
      NOLIG(NOIS)=NOLIG(NOIS)-1
      IF(NOLIG(NOIS).EQ.0)GOTO 2
      KTRAV=ISUILI(KTRAV)
      GOTO 3
    2 IRACOP=IPRELI(LIGIS)
      IRACOS=ISUILI(KTRAV)
C     ON EFFECTUE DES RACCORDS AUX EXTREMITES D"UNE LIGNE
      ISUILI(IRACOP)=IRACOS
      IPRELI(IRACOS)=IRACOP
    1 IF(NOCOL(NOIS).EQ.0)GOTO 6
      ICOLIS=INDCOL(NOIS)
      KTRAV=ICOLIS
C     POUR CHAQUE ELEMENT DE COLONNE NOIS ,ON DETRUIT LES LIENS LIGNE
    5 CALL SAUCOL(KTRAV,IPRELI,ISUILI,LIGN,NOLIG,INLIS,ISULIS,NN,MM)
      CALL ORDICO(KTRAV,NOLIG,INLIS,INDLIG,ISULIS,NN)
      LNEDFR=LNEDFR+1
      LEDFRE(LNEDFR)=KTRAV
      NOCOL(NOIS)=NOCOL(NOIS)-1
      IF(NOCOL(NOIS).EQ.0)GOTO 4
      KTRAV=ISUICO(KTRAV)
      GOTO 5
C     ON EFFECTUE DES RACCORDS AUX EXTREMITES DE LA COLONNE
    4 IRACP=IPRECO(ICOLIS)
      IRACS=ISUICO(KTRAV)
      ISUICO(IRACP)=IRACS
      IPRECO(IRACS)=IRACP
    6 LEDFRE(MM)=LNEDFR
      RETURN
      END
```

11- La routine ELISOS effectue l'élimination ( T4 ) de sommets du graphe, dans le cadre de la règle 2 énoncée dans la présentation de l'algorithme de C.K. ( Chapitre II ). Ce qui signifie que le sommet que l'on veut éliminer n'a qu'un seul suivant ou qu'un seul précédent.

NOIS : sommet auquel on applique l'élimination. Ce sommet est supposé n'avoir qu'un seul suivant. Pour envisager l'élimination d'un sommet n'ayant qu'un précédent, il suffit dans ELISOS, de permuter les arguments se rapportant aux colonnes avec ceux se rapportant aux lignes.

ELISOS = élimination de sommet avec un seul suivant

J : arc correspondant au seul suivant de NOIS.

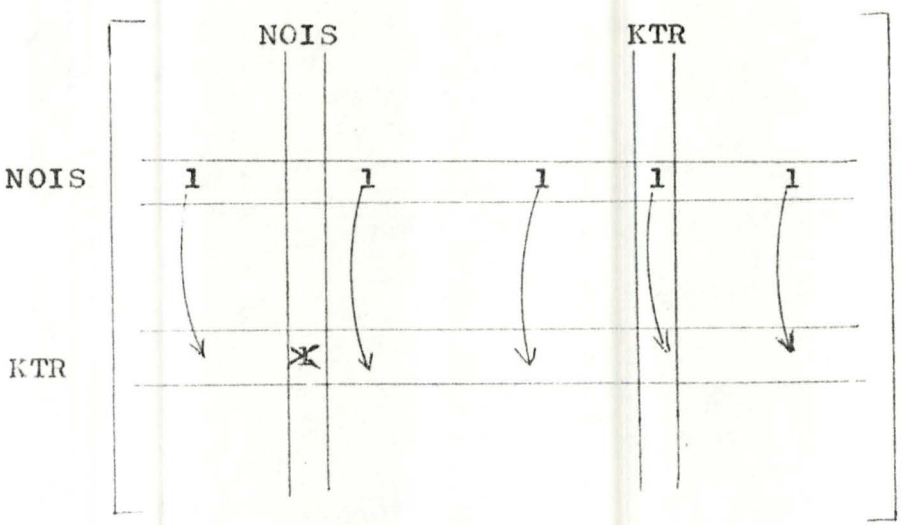
KTR : seul suivant de NOIS

IPREI : sommet précédent de NOIS envisagé.

IPREJ : sommet précédent de KTR envisagé.

Il peut être intéressant de se référer au petit shéma ci-dessous pour la compréhension de la routine.

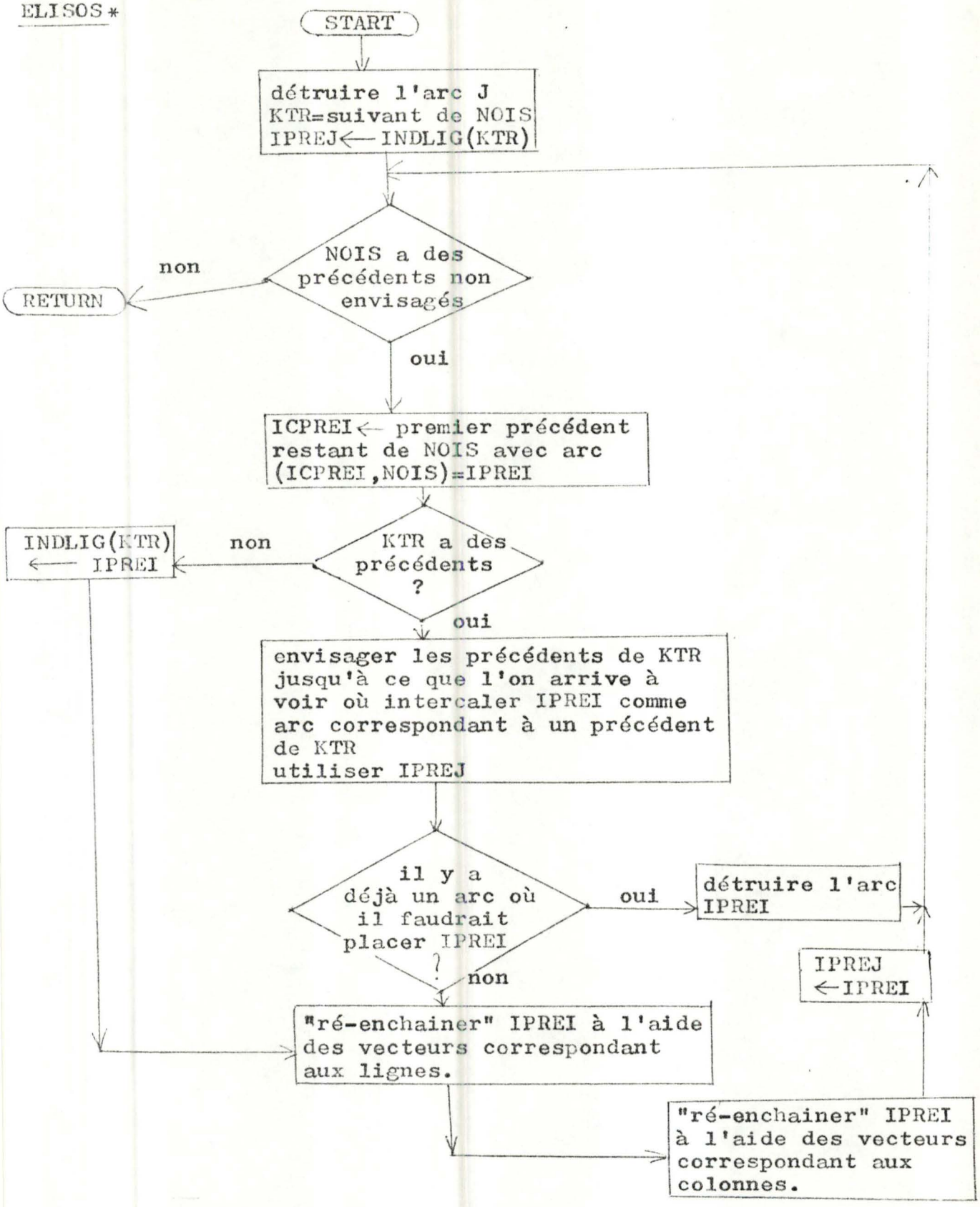
L'organigramme se trouve à la page suivante.




Matrice $M'^{T}$

Les flèches indiquent comment on traite les arcs correspondant aux précédents de NOIS : Ils doivent devenir des précédents de KTR.

ELISOS *


START
détruire l'arc J
KTR=suivant de NOIS
IPREJ← INDLIG(KTR)
NOIS a des précédents non envisagés
non
RETURN
oui
ICPREI← premier précédent restant de NOIS avec arc (ICPREI,NOIS)=IPREI
KTR a des précédents ?
non
INDLIG(KTR) ← IPREI
oui
envisager les précédents de KTR jusqu'à ce que l'on arrive à voir où intercaler IPREI comme arc correspondant à un précédent de KTR
utiliser IPREJ
il y a déjà un arc où il faudrait placer IPREI ?
oui
détruire l'arc IPREI
IPREJ ←IPREI
non
"ré-enchainer" IPREI à l'aide des vecteurs correspondant aux lignes.
"ré-enchainer" IPREI à l'aide des vecteurs correspondant aux colonnes.

```
      SUBROUTINE ELISOS(NOIS,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCO
     1L,INDLIG,NOCOL,NOLIG,NN,MM,LEDFRE)
C
C     ELIMINATION D'UN SOMMET
C
C     CE SOMMET EST SUPPOSE N'AVOIR QU'UN SEUL SUIVANT
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IPRECO(MM),ISUICO(MM),IPRELI(MM),ISUILI(MM),ICOL(MM),LIG
     1N(MM),INDCOL(NN),INDLIG(NN),NOCOL(NN),NOLIG(NN)
      DIMENSION LEDFRE(MM)
C     ON DETRUIT L'ARC CORRESPONDANT AU SUIVANT DE NOIS
      LNEDFR=LEDFRE(MM)
      J=INDCOL(NOIS)
      KTR=LIGN(J)
      IF(KTR.EQ.NOIS)RETURN
      CALL DELEAR(J,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN,NOLIG,IN
     1DCOL,INDLIG,NN,MM)
      LNEDFR=LNEDFR+1
      LEDFRE(LNEDFR)=J
C     INITIALISATION DU PROCESSUS
      IPREJ=INDLIG(KTR)
C     NOIS A ENCORE DES PRECEDENTS?
    5 IF(NOLIG(NOIS).EQ.0)GOTO 14
C     ON PREND UN PRECEDENT
      IPREI=INDLIG(NOIS)
      ICPREI=ICOL(IPREI)
C     ON FAIT COMME SI ICPREI N'EST PLUS PRECEDENT DE NOIS
      CALL SAUCOL(IPREI,IPRELI,ISUILI,LIGN,NOLIG,ILIGPL,ISULPL,NN,MM)
      CALL ORDICO(IPREI,NOLIG,ILIGPL,INDLIG,ISULPL,NN)
      IF(NOLIG(KTR).EQ.0)GOTO 1
C     ON PREND DES PRECEDENTS DE KTR JUSQU'A CE QU'ON SACHE OU INTERCALER KTR
    4 IF((LIGN(IPREJ).NE.KTR).OR.(ICOL(IPREJ).GT.ICPREI))GOTO 2
      IF(ICOL(IPREJ).EQ.ICPREI)GOTO 3
      IPREJ=ISUILI(IPREJ)
      GOTO 4
C     IL EXISTE DEJA UN ARC
    3 CALL SAUCOL(IPREI,IPRECO,ISUICO,ICOL,NOCOL,ICOLPL,ISUCPL,NN,MM)
      CALL ORDICO(IPREI,NOCOL,ICOLPL,INDCOL,ISUCPL,NN)
      LNEDFR=LNEDFR+1
      LEDFRE(LNEDFR)=IPREI
      GOTO 5
    2 IF(IPREJ.NE.INDLIG(KTR))GOTO 6
      INDLIG(KTR)=IPREI
C     ON MET ICPREI COMME PRECEDENT DE KTR
    6 IP=IPRELI(IPREJ)
C     EFFECTUER UN RACCORD-LIGNE
      IPRELI(IPREJ)=IPREI
      ISUILI(IPREI)=IPREJ
      IPRELI(IPREI)=IP
      ISUILI(IP)=IPREI
```

```
      9 IPREJ=IPREI
        NOLIG(KTR)=NOLIG(KTR)+1
C       ON REMET EN ORDRE LA COLONNE ICPREI
        KSOCOU=IPREI
        IF(KTR.LE.NOIS)GOTO 7
C       ON EFFECTUE UN RACCORD COLONNE EN "DESCENDANT" DANS LA COLONNE
      8 KSOCOU=ISUICO(KSOCOU)
        IF(KSOCOU.EQ.ISUICO(KSOCOU))GOTO 10
        IF((LIGN(KSOCOU).LT.KTR).AND.(ICOL(KSOCOU).EQ.ICPREI))GOTO 8
        LIGN(IPREI)=KTR
        IF(IPRECO(KSOCOU).EQ.IPREI)GOTO 5
        NOCOL(ICPREI)=NOCOL(ICPREI)+1
        CALL SAUCOL(IPREI,IPRECO,ISUICO,ICOL,NOCOL,ICOLPL,ISUCPL,NN,MM)
        CALL ORDICO(IPREI,NOCOL,ICOLPL,INDCOL,ISUCPL,NN)
        IP=IPRECO(KSOCOU)
        ISUICO(IP)=IPREI
        IPRECO(IPREI)=IP
        ISUICO(IPREI)=KSOCOU
        IPRECO(KSOCOU)=IPREI
        GOTO 5
C       ON EFFECTUE UN RACCORD COLONNE EN "MONTANT DANS LA COLONNE
      7 KSOCOU=IPRECO(KSOCOU)
        IF(KSOCOU.EQ.IPRECO(KSOCOU))GOTO 10
        IF((LIGN(KSOCOU).GT.KTR).AND.(ICOL(KSOCOU).EQ.ICPREI))GOTO 7
        LIGN(IPREI)=KTR
        IF(ISUICO(KSOCOU).EQ.IPREI)GOTO 5
        NOCOL(ICPREI)=NOCOL(ICPREI)+1
        CALL SAUCOL(IPREI,IPRECO,ISUICO,ICOL,NOCOL,ICOLPL,ISUCPL,NN,MM)
        CALL ORDICO(IPREI,NOCOL,ICOLPL,INDCOL,ISUCPL,NN)
        IP=ISUICO(KSOCOU)
        IPRECO(IP)=IPREI
        ISUICO(IPREI)=IP
        IPRECO(IPREI)=KSOCOU
        ISUICO(KSOCOU)=IPREI
        IF(IP.EQ.INDCOL(ICPREI))INDCOL(ICPREI)=IPREI
        GOTO 5
C       QUELQUES SITUATIONS PARTICULIERES
C       LES RACCORDS SPECIAUX QUI LEUR CORRESPONDENT
      1 INDLIG(KTR)=IPREI
        IRACLI=IPRELI(J)
        IF(IRACLI.EQ.IPREI)GOTO 11
        IPRELI(IPREI)=IRACLI
        ISUILI(IRACLI)=IPREI
     13 IRACLI=ISUILI(J)
        IF(IRACLI.EQ.IPREI)GOTO 12
        IPRELI(IRACLI)=IPREI
        ISUILI(IPREI)=IRACLI
        GOTO 9
     11 IRACLI=IPRELI(IPREI)
        ISUILI(IRACLI)=IPREI

        IRACLI=ISUILI(IPREI)
        IPRELI(IRACLI)=IPREI
        GOTO 13
     12 IRACLI=ISUILI(IPREI)
        IPRELI(IRACLI)=IPREI
        IRACLI=IPRELI(IPREI)
        ISUILI(IRACLI)=IPREI
        GOTO 9
C       UN SEUL ARC POUR LE GRAPHE REDUIT
     10 INDLIG(KTR)=IPREI
        NOLIG(KTR)=0
        NOCOL(KTR)=0
        RETURN
     14 LEDFRE(MM)=LNEDFR
        RETURN
        END
```

12- La routine R1CETK applique à un sommet la règle 1 de l'algorithme C.K. ( ch. II )

NOSO : nombre de sommets encore considéré comme non détruits.

NOIS : sommet auquel on applique R1CETK

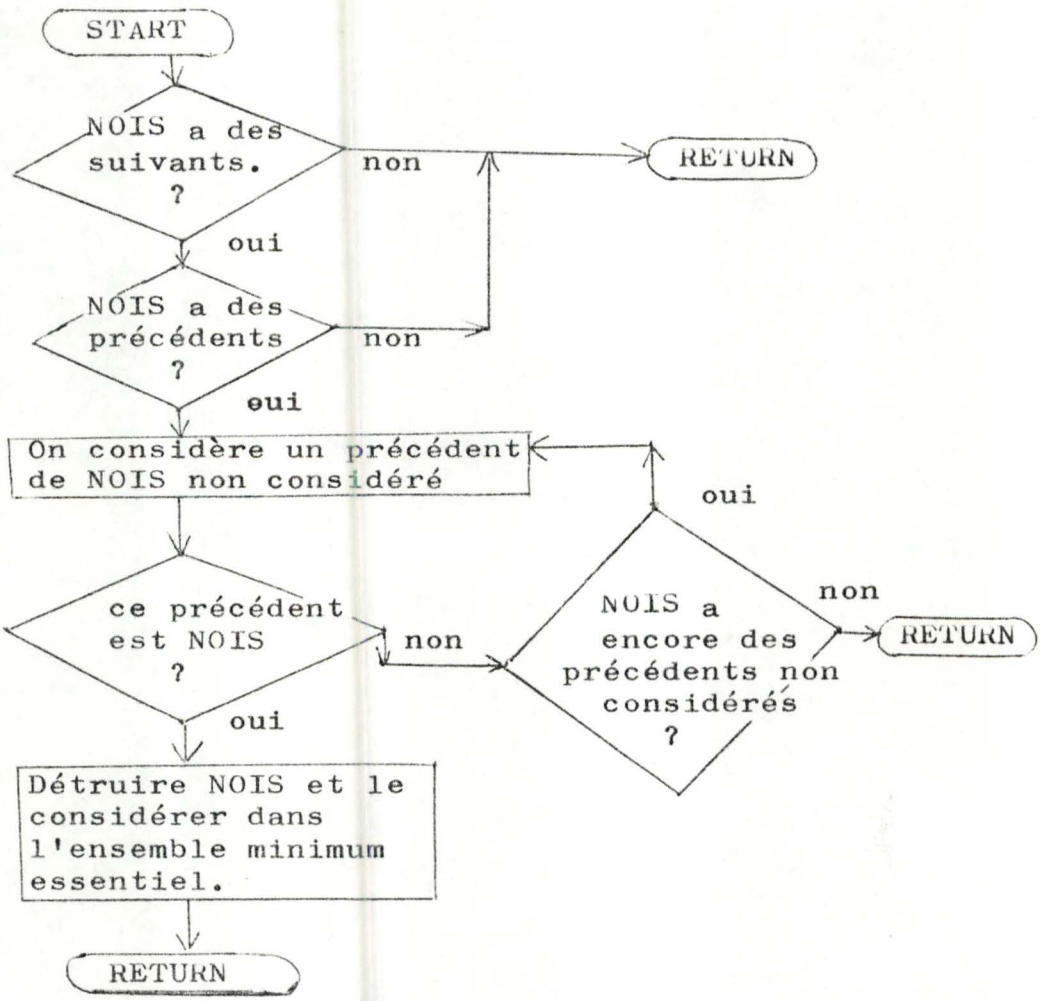
MES : vecteur- stack qui contiendra les sommets d'un ensemble minimum essentiel dont la recherche est en cours.

INDMES : "indice de remplissage" de MES.

I : variable qui prend la valeur 1 si la routine a permis de détruire un sommet ; 0 sinon.

Organigramme :

R1CETK



```
      SUPROUTINE R1CETK(I,NOIS,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,IND
     1COL,INDLIG,NOCOL,NOLIG,MES,INDMES,NOSO,NN,MM,LEDFRE)
C
C     REGLE 1 DE C.K.
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IPRECO(MM),ISUICO(MM),IPRELI(MM),ISUILI(MM),ICOL(MM),LIG
     1N(MM),INDCOL(NN),INDLIG(NN),NOCOL(NN),NOLIG(NN),MES(NN)
      DIMENSION LEDFRE(MM)
      I=0
      M=NOCOL(NOIS)
      IF(M.EQ.0)RETURN
      M=NOLIG(NOIS)
      IF(M.EQ.0)RETURN
      ISOC=INDLIG(NOIS)
      M1=1
C     BOUCLE DANS LE GRAPHE AU SOMMET NOIS ?
    2 IF(ICOL(ISOC).EQ.NOIS)GOTO 1
      IF(M1.EQ.M)RETURN
      M1=M1+1
      ISOC=ISUILI(ISOC)
      GOTO 2
    1 I=1
C     BOUCLE AU SOMMET NOIS . DETRUIRE LE SOMMET NOIS .NOIS EST ESSENTIEL
      CALL DELESO(NOIS,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCOL,INDL
     1IG,NOCOL,NOLIG,NN,MM,LEDFRE)
      NOSO=NOSO-1
      INDMES=INDMES+1
      MES(INDMES)=NOIS
      RETURN
      END
```

13- La routine R2CETK constate si un sommet x a un seul suivant ou un seul précédent et, si oui, élimine ce sommet x en lui appliquant ELISOS. Il s'agit de la règle 2 de l'algorithme C.K. ( voir ch.II étape 1 ).

I : variable qui prend la valeur 1 si le sommet x est éliminé.

NOIS : sommet auquel on applique R2CETK.

Pour cette routine, il n'y a pas d'organigramme.

14- La routine R3CETK constate si la règle 3 de l'algorithme C.K. s'applique au sommet x et, si oui, n'élimine cependant pas le sommet x.

NOIS : sommet auquel on applique la routine R3CETK.

I : variable qui prend la valeur 1 si la règle 3 s'applique au sommet NOIS.

IMAX : nombre de suivants de NOIS.

KSUI : un suivant de NOIS que l'on envisage à un moment donné.

INTMAX : nombre de suivants du sommet KSUI.

IARANT :
ICSUIV : compteurs qui permettent de détecter plus rapidement l'innaplicabilité de la règle 3 au sommet NOIS.
ICEGAL :

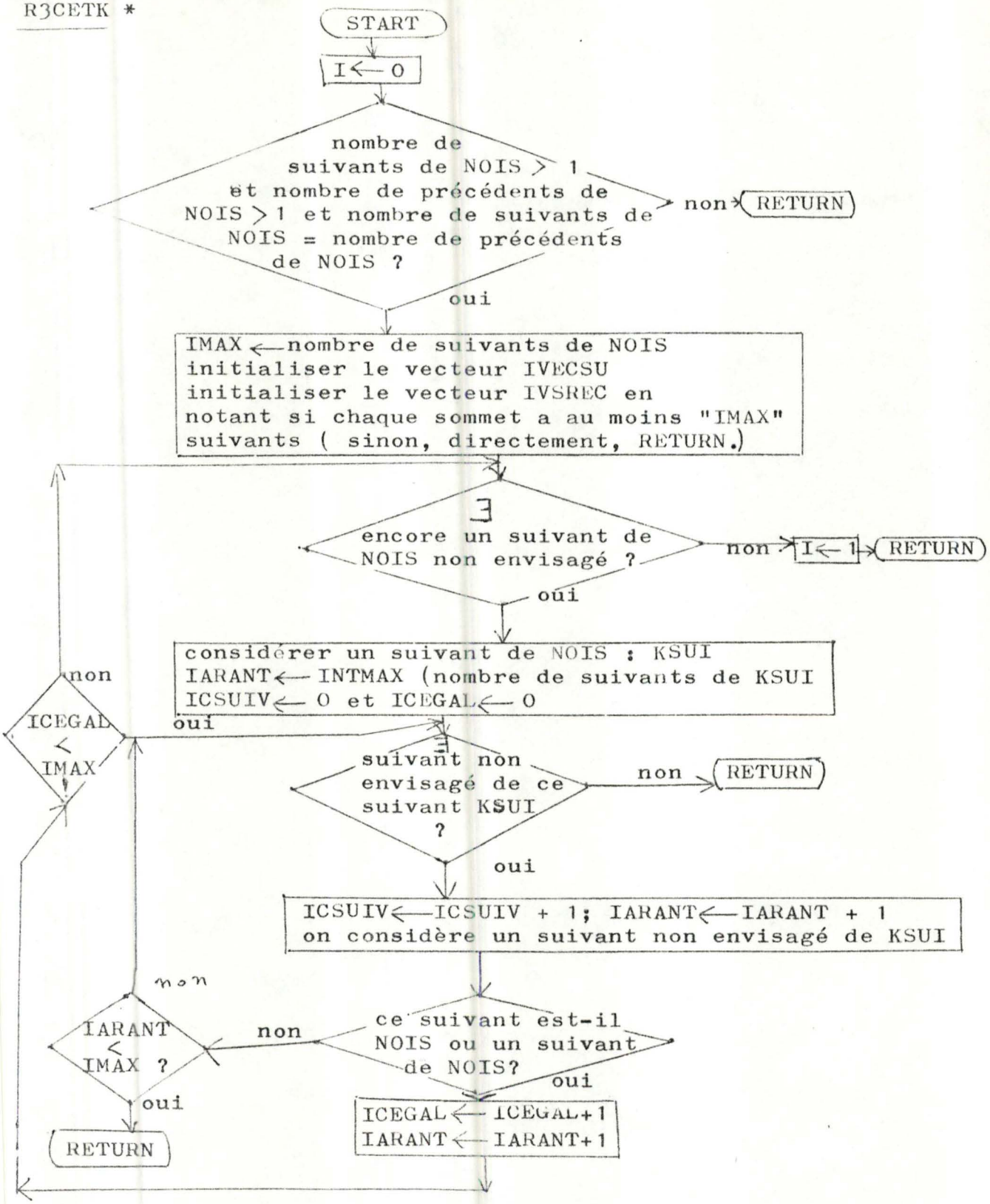
Ces compteurs se basent sur le principe suivant : quand on regarde si un suivant de KSUI est un suivant de NOIS ou NOIS lui-même, on effectue une opération inutile ( en regard du test : "sous-graphe complet"? - voir R3 de C.K. ) si à ce moment, d'autres suivants de KSUI ayant étés testés et n'étant, ni un autre suivant de NOIS ni NOIS lui-même, il ne reste de toute façon plus assez de suivants non testés de KSUI pour obtenir que le sous-graphe testé est complet.

IVECSU : vecteur faisant office de stack pour stocker les suivants du sommet NOIS.

IVSREC : vecteur indiquant, pour chaque sommet suivant de NOIS, combien le suivant a lui-même de sommets suivants.

voir organigramme.

R3CETK *


START
I ← 0
nombre de suivants de NOIS > 1 et nombre de précédents de NOIS > 1 et nombre de suivants de NOIS = nombre de précédents de NOIS ?
non
RETURN
oui
IMAX ← nombre de suivants de NOIS
initialiser le vecteur IVECSU
initialiser le vecteur IVSREC en notant si chaque sommet a au moins "IMAX" suivants ( sinon, directement, RETURN.)
∃
encore un suivant de NOIS non envisagé ?
non
I ← 1
RETURN
oui
considérer un suivant de NOIS : KSUI
IARANT ← INTMAX (nombre de suivants de KSUI
ICSUIV ← 0 et ICEGAL ← 0
non
ICEGAL < IMAX ?
oui
∃
suivant non envisagé de ce suivant KSUI ?
non
RETURN
oui
ICSUIV ← ICSUIV + 1; IARANT ← IARANT + 1
on considère un suivant non envisagé de KSUI
non
IARANT < IMAX ?
non
ce suivant est-il NOIS ou un suivant de NOIS?
oui
ICEGAL ← ICEGAL+1
IARANT ← IARANT+1
oui
RETURN

```
      SUBROUTINE R3CETK(I,NOIS,ISUICO,LIGN,INDCOL,NOCOL,NOLIG,IVECSU,IVS
     1REC,NOSO,NN,MM,ICVS)
C
C     REGLE 3 DE C.K.
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION ISUICO(MM),LIGN(MM),INDCOL(NN),NOCOL(NN),NOLIG(NN),IVECS
     1U(NN),IVSREC(NN)
      IVECSU(1)=1
      ICVS=0
      I=0
      IF((NOLIG(NOIS).LE.1).OR.(NOCOL(NOIS).LE.1).OR.(NOLIG(NOIS).NE.NOC
     1OL(NOIS)))RETURN
      IMAX=NOCOL(NOIS)
      IAR=INDCOL(NOIS)
    3 ISO=LIGN(IAR)
      ICVS=ICVS+1
C     CATALOGUER DANS IVECSU LES SUIVANTS DE NOIS
      IVECSU(ICVS)=ISO
      IF(NOCOL(ISO).LT.IMAX)GOTO 1
      IVSREC(ISO)=NOCOL(ISO)
      IF(ICVS.EQ.IMAX)GOTO 2
      IAR=ISUICO(IAR)
      GOTO 3
    2 ICVS=1
      IVSREC(NOIS)=IMAX
    9 KSUI=IVECSU(ICVS)
C     POUR CHAQUE SUIVANT DE NOIS VOIR SI IL FORME AVEC NOIS ET LES AUTRES
C     SUIVANTS DE NOIS RELIES PAR LES ARCS DU GRAPHE UN SOUS GRAPHE COMPLET
      INTMAX=IVSREC(KSUI)
      IARANT=INTMAX
      ICSUIV=0
      ICEGAL=0
      IARSU=INDCOL(KSUI)
    6 ISOSU=LIGN(IARSU)
      ICSUIV=ICSUIV+1
      IARANT=IARANT-1
C     VOIR SI CHAQUE SUIVANT DE NOIS A POUR SUIVANTS NOIS ET LES AUTRES
C     SUIVANTS DE NOIS
      IF(IVSREC(ISOSU).NE.0)GOTO 5
      IF(IARANT.LT.IMAX)GOTO 4
    7 IF(ICSUIV.EQ.INTMAX)GOTO 4
      IARSU=ISUICO(IARSU)
      GOTO 6
    5 ICEGAL=ICEGAL+1
      IARANT=IARANT+1
C     VOIR SI CELA VAUT LA PEINE DE CONTINUER
      IF(ICEGAL.LT.IMAX)GOTO 7
      IF(ICVS.EQ.IMAX)GOTO 8
      ICVS=ICVS+1

      GOTO 9
    8 I=1
    4 ICVS=IMAX
    1 RETURN
      END
```

15 - La routine R4CHKS constate, au sommet x, si l'on peut appliquer la règle 4 de C.K. étape 1, x étant un sommet ayant plus d'arcs adjacents vers l'extérieur, que vers l'intérieur.

IX : sommet auquel on applique R4CHKS

IMAX : nombre de suivants de IX

KLIMAX : nombre de précédents de IX

IS : un suivant de IX

IVECTS : vecteur faisant office de stack pour stocker tous les suivants du sommet IX.

IVEREC : vecteur, qui, à la place i correspondant au sommet i, contient le numéro IA si, dans le graphe, il existe un arc (IX,i) de numéro IA; sinon contient en i un 0

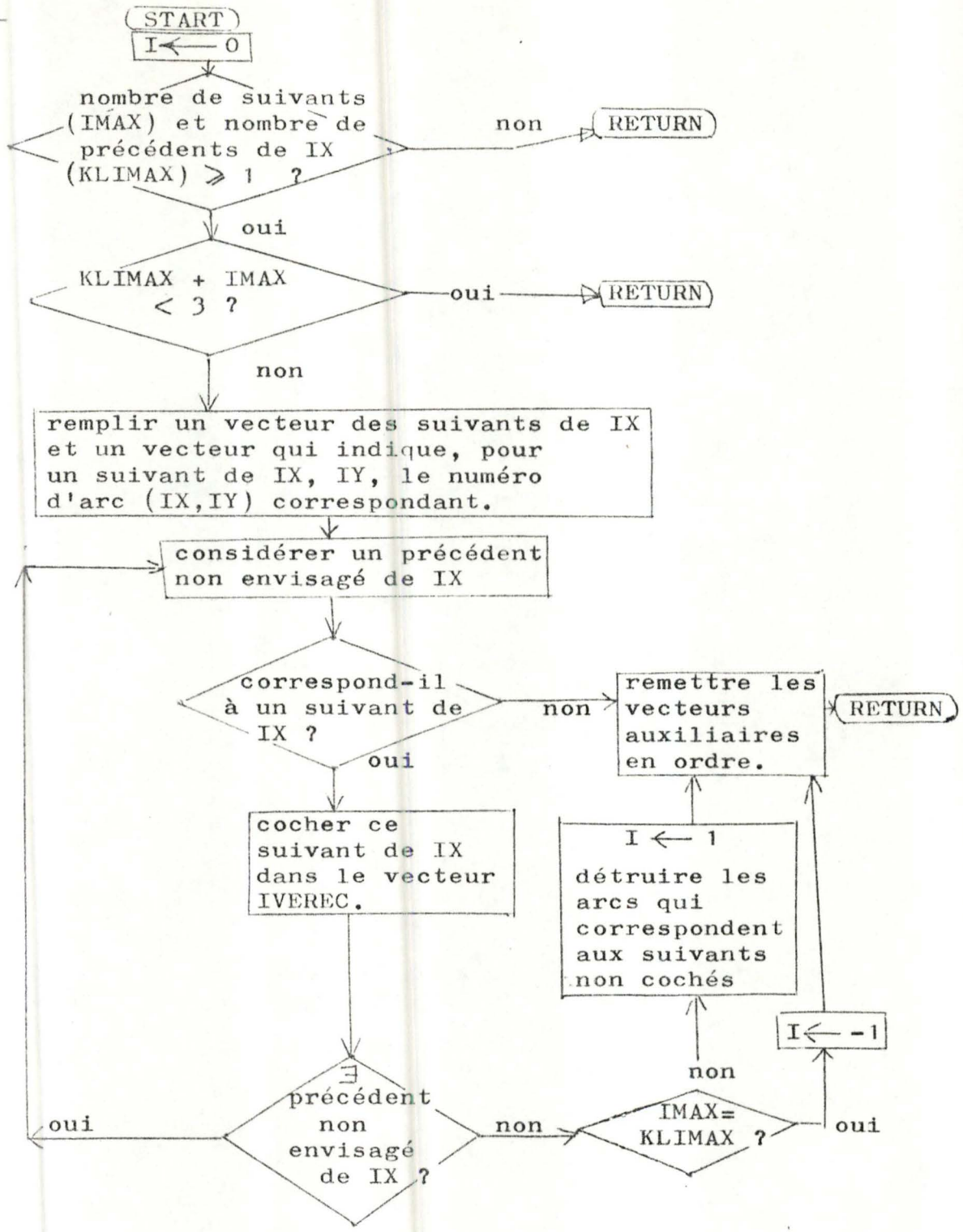
LEDFRE : stack d'arcs libres.

I : variable qui vaut 1 si la règle 4 s'applique au sommet IX; 0 sinon. -1 s'il n'y a pas au sommet IX d'autres arcs que des paires d'arcs formant doublets. Ceci évitera des tests inutiles en vue de l'application du corollaire 3 de Kevorkian.

Il est certain que si le sommet x a plus d'arcs adjacents vers l'intérieur que vers l'extérieur, il suffit d'appliquer R4CHKS en inversant les arguments se rapportant aux colonnes et ceux se rapportant aux lignes de $M'^{T}$ .

Voir organigramme.

R4CHKS


START
I ← 0
nombre de suivants (IMAX) et nombre de précédents de IX (KLIMAX) ≥ 1 ?
non
RETURN
oui
KLIMAX + IMAX < 3 ?
oui
RETURN
non
remplir un vecteur des suivants de IX et un vecteur qui indique, pour un suivant de IX, IY, le numéro d'arc (IX,IY) correspondant.
considérer un précédent non envisagé de IX
correspond-il à un suivant de IX ?
non
remettre les vecteurs auxiliaires en ordre.
RETURN
oui
cocher ce suivant de IX dans le vecteur IVEREC.
I ← 1
détruire les arcs qui correspondent aux suivants non cochés
I ← -1
∃ précédent non envisagé de IX ?
oui
non
IMAX= KLIMAX ?
non
oui

```
      SUBROUTINE R4CHKS(I,IX,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN
     1,NOLIG,INDCOL,INDLIG,IVECTS,IVEREC,NN,MM,LEDFRE)
C
C     REGLE 4 DE C.K.
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IPRECO(MM),ISUICO(MM),IPRELI(MM),ISUILI(MM),ICOL(MM),LIG
     1N(MM),INDCOL(NN),INDLIG(NN),IVECTS(NN),IVEREC(NN),NOCOL(NN),NOLIG(
     2NN)
      DIMENSION LEDFRE(MM)
      LNEDFR=LEDFRE(MM)
      I=0
      ICS=0
      IMAX=NOCOL(IX)
      KLIMAX=NOLIG(IX)
      IF((KLIMAX.LT.1).OR.(IMAX.LT.1).OR.((IMAX+KLIMAX).LT.3))RETURN
      IA=INDCOL(IX)
    2 IS=LIGN(IA)
C     ENREGISTRER LES SUIVANTS DE IX
      ICS=ICS+1
      IVECTS(ICS)=IS
      IVEREC(IS)=IA
      IF(ICS.EQ.IMAX)GOTO 1
      IA=ISUICO(IA)
      GOTO 2
    1 KCS=0
      KALI=INDLIG(IX)
    5 KIS=ICOL(KALI)
      KCS=KCS+1
      IF(IVEREC(KIS).EQ.0)GOTO 3
C     EXISTENCE DE DOUBLET
      IVEREC(KIS)=-IVEREC(KIS)
      IF(KCS.EQ.KLIMAX)GOTO 4
      KALI=ISUILI(KALI)
      GOTO 5
    4 I=1
      IF(IMAX.EQ.KLIMAX)GOTO 10
      DO 6 L=1,IMAX
      LTR=IVECTS(L)
      IF(IVEREC(LTR).LT.0)GOTO 7
C     DETRUIRE LES ARCS QUI NE FONT PAS PARTIE D'UN DOUBLET
      IDELE=IVEREC(LTR)
      CALL DELEAR(IDELE,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN,NOLI
     1G,INDCOL,INDLIG,NN,MM)
      LNEDFR=LNEDFR+1
      LEDFRE(LNEDFR)=IDELE
C     RE-INITIALISER
    7 IVEREC(LTR)=0
    6 IVECTS(L)=0
      LEDFRE(MM)=LNEDFR

      RETURN
   10 I=-1
    3 DO 8 L=1,IMAX
      LTR=IVECTS(L)
      IVEREC(LTR)=0
    8 IVECTS(L)=0
      RETURN
      END
```

16- La routine KEVCO3 essaie d'appliquer le corollaire 3 de Kevorkian au sommet x, et si le corollaire s'applique, détruit le sommet"en face" de x ( voir configuration testée dans le corollaire 3 de Kevorkian ch.II étape 1 de C.K. )

IX : sommet auquel on tente d'appliquer KEVCO3

LEDFRE : stack d'arcs libres

KONTR : variable qui prend la valeur 1 si la routine KEVCO3 s'applique au sommet IX.

I1,I2,I3 : respectivement les 3 suivants de IX ( se rappeler la configuration que repère le corollaire 3 de Kevorkian ch.II étape 1 de C.K. )

NOSU1,NOSU2,NOSU3 : le nombre de suivants de respectivement I1,I2,I3.

ITEM : variables qui permettent de détecter si une
L3 : situation a déjà été testée parmi les 3 situations
I : suivantes - I1 à supprimer
- I2 à supprimer
- I3 à supprimer

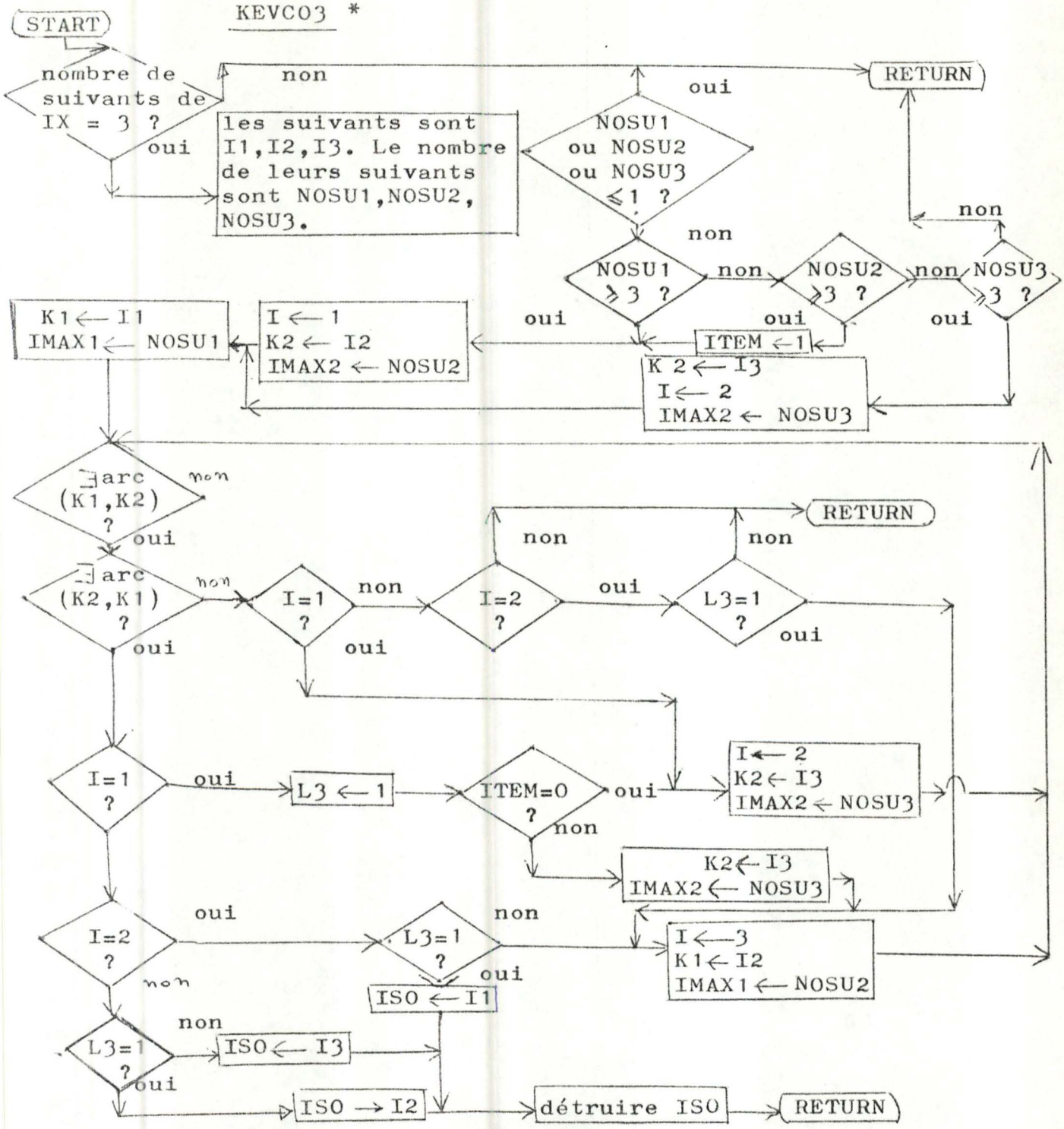
K1,K2 : variables qui facilitent des permutations de noms de sommets

IMAX1,IMAX2,IMAX3 : variables qui facilitent des permutations du rôle des variables NOSU1,NOSU2,NOSU3

ISO : sommet à détruire.

voir l'organigramme après le"listing" de la routine.

KEVCO3 *




Note : on essaiera d'appliquer KEVCO3 à un sommet x si l'application de R4CHKS a permis à I de prendre la valeur +1 ou -1
I=0 correspond à une situation à laquelle on ne pourrait de toute façon pas appliquer KEVCO3.

```
      SUBROUTINE KEVCO3(IX,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCOL,
     1INDLIG,NOCOL,NOLIG,NN,MM,LEDFRE,KONTR)
C
C     COROLLAIRE 3 DE KEVORKIAN
C
      DIMENSION LEDFRE(MM)
      DIMENSION IPRECO(MM),ISUICO(MM),IPRELI(MM),ISUILI(MM),ICOL(MM),LIG
     1N(MM),INDCOL(NN),INDLIG(NN),NOCOL(NN),NOLIG(NN)
      IMPLICIT INTEGER*2(I-N)
      KONTR=0
      IF(NOCOL(IX).NE.3)RETURN
C     INITIALISER LES 3 SUIVANTS DE IX
      ITRA=INDLIG(IX)
      I1=ICOL(ITRA)
      ITRA=ISUILI(ITRA)
      I2=ICOL(ITRA)
      ITRA=ISUILI(ITRA)
      I3=ICOL(ITRA)
      NOSU1=NOCOL(I1)
      NOSU2=NOCOL(I2)
      NOSU3=NOCOL(I3)
      IF((NOSU1.EQ.1).OR.(NOSU2.EQ.1).OR.(NOSU3.EQ.1))RETURN
      ITEM=0
      IF(NOSU1.GE.3)GOTO 1
      IF(NOSU2.GE.3)GOTO 2
      IF(NOSU3.LT.3)RETURN
      K2=I3
      IMAX2=NOSU3
      I=2
      GOTO 3
    2 ITEM=1
    1 I=1
      K2=I2
      IMAX2=NOSU2
    3 K1=I1
      IMAX1=NOSU1
      L3=0
   14 ICS=0
      ISTER=INDCOL(K1)
    6 ICS=ICS+1
C     TESTER L"EXISTENCE DE DOUBLETS
      IF(LIGN(ISTER).EQ.K2)GOTO 4
      IF((LIGN(ISTER).GT.K2).OR.(ICS.EQ.IMAX1))GOTO 5
      ISTER=ISUICO(ISTER)
      GOTO 6
    4 ICS=0
      ISTER=INDCOL(K2)
    8 ICS=ICS+1
C     TESTER L"EXISTENCE DE DOUBLETS
      IF(LIGN(ISTER).EQ.K1)GOTO 7
```

```
51          IF((LIGN(ISTER).GT.K1).OR.(ICS.EQ.IMAX2))GOTO 5
52          ISTER=ISUICO(ISTER)
53          GOTO 8
54  C
55  C       TOUTES DES PERMUTATIONS DE VARIABLES POUR REPERER LES DIFFERENTES
56  C       CONFIGURATIONS POSSIBLES PERMISES PAR LE COROLLAIRE 3
57  C
58        7 IF(I.EQ.1)GOTO 9
59          IF(I.EQ.2)GOTO 10
60          IF(L3.EQ.1)GOTO 11
61          ISO=I3
62          GOTO 12
63       11 ISO=I2
64          GOTO 12
65       10 IF(L3.EQ.1)GOTO 13
66       16 I=3
67          K1=I2
68          IMAX1=NOSU2
69          GOTO 14
70        9 L3=1
71          IF(ITEM.EQ.0)GOTO 15
72          K2=I3
73          IMAX2=NOSU3
74          GOTO 16
75       15 I=2
76          K2=I3
77          IMAX2=NOSU3
78          GOTO 14
79        5 IF(I.EQ.1)GOTO 15
80          IF(I.NE.2)RETURN
81          IF(L3.NE.1)RETURN
82          GOTO 16
83       13 ISO=I1
84       12 CONTINUE
85          KONTR=1
86  C       ON PEUT DETRUIRE LE SOMMET ISO
87          CALL DELESO(ISO,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCOL,INDLI
88         1G,NOCOL,NOLIG,NN,MM,LEDFRE)
89          RETURN
90          END
```

17- La routine KEVCO2 essaie d'appliquer le corollaire 2 de KEVORKIAN ( ch.II étape 1 de C.K. ) à toute paire de sommets qui n'ont pour arcs adjacents que des paires d'arcs formant doublet. Ce qui signifie que l'on appliquera KEVCO2 après application de R4CHKS à chaque sommet du graphe.

J : variable qui prend la valeur 1 si KEVCO2 modifie le graphe.

LESOCO : vecteur auxiliaire qui permet de stocker les sommets auxquels on a appliqué avec succès R4CHKS.

LEVECO : vecteur auxiliaire qui permet de stocker des sommets auxquels le corollaire 2 de Kevorkian est susceptible de s'appliquer.

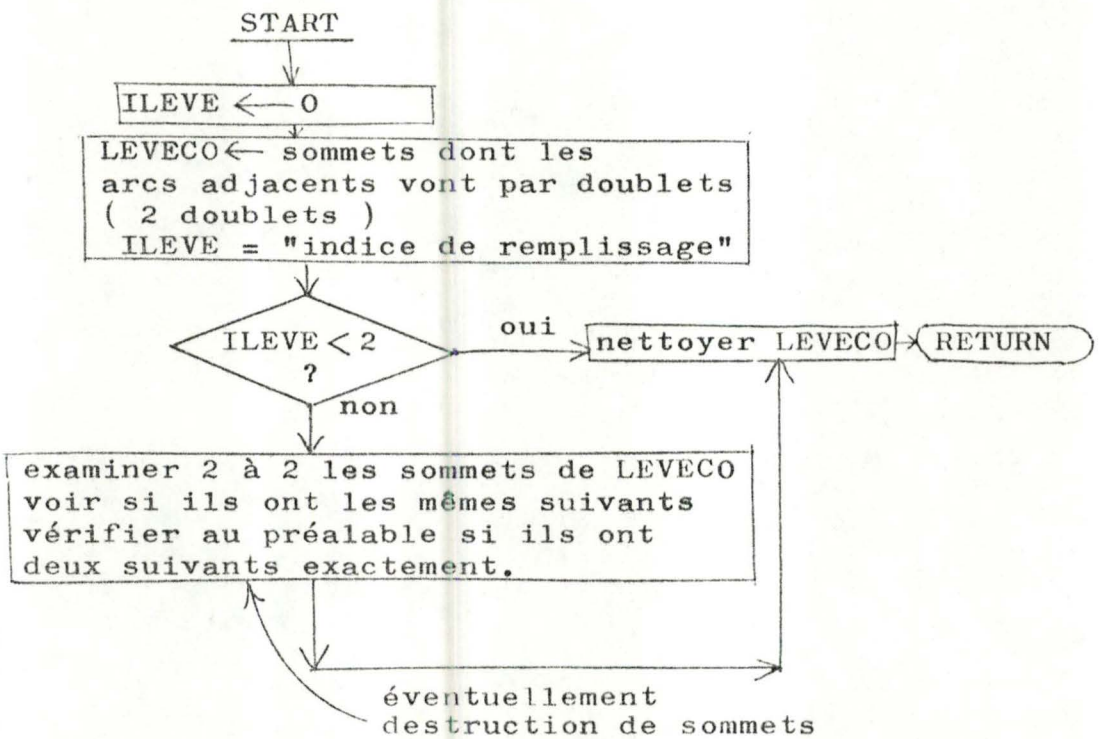
ILESO : nombre de sommets stockés dans LESOCO

ILEVE : nombre de sommets stockés dans LEVECO

ISCOU : paire de sommets à laquelle on tente

ISO : d'appliquer le corollaire 2 de Kevorkian.

KEVCO2



```
      SUBROUTINE KEVCO2(IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCOL,IND
     1LIG,NOCOL,NOLIG,NN,MM,LESOCO,LEVECO,MES,INDMES,LEDFRE,J)
C
C     COROLLAIRE 2 DE KEVORKIAN
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IPRECO(MM),ISUICO(MM),IPRELI(MM),ISUILI(MM),ICOL(MM),LIG
     1N(MM),INDCOL(NN),INDLIG(NN),NOCOL(NN),NOLIG(NN),LESOCO(NN),LEVECO(
     2NN),MES(NN)
      DIMENSION LEDFRE(MM)
      J=0
      ILEVE=0
C      LESOCO = VECTEUR QUI CONTIENT LES SOMMETS AYANT 2 PRECEDENTS
      ILESO=LESOCO(NN)
      DO 1 I=1,ILESO
      ISO=LESOCO(I)
      IF(NOLIG(ISO).NE.2)GOTO 1
      ILEVE=ILEVE+1
      LEVECO(ILEVE)=ISO
    1 CONTINUE
      IF(ILEVE.LT.2)GOTO 2
      ILM1=ILEVE-1
C     ENVISAGER TOUTES LES PAIRES DE SOMMETS AYANT 2 PRECEDENTS
      DO 3 I1=1,ILM1
      ISO=LEVECO(I1)
      IF(ISO.EQ.0)GOTO 3
      IF(NOLIG(ISO).EQ.2)GOTO 4
      LEVECO(I1)=0
      GOTO 3
C     RE-VERIFIER SI LES SOMMETS ONT 2 PRECEDENTS
    4 IVERI1=INDLIG(ISO)
      KTR=I1+1
      DO 5 I2=KTR,ILEVE
      DO 5 I2=2,ILEVE
      ISCOU=LEVECO(I2)
      IF(ISCOU.EQ.0)GOTO 5
      IF(NOLIG(ISCOU).EQ.2)GOTO 6
      LEVECO(I2)=0
      GOTO 5
    6 IVERI2=INDLIG(ISCOU)
      IF(ICOL(IVERI1).NE.ICOL(IVERI2))GOTO 5
      IVERI1=ISUILI(IVERI1)
      IVERI2=ISUILI(IVERI2)
      IF(ICOL(IVERI1).NE.ICOL(IVERI2))GOTO 5
      IVERI1=INDLIG(ISO)
C     SITUATION DECRITE DANS LE COROLLAIRE 2
C     ON PEUT DETRUIRE LE SOMMET IVERI2
      DO 7 L=1,2
      IVERI2=ICOL(IVERI1)
      CALL DELESO(IVERI2,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCOL,IN
     1DLIG,NOCOL,NOLIG,NN,MM,LEDFRE)
      J=1
      INDMES=INDMES+1
      MES(INDMES)=IVERI2
    7 IVERI1=ISUILI(IVERI1)
C     NETTOYER LEVECO
      LEVECO(I1)=0
      LEVECO(I2)=0
    5 CONTINUE
    3 CONTINUE
    2 DO 8 L=1,ILEVE
    8 LEVECO(L)=0
      RETURN
      END
```

18- La routine R4CETK rempli essentiellement 3 rôles

A. essaie d'appliquer la règle 4 de C.K. ( R4CHKS ) à chaque sommet du graphe.
   note : si au sommet x, le nombre d'arcs incidents vers l'intérieur est supérieur au nombre d'arcs incidents vers l'extérieur, on échange les arguments de R4CHKS se rapportant aux lignes et ceux se rapportant aux colonnes ( utilisation de la propriété de symétrie attachée à l'emploi des arguments se rapportant aux lignes et aux colonnes de la routine R4CHKS )

B. essayer d'appliquer le corollaire 3 de Kevorkian à certains sommets déterminés après application de R4CHKS.

C. essayer d'appliquer le corollaire 2 de Kevorkian en utilisant l'information donnée par l'application de R4CHKS et KEVCO3 à tous ou à certains sommets du graphe.

K : variable qui prend la valeur 1 ,si le graphe a été modifié par l'application de R4CETK; 0 sinon.

LK : variable qui prend la valeur 1 si l'application de R4CHKS à un sommet x permet à la variable I de R4CHKS de prendre la valeur -1; 0 sinon.

LESOCO : vecteur-stack qui stocke les sommets susceptibles de permettre l'application de la règle 4 de C.K.

INDIC : "indice de remplissage"de LESOCO

ISOADM : vecteur-stack qui stocke des sommets supposés avoir des arcs incidents ( sommets auxquels une règle peut éventuellement s'appliquer.)

NOSO : "indice de remplissage" de ISOADM

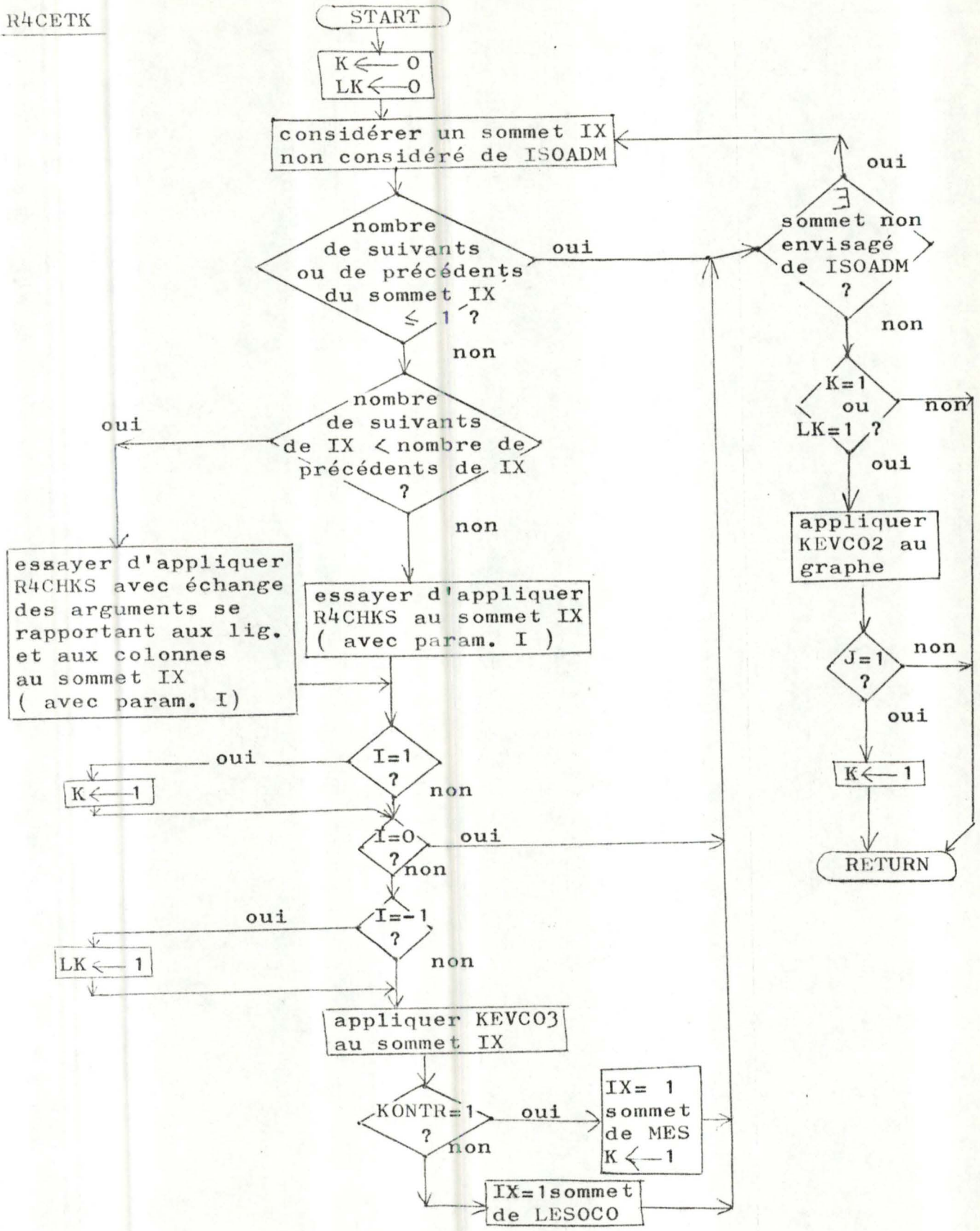
MES : vecteur-stack des sommets de l'ensemble minimum essentiel.

INDMES : "indice de remplissage" du vecteur MES.

LEDFRE : stack des arcs libres.

Voir organigramme.

R4CETK


START
K ← 0
LK ← 0
considérer un sommet IX non considéré de ISOADM
nombre de suivants ou de précédents du sommet IX ≤ 1 ?
oui
non
∃ sommet non envisagé de ISOADM ?
oui
non
K=1 ou LK=1 ?
oui
non
appliquer KEVCO2 au graphe
J=1 ?
oui
non
K ← 1
RETURN
nombre de suivants de IX < nombre de précédents de IX ?
oui
non
essayer d'appliquer R4CHKS avec échange des arguments se rapportant aux lig. et aux colonnes au sommet IX ( avec param. I)
essayer d'appliquer R4CHKS au sommet IX ( avec param. I )
I=1 ?
oui
K ← 1
non
I=0 ?
oui
non
I=-1 ?
oui
LK ← 1
non
appliquer KEVCO3 au sommet IX
KONTR=1 ?
oui
IX= 1 sommet de MES
K ← 1
non
IX=1sommet de LESOCO

```
      SUBROUTINE R4CETK(K,IX,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN
     1,NOLIG,INDCOL,INDLIG,IVECTS,IVEREC,NN,MM,ISOADM,NOSO,LESOCO,MES,IN
     2DMES,LEDFRE)
C
C     COORDONNER LA REGLE 4 DE C.K. ET LES COROLLAIRES DE KEVORKIAN 2-3
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IPRECO(MM),ISUICO(MM),IPRELI(MM),ISUILI(MM),ICOL(MM),NOC
     1OL(NN),LIGN(MM),NOLIG(NN),INDCOL(NN),INDLIG(NN),IVECTS(NN),IVEREC(
     1NN)
      DIMENSION ISOADM(NN),LESOCO(NN),MES(NN)
      DIMENSION LEDFRE(MM)
      INDIC=0
      K=0
      LK=0
      DO 1 L=1,NOSO
      IX=ISOADM(L)
      I=0
      IF((NOCOL(IX).LE.1).OR.(NOLIG(IX).LE.1))GOTO 1
      IF(NOCOL(IX).LT.NOLIG(IX))GOTO 2
C     ESSAYER TOUT D"ABORD D"APPLIQUER LA REGLE 4 DE C.K.
      CALL R4CHKS(I,IX,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN,NOLIG
     1,INDCOL,INDLIG,IVECTS,IVEREC,NN,MM,LEDFRE)
      IF(I.NE.1)GOTO 3
      K=1
      GOTO 3
    2 CALL R4CHKS(I,IX,IPRELI,ISUILI,IPRECO,ISUICO,LIGN,NOLIG,ICOL,NOCOL
     1,INDLIG,INDCOL,IVECTS,IVEREC,NN,MM,LEDFRE)
      IF(I.NE.1)GOTO 3
      K=1
    3 IF(I.EQ.0)GOTO 1
C     VOIR SI LA REGLE 4 S"APPLIQUERAIT SI IL Y AVAIT DES ARCS A DETRUIRE
      IF(I.EQ.-1)LK=1
C     ESSAYER D"APPLIQUER LE COROLLAIRE 3 DE KEVORKIAN
      CALL KEVCO3(IX,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCOL,INDLIG
     1,NOCOL,NOLIG,NN,MM,LEDFRE,KONTR)
C     VOIR SI LE COROLLAIRE 3 A PERMIS DE DTRUIRE DES ARCS
      IF(KONTR.EQ.0)GOTO 30
      INDMES=INDMES+1
      MES(INDMES)=IX
      K=1
   30 INDIC=INDIC+1
      LESOCO(INDIC)=IX
    1 CONTINUE
      LESOCO(NN)=INDIC
      J=0
C     VOIR SI CELA VAUT LA PEINE D"ESSAYER D"APPLIQUER LE COROLLAIRE 2
      IF((K.NE.1).AND.(LK.NE.1))RETURN
      CALL KEVCO2(IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCOL,INDLIG,NO
     1COL,NOLIG,NN,MM,LESOCO,IVECTS,MES,INDMES,LEDFRE,J)

      IF(J.EQ.1)K=1
      RETURN
      END
```

19- La routine R5CEKP essaie d'appliquer la règle 5 de l'algorithme de C.K. ( ch. II étape 1) à chaque arc incident vers l'intérieur à un sommet IX.

IX : sommet auquel on tente d'appliquer la règle 5.

I : variable qui prend la valeur 1 si la routine R5CEKP détruit au moins un arc du graphe.

IPRMAX : nombre de précédents du sommet IX.

ISUMAX : nombre de suivants du sommet IX

ICOPR : compteur d'arcs incidents à IX vers l'intérieur.

IY : un précédent courant de IX

ITEDO:compteurs de suivants envisagés de IX

ITER : arc incident à IX vers l'intérieur courant.

ICOSUY : compteurs auxiliaires de suivants envisagés

ICOSUX : de IY et IX

KCSY : suivants courants de IY et IX

KCSX :

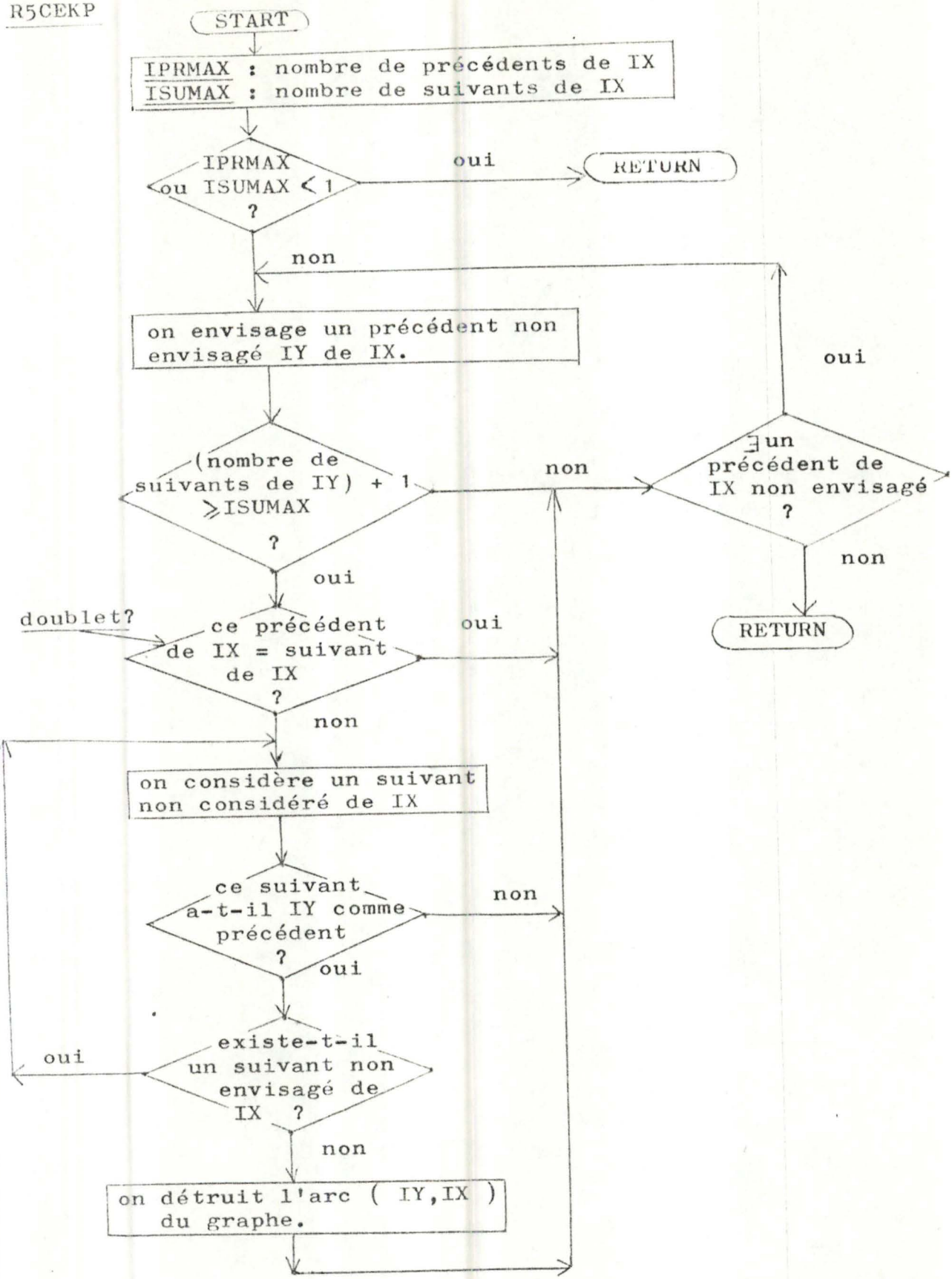
LEDFRE : vecteur-stack d'arcs libres.

IAIY : arc(IY,IX)

Note : essayer d'appliquer R5CEKP à chaque arc incident vers l'extérieur à IX correspond à échanger, dans les arguments de R5CEKP, les arguments correspondant aux lignes et ceux correspondant aux colonnes.

Voir organigramme.

R5CEKP


START
IPRMAX : nombre de précédents de IX
ISUMAX : nombre de suivants de IX
IPRMAX ou ISUMAX < 1 ?
oui
RETURN
non
on envisage un précédent non envisagé IY de IX.
(nombre de suivants de IY) + 1 ⩾ ISUMAX ?
non
∃ un précédent de IX non envisagé ?
oui
non
RETURN
oui
doublet?
ce précédent de IX = suivant de IX ?
oui
non
on considère un suivant non considéré de IX
ce suivant a-t-il IY comme précédent ?
non
oui
existe-t-il un suivant non envisagé de IX ?
oui
non
on détruit l'arc ( IY,IX ) du graphe.

```
      SUBROUTINE R5CEKP(I,IX,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN
     1,NOLIG,INDCOL,INDLIG,NN,MM,LEDFRE)
C
C     REGLE 5 DE C.K.
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION IPRECO(MM),ISUICO(MM),IPRELI(MM),ISUILI(MM),ICOL(MM),NOC
     1OL(NN),LIGN(MM),NOLIG(NN),INDCOL(NN),INDLIG(NN)
      DIMENSION LEDFRE(MM)
      I=0
      IPRMAX=NOLIG(IX)
      ISUMAX=NOCOL(IX)
      IF((IPRMAX.LE.1).OR.(ISUMAX.LE.1))RETURN
      LNEDFR=LEDFRE(MM)
      ICOPR=1
      IAIY=INDLIG(IX)
C     TRAITER UN PRECEDENT DE IX,SOIT IY
    3 IY=ICOL(IAIY)
      ISUMAY=NOCOL(IY)
      IF(ISUMAX.GT.ISUMAY+1)GOTO 1
      ITEDO=0
      ITER=INDCOL(IX)
C     VOIR SI IL EXISTE UN DOUBLET (IX,IY)
    3 ITEDO=ITEDO+1
      IF(LIGN(ITER).EQ.IY)GOTO 1
      IF((LIGN(ITER).GT.IY).OR.(ITEDO.EQ.ISUMAX))GOTO 2
      ITER=ISUICO(ITER)
      GOTO 3
    2 ICOSUY=1
      ISUY=INDCOL(IY)
      KCSY=LIGN(ISUY)
      ICOSUX=1
      ISUX=INDCOL(IX)
C     POUR CHAQUE SUIVANT DE IX, VOIR SI CE SUIVANT A COMME PRECEDENT IY
    7 KCSX=LIGN(ISUX)
    5 IF(KCSY.EQ.KCSX)GOTO 4
      IF((KCSY.GT.KCSX).OR.(ICOSUY.EQ.ISUMAY))GOTO 1
      ISUY=ISUICO(ISUY)
      KCSY=LIGN(ISUY)
      ICOSUY=ICOSUY+1
      GOTO 5
    4 IF(ICOSUX.EQ.ISUMAX)GOTO 6
      ISUX=ISUICO(ISUX)
      ICOSUX=ICOSUX+1
      IF(ICOSUY.EQ.ISUMAY)GOTO 1
      ISUY=ISUICO(ISUY)
      KCSY=LIGN(ISUY)
      ICOSUY=ICOSUY+1
      GOTO 7
    6 I=1

C     ON PEUT DETRUIRE L'ARC (IY,IX)
      CALL DELEAR(IAIY,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN,NOLIG
     1,INDCOL,INDLIG,NN,MM)
      LNEDFR=LNEDFR+1
      LEDFRE(LNEDFR)=IAIY
    1 IF(ICOPR.EQ.IPRMAX)GOTO 10
      ICOPR=ICOPR+1
      IAIY=ISUILI(IAIY)
      GOTO 8
   10 LEDFRE(MM)=LNEDFR
      RETURN
      END
```

20- La routine MISAJO sert à mettre à jour la liste des sommets de ISOADM, qui est une liste des sommets "admissibles", en ce sens qu'ils sont susceptibles d'avoir encore des arcs incidents. Les sommets isolés ne nous intéressent pas dans l'application des règles de simplification à un graphe.
Après application d'une routine qui a permis de détruire un sommet IX, on "retire" le sommet IX du stack ISOADM en "décalant" tous les sommets suivants de IX dans le stack de 1 position vers la gauche. Le manque de temps nous a amené à ne pas donner à cette liste une structure autre que la structure de stack qui s'adapte très peu à la suppression d'un sommet quelconque de la liste. Il est certain qu'une structure de listes enchainées s'impose par ses possibilités de "balayages" et de "suppression" ( voir KNUTH chapitre II ).
Note : il peut y avoir, après certaines applications de routines de simplification(une des règles ou corollaires du ch.II étape 1 ) à certains sommets, dans ISOADM, des sommets isolés qui ne soient pas supprimés du stack par application de MISAJO; en effet, MISAJO n'est appliquée à un sommet IX de ISOADM que quand on essaie d'appliquer une routine de simplification successivement à un sommet de ISOADM à l'aide d'un"balayage" faisant varier un indice de 1 jusque NOSO ( "indice de remplissage" de ISOADM), plus précisément à l'aide d'une "boucle DO"; MISAJO met alors à jour l'indice de boucle et NOSO et"décale"une partie de la liste ISOADM vers la gauche.
Pour cette routine, nous ne donnons pas d'organigramme.

```
      SUBROUTINE MISAJO(ISOADM,M,NOSO,NN)
C
C     MISE A JOUR DE ISOADM
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION ISOADM(NN)
      IF(M.EQ.(NOSO+1))RETURN
      MP1=M+1
      DO 1 L=M,NOSO
      ISOADM(L)=ISOADM(MP1)
    1 MP1=MP1+1
      M=M-1
      RETURN
      END
```

21- La routine STEP1 (étape 1 ) organise l'application des routines de simplification du graphe aux sommets du graphe.
Elle permet d'effectuer, à l'aide des routines R1CETK, R2CETK, R3CETK, R4CETK, R5CEKP, MISAJO, DELESO, l'étape 1 de l'algorithme de C.K. au sous-graphe d'une composante fortement connexe. ( cfr. chapitre II étape 1 ).

NSCOMP : nombre de sommets initial de la C.F.C. que l'on veut analyser.

MES : vecteur-stack qui stocke les sommets de l'ensemble minimum essentiel.

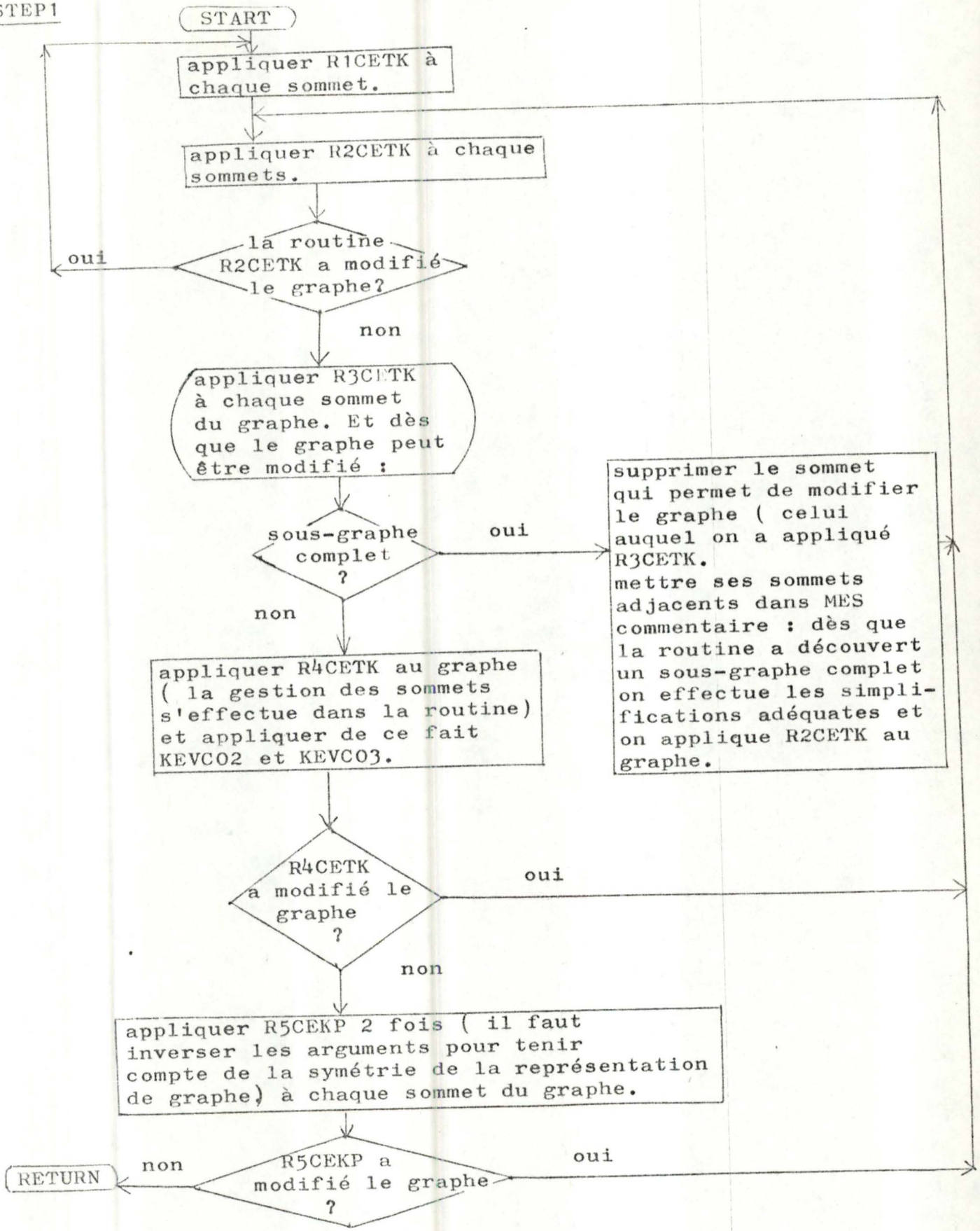
IACT : Nom d'un sommet auquel on va essayer d'appliquer une règle de simplification.

IVECO1 : vecteurs auxiliaires utilisés dans les routines
IVECO2 : R3CETK et R4CETK.

LESOCO : vecteur auxiliaire utilisé par R4CETK.

Voir organigramme à la page suivante.

STEP1


START
appliquer R1CETK à chaque sommet.
appliquer R2CETK à chaque sommets.
la routine R2CETK a modifié le graphe?
oui
non
appliquer R3CETK à chaque sommet du graphe. Et dès que le graphe peut être modifié :
sous-graphe complet ?
oui
supprimer le sommet qui permet de modifier le graphe ( celui auquel on a appliqué R3CETK.
mettre ses sommets adjacents dans MES commentaire : dès que la routine a découvert un sous-graphe complet on effectue les simplifications adéquates et on applique R2CETK au graphe.
non
appliquer R4CETK au graphe ( la gestion des sommets s'effectue dans la routine) et appliquer de ce fait KEVC02 et KEVC03.
R4CETK a modifié le graphe ?
oui
non
appliquer R5CEKP 2 fois ( il faut inverser les arguments pour tenir compte de la symétrie de la représentation de graphe) à chaque sommet du graphe.
R5CEKP a modifié le graphe ?
oui
non
RETURN

```
      SUBROUTINE STEP1(NSCOMP,INDCOL,INDLIG,IPRECO,IPRELI,ISUICO,ISUILI,
     1NOCOL,NOLIG,IVECO1,IVECO2,MES,INDMES,ISOADM,NOSO,NN,MM,ICOL,LIGN,L
     2ESOCO,LEDFRE)
C
C     CETTE SOUROUTINE EFFECTUE L"ETAPE 1 DE L"ALGORITHME DE CHEUNG ET KUH
C
      DIMENSION LEDFRE(MM)
      IMPLICIT INTEGER*2(I-N)
      DIMENSION INDCOL(NN),INDLIG(NN),IPRECO(MM),IPRELI(MM),ISUICO(MM),I
     1SUILI(MM),NOCOL(NN),NOLIG(NN),IVECO1(NN),IVECO2(NN),MES(NN),ISOADM
     2(NN),ICOL(MM),LIGN(MM)
      DIMENSION LESOCO(NN)
      INDMES=0
      NOSO=NSCOMP
      GOTO 7
    4 CONTINUE
      DO 2 M=1,NOSO
      IACT=ISOADM(M)
C
C     ESSAYER D"APPLIQUER LA REGLE 1
C
      CALL R1CETK(I,IACT,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCOL,IN
     1DLIG,NOCOL,NOLIG,MES,INDMES,NOSO,NN,MM,LEDFRE)
      IF(I.EQ.0)GOTO 2
      CALL MISAJO(ISOADM,M,NOSO,NN)
    2 CONTINUE
      IF(NOSO.EQ.0)RETURN
    7 J=0
      DO 3 M=1,NOSO
      IACT=ISOADM(M)
C
C     ESSAYER D"APPLIQUER LA REGLE 2
C
      CALL R2CETK(I,IACT,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCOL,IN
     1DLIG,NOCOL,NOLIG,NOSO,NN,MM,LEDFRE)
      IF(I.EQ.0)GOTO 3
      CALL MISAJO(ISOADM,M,NOSO,NN)
      J=1
    3 CONTINUE
      IF(J.EQ.1)GOTO 4
    6 CONTINUE
      DO 5 M=1,NOSO
      IACT=ISOADM(M)
C
C     ESSAYER D"APPLIQUER LA REGLE 3
C
      CALL R3CETK(I,IACT,ISUICO,LIGN,INDCOL,NOCOL,NOLIG,IVECO1,IVECO2,NO
     *1SO,NN,MM,ICVS)
C     DESTRUCTION DES SOMMETS ADJACENTS A IACT
      IF(I.EQ.0)GOTO 11
```

```
      CALL DELESO(IACT,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCOL,INDL
     1IG,NOCOL,NOLIG,NN,MM,LEDFRE)
      NOSO=NOSO-1
C     SUPPRIMER DE ISOADM UN SOMMET DETRUIT
      CALL MISAJO(ISOADM,N,NOSO,NN)
      IA=1
      IB=IVECO1(1)
      DO 9 NA=1,NOSO
      IF(IB.NE.ISOADM(NA))GOTO 9
      CALL DELESO(IB,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCOL,INDLIG
     1,NOCOL,NOLIG,NN,MM,LEDFRE)
      NOSO=NOSO-1
      CALL MISAJO(ISOADM,NA,NOSO,NN)
      IVECO2(IB)=0
      IVECO1(IA)=0
      INDMES=INDMES+1
      MES(INDMES)=IB
      IF(IA.EQ.ICVS)GOTO 11
      IA=IA+1
      IB=IVECO1(IA)
    9 CONTINUE
   11 DO 10 L=1,ICVS
      IT=IVECO1(L)
      IVECO2(IT)=0
      IVECO2(IACT)=0
   10 IVECO1(L)=0
      IF(I.EQ.1)GOTO 7
    5 CONTINUE
      K=0
C
C     ESSAYER D"APPLIQUER LA REGLE 4
C
      CALL R4CETK(K,IACT,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN,NOL
     1IG,INDCOL,INDLIG,IVECO1,IVECO2,NN,MM,ISOADM,NOSO,LESOCO,MES,INDMES
     2,LEDFRE)
      IF(K.EQ.1)GOTO 7
      L=0
      DO 8 M=1,NOSO
      IACT=ISOADM(M)
C
C     ESSAYER D"APPLIQUER LA REGLE 5
C
      CALL R5CEKP(I,IACT,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN,NOL
     1IG,INDCOL,INDLIG,NN,MM,LEDFRE)
      IF(I.EQ.1)L=1
      CALL R5CEKP(I,IACT,IPRELI,ISUILI,IPRECO,ISUICO,LIGN,NOLIG,ICOL,NOC
     1OL,INDLIG,INDCOL,NN,MM,LEDFRE)
      IF(I.EQ.1)L=1
    8 CONTINUE
      IF(L.EQ.1)GOTO 7

      RETURN
      END
```

22- La routine NET ré-initialise une colonne de la matrice COV ( voir IV.A.2.b.2.) à la valeur booléenne 0; met à jour le vecteur-stack des colonnes libres de COV; annule la position du vecteur de lien des étiquettes ( IPLUS ) d'un arc, qui correspond à la colonne ré-initialisée.

TABLE : autre nom pour la matrice COV

IPLUS : vecteur qui sert à "enchainer" les étiquettes.

ICOFRE : vecteur-stack des colonnes libres de la matrice COV

INCFRE : "indice de remplissage" de ICOFRE.

J : le numéro de la colonne que l'on ré-initialise.

ISOMCO : nombre de lignes utilisées de la matrice COV. ( ce nombre correspond au nombre de sommets non isolés restant dans le graphe au début de l'application de l'étape 2 de l'algorithme de C.K. )

Pour cette routine nous ne donnons pas d'organigramme.

```
      SUBROUTINE NET(TABLE,ISOMCO,IPLUS,J,ICOFRE,III,JJJ)
C
C     SOUROUTINE DE NETTOYAGE D"UNE COLONNE DE TABLE DE COUVERTURE
C
      IMPLICIT INTEGER*2(I-N)
      IMPLICIT LOGICAL*1(T)
      DIMENSION TABLE(JJJ,III)
      DIMENSION IPLUS(III),ICOFRE(III)
      DO 1 I=1,ISOMCO
    1 TABLE(I,J)=.FALSE.
      INCFRE=ICOFRE(III)+1
      ICOFRE(INCFRE)=J
      ICOFRE(III)=INCFRE
      IPLUS(J)=0
      RETURN
      END
```

23- La routine SOMBOO effectue la somme booléenne de deux colonnes de la matrice TABLE;"charge" le résultat dans une troisième colonne de TABLE; calcule le nombre de 1 booléens de cette nouvelle colonne; "charge" ce nombre à la place de NOCROS ( voir ch.IV.A.2.b.2. ) qui correspond à cette nouvelle colonne.

I,J : les numéros de colonne dont on effectue la somme booléenne.

IJ : le numéro de colonne qui contiendra le résultat de la somme booléenne des colonnes de numéro I et J.

NLI : autre nom de ISOMCO, nombre de lignes occupées de la matrice TABLE.

NOCROS : vecteur qui contient à la place I, le nombre de 1 booléens de la colonne de TABLE qui porte le numéro I

Pour cette routine nous ne donnons pas d'organigramme.

```
      SUBROUTINE SOMBOO(TABLE,I,J,IJ,NOCROS,NLI,III,JJJ)
C
C     SOUROUTINE QUI EFFECTUE LA SOMME BOOLEENNE DE DEUX COLONNES DE TABLE
C     DE COUVERTURE
C
      IMPLICIT INTEGER*2(I-N)
      IMPLICIT LOGICAL*1(T)
      DIMENSION TABLE(JJJ,III)
      DIMENSION NOCROS(III)
      NOCROS(IJ)=0
      DO 1 K=1,NLI
      TABLE(K,IJ)=TABLE(K,I).OR.TABLE(K,J)
      IF(TABLE(K,IJ))NOCROS(IJ)=NOCROS(IJ)+1
    1 CONTINUE
      RETURN
      END
```

24- La routine R6CETK essaie d'appliquer à un sommet IX la règle 6 de l'algorithme de C.K. étape 2 ( voir Ch.II )

LEDFRE : vecteur-stack des arcs libres.

COVERT : autre nom de la matrice COV

ICIPER : vecteur-stack qui recense les colonnes de la matrice COVERT qui correspondent à des circuits pertinents détectés.

IGRNTR : vecteur qui établi la correspondance entre le numéro d'un arc et le numéro d'une de ses étiquettes.

KMAX : nombre de suivants de IX

KPRMAX : nombre de précédents de IX

IPERCI : "indice de remplissage" de ICIPER.

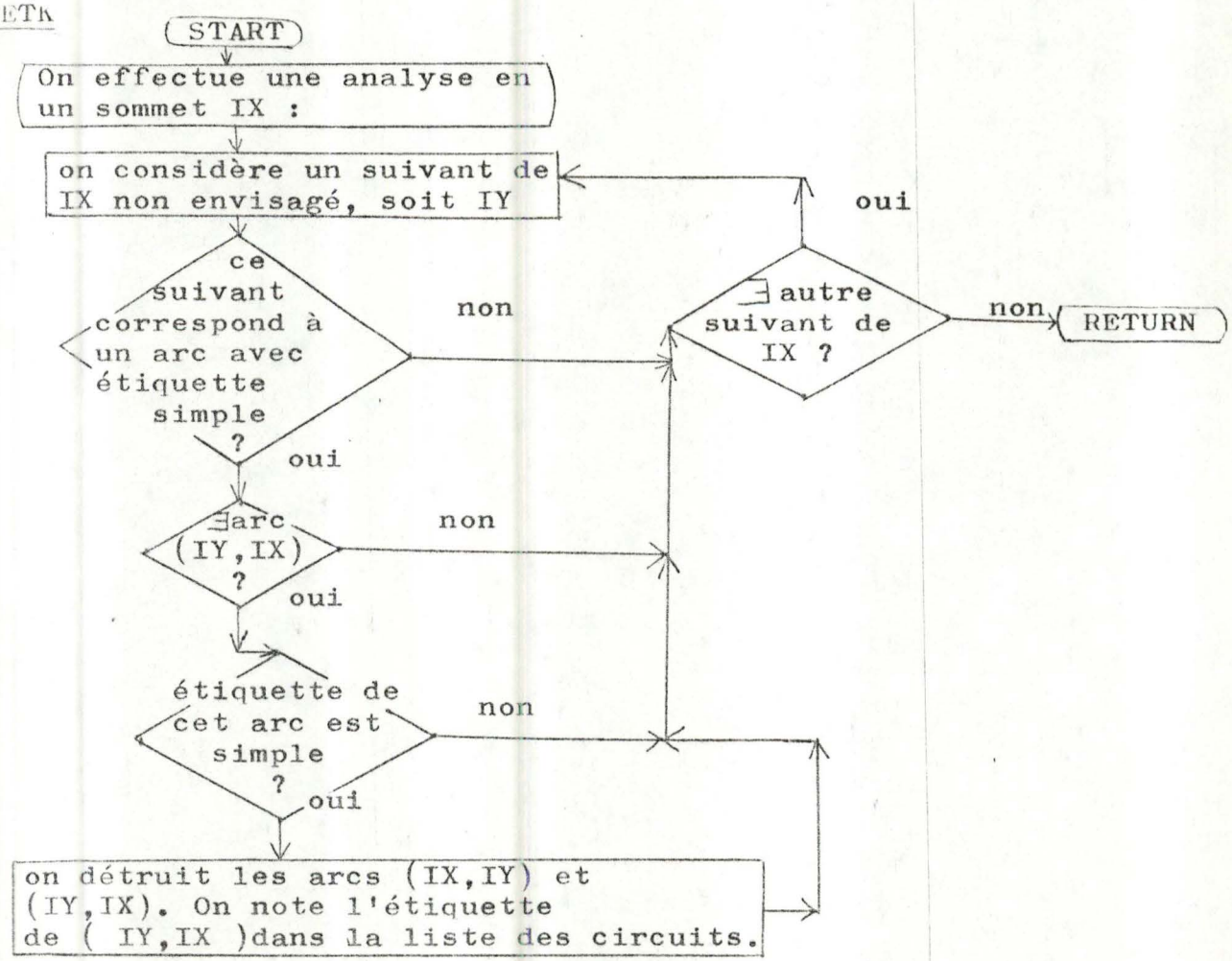
JA : arc ( IX,IY ) courant.

MPRCOU : arc ( IY,IX ) courant.

NTCOU : numéro de colonne courant de matrice COVERT.

Organigramme :

R6CETK



```
      SUBROUTINE R6CETK(IX,LEDFRE,NOCOL,NOLIG,INDCOL,INDLIG,ICOL,LIGN,IP
     1RECO,ISUICO,IPRELI,ISUILI,IGRNTR,NOCROS,IPLUS,ICIPER,COVERT,ICOFRE
     1,III,JJJ,NN,MM)
C
C     REGLE 6 DE C.K.
C
      IMPLICIT LOGICAL*1(C)
      IMPLICIT INTEGER*2(I-N)
      DIMENSION COVERT(JJJ,III)
      DIMENSION NOCOL(NN),NOLIG(NN),INDCOL(NN),INDLIG(NN),ICOL(MM),LIGN(
     1MM),IPRECO(MM),ISUICO(MM),IPRELI(MM),ISUILI(MM)
      DIMENSION LEDFRE(MM)
      DIMENSION IGRNTR(MM),NOCROS(III),IPLUS(III),ICIPER(III),ICOFRE(III
     1)
      ISOMCO=IPLUS(III)
      KMAX=NOCOL(IX)
      KPRMAX=NOLIG(IX)
      IF((KMAX.EQ.0).OR.(KPRMAX.EQ.0))RETURN
      LNEDFR=LEDFRE(MM)
      IPERCI=ICIPER(III)
      JA=INDCOL(IX)
C     CONSIDERER CHAQUE SUIVANT DE IX
      MPRCOU=INDLIG(IX)
      IND=1
      DO 1 L=1,KMAX
      LIJA=LIGN(JA)
      NTJA=IGRNTR(JA)
      IVAX=0
C     COMBIEN DE SOMMETS CONTIENT LE "LABEL"
    3 IF(NOCROS(NTJA).EQ.2)GOTO 2
      IVAX=NTJA
      NTJA=IPLUS(NTJA)
      IF(NTJA.EQ.0)GOTO 1
      GOTO 3
    2 JY=ICOL(MPRCOU)
      IF(JY.GT.LIJA)GOTO 1
C     EXISTENCE DE DOUBLET?
      IF(JY.EQ.LIJA)GOTO 5
      IF(IND.EQ.KPRMAX)GOTO 14
      IND=IND+1
      MPRCOU=ISUILI(MPRCOU)
      GOTO 2
    5 NTCOU=IGRNTR(MPRCOU)
C     EXISTENCE D'UNE ETIQUETTE(LABEL)AYANT 2 SOMMETS
    7 IF(NOCROS(NTCOU).EQ.2)GOTO 6
      NTCOU=IPLUS(NTCOU)
      IF(NTCOU.EQ.0)GOTO 1
      GOTO 7
    6 CONTINUE
C     NOUS AVONS UN CIRCUIT PERTINENT
```

```
      IPERCI=IPERCI+1
      ICIPER(IPERCI)=NTJA
      IF(IVAX.EQ.0)GOTO 8
      IPLUS(IVAX)=IPLUS(NTJA)
      NTRA=IGRNTR(JA)
      GOTO 9
    8 NTRA=IPLUS(NTJA)
      IF(NTRA.EQ.0)GOTO 10
    9 ILR=IPLUS(NTRA)
C     SUPPRIMER LES ETIQUETTES INUTILES
      CALL NET(COVERT,ISOMCO,IPLUS,NTRA,ICOFRE,III,JJJ)
      NTRA=ILR
      IF(NTRA.NE.0)GOTO 9
   10 IGRNTR(JA)=0
C     SUPPRESSION D'ARC
      CALL DELEAR(JA,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN,NOLIG,I
     1NDCOL,INDLIG,NN,MM)
      LNEDFR=LNEDFR+1
      LEDFRE(LNEDFR)=JA
      NTCOU=IGRNTR(MPRCOU)
   11 NSU=IPLUS(NTCOU)
C     SUPPRESSION D'ETIQUETTES
      CALL NET(COVERT,ISOMCO,IPLUS,NTCOU,ICOFRE,III,JJJ)
      NTCOU=NSU
      IF(NTCOU.NE.0)GOTO 11
      IGRNTR(MPRCOU)=0
      LNEDFR=LNEDFR+1
      LEDFRE(LNEDFR)=MPRCOU
C     SUPPRESSION D'ARC
      CALL DELEAR(MPRCOU,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN,NOL
     1IG,INDCOL,INDLIG,NN,MM)
    1 JA=ISUICO(JA)
   14 LEDFRE(MM)=LNEDFR
      ICIPER(III)=IPERCI
      RETURN
      END
```

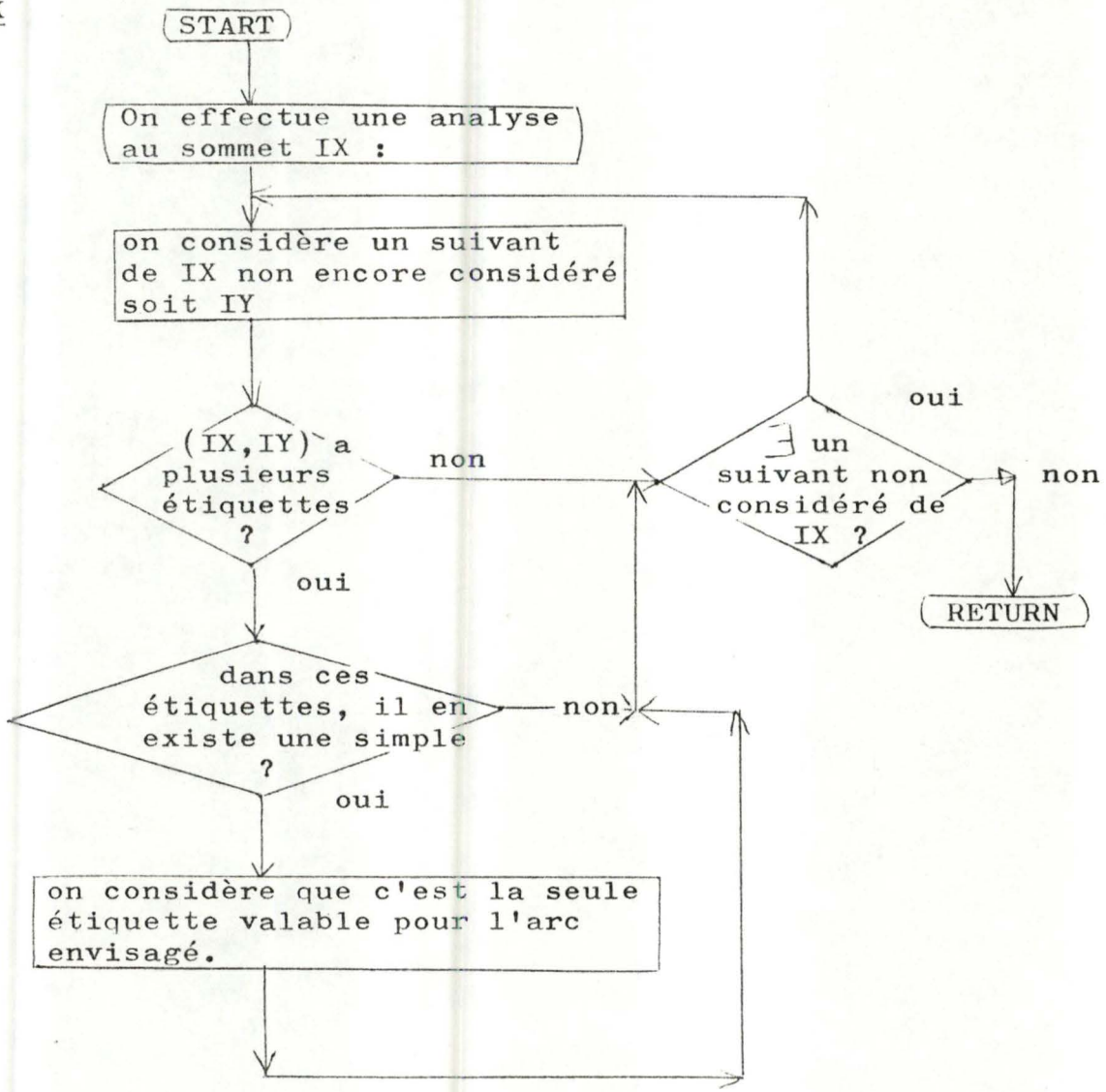
25- La routine R7CETK essaie d'appliquer au sommet IX la règle 7 de l'algorithme de C.K. étape 2 ( voir Ch. II )

IARCY : arc (IX,IY) courant

LABY : numéro courant d'une colonne de COVERT correspondant à une étiquette de l'arc IARCY.

voici l'organigramme.

R7CETK



```
      SUBROUTINE R7CETK(IX,NOCOL,INDCOL,ISUICO,IGRNTR,IPLUS,NOCROS,COVER
     1T,ICOFRE,III,JJJ,NN,MM)
C
C     REGLE 7 DE C.K.
C
      IMPLICIT LOGICAL*1(C)
      IMPLICIT INTEGER*2(I-N)
      DIMENSION COVERT(JJJ,III)
      DIMENSION NOCOL(NN),INDCOL(NN),ISUICO(MM)
      DIMENSION IGRNTR(MM),IPLUS(III),NOCROS(III),ICOFRE(III)
C     ENVISAGER CHAQUE SUIVANT DE IX
      IMAX=NOCOL(IX)
      IF(IMAX.EQ.C) RETURN
      ISOMCO=IPLUS(III)
      IARCY=INDCOL(IX)
      DO 1 I=1,IMAX
      LABY=IGRNTR(IARCY)
      IDEB=LABY
      LABYSU=IPLUS(LABY)
      IF(LABYSU.EQ.0) RETURN
      IPREY=0
C     IL EXISTE UNE ETIQUETTE AYANT 2 SOMMETS(ETIQUETTES SIMPLES) ?
    3 IF(NOCROS(LABY).EQ.2) GOTO 2
      IPREY=LABY
      LABY=IPLUS(LABY)
      IF(LABY.EQ.0) GOTO 1
      GOTO 3
    2 IGRNTR(IARCY)=LABY
      IF(IPREY.EQ.0) GOTO 4
      IPLUS(IPREY)=IPLUS(LABY)
      INIT=IDEB
      GOTO 5
    4 INIT=IPLUS(LABY)
    5 IPLUS(LABY)=0
    7 KC=INIT
      INIT=IPLUS(INIT)
C     SUPPRESSION D"ETIQUETTES
      CALL NET(COVERT,ISOMCO,IPLUS,KC,ICOFRE,III,JJJ)
      IF(INIT.NE.0) GOTO 7
    1 IARCY=ISUICO(IARCY)
      RETURN
      END
```

26- La routine R8CHKS essaie d'appliquer la règle 8 de l'algorithme de C.K. étape 2 ( Ch. II ) au sommet IX duquel on considère les suivants ( voir énoncé de la règle 8 ). Il suffit de remplacer les arguments de R8CHKS se rapportant aux colonnes ( NOCOL, INDCOL, ISUICO ) par les arguments analogues se rapportant aux lignes ( NOLIG, INDLIG, ISUILI ), pour obtenir que l'on peut essayer d'appliquer la règle 8 de C.K. étape 2 au sommet IX duquel on considère les précédents en utilisant la routine R8CHKS.

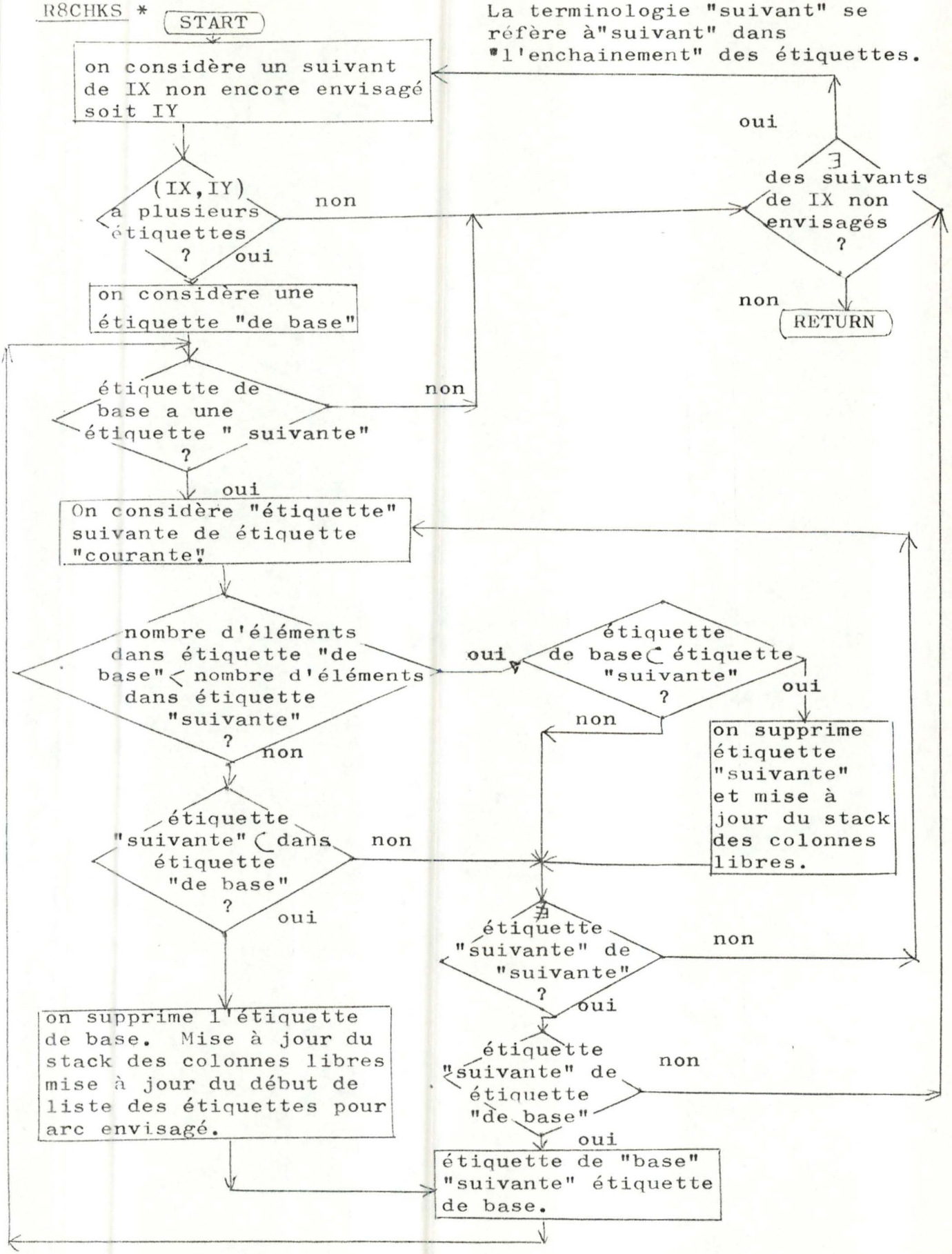
IMAX : nombre de suivants de IX
IEDY : arc courant ( IX,IY )
IARY :
LABBAS :
ISUILA :
IPREC : numéros courants de colonnes de
KRX : la matrice COVERT
LABELI :
ISUB :
IPREBA :

Voir l'organigramme à la page suivante.

27- La routine R9CETK essaie d'appliquer la règle 9 de l'algorithme C.K. étape 2 au sommet IX.

IMAX : nombre de suivants de IX.
KMA : nombre de précédents de IX.
INCO : compteur de suivants de IX.
INLI : compteur de précédents de IX.
IYPR : précédent courant de IX.
IY : suivant courant de IX
IAYPR : arc ( IYPR,IX )
IAY : arc ( IX,IY )
KLAB : numéro courant d'étiquette correspondant à l'arc IAYPR.
IADLAB : numéro courant d'étiquette correspondant à l'arc IAY.

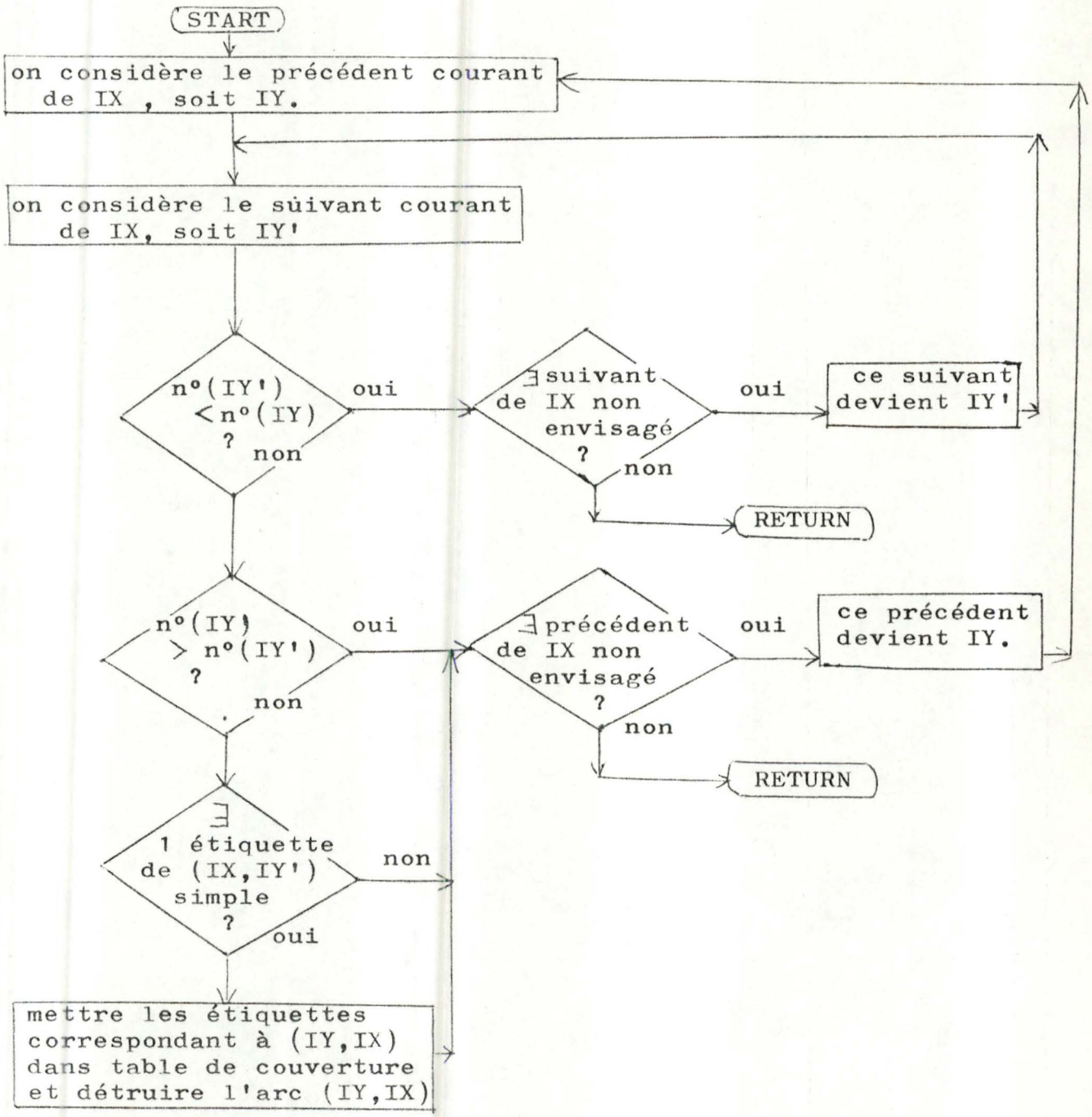
Voir organigramme après celui de R8CHKS et le "listing" de R8CHKS.


R8CHKS *
START
La terminologie "suivant" se réfère à"suivant" dans "l'enchainement" des étiquettes.
on considère un suivant de IX non encore envisagé soit IY
(IX,IY) a plusieurs étiquettes ?
non
oui
∃ des suivants de IX non envisagés ?
oui
non
RETURN
on considère une étiquette "de base"
étiquette de base a une étiquette " suivante" ?
non
oui
On considère "étiquette" suivante de étiquette "courante"
nombre d'éléments dans étiquette "de base" < nombre d'éléments dans étiquette "suivante" ?
oui
non
étiquette de base ⊂ étiquette "suivante" ?
oui
non
on supprime étiquette "suivante" et mise à jour du stack des colonnes libres.
étiquette "suivante" ⊂ dans étiquette "de base" ?
non
oui
∄ étiquette "suivante" de "suivante" ?
non
oui
on supprime l'étiquette de base. Mise à jour du stack des colonnes libres mise à jour du début de liste des étiquettes pour arc envisagé.
étiquette "suivante" de étiquette "de base"
non
oui
étiquette de "base" "suivante" étiquette de base.

```
      SUBROUTINE R8CHKS(IX,NOCOL,INDCOL,ISUICO,IGRNTR,IPLUS,COVERT,ICOFR
     1E,III,JJJ,NN,MM)
C
C     REGLE 8 DE C.K.
C
      IMPLICIT INTEGER*2(I-N)
      IMPLICIT LOGICAL*1(C)
      DIMENSION COVERT(JJJ,III)
      DIMENSION NOCOL(NN),INDCOL(NN),ISUICO(MM)
      DIMENSION IGRNTR(MM),IPLUS(III),ICOFRE(III)
      ISOMCO=IPLUS(III)
      LSOMCO=IPLUS(III)
      IMAX=NOCOL(IX)
      IF(IMAX.EQ.0) RETURN
      IEDY=INDCOL(IX)
C     CONSIDERER CHAQUE SUIVANT DE IX
      DO 1 I=1,IMAX
      IARY=IGRNTR(IEDY)
      IPREBA=0
      LABBAS=IARY
    8 IPREC=LABBAS
      ISUILA=IPLUS(LABBAS)
      IF(ISUILA.EQ.0) GOTO 1
    9 DO 2 K=1,LSOMCO
C     INCLUSION DE L"ETIQUETTE "LABBAS" DANS L"ETIQUETTE "ISUILA"
      IF(.NOT.((.NOT.COVERT(K,LABBAS)).OR.COVERT(K,ISUILA))) GOTO 3
    2 CONTINUE
      IPLUS(IPREC)=IPLUS(ISUILA)
      CALL NET(COVERT,LSOMCO,IPLUS,ISUILA,ICOFRE,III,JJJ)
      ISUILA=IPLUS(IPREC)
      GOTO 4
    3 DO 5 KA=1,LSOMCO
C     INCLUSION DE L"ETIQUETTE "ISUILA" DANS L"ETIQUETTE "LABBAS"
      IF((.NOT.COVERT(KA,ISUILA)).OR.COVERT(KA,LABBAS))GOTO 5
      IPREC=ISUILA
      ISUILA=IPLUS(ISUILA)
      GOTO 4
    5 CONTINUE
      LABELI=LABBAS
      KRX=IPLUS(LABBAS)
      LABBAS=KRX
      IF(IPREBA.EQ.0) GOTO 6
      IPLUS(IPREBA)=KRX
      GOTO 7
    6 IGRNTR(IEDY)=KRX
C     SUPPRESSION DE LA PLUS GRANDE DES 2 ETIQUETTES (ISUILA OU LABBAS)
    7 CALL NET(COVERT,LSOMCO,IPLUS,LABELI,ICOFRE,III,JJJ)
      GOTO 8
    4 IF(ISUILA.NE.0) GOTO 9
      ISUB=IPLUS(LABBAS)

      IF(ISUB.EQ.0) GOTO 1
      IPREBA=LABBAS
      LABBAS=ISUB
      GOTO 8
    1 IEDY=ISUICO(IEDY)
      RETURN
      END
```

R9CETK *


START
on considère le précédent courant de IX , soit IY.
on considère le suivant courant de IX, soit IY'
n°(IY') < n°(IY) ?
oui
non
∃ suivant de IX non envisagé ?
oui
non
ce suivant devient IY'
RETURN
n°(IY) > n°(IY') ?
oui
non
∃ précédent de IX non envisagé ?
oui
non
ce précédent devient IY.
RETURN
∃ 1 étiquette de (IX,IY') simple ?
non
oui
mettre les étiquettes correspondant à (IY,IX) dans table de couverture et détruire l'arc (IY,IX)

```
      SUBROUTINE R9CETK(IX,NOCOL,NOLIG,INDCOL,INDLIG,ICOL,LIGN,IPRECO,IS
     1UICO,IPRELI,ISUILI,IGRNTR,NOCROS,ICIPER,IPLUS,III,NN,MM,LEDFRE)
C
C     REGLE 9 DE C.K.
C
      IMPLICIT INTEGER*2(I-N)
      DIMENSION NOCOL(NN),NOLIG(NN),INDCOL(NN),INDLIG(NN),ICOL(MM),LIGN(
     1MM),IPRECO(MM),ISUICO(MM),IPRELI(MM),ISUILI(MM)
      DIMENSION IGRNTR(MM),NOCROS(III),ICIPER(III),IPLUS(III)
      DIMENSION LEDFRE(MM)
      IMAX=NOCOL(IX)
      KMA=NOLIG(IX)
      IF((IMAX.EQ.0).OR.(KMA.EQ.0)) RETURN
          LNEDFR=LEDFRE(MM)
      IPERCI=ICIPER(III)
      INCO=1
      INLI=1
      IAYPR=INDLIG(IX)
      IYPR=ICOL(IAYPR)
      IAY=INDCOL(IX)
      IY=LIGN(IAY)
    7 IF(IY.LT.IYPR) GOTO 1
      IF(IY.GT.IYPR) GOTO 2
C      IL EXISTE UN DOUBLET (IX,IY)
      IADLAB=IGRNTR(IAY)
C     ETIQUETTE CORRESPONDANT AU SUIVANT DE IX SIMPLE ?
      IF(NOCROS(IADLAB).NE.2) GOTO 2
      KLAB=IGRNTR(IAYPR)
    6 IPERCI=IPERCI+1
      ICIPER(IPERCI)=KLAB
      KLAB=IPLUS(KLAB)
      IF(KLAB.NE.0) GOTO 6
C     ON PEUT DETRUIRE L"ARC (PRECEDENT DE IX,IX)
    5 CALL DELEAR(IAYPR,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN,NOLI
     1G,INDCOL,INDLIG,NN,MM)
      LNEDFR=LNEDFR+1
      LEDFRE(LNEDFR)=IAYPR
      IGRNTR(IAYPR)=0
    2 IF(INLI.EQ.KMA)GOTO 3
      IAYPR=ISUILI(IAYPR)
      IYPR=ICOL(IAYPR)
      INLI=INLI+1
      GOTO 7
C     EFFECTUER CELA POUR TOUS LES DOUBLETS EN IX
    1 IF(INCO.EQ.IMAX)GOTO 3
      IAY=ISUICO(IAY)
      IY=LIGN(IAY)
      INCO=INCO+1
      GOTO 7
    3 ICIPER(III)=IPERCI

      LEDFRE(MM)=LNEDFR
      RETURN
      END
```

28- La routine LABEPL permet, ayant deux arcs ( IZ,IX ) et ( IX,IY ) et leurs étiquettes, d'associer à un arc ( IZ,IY ) les étiquettes constituées comme suit :
considérer la première étiquette $t_1$ de ( IZ,IX ), la première étiquette $u_1$ de ( IX,IY ); les unir pour former l'étiquette $v_1$ ; considérer $t_1$ et la seconde étiquette $u_2$ de ( IX,IY ) ; les unir pour former l'étiquette $v_2$; etc... jusqu'à épuisement des l étiquettes de (IX,IY). Considérer ensuite la seconde étiquette $t_2$ de ( IZ,IX ) et $u_1$; les unir pour former une étiquette $v_{1+1}$; ainsi de suite jusqu'à épuisement des étiquettes de (IX,IY). Répéter la même démarche pour les k étiquettes de (IZ,IX). De cette manière, on constitue k X l étiquettes, k et/ou l valant éventuellement 1.
Il est possible que l'arc ( IZ,IY ) auquel on associe les nouvelles étiquettes possède déjà des étiquettes.

IAPREX : arc ( IZ,IX )

IARIY : arc ( IX,IY )

IARC : arc ( IZ,IY )

KDEBY : numéro d'étiquette courant de IARIY.

KDEBX : numéro d'étiquette courant de IAPREX.

LABEL : nouvelle étiquette, à laquelle on associe l'union des étiquettes KDEBY et KDEBX.

IDERNL : indice qui retient le numéro de la dernière étiquette attribuée.

Pour cette routine nous ne proposons pas d'organigramme.

```
      SUBROUTINE LABEPL(IARIY,IAPREX,IARC,IGRNTR,COVERT,NOCROS,ISOMCO,IP
     1LUS,ICOFRE,III,JJJ,MM)
C
C     SOUROUTINE QUI EFFECTUE DES "CONDENSATIONS D"ETIQUETTES"
C
      IMPLICIT LOGICAL*1(C)
      IMPLICIT INTEGER*2(I-N)
      DIMENSION COVERT(JJJ,III)
      DIMENSION IGRNTR(MM),NOCROS(III),IPLUS(III),ICOFRE(III)
      KDEBY=IGRNTR(IARIY)
      KDEBX=IGRNTR(IAPRLX)
      IDERNL=IGRNTR(IARC)
      INCFRE=ICOFRE(III)
      IF(IDERNL.NE.0) GOTO 1
C     INITIALISATION DU PROCESSUS SI L"ARC IARC A DEJA DES ETIQUETTES
      LABEL=ICOFRE(INCFRE)
      INCFRE=INCFRE-1
      ICOFRE(III)=INCFRE
      IF(INCFRE.LE.1)RETURN
      IGRNTR(IARC)=LABEL
      CALL SOMBOO(COVERT,KDEBY,KDEBX,LABEL,NOCROS,ISOMCO,III,JJJ)
      GOTO 2
C     EFFECTUER LES CONDENSATIONS D"ETIQUETTES POUR TOUT COUPLE D"ETIQUETTES
    1 IF(IPLUS(IDERNL).EQ.0) GOTO 3
      IDERNL=IPLUS(IDERNL)
      GOTO 1
    3 LABEL=ICOFRE(INCFRE)
      INCFRE=INCFRE-1
      ICOFRE(III)=INCFRE
      IF(INCFRE.LE.1)RETURN
      CALL SOMBOO(COVERT,KDEBY,KDEBX,LABEL,NOCROS,ISOMCO,III,JJJ)
      IPLUS(IDERNL)=LABEL
    2 IDERNL=LABEL
      KDEBY=IPLUS(KDEBY)
      IF(KDEBY.NE.0) GOTO 3
      KDEBX=IPLUS(KDEBX)
      IF(KDEBX.EQ.0)GOTO 4
      KDEBY=IGRNTR(IARIY)
      GOTO 3
    4 ICOFRE(III)=INCFRE
      RETURN
      END
```

29- La routine T4 effectue l'élimination ( T4 ) apparaissant dans l'étape 2 de l'algorithme C.K. (cfr. ch.II ) à un sommet IX. La description de cette élimination est différente de celle exposée à propos de T2 ( routine 11 de ce paragraphe ), apparaissant dans l'étape 1 de l'algorithme de C.K. ( ch. II )

a. Les sommets peuvent avoir, ici, plusieurs suivants et plusieurs précédents.
b. Les arcs sont étiquetés.
c. Une simplification dans la conception provient de l'énoncé de l'étape 2 de l'algorithme C.K., qui suggère que les sommets sont envisagés dans l'ordre de leur numérotation.

IMAX : nombre de suivants de IX.

IPREMA : nombre de précédents de IX.

IPREX : un précédent de IX.

IAPREX : arc (IPREX,IX).

IY : un suivant de IX.

IARIY : arc (IX,IY)

ICCO : compteurs de suivants et précédents de IX.

ICLI :

IDESCO : arc courant correspondant à un suivant de IPREX.

KAVANL : vecteur auxiliaire qui contient une information permettant de détecter plus rapidement l'existence d'un arc (IPREX,IY).

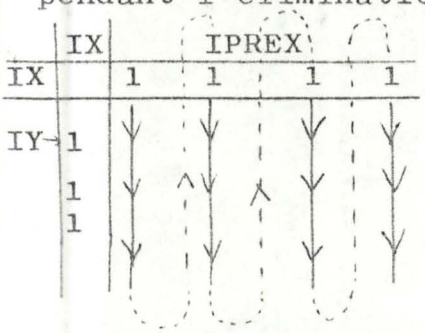
IAVANL : variable qui prend, au cours du déroulement de l'algorithme, la valeur"contenue" dans une position du vecteur KAVANL.

KDEBX : un numéro d'étiquette de IAPREX.

KDEBY : un numéro d'étiquette de IARIY.

Voir organigramme.

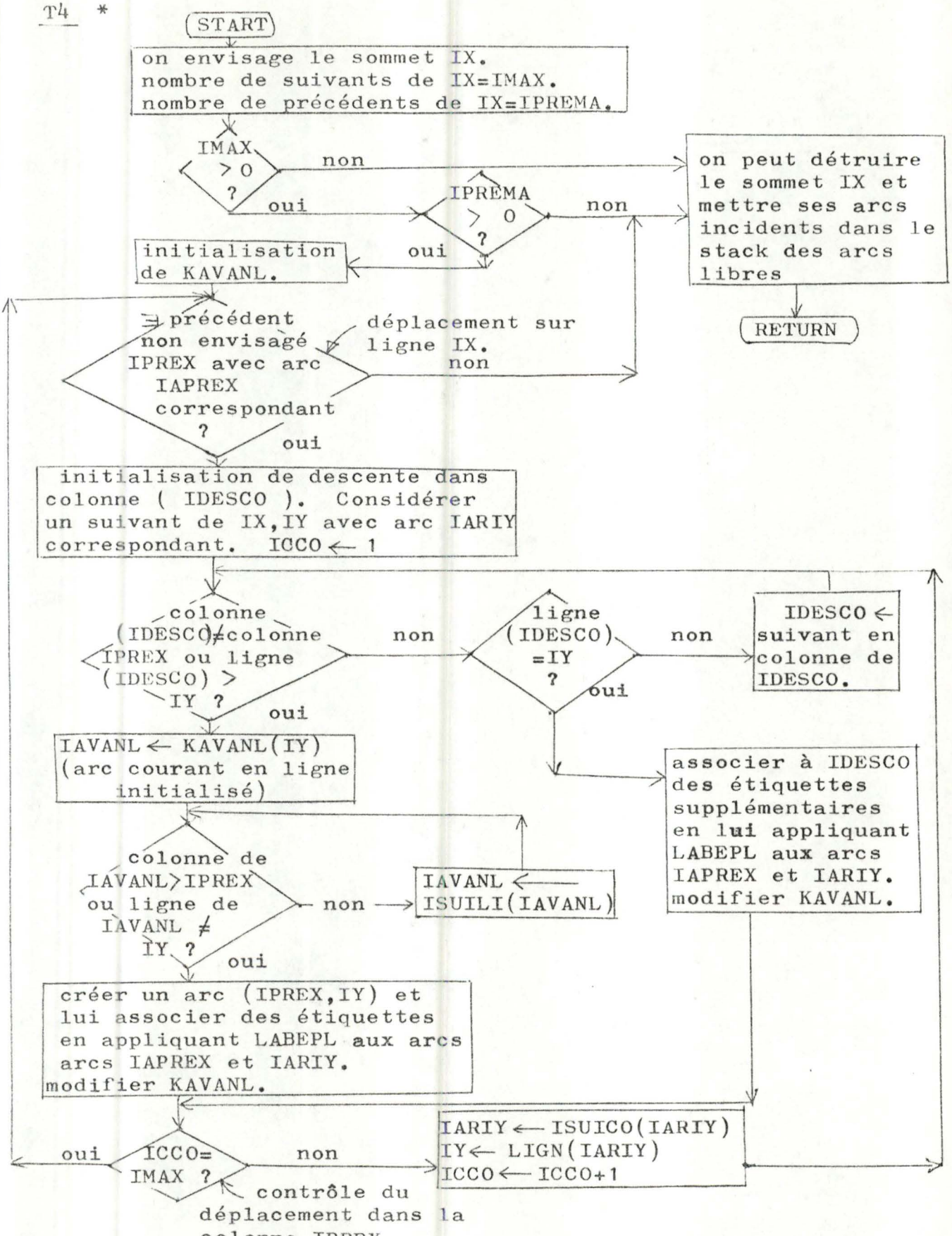
Le schéma ci-dessous indique comment s'effectue le balayage de $M'^{T}$ pendant l'élimination " T4 ".




$M'^{T}$

trajet d'inspection des précédents et suivants de IX en vue d'ajouter des arcs et étiquettes au graphe.

T4 *


START
on envisage le sommet IX.
nombre de suivants de IX=IMAX.
nombre de précédents de IX=IPREMA.
IMAX > 0 ?
non
oui
IPREMA > 0 ?
non
oui
on peut détruire le sommet IX et mettre ses arcs incidents dans le stack des arcs libres
RETURN
initialisation de KAVANL.
∃ précédent non envisagé IPREX avec arc IAPREX correspondant ?
déplacement sur ligne IX.
non
oui
initialisation de descente dans colonne ( IDESCO ). Considérer un suivant de IX,IY avec arc IARIY correspondant. ICCO ← 1
colonne (IDESCO) ≠ colonne IPREX ou ligne (IDESCO) > IY ?
non
oui
ligne (IDESCO) =IY ?
non
oui
IDESCO ← suivant en colonne de IDESCO.
IAVANL ← KAVANL(IY) (arc courant en ligne initialisé)
associer à IDESCO des étiquettes supplémentaires en lui appliquant LABEPL aux arcs IAPREX et IARIY. modifier KAVANL.
colonne de IAVANL > IPREX ou ligne de IAVANL ≠ IY ?
non
oui
IAVANL ← ISUILI(IAVANL)
créer un arc (IPREX,IY) et lui associer des étiquettes en appliquant LABEPL aux arcs arcs IAPREX et IARIY. modifier KAVANL.
ICCO = IMAX ?
oui
non
contrôle du déplacement dans la colonne IPREX.
IARIY ← ISUICO(IARIY)
IY ← LIGN(IARIY)
ICCO ← ICCO+1

```
      SUBROUTINE T4(IX,LEDFRE,NOCOL,NOLIG,INDCOL,INDLIG,IPRECO,IPRELI,IS
     1UICO,ISUILI,ICOL,LIGN,ISOADM,NOSO,KAVANL,IGRNTR,ICOFRE,COVERT,NOCR
     1OS,IPLUS,III,JJJ,NN,MM)
C
C     ELIMINATION "T4" DE L"ETAPE 2 DE L"ALGORITHME DE CHEUNG ET KUH
C
      IMPLICIT INTEGER*2(I-N)
      IMPLICIT LOGICAL*1(C)
      DIMENSION COVERT(JJJ,III)
      DIMENSION LEDFRE(MM)
      DIMENSION NOCOL(NN),NOLIG(NN),INDCOL(NN),INDLIG(NN),IPRECO(MM),IPR
     1ELI(MM),ISUICO(MM),ISUILI(MM),ICOL(MM),LIGN(MM),ISOADM(NN)
      DIMENSION KAVANL(NN),IGRNTR(MM),ICOFRE(III),NOCROS(III),IPLUS(III)
      ISOMCO=IPLUS(III)
      LNEDFR=LEDFRE(MM)
      IMAX=NOCOL(IX)
      IPREMA=NOLIG(IX)
      IF(IMAX.EQ.0) GOTO 11
      IF(IPREMA.NE.0) GOTO 13
      LEDFRE(MM)=LNEDFR
      GOTO 14
   13 DO 1 I=1,NOSO
      LX=ISOADM(I)
C     INITIALISER KAVANL
    1 KAVANL(LX)=INDLIG(LX)
      ICLI=1
      IAPREX=INDLIG(IX)
   12 IPREX=ICOL(IAPREX)
      IDESCO=INDCOL(IPREX)
C     POUR CHAQUE PRECEDENT DE IX :
      IARIY=INDCOL(IX)
      IY=LIGN(IARIY)
      ICCO=1
C     EXAMINER LES SUIVANTS DU PRECEDENT DE IX
    4 IF((ICOL(IDESCO).NE.IPREX).OR.(LIGN(IDESCO).GT.IY)) GOTO 2
      IF(LIGN(IDESCO).EQ.IY)GOTO 3
      IDESCO=ISUICO(IDESCO)
      IF(IDESCO.EQ.ISUICO(IDESCO)) RETURN
      GOTO 4
    2 IAVANL=KAVANL(IY)
    7 IF((ICOL(IAVANL).GT.IPREX).OR.(LIGN(IAVANL).NE.IY))GOTO 6
      IAVANL=ISUILI(IAVANL)
      GOTO 7
C     IL FAUT CREER UN ARC
    6 IARC=LEDFRE(LNEDFR)
      LNEDFR=LNEDFR-1
      IGRNTR(IARC)=0
      IP=IPRECO(IDESCO)
      IPRECO(IARC)=IP
      ISUICO(IARC)=IDESCO
```

```
      IPRECO(IDESCO)=IARC
      ISUICO(IP)=IARC
      IQ=IPRELI(IAVANL)
      IPRELI(IARC)=IQ
      ISUILI(IARC)=IAVANL
      IPRELI(IAVANL)=IARC
      ISUILI(IQ)=IARC
      LIGN(IARC)=IY
      ICOL(IARC)=IPREX
      NOCOL(IPREX)=NOCOL(IPREX)+1
      NOLIG(IY)=NOLIG(IY)+1
      KAVANL(IY)=IAVANL
      GOTO 10
C     IL EXISTE DEJA UN ARC
    3 IARC=IDESCO
      KAVANL(IY)=IDESCO
C     IL FAUT DONNER DES ETIQUETTES AU NOUVEL ARC OU BIEN EN AJOUTER SI
C     L"ARC EXISTAIT DEJA
   10 CALL LABEPL(IARIY,IAPREX,IARC,IGRNTR,COVERT,NOCROS,ISOMCO,IPLUS,IC
     1OFRE,III,JJJ,MM)
      INCFRE=ICOFRE(III)
      IF(INCFRE.LE.1)RETURN
      IF(ICCO.EQ.IMAX) GOTO 5
C     FAIRE CELA POUR CHAQUE SUIVANT DE IX
      IARIY=ISUICO(IARIY)
      IY=LIGN(IARIY)
      ICCO=ICCO+1
      GOTO 4
    5 IF(ICLI.EQ.IPREMA) GOTO 11
      IAPREX=ISUILI(IAPREX)
      ICLI=ICLI+1
      GOTO 12
   11 LEDFRE(MM)=LNEDFR
      IF(IPREMA.EQ.0)GOTO 21
      IAPREX=INDLIG(IX)
C     IL FAUT DETRUIRE LES ARCS CORRESPONDANT UAX PRECEDENTS DE IX
      DO 15 IZ=1,IPREMA
      KDEBX=IGRNTR(IAPREX)
   16 IREP=IPLUS(KDEBX)
C     IL FAUT DETRUIRE LES ETIQUETTES CORRESPONDANTES
      CALL NET(COVERT,ISOMCO,IPLUS,KDEBX,ICOFRE,III,JJJ)
      IF(IREP.EQ.0) GOTO 15
      KDEBX=IREP
      GOTO 16
   15 IAPREX=ISUILI(IAPREX)
   21 IF(IMAX.EQ.0)GOTO 20
C     IL FAUT DETRUIRE LES ARCS CORRESPONDANT AUX SUIVANTS DE IX
   14 IARIY=INDCOL(IX)
      DO 18 IZ=1,IMAX
      KDEBY=IGRNTR(IARIY)

   19 IREP=IPLUS(KDEBY)
C     IL FAUT DETRUIRE LES ETIQUETTES CORRESPONDANTES
      CALL NET(COVERT,ISOMCO,IPLUS,KDEBY,ICOFRE,III,JJJ)
      IF(IREP.EQ.0) GOTO 18
      KDEBY=IREP
      GOTO 19
   18 IARIY=ISUICO(IARIY)
   20 LEDFRE(MM)=LNEDFR
      RETURN
      END
```

30- La routine STEP2 effectue l'étape 2 de l'algorithme C.K. ( ch.II ) en organisant l'appel des routines de simplification décrites ci-plus haut ( R6CETK, R7CETK, R8CHKS, R9CETK, T4 ) après avoir étiqueté le graphe.

KTABCO : numéro courant d'une colonne libre de la table de couverture.

LSOMCO : nombre de sommets non isolés du graphe partiellement réduit après application de l'étape 1 de l'algorithme de C.K.

KSONTR : vecteurs qui permettent une renumérotation des
KSOETI : sommets du graphe.

KSONTR : vecteur qui, à chaque sommet du graphe partiellement réduit par l'étape 1 de l'algorithme C.K., associe 1 numéro i, 1 i LSOMCO de façon biunivoque.

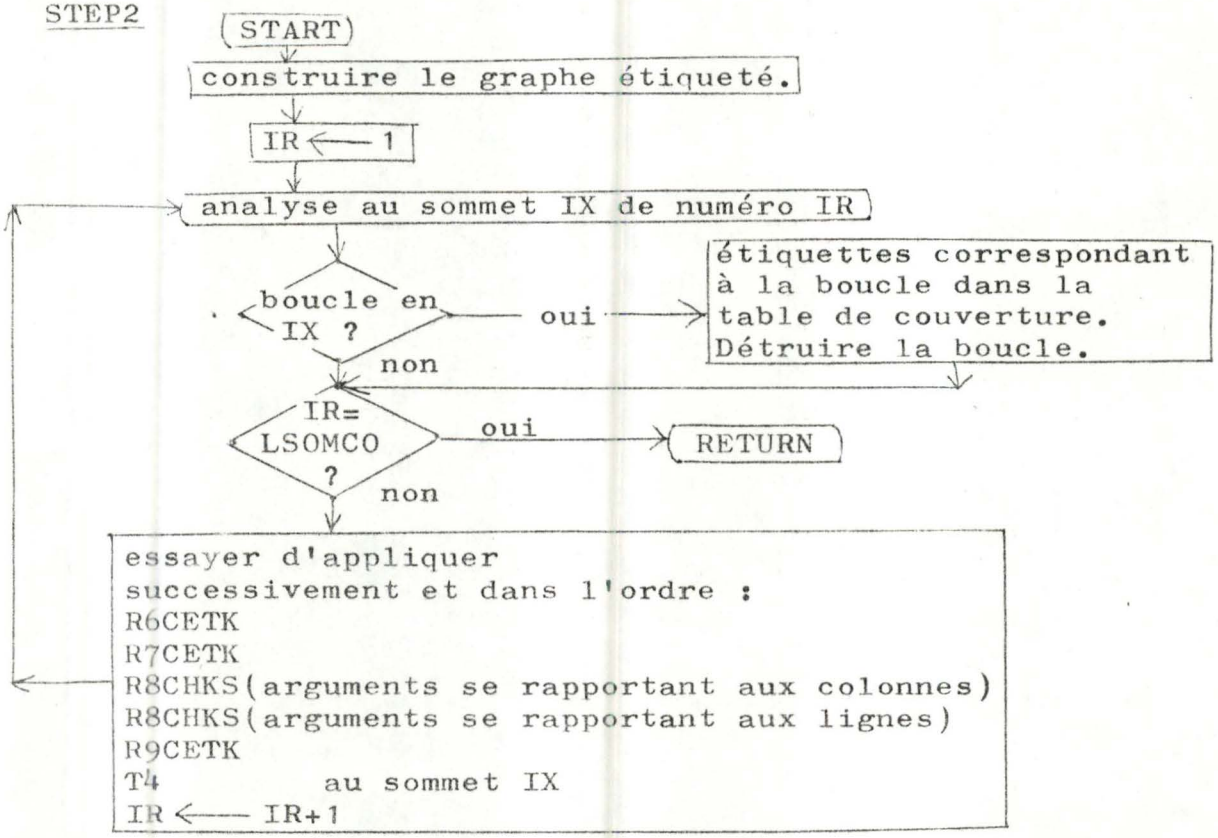
KSOETI : vecteur qui, à chaque sommet du graphe étiqueté, fait correspondre l'ancien numéro de ce sommet dans le graphe non totalement réduit.

IR : numéro du sommet à analyser.

IX : numéro du sommet qui porte maintenant le numéro IR.

Voici l'organigramme :

STEP2



```
      SUBROUTINE STEP2(COVERT,KSONTR,KSOETI,IPLUS,ICOFRE,LEDFRE,NOCOL,NO
     1LIG,INDCOL,INDLIG,ICOL,LIGN,IPRECO,ISUICO,IPRELI,ISUILI,IGRNTR,ICI
     2FER,ISOADM,NOSO,NOMBAR,NOCROS,KAVANL,III,JJJ,NN,MM)
C
C     STEP2 EFFECTUE TOUTE L"ETAPE 2 DE L"ALGORITHME DE CHEUNG ET KUH
C
      IMPLICIT LOGICAL*1(C)
      IMPLICIT INTEGER*2(I-N)
      DIMENSION COVERT(JJJ,III)
      DIMENSION KSONTR(NN),KSOETI(JJJ),IPLUS(III),ICOFRE(III),NOCROS(III
     1),IGRNTR(NM),ICIPER(III)
      DIMENSION KAVANL(NN)
      DIMENSION LEDFRE(MM),ISOADM(NN)
      DIMENSION NOCOL(NN),NOLIG(NN),INDCOL(NN),INDLIG(NN),ICOL(MM),LIGN(
     1MM),IPRECO(MM),IPRELI(MM),ISUICO(MM),ISUILI(MM)
      KTABCO=0
      LSOMCO=0
      ICIPER(III)=0
C
C      INITIALISATIONS
C
      DO 1 I=1,JJJ
    1 KSOETI(I)=0
      DO 12 I=1,NN
   12 KSONTR(I)=0
      DO 2 I=1,III
      IPLUS(I)=0
    2 ICOFRE(I)=I
      INCFRE=III-1
      KR=NOMBAR+1
      LNEDFR=LEDFRE(MM)
      ILM2=MM-1
      IF(KR.GE.ILM2)GOTO 11
C     CONSTITUTION D"UNE RESERVE D"ARCS LIBRES
      DO 10 KS=KR,ILM2
      IGRNTR(KS)=0
      LNEDFR=LNEDFR+1
   10 LEDFRE(LNEDFR)=KS
C
C     INITIALISATION DE TABLE DE COUVERTURE
C
   11 DO 3 K=1,NOSO
      KX=ISOADM(K)
      IF(NOLIG(KX).EQ.0)GOTO 3
      LSOMCO=LSOMCO+1
C     RENUMEROTATION DES SOMMETS
      KSONTR(KX)=LSOMCO
      KSOETI(LSOMCO)=KX
    3 CONTINUE
      LEDFRE(MM)=LNEDFR

      IF(LSOMCO.EQ.0)RETURN
      DO 4 KA=1,LSOMCO
      KSOM=KSOETI(KA)
      NUMSO=NOLIG(KSOM)
      KARC=INDLIG(KSOM)
C     ENVISAGER CHAQUE ARC SUCCESSIVEMENT
      DO 5 KB=1,NUMSO
      KTABCO=INCFRE
      INCFRE=INCFRE-1
C     ASSOCIER UNE ETIQUETTE A CHAQUE ARC
      IGRNTR(KARC)=KTABCO
      LI=LIGN(KARC)
      DO 6 KD=1,LSOMCO
    6 COVERT(KD,KTABCO)=.FALSE.
      COVERT(KA,KTABCO)=.TRUE.
      ICO=ICOL(KARC)
      IETICO=KSONTR(ICO)
      COVERT(IETICO,KTABCO)=.TRUE.
      NOCROS(KTABCO)=2
    5 KARC=ISUILI(KARC)
    4 CONTINUE
```

```
C
C       ENVISAGER CHAQUE SOMMET SUCCESSIVEMENT
        IR=1
        IPLUS(III)=LSOMCO
        ICOFRE(III)=INCFRE
      9 IX=KSOETI(IR)
        LNEDFR=LEDFRE(MM)
        IARIX=INDLIG(IX)
C       EXISTE-T-IL UNE BOUCLE AU SOMMET DE NUMERO IR
        IF(ICOL(IARIX).NE.IX)GOTO 7
        LABEL=IGRNTR(IARIX)
        IPERCI=ICIPER(III)
      8 IPERCI=IPERCI+1
        ICIPER(IPERCI)=LABEL
        LABEL=IPLUS(LABEL)
        IF(LABEL.NE.0)GOTO 8
C       DETRUIRE LA BOUCLE AU SOMMET DE NUMERO IR
        CALL DELEAR(IARIX,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,NOCOL,LIGN,NOLI
       1G,INDCOL,INDLIG,NN,MM)
        LNEDFR=LNEDFR+1
        LEDFRE(LNEDFR)=IARIX
        IGRNTR(IARIX)=0
        ICIPER(III)=IPERCI
      7 CONTINUE
        IF(IR.EQ.LSOMCO)RETURN
        LEDFRE(MM)=LNEDFR
C
C       ESSAYER D'APPLIQUER LA REGLE 6 DE CHEUNG ET KUH AU SOMMET DE NUMERO IR
        CALL R6CETK(IX,LEDFRE,NOCOL,NOLIG,INDCOL,INDLIG,ICOL,LIGN,IPRECO,I

       1SUICO,IPRELI,ISUILI,IGRNTR,NOCROS,IPLUS,ICIPER,COVERT,ICOFRE,III,J
       1JJ,NN,MM)
C
C       ESSAYER D'APPLIQUER LA REGLE 7 DE CHEUNG ET KUH AU SOMMET DE NUMERO IR
        CALL R7CETK(IX,NOCOL,INDCOL,ISUICO,IGRNTR,IPLUS,NOCROS,COVERT,ICOF
       1RE,III,JJJ,NN,MM)
C
C       ESSAYER D'APPLIQUER LA REGLE 8 DE CHEUNG ET KUH AU SOMMET DE NUMERO IR
        CALL R8CHKS(IX,NOCOL,INDCOL,ISUICO,IGRNTR,IPLUS,COVERT,ICOFRE,III,
       1JJJ,NN,MM)
        CALL R8CHKS(IX,NOLIG,INDLIG,ISUILI,IGRNTR,IPLUS,COVERT,ICOFRE,III,
       1JJJ,NN,MM)
C
C       ESSAYER D'APPLIQUER LA REGLE 9 DE CHEUNG ET KUH AU SOMMET DE NUMERO IR
        CALL R9CETK(IX,NOCOL,NOLIG,INDCOL,INDLIG,ICOL,LIGN,IPRECO,ISUICO,I
       1PRELI,ISUILI,IGRNTR,NOCROS,ICIPER,IPLUS,III,NN,MM,LEDFRE)
C       APPLIQUER LA REGLE D'ELIMINATION T4 AU SOMMET DE NUMERO IR
        CALL T4(IX,LEDFRE,NOCOL,NOLIG,INDCOL,INDLIG,IPRECO,IPRELI,ISUICO,I
       1SUILI,ICOL,LIGN,ISOADM,NOSO,KAVANL,IGRNTR,ICOFRE,COVERT,NOCROS,IPL
       1US,III,JJJ,NN,MM)
        INCFRE=ICOFRE(III)
        IF(INCFRE.LE.1)GOTO 500
C       ON PEUT DETRUIRE LE SOMMET DE NUMERO IR
        CALL DELESO(IX,IPRECO,ISUICO,IPRELI,ISUILI,ICOL,LIGN,INDCOL,INDLIG
       1,NOCOL,NOLIG,NN,MM,LEDFRE)
        IR=IR+1
        GOTO 9
    500 ICIPER(III)=-ICIPER(III)
        RETURN
        END
```

Les deux organigrammes suivants correspondent à une part de l'analyse du problème de réduction de table de couverture. (cfr. ch. II étape 3 de l'algorithme de C.K.). Les routines correspondantes n'ont pas été programmées.

30- La routine DOMICO effectue des suppression de colonnes T de la table de couverture quand il existe d'autres colonnes "incluses"dans T.

Manière de procéder :

fixer un numéro de colonne dite colonne de base; voir si, dans les colonnes qui suivent cette colonne dans la table de couverture ( colonnes suivantes de base ) il existe une colonne T

a- soit incluse dans la colonne de base
- supprimer la colonne de base de la table de couverture
- prendre comme colonne de base, la suivante de la colonne de base dans la table de couverture.

b- soit contenant la colonne de base.
- supprimer la colonne T
- prendre la colonne suivante de T dans la table de couverture comme colonne T . Si T n'existe pas, passer à une colonne de base suivante. Si cette colonne de base n'existe pas, arrêter le processus.

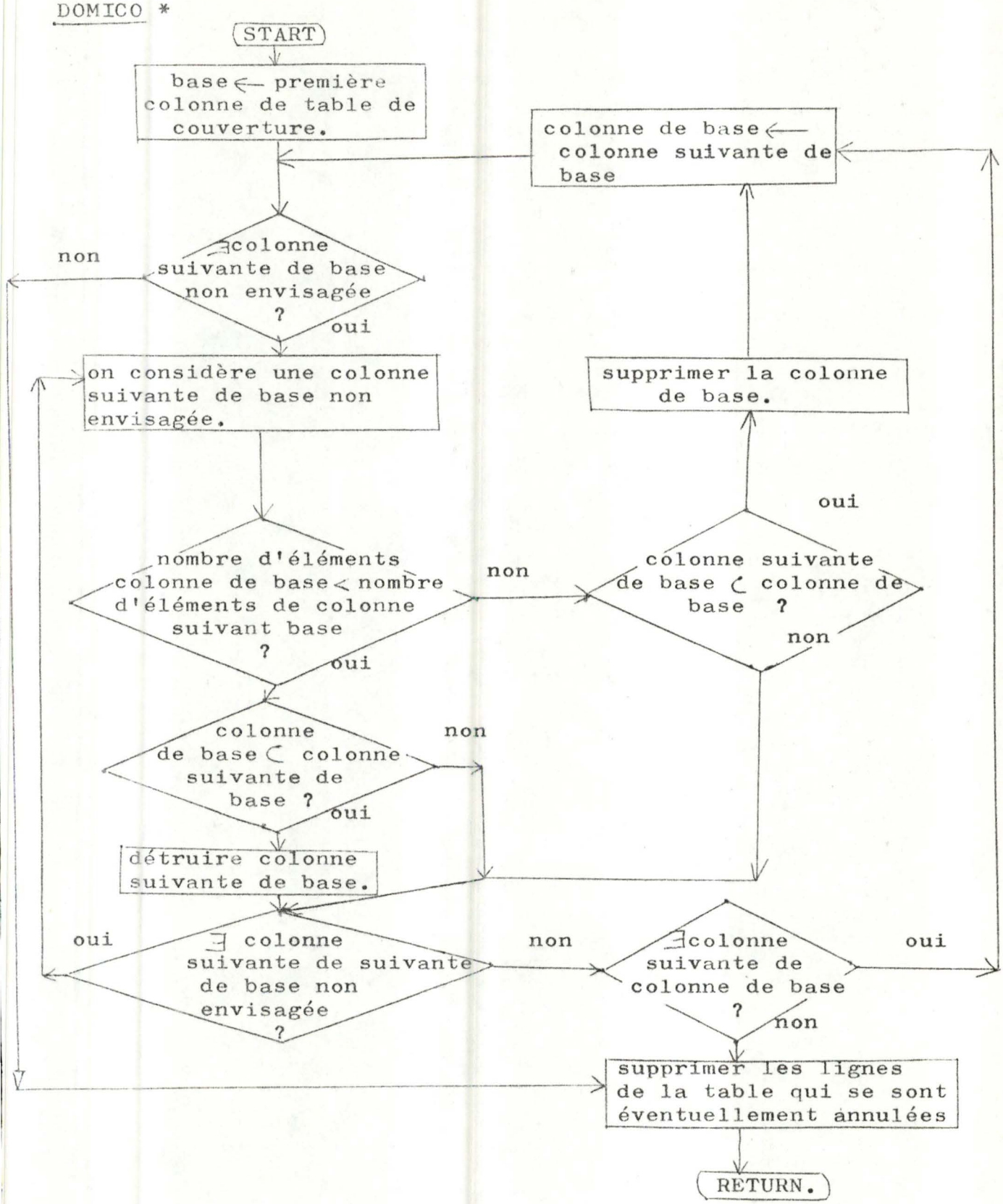
Voir organigramme à la page suivante.

31- La routine DOMILI effectue les suppressions de lignes de la table de couverture de manière analogue à DOMICO, à la différence près qu'une ligne T de la table de couverture sera supprimée quand elle est incluse dans une autre ligne de la table. Nous ne donnons donc pas l'organigramme de cette routine.

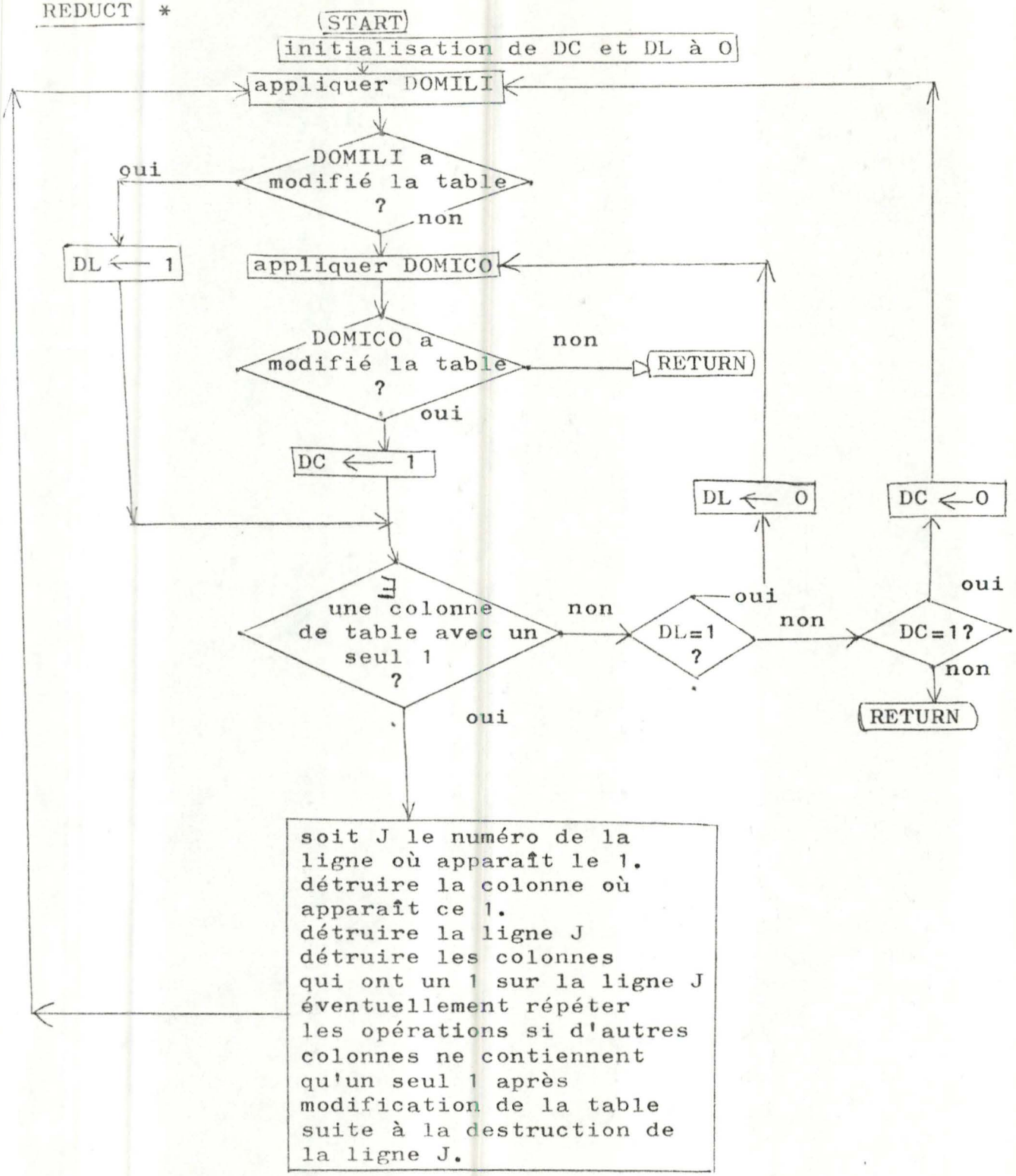
32- La routine REDUCT permet d'effectuer la réduction de table de couverture en se servant de routines DOMILI et DOMICO. Il faut tenir compte du fait qu'un sommet IX appartient à l'ensemble minimum essentiel quand une colonne de la table de couverture ne "contient" que un seul 1,qui se trouve sur la ligne de la table correspondant au sommet IX. On détruit alors la ligne correspondant au sommet IX et on détruit les colonnes de la table de couverture qui "contiennent" un élément = 1 sur la ligne de la table correspondant à IX.

Voir l'organigramme, après celui de DOMICO.

DOMICO *


START
base ← première colonne de table de couverture.
colonne de base ← colonne suivante de base
∃ colonne suivante de base non envisagée ?
non
oui
on considère une colonne suivante de base non envisagée.
supprimer la colonne de base.
nombre d'éléments colonne de base < nombre d'éléments de colonne suivant base ?
non
oui
colonne suivante de base ⊂ colonne de base ?
oui
non
colonne de base ⊂ colonne suivante de base ?
non
oui
détruire colonne suivante de base.
∃ colonne suivante de suivante de base non envisagée ?
oui
non
∃ colonne suivante de colonne de base ?
oui
non
supprimer les lignes de la table qui se sont éventuellement annulées
RETURN.

REDUCT *


START
initialisation de DC et DL à 0
appliquer DOMILI
DOMILI a modifié la table ?
oui
non
DL ← 1
appliquer DOMICO
DOMICO a modifié la table ?
non
RETURN
oui
DC ← 1
DL ← 0
DC ← 0
∃ une colonne de table avec un seul 1 ?
non
oui
DL=1 ?
non
oui
DC=1?
non
RETURN
oui
soit J le numéro de la ligne où apparaît le 1.
détruire la colonne où apparaît ce 1.
détruire la ligne J
détruire les colonnes qui ont un 1 sur la ligne J
éventuellement répéter les opérations si d'autres colonnes ne contiennent qu'un seul 1 après modification de la table suite à la destruction de la ligne J.

Le programme HAUDON effectue l'algorithme décrit dans les chapitres I et II en utilisant les routines que nous venons de proposer. Nous en donnons uniquement le "listing".

```
      PROGRAM HAUDON
C     ********************************************
C     *                                          *
C     *                                          *
C     *          GERTRUDE HAUSTRATE              *
C     *                                          *
C     *          PHILIPPE DENTAINE               *
C     *                                          *
C     *                                          *
C     ********************************************
C
C     CE PROGRAMME EFFECTUE UNE PARTIE D"UN ALGORITHME DE DECOMPOSITION
C
C     SUGGERE PAR KEVORKIAN ET CHEUNG ET KUH
C
C     POUR REFERENCE , VEUILLEZ CONSULTER LES CH. 1 ET 2 DU MEMOIRE DE
C
C     SECONDE LICENCE EN MATH. DES AUTEURS DU PROGRAMME
C
C     ANNEE SCOLAIRE 1975-1976   -  FAC.UNIV.N.-D. DE LA PAIX - NAMUR
C
C
C     LA DESCRIPTION DES VARIABLES UTILISEES DANS CE PROGRAMME SE TROUVE
C     LE TOME 2 DU MEMOIRE DE SECONDE LICENCE EN MATH. DES AUTEURS DU PROGAMME
C     CE PROGRAMME PRINCIPAL NE SERT QU"A DIMENSIONNER LES VECTEURS ET MATRICE
C     ET A APPELER LES SOUROUTINES QUI EFFECTUENT L"ALGORITHME DECRIT DANS LE
C     TOME 1 DU MEMOIRE DE SECONDE LICENCE DES AUTEURS DU PROGRAMME
C     LES VARIABLES DU PROGRAMME PRINCIPAL N"ONT DE SIGNIFICATION UNIQUE QUE
C     DANS LES SOUROUTINES QUI S"EN SERVENT . ICI ELLES PEUVENT AVOIR DIFFERENTS
C     ROLES
      IMPLICIT INTEGER*2(I-N)
      IMPLICIT LOGICAL*1(C)
      DIMENSION I1(101),I2(6500),I3(101),I4(101)
      DIMENSION I5(101),I6(101),I7(101),I8(101)
      DIMENSION I9(101),I10(6500),I11(101),I12(6500),I13(6500)
      DIMENSION I14(6500),I15(6500),I16(6500),I17(6500)
      DIMENSION I18(101),I19(101),I20(101),I21(101),I22(101)
      DIMENSION I23(101),I24(101),I25(101)
      DIMENSION I26(101)
      DIMENSION I27(6500)
      DIMENSION I28(101)
      DIMENSION COV(100,6500)
      DIMENSION I31(100),I32(6500)
      DIMENSION I33(6500)
      DIMENSION I35(6500),I36(6500),I37(6500),I38(101)
      MM=6500
      III=6500
      NN=101
      JJJ=100
      DO 7 NGRA=1,2
```

```
      PRINT 8
    8 FORMAT(1H1)
C
C     LECTURE DU GRAPHE DE FLUX
C
      CALL CODE(I1,I2,N,INDEDG,NN,MM)
      PRINT 500
      PRINT 3,(I1(I),I=1,N)
      PRINT 505
      PRINT 3,(I2(I),I=1,INDEDG)
      DO 9 I=1,N
    9 I3(I)=0
C
C     RECHERCHE DE COMPOSANTES FORTEMENT CONNEXES
C
      CALL STCNCT(I3,N,I1,I2,INDEDG,I4,NOCFCN,IFICF(,I5,I6,I7,I8,I9,MM,N
     1N,I21)
      PRINT 510
      DO 515 I=1,NOCFCN
      IP1=I+1
      LIM1=I4(I)
      LIM2=I4(IP1)-1
      PRINT 519,I
      PRINT 3,(I5(ICF),ICF=LIM1,LIM2)
  515 CONTINUE
C
C     CONSTRUCTION DU GRAPHE CONDENSE
C
      CALL GRACFC(NOCFCN,I6,I7,I4,I10,I3,I5,I1,I2,NN,MM,I8)
C
C     INITIALISATIONS
C
      DO 1 I=1,NN
      I18(I)=0
      I6(I)=0
      I9(I)=0
      I8(I)=0
      I11(I)=0
      I12(I)=0
      I20(I)=0
    1 I21(I)=0
C
C     DECOMPOSITION EN NIVEAUX
C
      CALL TOPSOR(NOCFCN,I8,I7,I10,I25,I24,NN,MM,NONIV)
      PRINT 525
      NONIV=NONIV-1
      DO 530 I=1,NONIV
      IP1=I+1
      LIM1=I24(I)
      LIM2=I24(IP1)-1
      IM1=I-1
      PRINT 535,IM1
      PRINT 3,(I25(INIV),INIV=LIM1,LIM2)
  530 CONTINUE
      DO 6 I=1,NN
    6 I8(I)=0
      DO 2 J=1,NONIV
      PRINT 8
      JM1=J-1
      PRINT 540,JM1
      J1=I24(J)
      IP1=J+1
      J2=I24(IP1)-1
      DO 2 ITTT=J1,J2
      ICOMPN=I25(ITTT)
      PRINT 545,ICOMPN
```

```
C
C     CONSTRUCTION D"UNE FORME ENCHAINEE DE LA MATRICE ASSOCIEE A LA COMPOSANTE
C
      CALL SPACFC(ICOMPN,I4,I5,I1,I2,I6,I9,I8,I11,I12,I13,I14,I15,I16,I1
     17,NN,MM,NSCOMP,NOMBAR,I18,I19)
      IF(NSCOMP.EQ.1)GOTO 160
      PRINT 550
      PRINT 555
      PRINT 3,(I18(I),I=1,N)
      PRINT 560
      PRINT 3,(I19(I),I=1,NSCOMP)
      PRINT 565
      PRINT 3,(I6(I),I=1,NSCOMP)
      PRINT 570
      PRINT 3,(I9(I),I=1,NSCOMP)
      PRINT 575
      PRINT 3,(I8(I),I=1,NSCOMP)
      PRINT 580
      PRINT 3,(I11(I),I=1,NSCOMP)
      PRINT 585
      PRINT 3,(I16(I),I=1,NOMBAR)
      PRINT 593
      PRINT 3,(I17(I),I=1,NOMBAR)
      PRINT 595
      PRINT 3,(I12(I),I=1,NOMBAR)
      PRINT 600
      PRINT 3,(I13(I),I=1,NOMBAR)
      PRINT 605
      PRINT 3,(I14(I),I=1,NOMBAR)
      PRINT 610
      PRINT 3,(I15(I),I=1,NOMBAR)
      I27(MM)=0
C     INITIALISATION DE ISOADM
      DO 10 NRT=1,NSCOMP
   10 I23(NRT)=NRT
      PRINT 680
C
C     ETAPE 1 DE L"ALGORITHME DE CHEUNG ET KUH
C
      CALL STEP1(NSCOMP,I6,I9,I12,I14,I13,I15,I8,I11,I20,I21,I22,INDMES,
     1I23,NOSO,NN,MM,I16,I17,I26,I27)
      IF(INDMES.EQ.0)GOTO 1000
      PRINT 620,INDMES
      DO 5 ID=1,INDMES
      IE=I22(ID)
    5 I22(ID)=I19(IE)
      PRINT 3,(I22(I),I=1,INDMES)
      GOTO 18
 1000 CONTINUE
      PRINT 615
   18 CONTINUE
      IF(NOSO.EQ.0)GOTO 1020
      NOSRES=0
      DO 1010 LTK=1,NOSO
      NOTR=I23(LTK)
      IF(I11(NOTR).NE.0)NOSRES=NOSRES+1
 1010 CONTINUE
      IF(NOSRES.EQ.0)GOTO 1020
      PRINT 810
      ITEST=0
```

```
C
C       ETAPE 2 DE L'ALGORITHME DE CHEUNG ET KUH
C
        CALL STEP2(COV,I30,I31,I32,I33,I27,I8,I11,I6,I9,I16,I17,I12,I13,I1
       14,I15,I35,I36,I23,NOSO,NOMBAR,I37,I38,III,JJJ,NN,MM)
        IPERCI=I36(III)
        IF(IPERCI.EQ.0)GOTO 130
        IF(IPERCI.LT.0)GOTO 1050
  140 CONTINUE
        ISOMCO=I32(III)
        PRINT 860
        DO 33 KTT=1,IPERCI
        IU=I36(KTT)
        PRINT 120,(COV(JU,IU),JU=1,ISOMCO)
   33 CALL NET(COV,ISOMCO,I32,IU,I33,III,JJJ)
        DO 12 L=1,NSCOMP
        LA=I31(L)
   12 I31(L)=I9(LA)
        PRINT 870
        PRINT 120,(I31(LB),LB=1,NSCOMP)
        IF(ITEST.EQ.1)GOTO 125
        PRINT 850
        GOTO 125

 1050 CONTINUE
        PRINT 840
        IPERCI=-IPERCI
        ITEST=1
        GOTO 140
  100 CONTINUE
        PRINT 900
        GOTO 125
  130 PRINT 820
        GOTO 125
 1020 CONTINUE
        PRINT 800
  125 CONTINUE
C
C       QUELQUES RE-INITIALISATIONS
C
        DO 4 I=1,NSCOMP
        I9(I)=0
        I6(I)=0
        I11(I)=0
        I12(I)=0
        I8(I)=0
        I20(I)=0
        I21(I)=0
        ITRA=I19(I)
        I18(ITRA)=0
    4 I19(I)=0
        DO 11 I=1,NOMBAR
        I12(I)=0
        I13(I)=0
        I14(I)=0
        I15(I)=0
        I16(I)=0
   11 I17(I)=0
    2 CONTINUE
    7 CONTINUE
        STOP
```

```
238     3 FORMAT(1X,2216)
239   120 FORMAT(1X,130L1)
240   122 FORMAT(1X,2216)
241   500 FORMAT(1X,'GRAPHE DE FLUX A ANALYSER',/,1X,25(1H*),//,1X,'PARTIE S
242      1OMMETS DU GRAPHE',/,1X,26(1H*))
243   505 FORMAT(1HC,'PARTIE SUIVANTS DU GRAPHE',/,1X,27(1H*))
244   510 FORMAT(1X,'COMPOSANTES FORTEMENT CONNEXES DU GRAPHE',/,1X,40(1H*))
245   519 FORMAT(1X,'COMP.NUMERO ',I4,/,1X,15(1H-),/)
246   525 FORMAT(1X,'DECOMPOSITION EN NIVEAUX',/,1X,24(1H*))
247   535 FORMAT(1X,'LE NIVEAU ',I4,'COMPORTE LES COMP. NUMERO',/)
248   540 FORMAT(1X,'ANALYSE DU NIVEAU ',I4,/,1X,22(1H*))
249   545 FORMAT(1X,'RECHERCHE D"ENSEMBLE ESSENTIEL MINIMUM DE COMP. NUMERO'
250      1,I5,/,1X,>2(1H-))

251   550 FORMAT(1X,'FORME ENCHAINEE DE LA MATRICE ASSOCIEE A LA COMPOS. :')
252   555 FORMAT(1X,'VECTEURS DE CONVERSION DES NUMEROTATIONS DE SOMMETS:',/
253      1/,1X,'SOMMET ET NUMERO DANS COMP.',/)
254   560 FORMAT(1X,'NUMERO D"UN SOMMET DANS COMP. ET SOMMET :',/)
255   565 FORMAT(1X,'VECTEUR INDIQUANT L"ARC DEBUTANT UNE COLONNE:')
256   570 FORMAT(1X,'VECTEUR INDIQUANT L"ARC DEBUTANT UNE LIGNE:')
257   575 FORMAT(1X,'VECTEUR INDIQUANT LE NOMBRE D"ELEMENTS NON NULS D"UNE C
258      1OLONNE:')
259   580 FORMAT(1X,'VECTEUR INDIQUANT LE NOMBRE D"ELEMENTS NON NULS D"UNE L
260      1IGNE:')
261   585 FORMAT(1X,'VECTEUR INDIQUANT LA COLONNE D"UN ARC:')
262   593 FORMAT(1X,'VECTEUR INDIQUANT LA LIGNE D"UN ARC')
263   595 FORMAT(1X,'VECTEUR INDIQUANT LE SUIVANT COLONNE D"UN ARC:')
264   600 FORMAT(1X,'VECTEUR INDIQUANT LE PRECEDENT COLONNE D"UN ARC:')
265   605 FORMAT(1X,'VECTEUR INDIQUANT LE SUIVANT LIGNE D"UN ARC:')
266   610 FORMAT(1X,'VECTEUR INDIQUANT LE PRECEDENT LIGNE D"UN ARC:')
267   615 FORMAT(1X,'PAS DE SOMMET ESSENTIEL DECOUVERT PAR L"ETAPE 1 DE C.K.
268      1')
269   620 FORMAT(1X,I5,' SOMMETS ESSENTIELS DETECTES PAR L"ETAPE 1 DE C.K.')
270   680 FORMAT(1X,'ETAPE 1 DE C.K.',/,1X,20(1H*))
271   800 FORMAT(1X,'SOUS GRAPHE TOTALEMENT REDUIT. ENSEMBLE ESSENTIEL TROUV
272      1E.')
273   810 FORMAT(1X,'SOUS GRAPHE NON TOTALEMENT REDUIT .  PASSER A L"ETAPE 2
274      1 DE C.K.')
275   820 FORMAT(1X,'PAS DE CIRCUITS PERTINENTS DETECTES')
276   840 FORMAT(1X,'VOICI UNE PARTIE DE LA TABLE DE COUVERTURE. PAS ASSEZ D
277      1E PLACE EN MEMOIRE POUR GENERER TOUS LES CIRCUITS PERTINENTS')
278   850 FORMAT(1X,'PASSAGE A L"ETAPE 3 DE C.K. QUAND ELLE EXISTERA DANS CE
279      1PROGRAMME')
280   860 FORMAT(1X,'TABLE DE COUVERTURE TRANSPOSEE',//)
281   870 FORMAT(1X,'SOMMETS AUXQUELS CORRESPONDENT LES COLONNES',//)
282   900 FORMAT(1X,'LA COMPOSANTE EST UN SOMMET ISOLE')
283       END
```

## IV.C. Résultats.

Nous ne comptons pas donner ici la solution d'un exemple traité par notre programme décrit en IV.B. Nous voulons plutôt donner les conclusions que nous tirons des résultats reçus.
Note : l'ordinateur à l'aide duquel nous avons testé le programme est un "Siemens 4004" . Nous nous référons à cette machine si nous donnons des ordres de grandeur de temps de calcul. Le langage FORTRAN IV dans lequel est écrit le programme n'est pas "standard" et se réfère aussi à la machine utilisée.
La mise au point du programme décrit est délicate :
La littérature à notre disposition proposait peu d'exemple de résultats des parties d'algorithme décrites. L'algorithme du ch.I peut se traiter sans machine pour des graphes de taille p.ex. 50 sommets et 200 arcs. L'algorithme du chapitre II se traite difficilement sans machine même pour des graphes de taille p.ex. 20 sommets et 60 arcs. Ce chapitre nécessite en effet une analyse beaucoup plus détaillée à chaque pas. L'analyse des solutions doit alors être plus fine pour détecter les failles éventuelles.

Nous n'avons pas de commentaires à apporter au déroulement de l'algorithme se rapportant au chapitre I. Trouver les sous-systèmes d'un système à"grande échelle" et le décomposer en niveaux hiérarchiques paraît être un problème résolu. La recherche des sous-systèmes irréductibles s'effectue comme prévu par Tarjan ( cfr. ch. I.2.C.b.)

Les résultats de la seconde partie de l'algorithme ( ch.II étape 1et 2 ) nous rassurent assez peu à propos de l'efficacité de la méthode proposée par Cheung et Kuh.

a - Il est difficilement réalisable du point de vue place occupée en mémoire centrale.
b - Couteux en temps de calcul

Considérons le point a :
Notre but était de constater l'utilité de la méthode de C.K.

pour des graphes de taille "honnête" en nous fixant comme taille 100 sommets et environ 350 arcs ( ce qui pourrait être le cas d'un système d'équations où peu de variables interviennent dans les fonctions décrivant le système ). Nous avons traité 2 exemples de graphe de cette taille.

Dans chaque cas, nous obtenons un"sous-système"de plus de 80 équations. Nous considérons alors un de ces sous-système à traiter par la technique de C.K. Par l'étape 1 de leur algorithme, il n'est pas possible de réduire le sous-graphe à un sous-graphe de moins de 50 sommets.( 54 exactement dans un exemple traité très précisément ) . Pour obtenir cela, nous avons favorisé l'application de la règle 2 de C.K.

L'étape 2 ne parvient pas à générer tous les circuits pertinents par manque de place dans la matrice COV (voir IV.A. ) pourtant dimensionnée 100 X65 00. Des 54 sommets, le 20ème est en cours de traitement au moment de l'arrêt du programme.( création de 306 arcs ).

Constatations :

- Il ne servirait à rien d'évacuer les circuits alors détectés dans des mémoires périphériques pour libérer une partie de la matrice COV. Ces circuits occupent seulement 54 colonnes de la matrice COV.
- Utiliser des listes enchainées pour représenter la matrice COV ? L'analyse du problème de représentation de COV peut nous indiquer quelle proportion d'éléments non nuls doit contenir au maximum la matrice COV pour gagner de la place avec des listes enchainées. Mais nous remarquons que les étiquettes ont une tendance à s'agrandir au point qu'il paraît probable que l'on perde de la place à utiliser des listes enchainées; ceci bien que ces listes nous font gagner de la place si l'on considère que COV a 100 lignes dont 54 sont occupées (54 sommets dugraphe non totalement réduit ).
- 100 X 6500 semble être très proche de la capacité maximale de la mémoire à accès rapide dont nous disposions pour stocker les variables du programme.

Considérons le point b :
Le temps de calcul au moment de constater le manque de place pour stocker les variables est de l'ordre de 2 minutes , ce qui compte tenu de la puissance de la machine est appréciable. N'oublions pas que la table de couverture incomplète ne sert plus à rien.
La conclusion que nous tirons de ces résultats est que la notion de"grande échelle" serait liée à l'algorithme qui traite le problème à grande échelle. Nous ne pensons pas que le fait que le système soit plus "creux" change fondamentalement le comportement de l'algorithme : les simplifications éventuelles devraient alors se produire moins souvent.

Ces conclusions nous mènent cependant à poser la question de savoir si un algorithme moins sophistiqué donnerait de meilleur résultats pratiques. Nous reproposerions l'approche de Kevorkian : appliquer des simplifications préliminaires augraphe, p.ex. l'étape 1 de C.K. et ensuite générer un circuit à partir duquel on procède par séparation en supposant successivement que chaque sommet sur le circuit appartient à l'ensemble minimum essentiel. Répéter alors les simplifications jusque réduction totale ou éventuellement devoir séparer de nouveau.
Pourrait-on alors se servir de l'étape 2 de C.K. pour générer le circuit ?
Beaucoup de travail reste à accomplir dans cette voie si l'on sait qu'une procédure de séparation exige le stockage de résultats intermédiaires en grande quantité généralement. Les résultats de Kevorkian ( cfr. Kev.II ) se limitent aux décompositions de systèmes de l'ordre de 100 à 200 équations(de même que LED. et HIMM.).

## Conclusions

Voici tout d'abord quelques perspectives :

- Nous venons de constater que comparer la méthode de Kevorkian et celle de Cheung et Kuh s'impose.
- Nous poserions une question : est-il possible, à l'aide de l'arbre constuit au cours du depth-first search ( I.2.C.b. ), de proposer un algorithme d'un autre type capable de trouver les ensembles minimaux essentiels de graphe? A notre connaissance il n'existe pas de résultats importants dans ce sens.
- En regard d'un algorithme de résolution de systèmes d'équations à l'aide de la forme triangulaire bordée ( II;2), n'est-il pas plus avantageux d'obtenir des ensembles"presque" minimaux essentiels si le caractère "minimum" de l'ensemble cherché est "difficilement"atteint et si on peut, au préalable, savoir si la recherche de l'ensemble minimum essentiel sera "couteuse"?
- La recherche de paires de sortie se formule sous forme de problème d'affectation avec matrice de couts "creuse"; ceci modifie-t-il la recherche de paires de sortie ?

Nous venons de proposer un algorithme permettant de décomposer un système d'équations nXn à grande échelle. Nous appliquons la théorie des graphes à un graphe de flux.

1- Une recherche de sous-systèmes irréductibles et une décomposition en niveaux hiérarchiques qui ne posent de problèmes particuliers que dans la recherche d'un ensemble de paires de sortie.

2- Une méthode de décomposition plus fine basée sur la recherche d'ensembles minimaux essentiels de graphes dont les problèmes théoriques paraissent résolus.

Un de nos buts étant de construire un programme de décomposition qui fonctionne effectivement, nous avons proposé une manière de construire un programme qui permettrait des décompositions de systèmes de"grande" taille si il n'y avait des problèmes particuliers dans la réalisation de l'algorithme du fait de propagation de données intermédiaires.

Nous montrons alors que la technique de décomposition proposée

a une efficacité pratique toute relative tant du côté de la taille des systèmes qu'elle permettrait de décomposer que du côté de la lenteur de la progression des calculs.
Il reste alors à examiner la technique proposée par Kevorkian dans la recherche des ensembles minimaux essentiels pour constater si, dans l'immédiat, des routines efficaces de décomposition peuvent être envisagées ( Kevorkian étant un technicien d'avant pointe de ces méthodes ). Nous laissons ce travail à nos successeurs.

## REFERENCES.


.BERGE C.,"Théorie des graphes et ses applications",-Collection Universitaire de Mathématiques,Dunod,-Paris 1967.

.CHEUNG L.K.C. ET KUH E.S.,"The bordered triangular matrix and minimum essentiel of a digraph",-IEE Transactions on circuit and systems,Volume C.A.S 21 numéro 5,-1974.

.DEO N.,"Graph theory with applications to engineering and computer science",-Prentice Hall,Englewoods Cliffs N.J.-1974.

.FICHEFET J.,"Cours de théorie des graphes et ses applications" -Institut d'Informatique,FNDP Namur,-1973.

.GARFINKEL R.S. ET NEMHAUSER G.E.,"Integer programming",-John Wiley,New-York,-1972.

.KEVORKIAN A.K. ET SNOEK J.,"Décomposition in large scale sys--tems:Theory and applications of structural analysis in par--titioning,disjointing and constructing hierarchical systems" -Nato Advanced Study Institute on Decomposition as a Tool for Solving Large Scale Problems,Cambridge University,p. 467-490-1972-I

.KEVORKIAN A.K. ET SNOEK J.,idem,p.491-515,-1972-II.

.KNUTH D.E.,"The art of computer programming",-Volume 1,Funda--mental Algorithms,Addison Wesley,Reading,-Mars 1972.

.LEDET W.P. ET HIMMELBLAU D.M.,"Decompcsition procedures for the solving of large scale systems",-Advan. Chem. Eng.,8, p.185,-1970.

.ROSE D.J. ET WILLOUGHBY R.A.,"Sparse matrices and their applications",-New-York-London:Plenum Publishing Corporation, -1972.

.ROY R.,"Algèbre moderne et théorie des graphes",-Tome 1,Coll. Finance et Economie appliquée,Vol. 31,Dunod,Paris,-1969.

.STEWARD D.V.,"on an approach to techniques for the analysis of the structure of large systems of équations",-Siam Review Vol. 4,numéro 4,p. 321,-1962.

.STEWARD D.V.,"Partitioning and tearing systems of equations",
-J. Siam,Numer. Anal.,Ser. B,2,numerc 2,p. 345,-1965.
.TARJAN R.,"Depth first search and linear graph algorithms",-
Siam J.,Comput. Vol. 1,numero 2,p. 146-160,- Juin 1972.
.TEWARSON R.P.,"computations with sparse matrices",-Siam Review
Vol. 12,p. 527-543,-1970.