UNIVERSITE DE NANCY — FACULTE DES SCIENCES

ETUDE D'ALGORITHMES POUR LES PROBLEMES DE CHEMINEMENT DANS LES GRAPHES FINIS

THESE

pour l'obtention du

DU DOCTORAT DE SPECIALITE MATHEMATIQUES (3ème CYCLE)

Soutenue devant le Jury le 20 Décembre 1966

par

Jean-Claude DERNIAME

Jury : Monsieur J. LEGRAS Président
Monsieur M. DEPAIX Examinateurs
Monsieur C. PAIR

UNIVERSITE DE NANCY FACULTE DES SCIENCES

Graphe, cheminement.
Algorithme.

ETUDES D'ALGORITHMES POUR LES PROBLEMES DE CHEMINEMENT DANS LES GRAPHES FINIS.

par

Jean-Claude DERNIAME

UNIVERSITE DE NANCY — FACULTE DES SCIENCES

Doyen : M.AUBRY
Assesseur : M. GAY

Doyens honoraires : MM. CORNUBERT - DELSARTE - URION - ROUBAULT

Professeurs honoraires : MM. CROZE - RAYBAUD - LAFFITTE - LERAY - JULY - LAPORTE - EICHHORN - GODEMENT - DUBREUIL - L.SCHWARTZ - DIEUDONNE - DE MALLEMANN - LONGCHAMBON - LETORT - DODE - GAUTHIER - GOUDET - OLMER - CORNUBERT - CHAPELLE - GUERIN - WAHL.

Maîtres de conférences honoraires : MM. LIENHART - PIERRET.

Professeurs

M. URION	Chimie bio.	* MALAPRADE	Chimie		GARNIER	Agronomie
DELSARTE	Analyse sup.	DUVAL	Chimie		NEEL	Chimie org.indus.
ROUBAULT	Géologie	FRUHLING	Physique		BERNARD	Géologie appli.
CAPELLE	Méca.Ration.	HILLY	Géologie	*	CHAMPIER	Physique
VEILLET	Biol. animale	LE GOFF	Génie chimique	*	GAY	Chimie bio.
BARRIOL	Chimie théor.	SUHNER	Phys.expériment.		STEPHAN	Zoologie
BIZETTE	Physique	CHAPON	Chimie bio.	*	CONDE	Zoologie
GUILLIEN	Electronique	HEROLD	Chimie minér.indus.	*	WERNER	Botanique
GILBERT	Chimie phys.	SCHWARTZ B.	Exploitation minière		EYMARD	Calcul dif.& int.
LEGRAS	Méca.Ration.	MANGENOT	Botanique		LEVISALLES	Chimie organique
BOLFA	Minér. & crist.	GAYET	Physiologie	Mme	HERVE	N.M.P.
NICLAUSE	Chimie	BONVALET	Méca.appliquée	*	GOSSE	Méca.physique
FAIVRE	Phys.appliquée	HADNI	Physique		FELDEN	Physique
AUBRY	Chimie minérale	*BASTICK	Chimie	*	DAVCINE	E.N.S.M.I.M.
COPPENS	Radiogéologie	DUCHAUFOUR	Pédologie			

* Professeur titulaire à titre personnel

Maîtres de Conférences

Mme BASTICK	Chimie MPC	Mlle HUET*	Math. SPCN		JOZEFOWICZ	Physico-chimie
M. GUDEFIN	Physique	MM. VIGNES	Métal.		JURAIN	Géologie SPCN
ROCCI	Géologie	BALESDENT	Thermodynam.chim.		N...	Proba. & stat.
VUILLAUME	Psychophysio.	BLAZY	Minéral.appl.ENSG		N...	Mécanique ISIN
FRENTZ	Biol.animale	JANOT	Phys.MPC (Epinal)		N...	Mathématiques
HORN	Phys.propé.	JACQUIN	Pédol.& chim.agr.		N...	Génie chimique
LAFON	Phys.(ISIN)	MAINARD	Phys. MGP		N...	Math. MPC
MARI	Chimie (ISIN)	CACHAN	Ento.appl.E.SA		N...	Math. SPCN
AUROUZE	Géologie	MARTIN	Chimie MPC		N...	Math. appliquées
DEVIOT	Phys.du solide	PAUMIER	Méca.expérim.		N...	Méca. des fluides
FLECHON	Phys. MPC	PROTAS	Minéralogie		N...	Physiol.animale

* en disponibilité

Secrétaire principal : C.CARON

Je tiens à exprimer ma profonde gratitude à Monsieur le Professeur LEGRAS, Directeur de l'Institut de Calcul Automatique de Nancy, pour la bienveillante sollicitude qu'il n'a cessé de me témoigner au cours de ces deux années.

Ce travail a été effectué sous la direction de Monsieur PAIR, à qui j'exprime ma profonde reconnaissance pour les conseils et l'aide qu'il m'a prodigués.

Je remercie Monsieur DEPAIX, qui a accepté de participer au Jury.

Je tiens également à remercier toute l'équipe de l'Institut de Calcul pour son aide et sa sympathie.

Ce travail a fait l'objet d'une convention de la Délégation Générale de la Recherche Scientifique et Technique.

Convention n° : 65 SR 189.

Ce travail a fait l'objet d'une convention de la Délégation Générale de la Recherche Scientifique et Technique.

Convention n° : 65 SR 189.

SOMMAIRE

INTRODUCTION

La résolution des problèmes de cheminement dans les graphes semble être liée à celle d'un problème plus élémentaire : trouver un chemin entre deux points a et b du graphe.

Dans les "Nouvelles annales de Mathématiques" 14, 1895, l'article de G. Tarry "Le problème des labyrinthes" décrit un algorithme de cheminement que Monsieur Berge résume ainsi :

"Ne jamais parcourir deux fois la même arête dans le même sens ; si on est en x, ne prendre l'arête qui nous a conduit la première fois au carrefour x, que lorsqu'on ne peut pas faire autrement".

Dans son livre "Programme, jeux et réseaux de transports" [5] , Monsieur Berge donne un algorithme très voisin , sous le nom de règle de Trémeaux :

" 1°) On part de a, et l'on suit un chemin quelconque aussi loin que possible ; on marque toujours d'une croix un arc parcouru, et l'on n'utilisera plus jamais un tel arc (du moins dans le sens de son orientation) ;

2°) Si l'on arrive au fond d'une impasse, on rétrograde, en marquant d'une deuxième croix l'arc parcouru en sens inverse ;

3°) Si l'on arrive par un arc non encore parcouru à un sommet déjà exploré, on rétrograde comme si l'on était au fond d'une impasse ;

4°) Si l'on arrive en rétrogradant en un sommet a_i, on repart par un arc non encore utilisé s'il en existe un ; ou sans cela, on rétrograde par l'arc, marqué d'une croix, qui nous a conduit la première fois en a_i ; on arrête la procédure lorsqu'on arrive au sommet b, ou lorsqu'on ne peut plus continuer".

Cette méthode, très simple et connue depuis longtemps donc, est pratiquement tombée dans l'oubli depuis le développement du calcul électronique, par manque d'un outil mathématique apte à le représenter.

L'étude de Monsieur PAIR : "Etude de la notion de pile. Application à l'analyse syntaxique" [44] a précisé cet outil déjà utilisé en théorie des langages : une pile. La formulation mathématique commode qui en est donnée et l'étude des applications à l'analyse des ramifications ont permis, bien que ce ne soit pas l'objet de cet ouvrage, de retrouver sous une forme

exploitable la règle de Trémeaux. Monsieur EMOND [18] en a déjà étudié deux applications sur ordinateur pour le problème de détermination de la fermeture transitive d'un graphe et celui de la recherche des chemins élémentaires. Des essais d'application à d'autres problèmes qui semblaient liés au cheminement ont donné naissance à de nouveaux algorithmes, basés sur la même méthode d'exploitation du graphe : seules différent les opérations à effectuer lors des sorties de la pile. Il restait à étudier cette relation.

On trouvera dans PAIR [45] une étude formelle détaillée de la façon dont ces problèmes sont liés au cheminement. De chacun des algorithmes de cheminement il devient alors possible de déduire immédiatement des algorithmes, nouveaux ou déjà connus, traitant divers problèmes parmi ceux de : recherche du préordre associé à un graphe, de plus courtes distances et de plus courts chemins, de plus longues distances et de plus longs chemins, de nombre de chemins, de chemin le plus problable, de passage d'un point à un autre etc...

Ce travail consiste à faire le point de ces diverses idées reçues, en étudier les applications et les diverses méthodes qui en découlent, et aussi à effectuer des comparaisons entre les algorithmes obtenus. On trouvera également diverses applications de l'algorithme utilisant une pile.

CHAPITRE 1

DEFINITIONS

D.1 GRAPHE KONIG [36]

On appelle graphe un couple (E, Γ) où E est un ensemble de points et Γ une relation binaire dans E.

D.2 MATRICE ASSOCIEE A UN GRAPHE [4]

La matrice associée à une relation binaire R dans E est une matrice carrée booléenne $\mathcal{R}$ dont le terme r_j^i vaut 1 si, et seulement si, $e_i \, R \, e_j$

0 sinon , pour tout e_i, $e_j \in E$.

La matrice associée au graphe (E, Γ) est la matrice associée à la relation Γ .

On notera $\Gamma(x)$ l'ensemble $\{ y, x \Gamma y \}$ et $\bar{\Gamma}$ la négation de Γ.

D.3 Fermeture transitive

La fermeture transitive de la relation Γ est une application $\hat{\Gamma}$ de X dans X définie par

$$\hat{\Gamma}(x) = \{x\} \cup \Gamma(x) \cup \Gamma^2(x) \cup \Gamma^3(x) \cup \cdots$$

Le produit des relations ayant le sens habituel en algèbre.

L'application

$$\Gamma(x) \cup \Gamma^2(x) \cup \Gamma^3(x) \ldots$$ est appelée fermeture transitive stricte de la relation Γ .

Note : $\hat{\Gamma}$ est le préordre associé au graphe.

D.4 RELATION D'EQUIVALENCE

associée à un graphe, classe d'équivalence définie par le préordre Γ .

$$x \sim y \iff x \hat{\Gamma} y \text{ et } y \hat{\Gamma} x.$$

quand, dans la suite, on parlera de classe d'équivalence, il s'agira d'une classe pour cette relation. C'est aussi une composante fortement connexe du graphe [4]

D.5 ANCETRE, DESCENDANT, PREDECESSEUR, SUCCESSEUR

Si $x \hat{\Gamma} y$ on dit que x est un ancêtre de y et y est un descendant de x.

Si $x \Gamma y$ on dit que x est un predecesseur de y et y un successeur de x.

D.6 FEUILLE, NOEUD, RACINE

Si $\Gamma(x) = \emptyset$ x est appelé feuille ou point de sortie

Si $\Gamma(x) \neq \emptyset$ x est appelé noeud

Si $\Gamma^{-1}(x) = \emptyset$ x est appelé racine ou point d'entrée.

D.7. CHEMINS, CIRCUITS D'UN GRAPHE (E, Γ) BERGE [4]

Un couple (xy) tel que $x \Gamma y$ est appelé arc du graphe. On dit que le sommet a est son extremité initiale (ou origine) et que b est son extremité terminale.

Si $x = y$ (xy) est une boucle.

On appelle chemin d'un graphe (E, Γ) une séquence (μ_1, μ_2,..) d'arcs telle que l'extremité terminale de chaque arc coïncide avec l'extremité initiale de l'arc suivant. Un chemin est simple s'il n'utilise pas deux fois le même arc et composé dans le cas contraire.

Si un chemin rencontre successivement les sommets $x_1, x_2, \ldots, x_k, x_{k+1}$, on peut aussi le noter $\mu = [x_1, x_2, \ldots, x_k, x_{k+1}]$; le point x_1 sera eppelé origine du chemin μ, x_{k+1} extremité du chemin μ. Un chemin qui ne rencontre pas deux fois le même sommet est dit élémentaire ; un chemin peut être fini ou infini. Un circuit est un chemin fini $\mu = [x_1, x_2, \ldots, x_k]$ dans lequel le sommet initial x_1 coïncide avec le sommet terminal x_k ; Un circuit sera encore dit élémentaire si tous les sommets qu'il rencontre sont distincts (excepté le sommet initial et le sommet terminal qui coïncident).

D.8 GRAPHE COMPLET

Un graphe (E, Γ) est dit complet si

$(\forall x \in E)\ (\forall y \in E)\quad (x \Gamma y \Longrightarrow y \Gamma x.)$

D.9 GRAPHE ANTISYMETRIQUE

Un graphe (E, Γ) est dit antisymétrique si

$(\forall x \in E)\ (\forall y \in E)\quad (x \Gamma y \text{ et } y \Gamma x \Longrightarrow x = y)$

D.10 GRAPHE SYMETRIQUE

Un graphe (E, Γ) est dit symétrique si

$(\forall x \in E)$

D. 11 MONOÏDE LIBRE

Une suite finie $\alpha = [a_0, a_1, \ldots, a_n]$ d'éléments d'un ensemble E est appelé mot sur E ; $n+1 = |\alpha|$ est sa longueur. Si $a_i = b$ nous dirons que i est une occurence de b dans α et que b possède une occurence dans α.

L'ensemble E^* des mots sur l'ensemble E est muni d'une loi de composition interne : concaténation, telle que

$$(a_0, a_1, \ldots a_n) \cdot (b_0, b_1, \ldots b_p) = (a_0, \ldots, a_n, b_0, b_1, \ldots b_p)$$

E^* est un monoïde. On dit que c'est le monoïde libre déduit de E.

Le mot vide, noté $\wedge$ est élément neutre de E^*.

Si $\alpha, \beta, \gamma, \delta$ sont des mots tels que

$$\alpha = \beta . \delta . \gamma$$

nous dirons que β est facteur gauche, γ facteur droit de α.

Exemple : Un chemin d'un graphe est un élément du monoïde libre déduit de l'ensemble des arcs du graphe. L'élément neutre de ce monoïde est Θ chemin vide. Nous conviendrons que pour tout point x du graphe e joint x à x.

D. 12 PILE . PAIR [44])

On appelle pile sur l'ensemble E toute suite finie $U = (u_0, u_1, \ldots, u_n)$ d'éléments du monoïde libre E^* telle que

1) $u_0 = u_n = \wedge$

2) pour $i = 1, 2, \ldots n$

- u_{i-1} est facteur gauche de u_i et $|u_i| = |u_{i-1}| + 1$

ou - u_i est facteur gauche de u_{i-1} et $|u_i| = |u_{i-1}| - 1$.

D.13 SOMMET DE LA PILE PAIR [44]

$u_0, u_1, \ldots, u_n$ sont les états de la pile et le dernier élément de E dans le mot u_i est le sommet de l'état u_i de la pile

Exemples :

$$E = \{a, b, c, d\}$$

1) u_0 = e
u_1 = a
u_2 = a b
u_3 = a b c
u_4 = a b
u_5 = a
u_6 = a c
u_7 = a
u_8 = e

2) u_0 = e
u_1 = a
u_2 = a b
u_3 = a b c
u_4 = a b c d
u_5 = a b c
u_6 = a b
u_7 = a b c
u_8 = e

dans 1) u_3 = a b c est un état de la pile c est le sommet de cet état de la pile.

D.14 ENTREE, SORTIE DE LA PILE

a) u_{i-1} est facteur gauche de u_i et $|u_i| = |u_{i-1}| + 1$ signifie qu'il existe $e \in E$ et $u_i = u_{i-1} \cdot e$. On dit que i est une entrée de e.

b) u_i est facteur gauche de u_{i-1} et $|u_i| = |u_{i-1}| - 1$ signifie qu'il existe $e \in E$ et $u_{i-1} = u_i \cdot e$ On dit que i est une sortie de e.

Dans l'exemple 1 (D. 13)
1 est une entrée de a, 3 une entrée de c, b une autre entreé de c, 7 une sortie de c.

Dans l'exemple 2 1,2,3,4 sont des entrées
5,6,7,8 sont des sorties.

D.15 PILE SIMPLE

Une pile U sur un ensemble fini E est simple lorsque tout élément de E possède une entrée et une seule dans U.

Dans D. 13 l'exemple 2 est une pile simple, mais non l'exemple 1. (c a deux entrées).

D.16 PILE ATTACHEE A UN GRAPHE (E, Γ) [PAIR 45]

C'est une pile simple sur l'ensemble des arcs du graphe (E U $\{r\}$, Γ). Les états de la pile sont des chemins du graphe (les chemins du graphe sont des éléments du monoïde libre déduit de l'ensemble des arcs, cf(D. 10).

On a ajouté à l'ensemble E un point r tel que $(\forall x \in E)$ $r \Gamma x$ et $x \bar{\Gamma} r$, ceci pour qu'aucun point du graphe (E, Γ) ne soit origine d'un des états de la pile, c'est-à-dire d'un chemin. Une pile attachée à un graphe (E, Γ) est donc une pile simple sur l'ensemble des arcs de (E U $\{r\}$, Γ) telle que :

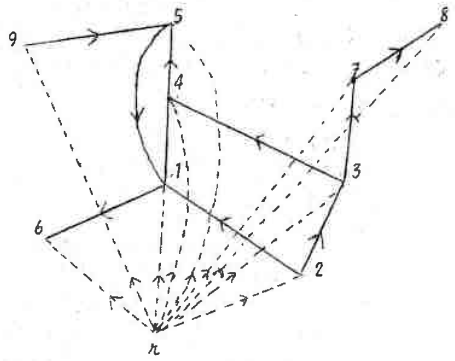
a) si $u_{i-1} = e$:
s'il existe un point x de E tel qu'aucun arc d'extrémité x n'ait d'entrée inférieure à i, i est l'entrée de l'arc joignant r à x ; sinon a_{i-1} est le dernier état de la pile

b) si i-1 est l'entrée de α :
s'il existe des arcs dont l'origine est l'extrémité de α et si l'entrée d'aucun d'eux n'est inférieure à i, i est l'entrée d'un tel arc ; sinon i est la sortie de α.

c) si i-1 est la sortie de α et $u_{i-1} \neq e$:
s'il existe un arc de même origine que α qui n'a pas d'entrée inférieure à i, i est l'entrée d'un tel arc ; sinon i est la sortie de α.

Exemple :

e
r1.
r1.14
r1.14.45
r1.14.45.51
r1.14.45
r1.14
r1.
r1.16
r1.



La suite ci-contre est une des piles attachées au graphe représenté ci-dessus Une autre pile attachée pourrait être obtenue en faisant entrer d'abord r2, par exemple, au lieu de r1.

|
e
r2
r2.21
r2.
r2.23
r2.23.34
r2.23
r2.23.37
r2.23.37.78
r2.23.37
r2.23
r2
e
r9
r9.95
r9.
e

D.17 PILE ANNEXE A UN GRAPHE [EMOND 18]

C'est une pile $V = (a_0, a_1, \dots a_n)$ sur l'ensemble E des points du graphe, telle que l'on passe de l'état v_{i-1} à l'état v_i par le processus suivant :

a) si $v_{i-1} = \wedge$:
alors i est l'entrée du premier $z \in E$ n'ayant pas d'entrée inférieure à i, s'il existe, sinon v_{i-1} est l'état final de V.

b) si $v_{i-1} \neq \wedge$ et a pour sommet x :

1) si $i-1$ est une sortie de y :
- si y n'est pas le dernier élément de $\Gamma(x)$, alors i est une entrée de z qui suit y dans $\Gamma(x)$.
- si y est le dernier élément de $\Gamma(x)$, i est une sortie de x.

2) si $i-1$ est la première entrée de x :
- si $\Gamma(x) = \emptyset$, i est une sortie de x.
- si $\Gamma(x) \neq \emptyset$, i est une entrée du premier élément de $\Gamma(x)$.

3) si $i-1$ est une autre entrée de x :
alors i est une sortie de x.

Note : Nous avons vu (cf D.7) qu'un chemin défini comme une suite d'arcs peut aussi être représenté par la suite des points qu'il rencontre. On peut donc obtenir une pile annexe à un graphe en remplaçant tout état d'une pile attachée à ce graphe par la suite des points du chemin (sauf le point r) qui constitue cet état.

Exemple :

Décrivons la pile annexe au graphe représenté en D.16 déduite de la pile attachée citée :

				1														8								
			5	5	5											4		7	7							
		4	4	4	4	4		6				1		3	3	3	3	3	3	3				5		
∧	1	1	1	1	1	1	1	1	1	∧	2	2	2	2	2	2	2	2	2	2	2	∧	9	9	9	∧

D.18 ENTREE ET SORTIE VRAIE DE LA PILE ANNEXE :

On appelle entrée vraie de x et on note $\varepsilon(x)$ la première entrée de x dans la pile V. (entrée de la pointe vers x dans la pile attachée [45]).

On appelle sortie vraie de x et on note $\Sigma(x)$ l'instant i tel que $V_{\varepsilon(x)} = V_{i-1}$ et i est une sortie.

D.19 FERMETURE DE CIRCUIT :

Si un point x possède une sortie inférieure à sa sortie vraie, on dira que x est fermeture de circuit.

Cette notion est liée au choix de la pile attachée à un graphe (ou de la pile annexe).

Exemple : Pour la pile annexe décrite en D.18 pour le graphe de D.17, 1 est fermeture de circuit car v_6 est une sortie de 1 inférieure à sa sortie vraie (v_{11}).
Par contre pour le même graphe, et pour la pile annexe suivante :

```
          5                       8
     6   4 4 4               4   7 7 7
   1 1 1 1 1 1 1         1   3 3 3 3 3 3 3         5
 ^ 5 5 5 5 5 5 5 5 5 ^ 2 2 2 2 2 2 2 2 2 2 ^ 9 9 9 ^
```

c'est le point 5 qui est premier de circuit (pour le même circuit).

D.19 DIFFERENTES REPRESENTATIONS DU GRAPHE

On peut associer aux différents algorithmes des représentations différentes du graphe. Nous avons défini en D.2 la matrice associée au graphe.

D.19.1 MATRICE D'INCIDENCE : [BERGE 4]

Désignons par u_1, u_2, u_m les arcs d'un graphe G, par x_1, x_2,...,x_n ses sommets, et posons :

$$s_j^i = \begin{cases} +1 & \text{si } x_i \text{ est l'extrémité initiale de } u_j. \\ -1 & \text{si } x_i \text{ est l'extrémité terminale de } u_j \\ 0 & \text{si } x_i \text{ n'est pas extrémité de } u_j. \end{cases}$$

$S = (s_j^i)$ s'appelle la matrice d'incidence aux arcs du graphe G. L'encombrement en mémoire est de $M \times N$, N étant le nombre de sommets M celui des arcs.

D. 19.2 FILE DES SUCCESSEURS DES DIFFERENTS SOMMETS :

A chaque sommet du graphe on associe une partie de cette file contenant les noms de tous les successeurs du sommet envisagé. Cette technique est bien adaptée à la représentation des graphes sans circuits. L'encombrement en mémoire est seulement de N+M, N étant le nombre de sommets M celui des arcs.

D.19.3 DOUBLE FILE DES SUCCESSEURS ET DES PREDECESSEURS DES DIFFERENTS SOMMETS :

A chaque sommet du graphe est associé une partie de cette file contenant d'une part la suite des noms des successeurs de ce sommet, d'autre part la suite des noms de ses predecesseurs. L'encombrement en mémoire est alors de 2N + 2M. Notons qu'on peut aussi structurer ces files en listes pour obtenir une représentation par des listes de successeurs des sommets ou par des doubles listes de successeurs et de prédecesseurs.

CHAPITRE II

ALGORITHMES ETUDIES

Soit F le monoïde libre déduit de l'ensemble des arcs du graphe, . la concaténation dans F (D. 10), e l'élément neutre de F pour. - ou chemin vide - F contient les chemins du graphe (D. 10, exe- exemple).

Appelons F^* l'ensemble des suites finies de chemins, c'est-à-dire le monoïde libre déduit de F, $\cup$ la loi de composition interne dans F^* définis par :

$$(a_1, a_2, \ldots, a_n) \cup (a'_1, \ldots a'_p) = (a_1, \ldots, a_n, a'_1, \ldots, a'_p),$$

et $\wedge$ l'élément neutre de F^* pour $\cup$. ($\wedge$ est la suite vide).

Définissons encore :

$\odot$ produit, par

$$\begin{cases} (a_1, \ldots, a_n) \odot (a'_1, \ldots, a'_p) = (a_1.a'_1, a_1.a'_2, \ldots, a_1.a'_p, \\ \qquad\qquad a_2.a'_1, \ldots a_n.a'_p) \\ (\forall\ K \in F^*)\ \ (K \odot \wedge = \wedge \odot K = \wedge) \end{cases}$$

E un ensemble muni de deux lois de composition interne + et $\times$ et Π un homomorphisme de $(F^*, \cup, \odot)$ dans $(E, +, \times)$:

$$\Pi(K \cup K') = \Pi(K) + \Pi(K') \ ; \ \Pi(K \odot K') = \Pi(K) \times \Pi(K').$$

Alors [45], de tout algorithme de construction de suites de chemins, procédant par réunions et produits, on peut déduire un algorithme de construction de leurs transformés par Π : il suffit de traduire $\cup$ en + et $\odot$ en $\times$

I - PROBLEMES POUVANT ETRE AINSI ABORDES :

Nous nous bornons ici à une énumération de ces problèmes, on pourra trouver une étude plus approfondie dans [PAIR 45].

1) Existence de chemins

E = {VRAI, FAUX} ; si $K \in F^*$, $\Pi(K)$ = VRAI si, et seulement si, $K \neq \wedge$; $+ = \cup$, $\times = \wedge$.

2) Nombre de chemins

E = ensemble des entiers naturels, $\times$ et + sont la multiplication et l'addition usuelles.

3) Recherche de chemins et circuits élémentaires.

$E = F^*$, + réunion dans E, × défini par

$L \times L = \Pi(L \odot L)$.

$\Pi(K)$ associe à toute suite K, la sous suite de ses chemins (resp. chemins et circuits) élémentaires.

4) Recherche de chemins et circuits vérifiant un condition C.

Il faut alors faire l'hypothèse :

$C(a.a') \Longrightarrow C(a) \wedge C(a')$

où C(a) signifie "le chemin a vérifie la condition C".

5) Recherche de plus courts chemins

Chaque arc du graphe possède une longueur qui est un nombre réel ; on définit la longueur $|a|$ d'un chemin a comme la somme des longueurs de ses arcs, le chemin vide e ayant pour longueur 0. Nous conviendrons que, dans une suite K de chemins, a est plus court que b si $|a| < |b|$ ou si $|a| = |b|$ mais une occurrence de a précède la première occurrence de b.

E est ici l'ensemble F auquel nous adjoignons un élément $\omega = \Pi(\wedge)$, en posant $|\omega| = \infty$.

$a + b =$ SI $|a| \leq |b|$ alors a sinon b avec $\ell < \infty$ pour tout réel ℓ.

$a \times b = a.b$ en convenant que $a.\omega = \omega.a = \omega$.

Remarque :

Lorsqu'on cherche le plus court chemin (ou plus long, ou plus probable..) de tout point x à un point y donné, il suffit de connaître pour chaque x, le second point z du chemin car l'arc xz doit être suivi du plus court chemin de z à y.

Cette remarque permet de réduire considérablement l'espace nécessaire au stockage.

6) Recherche des p plus courts chemins :

. p = 2.

E = ensemble des couples (a,b) d'éléments de $F \cup \{\omega\}$ tels que $|a| \leq |b|$ et $(a \neq b$ ou $a = b = \omega)$;

$(a,a') + (b,b') = \Pi(a,a',b,b')$

$(a,a') \times (b,b') = \Pi\Pi(a.b, a.b', a'.b)$

où Π associe à toute suite K de chemins généralisés [cf 45] le couple de son plus court élément et de son second plus court, si K contient deux éléments distincts, et (a, ω) si K ne contient que a, (ω, ω) si K est vide.

On peut généraliser de la même façon à p plus courts chemins.

7) Plus courte distance.

$E = R^+ \cup \{\infty\}$ (on pourra étendre dans certains cas à R)

$a+b = Min\ (a,b)$

$a \times b = |a| + |b|$

8) Plus longue distance

$E = R^+ \cup \{-\infty\}$

$a+b = max\ (a,b)$, $a \times b = |a| + |b|$

9) Probabilité maximum d'un chemin :

$E = [0,1] \cup \{-\infty\}$; $a+b = max(a,b)$; $a \times b = a \times b$

10) Probabilité minimum d'un chemin :

$E = [0,1] \cup \{+\infty\}$; $a+b = min\ (a,b)$; $a \times b = a \times b$.

ceci n'a de sens que pour des graphes sans circuits.

11) Probabilité de passage d'un point à un autre :

$E = [0,1]$, + et × sont les opérations usuelles

II - ALGORITHMES DE RECHERCHE DE CHEMINS :

Nous utiliserons les notations suivantes définissant certaines suites de chemins associées aux couples x,y de points de E :

$I(x,y) =$ SI $x=y$ ALORS (e) SINON $\wedge$

$J(x,y) =$ SI $x \Gamma y$ ALORS (arc xy) SINON $\wedge$

$J'(x,y) = I(x,y) \cup J(x,y)$.

Algorithme 1 : Chemins de p arcs de x fixé à y quelconque

- $K(x,y,o) = I(x,y)$ pour tout $y \in X$
- pour $i=1,2,\ldots,p$: $K(x,y,i) = \bigcup_{z \in x} [K(x,z,i-1) \odot J(z,y)]$ pour tout $y \in X$

Cet algorithme demande $pn^2 \odot$ et pn^2 opérations $\cup$.

On peut écrire un algorithme analogue pour chercher les chemins de p arcs de x quelconque à y fixé.

Algorithme 2 : Chemins de x fixé à y quelconque, dont le nombre d'arcs est compris entre p_1 et p.

- $K'(x,y,p_1) = K(x,y,p_1)$ pour $y \in X$; (fourni par l'algorithme 1)
- (pour $i=p_1+1, p_1+2, \ldots p$) ($\forall y \in X$) ($K'(x,y,i) = \bigcup_{z \in X} \left[K'(x,z,i-1) \odot J'(z,y)\right]$)

En appliquant cet algorithme au problème du plus court chemin on obtient l'algorithme de Bellman-Kalaba [cf 3] : pour la recherche de la longueur du plus court chemin d'au plus p arcs joignant x à y :

- $k(x,y,o) = SI\ X=Y$ ALORS 0 SINON ∞
- (pour $i=1,2,..,p$) $(k(x,y,i) = \min_{x \in X} \left[k(x,z,i-1) \oplus j'(z,y)\right]$

avec $j'(z,y) = \begin{cases} \min(0, |arc\ zz|) & si\ z=Y \text{ et } Z \Gamma Z \\ 0 & si\ z = Y \text{ et } (non\ z \Gamma z) \\ |arc\ z\ y| & si\ z \neq y \text{ et } z \Gamma y \\ \infty & si\ z \neq y \text{ et } (non\ z \Gamma y) \end{cases}$

En remarquant que :

$$\bigcup_{z \in X} \left[K'(x,z,i-1) \odot J'(x,y)\right] = \bigcup_{z \in X} K'(x,z,i-1) \odot \left[I(z,y) \cup J(z,y)\right] = K'(x,y,i-1) \cup \left[\bigcup_{z \in X} K'(x,z,i-1)\right] \odot J(z,y)]$$

on peut écrire, pour le même problème (longueur généralisée) l'algorithme :

$k(x,y,o) = SI\ x=y$ ALORS 0 SINON ∞

$(i=1,..p)(k(x,y,i) = \min [k(x,y,i-1), \min_{z \in X} [k(x,z,i-1) \oplus j(z,y)]$

avec $j(z,y) = SI\ z \Gamma y$ Alors $|arc\ xy|$ SINON ∞ .

Il s'agit de l'algorithme de Ford [cf 20] .

Les suites obtenues peuvent présenter des répétitions : l'algorithme 2 est donc inutilisable pour le calcul de nombre de chemins de x à y ou de la probabilité de passage de x à y.

Algorithme 3 : Chemins de 2^q arcs de tout point x à tout point y.

- $K(x,y,o) = J(x,y)$ pour $x \in X$ et $y \in X$;
- (pour $i=1,2,\ldots,q$) (pour $x \in X$, $y \in X$)

$(K(x,y,i) = \bigcup_{z \in X} \left[K(x,z,i-1) \odot K(z,y,i-1)\right]$)

Pour étudier les chemins de p' arcs, p' n'étant pas une puissance de 2, on pourra combiner les algorithmes 1 et 3.

En posant $2^q=p$, les algorithmes issus de celui-ci demandent $n^3 \log_2 p$ multiplications et $n^2 (n-1) \log_2 p$ additions. Nous verrons (cf chapitre 10) que pour les graphes symétriques on peut ramener ces nombres à $\frac{n^2 (n+1)}{2} \log_2 p$ et $\frac{n (n^2-1)}{2} \log_2 p$.

Algorithme 4 : Chemins d'au plus 2^q arcs de tout point x à tout point y.

- $K(x,y,o) + J'(x,y)$ pour $x \in X$ et $y \in X$;
- (pour $i=1,2,\ldots,q$) (pour $x \in X$, $y \in X$)
  $(K(x,y,i) = \bigcup_{z \in X} K(x,z,i-1) \odot K(z,y,i-1))$

$K(x,y,i)$ est formé, avec répétitions, des chemins d'au plus 2^i arcs joignant x à y, y compris le chemin vide si x=y. On peut faire les mêmes remarques que pour l'algorithme 3 dans le cas des graphes symétriques ou dans le cas où p', nombre d'arcs, n'est pas une puissance de 2.

Algorithme 5 : Suites contenant les chemins et circuits élémentaires de tout point x à tout point y.

On ordonne les points du graphe : $z_1, z_2, z_3, \ldots, z_n$.

- $(\forall x \in X)$ $(\forall y \in X)$ $(K[x,y,o] = J'[x,y]$
- (pour $i=1,2,..,n$) $(\forall x \in X$ et $\forall y \in X)$
  $(K(x,y,i) = K(x,y,i-1) \cup \left[K(x,z_i,i-1) \odot K(z_i,y,i-1)\right]$)

Appliqué au problème de la recherche du préordre associé à un graphe on retrouve la méthode exposée dans [WARSHALL,58] . On peut cependant trouver dans [PAIR,45] une démonstration beaucoup plus simple : "Par récurrence sur i, $K(x,y,i)$ contient tous les chemins et circuits élémentaires joignant x à y et ne passant par aucun point d'indice supérieur à i, sauf peut-être x et y (il contient aussi d'autres chemins) : en effet, un tel chemin passe par z_i, il s'écrit a. a', où a joint x à z_i, a' joint z_i à y, ni a ni a' ne passant par un point d'indice supérieur à i-1 autre que ses

extrémités, puisque a.a' est élémentaire. K(x,y,n) contient donc les chemins et circuits élémentaires de x à y .

Le fait que K(x,y,i) ne contienne pas que des chemins et circuits élémentaires empêche d'utiliser cet algorithme pour certains des problèmes posés (nombre de chemins, chemins élémentaire, etc). Par contre si le graphe est sans circuits, on pourra l'appliquer à tous les problèmes.

Ces algorithmes demandent n^3 multiplications et n^3 additions. On peut ramener ces nombres à $\frac{n^2\,(n+1)}{2}$ dans le cas des graphes symétriques [cf chap.10].

Il est possible encore de diminuer notablement ce nombre en utilisant la variante suivante :

- $(\forall\ x\in X)\ (\forall\ y\in X)\ (K\ [x,y,o] = J'\ [x,y]\ )$
- (pour $i=1,2,\ldots,n$) $(\forall\ x)$
  (SI $K(x,z_i,i-1) \neq \wedge$ ALORS

$(\forall\ y\in X)\ (K\ [x,y,i] = K(x,y,i-1) \cup [K(x,z_i,i-1) \odot K(z_i,y,i-1]$

le test $K(x,z_i,i-1) \neq \wedge$ permet d'éviter lorsqu'il répond 'NON', n opérations.

On verra au chapitre VII, l'importance de cette modification.

Algorithme 6 : Nouvel algorithme pour la recherche des chemins et circuits entre tout couple de points du graphe.
Il sera décrit au chapitre III.

Algorithme 7 : Chemins de tout point x à un point y fixé, dans le cas des graphes sans circuits.
Cet algorithme utilise une pile et sera décrit au chapitreIII.

Algorithme 8 : Chemins de tout point x à tout point y du graphe.
Cet algorithme utilise une pile et sera décit au chapitre IV.

Les algorithmes décrits permettent de constuire des suites de chemins contenant les chemins et circuits élémentaires mais aussi éventuellement d'autres chemins. Ils pourront donc être utilisés pour les problèmes ne nécessitant pas d'écarter les chemins non élémentaires : plus courtes distances et plus courts chemins, plus longues distances, matrice de fermeture transitive etc.. mais ne pourront être utilisés pour les problèmes de recherche de nombre de chemins ou de probabilité de passage que si les suites constuites sont sans répétition.

CHAPITRE III

ALGORITHMES 6 ET 7 UTILISANT UNE PILE.

I - ALGORITHME 6 : CHEMINS DE TOUT POINT x A UN POINT y FIXE D'UN GRAPHE SANS CIRCUIT.

I-1 METHODE :

Soit Γ' la relation définie dans l'ensemble des arcs du graphe par :

$\alpha\,\Gamma'\beta$ si, et seulement si, l'extrémité de α est l'origine de β . Dans la suite m désigne le nombre d'arcs du graphe. Supposons les arcs ordonnés en $\alpha_1, \alpha_2, \ldots \alpha_m$ de manière que

$$(1)\quad \alpha_i\,\Gamma'\alpha_j \Longrightarrow i > j$$

ce qui n'est évidemment pas possible si le graphe possède un circuit. Désignons par w_i l'origine de α_i et z_i son extrémité. Alors l'algorithme suivant donne les chemins de tout point x à un point y :

- $K(x,y,o) = I(x,y)$ pour $x\in X$
  (I défini au chapitre II)
- pour $i = 1,2,\ldots,m$.
  $\begin{cases} K(w_i,y,i) = K(w_i,y,i-1) \cup [I(\alpha_i) \odot K(z_i,y,i-1)] \\ K(x,y,i) = K(x,y,i-1) \text{ pour } x \neq w_i. \end{cases}$

$K(x,y,i)$ est une suite sans répétition formée de chemins joignant x à y (récurrence).

Montrons par récurrence sur i que $K(x,y,i)$ contient tout chemin a de x à y, vide ou dont le premier arc est α_k avec $k \leq i$:

si $a = e$ ou $k < i$ a est dans $K(x,y,i-1)$;

si $k = i$, $x = w_i$, $a = \alpha_i.a'$ dans lequel a' est vide ou commence par α_j tel que $\alpha_i\,\Gamma\,\alpha_j$ $(j \leq i-1)$.

Donc a' appartient à $K(z_i,y,i-1)$ et a appartient à $K(w_i,y,i)$ c'est-à-dire $K(x,y,i)$.

Pour écarter le chemin vide il suffit de modifier l'initialisation en $K(x,y,o) = J(x,y)$ (cf chap. 2)

On obtient de même un algorithme qui donne les chemins de x fixé à y quelconque :

$$\begin{cases} - K(x,y,o) = I(x,y) \text{ pour } y \in X \\ - \text{pour } i = m, m-1, \ldots, 2, 1 \\ \quad \begin{cases} K(x,z_i,m-i+1) = K(x,z_i,m-i) \cup [K(x,w_i,m-i) \odot (\alpha_i)] \\ K(x,y,m-i+1) = K(x,y,m-i) \text{ pour } y \neq z_i. \end{cases} \end{cases}$$

On en déduit des algorithmes demandant m multiplications et m additions : comme m, nombre d'arcs est au plus égal, dans un graphe sans circuit à $C_n^2 = \frac{n(n-1)}{2}$, et en général bien inférieur ces algorithmes lorsqu'ils s'appliquent, sont meilleurs que tous ceux qui précèdent.

Notons que si on les applique pour tous les couples x,y du point du graphe, ils nécessitent m n, soit $\frac{n^2 \times (n-1)}{2}$ opérations. On trouvera au chapitre VII, les algorithmes correspondant à la recherche des plus longs chemins et des comparaisons aux autres algorithmes, et au chapitre XI, une application aux réseaux PERT dans laquelle on ne détermine l'ordre des points qu'une seule fois et où on utilise les deux formes citées ci-dessus.

II - DETERMINATION D'UN ORDRE DES ARCS SATISFAISANT A L'HYPOTHESE (1).

Encore faut-il que les arcs soient ordonnés comme il a été dit. Pour cela on utilisera l'ordre de sortie des arcs de la pile attachée définie en D.16.

Il est démontré dans PAIR [45] que l'ordre de sortie des arcs de la pile attachée répond à l'hypothèse (1). Soit $\sigma_1, \sigma_2, \sigma_m$ la suite des sorties de la pile.

Le nombre d'opérations nécessaires pour construire la pile est proportionnel au nombre d'arcs ; si le nombre de points du graphe est grand et si on traite tous les couples de points du graphe, il devient négligeable vis à vis du nombre de multiplications et d'additions.

III - UTILISATION DE LA PILE ANNEXE :

En remplaçant tout état d'une pile attachée à un graphe par la suite des poinst de E situés sur ce chemin, on obtient une pile sur l'ensemble E : la pile annexe définie en D.17.

Chacune de ces deux piles détermine l'autre. Si les arcs sont ordonnés en $z_1, z_2, \ldots, z_n$ de manière que $z_i \sqcap' z_j$ entraine $i > j$ il suffit d'ordonner les arcs suivant leur origine pour satisfaire l'hypothèse (1).

Les algorithmes utilisant la pile annexe semblent plus faciles à mettre en oeuvre quand ils sont exploités immédiatement. Cependant si on désire déterminer l'ordre des arcs pour effectuer plusieurs exploitations, ou pour une exploitation differée on doit utiliser la pile attachée.

On obtient avec la pile annexe l'algorithme suivant :

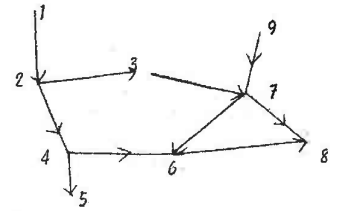
1) $K(x,y,o) = I(x,y)$
   c'est-à-dire = arc (x,y) si $x \sqcap y$
2) si σi est une sortie de z. e le sommet
   $K(e,y,i) = K(e,y,i-1) \cup [(e,z) \odot K(z,y,i-1)]$

On trouvera au chapitre VII, l'application de cet algorithme au problème du plus long chemin .

IV - EXEMPLE

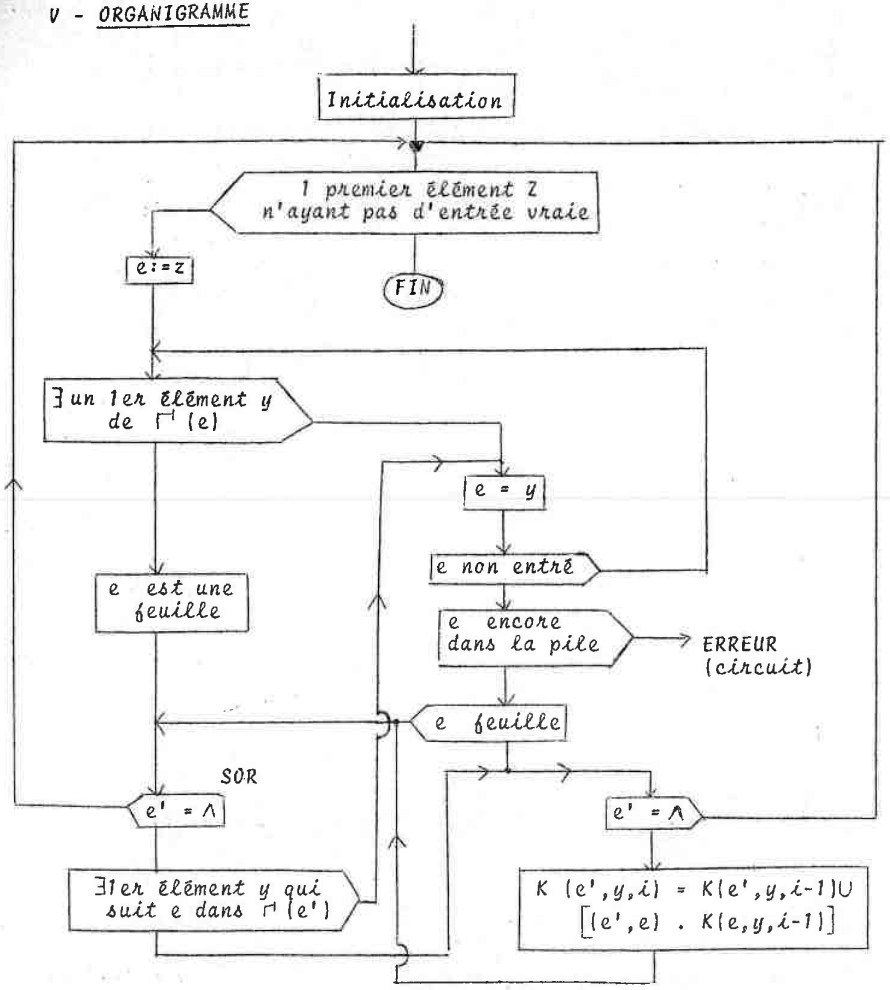
Pour le graphe représenté ci-dessous une pile annexe répondant à la définition est la suivante :

```
^
1
12
123
1237
12376
123768
12376
12378
1237
123
12
124
1245
124
1246
124
12
1
^
97
9
^
```



l'ordre de sortie vraie des points de la pile est alors le suivant :
8,6,7,3,5,4,2,1,9
l'ordre des arcs sur lesquels portent les opérations :
68,76,37,23,45,46,24,12,97.

V - ORGANIGRAMME



VI - PROGRAMMATION

Le graphe est représenté sous forme de sa matrice associée (tableau A).

La pile est un tableau à une dimension P $[1:N+1]$;

La position du sommet est repérée par son indice k. Comme il ne peut exister simultanément dans un état deux arcs ayant même origine, et comme on obtient la pile annexe en remplaçant tout état de la pile attachée par la suite des points de E situés sur ce chemin, il ne peut exister simultanément deux occurrences du même sommet dans la pile.

L'existence de deux occurrences du même sommet dans la pile caractérise un circuit (cf chap. IX) donc une erreur.

Les situations des sommets (non entre, entré, sorti, feuille) sont caractérisés par un indicateur (tableau T $[1:N]$).

pour lequel		signifie	
	T(I) = 1	signifie	I non encore entré
	T(I) = 2		I déjà entré, non sorti
	T(I) = 3		I est sorti et n'est pas une feuille
	T(I) = 4		I est sorti et est une feuille

La distinction entre "feuilles" et non "feuilles" permet d'éviter certaines opérations ($\cup$ et ;) ce qui peut être interessant si ces opérations sont longues et s'il existe beaucoup de feuilles, ou si on veut les chercher. (cf chap. VII).

L'opération "Initialisation" consiste donc à donner les valeurs :

$k = 1$, (initialise la pile à son état $\wedge$)

$T[I] = 1 \quad \forall I$ (aucun point n'est entré).

On trouvera au chapitre VII, des applications de cet algorithme.

II - ALGORITHME 7A. CHEMINS DE TOUT POINT x A TOUT POINT y DU GRAPHE(E, Γ^+), UTILISANT UNE PILE.

On utilise la suite des sorties de la pile annexe du graphe (E, Γ^+). Dans la suite :

$\xi(x)$ désignera l'entrée vraie de x dans la pile

$\sum(x)$ celui de la sortie vraie de x (D.18)

$\sigma(x)$ celui de la première sortie de x.

II - 1. ALGORITHME

1) *Initialisation*

$(\forall x \in E)\ (\forall y \in E)\ (K\ [x,y,o] = \wedge)$.

2) e étant le sommet et r désignant le nombre des sorties de la pile, (pour $i=1,2,\ldots,r$)

a) *si* $\sigma_i < \sum(z)$:

$K(e,z,i) = K\ (e,z,i-1) \cup (e,z)$

b) *si* σ_i *est une sortie de* $z \geq \sum(z)$ *ou (si* $\sigma_i = \sum(z)$ *et z non fermeture de circuit : (cf D.19)*

$(\forall y \in E,\ y \neq z)\ (K\ [e,y,i] = K\ [e,y,i-1] \cup [(e,z).\ K[z,y,i-1]])$

c) *si* $\sigma_i = \sum(z)$ *et z fermeture de circuit* :

$(\forall x \in E$ *et vérifiant* $\xi(z) < \sum(x) < \sum(z))\ (\forall y \in E)$

$(K\ [x,y,i] = K\ [x,y,i-1] \cup [K\ [x,z,i-1]\ .\ K\ [z,y,i-1]])$

Pour tout couple x, y non défini ci-dessus

$K\ [x,y,i] = K\ [x,y,i-1]$

II - 2. JUSTIFICATION :

Il s'agit de l'algorithme décrit dans PAIR [45] utilisant une pile annexe au lieu de la pile attachée. Nous l'avons donc réécrit, en lui faisant subir la transformation qui nous permet de passer de la pile attachée à la pile annexe : tout chemin (suite d'arcs) constituant un état de la pile attachée est représenté par la suite des points par lesquels il passe ; la sortie d'un arc de la pile attachée est représentée par la sortie d'un sommet de la pile annexe. Il est possible d'écrire une démonstration complète de la méthode (ou d'une variante) sous cette forme.

Nous nous contenterons de signaler les phases essentielles de cette démonstration. Pour plus ample détail, on pourra consulter notre référence.

Lemme 1 Si i est inférieur à la premiere sortie de y,

$\forall x,\ K(x,y,i) = \wedge$

Lemme 2 Si $x \Gamma y$, une sortie de y précède la sortie vraie de x et x est le sommet lors de cette sortie.

Lemme 3 Soit i une sortie de y, e le sommet de la pile lors de cette sortie. Si la sortie vraie de y, $\sum(y)$ est postérieure à i et si y a une sortie inférieure ou égale à i, $\sum(y) \geq \sum(e)$.

Définition Un chemin $y_0, y_1 \ldots y_n$ est présent en i si

$\exists \ell\ (1 \leq \ell \leq n)$ tel que

- $y_0 \ldots y_\ell \in K\ [y_0, y_\ell, i]$
- $\ell = n$ ou (y_ℓ est fermeture de circuit et $\sum(y_\ell) > i$)

Théorème : Un chemin (ou circuit) élémentaire $y_0 \ldots y_n$ est présent en tout $i \geq \sum(y_0) - 1$.

Conséquence : Si p est le numéro de la dernière sortie, tout chemin ou circuit élémentaire $y_0 \ldots y_n$ appartient à $K\ [y_0, y_n, p]$. En effet il est présent en p et aucune sortie $\sum(y_\ell)$ n'est supérieure à p.

II - 3 VARIANTES DE LA METHODE :

II-3a) Les boucles (D. 7) peuvent être négligées sans modifier les chemins élémentaires ni les circuits élémentaires.

II-3b) Le lemme 1 nous permet de n'effectuer les opérations 2.b et 2.c que pour les points y ayant une sortie inférieure à σ_i.

II-3c) Si z est une feuille (D. 6) $K\ [z,y,i] = \wedge$ pour tout y et pour tout i. Il est donc inutile d'effectuer l'opération 2b lors de la sortie de tels points z.

II-3d) On peut remplacer dans 2.c l'ensemble des x tels que $\varepsilon(z) < \sum(x) < \sum(z)$ par des ensembles plus grands, chaque suite K(x,y,i) contiendra toujours les chemins qu'elle contenait et éventuellement d'autres.
Cet ensemble $\{x,\ \varepsilon(z) < \sum(x) < \sum(z)\}$ pouvant être délicat à délimiter on pourra utiliser l'ensemble $\{x,\ \sum(x) < \sum(z)\}$.
On pourrait aussi effectuer l'opération 2.c pour tout l'ensemble des x appartenant à E.

II-3e) On peut aussi diminuer encore cet ensemble et effectuer l'opération 2.c pour l'ensemble des x tels que

$$\varepsilon(z) < \sum(x) < \sum(z) \text{ et } K\ [x,z,i-1] \neq \Lambda .$$

En effet la concaténation des chemins de K(x,y,i-1) et de $K\ [z,y,i-1]$ n'apporte aucun chemin à la suite $K\ [x,y,i]$ si $K\ [x,y,i-1] = \Lambda$

II-3f) Si on se limite en outre, aux x tels que

$\varepsilon(z) < \sum(x) < \sum(z)$ et $K\ [x,z,i-n] \neq \Lambda$ et $K\ [z,y,i-1] \neq \Lambda$,
Les suites obtenues contiendront toujours les chemins et circuits élémentaires ; il est alors possible d'imposer aux points x d'être équivalent à z.
Nous verrons au chapitre 7, l'utilisation de certaines de ces variantes.

II - 4 ORGANIGRAMME

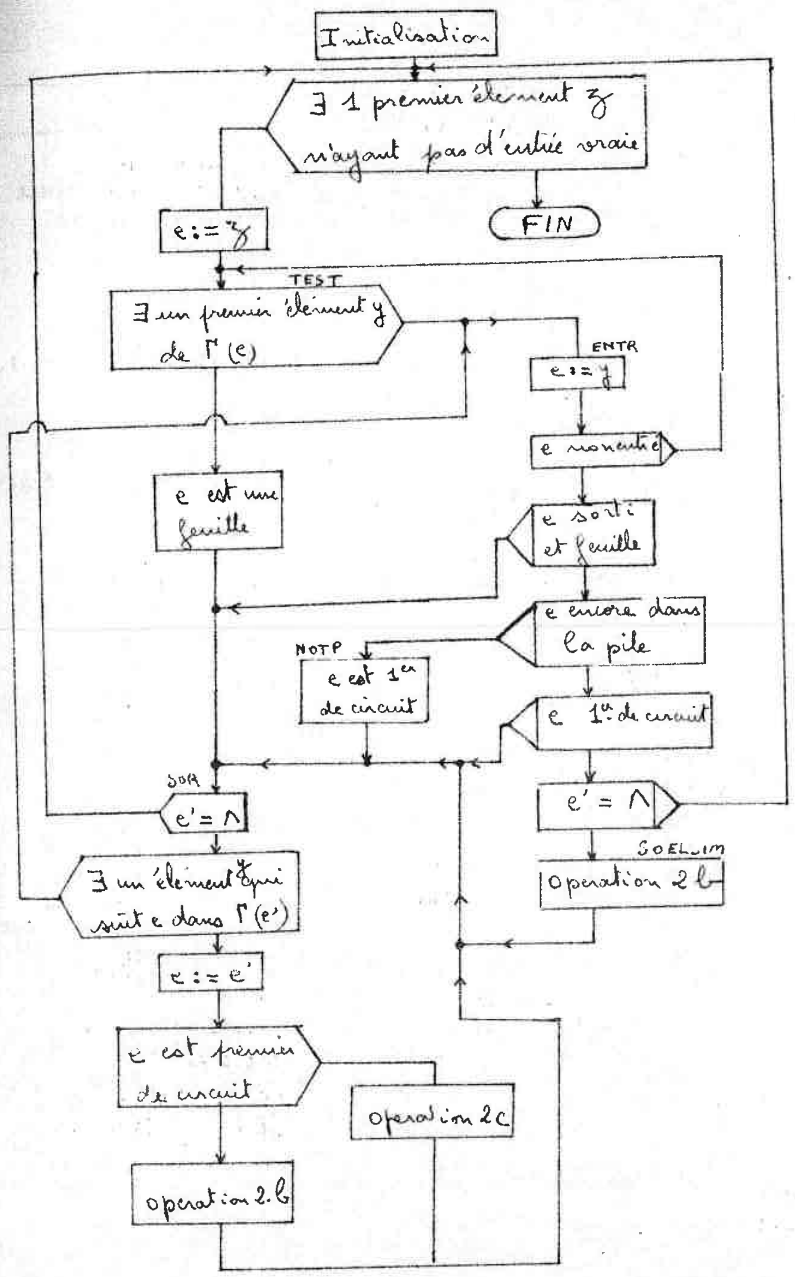
Voir page 26, l'organigramme théorique de l'algorithme 7A.

II - 5 PROGRAMMATION

On ne décrira ici que la programmation de la gestion de la pile et non les opérations 2b et 2c qui varient avec les problèmes.

Le graphe est supposé représenté par sa matrice associée $A[i,j]$

La pile est un tableau à une dimension $P\ [1:N+1]$; la position de son sommet est repérée par son indice k. (cf I.VI). Nous utiliserons encore des indicateurs (tableau $T\ [1:N]$) permettant de connaître la situation de chaque sommet :



T [I] = 1 , I non encore entré
2 , déjà entré non sorti
3 , I a eu une sortie et est fermeture de circuit
4 , I a eu une sortie mais n'est ni premier de circuit ni une feuille
5 , I a eu une sortie et est une feuille

I sera réservé pour désigner le sommet.

Initialisation : $\forall i$, T [i] =1, K = 1 et on effectue l'opération 1.

∃1 premier élément z n'ayant pas d'entrée vraie ?

On cherche le premier élément z tel que T [I] = 1. S'il en existe ; sinon tous les arcs ont été étudiés et le travail est terminé.

∃1 premier élément y de (e) ? :

Dans la Iième ligne de A, on cherche le premier J tel que A [I,J] ≠ Λ. J entrera alors dans la pile.

Tests sur la situation de e :

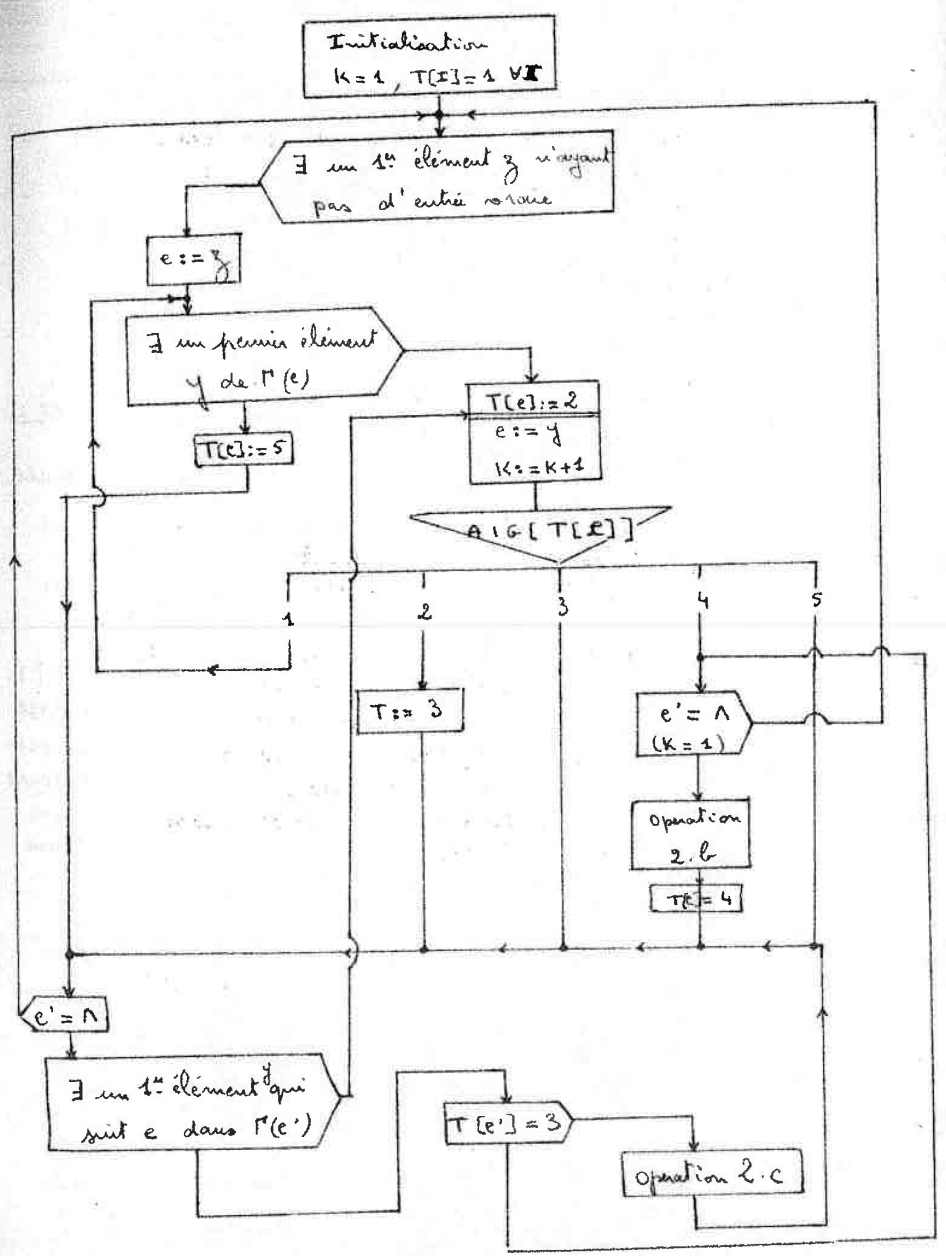
On utilisera en fait un aiguillage AIG [T [I]] avec
AIG := TEST, NOTP, SOR, SOELSIM, SOR ;

e' = Λ peut être effectué en cherchant si K = 1.

Ceci nous permet de donner une autre forme de cet organigramme qui sera utilisée dans nos programmes. (voir organigramme page 28) - Organigramme pratique pour l'algorithme 7A.

III - ALGORITHME 7B. - CHEMINS DE TOUT POINT x D'UN GRAPHE QUELCONQUE A UN POINT Y FIXE :

Pour la recherche des chemins élémentaires d'extrémités y l'opération 2b ne fait intervenir que des chemins élémentaires d'extrémités y et des arcs. Par contre l'opération 2.c fait alors intervenir des chemins d'extrémités y (K [z,y,i-1]) mais aussi des chemins d'extrémité x fermeture de circuit. Il est donc possible de n'efectuer les opération 2b et 2c que pour les points y et les fermetures de circuit. D'après le lemme 1, on peut affirmer qu'il



est inutile de les effectuer pour les fermetures de circuit avant leur première sortie. D'où l'algorithme suivant :

1) Initialisation
   $(\forall x \in E) \quad (K\ [x,y,o] = \wedge\ )$

2) a) si $\sigma_i < \sum(z)$:

$$K\ [e,z,i] = K\ [e,z,i-1] \cup (e,z)$$

b) si σ_i est une sortie de $z > \sum(z)$ ou si ($\sigma_i = \sum(z)$ et z non fermeture de circuit) :
pour $v = y$ et pour $v = v_1, v_2, \ldots, v_k$ fermetures de circuit tels que $\sigma(v_j) < \sum(z)$

$$K\ [e,v,i] = K\ [e,v,i-1] \cup \left[(e,z).\ K\left[z,v,i-1\right]\right]).$$

c) si $\sigma_i = \sum(z)$ et z fermeture de circuit :
$(\forall x \in E$ et vérifiant $\xi(z) < \sum(x) < \sum(z))$ (pour les mêmes v)

$$(K\ [x,v,i] = K\ [x,v,i-1] \cup K\ [x,z,i-1]\ .K\ [z,v,i-1\ ]\ ).$$

Cet algorithme a un intérêt certain si le graphe envisagé n'a pas trop de circuits particulièrement si on veut chercher les chemins entre tout point x du graphe et plusieurs points extrémités, ce qui est impossible à réaliser avec les autres algorithmes autrement qu'en recommençant pour chaque extrémité différente.

Nous écrirons ici deux procédures pour l'algorithme 7A.

```
PROCEDURE ALG7PILE (A,N) ;
   COMMENTAIRE Cette procédure assure la gestion de la pile
       La procédure IGAMMAJ est booléenne et prend la valeur
       vrai quand il existe un arc de I à J. La procédure OPE2B
       réalise l'opération 2B et OPE2C l'opération 2c.
DEBUT  ENTIER I,J,K,R ; ENTIER TABLEAU T [1:N] , P [1:N+1] ;
       AIGUILLAGE AIG:= TEST, NOTP, SOR, SOELSIM, SOR ;
       POUR i:=1 PAS 1 JUSQUA N FAIRE  T [I] :=1 ;
       K:=1 ;
INIT : POUR I:=1 PAS 1 JUSQUA N FAIRE
       SI T [I] =1 ALORS ALLERA INIP ; ALLERA EXIT ;
INIP : P [I] := I ;
TEST : POUR J:=1 PAS 1 JUSQUA N FAIRE
       SI IGAMMAJ ALORS DEBUT T [I] :=2 ; ALLERA ENTR FIN ;
       T [I] :=5 ; ALLERA SOR ;
NOTP : T [I] :=3 ; ALLERA SOR ;
ENTR : K:=K+1 ; P [K] :=5 ; I:=J ; ALLERA AIG [T [I] ] ;
SOR  : SI K=1 ALORS ALLERA INIT ; K:=K-1 ;
SUIT : R:= I+1 ; I:= P [K] ;
       POUR J:=R PAS 1 JUSQUA N FAIRE
       SI IGAMMAJ ALORS ALLERA ENTR ;
       SI T [I] =3 ALORS ALLERA SORIER ;
SOELSIM : SI K=1 ALORS ALLERA INIT ; J:= P [K-1] ;
       OPE2B ; T [I] :=4 ; ALLERA SOR ;
SORIER : POUR R:=1 PAS 1 JUSQUA N FAIRE
       SI T [R] ⩾ 3 ALORS OP2C ; ALLERA SOELSIM ;
EXIT : FIN ;
```

Cette version réalise l'opération 2c pour l'ensemble des x tels que $\sigma(x) < \sum(z)$ qui nous semble le plus facile à réaliser.
La version suivante (qui est celle décrite dans ce chapitre) semble plus délicate à mettre en oeuvre.

```
PROCEDURE PILE7MOTSORTIE (A,N) ; ENTIER TABLEAU A ; ENTIER N ;
COMMENTAIRE  Cette procédure assure la gestion de la pile.
    La procédure IGAMMAJ est booléenne et prend la valeur vrai
    quand il existe un arc de I à J.
    La procédure OPE2B réalise l'opération 2.b, OPE2C réalise
    l'opération 2.c.
DEBUT ENTIER I,J,K,R,E,V ; ENTIER TABLEAU S,T [1:N] , P [1:N+2] ;
      AIGUILLAGE AIG:=TEST, NOTP, SOR, SOELSIM, SOR ;
      POUR I:=1 PAS 1 JUSQUA N FAIRE T [I] :=1 ;
      K:=1 ; E:=N+2 ;
INIT: POUR I:=1 PAS 1 JUSQUA N FAIRE
      SI T [I] =1 ALORS ALLERA INIP ; ALLERA EXIT ;
INIP: P [J] :=I ;
TEST: POUR J:=1 PAS 1 JUSQUA N FAIRE
      SI IGAMMAJ ALORS DEBUT T [I] :=2 ; S [I] :=E ; ALLERA ENTR
                        FIN ; T [I] :=5 ; ALLERA SOR ;
NOTP: T [I] :=3 ; ALLERA SOR ;
ENTR: K:=K+1 ; P [K] :=J ; I:=J ; ALLERA AIG [T [I]] ;
SOR : SI K=1 ALORS ALLERA INIT ; K:=K-1 ;
SUIT: R:=I+1 ; I:= P [K] ; POUR J:=R PAS 1 JUSQUA N FAIRE
      SI IGAMMAJ ALORS ALLERA ENTR ;
      SI T [I] =3 ALORS ALLERA SOR1ER ;
SOVRAI : P [E] :=I ; E:=E-1 ;
SOELSIM : SI K=1 ALORS ALLERA INIT ; J:= P [K-1] ;
          OPE2B ; T [I] :=4 ; ALLERA SOR ;
SOR1ER : POUR J:= S [I] PAS 1 JUSQUA E-1 FAIRE
         DEBUT V:=P [ J] ; OPE2C ; FIN ;
         ALLERA SOVRAI ;
EXIT : FIN ;
```


La partie haute de la pile est réservée pour y indiquer la suite des sorties vraies ; nous avons vu que dans un état de la pile ne peut exister qu'un seul point ayant déjà eu une sortie, il suffit donc de n+1 éléments pour contenir la pile et sa partie haute.

L'opération NOTP permet d'indiquer que I est un premier de circuit ; Pour connaître l'ensemble des x tels que $\xi(z) < \sum(x) < \sum(z)$ *il suffit de noter à chaque entrée vraie (dans TEST) le niveau atteint dans la suite des sorties vraies : S [I] :=E.*

CHAPITRE IV

ALGORITHME 8

1 - PRINCIPE DE LA METHODE

Etant donné un graphe (E, Γ), $E = \{1,2,\ldots,n\}$, soit (S_k, Γ) un sous-graphe de (E, Γ), $S_k = \{1,2,\ldots,k\}$.
Cherchons à construire les chemins élémentaires de S_k à partir de ceux de S_{k-1}.

a) Soit une chemin élémentaire de S_k. Plusieurs cas peuvent se présenter :

1°) Il ne contient pas k : c'est alors un chemin élémentaire de S_{k-1} ;

2°) Il a k pour extrémité : puisqu'il est élémentaire, il est obtenu par concaténation d'un chemin élémentaire de S_{k-1} d'extrémité j et de l'arc (jk) ;

3°) Il a k pour origine : puisqu'il est élémentaire, il est obtenu par concaténation d'un arc (kl) et d'un chemin élémentaire de S_{k-1}, d'origine l ;

4°) Il contient k qui n'est ni son origine, ni son extrémité : puisqu'il est élémentaire il ne contient k qu'une seule fois ; il est donc obtenu par concaténation d'un chemin d'extrémité k (décrit au 2°) et d'un chemin d'origine k (décrit au 3°).

b) Envisageons un circuit élémentaire $(x_0, x_1, \ldots, x_n)$. Les mêmes cas se présentent :

1°) Il ne contient pas k : c'est alors un circuit élémentaire de S_{k-1} ; (il ne peut être en particulier une boucle).

2°) $x_0 = k$: $x_1, \ldots x_0$ est donc un chemin élémentaire de S_k d'extrémité k il a été obtenu par a) 2°). Le circuit $x_0 \ldots x_0$ est obtenu par concaténation d'un chemin élémentaire de S_k (non plus S_{k-1}) et d'un arc d'origine k. (a 3°)

4°) Il contient k, qui n'est pas x_0 : puisqu'il est élémentaire il ne le contient qu'une seule fois, il est donc obtenu par concaténation d'un chemin élémentaire $x_0..k$ de S_k (décrit en a)2°) et d'un chemin élémentaire $k...x$ de S_k (décrit en a)3°).

II - ALGORITHME

Il décrit la façon d'obtenir des suites de chemins contenant les chemins élémentaires ainsi distingués.

1) $(\forall x \in E)\ (\forall y \in E)\ \ K[x,y] := I[x,y]$

2) $(k = 2,\ldots,n)$

a) $(\ell = 1,\ldots k-1)\ (\forall j \in S_{k-1})$
$K[k,\ell] = \bigcup_j (kj) \odot K[j,\ell]$

b) $(\ell = 1,\ldots,k-1)\ (\forall j \in S_{k-1})$
$K[\ell,k] = \bigcup_j K[\ell,j] \odot (j,k)$

c) $(\forall i \in S_{k-1})\ (\forall j \in S_{k-1}$ et $j \neq i)$
$K[i,j] = K[i,j] \cup [K[i,k] \,.\, K[k,j]]$

Les suites de chemins ainsi obtenues sont sans répétitions : si, à l'itération k on ajoute un chemin il contient k, il ne pouvait donc pas figurer dans cette suite à l'itération précédente ; de plus il ne peut y avoir de répétitions entre les chemins obtenus à l'itération k, les types de chemins décrits étants incompatibles. Notons que seule l'opération c apporte dans ces suites des chemins non élémentaires.

III - VARIANTES DE L'ALGORITHME

III-1 Suites contenant les chemins et circuits élémentaires du graphe.

Pour obtenir les circuits élémentaires il faut réaliser les opérations permettant d'obtenir les circuits décrits en b)2 à partir des chemins de a)2, ainsi que les circuits décrits en b)3. Il suffit de réaliser 2)b pour $\ell = 1,2,\ldots k$ et 2c) pour $(\forall j \in S_{k-1})$.

III-2 TESTS :

L'algorithme ainsi rédigé nécessite $n(n-1)^2$ opérations réunion et produit (voir § III-3). Il est cependant possible d'éviter des opérations qui réalisent des concaténations entre éléments dont l'un peut être vide. On pourra adopter la rédaction suivante.

1) $(\forall x \in E)\ (\forall y \in E)\ (K[x,y] = I[x,y])$

2) $(k = 2,3,\ldots,n)$

a) $(\forall j \in S_{k-1}$ et tel que $k \Gamma j)$ (pour $\ell = 1,..,j-1,j+1,..k-1$)
$(K[k,\ell]_j = K[k,\ell]_{j-1} \cup \{(k,j) \odot K[j,\ell]\})$

b) $(\forall j \in S_{k-1}$ et tel que $j \Gamma k)$ (pour $\ell = 1,..,j-1,j+1,..,k-1$)
$(K[\ell,k]_j = K[\ell,k]_{j-1} \cup \{K[\ell,j] \odot (j,k)\})$

c) $(\forall i \in S_{k-1}$ et tel que $K[i,k] \neq \wedge)\ (\forall j \in S_{k-1},\ j \neq i)$
$(K[i,j] = K[i,j] \cup \{K[i,k] \odot K[k,j]\})$.

Chaque test effectué en a,b ou c permet d'éviter k opérations à l'itération d'ordre k. Il est remarquable que les tests a et b ne portent que sur les arcs et non sur les chemins du graphe, en ce sens qu'ils sont plus faciles à faire (on peut par exemple utiliser une liste des arcs) et qu'ils permettent d'éviter des opérations beaucoup plus souvent que ceux portant sur les chemins ($(i,j) = \wedge$ plus souvent que $K[i,j]$).

III-3 Nombre d'opérations

a) *Faisons le compte pour la variante cherchant aussi les circuits :*

Il faut pour a) $(k-1)(k-2)$ opérations
pour b) $(k-1)(k-2)$
c) $(k-1)^2$
b') $(k-1)$

Soit au total

$$\sum_{k=2}^{n} (k-1)(k-2) + (k-1)(k-2) + (k-1) + (k-1)^2 = n(n-1)^2$$

b) *Pour la variante de III-2. $(n-1)^2$ représente un maximum dont on sera d'autant plus loin que le graphe sera moins plein.*

Si on effectue les tests pour les opérations a,b,c on peut ne pas le faire pour c, qui est la moins favorable , il faut compter à chaque itération sur k $3(k-1)$ tests soit au total

$$\sum_{k=2}^{n} 3(k-1) = \frac{3}{2}n^2 - \frac{3}{2}n - 9 \text{ tests soit environ } \frac{3}{2}n^2.$$

Notons que pour l'algorithme 5 il en faut n^2.

On trouvera au chapitre VII, une comparaison expérimentale de ces nombres d'opérations entre les diverses méthodes. Notons aussi, que pour cet algorithme, on ne peut plus parler de couples qui opèrent", on comptera donc les opérations explicitement.

III-4 Importance de l'ordre des sommets :

Chaque test peut permettre d'éviter (k-1) opérations additions et multiplications . Il est donc plus intéressant d'éviter des opérations vers les dernières itérations. L'ordre des points est donc important, mais il est facile de trouver un ordre favorable alors que pour l'algorithme 5 on ne connait que l'ordre fourni par la pile : on peut prendre l'ordre des sommets par degré décroissant.

On ne peut pas affirmer que ce soit l'ordre optimal, car dans l'opération b les test portent sur les arcs entrants du sommet k, dans c ils portent sur les chemins entrants de k et dans a sur les arcs sortants du sommet k, et il n'y a pas un nombre égal d'opérations évitées dans chaque type.

Néanmoins on peut dire, surtout si on n'effectue pas les tests c), que l'ordre des sommets par degré décroissant est une bonne approximation de l'ordre optimal.

En utilisant cet ordre il est sans doute inutile d'effectuer des tests pour les premières itérations. On ne commence donc qu'au delà d'un ordre k' déterminé empiriquement.

IV - PROGRAMMATION

Les programmes réalisés acceptent le graphe sous forme de sa matrice associée. (tableau A).

Ce tableau peut aussi servir de support à la matrice des distances mais pas aux calculs intermédiaires :

- les éléments A(i,k) et A(k,j) n'ont jamais été modifiés avant l'itération k. Au début de cette itération la kième ligne et la kième colonne sont donc celles de la matrice associée.
- les arcs d'extrémités k ne sont utiles qu'à l'itération k
- les éléments A(i,j), i k, j k représentent des distances

Il suffit donc d'utiliser des mémoires de travail représentant la kième ligne et la kième colonne pendant l'itération k. rôle des tableauw B et C . On trouvera page 37 le programme Algol CAE 510, correspondant au problème de la plus courte distance. Il s'agit de la variante du § III-2.


```
'PROCEDURE'PCDDANTZIG(N,A); 'ENTIER'N; 'ENTIER''TABLEAU'A;
'DEBUT''ENTIER''TABLEAU'B,C.(1:N).; 'ENTIER'I,J,K,Q,L;
'POUR'I=1'PAS'1'JUSQUA'N'FAIRE''DEBUT'
'POUR'J=1'PAS'1'JUSQUA'N'FAIRE'
'SI'A.(I,J).'EGAL'0'ALORS'A.(I,J).=10000 ;
A.(I,I).=0 'FIN' ; Q=0;
'POUR'K=1'PAS'1'JUSQUA'N'FAIRE''DEBUT'
'POUR'I=1'PAS'1'JUSQUA'K-1'FAIRE''DEBUT'
B.(I).=C.(I).=10000  'FIN';
'POUR'J=1'PAS'1'JUSQUA'K-1'FAIRE'
'SI'A.(K,J).'INF'10000 'ALORS'
'POUR'L=1'PAS'1'JUSQUA'K-1'FAIRE''DEBUT'
'SI'A.(K,J).+A.(J,L).'INF'B.(L).'ALORS'
B.(L).=A.(K,J).+A.(J,L).; Q=Q+1 'FIN';
'POUR'J=1'PAS'1'JUSQUA'K-1'FAIRE'
'SI'A.(J,K).'INF'10000 'ALORS'
'POUR'L=1'PAS'1'JUSQUA'K-1'FAIRE''DEBUT'
Q=Q+1; 'SI'A.(J,K).+A.(L,J).'INF'C.(L).'ALORS'
C.(L).=A.(J,K).+A.(L,J). 'FIN';
'POUR'I=1'PAS'1'JUSQUA'K-1'FAIRE'
'SI'A.(I,K).'INF'10000 'ALORS'
'POUR'J=1'PAS'1'JUSQUA'K-1'FAIRE''DEBUT'
'SI'A.(I,K).+A.(K,J).'INF'A.(I,J).'ALORS'A.(I,J).=A.(I,K).+A.(K,J).;
Q=Q+1 'FIN'; 'POUR'L=1'PAS'1'JUSQUA'K-1'FAIRE''DEBUT'
A.(K,L).=B.(L).; A.(L,K).=C.(L). 'FIN' 'FIN';
EXE(6,Q); IMPR;
'SI'CLE(6)'ALORS''POUR'I=1'PAS'1'JUSQUA'N'FAIRE'
'DEBUT''POUR'J=1'PAS'1'JUSQUA'N'FAIRE'EXE(4,A.(I,J).);
IMPR 'FIN' 'FIN' PCDDANTZIG;
```

CHAPITRE 5

COMPARAISON DES DIVERS ALGORITHMES

Dans ce chapitre nous essaierons de comparer les différents algorithmes sous divers points de vue : nombre théorique d'opérations à effectuer, temps de calcul, place occupée en mémoire... Tous les essais mentionnés ont été effectués sur le problème de la recherche de la plus courte distance, les conclusions que l'on pourra en tirer restant valables pour les autres problèmes, puisque seule diffère la nature des opérations à effectuer et non leur nombre.

I - ALGORITHMES CONSTRUISANT DES SUITES DE CHEMIN ENTRE LES POINTS x DU GRAPHE ET UN POINT y EXTREMITE :

Rappelons que l'on peut écrire des algorithmes analogues pour un point x fixé et tous les points du graphe.
On peut utiliser les algorithmes 2 (variantes de FORD ou de BELLMANN KALABA) et 7B pour les graphes quelconques et l'algorithme 6 si le graphe est sans circuit.

I.1 Cas des graphes sans circuit : Algorithme 2 et 6.

Encombrement :

L'écriture des programmes pour l'algorithme 2 est certainement plus facile que pour l'algorithme 6 et le programme obtenu plus court. De plus l'algorithme 6 nécessite 2 tableaux de n éléments en plus de la matrice associée et des emplacements prévus pour les résultats.

Nombre d'opérations :

Pour l'algorithme 2, p étant le maximum des nombres d'arcs des différents chemins et circuits élémentaires du graphe, les algorithmes déduits par homomorphisme de l'algorithme 2 demandent pn^2 multiplications et pn^2 additions [au sens du chapitre 2] , c'est-à-dire un nombre de l'ordre de n^3 (p est en général assez voisin de n). Les tests $J[z,y,i-1] \neq \Lambda$ permettent d'en éviter certaines. On a vu que l'algorithme 6 nécessite exactement m (nombre d'arcs du graphe) opérations ; c'est-à-dire un nombre inférieur à n^2. Le nombre d'opérations nécessaires à la gestion de la pile étant lui aussi proportionnel à m, cet algorithme est incontestablement le plus intéressant pour les graphes sans circuit D'autre part si on veut faire le même calcul pour plusieurs points extrémités on peut

faire les opérations pour tous ces points à chaque sortie de la pile sans recommencer une gestion complète, ni une exploitation du graphe , ce qui est un avantage supplémentaire pour l'algorithme 6.

Essais

La courbe t = f(m) située page 39' représente les variations du temps de calcul par les deux méthodes, lorsqu'on augmente le nombre d'arc d'un graphe.

Les essais ont été effectués sur une série de graphes d'ordre 20 : on passe d'un graphe au suivant en ajoutant des arcs.

Cette courbe montre :

- la disproportion entre les temps de calcul par la méthode de FORD (ce serait la même pour la méthode de Bellmann-Kalaba) et celle utilisant un pile [Alg. 6]
- l'importance pour l'algorithme 2 du test $J[z,y,i-1] \neq \Lambda$ qui est très intéressant pour les graphes assez creux. Sans ce test le temps de calcul serait pratiquement indépendant du nombre d'arcs.
- la proportionnalité du temps de calcul au nombre d'arcs pour l'algorithme 6 .

I.2 - Graphes quelconques : Algorithmes 2 et 7 B

Algorithme 2

Seule change par rapport au cas précédent la proportion des tests pour lesquels $J[z,y,i-1] = \Lambda$ qui devient beaucoup plus faible.

Algorithme 7.B :

Notons que si le graphe est sans circuit il effectue exactement les mêmes opérations que l'algorithme 6.

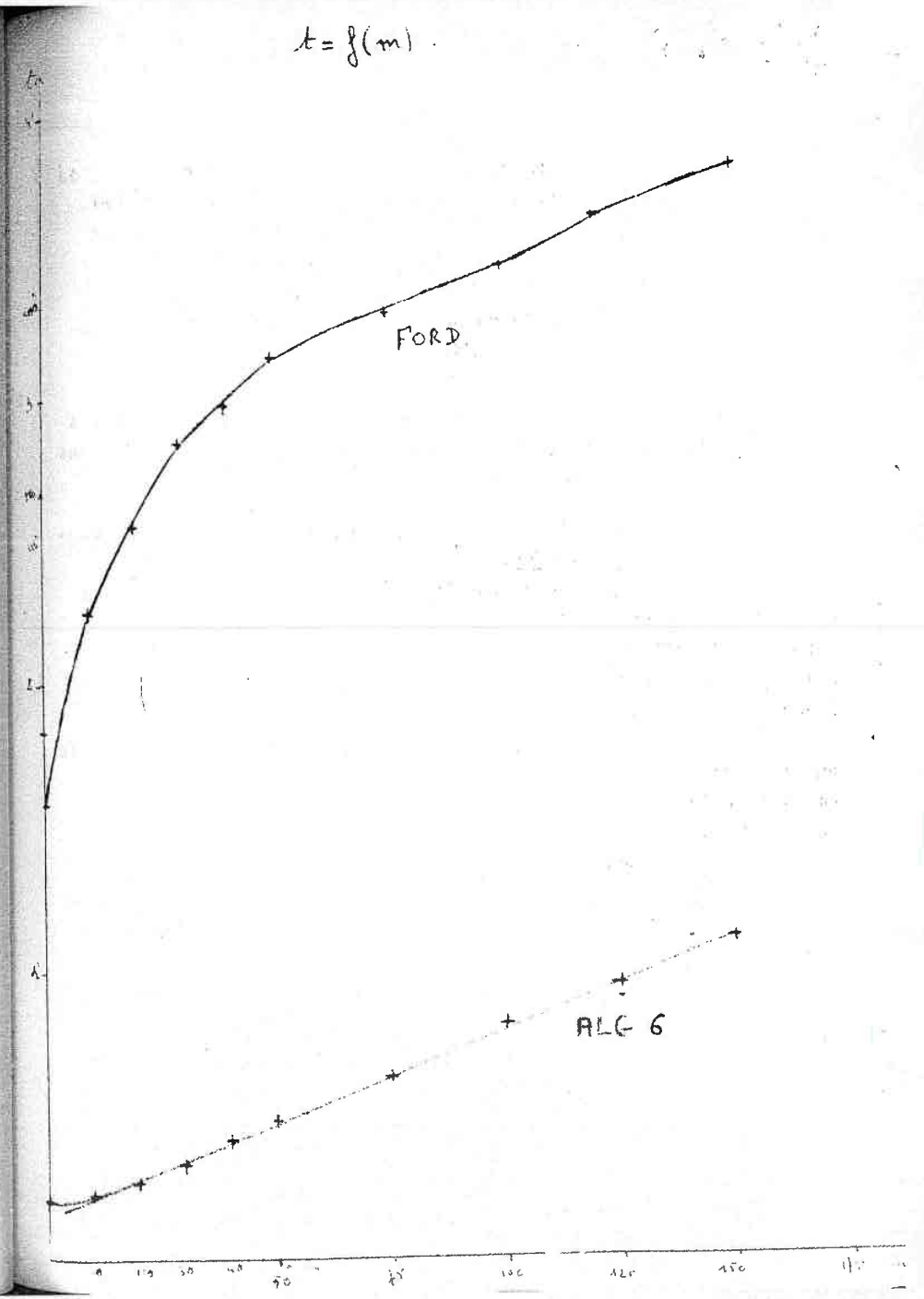
Encombrement :

Il devient plus important que pour l'algorithme 6 ; le programme est plus long et surtout il faut stocker plus de résultats intermédiaires (les chemins d'extrémité les fermetures de circuit).

Nombre d'opérations :

Il est plus délicat à estimer :

q étant le nombre de fermetures de circuit ($q < n$) il reste de l'ordre de qn^2 et, en général, bien inférieur : on ne commence à faire des opérations pour une fermeture de circuit qu'après la première sortie de celle ci.



Essais

De nombreux essais ont été réalisés sur des graphes d'ordre 20 et sur des graphes d'ordre 30 en faisant varier le nombre de circuits.

Pour calculer une ligne de la matrice des distances il faut de 2,5 [quand il y a beaucoup de circuits] à 10 fois [quand il n'y a pas de circuit] plus de temps par l'algorithme 2 que par l'algorithme 7B. Pour les graphes d'ordre 20, de 3 à 14 fois pour les graphes d'ordre 30.

On aura donc dans tous les cas, intérêt à utiliser les algorithmes utilisant une pile 6 quand on sait que les graphes étudiés sont sans circuit., 7B dans le cas général).

II - ALGORITHMES CONSTRUISANT DES SUITES DE CHEMINS ENTRE TOUT COUPLE DE POINTS DU GRAPHE.

On peut utiliser les algorithmes 4,5,7,8, ou les algorithmes 1,2,6,7B en prenant successivement tous les points du graphe comme point extrémité.

Les algorithmes 1,2,7B demanderont alors un nombre d'opérations de l'ordre de n^4; nous les écarterons.

L'algorithme 6 demanderait un nombre d'opérationsde l'ordre de mn et n fois la gestion de la pile : il est plus intéressant de ne la faire qu'une fois et de traiter tous les points extrémités à chaque sortie de la pile (si le graphe est sans circuit).

L'algorithme 4 demandant $n^3 \log_2 p$ opérations, p étant le maximum des nombres d'arcs des chemins élémentaires du graphe, nous l'écarterons également.

Seuls restent les algorithmes 5,7,8 :

Les algorithmes 5 et 8 demandent exactement n^3 opérations chacun si on n'effectue aucun test. Si on en effectue ils peuvent comme l'algorithme 7 résoudre ce problème en mois de n^3 opérations.

II-1 Comparaison théorique des algorithmes 5 et 7.

Nous avons vu (cf chapitre II) que pour ces deux algorithmes, l'ordre dans lequel sont rangés les sommets du graphe est important [pour l'algorithme 7 il existe de nombreuses piles annexes au même graphe] , on pourra donc obtenir des réalisations différentes

de ces algorithmes selon l'ordre choisi,et des nombres d'opérations nécessaires, également différents.

Nous appelerons couple qui opère, un couple de points du graphe xz pour lequel on effectue une opération du type :

$$K(x,y,i) = K(x,y,i-1) \cup [K(x,z,i-1) \odot K(z,y,i-1)] \quad .$$

$\forall y \in E.$

Dans l'algorithme 5 :

$K(x,y,o) = J(x,y)$

pour $i=1,\dots,n$

a) si $K(x,z_i,i-1) \neq \wedge$ et $x \neq z_i$

$K(x,y,i) = K(x,y,i-1) \cup [K(x,z_i,i-1) \odot K(z_i,y,i-1)]$

b) sinon

$K(x,y,i-1) = K(x,y,i-1)$

les couples qui opèrent sont les couples (x,z_i).

Dans la méthode de la pile :

$K(x,y,o) = J(x,y)$

a) $\sigma i \geqslant \sum(z_i)$ $\quad \forall y \in E$, on effectue

$K(e,y,i) = K(e,y,i-1) \cup [(ez) \odot K(z,y,i-1)]$

e étant le sommet

b) $\sigma i = \sum(z_i)$ fermeture de circuit :

$K(x,y,i) = K(x,y,i-1) \cup [K(x,y,i-1) \odot K(z_i,y,i-1)]$

$\forall y \in E$ et $\forall x$ tel que $\quad \mathcal{E}(z_i) < \sum(x) \leqslant \mathcal{E}(z_i)$ et $K(x,z_i,i-1) \neq \emptyset$

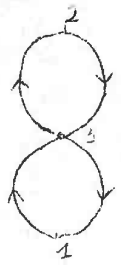
les couples qui opèrent sont les couples (e,z) pour l'opération a) et (x,z_i) pour l'opération b).

Proposition 1 :

Le nombre d'opérations pour l'algorithme 7 n'est pas toujours inférieur au nombre d'opérations nécessaires, quelque soit l'ordre des sommets, pour l'algorithme 5.

Il suffit pour le démontrer de donner un exemple en précisant l'ordre des sommets utilisé dans chaque algorithme :

Exemple :



ordre 1,2,3

On peut représenter le fonctionnement de la pile :

			3				
	1		2	2	2		
3	3	3	3	3	3	3	
1	1	1	1	1	1	1	1

numéros de sortie : 1 2 3 4 5

couples qui opèrent :

Algorithme 6 : 1 : 3,1
2 : 3,2
3 : 1,3.2,3

$\underline{4}$

Algorithme 7

$\underline{5}$

sortie	couple
1	
2	
3	3,2 (a)
4	1,3(a).2,3(b)
5	2,1(b).3,1(b)

Proposition 2 :

Si le graphe est plein, il y a autant de couples qui opèrent pour les deux algorithmes 5 et 7.

Si le graphe est plein, pour tout couple (x,z_i), $K(x,z_i,i-1) \neq$.
Il y a donc $n(n-1)$ couples qui opèrent dans l'algorithme 5. (On peut en effet ne pas tenir compte des couples x,x).
Pour l'algorithme 7 :
Etant donné un couple (x,z), $x \neq z$; $x \Gamma z$.

a) si $\sum(z) < \sum(x)$, à la sortie de l'arc xz, z n'est plus dans la pile (sinon on aurait $\sum(x) < \sum(z)$), la sortie de x correspondant à l'arc xz est supérieure ou égale à $\sum(z)$: le couple (x,z) opère par a).

b) si $\sum(x) < \sum(z)$ l'arc x,z sort avant $\sum(x)$ (Lemme 2, chap. III). z est alors fermeture de circuit et $E(z) < \sum(x) < \sum(z)$: le couple xz opère par l'opération b.

Dans tout couple (x,z), $x \neq z$ opère. Notons cependant qu'il y a davantage d'opérations annexes dans l'algorithme 7 (gestion de la pile).

Proposition 3 :

Si le graphe est sans circuit, le nombre de couples qui opèrent pour l'algorithme 7 est égal à m, nombre d'arcs du graphe. La pile permet de trouver un ordre des sommets pour lequel le nombre de couples qui opèrent pour l'algorithme 5 est aussi égal à m. Pour les autres ordres, ce nombre est supérieur ou égal à m.

Pour la pile la démonstration est immédiate : tous les couples opèrent par l'opération a) et y a m sorties de la pile.

Pour l'algorithme 5 : il y a au moins m couples (x,z_i) tels que $K(x,z_i,i-1) \neq \wedge$ et souvent plus. Montrons qu'il existe un ordre des sommets tel que pour l'algorithme 5 correspondant, exactement m couples opèrent :

Soit un ordre total S tel que $x \Gamma y \Longrightarrow ySx$ (l'ordre des sorties vraies de la pile remplit cette condition, cf chapitre III).
$K(x,z_i,i-1) \neq \wedge \Longrightarrow \exists$ un chemin de x à z_i dont aucun point intérieur n'a d'indice au plus égal à i.
Or si ce chemin a un point intérieur v, z_i Sv, ce qui est impossible : donc v n'existe pas et $x \Gamma z_i$

$$K(x,z_i,i-1) \neq \wedge \quad \Longrightarrow \quad x \Gamma z_i$$

Il y a donc exactement m couples qui opèrent pour l'algorithme 5 dans ce cas.

Nous dirons qu'une classe C peut être atteinte en un point y $\in$ C si $\exists\, x \notin C$ tel que $x \Gamma y$.

Proposition 4 :

Si pour chaque classe d'équivalence, les points de cette classe ne forment qu'un circuit élémentaire, quels que soit les ordres de sommets utilisés, l'algorithme 7 nécessite un nombre de couples qui opèrent au plus égal à celui de l'algorithme 5, et inférieur, dès qu'il existe un classe pouvant être atteinte en deux points.

α) Pour l'algorithme 5

1. $K(x,z,i) \neq \wedge$ si $x \Gamma z$
2. Soit z_i le dernier élément d'une classe :

si $x \sim z_i$ et $x \neq z_i$, il existe un chemin de x à z_i dont l'indice de tout premier point intérieur est inférieur à i (puisque z_i est le dernier de la classe) donc $K(x,z_i,i-1) \neq \wedge$. Si la classe contient p points il y a p-1 points tels que x.

3. Pour tout point x tel que $x \bar{\Gamma} v \sim z_i$ on a aussi

$$K(x,z_i,i-1) \neq \wedge$$

Soit m nombre d'arcs (x,z) $x \neq z$
c le nombre de classes
c' le nombre de classes à plus d'un point.
n le nombre de points.

Il y a m couples pour lesquels $K(x,z,i) \neq \wedge$ par 1, il y en a n-c par 2, les couples communs étant les couples x,z_i tels que $x \sim z$ et $x \Gamma z$, z dernier de classe : pour toute classe comportant plus d'un point, il y a au moins un de ces couples. D'après l'hypothèse, il y en a un et un seul : s'il y en avait deux x,z et x',z il y aurait deux circuits dans la classe z...xz et z...x'z.

Il y a donc au moins m+n-c-c' couples qui opèrent

Il y en a plus s'il existe une classe c pouvant être atteinte en deux points :

$\exists\, u, u' \in C \qquad u \neq u'$ et $x,x' \notin c$ tels que $x \Gamma u$ et $x' \Gamma u'$.
$K(x,z,i-1) \neq \wedge$ et $K(x',z_i,i-1) \neq \wedge$
et un chemin de x à z_i contient u et un chemin de x' à z_i contient u'. Au moins un de ces chemins n'est pas un arc car $u \neq u'$.

β) Pour la pile

Si z est fermeture de circuit, il existe un circuit pour lequel z est le premier point entré (cf chapitre III lemme 3) . z est donc base de classe (cf chapitre 6) puisqu'une classe ne contient qu'un circuit.

Les couples qui opèrent sont les couples :

- $\omega_i\ z_i$, $\sigma i \geqslant \sum(z_i)$.

$\omega_i \Gamma z_i$, $\sum(\omega_i) > \sigma i \geqslant \sum(z_i) \Longrightarrow z_i$ non base de classe ou $\omega_i \not\sim z_i$

Réciproquement

- $\omega_i \,''\, z_i \Longrightarrow \sigma i \geqslant \sum(z_i)$ (cf chapitre III Lemme 3)
- $\omega_i \sim z_i$ et z_i non base de classe $\Rightarrow \sigma i > \sum(z_i)$

Ces couples sont au nombre de m-c'.

- x, z_i : z_i base de classe et $x \sim z_i$ $(x \neq z_i)$

Ils opèrent à l'instant $i = \sigma(z_i)$.

Il y en a n-c.

Le nombre de couples qui opèrent est donc égal à m+n-c-c'. C'est-à-dire inférieur ou égal au nombre obtenu pour l'algorithme 5. On retrouve m quand le graphe est sans circuit : n=c . c'=0.

Notons que dans ce cas le nombre ne dépend pas de la pile choisie.

Cas général :

Les comparaisons sont plus complexes et il est difficile de donner des résultats valables dans tous les cas. On peut cependant étudier des couples qui opèrent :

a) Les couples xz tels que $x \Gamma z$ opèrent dans les deux cas.

- évident pour algorithme 5 car $(x,z,o) \neq \wedge$
- pour la pile

  si $\sigma(xz) \geqslant \Sigma(z)$ le couple xz opère

  si $\sigma(xz) < \Sigma(z)$ alors $x \sim z$, $\xi(z) < \xi(x) < \Sigma(z)$

  et $K(x,z,\Sigma(z)-1)$ contient (x,z).

Donc le couple xz opère aussi

b) Aucun autre couple xz, tel que $x \not\sim z$ n'opère pour la pile alors que pour l'algorithme 5, il en existe nécessairement dans le cas où plusieurs arcs mènent à une classe non réduite à un point. (cf. Lemme 4). On peut cependant réduire leur nombre si on ordonne les points du graphe de telle sorte que :

$$x \in c,\ x' \in c',\ c \Gamma_r c' \Longrightarrow x < x'$$

où c et c' sont deux classes distinctes, Γ_r est la relation du graphe des classes (cf chapitre IX)

En effet dans ce cas, s'il existe un chemin de x à z_i il possède un point intérieur $y \not\sim z_i$, y suit z_i, le chemin $\in K(x,z_i,i-1)$ et $|K(x,z_i,i-1)| =$. Les seuls couples restants sont tels qu'il existe $y \sim z$ et $x \Gamma y$. Mais cet ordre ne peut être obtenu directement. La meilleure méthode (cf chapitre 6) semble être celle utilisant une pile qui détermine les classes : l'ordre de sortie des bases de classe détermine l'ordre cherché sur les classes. C'est d'ailleurs le seul rôle joué par la pile dans le cas des graphes sans circuits. Pour les autres cas les comparaisons théoriques ne permettent guère de conclusion.

II-2 ESSAIS

Courbe 1 (page 48) sans tests.

Les essais ont été effectués sur des graphes sans circuits d'ordre croissant de 10 à 32.
Cette courbe montre bien l'allure en n^3 de cette fonction.

Courbe 2 (page 49)

Ce graphique montre l'évolution du nombre de couples qui opèrent d'une part, du temps de calcul d'autre part en fonction du nombre d'arcs. A chaque couple qui opère correspondent n opérations. Les graphes sont tous sans circuit d'ordre 20 et on passe d'un exemple au suivant par adjonction d'arcs.
Il montre bien l'importance des tests permettant d'éviter des opérations sans tests. Ce maximun qui est n^3 dans le cas général est $\frac{n^2(n-1)}{n}$ dans le cas des graphes sans circuits.

Courbes 3(page 50)

Elles montrent une comparaison entre les différentes méthodes permettant de calculer la matrice des distances. Il s'agit encore de graphes sans circuit d'ordre 20, pour lesquels on fait varier le nombre d'arcs.
On peut y constater l'avantage de l'algorithme 7 (ou 6 dans ce cas) comme on l'a dit au § I de ce chapitre, ainsi que l'amélioration apportée à l'algorithme 7 en utilisant le même support pour la matrice associée et la matrice des distances (cf chapitre 7,§ III.3)

Courbes 4(page 51)

On y compare le nombre d'opérations (nombre de couples qui opèrent multiplié par n pour les algorithmes 5 et 7) pour les 3 algorithmes 5,7,8. Il s'agit d'un graphe d'ordre 20, à 49 arcs sur lequel on crée des circuits en ajoutant à chaque fois un arc d'extrémité non encore fermeture de circuit.
Le nombre d'opérations pour l'algorithme 7 n'est pas strictement croissant, car le nombre de fermeture de circuits n'est pas le seul facteur qui intervient.

Pour chaque classe d'équivalence de ces graphes, les points de la classe n'appartiennent qu'à un circuit. Dans ce cas il semblerait que la méthode de Dantzig (algorithme 8) soit avantageuse, ce n'est pas toujours le cas (courbe 6). On peut cependant remarquer la stabilité du nombre d'opérations pour cette méthode.

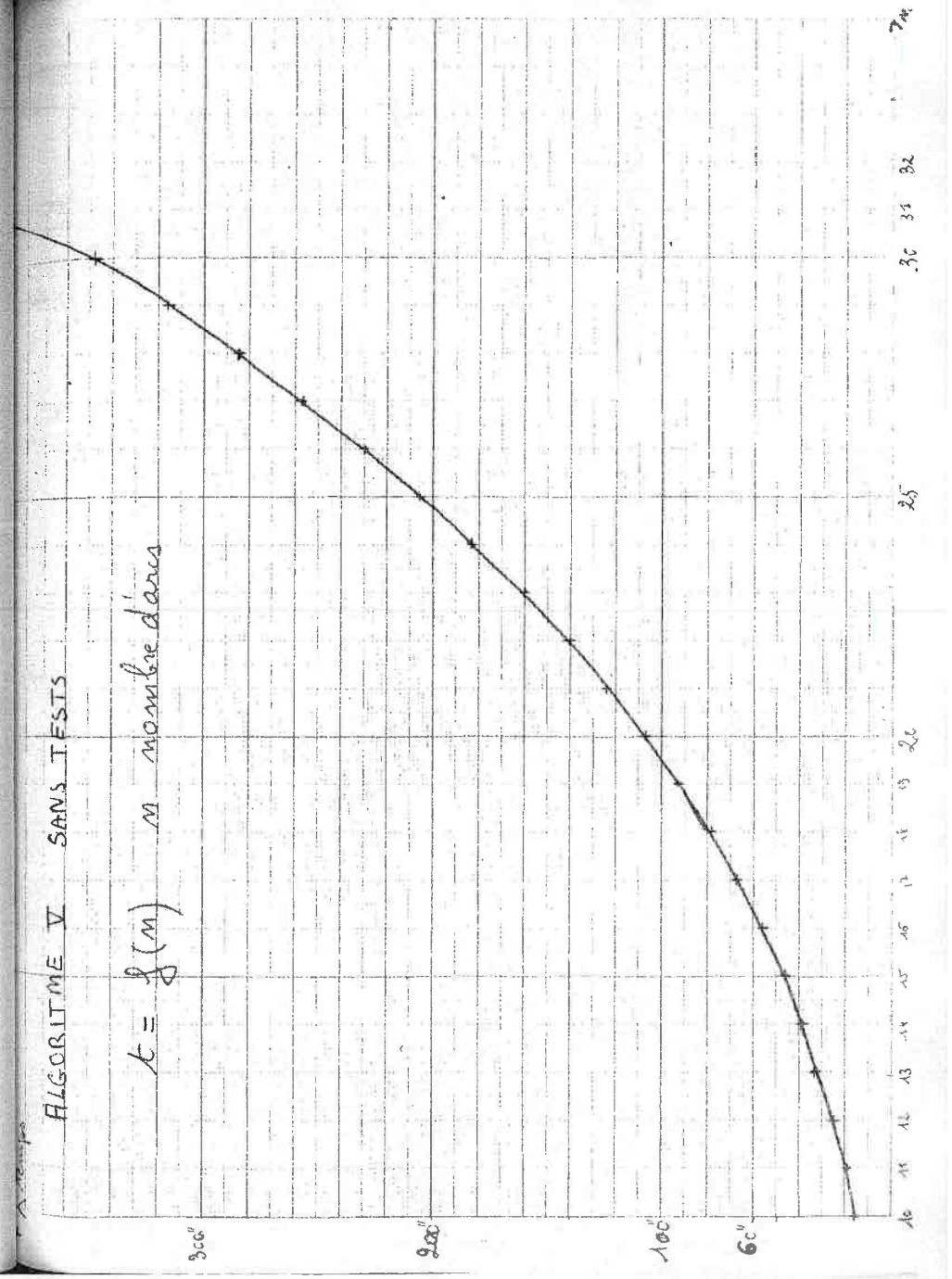
Courbes 5 (page 51-1)

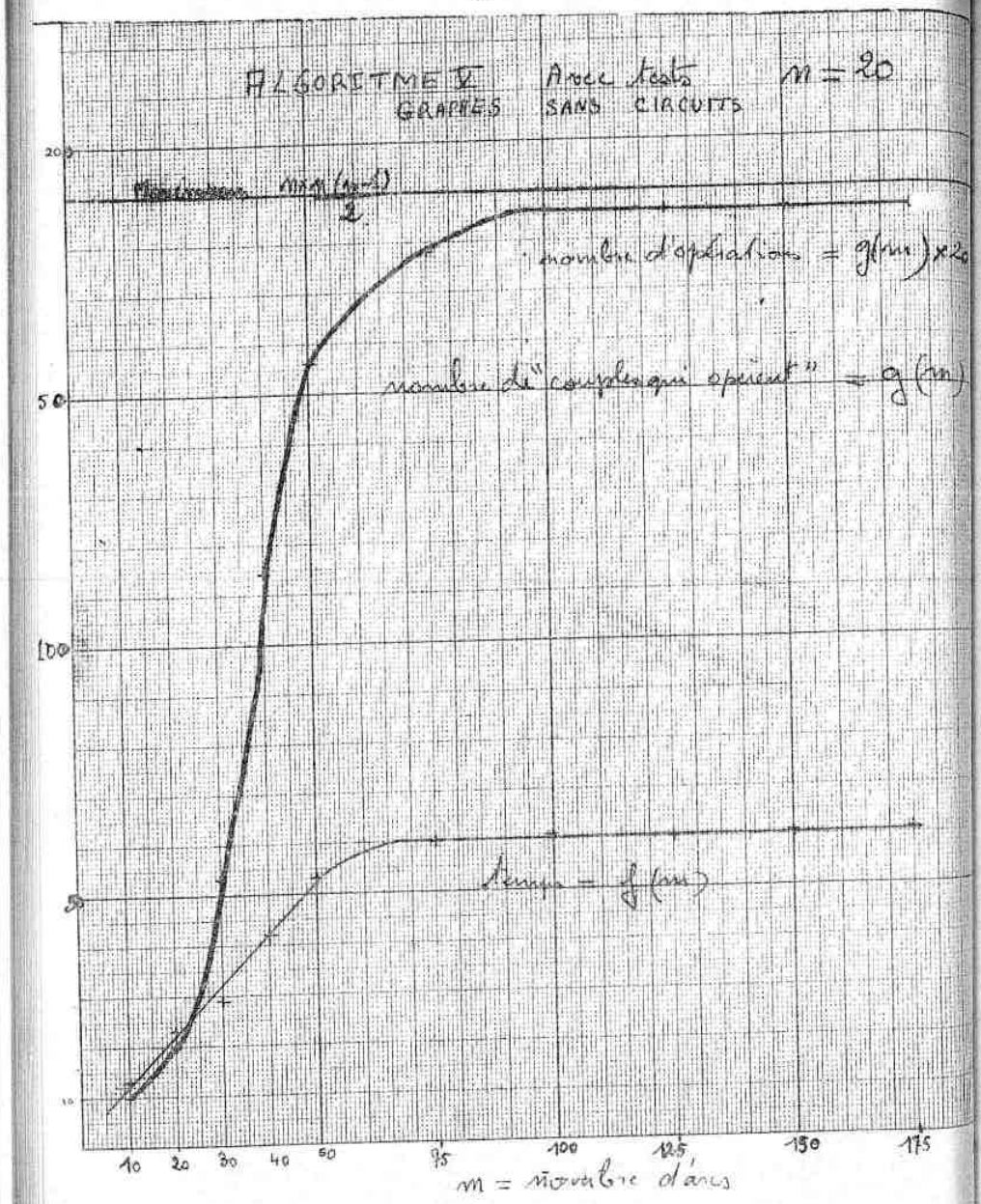
Il s'agit encore d'un graphe d'ordre 20, pour lequel on augmente progressivement le nombre de fermetures de circuit . Cette fois les points de tous les circuits créés appartiennent à la même classe d'équivalence.
On peut constater avec ces deux exemples que l'algorithmes 7 reste le plus avantageux tant que les graphes ne deviennent pas trop complexes : 15 fermetures de circuits dans un graphe de 20 points représente un graphe très complexe.

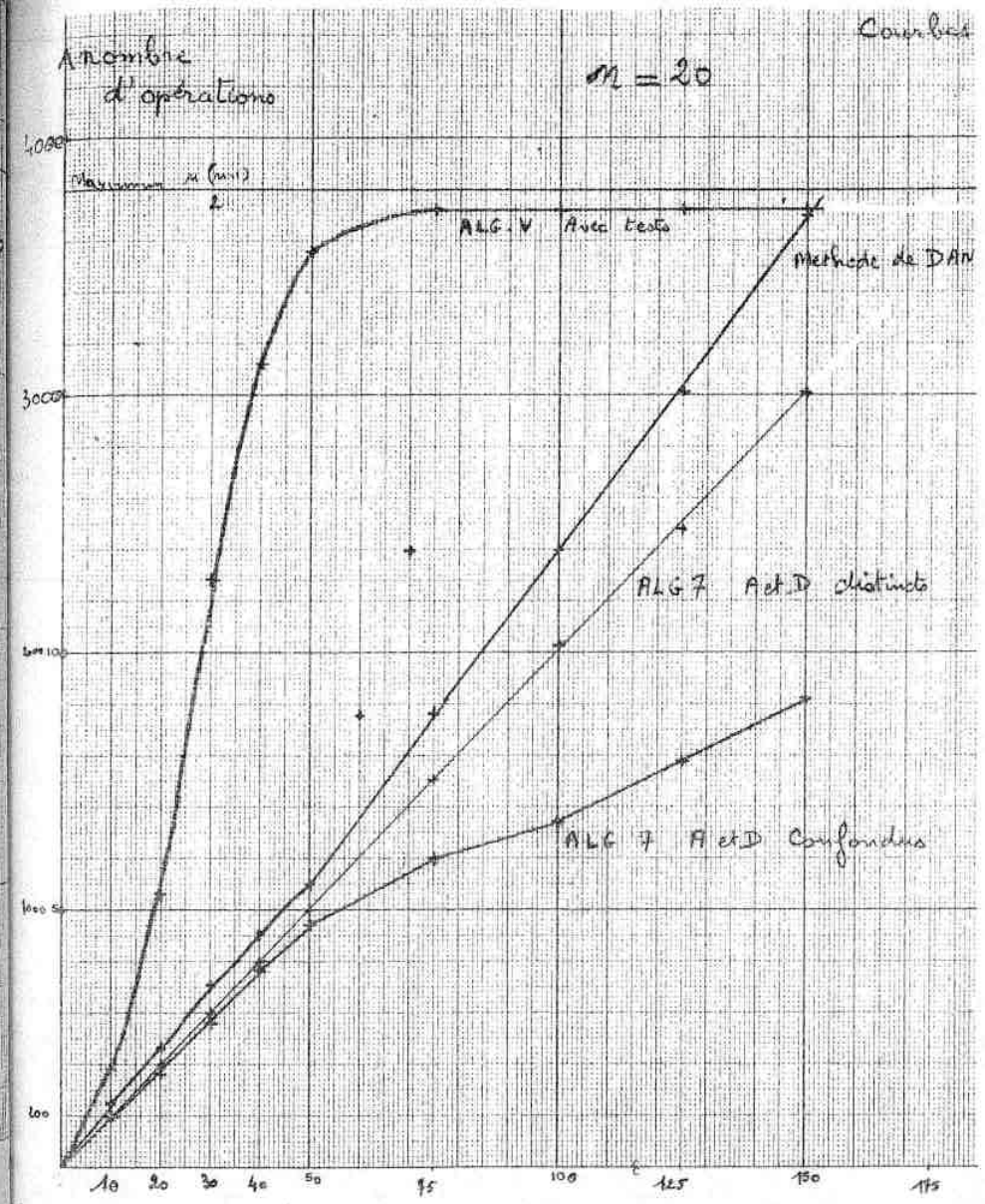
Courbes 6 (page 51-2) Description des graphes

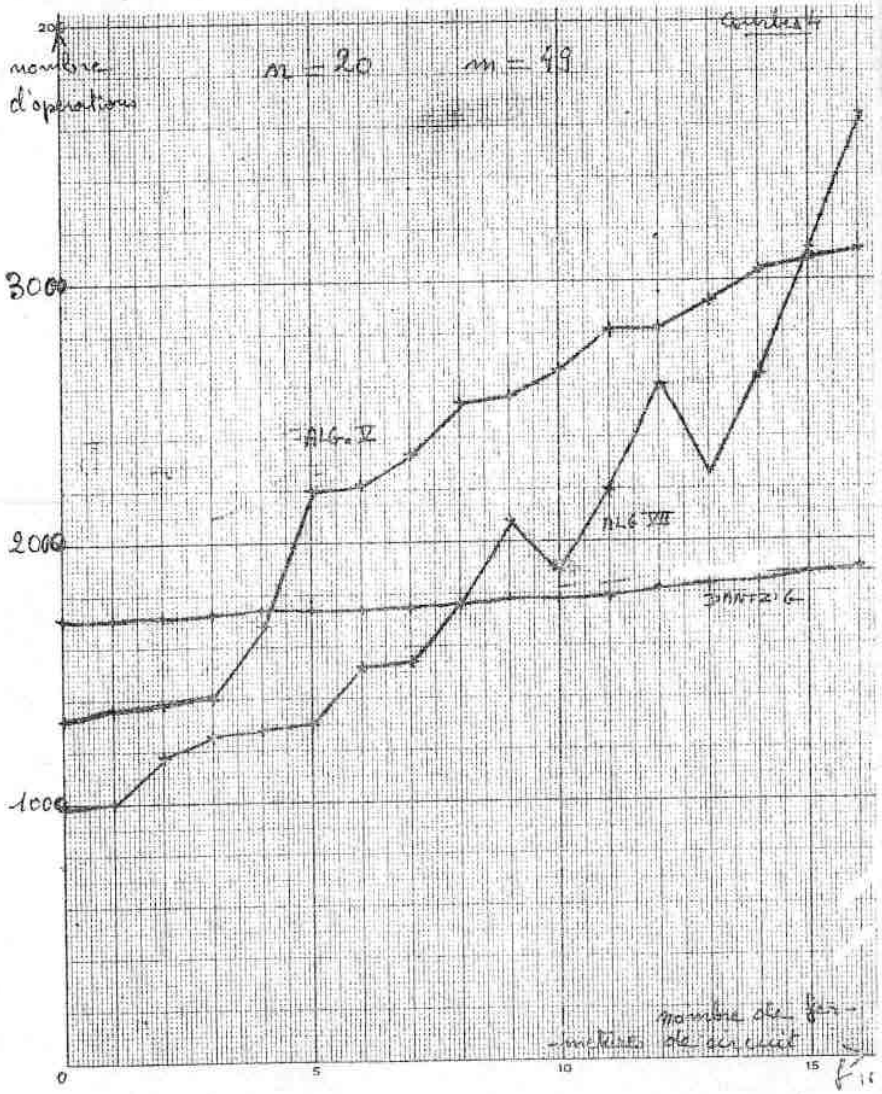
Elles montrent bien que l'algorithme 7 est aussi plus intéressant que l'algorithme de Dantzig dans le cas de graphes ne possèdant pas beaucoup de circuits (jusqu'à 25 fermetures de circuits différentes sur 30 points).

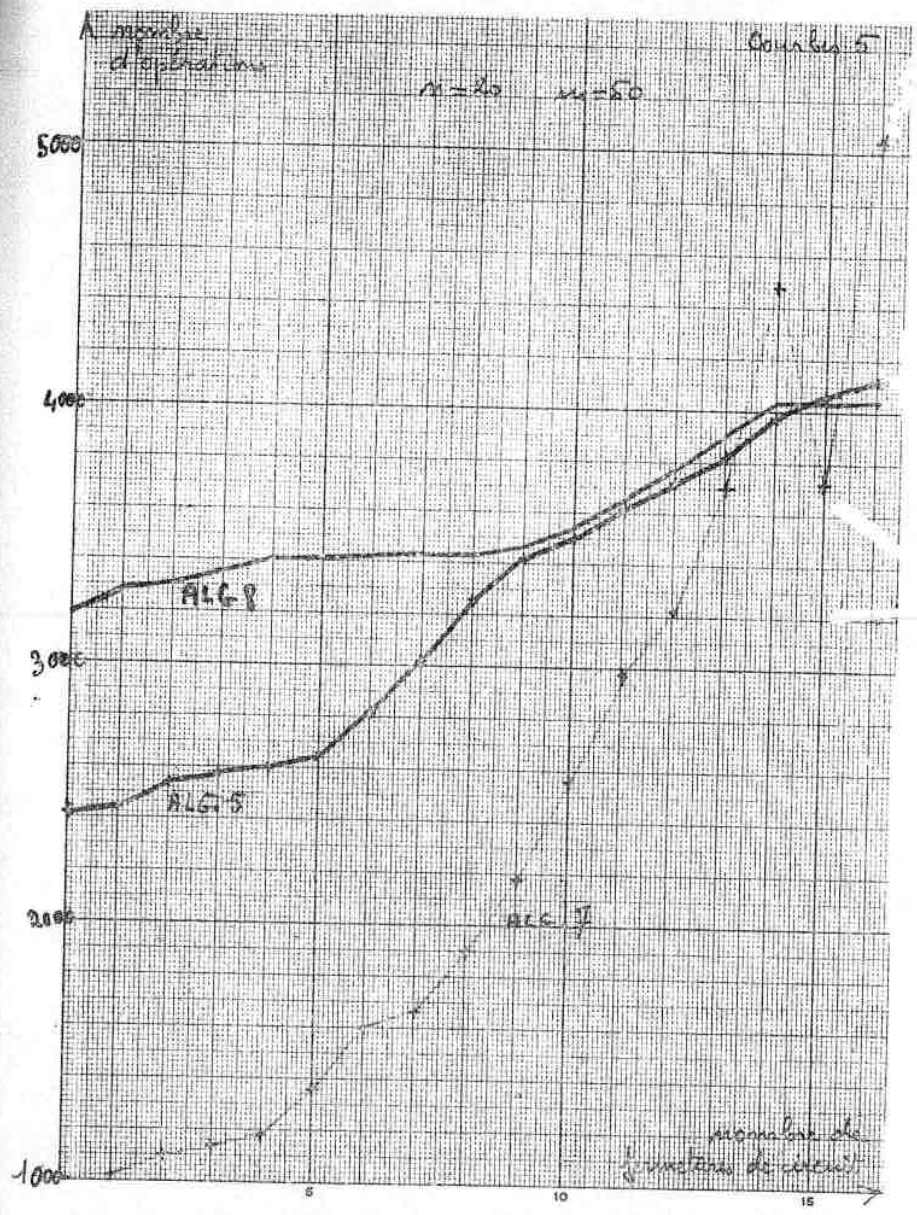
Les courbes 4,5, et 6 permettent aussi de constater que les rédactions proposées pour les trois algorithmes permettent de traiter le problème avec des nombres d'opérations bien inférieurs à n^3 qui vaut 8 000 pour 4 et 5, 27 000 pour 6.

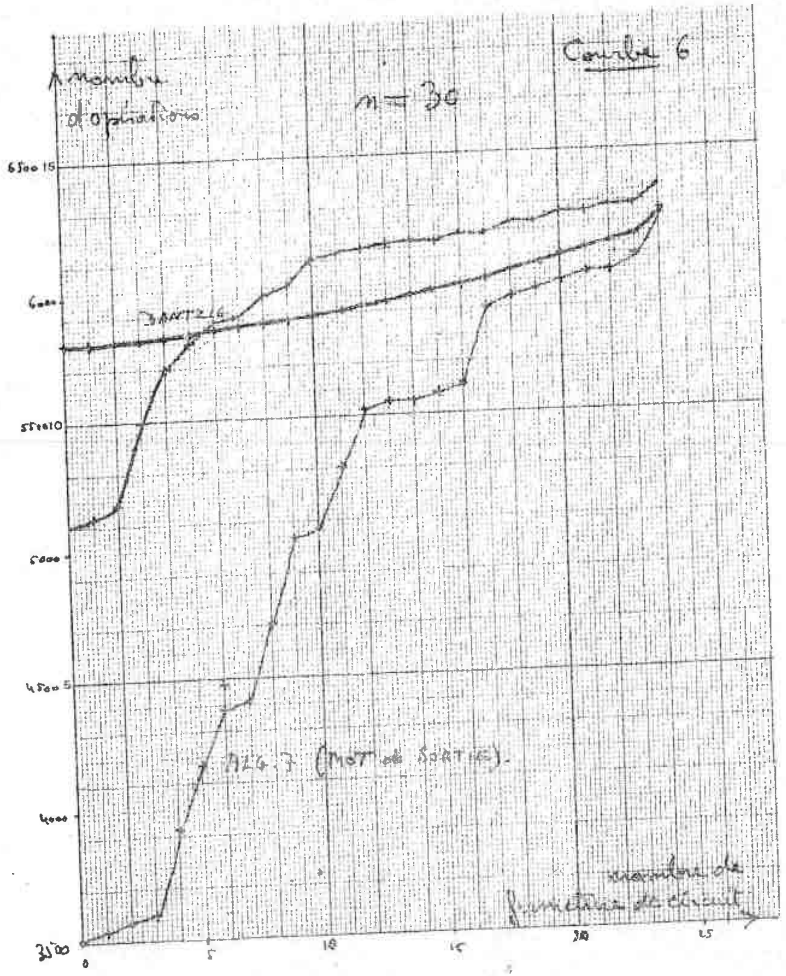



ALGORITME V Avec tests m = 20
GRAPHES SANS CIRCUITS
Maximum max((n-1)/2)
nombre d'opérations = g(nombre d'arcs) x 2
nombre de "comparaisons opérant" = g(m)
Temps = g(arcs)
200
50
100
50
10
10 20 30 40 50 75 100 125 150 175
m = nombre d'arcs


Courbes
Nombre d'opérations
m = 20
Maximum n (nombre)
2
ALG. V Avec tests
Méthode de DAN
ALG 7 A et D distincts
ALG 7 A et D Confondus
1000
3000
500 1000
1000 5
100
10 20 30 40 50 75 100 125 150 175


nombre d'opérations
n = 20
m = 49
ALG. V
ALG VII
DANTZIG
3000
2000
1000
0
5
10
15
nombre de fonctions de circuit


nombre d'opérations
n = 20
m = 50
5000
4000
3000
2000
1000
ALG 8
ALG 5
ALG V
5
10
15
nombre de fonctions de circuit



CHAPITRE VI

EXISTENCE DE CHEMINS ENTRE TOUT COUPLE DE POINTS DU GRAPHE
RECHERCHE DE LA MATRICE DE FERMETURE TRANSITIVE.

On utilise l'homomorphisme défini au chapitre II pour lequel $E = \{VRAI, FAUX\}$; *si* $K \in F^*$, $\Pi(K) = VRAI$ *si, et seulement si,* $K \neq \Lambda$; $+ = \vee$, $X = \wedge$.

Trois algorithmes peuvent être retenus pour ce problème : 5.7.8. Nous étudierons ici la forme qu'ils peuvent prendre et les améliorations qu'on peut leur apporter pour traiter ce problème.

I - ALGORITHME 5 :

```
PROCEDURE MFTALG5 (A,N) ; ENTIER N ; BOOLEEN TABLEAU A ;
   DEBUT ENTIER I,J,K ;
   POUR I:=1 PAS 1 JUSQUA N FAIRE
    POUR J:=1 PAS 1 JUSQUA N FAIRE
       SI  A [J,I] ALORS
    POUR K:=1 PAS 1 JUSQUA N FAIRE
       A [J,K] := A [J,K] OU A [I,K]
   FIN ;
```

On peut constater que la rédaction de l'algorithme 5 devient particulièrement simple.

L'opération

$$\text{pour } i = 1,2,\ldots n \quad (\forall x \in E \text{ et } K(x,z_i,i-1) \neq \emptyset)\ (\forall y \in E)\ \left(K(x,y,i) = K(x,y,i-1) \cup \left[K(x,z_i,i-1) \odot K(z_i,y_i,i-1)\right]\right)$$

devient, les points $z_i \in E$ *étant notés* $1,2,\ldots n$:

$$\text{pour } i=1,2,\ldots n \quad (\forall x \in E \text{ et } K(x,i,i-1) = \text{'VRAI'})\ (\forall y \in E)\ \left(K(x,y,i) = K(x,y,i-1) \vee \left[K(x,i,i-1) \wedge K(i,y,i-1)\right]\right)$$

qu'on peut écrire aussi

$$\text{pour } i=1,2,\ldots n \quad (\forall x \in E)\ (\forall y \in E)$$

$K(x,y,i)$ = *SI* $K(x,i,i-1) = VRAI$ *ALORS* $K(x,y,i-1)$
SINON $K(x,y,i-1) \wedge K(i,y,i-1)$.

Notons que cette présentation permet d'éviter, quand K(x,i,i-1) = FAUX exactement n opérations.
Le nombre d'opérations est donc égal à n^2 tests K(x,i,i-1)='VRAI' et moins de n^3 opérations ou.
Signalons également la modification qui nous fut suggérée par Mr GIRAULT (Institut Blaise Pascal) qui ne correspond pas à une modification théorique mais à une modification de programmation:en algol, la recherche d'un élément de tableau à une dimension se faisant plus rapidement que celle d'un élément de tableau à deux dimensions.
La rédaction proposée est donc la suivante.

```
PROCEDURE MFT5 (A,N) ; ENTIER N ; BOOLEEN TABLEAU A ;
   DEBUT ENTIER I,J,K ; BOOLEEN TABLEAU B [1:N] ;
         POUR I:=1 PAS 1 JUSQUA N FAIRE
         POUR J:=1 PAS 1 JUSQUA N FAIRE
             DEBUT B [J] := A [I,J] ;
             SI A [J,I] ALORS
             POUR K:=1 PAS 1 JUSQUA N FAIRE
             A [J,K] :=A [J,K] OU B [K] FIN FIN ;
```

II - ALGORITHME 8 :

```
PROCEDURE MFT8 (A,N) ; ENTIER N ; BOOLEEN TABLEAU A ;
DEBUT ENTIER I,J,K,L ; BOOLEEN TABLEAU B,C [1:N] ;
      POUR K:=1 PAS 1 JUSQUA N FAIRE
   DEBUT
      POUR I:=1 PAS 1 JUSQUA K-1 FAIRE DEBUT B [I]:=A [K,I] , C [I]:=
                                                          A [I,K] FIN
      POUR J:=1 PAS 1 JUSQUA K-1 FAIRE SI A [K,J] ALORS
      POUR L:=1 PAS 1 JUSQUA K-1 FAIRE B [L] := B [L] OU A [J,L] ;
      POUR J:=1 PAS 1 JUSQUA K-1 FAIRE SI A [J,K] ALORS
      POUR L:=1 PAS 1 JUSQUA K-1 FAIRE C [L] := C [L] OU A [L,J] ;
      POUR I:=1 PAS 1 JUSQUA K-1 FAIRE SI C [I] ALORS
      POUR J:=1 PAS 1 JUSQUA K-1 FAIRE A [I,J] := A [I,J] OU B [I] ;
      POUR L:=1 PAS 1 JUSQUA K-1 FAIRE DEBUT A [K,L] := B [L] ;
           A [L,K] := C [L] FIN ;
   FIN  FIN ;
```


Là encore la rédaction est simplifiée du fait de l'information sur A(i,k) apportée au moment même de la recherche des points i pour lesquels il faut faire l'opération. Toutefois, comme pour toutes les utilisation de l'algorithme 8, il semblerait préférable de ne faire des tests A(i,k) ='VRAI, A [k,j] ='VRAI' que pour des valeurs de k suffisamment grandes.

III- ALGORITHME 7

Les opérations à effectuer deviennent.

1) Initialisations

$(\forall x \in E)\ (\forall y \in E)\ (K[x,y,0] = \text{'FAUX'}$

2) e étant le sommet et r le nombre des sorties de la pile :
(pour $i=1,2,\ldots r$)

a) si $\sigma_i < \Sigma(z)$, e étant le sommet

$K(e,z,i) = K(e,z,i-1)$ ou (e,z)

b) σ_i est sortie de $z > \Sigma(z)$ ou $(i = \Sigma(z)$ et z non fermeture de circuit).

$(\forall y \in E, y \neq z)\ (K(e,y,i) = K(e,y,i-1)$ ou $[(e,z)$ et $K(z,y,i-1)])$

c) $\sigma_i = \Sigma(z)$, z fermeture de circuit

$(\forall x \in E)\ (\forall y \in E)\ (K(x,y,i) = K(x,y,i)$ ou $[K(x,z,i-1)$ et $K(z,y,i-1)])$

d) pour tout couple x,y non défini ci-dessus

$K(x,y,i) = K(x,y,i-1)$.

On pourra se reporter à la procédure ALG7PILE du chapitre III et au programme figurant en annexe.
On peut apporter à cet algorithme des modifications tenant compte de la nature particulière de ce problème : il n'est pas indispensable de décrire et d'expliciter tous les circuits comme dans le cas général, il suffit au contraire de se limiter à la recherche des classes d'équivalence, notion plus importante pour ce problème que la notion de circuit : deux points équivalents sont pour la fermeture transitive en relation avec les mêmes points. D'autre part on peut remarquer que puisque l'opération 2b s'effectue lors de la sortie de z de la pile annexe (sortie de l'arc ez de la pile attachée) l'élément (e,z) de la matrice associée a pour valeur 'vrai' et il est inutile d'effectuer l'opération 'ET'.

Notons que la première méthode de ce type, utilisant la fonction lien a été apportée par EMOND [18] .

IV - DETERMINATION DES CLASSES D'EQUIVALENCE :

Toute cette partie est largement inspirée des travaux de EMOND [18] et PAIR [45] .

Rappelons qu'il s'agit des classes de la relation d'équivalence : $x \sim y$ si, et seulement si $x \Gamma^* y$ et $y \Gamma^* x$ (D.4) . (qui est aussi la relation "appartenir à un même circuit").

IV-1 Base de classe :

Parmi les points y d'une classe, celui pour lequel $\mathcal{E}(y)$ est le plus petit sera appelé base de classe.

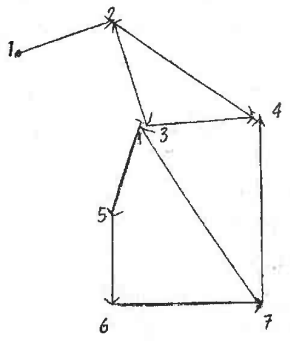
Montrons que c'est aussi celui pour lequel $\Sigma(y)$ est le plus grand :

Soit v la base de classe de x ; un chemin joint v à x ; nous montrerons que $\Sigma(v) \geqslant \Sigma(x)$ par récurrence sur le nombre d'arcs de ce chemin : c'est clair si ce nombre est 0 ; sinon, x' est l'avant dernier point sur ce chemin et $\Sigma(x') \leqslant \Sigma(v)$; soient j et j' l'entrée et la sortie de x correspondant à l'arc x' x :

$$\mathcal{E}(v) \leqslant \mathcal{E}(x) \leqslant j < j' < \Sigma(x') \quad \text{(lemme 2)} ;$$

d'où $\mathcal{E}(v) \leqslant \mathcal{E}(x) < \Sigma(v)$ et par suite $\Sigma(x) \leqslant \Sigma(v)$.

exemple :



Une pile annexe à ce graphe :

1 2	base de classe
1 2 3	
1 2 3 4	
1 2 3 4 2	1er de circuit
1 2 3 4	
1 2 3	
1 2 3 5	
1 2 3 5 6	
1 2 3 5 6 7	
1 2 3 5 6 7 3	1er de circuit

1 2 3 5 6 7	
1 2 3 5 6 7 4	1er de circuit
1 2 3 5 6 7	
1 2 3 5 6	
1 2 3 5	
1 2 3	
1 2	
1	base de classe

2 est premier de circuit comme il est aussi le premier point de la classe 2,3,4,5,6,7 à entrer dans la pile

1 est aussi base de classe, étant seul dans cette classe

IV-2 Fonction lien :

Rappelons la définition de l'application ℓ (fonction lien) que donne Monsieur PAIR [45] .

A tout point x et à tout entier i de 0 à m' (nombre d'arcs) associons $\ell(x,i)$, extrémité d'un représentant en i d'un chemin (pas nécessairement élémentaire) de x à v s'il en existe, égal à x sinon : $\ell(x,i)$ est équivalent à x. (v est la base de classe de x).

L'application ℓ sera utile pour trois raisons :

1) elle permet de déterminer pour chaque point x sa base de classe v car $\ell(n,m')$ est la base v de la classe de x.
2) elle permet de reconnaître les bases de classes : pour un tel point x $\ell(x,j) = x$ quel que soit j
3) soient v la base de classe d'un point x, et i un entier entre o et m', ℓ permet de comparer σ_i à $\Sigma(v)$ (cf. PAIR 45)

L'utilisation de la fonction lien permet donc de repérer les classes d'équivalences : la classe d'équivalence de base v est l'ensemble des x tels que $\ell(x, \Sigma(v)) = v$.

Encore faut-il préciser l'évolution de cette fonction lien :

1) $\ell(x,o) = x$ pour tout $x \in E$
2) pour $i=1,2,\ldots,r$ nombre (nombre de sorties de la pile σ_i est une sortie de z_i)
   a) si $\mathcal{E}[\ell(z_i,i-1)] < \sigma_i < \Sigma[\ell(z_i,i-1)]$, $\ell(e,i) = \ell(z_i,i-1)$
   b) si $\sigma_i = \Sigma(z_i)$, z_i fermeture de circuit et $\ell(z_i,i-1) \neq z_i$: $\ell(x,i) = \ell(z_i,i-1)$ pour les x tels que $\ell(x,i-1) = z_i$
   c) $\ell(x,i) = \ell(x,i-1)$ pour les points x dont $\ell(x,i)$ n'est pas défini par ce qui précède.

V - ALGORITHME

1) $(\forall x \in E)\ (\forall y \in E)$

$(K\ [x,y,o] =$ Si $x \Gamma y$ ALORS VRAI SINON FAUX$)$

2) pour $i = 1,2,\ldots,m$:

a) si $\sigma(z_i)$ $\Sigma(v_i)$ où v_i est la base de la classe z_i

(c'est-à-dire si $\sigma_i > \Sigma(\ell(z_i,i-1))$:

$(\forall y \in E)\ (K\ [e,y,i] = K\ [e,y,i-1] \vee K\ [z_i,y,i-1]\ )$

b) si $\sigma(z_i) = \Sigma(z_i)$ et z_i base de classe ,

(c'est-à-dire $\ell(z_i,i-1) = z_i$) :

. $(\forall y \in E)\ (K\ [z_i,y,i] = \bigvee_{x \cap z_i} K\ [x,y,i-1]\ )$

. $(\forall x \sim z_i$, c'est-à-dire tel que $\ell(x,i-1) = z_i)(\forall y \in E)$

$(K\ [x,y,i] = K\ [z_i,y,i]\ )$

. $(\forall y \in E)\ (K\ [e,y,i] = K\ [e,y,i-1] \vee K[z_i,y,i]\ )$

c) Pour les autres couples x,y $K\ [x,y,i] = K\ [x,y,i-1]$.

Cet algorithme permet de trouver la fermeture transitive stricte de la relation ; pour obtenir la fermeture transitive il faut aussi effectuer dans 1)

$K(x,x,o) =$ VRAI pour tout x.

VI - REALISATION :

Nous présenterons deux réalisations différentes quand au mode de représentation de la fonction lien ou au choix de la définition du lien lors de l'opération 2 a.

VI-1 Lien dans la matrice :

Dans le cas où pour le matériel utilisé, la représentation interne d'un booléen nécessite autant de place que pour un entier (cas fréquent) on pourra avoir intérêt à utiliser cette méthode qui consiste à représenter les liens dans la matrice associée elle-même.
A l'instant i

$A(x,y) = -1$ si et seulement si, $\ell(x,i) = y$.

On définit $\ell(e,i)$ dans l'opération 2 a par $\ell(e,i) = \ell(z,i-1)$ même si e avait déjà un autre lien.

VI-2 Lien explicite :

Un tableau (ce peut être celui contenant les indicateurs défini au chapitre III §II.5) est utilisé pour contenir en I le nom du lien de I.
On définit $\ell(e,i)$ dans l'opération 2a par

$\ell(e,i) = \ell(z,i-1)$ si e n'avait pas de lien.

Si e a un lien, soit j_e le niveau de $\ell(e,i-1)$ dans la pile, j_z celui de $\ell(z,i-1)$

$\ell(e,i) = \ell(e,i-1)$ si $j_e < j_z$

$\ell(z,i-1)$ sinon.

Les méthodes VI-2 et VI-3 ont été programmées et comparées à VI-1 et aux deux formes de l'algorithme 5.

VI-3 Nombre d'opérations :

a) Algorithme 5

Le nombre d'opérations ET et OU est inférieur à n^3 du fait du test "Si A $[J,I]$ " mais reste cependant de l'ordre de n^3, ce test ne porte pas sur les arcs du graphe mais sur les chemins ne passant pas par des points d'indice supérieur à I. A $[J,I]$ est donc vrai un nombre de fois bien supérieur au nombre d'arcs (cf chapitre 7

b) Algorithme 7

Nous avons vu (§ III de ce chapitre) que les opérations 'ET' sont inutiles. Le nombre d'opérations de gestion de la pile est proportionnel au nombre d'arcs.
Les opérations 'OU' peuvent être classées en :

- opérations a) pour les $\sigma(z_i) > \Sigma(v_i)$ v_i base de la classe de z_i, et opérations b 3):

$(\forall y \in E)\ (K\ [e,y,i] = K\ [e,y,i-1] \vee K\ [z_i,y,i]\ )$

Leur nombre est donc égal aux nombres d'arcs du graphe diminué du nombre d'arcs entraînant des sorties inférieures à des sorties vraies.
Chaque opération a au b 3 représente n opérations "OU".

- opérations b 1)

Leur nombre est majoré par n.
En effet les points du graphe ne peuvent appartenir à deux classes différentes. Chaque opération b 1) représente n opé-

tions "OU".

Il faut ajouter à ces opérations les opérations b 2 beaucoup plus rapides et dont le nombre est lui aussi majoré par n, donc ajouter un nombre de "transferts" inférieur à n^2.

On peut donc conclure que dans le cas des graphes peu plein ($m/n^2 << 1$) le nombre de couples qui opèrent est de l'ordre de m+n donc reste de l'ordre de n. Comme chaque couple opère n opérations 'OU' (au lieu d'opérations 'ET' et 'OU' pour les algorithmes 5 et 8) cette méthode sera nettement avantageuse. Dans le cas des graphes pleins (m/n^2 voisin de 1), il y a m, donc près de n^2, couples qui opèrent par a). Il y a donc un nombre d'opération 'OU' de l'ordre de n^3. Comme on n'effectue pas d'opérations 'ET' il reste plus intéressant que les autres. Notons que c'est le seul exemple d'utilisation de l'algorithme 7 où l'augmentation du nombre de fermetures de circuit à l'intérieur d'une classe d'équivalence donnée n'augmente pas le nombre d'opérations ∪. (elle augmente au contraire le nombre de sorties inférieures à des sorties vraies, donc elle diminue le nombre d'opérations).

VI-4 Résultats

Les exemples cités ici sont des graphes d'ordre 40.

- L'exemple 1 n'a pas de circuits, possède 3 boucles, 18 feuilles, 39 arcs.
- L'exemple 2 possède 1 circuit à 4 points, 2 boucles, 14 feuilles, 50 arcs
- L'exemple 3 possède 2 petits circuits, pas de boucles, 18 feuilles, 52 arcs.
- L'exemple 4 possède 5 fermetures de circuit, 3 boucles, 55 arcs, 37 points appartenant à la même classe d'équivalence.
- L'exemple 5 possède 10 fermetures de circuit, 57 arcs 38 points appartenant à la même classe d'équivalence.

Lien explicite

```
'DEBUT' 'ENTIER' N,I,J,K,C,R,L;
DEB:   EXL(<#GRAPHE_D'ORDRE#>);IMPR; LICLAV(N);
       'DEBUT' 'ENTIER' 'TABLEAU' M.(1:N,1:N).,T.(2:N).,P.(1:N-1).;
       'ALGUILLAGE' AIG:=TEST, LIEN, TRA, SOR;
       LIRT(N); C:=0; R:=0;
       'SI' CLE(6) 'ALORS' 'DEBUT' EXL(<#MATRICE_ASSOCIEE#>);IMPR;'ALLERA' RES 'FIN';
DEBCL: 'POUR' I:=1'PAS' 1 'JUSQUA' N 'FAIRE' T.(I):=1 ; K:=1 ;
INIT : 'POUR' I:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
       'SI' T.(I). 'EGAL' 1 'ALORS' 'ALLERA' INIP ; R:=1 ; 'ALLERA' EXIT ;
INIP : P.(I):=I ;
TEST : 'POUR' J:=1 'PAS' 1 'JUSQUA' N 'FAIRE' 'SI' M.(I,J). 'EGAL' 1 'ALORS'
       'DEBUT' 'SI' I 'EGAL' J 'ALORS' 'ALLERA' BI;
       T.(I):=2;'ALLERA' ENTR 'FIN';T.(I):=4;
       'ALLERA' SOR ;
ENT  : 'SI' T.(I). 'EGAL' 4 'ALORS' T.(I). :=2 ;
ENTR : K:=K+1;P.(K).:=J;
       I:=J ; 'ALLERA' 'SI' T.(I). 'INF' C 'ALORS' RESOL 'SINON' AIG.(T.(I).). ;
RESOL: 'SI' T.(I). 'DIF' -1 'ALORS' 'ALLERA' TRALIEN ;
       T.(I):=3 ; EXL(<#CLASSE D'EQUIVALENCE#>) ; EXE(3,I) ; IMPR ;
       L:=0 ; 'POUR' J:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
       'SI' T.(J). 'EGAL' -1 'ALORS' 'DEBUT' 'POUR' R:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
       'SI' M.(J,R). 'SUP' C 'ALORS' M.(I,R).:=2 ; C:=C+1 'FIN' ;
       'POUR' J:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
       'SI' T.(J). 'EGAL' -1 'ALORS' 'DEBUT' T.(J).:=3 ; C:=C+1 ; EXE(4,J) ; L:=L+1 ;
       'POUR' R:=1 'PAS' 1 'JUSQUA' N 'FAIRE' M.(J,R).:=M.(I,R). ;
       'SI' L 'EGAL' 0 'ALORS' 'DEBUT' IMPR ; L:=0 ; 'FIN' 'FIN' ;
       'SI' L 'SUP' 0 'ALORS' IMPR ;
TRA  : T.(I).:=3;'SI' K 'EGAL' 1 'ALORS' 'ALLERA' INIT;
       R:= P.(K-1).;C:=C+1;
       'POUR' J:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
       'SI' M.(I,J). 'SUP' 0 'ALORS' M.(R,J).:=2 ;
       T.(I).:=3 ;
SOR  : 'SI' K 'EGAL' 1 'ALORS' 'ALLERA' INIT;
       K:=K-1;
```

```
SORTIE: R:=I-1; I:=P.(K).;
        'POUR' J:=R 'PAS' 1 'JUSQUA' N 'FAIRE'
          'SI' M.(I,J). 'EGAL' 1 'ALORS' 'ALLERA' 'SI' I 'EGAL' J 'ALORS' BOU 'SINON' ENT ; J:=T.(I). ;
          'ALLERA' 'SI' J 'EGAL' 2 'ALORS' TRA 'SINON'
                   'SI' J 'EGAL' 4 'ALORS' SOR 'SINON' RESOL ;
TRALIEN : 'POUR' J:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
            'SI' T.(J). 'EGAL' -1 'ALORS' T.(J).:=T.(I). ;
L:=T.(I). ;'ALLERA' EVOLIEN;
LIEN    : L:=-1 ;
EVOLIEN: J:=P.(K-1). ;'SI' T.(J). 'SUP' 0 'ALORS' 'DEBUT' T.(J).:=L ; 'ALLERA' SOR 'FIN' ;
           'POUR' R:=K-1 'PAS' -1 'JUSQUA' 1 'FAIRE'
             'SI' P.(R). 'EGAL' -L 'ALORS' 'ALLERA' SOR
           'SINON' 'SI' P.(R). 'EGAL' -T.(J). 'ALORS' 'DEBUT' T.(J).:=L;'ALLERA'SOR 'FIN';
BI  :  T.(J).:= 4 ;
BOU :  EXL(<'BOUCLE'>) ; EXE(3,1) ; IMPR ; 'ALLERA' SORTIE;
EXIT :  EXE(3,C) ; IMPR ;
        EXL(<'MATRICE DE FERMETURE TRANSITIVE'>) ; IMPR;
RES  :  'POUR' I:=1 'PAS' 1 'JUSQUA' N 'FAIRE' 'DEBUT' EXE(3,I) ;
          ESPACE(2) ; 'POUR' J:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
          'SI' M.(I,J). 'EGAL' C 'ALORS' EXL(<C>) 'SINON' EXL(<_>) ;
          IMPR 'FIN' ;
          'ALLERA' 'SI' R 'EGAL' C 'ALORS' DEBCL 'SINON' DEB ;
          'FIN'

'FIN'
#
```


	ALG 5	Procedure MOTS	ALG 8	Lien dans la matrice	Lien explicite	
mémoires occupées par les données	n^2+5	n^2+n+5	n^2+2n+6	n^2+2n+8	n^2+2n+8	
On obtient aussi				les classes d'équivalence	"	
Exemple 1	95"	58"	40"	31"	25"	
	112	112	*22	17	17	
Exemple 2	175"	105"	*50"	40"	34"	
	263	263	*42	36	34	
Exemple 3	130"	70"	84"	55"	30"	
	188	188	*56	40	39	
Exemple 4	460"	270"	155"	112"	70"	
	703*	703	*100	75	71	
Exemple 5	840"	470"	185"	133"	90"	
	1099	1099	*120	80	75	

* indique qu'il s'agit ici du nbre d'opérations élémentaires pour m, puisqu'on ne peut plus parler de "temps" que ci-avant.

CHAPITRE VII

PROBLEMES DE PLUS COURTS CHEMINS

INTRODUCTION :

Nous avons vu au chapitre II, qu'on pouvait définir des homomorphismes π de $(F^*, \cup, \odot)$ dans $(E, +, \times)$ pour les problèmes de plus courte distance, et de plus court chemin, de plus longue distance et de plus long chemin, de chemin le plus probable et sa probabilité, de chemin le moins probable et sa probabilité, ce qui nous amène à donner des définitions très voisines des différents ensembles E, et des différentes opérations + et × associées. Nous pouvons grâce à l'introduction des "longueurs généralisées" ramener ces quatre problèmes à un seul type [45] .

"Soit D un ensemble muni d'une loi de composition interne $\oplus$ associative, possédant un élément neutre 0 et un élément absorbant ∞, totalement ordonné par une relation compatible avec + :

$$\ell_1 < \ell_2 \Rightarrow \ell \oplus \ell_1 < \ell \oplus \ell_2$$

et $\ell_1 \oplus \ell < \ell_2 \oplus \ell$, pour laquelle ∞ est élément maximum. A tout arc α d'un graphe, on associe une longueur généralisée $|\alpha| \in D$ et on définit la <u>longueur généralisée</u> $|a|$ d'un chemin généralisé $a = [\alpha_1, \ldots, \alpha_n]$ comme $|\alpha_1| \oplus \ldots \oplus |\alpha_n|$ et $|e| = 0$. Nous dirons que les longueurs sont <u>positives</u> dans le cas où 0 est minimum dans D.

Définissons E et ses lois de composition + et ×

$E = F \cup \{\omega\}$, $|\omega| = \infty$,

$a+b$ = SI $|a| \leq |b|$ ALORS a SINON b,

$a \times b = a.b$ en convenant que $a.\omega = \omega = b.\omega$.

L'application π_0 qui associe à toute suite K non vide son plus court chemin (au sens fixé plus haut) est telle que $\pi_0(\wedge) = \omega$ est un homomorphisme de $(F^*, \cup, \odot)$ dans, $(F \cup \omega, \pm, \times)$. *

Soit ϖ une application de $F \cup \{\omega\}$ dans D associant à chaque élément sa longueur :

+ définie par : $a+b$ est le minimum de a,b,

× est l'opération $\oplus$; ω est un homomorphisme de $(F \cup \omega, \pm, \times)$ dans $(D, +, \times)$. L'application $\pi = \varpi \circ \pi_0$ est un homomorphisme de $(F^*, \cup, \odot)$ dans $(D, +, \times)$ qui associe à toute suite non vide la plus petite des longueurs généralisées de ses chemins ($\varpi \circ \pi_0(\wedge) = \infty$).

* $\pm$ et $\times$ sont les opérations + et × définies page 13 §I.5

Dans le cas des longueurs positives, la plus petite des longueurs généralisées des chemins joignant un point x à un point y (y compris si x = y) est celle d'un chemin élémentaire : on l'appelle distance de x à y ; elle est égale à $\Pi(K)$ si K est formé de chemins joignant x à y, parmi lesquels tous les chemins élémentaires.

Dans le cas de longueurs non toutes positives, les algorithmes resterons valables tant qu'il n'y aura pas de circuit de longueur totale négative. Il faut donc que l'algorithme de construction de suites de chemin choisi construise également des circuits dans ces suites, ce qui est le cas pour tous les algorithmes envisagé. On pourra donc, dans ce cas, résoudre le problème suivant : Recherche des courtes distances (et plus courts chemins, au sens généralisé) entre tout couple de points du graphe s'il n'y a pas de circuit de longueur totale négative, et détection de ces circuits s'il en existe.

Pour le problème de la plus courte distance (sens habituel) on défini D par l'ensemble des réels positifs

+ par : a+b est le minimum de a,b

x par l'addition usuelle.

On peut étendre D à l'ensemble des réels s'il n'existe pas de circuit de longueur totale négative.

Plus longues distances :

D = ensemble des réels négatifs

+ par : a+b est le maximum de a,b

x par l'addition usuelle.

On peut étendre D à l'ensemble des réels s'il n'existe pas de circuit de longueur totale positive, a fortiori s'il n'existe pas de circuit.

Probabilité du chemin le plus probable

$D = \{x,\ 0 \leq x \leq 1\}$

+ : a+b est le maximum de a,b

x est la multiplication usuelle

Probabilité du chemin le moins probable

$D = \{x,\ 0 \leq x \leq 1\}$

+ : a+b est le minimum de a,b

x est la multiplication usuelle

I - PLUS COURTES DISTANCES ENTRE TOUS LES POINTS DU GRAPHE ET UN POINT EXTREMITE :

On pourrait écrire des algorithmes analogues à ceux qui vont suivre pour la recherche des plus courtes distances entre un point origine fixé et tous les points du graphe.

I.1 - Algorithme 2

I.1.1 Algorithme de FORD [20] , [4] :

n est l'indice du point extremité choisi, c'est aussi le nombre de points de E.

1°) Marquer chaque sommet avec l'indice ti

initialisation : $tn = 0$; $ti = \infty$ pour $i \neq n$

2°) On cherche un arc (x_i, x_j) tel que $tj - ti > l(x_i, x_j)$; on remplace alors tj par $ti + l(x_i, x_j)$. On continue jusqu'à ce qu'aucun arc ne permette plus de diminuer les ti.. Il s'agit en fait d'une variante de l'algorithme 2 appliquée au problème de la plus courte distance (longueur généralisée) :

$$K[x,y,o] = \underline{SI}\ x=y\ \underline{ALORS}\ 0\ \underline{SINON}\ \infty$$

$$K[x,y,i] = \min\left[K[x,y,i-1]\ ,\ \min K[x,z,i-1] \oplus J[z,y]\right]$$

$$= \min\left[K[x,y,i-1]\ ,\ K[x,z_1,i-1] \oplus J[z_1,y]\right.$$

$$\left.,\ldots,\ K[x,z_n,i-1] \oplus J[z_n,y]\right]$$

$z_1, \ldots z_n$ étant les points du graphe.

Notons que cette forme de rédaction permet de voir, plus clairement, qu'il est possible de se fixer un nombre maximum d'arcs dans chaque chemin soit p; il suffit de calculer les $K[x,y,p]$.

Sous cette forme il est facile de voir qu'on peut éviter un certain nombre d'opérations lorsque $J[z_1,y] = \wedge$

On peut écrire le programme suivant pour lequel nombre d'opérations est inférieur à n^3 pour chaque ORIG fixé.

```
'DEBUT''COMMENTAIRE' PROBLEME DU PLUS COURT CHEMIN
                     MATRICE DES LONGUEURS . ALGORITHME DE FORD ;
  'ENTIER' I,J,N, INDIC, ORIG;
  LICLAV (N) ;
'DEBUT''ENTIER''TABLEAU' L.(1:N,1:N)., LAMBDA.(1:N). ;
DEB:LIBA(O,L); INDIC:=O ; ORIG:=O ;
                'POUR' I :=1 'PAS' 1 'JUSQUA' N 'FAIRE'
                'POUR' J :=1 'PAS' 1 'JUSQUA' N 'FAIRE'
                'SI' L.(I,J). 'EGAL' O 'ALORS' L.(I,J).:=-1;
EXL(<DEPART>); IMPR;PT DE DEPART: ORIG:=ORIG+1;
                'SI' ORIG 'SUP' N 'ALORS''ALLERA' END ;
                'POUR' I:=1 'PAS'1 'JUSQUA' N 'FAIRE' LAMBDA.(I).:=90000 ;
LAMBDA.(ORIG).:=O ; INDIC:=O ;
DEBCL:
                'POUR' I:=1 'PAS'1 'JUSQUA' N 'FAIRE'
                'POUR' J:=1 'PAS'1 'JUSQUA' N 'FAIRE'
                'SI' L.(I,J). 'SUG' O 'ALORS' 'DEBUT''SI'
                  LAMBDA.(J).-LAMBDA.(I).'SUP'L.(I,J). 'ALORS'
                'DEBUT'
                  LAMBDA.(J).:=LAMBDA.(I).+L.(I,J).;
                  INDIC:=1
                'FIN''FIN' ;
                'SI' INDIC 'EGAL'1 'ALORS''DEBUT' INDIC:=O ;
                                  'ALLERA' DEBCL 'FIN' ;Impression (LAMBDA) ;
'ALLERA' PT DE DEPART ;
END: EXL(<TERMINE>); IMPR;'ALLERA'DEB'FIN'
'FIN'
#
```


I.1.2 - Algorithme de Bellmann Kalaba [3] [4] :

Il est connu comme une variante du précédent.

On résoud le système d'équations :

$$v_i = \min_j \left[\ell(x_i, x_j) + v_j \right], \quad (i = 1,2,\ldots,n-1) \quad (j = 1,2,\ldots,n)$$

$v_n = 0$ *(n est l'indice du point extrémité choisi c'est aussi le nombre de point).*

On opère comme suit :

$$(i = 1,2,\ldots n-1)$$

$$(v_i^1 = \ell(x_i, x_n).$$

$$v_n^1 = 0$$

$$\left(\begin{array}{l} (k = 2,\ldots n) \quad (i = 1,2,\ldots n-1) \\ v_i^k = \min_j \left[\ell(x_i, x_j) + v_j^{k-1} \right] \\ \qquad\qquad (j = 1,2,\ldots n) \\ v_n^k = 0 \end{array} \right)$$

On s'arrête lorsque $(\forall i)\ (v_i^k = v_i^{k-1})$

Il s'agit cette fois de l'algorithme 2 sous sa forme initale :

- $K\,[x,y,o]$ = *SI* $x=y$ *ALORS 0 SINON* ∞
- $K\,[x,y,i] = \min_{z \in E} \left[K\,[x,z,i-1] \oplus j'\,[z,y] \right]$

$$j'\,[z,y] \quad \begin{cases} \min\,(0, |arc\ zz|) & si\ z=y \text{ et } z \Gamma z \\ 0 \ si\ z=y \text{ et } (non\ z \Gamma z) \\ |arc\ zy| \ si\ z \neq y \text{ et } z \Gamma y \\ \infty si\ z \neq y \text{ et } (non\ z \Gamma y) \end{cases}$$

On peut écrire le programme suivant qui donne les plus courtes distances entre tous les points d'un graphe donné par sa matrice associée A et un point extrémité E.

Les algorithmes I.1.1 et I.1.2 sont utilisables pour des graphes quelconques.

```
'DEBUT'  EXL(<#DERNIAME_ALGORITHME_DE_BELLMANN-KALABA$>);!MPR;
DEB:       'DEBUT' 'ENTIER' I,J,N,E,W;
             LICLAV(N);
             'DEBUT' 'ENTIER' 'TABLEAU'A.(1:N,1:N).,V.(1:N).;
             'BOOLEEN' 'TABLEAU'C.(1:N).;
             'BOOLEEN' D;
             LIRT(A);EXL(<EXTREMITE>);!MPR;
EXT:         LICLAV(E);'SI' E 'EGAL' O 'ALORS' 'ALLERA' DEB;
             'POUR' I:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
             'DEBUT' 'POUR' J:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
             'SI' A.(I,J). 'EGAL' O 'ALORS' A.(I,J).:=-10000;
             C.(1).:='VRAI';V.(1).:=A.(1,E).'FIN';
             V.(E).:=0;C.(E).:='VRAI';
             D:='FAUX';
ITER:        'POUR' I:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
             'SI' C.(I). 'ALORS' 'DEBUT' C.(I).:='FAUX';W:=0;
             'POUR' J:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
             'SI' A.(I,J).+V.(J).'SUP' W 'ALORS' W:=A.(I,J).+V.(J).:'SI' W 'DIF' V.(I). 'ALORS'
             'DEBUT' V.(I). :=W;C.(I).:='VRAI';D:='VRAI''FIN''FIN';
             'SI' D 'ALORS' 'ALLERA'ITER;
             'POUR' I:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
             'SI' V.(I). 'INF'O 'ALORS' EXE(3,0)'SINON'EXE(3,V.(I).);
             !MPR: 'ALLERA' FXT:'FIN' 'FIN' 'FIN' 'FIN' * #
```


I.2 - Algorithme de la méthode PERT [32].

La méthode est décrite dans son ensemble au chapitre II. Son emploi est limité aux graphes sans circuit.

I.3 Algorithme 6. Pile

Son emploi est limité aux graphes sans circuit. Les seules opérations à effectuer en plus de la gestion de la pile (décrites au chapitre III) sont du type :

$$K[e,y,i] := K[e,y,i-1] \cup [\ell(e,z) \odot K[z,y,i-1]]$$

y étant le point extrémité fixé.

L'algorithme s'écrit alors :

1) $K[x,y,o] := \begin{cases} \ell(x,y) \text{ si } x \Gamma y & \text{pour tout } x \in E - \{y\} \\ +\infty \text{ sinon} \end{cases}$

$K[y,y,o] := 0$

2) Si i est la sortie de z non feuille ($\Gamma(3) \neq \emptyset$)

$K[e,y,i] := \min (K[e,y,i-1], \ell(e,z) + K[z,y,i-1])$.

Les suites $K[e,y,i]$ sont représentées en mémoire par une matrice à n éléments, puisqu'elles contiennent uniquement une distance. Les arcs sont supposés connus par la matrice associée au graphe.

L'encombrement en mémoire de tous ces algorithmes est limité à la matrice associée au graphe et un tableau à n éléments. Seul l'algorithme 6 nécessite un tableau supplémentaire pour la gestion de la pile.

I.4 - Algorithme 7.B :

Plus courtes distances de tout point x d'un graphe à un point y fixé.

Il est applicable aux graphes quelconques.

Il devient :

1) $(\forall x \in E)\ (K[x,y,o] = \infty)$.

2) a) si $\sigma i < \Sigma(z)$

$K[e,z,i] = \min (\ell(e,z), K[e,z,i-1])$.

b) si σi est une sortie de $z > \Sigma(z)$ ou si ($\sigma i = \Sigma(z)$ et z fermeture de circuit) :

pour $v = y$ et pour $v = v_1, v_2, \ldots, v_k$ fermeture de circuit et tels que $\sigma(v_j) < \Sigma(z)$

$K[e,v,i] = \min (K[e,v,i], \ell(e,z) + K[z,v,i-1]$

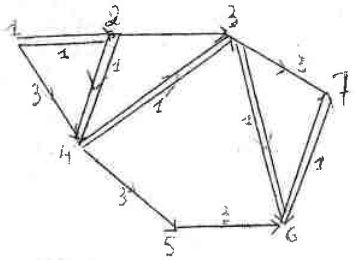
c) Si $\sigma i = \sum(z)$ et z fermeture de circuit :
($\forall x \in E$ et vérifiant $\varepsilon(z) < \sum(x) < \sum(z)$)
(pour les mêmes v)
$(K[x,v,i] = \min(K[x,v,i-1] , K[x,z,i-1] + K[z,v,i-1]))$

On trouvera le programme correspondant en annexe.

II - PLUS COURTES DISTANCES ET PLUS COURTS CHEMINS ENTRE TOUS LES POINTS DU GRAPHE ET UN POINT EXTREMITE

Le principe d'optimalité de Bellmann nous permet d'affirmer que le plus court chemin de x à y, soit $x\, x_1\, x_2 \ldots x_n y$ est obtenu par concaténation de l'arc xx_1 et du plus court chemin de x_1 à y. Il suffit donc pour tout point x de connaître le second point du plus court chemin de x à y pour pouvoir décrire le plus court chemin de x à y.

Exemple :



L'ensemble des plus courts chemins des points de ce graphe au point 7 peut être décrit par le tableau

	1	2	3	4	5	6	7
T	2	4	6	3	6	7	x

Le plus court chemin de x à 7 s'écrit
$x,\ T[x]\ ,\ T[T[x]]\ ,\ T[T]T[x]]]\ ,\ \ldots$ 7

soit 1 1 2 4 3 6 7
2 2 4 3 6 7
3 3 6 7
4 4 3 6 7
5 5 6 7
6 6 7

Ce procédé permet de diminuer considérablement l'espace de stockage en mémoire. Il suffira d'utiliser l'homomorphisme définit au chapitre II

Celui-ci peut être utilisé pour toutes les méthodes citées en I. Les comparaisons et les conclusions de I restent valables, les différences étant simplement plus accusées.

Pour l'algorithme 6 l'opération à effectuer devient :
$K[e,y,i] := \min(K[e,y,i-1] , l(e,z) + K[z,y,i-1])$,
$K'[e,y,i] := z$ si $l(e,z) + K[z,y,i-1] < K[e,y,i-1]$
$K[e,y,i]$ contenant la plus courte distance de e à y à l'instant i et $K'[e,y,i]$ le second point du plus court chemin de e à y.
On réalise évidemment cette opération sous la forme :

SI $L[e,z] + K[z,y] < K[e,y]$ ALORS
DEBUT $K[e,y] := L[e,z] + K[z,y]$; KPRIME $[e,y] := z$ FIN;

Notons que l'on peut aussi pour chaque algorithme cité en I utiliser la méthode décrite pour l'algorithme de FORD. [cf 4] .

"Il existe un sommet x_{p1} tel que $ti - t_{p1} = l(x_i, x_{p1})$ car ti a diminué de façon monotone, et au cours de la procédure, et x_{p1} est le dernier sommet utilisé pour diminuer ti. x_{p1} est le second point du plus court chemin de x_i à x_n.
De même soit x_{p2} tel que $t_{p1} - t_{p2} = l(x_k, x_{p2})$; etc... La séquence ti, t_{p1}, t_{p2},... étant strictement décroissante, on aura à un certain moment $x_{pk+1} = x_n$.
$\mu = [x_i, x_{p1}, \ldots x_n]$ est le plus court chemin de x_i à x_n."

III - PLUS COURTES DISTANCES ENTRE TOUT COUPLE DE POINTS DU GRAPHE

Chacun des algorithmes précédents peut être utilisé successivement pour tout point du graphe comme point extrémité. Il est plus économique de procéder globalement.

III - 1 Algorithme 5 :

Les points du graphe étant ordonnés $z_1, z_2, \ldots, z_n$

- $(\forall x \in E)\ (\forall y \in E)(K(x,y,o) = \begin{cases} si\ x \Gamma y\ alors\ l(x,y)\ sinon\ \infty \\ \quad pour\ x \neq y \\ 0\ si\ x = y \end{cases}$

- (pour $i = 1,2,\ldots n$) $(\forall x \in E$ et $\forall y \in E)$
$K[x,y,i] = \min(K[x,y,i-1] , K[x,y,i-1] + K[z_i,y,i-1])$.

On peut écrire la procédure :

```
PROCEDURE PCD5 (N,A) ; ENTIER N ; ENTIER TABLEAU N ;
   COMMENTAIRE  Cette procédure calcule la matrice des plus courtes
                distances d'un graphe d'ordre n donné par sa ma-
                trice associée A par l'algorithme 5. La matrice des
                distances est également portée par le tableau A ;
DEBUT ENTIER I,J,K ; ENTIER TABLEAU B [1:N] ;
      POUR I:=1 PAS 1 JUSQUA N FAIRE
      POUR J:=1 PAS 1 JUSQUA N FAIRE
      SI A [I,J] = 0 ALORS A [I,J] :=10000 ;
      POUR I:=1 PAS 1 JUSQUA N FAIRE
      POUR J:=1 PAS 1 JUSQUA N FAIRE
          DEBUT B [J] := A [I,J] ;
                SI A [J,I] <10000 ALORS
                POUR K:=1 PAS 1 JUSQUA N FAIRE
                SI A [J,I] + B [K] < A [J,K] ALORS
                A [J,K] := A [J,I] + B [K]
          FIN
FIN ;
```

Cet algorithme demande n^3 additions et n^3 comparaisons si on n'effectue pas les tests A [J,I] <10000, et moins de n^3 avec la rédaction proposée (cf chapitre V).

III.3 ALGORITHME 7. Plus courtes distances entre tout couple de points du graphe :

Il devient :

1) Initialisation

$(\forall x \in E)\ (\forall y \in E)\ (K\ [x,y,0] = \begin{cases} \ell(x,y) \text{ si } x \Gamma y \\ \infty \text{ sinon} \end{cases}$

2) e étant le sommet et r désignant le nombre de sorties de la pile (pour $i=1,2,\ldots,r$)

a) si $\sigma i \sum (z)$

aucune opération avec cette initialisation.

b) si σi est une sortie de $z > \sum(z)$ ou si $\sigma i = \sum(z)$ et z non fermeture de circuit : (cf D.19)

$(\forall y \in E, y \neq z)\ (K\ [e,y,i] = \min \left[K\ [e,y,i-1]\ ,\ \ell(e,z) + K\ [z,y,i-1] \right])$

c) si $\sigma i = \sum(z)$ et z fermeture de circuit

$(\forall x \in E$ et vérifiant $\Sigma(z) < \sum(x) < \sum(z))\ (\forall y \in E)$

$K\ [x,y,i] = \min \left[K\ [x,y,i-1]\ ,\ K\ [x,z,i-1] + K\ [z,y,i-1] \right])$

Pour tout couple x,y non défini ci-dessus

$K\ [x,y,i] = K\ [x,y,i-1]$

Il suffit dans la procédure ALG7PILE et dans PILE MOT SORTIE de définir les opérations OPE2B et OPE2C. (cf page 30)

```
PROCEDURE OPE2B  ; DEBUT ENTIER W ;
        POUR R:=1 PAS 1 JUSQUA N FAIRE
        SI T [R]≥3 ALORS DEBUT W:=A [I,R] + A [J,I] ;
        SI ABS (A [J,R]) > W ALORS A [J,R] := W FIN
        FIN ;

PROCEDURE OPE2C  ; DEBUT ENTIER W ;
        POUR V:=1 PAS 1 JUSQUA N FAIRE
        SI T [V]>3 ALORS DEBUT
        W:= A [V,I] + A [I,R] ;
        SI A [V,R] >W ALORS A [V,R] := W FIN
        FIN ;
```

On peut utiliser toutes les variantes citées au chapitre III quant au choix des ensembles de points pour lesquels on effectue les opérations 2c. Pour ce problème particulier de recherche des plus courtes distances (au sens des longueurs généralisées) il est aussi possible d'utiliser le tableau A support de la matrice associée pour y représenter les ensembles K donc la matrice des distances, ce qui permet un gain de place en mémoire notable.

Les éléments non nuls de la matrice associée (arcs) seront affectés du signe moins pour les distinguer des éléments de la matrice associée. Cette transformation aura aussi l'avantage de supprimer des opérations : il est inutile d'étudier l'influence de l'arc (i,j) dès que l'on a trouvé une distance de I à J inférieure à la longueur de l'arc I,J.

Les courbes 3 (chapitre 5) montrent bien cet avantage. Elles montrent également que cette différence augmente avec le nombre d'arcs du graphe.

III.4 Algorithme 8

L'application de l'algorithme 8 aux problèmes de plus courts chemins nous permet de retrouver l'algorithme publié par [DANTZIG 16 .]. On obtient ainsi un algorithme calculant la matrice des plus courtes distances d'un graphe d'ordre n en n^3 opérations (addition et comparaison). Dans la publication [16], il est utilisé pour calculer les plus courtes distances entre tout point du graphe (au sens strict) avec des longueurs positives ou négatives pour des graphes quelconques ; le problème n'ayant plus de sens dans le cas de chemins non élémentaires, (cf, introduction de ce chapitre) il faudra supprimer les circuits à longueur totale négative qui seront détectés.
Il est possible [cf chapitre IV] d'eviter des opérations en effectuant des tests A $[I,K] \neq \Lambda$ (cf chapitre IV). C'est sous cette forme que l'algorithme sera utilisé. On peut remarquer sur les courbes 3 (chapitre V) l'intérêt de cet algorithme tout au moins tant que les graphes à traiter ne possède pas trop d'arcs.

CHAPITRE VIII

PROBLEMES LIES A LA RECHERCHE DU PREORDRE ASSOCIE A UN GRAPHE

On peut distinguer dans ces problèmes deux groupes selon qu'il s'agit uniquement de chercher le préordre associé au graphe [calcul de la matrice de fermeture transitive, déjà traitée, (cf chapitre 6)] ou qu'on désire utiliser explicitement ce préordre (construction du graphe réduit, recherche des classes d'équivalences ou des classes ergodiques des chaînes de Markov, chemins hamiltoniens, etc...).

Notons que dans la plupart des algorithmes connus traitant les problèmes du second cas (ex/ Roy [49] , construction du graphe réduit, FOULKES [19] recherche des classes d'équivalences et chemins hamiltoniens), on calcule d'abord la matrice de fermeture transitive et on en tire les résultats du problème particulier. Ces méthodes nécessitent donc un nombre d'opérations ("et" et "ou") au moins de l'ordre de n^3. Nous nous proposons d'exposer ici des méthodes utilisant une pile, permettant de résoudre certains de ces problèmes soit directement en ne demandant alors qu'un nombre d'opérations de l'ordre du nombre d'arcs (qui est au plus n^2) soit simultanément au calcul de la matrice de fermeture transitive.

I - RECHERCHE DES CLASSES D'EQUIVALENCE :

I.1 Algorithme de Foulkes [19]

Il consiste à calculer d'abord la matrice de fermeture transitive en utilisant l'algorithme 4 du chapitre II. (ce qui nécessite donc environ $\frac{n^3}{2} \log_2 n$ opérations "et" et "ou"). Puis on utilise la propriété :

si $\hat{\Gamma}(X_i) = \hat{\Gamma}(X_j)$ alors $X_i \in \Gamma(X_j)$ et $X_j \in \Gamma(X_i)$.

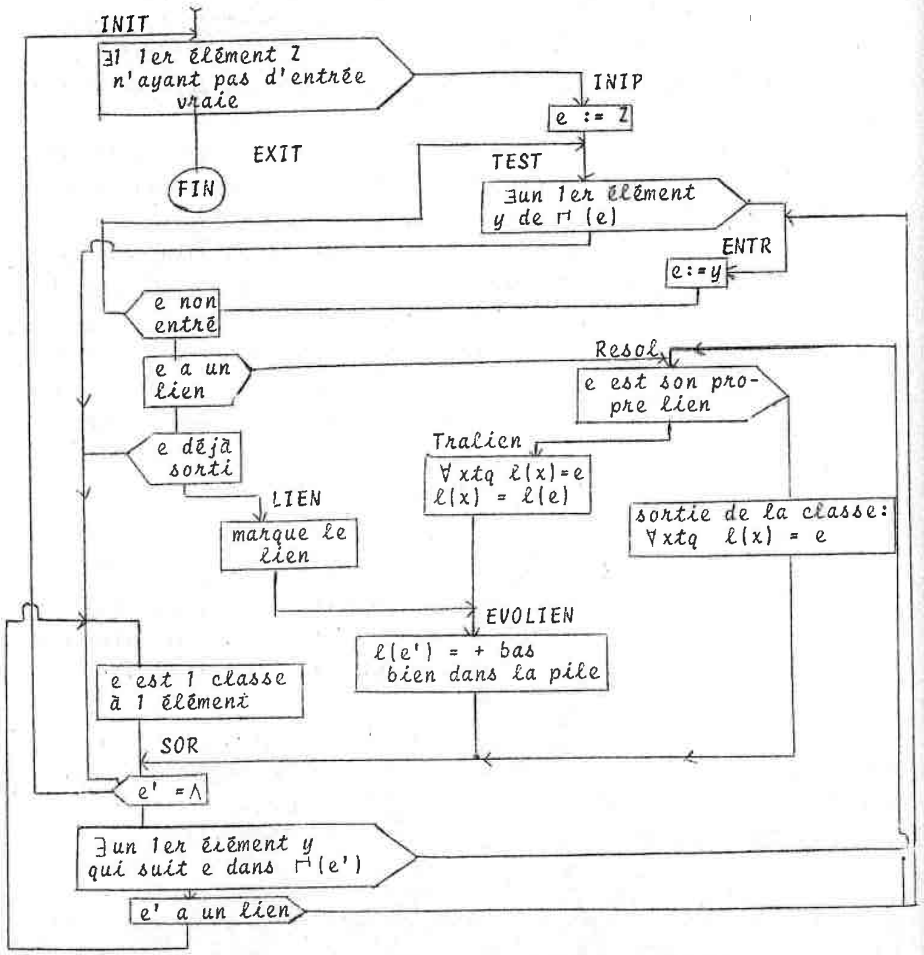
Il suffit donc de grouper les sommets du graphe tels que les lignes et colonnes correspondantes de la matrice de fermeture transitive contiennent des éléments non vides à leur intersection.

I.2 Algorithme proposé :

Il utilise une pile et correspond à l'algorithme _____ utilisant un lien explicite.

A la sortie vraie d'une base de classe e (élément de la classe pour lequel $\ell(x)$ est minimum), sa classe d'équivalence est égale à l'ensemble $\{x, \ell(x) = e\}$ (cf. V§5) ; ceci permet donc de trouver les classes d'équivalences à plus d'un point. Les classes d'équivalence à un seul point peuvent être notées lors des sorties vraies d'éléments n'ayant pas de lien.

On obtient donc l'organigramme suivant :



Recherche des classes d'équivalences

Un programme correspondant à ce problème a été rédigé en Algol CAE 510. Il nécessite m (nombre d'arcs) opérations (gestion de la pile, marquage des sommets) donc moins de n^2 opérations.

II - CONSTRUCTION DU GRAPHE REDUIT

Définition : (Roy [49]). On appelle graphe réduit $G_r = (X_r, \Gamma_r)$ d'un graphe $G = (X, \Gamma)$, un graphe dont les sommets sont les composantes fortement connexes du graphe G et tel que : pour tout $C_i \in X_r$, $C_j \in X_r$ $\quad C_i \ \Gamma_r \ C_j$ si et seulement si $C_i \neq C_j$ et s'il existe $x_m \in C_i$ et $x_n \in C_j$ tels que $x_m \ \Gamma \ x_n$.

On trouvera dans Roy [49], un algorithme de construction du graphe réduit à partir de la matrice de fermeture transitive et utilisant des opérations matricielles.

On y construit la matrice de fermeture transitive, la matrice de décomposition du graphe puis la matrice associée du graphe réduit.

La matrice de décomposition D, est une matrice à n lignes p colonnes (n = nombre de sommets de X, p nombre de sommets de X_r).

$D(i,j)$ = 'VRAI' si i appartient à la classe j

Cet algorithme demande donc au moins un nombre d'opérations de l'ordre de n^3.

Algorithme proposé :

Il dérive directement de l'algorithme de recherche des classes d'équivalence du § I.2 de ce chapitre et permet de construire la matrice de décomposition du graphe réduit et sa matrice associée. Les classes sont numérotées dans l'ordre de sortie des bases de classes de la pile soit $C_1\ C_2 \ldots C_j \ldots C_k$.

Nous préférerons à la matrice de décomposition, un vecteur de décomposition V défini comme suit :

$V(i) = j$ si i appartient à C_j.

On procédera comme suit :

lors de la sortie vraie d'une base e de classe C_j

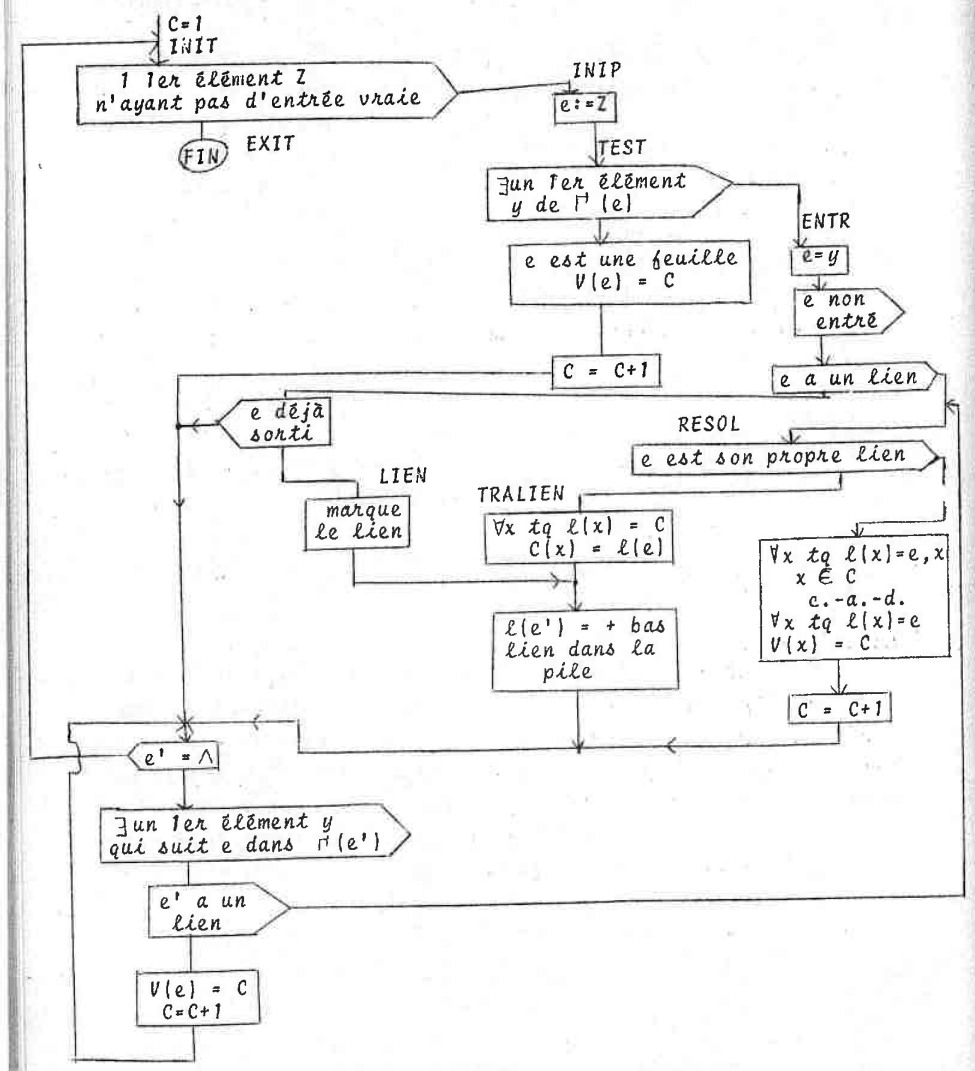
- $\forall_i \in C_j \quad V(i) = j$,

La construction de la matrice M_r associe à (X_r, Γ_r) x fait ensuite

$\forall i \in X, \quad \forall j \in X$

si $i \ \Gamma \ j$ $\quad M_r(V(i), V(j))$ = 'VRAI'.

On peut dessiner l'organigramme de construction du vecteur de décomposition :



CONSTRUCTION DU VECTEUR DE DECOMPOSITION

III - RECHERCHE DES CLASSES ERGODIQUES ET CLASSES TRANSITOIRES DES CHAINES DE MARKOV.

Définitions :

Classe ergodique : Les éléments minimaux de l'ordre partiel des classes d'équivalences sont appelés classes ergodiques

$C_j < C_i$ si $C_i \Gamma_r C_j$

Classe transitoire : Les autres éléments sont appelés classes transitoires.

D'après JOHN. G. KEMENY "FINITE MARKOV CHAINS" p.35 VAN NOSTRAND.

Une classe ergodique est donc une classe d'équivalence telle que la descendance d'un de ses points est limitée à la classe elle-même. Une classe transitoire est telle que la descendance de ses points est plus étendue que la classe elle-même.

Il ne suffit donc pas de chercher les classes d'équivalences mais aussi les relations entre ces classes. Une solution peut donc consister à constuire le graphe réduit (sa matrice associée) les classes égordiques étant les "feuilles du graphe réduit.

Une autre solution peut être adoptée :

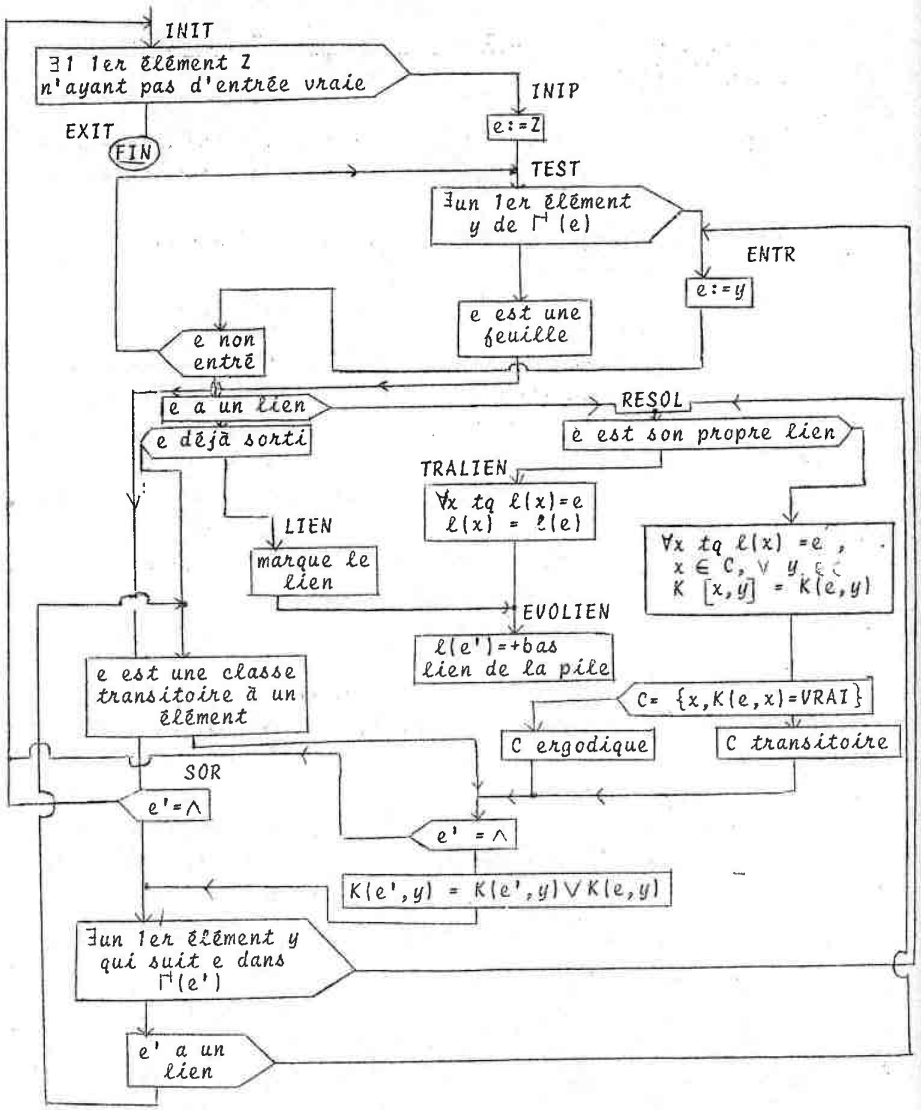
1) Classes réduites à un point

Les classes ergodiques à un seul point sont les "feuilles" les classes transitoires les "noeuds" : on peut donc les détecter dans l'opération "Test".

2) Classes à plus d'un point :

Lors de la sortie de e base de classe à plus d'un point, il suffit de vérifier si l'ensemble des x tels que ℓ(x) = e, est identique à Γ(e) pour savoir si la classe est ergodique.

On peut dessiner l'organigramme suivant.



CHAPITRE IX

PROBLEMES LIES A LA DETECTION DES CIRCUITS

Nous avons vu au chapitre II, §I.3 qu'il est possible de définir des algorithmes construisant tous les chemins et circuits élémentaires du graphe (E, Γ). Ils sont longs et nécessitent une grande place en mémoire. Nous verrons ici des algorithmes plus simples permettant de trouver des circuits vérifiant certaines conditions particulières.

Lemme 4

Si $x_0, x_1, x_2, \ldots x_{\ell-1}, x_\ell$ appartient à $K(x_0, x_{\ell-1}, i)$, la sortie de l'arc correspondant a $x_{\ell-1}\ x_\ell$ est inférieure à i.

Le chemin $x_0, x_1, x_2, \ldots x_{\ell-1}\ x_\ell$ est présent en i par la première forme (cf Définition page 24) ; c'est-dire qu'aucun p... de ce chemin n'est tel que x_i est fermeture de circuit et $\sum(x_i) \leq i$. Donc $\sum(x_{\ell-1}) \leqslant i$. D'après le lemme 2 (page 24) $x_{\ell-1}\,\Gamma x_\ell \Longrightarrow$ la sortie $s(x_\ell)$ correspondant à l'arc $x_{\ell-1}\ x_\ell$ précède la sortie vraie de $x_{\ell-1}$

$$s(x_\ell) < \sum(x_{\ell-1}) \leqslant i \quad \Longrightarrow \quad s(x_\ell) < i.$$

I - THEOREME 2 :

"Pour toute pile annexe à un graphe, tout circuit contient une fermeture de circuit x_ℓ telle que $\sum(x_\ell)$ est supérieure à la sortie de x_ℓ correspondant à l'arc $x_{\ell-1}\ x_\ell$ de ce circuit".

Il suffit de le démontrer pour un circuit élémentaire :

D'après le théorème 1 du chapitre III le circuit élémentaire $x_0, x_1, x_2 \ldots x_n = x_0$ est présent en tout instant $i \geqslant \sum(x_0)-1$, c'est-à-dire qu'il existe un nombre ℓ. $(1 \leqslant \ell \leqslant n)$ tel que :

a) $x_0 \ldots x_\ell \quad K\,[x_0, x_\ell\ ]$

et - (b1) $\ell = n$ ou (b2) x_ℓ est fermeture de circuit et $\sum(x_\ell) > i$.

S'il est présent en $\sum(x_0)-1$ par (b2), le circuit envisagé contient une fermeture de circuit et la sortie de x_ℓ correspondant à l'arc $x_{\ell-1}\ x_\ell$ $\overline{\sigma x_{\ell-1}}\ x_\ell$, est telle que $\overline{\sigma x_{\ell-1}}, x_\ell \leq i < \sum(x_\ell)$, d'après le lemme 4. En effet on a bien $x_0 \ldots x_{\ell-1}\ x_\ell \in K\,[x_0, x_\ell, i]$.

S'il est présent en $\sum(x_0)-1$ par (b1) $\ell = n$ et

$$x_0, x_1 \ldots x_{n-1}, x_0 \in K\,[x_0, x_0, i] \quad \forall i \geqslant \sum(x_0)-1.$$

D'après le lemme 4 la sortie de x_0 correspondant à l'arc $x_{n-1}\ x_0$ est inférieure ou égale à $\sum(x_0)-1$.
Donc x_0 est une fermeture de circuit convenable

II - DETECTION DES CIRCUITS A TROIS POINTS :

C'est une question rencontrée en particulier dans l'étude des système de préférence (étude de marché) :
Considérons un graphe complet (D.8), antisymétrique (D.9), représentant un système de préférence. Toute "intransitivité" dans le système se traduit par la présence d'un circuit à trois sommets. Ceux-ci peuvent donc caractériser une certaine indifférence du marché qu'il est intéressant de déceler.

Toutefois s'il s'agit simplement de dénombrer ces circuits à 3 sommets et de calculer le coefficient d'inconstance du marché, il suffira d'appliquer la relation donnée par C. Berge [4] , théorème 3 page 126.

Dans un graphe complet antisymétrique le nombre de circuits de 3 sommets a pour valeur

$$\zeta = \frac{1}{12}\, n\,(n-1)\,(2n-1) - \frac{1}{2}\sum_{i=1}^{n} (r_i)^2$$

où r_i représente le 1/2 degré extérieur du sommet x_i, le nombre maximum de ces circuits étant

$$\begin{cases} \dfrac{n^3-n}{24} & \text{si } n \text{ est impair} \\ \dfrac{n^3-4n}{24} & \text{si } n \text{ est pair} \end{cases}$$

II -1 ALGORITHME PROPOSE

S'il s'agit de trouver explicitement ces circuits, on pourra choisir la méthode suivante utilisant une pile annexe au graphe (cf D.17). Notons qu'elle n'est pas limitée aux graphes complets antisymétriques et qu'elle peut être utilisée pour des graphes quelconques.

Si i est une sortie de z inférieure à sa sortie vraie (z est fermeture de circuit) :

soit y le sommet de la pile, y est le second point d'un circuit de premier point z. Tout point commun, s'il en existe, à la liste des successeurs de z et celle des prédecesseurs de y est le troisième point d'un circuit à trois points z x y z.

Justification :

1) Soit un circuit à 3 points x y z : il contient une fermeture de circuit (théorème 2) soit z.
La sortie de z correspondant à l'arc y z (sortie de l'arc yz de la pile attachée) est inférieure à $\sum(z)$. Il existe donc une sortie de z, soit i, telle que $i < \sum(z)$ et μ_i a pour sommet y.
De plus $x \in \Gamma(z) \cap \Gamma^{-1}(y)$.

2 - RECIPROQUEMENT :

$\mu_i = \alpha y$ et i sortie de z $\Longrightarrow$ $\exists$ arc y z
$x \in \Gamma(z) \cap \Gamma^{-1}(y) \Longrightarrow$ $\exists$ arc z x et arc x y.

Il suffit donc dans l'organigramme de la recherche des chemins de tout point x à y fixé (chapitre III §1) de remplacer le branchement à ERREUR, qui caractérise une sortie de J inférieure à sa sortie vraie par le schéma de ces opérations. On obtient l'organigramme suivant : (voir page suivante). On trouvera en annexe le programme correspondant

II -2 ALGORITHME DE ROY :

Soit $S_i = (E_i, \Gamma)$ un sous-graphe de (E, Γ) tel que
si $E = \{x_1, x_2, \ldots x_n\}$, $E_i = \{x_{i+1}, \ldots x_n\}$

L'algorithme peut s'écrire :

pour $i = 1, 2, \ldots, n-3$
$(\forall k \in E_i$ et tel que $k \Gamma_i)$ $(\forall j \in E_i$ et tel que $j \Gamma k)$
(si $i \Gamma j$, $i\ j\ k\ i$ est un circuit à trois points).

La justification en est évidente puisqu'on n'y fait que vérifier que le triplet i, j, k vérifie bien la définition d'un circuit à trois points :

$$i \Gamma j \Gamma k \Gamma i$$

Cette rédaction est équivalente à la rédaction proposée par ROY [54] .

Algorithme II -4 :

α) Supprimer dans le graphe tout sommet ayant un degré intérieur ou extérieur nul, puis préparer un tableau à 3 colonnes et une ligne de moins que de sommets restants. Soit x_i le sommet auquel ne correspond pas de ligne.

β) Inscrire I en première colonne dans toute ligne associée à un successeur de x_i et marquer une étoile en dernière colonne dans toute ligne associée à un prédecesseur de x_i.
Marquer en colonne centrale, sur chaque ligne possèdant une étoile, l'indice de tous les prédecesseurs du sommet associé. Rayer enfin de la colonne centrale tout nombre correspondant à une ligne ne possèdant pas de I en colonne 1. Tout triplet i,j,k tel que j demeure en colonne centrale à la ligne k.

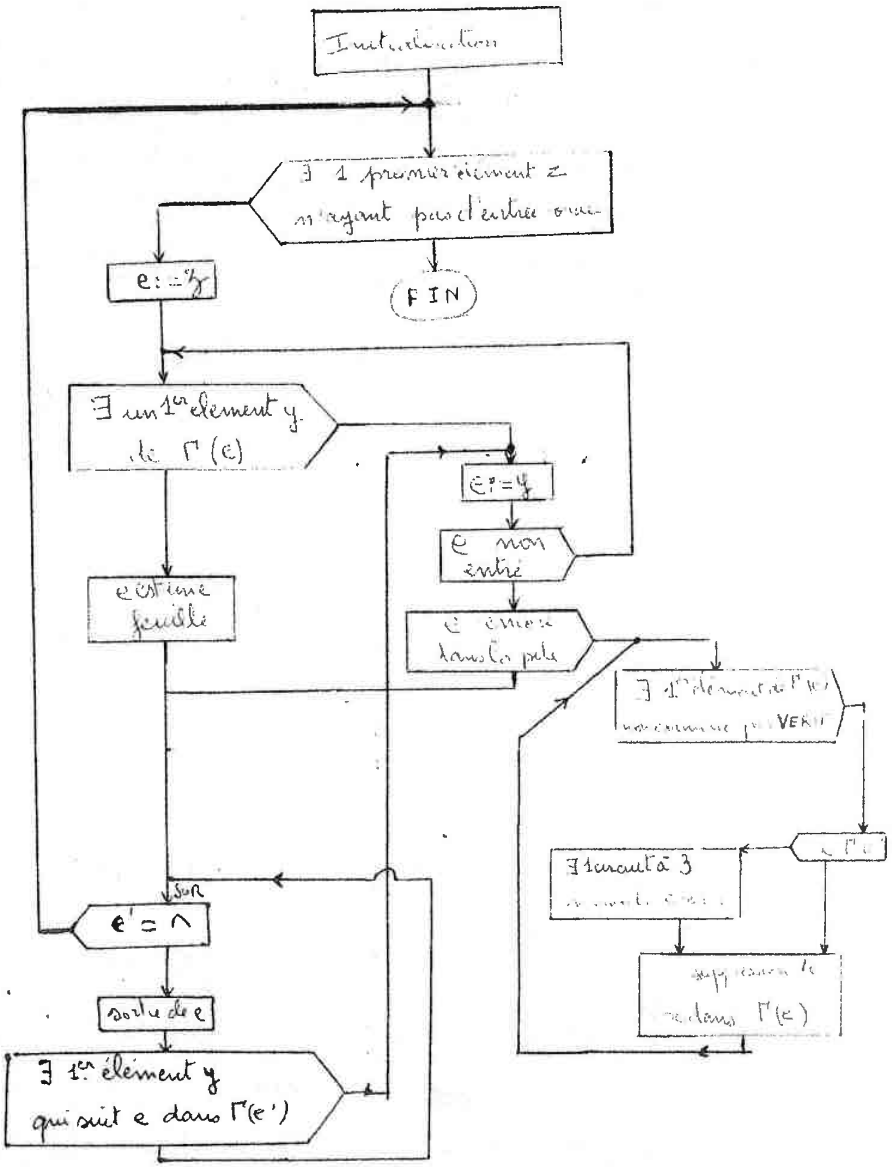
γ) Supprimer le sommet x_i et appliquer les procédures α et β on peut alors écrire la procédure suivante :

```
PROCEDURE CIR3ROV(A,N) ; ENTIER N ; ENTIER TABLEAU A ;
    COMMENTAIRE Recherche des circuits à trois points dans le
                graphe d'ordre N et de matrice associée A,
                par la méthode de ROY ;
DEBUT ENTIER I,J,K ;
      POUR I:=1 PAS 1 JUSQUA N-3 FAIRE
      POUR K:=I PAS 1 JUSQUA N FAIRE
      SI A [K,I] ≠ 0 ALORS
      POUR J:=I PAS 1 JUSQUA N FAIRE
      SI A [J,K] ≠ 0 ALORS
          DEBUT SI A(I,J) ≠ 0 ALORS IMPRIMER (I,J,K)
          FIN
FIN ;
```

II -3 Comparaisons

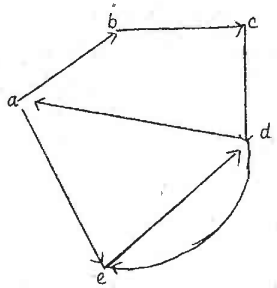
Nous comparerons d'abord la méthode sur des graphes sans circuits à 3 points, puis sur des graphes en contenant.
Notons que les listes obtenues par l'algorithme III-2 sont sans répétitions alors que pour l'algorithme III-1 elles peuvent en présenter.

Recherche des circuits à 3 points.



Tout circuit possèdant une fermeture de circuit, on peut affirmer qu'on décèlera tous les circuits à trois points par cette méthode. Cependant la liste ainsi obtenue peut présenter des répétitions lorsque deux points d'un même circuit à trois points sont des fermetures de circuit.

exemple :



états de la pile annexe :

^
a
a b
a b c
a b c d
a b c d a
a b c d
a b c d e
a b c d e d
a b c d e
a b c d
a b c
a b
a
^

} on trouvera le circuit a e d

} on trouve d a e qui est le même circuit que a e d a

III - DETECTION DES CIRCUITS PARASITES :

L'homomorphisme défini au chapitre II appliqué aux algorithmes de recherche de chemins et circuits entre tous les points du graphe pourrait nous fournir une liste exhaustive des circuits élémentaires du graphe. Ici le problème est plus simple et cette méthode serait trop onéreuse (il faut stocker en mémoire tous les chemins du graphe et le nombre d'opérations "produit de suites de chemins" est l'ordre de n^3).
Ici, il s'agit, en effet de vérifier qu'un graphe supposé sans circuit l'est effectivement, et, le cas échéant de permettre la suppression des circuits parasites trouvés.

III-1 Méthode dérivée de celle de Rosalind Marimont [39] :

a) Théorème 3

"Un graphe fini (E, Γ) possède un circuit si, et seulement si, il existe un sous-graphe de (E, Γ) sans point de sortie. (cf D. 6)".

1) Soit un circuit $c=x_0, x_1, x_2, \ldots x_n = x_0$ du graphe E, Γ et soit (E', Γ) sous-graphe de E, Γ formé uniquement des points du circuit c. (E', Γ) ne possède aucun point de sortie car pour

tout $x_i \in E'$ $\left| \begin{array}{ll} x_i \Gamma x_{i+1} & si\ i \neq n \\ x_n \Gamma x_1 & si\ i = n \end{array} \right.$

2) Soit (Ep, Γ) un sous-graphe de (E, Γ) sans point de sortie

$Ep = \{x_1, x_2, \ldots x_p\}$

$\forall x_i$, $x_j \in Ep$ tel que $x_i \Gamma x_j$

Si $x_{i_1}, x_{i_2}, x_{i_3}, \ldots, x_{i_q}$ est un chemin de Ep

$x_{i_1}, x_{i_2}, \ldots x_{i_{q+1}}$ est aussi un chemin .

On peut donc construire des chemins arbitrairement longs dès que (ou avant) $q > p$ le chemin n'est plus élémentaire. Il contient donc un circuit.

Corollaire 1 :

"Un graphe fini (E, Γ) possède un circuit, si, et seulement si, il existe un sous-graphe de (E, Γ) sans point d'entrée". (cf D. 6)

On le déduit immédiatement du théorème 3 en substituant Γ^{-1} à Γ

Corollaire 2 :

"Un graphe (E, Γ) possède un circuit, si et seulement si, il existe un sous-graphe de (E, Γ) sans point d'entrée, ni point de sortie.

b) Méthode de Marimont

Elle utilise le corollaire 2. Il suffit donc de :

- chercher les éléments sans successeurs (ligne nulle de la matrice associée).
- chercher les éléments sans prédécesseurs (colonne nulle de la matrice associée)
- supprimer les lignes et colonnes des éléments retenus.

Si l'algorithme est applicable jusqu'à ce qu'il ne reste qu'un seul point il n'y a pas de circuit.

S'il n'est plus possible de supprimer des éléments de la matrice associée, il existe au moins un circuit dont les points figurent parmi les sommets restants. L'ensemble des sommets restants est la réunion des classes d'équivalence à plus d'un point du graphe.

Remarques

α) Cette méthode permet en fait de trouver tous les points appartenant à des classes d'équivalences à plusieurs éléments

mais ne donne aucun renseignement sur la configuration de ces classes, et il est assez difficile d'éliminer des circuits dans le cas où les points restants ne constituent pas un seul circuit.

β) Cette rédaction de la méthode implique une représentation du graphe par sa matrice associée, ou par des listes des successeurs des sommets et de leurs prédecesseurs. Dans le cas de gros graphes (utilisation de mémoire périphérique) on est alors obligé (1er cas) de manipuler ces données de façon séquentielle pour les lignes mais non séquentielles pour les colonnes ou (2ème cas) d'utiliser un trop gros volume d'information.

La modification introduite par l'utilisation du théorème 3 au lieu du corollaire 2 entraine une modification du nombre des opérations à effectuer qui dépend surtout de la configuration du graphe, mais il devient possible de n'utiliser les éléments de la matrice associée que ligne par ligne, partant, de permettre un traitement séquentiel pour les gros graphes, ou encore de ne représenter le graphe que par des listes de successeurs de chaque point

γ) La méthode proposée par B. ROY [54] (p. 68 algorithme VI 1), est la méthode de Rosalind Marimont appliquée au cas où le graphe est donné sous la forme d'une liste des successeurs de chaque point.

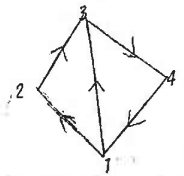
III-2 Méthode proposée :

On utilise la pile annexe au graphe (D.17) et s'appuie sur le théorème 3. Tout circuit possédant une fermeture de circuit x_ℓ (donc la sortie correspondant à l'arc du circuit $x_{\ell-1}\ x_\ell$ est inférieure à $\sum(x_\ell)$), on peut affirmer que tout circuit peut être détecté en cherchant les sorties de points x inférieurs à leur sortie vraie.

La suite des sommets du graphe ayant une occurrence dans la pile, comprise entre les deux occurrences de e constitue un circuit qu'il suffit de noter. En effet tout état de la pile représentant un chemin (cf D.17) l'état de la pile lors de cette sortie de $< \sum(e)$ s'écrit .... e $x_1\ x_2\ x_3 \ldots x_{n_e}$.

Cependant, il est impossible d'affirmer que tout nouveau circuit élémentaire entraine la présence d'une nouvelle fermeture de circuit car ; fermeture de circuit peut appartenir à deux circuits différents.

Exemple :




```
          1   <------ fermeture de circuit
        4 4 4
      3 3 3 3 3
    2 2 2 2 2 2 2 2     3
∧ 1 1 1 1 1 1 1 1 1 1 1 1∧
```

Il est alors impossible de détecter le circuit 1341. Cependant la suppression de l'arc (41) qui a provoqué une sortie inférieure à une sortie vraie permet de supprimer les deux circuits.
La suppression des arcs pointant vers des fermetures de circuits et entrainant une sortie inférieure à leur sortie vraie permet de supprimer tous les circuits. (Théorème 2).

III-3 Réalisations

Trois programmes ont été réalisés

1) Méthode utilisant le théorème 3.
2) Méthode de R. Marimont (corollaire 2)
3) Méthode de la pile annexe

Pour la méthode III-2. On peut écrire la procédure suivante :

```
PROCEDURE CIRCUIT (N,A) ; ENTIER N ; ENTIER TABLEAU
    COMMENTAIRE  Cette procédure recherche les circuits d'un graphe
                 d'ordre N représenté par sa matrice associée A ;
    DEBUT  ENTIER K,I,J,R ; ENTIER TABLEAU T [1:N] , P [1:N+1] ;
           POUR I:=1 PAS 1 JUSQUA N FAIRE DEBUT
           T [I] :=1 ; A [I,I] :=0 FIN ; K:=1 ;
    INIT : POUR I:=1 PAS 1 JUSQUA N FAIRE
           SI T [I] =1 ALORS ALLERA INIP ; ALLERA EXIT ;
    INIP : P [I] :=I ;
    TEST : POUR J:=1 PAS 1 JUSQUA N FAIRE
           SI A [I,J] ≠ 0 ALORS DEBUT
           T [I] :=2 ; ALLERA ENTR FIN ; T [I] :=3 ;
```



```
    SOR :  SI K=1 ALORS ALLERA INIT ;
           K:=K-1 ; R:=I+1 ; I:=P [K] ;
           POUR J:=R PAS 1 JUSQUA N FAIRE
           SI A [I,J] ≠ 0 ALORS ALLERA ENTR ;
           T [I] :=3 ; ALLERA SOR ;
    ENTR : K:=K+1 ; P [K] :=I:=J ; R:= T [I] ;
           ALLERA SI R=1 ALORS TEST SINON
                  SI R=2 ALORS ERREUR SINON SOR ;
    ERREUR : IMPRIMER (CIRCUIT,I) ;
             POUR R:=K-1 PAS -1 JUSQUA 1 FAIRE
             DEBUT IMPRIMER (P [R] ) ; SI P [I] =I ALORS
                   ALLERA SOR FIN ;
    EXIT :   IMPRIMER (PLUS DE CIRCUIT)
FIN ;
```

Comparaisons :

Le nombre d'opérations à effectuer dans l'algorithme de Marimont est proportionnel à n^2. De plus elles sont assez longues à réaliser (suppression de sommets). Dans l'algorithme de la pile ce nombre est égal à m nombre d'arcs qui est souvent, dans le cas des graphes sans circuits, inférieur à $\frac{n(n-1)}{2}$

IV - DETECTION DE CIRCUITS A 2 POINTS

Le fonctionnement de la pile annexe à un graphe nous entraine à effectuer lors de la sortie vraie d'une fermeture de circuit des opérations (2c) pour un ensemble de points x qui peut être, dans un certain cas, soit beaucoup plus vaste que l'ensemble des éléments à traiter, soit délicat à repérer. Il peut donc sembler avantageux, lors de la sortie vraie de la fermeture d'un circuit à deux points de n'effectuer cette opération 2c que pour le deuxième point. Il ne s'agit pas ici de chercher tous les circuits à deux points problème plus simple pour lequel il suffit de chercher les couples tels que A(i,j) = A(j,i).

Proposition 1 : (x,y) est une classe à 2 points distincts et $\mathcal{E}(x) < \mathcal{E}(y) \Rightarrow \exists i$ tel que $\mu_i = \alpha x\ y\ x$

D'après le théorème 2 le circuit x y n contient une fermeture de circuit qui est nécessairement x.

donc $\exists i$ tel que $u_i = \alpha x \beta x$.

Si β contient d'autres points que y la classe possède plus de deux points.

Proposition 2 : (x y) est une classe à deux points distincts et $\Gamma(x) < \Gamma(y) \Longrightarrow \exists i$ tel que $u_i = \alpha x\ y\ n$ et n n'a qu'une entrée entre $\Gamma(x)$ et $\Gamma(x)$.

En effet s'il y a une autre entrée cela signifie qu'il existe un point z tel que z x est un arc d'un circuit de fermeture x : il ne s'agirait donc plus d'une classe à deux points.
Les réciproques sont fausses : contre-exemple : et les états de de la pile correspondants.

x
y y y y
x x x x x x x
z z z z z z z z z

Il existe bien un état x y x mais il s'agit d'une classe à trois points.

La recherche des états de la pile du type $\alpha n\ y\ n$ permet donc de trouver toutes les classes d'équivalence à 2 points distincts, mais aussi des couples de 2 points qui n'en sont pas (ce sont quand même des circuits à deux points).
Il suffira donc à la première entrée de x supérieure à $\Gamma(x)$ et inférieure à $\Sigma(x)$ de chercher si on a une telle configuration.
Pour toute autre entrée de x supérieure à $\Gamma(x)$ et inférieure à $\Sigma(x)$ il faudra supprimer le couple éventuellement trouvé .
Il est possible alors de simplifier le traitement de graphes non orientés à l'aide d'une pile (cf chapitre 10).
Notons cependant que cet algorithme est peu performant.

CHAPITRE X

APPLICATION AUX GRAPHES NON ORIENTES

I - GENERALITES

On pourra les considérer comme des graphes orientés symétriques, c'est-à-dire des graphes (E, Γ) tels que :

$$x \Gamma y \iff y \Gamma x.$$

On en déduit immédiatement que si, dans un tel graphe, il existe un chemin $x_0, x_1, x_2, \ldots, x_n$ de x_0 à x_n, $x_n, x_{n-1}, \ldots x_2, x_1, x_0$ est un chemin du graphe de x_n à x_0, c'est-à-dire que pour tous les algorithmes envisagés on doit avoir, à l'instant final q :

$$(\forall x \in E)\ (K[x,y,q] = \widetilde{K}[y,x,q]) \quad (1)$$

$\widetilde{K}[x,y,q]$ étant l'ensemble des réflechis des chemins de $K[x,y,q]$.
On peut remarquer que pour la plupart des algorithmes envisagés, on a aussi :

$$(\forall i)\ (\forall x \in E)\ (\forall y \in E)\ (K[x,y,i] = \widetilde{K}[y,x,i]). \quad (2).$$

Alors on peut se limiter, dans le cas des graphes non orientés, à ne calculer les $K(x,y,i)$ que pour un seul des couples x y et y x, ce qui nous permettra de diminuer de moitié le nombre d'opérations à effectuer, ainsi que la place de stockage en mémoire.
Convenons que l'on écrira $x < y$ si dans l'ordre de dénomination des sommets x précède y.

Un choix particulièrement entre les couples x y et y x est celui qui consiste à prendre

(x,y) si $x \leq y$
$(y\ x)$ sinon

car il permet de retrouver facilement les couples choisis. Pratiquement on ne conserve que la moitié supérieure droite de la matrice associée (on obtient alors un graphe orienté sans circuit) et de la matrice contenant les $K(x,y,i)$.
Détaillons cette transformation pour l'algorithme 1 par exemple.
Il s'écrirait :

a) $(\forall x \in E)\ (\forall y \in E)\ (K[x,y,o] = I[x,y])$
b) $(\forall x \in E)\ (\forall y \in E)$
$(K[x,y,i] = \bigcup_{z \in E} K[x,z,i-1] \odot J[z,y])$.

On peut réécrire le calcul de $K\,[x,y,i]$.

$$K\,[x,y,i] = \Big(\bigcup_{\substack{z\in E\\ z\geqslant x}} K\,[x,z,i-1] \odot J\,[z,y]\Big) \cup \Big(\bigcup_{\substack{z\in E\\ z< x}} K\,[x,z,i-1] \odot J\,[z,y]\Big)$$

soit en tenant compte de (2) :

$$K\,[x,y,i] = \Big(\bigcup_{\substack{z\in E\\ z\geqslant x}} K\,[x,z,i-1] \odot J\,[z,y]\Big) \cup \Big(\bigcup_{\substack{z\in E\\ z< x}} \widetilde{K}\,[z,x,i-1] \odot J\,[z,y]\Big)$$

La relation (2) nous permettrait d'en déduire $K\,[y,x,i]$,mais en fait aucun $K\,[x,y,i]$ tel que $x > y$ n'intervenant dans le calcul il est inutile de les calculer.

L'algorithme 1 pour les graphes non orientés s'écrit donc :

a) $(\forall x\in E)\ (\forall y\in E,\ y\geqslant x)\ (K\,[x,y,o] = I\,[x,y]\,)$.

b) $(\forall x\in E)\ (\forall y\in E \text{ et } y\geqslant x)$

$$(K\,[x,y,i] = \Big(\bigcup_{\substack{z\in E\\ z\geqslant x}} K\,[x,z,i-1] \odot J\,[z,y]\Big) \cup \Big(\bigcup_{\substack{z\in E\\ z< x}} \widetilde{K}\,[z,x,i-1] \odot \widetilde{J}\,[y,z]\Big)$$

On constate que le principe de l'algorithme n'est pas modifié mais uniquement sa forme. Le nombre de couples pour lesquels il faut effectuer ces calculs ainsi que le nombre d'emplacements en mémoire (un emplacement pour un $K\,[x,y,.]$) sont presque diminués de moitié : ils deviennent $\frac{n^2}{2}+n$ au lieu de n^2.

Si la diminution de la place nécessaire en mémoire n'est guère intéressante pour les problèmes de plus courts chemins [longueur généralisée cf. chapitre VII, pour lesquels on utilise une représentation matricielle elle est par contre, très importante pour les problèmes de recherches de chemins et circuits élémentaires. Les algorithmes pouvant être simplifiés par cette méthode sont ceux donnant des suites de chemins entre tout couple de points du graphes et n'utilisant pas de pile donc les algorithmes

1. 5,8

Nous étudierons en particulier les transformés des algorithmes 5 et 8 car ce sont les plus efficaces parmi ceux qui recherchent tous les chemins entre tous couples de points du graphe sans utiliser de pile.

Notons que tous les algorithmes qui en sont déduits par hormomorphisme sont susceptibles de la même simplification.

II - TRANSFORME DE L'ALGORITHME 5 :

Il vient :

1) $(\forall x\in E)\ (\forall y\in E \text{ et } y\geqslant x)\ (K\,[x,y,o] = I\,[x,y]\,)$,

2) $(\forall x\in E)\ (\forall y\in E \text{ et } y\geqslant x)$ (pour $i=1,2\ldots n$)

$$K\,[x,y,i] = \begin{cases} \text{a) si } z_i < x \leq y \\ \quad K\,[x,y,i-1] \cup \big(\widetilde{K}\,[z_i,x,i-1] \odot K\,[z_i,y,i-1]\big) \\ \text{b) si } x < z_i \leq y \\ \quad K\,[x,y,i-1] \cup \big(K\,[x,z_i,i-1] \odot K\,[z_i,y,i-1]\big) \\ \text{c) si } x \leq y < z_i \\ \quad K\,[x,y,i-1] \cup \big(K\,[x,z_i,i-1] \odot \widetilde{K}\,[y,z_i,i-1]\big) \end{cases}$$

L'écriture formelle est certes plus complexe mais le programme ne le devient guère plus que dans la première forme de l'algorithme 5 car, en fait, il suffit de reconstituer à chaque itération la ligne z_i de la matrice contenant les $K\,[x,y,.]$.

On obtient alors la procédure suivante pour le problème de la plus courte distance :

```
PROCEDURE PCD5SYM(A,N) ; ENTIER TABLEAU A ; ENTIER N ;
        COMMENTAIRE La procédure PCD5SYM calcule la matrice
                    des plus courtes distances dans un graphe
                    non orienté donné par sa matrice associée A.
                    N est l'ordre du graphe l'élément infini
                    est représenté par 10 000 ;
DEBUT ENTIER I,J,K,W ; ENTIER TABLEAU B [1:N] ;
      POUR I:=1 PAS 1 JUSQUA N FAIRE
      DEBUT POUR J:=1 PAS 1 JUSQUA N FAIRE
        SI A [I,J] =0 ALORS A [I,J] :=10000 ; A [I,I] :=0
      FIN ;
      POUR K:=1 PAS 1 JUSQUA N FAIRE
      DEBUT POUR I:=1 PAS 1 JUSQUA K FAIRE
            B [I] := A [I,K] ;
            POUR I:=K+1 PAS 1 JUSQUA N FAIRE
            B [I] := A [K,I] ;
            POUR I:=1 PAS 1 JUSQUA N FAIRE
            POUR J:=1+1PAS 1 JUSQUA N FAIRE
```

```
            DEBUT W:=B [I] + B [J] ;
                  SI W<A [I,J] ALORS  A [I,J] :=W
            FIN
      FIN
FIN ;
```

Notons qu'il est possible de prendre la partie non utilisée de la matrice associée pour y faire figurer un i,j le second point du plus court chemin de j à i [cf chap. VII]

III - TRANSFORME DE L'ALGORITHME 8 :

Il devient

1) $(\forall x \in E)\ (\forall y \in E,\ y \geqslant x)\ (K[x,y,o] = I[x,y])$

2) $k = 2,3,\ \ldots\ n.$

a) $(\forall j \in S_{k-1})$ (pour $\ell = 1,2,\ldots\ j-1$)
$K[\ell,k]_j = K[\ell,k]_{j-1} \cup (K[\ell,j] \odot (j,k))$

b) $(\forall j \in S_{k-1})$ (pour $\ell = J, J+1, \ldots\ K-1$)
$(K[\ell,k]_j = K[\ell,k]_{j-1} \cup \{\check{K}[j,\ell] \odot (k,j)\})$

c) $(\forall i \in S_{k-1}$ et tel que $K[i,k] \neq \wedge)\ (\forall j \in S_{k-1},\ j > i)$
$(K[i,j] = K[i,j] \cup (K[i,k] \odot K[k,j]))$

L'opération c) n'est pas modifiée mais on l'effectue pour moitié moins de couples i,j.

Pour le problème du calcul de la matrice des plus courtes distances on peut écrire la procédure PCD8SYM.

```
PROCEDURE  PCD8SYM(A,N) ; ENTIER  TABLEAU A ; ENTIER N ;
   COMMENTAIRE Cette procédure calcule la matrice des plus courtes
               distances d'un graphe non orienté donné par sa
               matrice associée A. N est l'ordre du graphe. L'élé-
               ment infini est représenté par 10000 ;
DEBUT ENTIER I,J,K,L ; ENTIER  TABLEAU C [1:N] ;
      POUR I:=1 PAS 1 JUSQUA N FAIRE
      DEBUT POUR J:=1 PAS 1 JUSQUA N FAIRE
            SI A [I,J] =0 ALORS A [I,J] :=10000 ; A [I,J] :=0
      FIN ;
```



```
      POUR K:=1 PAS 1 JUSQUA N FAIRE
      DEBUT POUR I:=1 PAS 1 JUSQUA K-1 FAIRE
            C [I] := A [I,K];
            POUR J:=1 PAS 1 JUSQUA K-1 FAIRE
            SI A [J,K]<10000 ALORS
            DEBUT POUR L:=1 PAS 1 JUSQUA J-1 FAIRE
                  SI A [J,K] + A [L,J] < C [L] ALORS
                              C [L] := A [J,K] + A [L,J] ;
                  POUR L:=J PAS 1 JUSQUA K-1 FAIRE
                  SI A [J,K] + A [J,L] < C [L] ALORS
                              C [L] := A [J,K] + A [J,L] ;
            FIN de l'opération b ;
            POUR I:=1 PAS 1 JUSQUA K-1 FAIRE
            DEBUT SI C [I] < 10000 ALORS
                  POUR J:=1 PAS 1 JUSQUA I-1 FAIRE
                  SI C [I] + C [J] < A [J,I] ALORS
                              A [J,I] := C [I] + C [J] ;
                  A [I,K] := C [I]
            FIN
      FIN
FIN ;
```

Le tableau C est utilisé comme zone de travail, C [I] contient successivement à kième itération les différentes valeurs de la plus courte distance connue à l'instant considéré entre les points I et K alors qu'en A [I,K] on conserve la longueur de l'arc (i,k).

IV - ALGORITHME 7 - PILE :

La relation (2) $(\forall i)\ (K[x,y,i] = \check{K}[y,x,i])$ n'est plus vérifiée : en particulier lors de l'opération 2b, e étant le sommet, on modifie K(e,y,.) pour tout y et non les K(y,e,.). La méthode précédente n'est donc plus applicable : si on remplace le graphe non orienté par deux graphes orientés sans circuit on pourra bien trouver tous les chemins du graphe initial composés d'arcs ayant la même orientation ((x y) et (z v) ont la même orientation si $x \leqslant y$ et $z \leqslant v$ ou $y \leqslant x$ et $v \leqslant z$), mais il sera délicat d'en déduire tous les chemins ou circuits élémentaires.

L'utilisation de l'algorithme 7 pour les graphes symétriques fait détecter la présence de nombreux premiers de circuits, d'où de nombreuses opérations c. Cependant dans la plupart des cas il s'agit de premiers d'un circuit à deux points uniquement. Dans ce cas l'ensemble des points x pour lesquels il faut effectuer l'opération c lors de la sortie vraie d'un tel premier de circuit peut être limité à ce deuxième point.
Le problème est donc de repérer et de conserver en mémoire le nom de ce deuxième point, tout au moins jusqu'à la sortie vraie du premier de circuit.

Envisageons un tel circuit $x\ y\ x$

et supposons que $\varepsilon(x) \prec \varepsilon(y)$.

Soit μ l'état de la pile à l'entrée vraie de x.

$$\mu = \mu\ \varepsilon(x)\text{-}1 \cdot x.$$

Il existe une entrée de $x > \varepsilon(x)$ telle que l'état correspondant de la pile soit

$$\mu.\ x.\ y.\ x.$$

Pour détecter ces circuits il suffira donc de vérifier lors d'une entrée de x supérieure à son entrée vraie, si l'élément sous le sommet est x lui-même.

Ceci sera repris au chapitre IX, et on trouvera le programme correspondant en annexe.

V - REALISATION

Les 3 méthodes ont été comparées sur le problème de la recherche de la plus courte distance entre tous couples de points d'un graphe symétrique.
Tableau de résultat :

CHAPITRE XI

APPLICATION A L'ETUDE DES RESEAUX PERT
PROBLEME CENTRAL D'ORDONNANCEMENT.

La méthode PERT (Program Evaluation Research Task) est un ensemble très complet réalisant un ordonnancement optimal dans un programme de travaux selon un certain nombre de critères : durée minimale, coût minimal, probabilité maximale...

Nous nous intéresserons ici uniquement au problème central d'ordonnancement : la recherche du chemin critique, pour donner un exemple d'utilisation pratique d'un algorithme de recherche de plus long chemin entre les points d'un graphe. Rappelons d'abord les éléments essentiels de la méthode PERT et l'algorithme de recherche du chemin critique utilisé.

I - METHODE PERT [Kaufmann et Desbazeilles, 32]

I-1 Généralités

Un programme" est un ensemble "d'opérations" concourant à la réalisation d'un objectif telles qu'on connaisse, pour chacune d'elles, sa durée (déterminée ou aléatoire) et les relations d'ordre la concernant (antériorités obligatoires).
Un programme peut être représenté par un "graphe d'ordonnancement" dont les arcs représentent les relations, la "longueur" des arcs étant une valeur non négative représentant la durée de l'opération. Les sommets, ou "événements" peuvent s'interpréter comme marquant la réalisation d'objectifs partiels.
Le graphe d'un programme est sans circuit. Nous nous limiterons au cas des graphes à une seule "entrée" E_1 (sommet sans prédécesseur) représentant l'événement de départ et une seule "sortie" E_n (sommet sans successeur) représentant l'achèvement des travaux, en notant qu'on peut toujours se ramener à ce cas en introduisant des opérations fictives.
Le problème central consiste à chercher la date minimale de réalisation et les "évenements critiques" (ceux qui ne peuvent souffrir aucun retard sans retarder l'ensemble des travaux). Cette durée ne peut être inférieure à la somme des temps opératoires pris sur le chemin le plus défavorable de E_1 à E_n c'es-à-dire le chemin le plus long ou "chemin critique".

I-2 Définitions

Date attendue ou au plus tôt de l'évènement E_i : notée t_i, c'est le temps de réalisation à partir de E_1, c'est-à-dire la durée correspondant au plus long chemin entre E_1 et E_i.

Date limite ou au plus tard de E_i, notée t_i^*
c'est la date limite de réalisation de E_i au delà de laquelle le temps total d'éxecution est modifié. La différence $t_n - t_i^*$ est égale à la durée correspondant au plus long chemin de E_i à E.

Intervalle de flottement
c'est l'intervalle $[t_i, t_i^*]$ dans lequel pourra se déplacer l'évènement non critique E_i.

Marge libre délai pouvant être apporté au démarrage de l'opération Pij qui fait passer de E_i à E_j. Soit tij sa durée. La marge libre est égale à ti-tj-tij

U ensemble des arcs du graphe

U_i^- ensemble des arcs incidents intérieurement à E_i (type Ej Ei)

U_i^+ ensemble des arcs incidents extérieurement à Ei (type Ei Ej)

I-3 Algorithme

I.3.1 $t_1 = 0$

I.3.2 $t_j = \underset{(i,j)\in U_j^-}{MAX} (t_i + t_{ij})$ $j = 2,3,\ldots n$

I.3.3 $t_n^* = t_n$ = date de réalisation du projet complet

I.3.4 $t_i = \underset{(i,j)\in U_i^+}{MIN} (t_j^* - t_{ij})$, $i=2,\ldots,n-1$.

I.3.5 $t_i^* = t_1 = 0$ = date de démarrage du projet.

On en déduit :

I.3.6 intervalle de flottement de Ei = $t_i^* - t_i$

I.3.7 marge libre de Pij = $t_j - t_i - t_{ij}$

I.3.8 marge totale de Pij = $t_j^* - t_i - t_{ij}$

I.3.9 marge certaine de Pij = $t_j - t_i^* - t_{ij}$

I-4 Remarques

La relation I.3.2 entraîne qu'on ne pourra marquer Ej d'une date attendue que si tous les prédecesseurs de Ej ont été marqués.

De même I.3.4 entraine qu'on ne pourra marquer Ej d'une date limite que si tous les successeurs de Ej ont été marqués.

Plusieurs solutions sont possibles pour remédier à cet inconvénient :

- On peut chercher si tous les prédécesseurs (resp. successeurs) de Ej sont marqués, auquel cas on marque Ej, sinon on cherche à marquer d'autres points en attendant de retrouver Ej par un nouvel essai. Les performances de cette méthode dépendront beaucoup de la configuration du graphe. Le nombre d'essais à effectuer peut être grand, voire infini s'il y a un circuit parasite (dû à une faute de frappe par exemple) qu'on ne pourra détecter par cette méthode :

- On peut aussi, dans une première phase déterminer un ordre total (le graphe est connexe et sans circuit) entre les sommets, ou entre les arcs du graphe, de telle sorte qu'il soit toujours possible de marquer un point. La méthode généralement utilisée consiste à déterminer des niveaux dans le graphe, soit sur le dessin, soit par la méthode de Demoucron [Kaufmann et Desbazeilles 32] , soit en calculant la fermeture transitive de chaque sommet.

- On peut aussi (méthode proposée) utiliser l'ordre total sur les arcs représenté par le mot de sortie de la pile associée au graphe.

II - METHODE DE LA PILE

II-1 Présentation

L'ordre total 0, associé au graphe est entièrement déterminé par la relation Γ et la relation d'ordre Ω_i dans la famille des successeurs d'un point i (Ω_i est arbitraire).

La suite des sorties des arcs de la pile (mot de sortie) représente cet ordre (Pair [45 § 4.3]).

Soit $(\mu_0, \mu_1, \ldots \mu_n)$ cette suite .
On l'utilise pour chercher la date de réalisation (longueur du plus long chemin), le chemin critique et les dates au plus tard. Pour les dates au plus tôt il faut chercher le plus long chemin du point d'entrée à tout point du graphe. Ce serait aussi le plus long chemin dans le graphe (X, Γ^{-1}) de tout point du graphe au point de sortie.

En remplaçant l'ordre Ω_i dans chaque famille des successeurs d'un point par Ω_i^{-1} et Γ par Γ^{-1}, l'ordre fourni par la pile annexe au graphe est 0^{-1} et le mot de sortie de la pile est alors :

$$(\mu_n, \mu_{n-1}, \ldots \mu_0)$$

Il suffira donc d'analyser le graphe à l'aide d'une pile une seule fois.

II-2 Réalisation

Le programme réalisé (sur CAE 510, 1 dérouleur de bande magnétique) calcule la date de réalisation totale, les dates au plus tard, au plus tôt, les intervalles de flottement et cherche un chemin critique. Il signale également les circuits parasites.

II-2-1 Présentation des données

a) Support externe :

Les données sont entrées sous forme de listes des successeurs des différents sommets et des longueurs des arcs correspondants. Ces listes sont séparées par des zéros. On a la structure suivante :

$0 \sqcup A \quad \ldots\ldots \sqcup 0 \sqcup I \sqcup J_{i1} \sqcup L_{i1} \sqcup J_{i2} \sqcup L_{i2} \cdots \sqcup L_n \sqcup 0 \ldots$

ou I est le numéro de code du ième sommet, J_{i1} est le premier successeurs de I dans l'ordre Ω_i et L_{i1} la longueur de l'arc IJ_{i1}
La fin des données est représentée par deux zéros successifs.

b) Mise en place des données sur bandes magnétiques

La première phase du traitement consiste à créer une liste composée uniquement des suites J_{ij} L_{ij} séparées par des zéros. Pour cela, on lit les données 2 par 2, il y a suppression des I et calcul de l'adresse relative du début de la liste des successeurs de I dans la suite des données. Elle est rangée dans le tableau ADRESSE.

Il est possible d'effectuer à ce niveau un premier contrôle de l'impression des données en détectant les décalages (il y a un nombre impair de valeurs entre deux zéros successifs).

La liste ainsi construite est écrite sur bande par blocs de 180 entiers.

II-2.2 Analyse des données

a) écriture du mot de sortie

On construit le mot de sortie de la pile par blocs de 180 mots implantés en mémoire dans le tableau MOT. Soit L l'indice courant dans MOT. Soit i l'instant de sortie de la pile de l'élément x et y le sommet.

α) i-1 était une entrée de x dans la pile alors :

MOT [L+1] := 0
MOT [L+2] := x
MOT [L+3] := y
MOT [L+4] := $\ell(y,x)$

β) i-1 était une sortie de z, x était alors le sommet c'est dire que MOT [L-1] est égal à x, alors :

MOT [L+1] = y
MOT [L+2] = $\ell(y,x)$

Le mot de sortie a donc la structure suivante :

$-1 \sqcup 0 \ldots 0 \ldots . 0 \; x \sqcup y \sqcup \ell(y,x) \sqcup z \sqcup \ell(z,y) \sqcup v \sqcup \ell(v,z) \ldots 0 \ldots . 1$

Les zéros marquent les débuts de séquence. Les extrémités du mot sont repérées par (-1).

b) gestion de la mémoire

La capacité de la mémoire centrale étant assez limitée, le traitement nécessite un swapping important (va et vient de l'information entre la mémoire centrale et les mémoires externes). Signalons que l'information n'étant pas modifiée, il est inutile de la réécrire. D'autre part l'écriture et l'exploitation du mot de sortie étant séquentielles ne présentent aucun problème particulier. Seule l'analyse des données (pile) n'est pas séquentielle. Pour diminuer le nombre de lectures sur la bande nous avons utilisé un procédé de pagination (inspiré de la pagination "hardware" des grands ordinateurs) :

La mémoire disponible est divisée en n blocs physiques appelés pages. (Dans le programme ce sont les tableaux R_1 R_2...R_5). L'information est elle-même divisée en blocs de même dimensions (180 entiers).

Une table des pages (tableau PAGE) indique le numéro du bloc d'information que contient chaque page. A chaque bloc d'information est associé un indicateur (POSITION) contenant 0 si le bloc n'est pas en mémoire centrale, le numéro de la page physique qui le contient s'il l'est. L'utilisation des deux tableaux PAGE et POSITION introduit une certaine redondance mais facilite les transferts.

Pour faire entrer un nouveau bloc d'information, il est nécessaire de libérer une page physique. Le critère retenu pour l'élimination repose sur la fréquence d'appel. Le bloc éliminé sera celui qui est resté le plus longtemps inactif. La CAE 510 n'ayant pas d'horloge interne accessible, nous utiliserons un compteur, incrémenté à chaque appel à la procédure de gestion. Cette procédure (PRELI) cherche le premier élément disponible de la suite des successeurs de I. Le tableau TEMPS contient la valeur de ce compteur au moment de la dernière utilisation de la page considérée. Le tableau ADRESSE permet de connaître la page d'information qui contient la suite cherchée.
La recherche peut être schématisée par l'organigramme II2.2.b (voir page suivante).

II-2.3 Problème central :

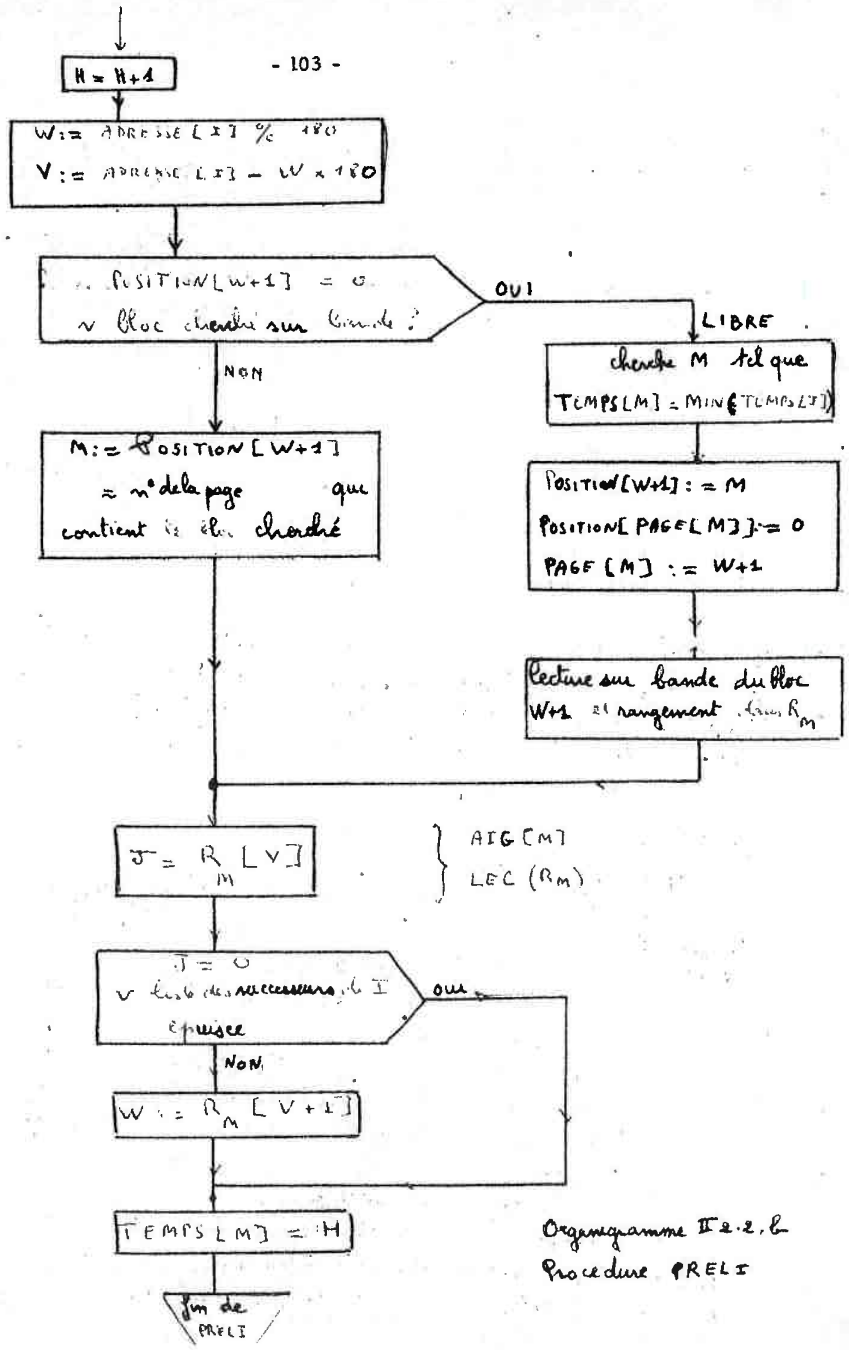
Soit $u_0\ u_1 \ldots u_n$ le mot de sortie de la pile

O_{u_i}	l'origine de l'arc u_i
E_{u_i}	l'extrémité de l'arc u_i
$\ell(u_i)$	sa longueur
E_1	point d'entrée
E_n	point de sortie

On réalise les opérations :

II.2.3.1 $K(x,o) = \begin{cases} \ell(x,E_1) & \text{si } x \sqcap E_n \\ \infty \text{ sinon} \end{cases}$

$j = 1,2,\ldots n$



Organigramme II 2.2.b
Procedure PRELI

II.2.3.2 $K(O_{u_{n-j}}, j) = \max \left[K(O_{u_{n-j}}, j-1),\ K(E_{u_{n-j}}, j-1) + \ell(u_{n-j}) \right]$

$K(O_{u_i}, j) = K(O_{u_i}, j-1) \quad \forall i \neq n-j$

pour calculer les dates au plus tard et pour chercher le chemin critique on utilise l'homomorphisme décrit au § 2.7 [PAIR]; CHE [x] est un tableau qui contient le 2ème point du plus long chemin de x à x_n.

II.2.3.3 $CHE_o(x) = x_n \quad si \quad x \sqcap x_n$

II.2.3.4 $CHE_j(O_{u_{n-j}}) = \begin{cases} CHE_{j-1}(O_{u_{n-j}}) \text{ si } K(O_{u_{n-j}}, j-1) \geqslant \\ \qquad K(E_{u_{n-j}}, j-1) + \ell(u_{n-j}) \\ E_{u_{n-j}} \quad \text{sinon} \end{cases}$

$CHE_j(O_{u_i}) = CHE_{j-1}(O_{u_i}) \quad pour \ \forall i \neq n-j$

L'initialisation de K(x,o) peut être réalisée en cherchant les arcs correspondants dans la suite des données, il est préférable de disposer d'une liste des prédecesseurs de E_n (représentée dans TAR). De même pour les successeurs de E_1 (tableau TO). On calcule les dates au plus tôt de façon analogue.

III- COMPARAISON DES METHODES

a) Problème central

L'utilisation du mot de sortie nécessite exactement m (nombre d'arcs) opérations + Max utilisant les données (mot de sortie) de façon séquentielle pour les dates au plus tôt. Pour les dates au plus tard le découpage en blocs du mot de sortie permet une utilisation quasi-séquentielle.
L'algorithme décrit au § I.3 de ce chapitre nécessite lui aussi m opérations + et Max quand l'ordre des points a été déterminé.
L'utilisation peut aussi être séquentielle a condition d'effectuer préalablement un tri des données.

b) Détermination de l'ordre des arcs

L'utilisation de la pile nécessite par arc : une entrée dans la pile, la consultation d'un indicateur, l'écriture de la partie correspondante du mot de sortie, l une sortie de la pile.

Le graphe étant sans circuit le nombre d'arcs reste toujours inférieur à $\frac{n(n-1)}{2}$. Signalons que dans les réseaux PERT le demi-degré extérieur moyen reste très faible (au maximum 10) et nettement inférieur à $\frac{n(n-1)}{2}$. Sur l'exemple traité d'ordre 180.Il est de l'ordre de 2,5 : il y a 423 arcs pour 180 sommets.
La méthode de Demoucron [Kaufmann et Desbazeilles, 32, § 1.4] nécessite par niveau dans le graphe :
n différences d'éléments de la matrice associée, la recherche des zéros dans la colonne obtenue, la supression des lignes correspondantes. Le nombre de niveaux du graphe reste en général aussi assez faible mais est très souvent très supérieur au demi-degré extérieur moyen (Dans l'exemple d'ordre 180 il y a 22 niveaux). La détermination des niveaux peut être suivie d'un tri des données pour les ranger selon l'ordre croissant des niveaux afin de permettre une utilisation séquentielle, ou même directement du traitement : l'exploitation n'est plus séquentielle et comme les données doivent être lues 2 fois (dates au plus tôt, dates au plus tard) cela allonge le traitement. L'utilisation de la matrice de fermeture transitive [Kaufmann et Desbazeilles § I.4,32] nécessite d'abord n^3 opérations ET et OU pour le calcul de la matrice de fermeture transitive (Algorithme 5) , puis la recherche des niveaux nécessite par niveau : une lecture de toute la matrice pour chercher les lignes à un seul élément non nul, suppression des lignes et colonnes correspondantes. Si on veut une exploitation séquentielle il sera aussi nécessaire de faire un tri des données.

L'inconvénient majeur de ces deux méthodes est qu'elles nécessitent l'utilisation de la matrice associée au graphe, inutile par ailleurs.

CONCLUSION

Notre ambition n'était pas de rédiger l'équivalent d'un programme PERT, mais de montrer une application possible de l'utilisation d'une pile et de donner quelques suggestions pour une telle rédaction.

S'il semble difficile d'étendre la méthode aux problèmes de flots, l'utilisation de la pile peut néanmoins être étendue à d'autres problèmes dans l'établissement d'un programme de travail: prise en compte des probabilités de réalisation des évenements et calcul des probabilités de réalisation des évenements terminaux ; extension au cas des contraintes négatives (à condition qu'il n'y ait pas de circuit de longueur totale négative).

Cette méthode permet d'éviter la redondance des informations d'entrée (liste des successeurs des points et de leurs prédecesseurs), de détecter immédiatement les circuits parasites, de réaliser rapidement l'ordre de traitement des sommets. Elle peut donc présenter un certain intérêt particulièrement pour une implantation sur machine à pile cablée et à grande capacité de mémoire à accès aléatoire (disques, feuillets magnétiques etc...)

ANNEXE

ALGORITHME 7 8 PLUS COURTES DISTANCES

```
 LO'DEBUT' 'ENTIER' N,I,J,K,R,Z,H,L,W,V;'BOOLEEN'EF;     'ENTIER'C,Q;

DEB   : EXL(&#GRAPHE_D_ORDRE$@);IMPR;LICLAV(N);

'DEBUT' 'ENTIER' 'TABLEAU'B,A.(1:N,1:N).,S,T.(I:N).,P.(1:N+1).;

        DEA:LIRT(B);
      'POUR'Z=1'PAS'1'JUSQUA'N'FAIRE''DEBUT'
'AIGUILLAGE' AIG=TEST,NOTP,SOR,SOELSIM,SOR;
  'POUR'I=1 'PAS' 1 'JUSQUA' N 'FAIRE'
 'POUR'J=1 'PAS' 1 'JUSQUA' N 'FAIRE' A.(I,J).= B.(I,J). ;
                    C=     H=0; EF='FAUX' ;
DEBCL  : 'POUR' I=1 'PAS' 1 'JUSQUA'N 'FAIRE'
   'DEBUT' T.(I).=1;'POUR' J=1 'PAS' 1 'JUSQUA' N 'FAIRE'
  'SI' A.(I,J).'EGAL' 0 'ALORS'A.(I,J).=10000 'SINON'
   A.(I,J).=-A.(I,J).;A.(I,I).=0 'FIN' ; K=1;
INIT  :'POUR' I=1 'PAS'1 'JUSQUA' N 'FAIRE'
      'SI' T.(I).'EGAL' 1 'ALORS' 'ALLERA' INIP;'ALLERA'EXIT;
INIP  : P.(1).=I;
TEST  : 'POUR' J=1 'PAS' 1 'JUSQUA' N 'FAIRE'
'SI' A.(I,J). 'INF' 0 'ALORS'  'DEBUT' T.(I).=2;'ALLERA' ENTR'FIN';
     T.(I).=5; 'ALLERA' SOR;
ENTR : K=K+1; A.(I,J).=-A.(I,J).;P.(K). =J; I=J;
     'ALLERA' AIG.(T.(I).);
NOTP  :  T.(I).=3;      'SI'EF'ALORS''DEBUT'H=H+1;S.(H).=I'FIN';
SOR   : 'SI' K 'EGAL' 1 'ALORS' 'ALLERA' INIT;  K=K-1;
    'SI' I'EGA'Z 'ALORS''DEBUT'EF='VRAI';H=1;S.(1).=Z'FIN';
SUIT  : R=I+1; I=P.(K).;
    'POUR' J=R 'PAS'1 'JUSQUA' N 'FAIRE'
   'SI' A.(I,J). 'INF' 0 'ALORS' 'ALLERA' ENTR;
   J=T.(I).; 'ALLERA' 'SI' J 'EGAL' 2 'ALORS'SOELSIM
     'SINON' 'SI' J 'EGAL' 5 'ALORS' SOR 'SINOM' SOR1ER;
SOELSIM  : 'SI' K 'EGAL' 1 'ALORS' 'ALLERA' INIT;J=P.(K-1).;
  'SI' EF 'ALORS''DEBUT'  'POUR'L=1'PAS'1 'JUSQUA'H'FAIRE'
  'DEBUT'R=S.(L).;W=A.(I,R).+A.(J,I).;
     'SI' ABS(A.(J,R).) 'SUP' W 'ALORS'A.(J,R).=W;'FIN';    C=C+1;'FIN';
   T.(I).=4; 'ALLERA' SOR;
SOR1ER:'SI'EF'ALORS''POUR'L=1'PAS'1'JUSQUA'H'FAIRE''DEBUT'R=S.(L).; C=C+1;
   'POUR' V= 1 'PAS' 1 'JUSQUA' N 'FAIRE'
    'DEBUT' W=ABS(A.(V,I).); 'SI' W 'INF' 10000 'ALORS'
    'DEBUT'          W=W+A.(I,R).;'SI' A.(V,R). 'SUP' W 'ALORS' A.(V,R).=W
    'FIN' 'FIN' 'FIN'  ;'ALLERA' SOELSIM;        EXIT:
  'SI' CLE(6)'ALORS''POUR'I=1'PAS'1'JUSQUA'N 'FAIRE'
      'SI'A.(I,Z).'INF'10000'ALORS'EXE(3,A.(I,Z).)'SINON'EXE(3,0);IMPR;
    EXE(5,   C);IMPR'FIN';
'ALLERA''SI'CLE(7)'ALORS' DEB 'SINON' DEA 'FIN' 'FIN' #
```

```
LISTE SOURCE
010 0000
001 0000 'DEBUT''COMMENTAIRE' ESSAIS PROCEDURE MARIMONT;
002 0000          'ENTIER'I,N; D:   LICLAV(N);
003 0013          'DEBUT''ENTIER''TABLEAU'M.(1:N,1:N).;
100 0028 'BOOLEEN''PROCEDURE' MARIMONT(M,N); 'ENTIER'N; 'ENTIER''TABLEAU' M;
101 0044 'DEBUT''ENTIER'I,J,S; 'BOOLEEN' NODEL;
102 0056          'BOOLEEN''TABLEAU' IN.(1:N).;
103 0066 'POUR'I=1'PAS'1'JUSQUA'N'FAIRE'IN.(I).='VRAI';
104 0082 ITER:    NODEL='VRAI';
105 0098          'POUR'I=1'PAS'1'JUSQUA'N'FAIRE''SI'IN.(I).'ALORS'
106 0103          'DEBUT' S=0;
107 0108          'POUR'J=1'PAS'1'JUSQUA'N'FAIRE'S=S+M.(I,J).;
108 0128          'SI'S'EG'0'ALORS''DEBUT'IN.(I).='FAUX';
109 0141          NODEL='FAUX''FIN''FIN';
111 0147          'POUR'J=1'PAS'1'JUSQUA'N'FAIRE''SI'IN.(J).'ALORS'
010 0162          'DEBUT'S=0;'POUR'I=1'PAS'1'JUSQUA'N'FAIRE'S=S+M.(I,J).;
112 0187          'SI'S'EG'0'ALORS''DEBUT'IN.(J).='FAUX';
113 0200          NODEL='FAUX''FIN''FIN';
114 0206          'SI'NODEL'ALORS''DEBUT'MARIMONT='VRAI';'ALLERA'END'FIN';
115 0218          'POUR'I=1'PAS'1'JUSQUA'N'FAIRE'
116 0227          'SI'IN.(I).'ALORS''ALLERA'ITER;
117 0236               MARIMONT='FAUX';
118 0240 END:     'FIN' MARIMONT;
004 0244 LIRT(N);
005 0249 'SI'MARIMONT(N,N)'ALORS'EXL(&AVEC_CIRCUITS@)
006 0261 'SINON'EXL(&SANS_CIRCUIT@);IMPR;        'ALLERA'D;
007 0274 'FIN''FIN';
CODIFICATION TERMINEE
*FF016731 FP010074 DP007426
GENERATION TERMINEE
```

* PLUS LONGS CHEMINS ALGORITHME 5

```
$ALGOL      L

LISTE SOURCE
001 0000 'DEBUT''ENTIER'W,N,I,J,K,C;
002 0014 DEB:EXL(&GRAPHE_D_ORDRE@);IMPR;
003 0024 LICLAV(N);
004 0029 'DEBUT''ENTIER''TABLEAU'A.(1:N,1:N).,B.(1:N).;
005 0053 LIRT(A);
006 0058 'POU'I=1'PAS'1'JUSQUA'N'FAIRE'
007 0067 'DEBUT''POUR'J=1'PAS'1'JUSQUA'N'FAIRE'
008 0077 'SI'A.(I,J).'EGAL'0'ALORS'A.(I,J).=-90000;
009 0097 A.(I,I).=0 'FIN';
010 0107 DEBCL: 'POUR'I=1'PAS'1'JUSQUA'N'FAIRE'
011 0118 'DEBUT''POUR'J=1'PAS'1'JUSQUA'N'FAIRE'
012 0128 B.(J).=A.(I,J).;
013 0140 'POUR'J=1'PAS'1'JUSQUA'I-1,I+1'PAS'1'JUSQUA'N'FAIRE'
014 0159 'SI'A.(J,I).'SUP'      0 'ALORS'
015 0169 'DEBUT'W=A.(J,I).;
016 0179 'POUR'K=1'PAS'1'JUSQUA'N'FAIRE'
017 0188 'SI'B.(K).+W'SUP'A.(J,K).'ALORS'A.(J,K).=W+B.(K).
018 0216 'FIN' 'FIN';
019 0219 'SI' 'NON' CLE(6) 'ALORS' 'ALLERA' DEB ;
020 0229 EXL(&MATRICE_DES_DISTANCES$@);IMPR;
021 0237 'POUR'I=1'PAS'1'JUSQUA'N'FAIRE'
022 0246 'DEBUT''POUR'J=1'PAS'1'JUSQUA'N'FAIRE'
023 0256 'SI'A.(I,J).'DIF'90000'ALORS'EXE(3,A.(I,J).)
024 0276 'SINON'EXE(3,0);IMPR'FIN';
025 0288 'ALLERA'DEB'FIN''FIN'$
CODIFICATION TERMINEE
=FF016641 FP010200 DP007430
GENERATION TERMINEE
```

```
'DEBUT'   EXL(<#DERNIAME__ALGORITHME_DE_BELLMANN-KALABA$>);IMPR;
DEB:              'DEBUT' 'ENTIER' I,J,N,E,W;
                  LICLAV(N);
                  'DEBUT' 'ENTIER' 'TABLEAU'A.(1:N,1:N).,V.(1:N).;
                  'BOOLEEN' 'TABLEAU'C.(1:N).;
                  'BOOLEEN' D;
                  LIRT(A);EXL(<EXTREMITE>);IMPR;
EXT:              LICLAV(E);'SI' E 'EGAL' 0 'ALORS' 'ALLERA' DEB;
                  'POUR' I:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
                  'DEBUT' 'POUR'J:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
                  'SI' A.(I,J). 'EGAL' 0 'ALORS' A.(I,J).:=-10000;
                  C.(I).:='VRAI';V.(I).:=A.(I,E).'FIN';
                  V.(E).:=0;C.(E).:='VRAI';
ITER:             D:='FAUX';
                  'POUR' I:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
                  'SI' C.(I).'ALORS'  'DEBUT' C.(I).:='FAUX';W:=0;
                  'POUR' J:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
                  'SI' A.(I,J).+V.(J).'SUP' W 'ALORS' W:=A.(I,J).+V.(J).;'SI' W 'DIF' V.(I). 'ALORS'
                  'DEBUT' V.(I). :=W;C.(I).:='VRAI';D:='VRAI''FIN' 'FIN';
                  'SI' D 'ALORS' 'ALLERA'ITER;
                  'POUR' I:=1 'PAS' 1 'JUSQUA' N 'FAIRE'
                  'SI' V.(I). 'INF'0 'ALORS' EXE(3,0)'SINON'EXE(3,V.(I).);
                  IMPR;  'ALLERA' EXT;'FIN' 'FIN' 'FIN' ;
                  #
```

BIBLIOGRAPHIE

1 ARINAL J. C. — Réduction maximale du chemin minimal entre deux sommets d'un graphe. Revue AFIRO n° 37 pp 323-332

2 ABERTH O. — On the sum of graphs Revue AFIRO n° 33

3 BELLMANN R. — On a routing problem quart. Appl. Math. 16 (1958) p. 87-90

4 BERGE C. — Théorie des graphes et ses applications. Dunod 1958 - 1963

5 BERGE C. et GHOUILA-HOURI — Programmes jeux et réseaux de transport Dunod 1962

6 BERTIER P. — Quelques algorithmes pour les problèmes de tpurnée Metra Vol IV n° 4 . 1965 p. 607

7 BOCK-KANTNER — An algorithm(the r-th best path algorithm) for finding and ranking paths through a network Arman Research Foundation Research Report May 1958

8 BORILLO M. — Sur la détermination de l'algorithme optimal pour la recherche du plus court chemin dans un graphe sans circuit. Revue AFIRO 33.

9 CAMION — Quelques propriétés des chemins et circuits hamiltoniens dans la théorie des graphes. Cahiers du Centre de R. O. Belge V. 2 n°1 p. 60

10 CARRE P. — Le problème du commis voyageur . Revue des Ingénieurs Fev. 1964

11 CLARKES - KRIKORIAN - RANSEN. Computing the N best loopless paths in a network. J. Soc. Industr. Appl. Math. 11n°4 dec. 63

12 CRUON-HERVE — Quelques résultats relatifs à une structure algébrique et à son application au problème central d'ordonnancement. Revue AFIRO n° 34

13 DANTZIG G. B. "The shortest route problem" Opérations Research 5, (1957), 270, 273.

14 DANTZIG G. B. On the shortest route through a network Management Sc., V. 6 n°2, 1960 p 187

15 DANTZIG G. B. "All shortest routes in a graph" 10 juin 1966 comunication personnelle.

16 DANTZIG G. B. All shortest routes from a fixed origin in a graph Communication personnelle.

17 DIRAC G. A. Paths and Circuits in graphs extreme cases. Acta. Math. Acad. Sci. Hungar V. 10 (3-4) 1959 p 351-361

18 EMOND A. Application de la notion de pile à des problèmes portant sur les chemins des graphes. Thèse de 3ème cycle - Faculté des Sciences NANCY (1965).

19 ERDOS P. - GALLAI T. On maximal Paths and Cirduits of Graphs Acta Math. Acad. Sci. Hungar - V. 10 (3-4) 1959 p. 337 - 350

20 FORD, L. R. Network Flow Theory Rand Memo p. 923 1956

21 FORD L. R. - FULKERSON A Network flow feasibility theorem and combinational applications. Canad. J. Math. 11, 1959, p440.

22 FOULKES Directed graphs and assembly schedules 10th symposium on applied math. p. 281. Ameri. Soc. 1960

23 FULKERSON D. R. Expected critical path longths in PERT networks. 1962 Opns. Res. Vol 10 n° 6.

24 GHOUILA-HOURI Une conditions suffisante d'existence d'un circuit hamiltonien. C. R. Acad. Sci. PARIS Juillet 1960

25 GONZALES Solution of the salesman problem by dynamic programming on the hypercubes. MIT Operations Res. CTR Interim TECH. report NIB. 1962

26 HOFFMAN W. - PAVLEY R. A method for the solution of the N. th best path problem. J. Assoc. Comput. mach. 6 (1959) p. 506-514

27 HU T. C. Multiterminal shortest paths Operations research Cent. Univ. California Berkeley V. ORC 65-11 (1965)

28 IVANESCU P. L. - ROSENBERG I. Application of pseudo-booleen programming to the theory of graphs. Z. Wahrscheinlichkeitstheorie und verru. geluite V. 3 (1964) p. 163/76.

29 KALABA R. - BELLMANN R. On h-th best Policier Industrial and Applied Math. Vol. 8 n° 4 Dec. 1960 p. 582-588

30 KARG R. - THOMPSON G. L. A heuristic approach to solving travelling salesman problems. Management Sci. Vol. 10 n°2 Janv. 1964

31 KAUFMANN A. La théorie des graphes et ses applications Informations scientifiques BULL.

32 KAUFMANN A. - DESBAZEILLES La méthode du chemin critique. Dunod ed. 1964

33 KAUFMANN A. - et CRUON La programmation dynamique Dunod ed. 1965

34 KAUFMANN A. - MALGRANGE Recherche des chemins et circuits hamiltoniens d'un graphe Revue française de R. O. n° 26

35 KLEE V. A string algorithm for shortest path in directed networks. Opns. Res. V. 12 (1964) p. 428-432

36 KONIG Theorie der endlichen und umendlichen graphen Leipzig 1936

37 KRUSKAL J. B. On the shortest spanning subtree on a graph. Proc. Amer. Math. Soc. n° 7 p 48 (1956)

38 MAGHOUT K. Sur la détermination des nombres de stabilité et du nombre chromatique d'un graphe. C. R. Acad. T 248 p. 3522 (1959)

39 MARIMONT B. - ROSALIND — A new method of Checking the Consistency of Precedence Matrices. J. of the A. C. M. 1959

40 MARIMONT R. — Applications of graphs and bolean Matrices to Computer Programming SIAM Rev. V. 2 (4) 1960 p. 258-268

41 MINTY C. J. — A comment on the shortest route problem. Opns. Res. 5 (1957) p. 274

42 NARAHARI — The shortest route problem Opns. Res V. 9 (1) 1961 p. 129-132

43 NOLIN L. — Traitement des données groupées Institut Blaise Pascal - PARIS

44 PAIR C. — Etude de la notion de pile et application à l'analyse syntaxique. Thèse d'Etat. Université de Nancy

45 PAIR C. — Sur des algorithmes pour les problèmes de cheminement dans les graphes finis. Actes du 3ème Congrés Int. sur la théorie des graphes. Rome Juillet 1966 - A paraître.

46 PEART R. M. - RANDOLPH R. H. - BARTLETTT. E. — The shortest route problem. Opns. Res 8 (6) (1960) p. 866-868

47 PICARD C. — Théorie des questionnaires Mai 1963 n° 19 3. 3. /BI - Institut Blaise Pascal

48 PICARD C. — Graphes complémentaires et graphes planaires Rev. AFIRO n° 33

49 POLLAC M. - WIEBENSON W. — Solutions of the shortest route problem. Anreview. Opns. res. V 8 (2) (1960) p 224-230

50 POSA L. — On circuits of finite graphs Magyar Tud. Akad. V. BA3 (1963) p. 355-361

51 QUAST J. - SCHUCH F. — A number of paths problem Simon Stevin V. 27 (1450) p. 201-211

52 QUINTAS L. SUPNICK — Extrem: Hamiltonian Circuits. Resolution of the Convex - Even Case. Amer. Math. Soc. V 16 n° 5 - Oct. 65

53 ROBACKER J. T. — Some Experiments on the travelling salesman problem. Management Sci. Vol. 10 n° 2 (1964)

54 ROY B. — Cheminnement et connexité dans les graphes. Thèse PARIS (1961) - Metra n°1 Mars 1962

55 ROY B. — Graphes et ordonnancements SOFRO n° 25

56 ROY B. — Contribution de la théorie des graphes à l'étude de certains problèmes linéaires. C. R. Acad. Sciences T. 248 p 2437 Avril 59

57 RUDEANU S. — Notes sur l'exitence et l'unicité du noyau d'un graphe Revue AFIRO n° 33

58 WARSHALL — Atheorem on boolean matrices J. of the A. C. M. Jan. 1962 p. 11-12

59 WECHSEL P. M. — The Kronecker product of graphs Proc. Amer. Math. Soc. 13 (1962) p. 47-57

60 WHITING P. D. - HILLIER J. A. — A method for finding the shortest route through a road network Opns. Res. quart. V 11 (1-2) 1960 p. 37/40

61 WING O. — Algorithms to find the most reliable path in a network IRE Trans. C. T. V 8 (1) (1961) p. 78/79

Vu et Approuvé
Nancy, le
Le Doyen de la Faculté
des Sciences de NANCY,

M. AUBRY

Vu et Permis d'imprimer
Nancy, le
le Recteur :
Président du Conseil de
l'Université,

P. IMBS