Codage des automates asynchrones

Gabrièle Saucier Schnebelen

N° d'ordre

THESES

présentées à

LA FACULTE DES SCIENCES DE L'UNIVERSITE DE GRENOBLE

pour obtenir

LE GRADE DE DOCTEUR ES SCIENCES MATHEMATIQUES

par

Gabrièle SAUCIER SCHNEBELEN

Première Thèse :

Codage des automates asynchrones

Deuxième thèse :

CORRESPONDANCE DE POST

Thèses soutenues le 16 Novembre 1970 devant la Commission d'Examen

Monsieur	J. KUNTZMANN	Président
Messieurs	R. PERRET	Examinateur
	C. BENZAKEN	Examinateur
	A. GRASSELLI	Invité

L I S T E D E S P R O F E S S E U R S

Doyen honoraire : Monsieur M. MORET
Doyen : Monsieur E. BONNIER

PROFESSEURS TITULAIRES

MM.	NEEL Louis	Physique Expérimentale
	KRAVTCHENKO Julien	Mécanique Rationnelle
	CHABAUTY Claude	Calcul différentiel et intégral
	BENOIT Jean	Radioélectricité
	CHENE Marcel	Chimie Papetière
	FELICI Noël	Electrostatique
	KUNTZMANN Jean	Mathématiques Appliquées
	BARBIER Reynold	Géologie Appliquée
	SANTON Lucien	Mécanique des Fluides
	OZENDA Paul	Botanique
	FALLOT Maurice	Physique Industrielle
	KOSZUL Jean-Louis	Mathématiques
	GALVANI Octave	Mathématiques
	MOUSSA André	Chimie Nucléaire
	TRAYNARD Philippe	Chimie Générale
	SOUTIF Michel	Physique Générale
	CRAYA Antoine	Hydrodynamique
	REULOS René	Théorie des Champs
	BESSON Jean	Chimie Minérale
	AYANT Yves	Physique Approfondie
	GALLISSOT François	Mathématiques
Melle.	LUTZ Elisabeth	Mathématiques
MM.	BLAMBERT Maurice	Mathématiques
	BOUCHEZ Robert	Physique Nucléaire
	LLIBOUTRY Louis	Géophysique
	MICHEL Robert	Minéralogie et pétrographie
	BONNIER Etienne	Electrochimie et Electrométallurgie
	DESSAUX Georges	Physiologie animale
	PILLET Emile	Physique Industrielle-Electrotechnique
	YOCCOZ Jean	Physique Nucléaire théorique
	DEBELMAS Jacques	Géologie Générale
	GERBER Robert	Mathématiques
	PAUTHENET René	Electrotechnique
	MALGRANGE Bernard	Mathématiques Pures
	VAUQUOIS Bernard	Calcul Electronique
	BARJON Robert	Physique Nucléaire

MM.	BARBIER Jean-Claude	Physique
	SILBER Robert	Mécanique des Fluides
	BUYLE-BODIN Maurice	Electronique
	DREYFUS Bernard	Thermodynamique
	KLEIN Joseph	Mathématiques
	VAILLANT François	Zoologie et Hydrobiologie
	ARNAUD Paul	Chimie
	SENGEL Philippe	Zoologie
	BARNOUD Fernand	Biosynthèse de la Cellulose
	BRISSONNEAU Pierre	Physique
	GAGNAIRE Didier	Chimie Physique
Mme.	KOFLER Lucie	Botanique
MM.	DEGRANGE Charles	Zoologie
	PEBAY-PEROULA Jean-Claude	Physique
	RASSAT André	Chimie Systématique
	DUCROS Pierre	Cristallographie Physique
	DODU Jacques	Mécanique Appliquée I. U. T.
	ANGLES D'AURIAC Paul	Mécanique des Fluides
	LACAZE Albert	Thermodynamique
	GASTINEL Noël	Analyse numérique
	GIRAUD Pierre	Géologie
	PERRET René	Servo-mécanisme
	PAYAN Jean-Jacques	Mathématiques Pures

PROFESSEURS SANS CHAIRE

MM.	GIDON Paul	Géologie
Mme.	BARBIER Marie-Jeanne	Electrochimie
Mme.	SOUTIF Jeanne	Physique
	COHEN Joseph	Electrotechnique
	DEPASSEL R.	Mécanique des Fluides
	GLENAT René	Chimie
	BARRA Jean	Mathématiques Appliquées
	COUMES André	Electronique
	PERRIAUX Jacques	Géologie et Minéralogie
	ROBERT André	Chimie Papetière
	BIARREZ Jean	Mécanique Physique
	BONNET Georges	Electronique
	CAUQUIS Georges	Chimie Générale
	BONNETAIN Lucien	Chimie Minérale
	DEPOMIER Pierre	Physique Nucléaire-Génie Atomique
	HACQUES Gérard	Calcul numérique
	POLOUJADOFF Michel	Electrotechnique
Mme.	KAHANE Josette	Physique
Mme.	BONNIER Jane	Chimie
MM.	VALENTIN Jacques	Physique
	REBECQ Jacques	Biologie
	DEPORTES Charles	Chimie
	SARROT-REYNAULD Jean	Géologie
	BERTRANDIAS Jean-Paul	Mathématiques Appliquées
	AUBERT Guy	Physique

PROFESSEURS ASSOCIES

MM.	RODRIGUES Alexandre	Mathématiques Pures
	MORITA Susumu	Physique Nucléaire
	RADHAKRISHNA	Thermodynamique

MAITRES DE CONFERENCES

MM.	LANCIA Roland	Physique Atomique
Mme.	BOUCHE Liane	Mathématiques
MM.	KAHANE André	Physique Générale
	DOLIQUE Jean Michel	Electronique
	BRIERE Georges	Physique
	DESRE Georges	Chimie
	LAJZEHOWICZ Joseph	Physique
	LAURENT Pierre	Mathématiques Appliquées
Mme.	BERTRANDIAS Françoise	Mathématiques Pures
MM.	LONGEQUEUE Jean-Pierre	Physique
	SOHM Jean-Claude	Electrochimie
	ZADWORNY François	Electronique
	DURAND Francis	Chimie Physique
	CARLIER Georges	Biologie végétale
	PFISTER Jean-Claude	Physique
	CHIBON Pierre	Biologie animale
	IDELMAN Simon	Physiologie animale
	BLOCH Daniel	Electrotechnique I. P.
	MARTIN-BOUYER Michel	Chimie (C. S. U. Chambéry)
	SIBILLE Robert	Construction mécanique (I. U. T.)
	BRUGEL Lucien	Energétique I. U. T.
	BOUVARD Maurice	HYdrologie
	RICHARD Lucien	Botanique
	PELMONT Jean	Physiologie animale
	BOUSSARD Jean-Claude	Mathématiques Appliquées (I. P. G.)
	MOREAU René	Hydraulique I. P. G.
	ARMAND Yves	Chimie I. U. T.
	BOLLIET Louis	Informatique I. U. T.
	KUHN Gérard	Energétique I. U. T.
	PEFFEN René	Chimie I. U. T.
	GERMAIN Jean-Pierre	Mécanique
	JOLY Jean-René	Mathématiques Pures
Melle.	PIERY Yvette	Biologie animale
	BERNARD Alain	Mathématiques Pures
	MOHSEN Tahsin	Biologie (C. S. U. Chambéry)
	CONTE René	Mesures Physiques I. U. T.
	LE JUNTER Noël	Génie Electrique Electronique I. U. T.
	LE ROY Philippe	Génie Mécanique I. U. T.
	ROMIER Guy	Techniques Statistiques quantitatives I. U. T.
	VIALON Pierre	Géologie
	BENZAKEN Claude	Mathématiques Appliquées
	MAYNARD Roger	Physique

MM.	DUSSAUD René	Mathématiques (C. S. U. Chambéry)
	BELORIZKY Elie	Physique (C. S. U. Chambéry)
Mme.	LAJZEROWICZ Jeannine	Physique (C. S. U. Chambéry)
M.	JULLIEN Pierre	Mathématiques Pures
Mme.	RINAUDO Marguerite	Chimie
MM.	BLIMAN Samuel	E. I. E.
	BEGUIN Claude	Chimie Organique
	NEGRE Robert	I. U. T.

MAITRES DE CONFERENCES ASSOCIES

MM.	YAMADA Osamu	Physique du Solide
	NAGAO Makoto	Mathématiques Appliquées
	MAREZIO Massimo	Physique du Solide
	CHEECKE John	Thermodynamique
	BOUDOURIS Georges	Radioélectricité
	ROZMARIN Georges	Chimie Papetière

Je remercie Monsieur le Professeur KUNTZMANN qui a jalonné ce travail de ses conseils et de ses critiques. Je lui suis spécialement reconnaissante de sa grande compréhension des problèmes inextricables posés par le cumul des fonctions d'épouse, de mère et de chercheur.

Je remercie Monsieur le Professeur PERRET d'avoir accepté de faire partie de mon jury et me rappelle que c'est son enseignement de valeur qui m'a fait découvrir le charme des machines séquentielles.

Je suis très touchée et très honorée de la présence de Monsieur le Professeur GRASSELLI de l'Université de Pise. Je lui suis particulièrement reconnaissante de m'avoir mise en contact avec des spécialistes étrangers, travaillant sur le même problème.

Je remercie Monsieur BENZAKEN, Maître de Conférences d'avoir accepté de participer au jury et d'avoir élargi mes connaissance en me donnant le sujet d'une deuxième thèse.

J'ai beaucoup apprécié la compétence de Melle BICAÏS et du service de tirage qui ont assuré la réalisation matérielle de cette thèse et les en remercie.

TABLE DES MATIERES

INTRODUCTION

C'est un problème concret et précis qui nous amené à entreprendre ce travail. Le problème du codage des automates asynchrones est, en effet, un problème non résolu d'une manière satisfaisante en théorie comme en pratique et qui se pose au cours de la recherche de la synthèse d'un tel automate. On peut chercher soit à éviter ce problème en cherchant des solutions universelles très simples, qui seront forcément onéreuses, soit à améliorer les solutions heuristiques existantes. On se reportera à [26] pour la contribution apportée au problème de la recherche de solutions universelles. Seul le deuxième problème sera examiné ici.

L'outil de choix pour l'étude du problème de codage semble avoir été, jusqu'à présent, la théorie des partitions (LIU [18] ; TRACEY [30] , etc...). Néanmoins, les limites pratiques de cet outil qui sont celles des méthodes de couverture sont bien connues. Il nous a semblé qu'un outil avait été trop vite abandonné, dans ce domaine : la théorie des graphes. Nous avons donc reformulé le problème d'une manière adéquate (Chap. 1 et Chap. 2). Deux méthodes d'immersion d'un graphe connexe dans le n-cube sont proposées dans les chapitres 3 et 4 et la méthode de codage qui en résulte exposée au Chapitre 5. Nous avons tenu à vérifier, en pratique, par des programmes, certes rudimentaires, la compétitivité de cette méthode par rapport aux précédentes et l'amélioration en temps d'obtention et en dimension de la variable interne sur les exemples traités dans la littérature.

Un chapitre spécial (Chapitre 6) a été consacré à un point qui nous paraît être l'intérêt essentiel d'une méthode fondée sur la théorie des partitions : la forme systématique des équations de la variable interne.

Nous avons également consigné les résultats obtenus relatifs au problème de la recherche d'un isomorphisme entre 2 graphes. L'algorithme proposé, nous semble améliorer très nettement ceux proposés par UNGER [32] ou STEEN [28].

CHAPITRE - I

RAPPELS ET DEFINITIONS

I - AUTOMATE CONSIDERE

I-1 - Si nous reprenons la nomenclature d'Hartmanis [13] , un automate séquentiel est défini par le quintuplet $M = (S,E,O,\delta,\lambda)$

avec S : ensemble fini d'états

E : ensemble fini d'entrées

O : ensemble fini de sorties

δ : application $S \times E \xrightarrow{\delta} S$ appelée fonction état suivant

λ : une des 2 fonctions $S \xrightarrow{\lambda} O$

$$S \times E \xrightarrow{\lambda} O$$

δ s'étend naturellement à $S \times E^* \to S$ où E^* est le monoïde libre engendré par E.

I-2 - Automate asynchrone

Les circuits séquentiels réalisant, en pratique, le fonctionnement d'automates séquentiels, sont habituellement classés en 2 catégories : les circuits synchrones et les circuits asynchrones. Les circuits asynchrones sont caractérisés par l'absence d'horloge, et après application d'un signal d'entrée évoluent d'une façon autonome et continuellement dans le temps.

Un automate séquentiel peut être réalisé par un circuit séquentiel séquentiel asynchrone si

pour tout couple (s,e) $s \in S$, $e \in E$ il existe un entier r tel que $\delta(s,e^{r+1}) = \delta(s,e^{r})$ où e^{r} représente une séquence d'entrée $\underbrace{e\ e\ \ldots\ e}_{r \text{ termes}}$

L'état $\delta(s,e^{r})$ sera appelé état stable successeur du couple (s,e).

Un automate séquentiel possédant cette propriété sera appelé un automate asynchrone.

I-3 - Automate asynchrone proprement dit

Tout automate asynchrone peut être ramené à un automate tel que pour tout $s \in S$ et pour tout $e \in E$ $\delta(s,e^{2}) = \delta(s,e)$; tout successeur d'un état est un état stable. Nous considèrerons donc de tels automates dans cette étude.

I-4 - Extension de la définition des automates séquentiels ; automates incomplètement définis

Nous considérons dans cette étude, le cas général, effectivement rencontré en pratique où la fonction $\delta(s,e)$ peut n'être pas définie pour certains couples (s,e) . Si l'ensemble de tels couples est vide, nous sommes ramenés au cas particulier d'automates complètement définis. Dans la suite de cette étude, nous désignerons donc par automate séquentiel, le cas le plus général d'automates incomplètement définis.

II - ENSEMBLE DE PARTITIONS $\{\pi_i\}$ RELATIVES AUX DIFFERENTES ENTREES D'UN AUTOMATE

II-1 - Définition

Considérons une entrée E_j d'un automate séquentiel asynchrone ; soit $S_{E_j} \subseteq S$, l'ensemble des états de l'automate tel que, pour tout $s \in S_{E_j}$, $\delta(s,E_j)$ est défini. On définit la relation d'équivalence sur S_{E_j} :

$$s \equiv t \iff \delta(s,E_j) = \delta(t,E_j).$$

Soit π_j la partition de S_{E_j} ainsi définie, appelée *partition relative à l'entrée* E_j *de l'automate*. En considérant les entrées $E_1 \ldots E_i \ldots E_n$ de l'automate, on obtient ainsi un ensemble de partitions $\pi_1 \ldots \pi_i \ldots \pi_n = \{\pi_i\}$ appelé dans la suite de cette étude, *ensemble de partitions relatives aux différentes entrées*

II-2 - Exemple

Soit l'automate dont le tableau des successeurs ou "tableau des états" est le suivant :

	successeurs		
états	E_1	E_2	E_3
1	2	3	(1)
2	(2)	4	(2)
3	5	(3)	1
4	5	(4)	1
5	(5)	(5)	2
6	-	3	2

Remarque :
Les états encerclés représentent les états stables ;
Nous avons :

$$\pi_1 = (12,345)$$

$$\pi_2 = (136,24,5)$$

$$\pi_3 = (134,256)$$

Cet ensemble de partitions joue un rôle fondamental pour le problème étudié.

III - CONTRAINTES DE CODAGE D'UN AUTOMATE SEQUENTIEL ASYNCHRONE

III-1 - Définition du codage

Dans l'ensemble des travaux exposés ici, coder un automate consiste à associer injectivement à chaque état de l'automate un sommet et un seul d'un n-cube. Nous aurons donc à associer à chaque état s_i une image $I(s_i)$ sur le cube telle que :

$$s_i \neq s_j \iff I(s_i) \neq I(s_j).$$

Remarque :
Dans d'autres travaux [15], on associe à un état un sous-ensemble de sommets du cube.

III-2 - Contrainte impérative

III-2-1 - Définition

La contrainte exposée ci-dessous est nécessaire au bon fonctionnement du circuit séquentiel obtenu après codage.

Considérons une entrée E_i de l'automate et soit π_i la partition relative à cette entrée (définie au chapître précédent). Si pour l'entrée E_i, nous avons q états stables, la partition π_i contiendra q blocs $(B_{i_1}, B_{i_2} \ldots B_{i_q})$.
Le codage de l'automate (ou application I) est compatible avec le fonctionnement désiré, si, quelque soit l'entrée E_i, on peut associer à chaque bloc B_{i_j} un sous-graphe partiel G_{i_j} du n-cube tel que

α) $I(B_{i_j}) \subseteq \{\text{sommets de } G_{i_j}\}$

β) G_{i_j} est connexe

γ) $G_{i_j} \cap G_{i_k}$ est un graphe vide pour $j \neq k$;

on se réfèrera à Berge [3], pour la définition des termes graphe partiel, sous graphe, sous-graphe partiel employés ici.

III-2-2 - Sommets supplémentaires, états supplémentaires

On appelle ensemble de sommets supplémentaires associés à B_{i_j} et noté X_{i_j}, l'ensemble défini par

$$X_{i_j} = \{\text{sommets de } G_{i_j}\} - I(B_{i_j})$$

Un sommet de X_{i_j} peut être l'image par I d'un état de l'automate, n'ayant pas de successeur pour l'entrée E_i, soit l'image d'aucun état de l'automate. On prolongera l'application I en créant des états artificiels, dans le second cas. On appelera, alors, états supplémentaires S_{i_j} associés à B_{i_j}, l'ensemble défini par

$$I(S_{i_j}) = X_{i_j}.$$

S_{i_j} est donc composé soit d'états de l'automate n'ayant pas de successeurs pour E_i, soit d'états artificiellement créés.

Remarque :

S_{i_j} peut être vide

$S_{i_j} \cap S_{k_\ell}$ $(i \neq k)$ est, en général, non vide.

Exemple :

Soit l'exemple tiré de TRACEY [30] qui sera repris ultérieurement :

	E_1	E_2	E_3	E_4
a	(a)	c	d	c
b	a	f	c	(b)
c	f	(c)	(c)	(c)
d	-	(d)	(d)	b
e	a	d	c	(e)
f	(f)	(f)	-	e

les partitions $\{\pi_i\}$ associées sont :

$\pi_1 = \{\underline{ab}e, \underline{fc}\}$

$\pi_2 = \{\underline{ca}, \underline{de}, \underline{fb}\}$

$\pi_3 = \{\underline{cb}e, \underline{da}\}$

$\pi_4 = \{\underline{bd}, \underline{ca}, \underline{ef}\}$

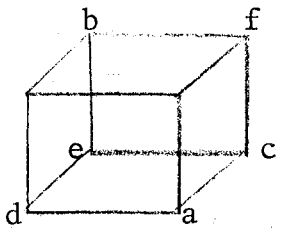
Les blocs à l'intérieur d'une partition sont rangés dans l'ordre alphabétique des états stables y figurant.

Montrons que le codage suivant $\begin{array}{cccccc} a & b & c & d & e & f \\ \hline 0 & 4 & 1 & 2 & 3 & 5 \end{array}$

où le code d'un état sur le 3-cube est donné par son équivalent décimal, est compatible avec l'automate.

Tableau des sommets supplémentaires

	E_1	E_2	E_3	E_4
B_{i_1}	2	-	5	6
B_{i_2}	-	-	-	-
B_{i_3}		-		7



- On vérifie qu'à chaque bloc est ainsi associé un sous-graphe partiel connexe du 3-cube.

- l'image de l'état d dont le successeur n'est pas défini pour E_1 est utilisé comme élément de X_{1_1}.

III-3 - Autres contraintes

III-3-1 - Minimalité du nombre de composantes de la variable interne ; ce nombre est directement lié au nombre de mémoires nécessaires pour la réalisation du circuit.

III-3-2 - Temps d'obtention du codage.

Ce temps peut se chiffrer en temps homme, ou temps machine et est, bien sûr, proportionné à la dimension du tableau d'états de l'automate considéré.

III-3-3 - Temps de transition.

On peut s'imposer ce critère comme critère impératif et rechercher des codages à temps de transition unitaires (S.T.T. assignments). Dans ce travail nous ne minimiserons ces temps de transition que dans un second temps.

III-3-4 - Simplicité des équations obtenues.

Ce critère est particulièrement important en vue des réalisations modulaires ou de réalisations avec des éléments donnés (mémoire et opérateurs logiques). Il serait souhaitable d'en tenir compte au moment du codage, mais le problème est alors très complexe.

IV - CODAGES UNIVERSELS ET CODAGES SPECIFIQUES

Un codage universel pour les automates dont le tableau d'états à n lignes est un codage respectant la contrainte impérative ci-dessus, quelque soit l'automate particulier, dont le tableau d'états à n lignes, considéré.
Néanmoins, l'ensemble des sommets supplémentaires utilisés (X_{i_j}) est en général, redéterminé spécifiquement pour chaque tableau d'états donné.

On ne considèrera dans ce travail que la contribution apportée à l'étude des codages spécifiques. On trouvera dans [26] la contribution au problème du codage universel.

V - NOTATIONS

Au cours de la synthèse d'un automate, nous noterons Y la variable interne, qui est une variable booléenne générale. Soit $Y_t = f(Y_{t-1},E)$ l'équation de la variable interne où Y_t représente la valeur de Y à l'instant t

Y_{t-1} représente la valeur de Y à l'instant t-1.

Pour alléger cette écriture, nous écrirons cette équation sous la forme :

$Y = f(y,E)$ où Y_t est remplacée par Y

Y_{t-1} est remplacée par y.

CHAPITRE - II

ETUDE DES GRAPHES DE FONCTIONNEMENT ASSOCIES A UN AUTOMATE ASYNCHRONE

Nous nous proposons, dans ce chapitre, de rechercher comment traduire le fonctionnement d'un automate asynchrone au moyen d'un graphe ; le problème posé est indépendant, à ce stade, du problème du codage.

I - RECHERCHE DE TELS GRAPHES

I-1 - Ensemble de sommets

On associe bijectivement à l'ensemble des états de l'automate, un ensemble de sommets X; on note x le sommet image de l'état s.

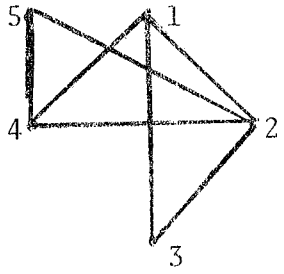
I-2 - Graphe des transitions directes

Une idée élémentaire consiste à associer à chaque transition de l'automate $(s,\underline{s})$, $\underline{s}$ étant un état stable pour une entrée donnée, distinct de s, une arête $(x,\underline{x})$ du graphe ; $\underline{x}$ étant l'image de $\underline{s}$. En considérant toutes les transitions de ce type, pour toutes les valeurs d'entrée, nous obtenons un graphe $G(X,U)$, U étant l'ensemble des arêtes. Ce graphe sera appelé graphe des transitions directes.

Exemple

	E_1	E_2	E_3
1	2	3	4
2	②	4	②
3	2	③	③
4	④	④	④
5	2	4	4

Nous obtenons le graphe :



Propriétés de ce graphe

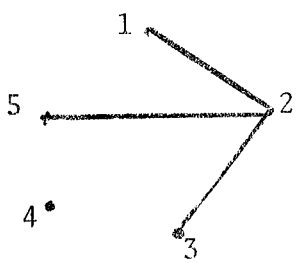
a) Nous définissons ainsi un graphe non orienté ; cette perte apparente d'information est justifiée par le traitement ultérieur ; en effet, nous chercherons une immersion possible dans un n-cube qui est un graphe non orienté.

b) Décomposition de ce graphe

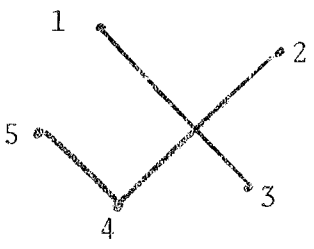
Ce graphe peut être considéré comme la réunion de graphes partiels ou sous-graphes partiels (dans le cas d'automates incomplets) relatifs aux différentes entrées. Un état stable isolé pour l'entrée considérée est représenté par un sommet isolé, une transition pour cette valeur d'entrée par l'arête correspondante.

Exemple précédent :

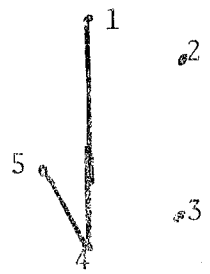
Entrée E_1



Entrée E_2



Entrée E_3

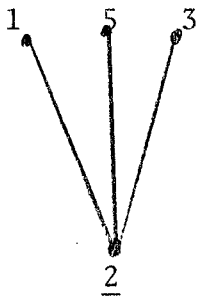


Le graphe (ou sous-graphe) partiel relatif à une entrée donnée peut, à son tour, être considéré comme la réunion de sous-graphes partiels relatifs aux blocs de la partition associée à cette entrée.

Le sous-graphe partiel relatif à un bloc B_{i_j} , ($|B_{i_j}| > 1$), de la partition π_i associée à E_i est un arbre dans lequel le sommet image de l'état stable de B_{i_j} est de degré ($|B_{i_j}| - 1$), les autres sommets étant de degré 1. Cet arbre sera appelé l'arbre des transitions directes associé à B_{i_j}.
Si $|B_{i_j}| = 1$ nous avons vu que cet arbre se réduit à un sommet isolé.

Exemple précédent

Arbre associé à B_{1_1}



Arbre associé à B_{2_2}

Le graphe des transitions directes peut donc être considéré comme la réunion des arbres de transitions directes associées aux blocs B_{i_j}, i décrivant les différentes entrées, j les blocs de la partition relative à l'entrée considérée.

Remarque

On suppose que tout état de l'automate a, au moins, un successeur défini.

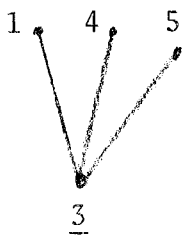
I-3 - Ensemble de graphes traduisant le fonctionnement d'un automate asynchrone

Le graphe des transitions directes n'est pas le seul graphe traduisant le fonctionnement d'un automate asynchrone. La transition d'un état à l'état stable d'un bloc peut se faire, par l'intermédiaire d'autres états de ce bloc.

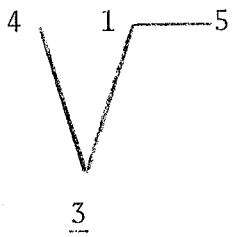
I-3-1 - Ensemble d'arbres A_{i_j} associés à un bloc B_{i_j} d'une partition π_i.

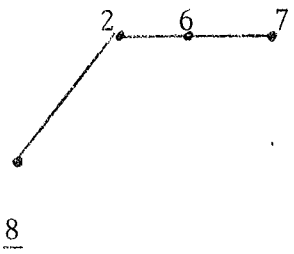
Exemple :

Soit une partition $\pi_i = (1\underline{3}45, 2678)$ d'un automate à 8 états. En s'imposant des transitions directes, on associera à B_{i_1} et B_{i_2} les 2 arbres :



En ne s'imposant pas des transitions directes, on peut avoir les 2 schémas suivants :





La transition de l'état 7 à l'état stable 8 se fait par l'intermédiaire des états 6 et 2 de ce bloc.

En netenant pas compte de l'orientation, l'ensemble des transitions des états d'un bloc B_{i_j} à l'état stable de ce bloc, se traduit, en terme de graphe par un arbre dont les sommets sont les images des états de ce bloc. Il suffit de remarquer que tout état d'un bloc a un successeur et un seul à l'intérieur de ce bloc.

On peut donc associer à un bloc B_{i_j} de n états l'ensemble des arbres de n sommets, ces sommets étant l'image des n états.

Remarque :

Nous avons vu (Chap. I, § I-3) qu'un automate asynchrone peut toujours se ramener à un automate tel que le successeur de tout état soit un état stable. Nous considérons ici la propriété inverse à savoir que dans un automate asynchrone la transition d'un état à l'état stable du bloc correspondant peut se faire par l'intermédiaire d'autres états du bloc.

I-3-2 - Ensemble de graphes traduisant le fonctionnement d'un automate asynchrone

On peut donc plus généralement, associer à un automate asynchrone une famille de graphes $G_i(X,U_i)$; un graphe élément de cette famille est obtenu comme suit :

- à tout état de l'automate est associé bijectivement un sommet de X
- à tout bloc B_{i_j} de la partition π_i relative à l'entrée E_i est associé un arbre $\mathcal{A}_{i_j}$ dont les sommets sont les images des états de B_{i_j} . L'ensemble des arêtes du graphe est la réunion des arbres, obtenus en considérant tous les blocs B_{i_j} .

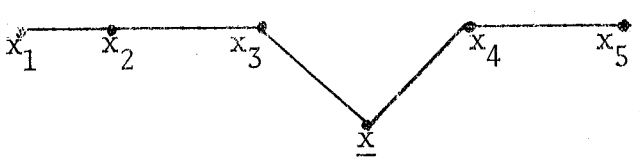
I-4 - Caractéristiques de temps de transition attachées à un arbre $\mathcal{A}_{i_j}$

Considérons l'ensemble des arbres pouvant être associés à B_{i_j}. Tous ces arbres ne présentent pas le même intérêt relatif au fonctionnement de l'automate. En effet soit $\underline{x}$ l'image de l'état stable de bloc. A un état s de ce bloc, ayant comme image x , on peut associer indépendamment du codage ultérieur un temps de transition égal à la distance $(\underline{x},x)$ sur l'arbre. Posons :

$\Sigma_T(\mathcal{A}_{i_j})$ = somme des temps de transition étendue à tous les états du bloc.

$T_{MAX}(\mathcal{A}_{i_j})$ = valeur maximale des temps de transition pour le bloc considéré.

Exemple :



$\Sigma_T = 3+2+1+1+2 = 9$

$T_M = 3$

I-5 - Intérêt d'un arbre par rapport au fonctionnement

On dira qu'un arbre $\mathcal{A}$ associé à B_{i_j} est plus intéressant qu'un arbre $\mathcal{A}'$ si nous avons l'une des 2 possibilités

a) $\Sigma_T(\mathcal{A}) < \Sigma_T(\mathcal{A}')$

b) $\Sigma_T(\mathcal{A}) = \Sigma_T(\mathcal{A}')$ et $T_{MAX}(\mathcal{A}) < T_{MAX}(\mathcal{A}')$

Nous nous proposons donc d'étudier l'ensemble des arbres pouvant être attachés à B_{i_j} en les classant selon l'intérêt défini ici.

II - ETUDE DE L'ENSEMBLE DES ARBRES DE n SOMMETS DANS LEQUEL ON A DISTINGUE UN SOMMET PRIVILEGIE

II-1 - Représentation

Un tel arbre sera représenté par l'arbre des distances relatives au sommet privilégié. Nous aurons un graphe en couches ; la couche 0 sera constituée par le sommet privilégié, la couche i par les sommets de l'arbre à distance i de ce sommet référence.

II-2 - Définition

Un arbre sera dit du type $(p_1)\ (p_2)\ldots(p_q)$ si l'arbre des distances à q couches et si la première couche contient p_1 sommets
la deuxième couche contient p_2 sommets...

Propriété :

2 arbres de n sommets qui sont de même type ont même Σ_T (somme des temps de transition) et même T_{MAX} (temps maximal de temps de transition).

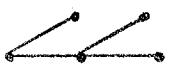
Ceci est évident puisque $\Sigma_T = p_1+2p_2+\ldots qp_q$

$$T_{MAX} = q$$

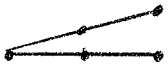
2 arbres de même type présente donc le même intérêt par rapport au codage selon les critères fixés.

Remarque

2 arbres de même type ne sont pas obligatoirement isomorphes.
Exemple :



S



S

II-3 - Etude de ces types d'arbres

Il existe n^{n-2} arbres distincts de n sommets (RIORDAN [25], BERGE [3]).

II-3-1 - Propriété 1

Ces n^{n-2} arbres se rangent en 2^{n-2} types d'arbres précédemment définis. En effet, rappelons que dans l'ensemble des arbres précédemment définis, on a distingué un sommet référence fixée (correspondant à l'état stable du bloc). Les différents types d'arbres de n sommets correspondent aux partitions en une suite ordonnée de blocs des (n-1) sommets distincts du sommet référence (ces partitions sont appelées compositions dans Riordan [25]). Il existe donc 2^{n-2} types d'arbres de n sommets.

II-3-2 - Génération de ces types d'arbres

Il est facile de générer, par récurrence, ces différents types.
Supposons donnée la liste des types d'arbres de n-1 sommets, chaque type étant donné sous la forme $(p_1)\ (p_2)\ldots$
Les types d'arbre de n peuvent s'obtenir à partir des précédents par les 2 opérations suivantes :

α) Ajouter un sommet en couche 1 aux types précédents.

β) Mettre un sommet unique en couche 1 et reproduire à partir de la couche 2 les types précédents.

Exemple

	p_1	p_2	p_3	p_4
n = 4 il existe 4 types	3			
	2	1		
	1	2		
	1	1	1	
n = 5 il existe 8 types	p_1	p_2	p_3	p_4
opération α	4			
	3	1		
	2	2		
	2	1	1	
opération β	1	3		
	1	2	1	
	1	1	2	
	1	1	1	1

De même, on aurait pu considérer les opérations α' et β'

α') Ajouter un sommet en dernière couche existante.

β') Créer un sommet à droite.

Si ces 2 groupes d'opérations engendrent de façon irrédondante les types d'arbres de n sommets, ils ne donnent pas le rangement souhaité de ces types d'arbres à partir du rangement des types d'arbres de n-1 sommets. Nous allons indiquer 2 procédés pour obtenir le rangement souhaité.

Rappel du rangement souhaité

Supposons donnés les types d'arbres de n-1 sommets dans un tableau T_{n-1} Ces types d'arbres sont rangés suivant un intérêt décroissant pour le codage si la condition suivante est remplie :

Soient $\mathcal{C}$ et $\mathcal{C}'$, 2 types figurant dans 2 lignes consécutives de T_{n-1}, nous avons une des 3 possibilités :

a) $\Sigma_T(\mathcal{C}) < \Sigma_T(\mathcal{C}')$

b) $\Sigma_T(\mathcal{C}) = \Sigma_T(\mathcal{C}')$ et $T_M(\mathcal{C}) < T_M(\mathcal{C}')$

c) $\Sigma_T(\mathcal{C}) = \Sigma_T(\mathcal{C}')$ et $T_M(\mathcal{C}) = T_M(\mathcal{C}')$

Premier procédé

Générons les éléments de T_n à partir des éléments de T_{n-1} par les opérations α et β . Chacune de ces opérations génère un sous-tableau de T_n respectant le rangement souhaité. En effet, l'opération α ajoute 1 à Σ_T et conserve T_{MAX}. L'opération β ajoute n-1 à Σ_T et 1 à T_{MAX}.

Exemple

T5

	Σ_T	T_{MAX}
4	4	1
3 1	5	2
2 2	6	2
1 3	7	2
2 1 1	7	3
1 2 1	8	3
1 1 2	9	3
1 1 1 1	10	4

Opération α

	Σ_T	T_{MAX}
5	5	1
4 1	6	2
3 2	7	2
2 3	8	2
3 1 1	8	3
2 2 1	9	3
2 1 2	10	3
2 1 1 1	11	4

Opération β

	Σ_T	T_{MAX}
1 4	9	2
1 3 1	10	3
1 2 2	11	3
1 1 3	12	3
1 2 1 1	12	4
1 1 2 1	13	4
1 1 1 2	14	4
1 1 1 1 1	15	5

Il reste à interclasser ces 2 tableaux selon les critères fixés (Comparaison des T_{MAX} , puis de Σ_T).

Si ce procédé permet de générer les types de façon irrédondante, il oblige néanmoins à interclasser les 2 sous-tableaux. L'algorithme suivant donne immédiatement le rangement souhaité mais ne génère pas de façon irrédondante les types d'arbres.

2ème procédé

Considérons le tableau T_{n-1} ; à chaque type d'arbre est associé la grandeur Σ_T correspondante. Nous allons générer les types de T_n dans un ordre croissant de Σ_T noté X au cours de la procédure $X_{initial} = n-1$ (arbre des transitions directes) ; soit X la valeur de Σ_T considérée.

Nous considérons dans T_{n-1} les types d'arbres dont la grandeur associée Σ_T est inférieure à X . Soit $(p_1)\ (p_2)\ldots(p_q)$ un tel type. Nous ajoutons une unité au dernier (p_i) non coché et cochons le (p_i) correspondant. Le nouveau type est rangé dans T_n s'il n'y figure pas déjà ; il suffit de vérifier les éléments dont la caractéristique $\Sigma_T = X$. Après épuisement des possibilités, on fait $X = X+1$. Le processus est terminé si tous les types de T_{n-1} sont épuisés, en ayant soin d'ajouter le dernier type de T_n à savoir

$\underbrace{(1)\ (1)\ \ldots\ (1)}_{n-1 \text{ termes}}$

X est alors égal à $1+2+3+\ldots+(n-1)$.

Exemple

Soit T4

			Σ_T
3			3
2	1		4
1	2		5
1	1	1	6

Cherchons T5

X = 4 T5 =

	Σ_T
4	4

T4 =

	Σ_T
~~3~~	3
2 1	4
1 2	5
1 1 1	6

X = 5 T5 =

	Σ_T
4	4
3 1	5

T4 =

	Σ_T
~~3~~	3
~~2~~ 1	4
1 2	5
1 1 1	6

X = 6 T5 =

	Σ_T
4	4
3 1	5
2 2	6

T4 =

	Σ_T
~~3~~	3
~~2~~ ~~1~~	4
~~1~~ 2	5
1 1 1	6

X = 7 T5 =

	Σ_T
4	4
3 1	5
2 2	6
1 3	7
2 1 1	7

T4

	Σ_T
~~3~~	3
~~2~~ ~~1~~	4
~~1~~ ~~2~~	5
~~1~~ 1 1	6

etc... finalement on obtient :

T_5 =

	Σ_T	T_{MAX}
4	4	1
3 1	5	2
2 2	6	2
1 3	7	2
2 1 1	7	3
1 2 1	8	3
1 1 2	9	3
1 1 1 1	10	4

On vérifie facilement que l'on obtient le rangement souhaité. En effet, Σ_T est bien considéré d'une façon croissante ; de même T_{MAX} se conserve par passage de T_{n-1} à T_n.

II-3-3 - Nombre d'éléments d'un type donné

Rappelons que le sommet référence fixé de l'ensemble étudié des arbres de n sommets est image de l'état stable du bloc des états considéré. Aux autres sommets pourront donc être affectés les états restants du bloc.

Dans l'ensemble des arbres de n sommets du type $(p_1)\ (p_2)\ \ldots\ (p_r)$ avec $p_1 + \ldots + p_r = n-1$

$p_o = 1$

nous aurons donc $\prod_{i=1}^{r} C_{n-\sum_{0}^{i-1} p_j}^{p_i} \cdot p_{i-1}^{p_i}$ éléments distincts.

Il suffit de remarquer que

- Ayant affecté les éléments des couches $1,\ldots,i-1$, nous aurons $C^{p_i}_{n-p_o-\ldots-p_{i-1}}$ combinaisons possibles dans la $i^{ème}$ couche.
- Que les possibilités de connexions reliant chaque sommet de la $i^{ème}$ couche à un sommet, au choix, de la $(i-1)^{ième}$ couche sont au nombre de $p_{i-1}^{p_i}$.

Cette expression peut se simplifier en remarquant qu'elle peut encore s'écrire

$$\prod_{i=1}^{r} \frac{(p_i+p_{i+1}+\ldots p_r)!}{p_i!(p_{i+1}+\ldots+p_r)!}\ p_{i-1}^{p_i}$$

soit en explicitant ce produit

$$(n-1)!\ \prod_{i=1}^{r} \frac{p_{i-1}^{p_i}}{p_i!}$$

Ce rappel combinatoire nous indique que nous serons rapidement devant un grand nombre de solutions et qu'il ne sera certainement pas intéressant de générer tous les arbres possibles.
Nous allons néanmoins indiquer les applications immédiates suivantes :

III - APPLICATIONS

III-1 - Génération de tous les graphes possibles traduisant le fonctionnement d'un automate asynchrone.

Soient $B_1, B_2, \ldots B_r$ la liste des blocs des partitions π_i relatives aux différentes entrées E_i. Nous excluons de cette liste les blocs comprenant tous les états de l'automate, d'une façon générale, de tels blocs, ne posant aucun problème relatif à la contrainte impérative du codage, ne sont pas considérés dans de telles études.

Nous avons vu qu'à un bloc B_i , on peut associer l'un parmi les n_i arbres ($n_i = |B_i|^{(|B_i|-2)}$), ces arbres étant rangés suivant un intérêt décroissant. On notera ce choix possible par une somme booléenne ; chaque arbre sera représenté par un monôme booléen dans lequel figurent les arêtes de cet arbre sous forme de variables booléennes. Tous les graphes de fonctionnement de l'automate considéré, sont alors donnés par le produit booléen

$$\prod_{i=1}^{r} \left(\sum_{j=1}^{n_i} \mathcal{R}_{ij} \right)$$

Rappel : dans un produit booléen, nous avons $x.x = x$.

Exemple :

Soit l'automate :

	E_1	E_2	E_3
1	4	2	3
2	5	②	4
3	5	5	③
4	④	5	④
5	⑤	⑤	4

Appelons les arêtes, comme suit

	1	2	3	4	5
1		a_1	a_2	a_3	a_4
2			a_5	a_6	a_7
3				a_8	a_9
4					a_{10}
5					

les graphes cherchés sont donnés par chaque terme de l'expression :

$$\prod_{j=1}^{r}\left(\sum_{j=1}^{n_i} \mathcal{A}_{ij}\right) = a_1 \cdot a_2 \cdot a_3 (a_7a_9+a_5a_7+a_5a_9)(a_9a_{10}+a_8a_{10}+a_8a_9)(a_6a_{10}+a_6a_7+a_7a_{10}).$$

Remarque :

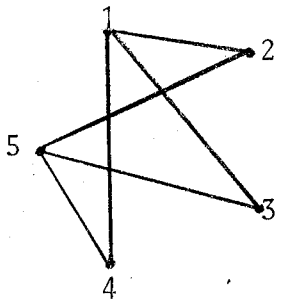
Les blocs de 2 éléments sont représentés par une arête unique et placée en tête de cette somme.

III-2 - Graphes de fonctionnement d'un automate asynchrone ayant un nombre minimal d'arêtes.

Il suffit, dans les termes effectués ci-dessus, de choisir ceux qui sont un nombre minimal de lettres.

Exemple :

Dans l'exemple, ci-dessus nous trouvons un terme minimal à savoir $a_1a_2a_3a_7a_9a_{10}$ correspondant au graphe de 6 arêtes :



III-3 - Choix d'un graphe minimal (défini au § ci-dessus) ayant des caractéristiques optimales de temps de transition.

Supposons que nous ayons trouvé plusieurs graphes ayant un nombre minimal d'arêtes. On peut chercher parmi ces solutions, une solution donnant de bonnes performances de temps de transition. Considérons un tel graphe il est la réunion des arbres $(\mathcal{A}_j)$ associés à chaque bloc B_j .

Soit $\Sigma_T(\mathcal{A}_j)$ et $T_{MAX}(\mathcal{A}_j)$ les 2 caractéristiques temps de transition associées à $\mathcal{A}_j$.

On peut associer à chaque arbre T_{MAX} et un temps moyen $= \frac{\Sigma_T}{\text{nombre de sommets}}$

Au graphe sera alors associé les caractéristique : $\underset{\mathcal{A}_j}{\text{Max}}$ (temps moyen) $\underset{\mathcal{A}_j}{\text{Max}}$ (T_{MAX}). On pourra ainsi définir si une solution trouvée est plus intéressante qu'une autre, compte tenu des temps de transition.

IV - PONDERATION DES GRAPHES DE FONCTIONNEMENT ATTACHES A UN AUTOMATE ASYNCHRONE

IV-1 - Définitions préliminaires

Soit une entrée E_i d'un automate asynchrone. Considérons un graphe de fonctionnement de cet automate (défini ci-dessus).

Une arête (x,x') correspondant aux états (s,s') est dite arête obligatoire pour cette entrée si (s,s') constituent un bloc de 2 éléments de la partition π_i. Une arête (x,x') est dite arête directe si s ou s' est un état stable pour cette entrée.

Une arête (x,x') est dite arête indirecte si s et s' appartiennent à un même bloc de π_i et sont tous deux distincts de l'état stable de ce bloc.

IV-2 - Pondération

Soit un graphe de fonctionnement associé à un automate asynchrone. On associe à une arête a un poids p. défini comme suit :

soit n le nombre d'entrées pour lesquelles cette arête est obligatoire.

soit q le nombre d'entrée pour lesquelles cette arête est directe sans être obligatoire.

soit r le nombre d'entrées pour lesquelles cette arête est indirecte.

On pose $p = \alpha n+\beta q+\gamma r$ où α, β, γ sont des coefficients de pondération tels que $\alpha > \beta > \gamma > 0$.

V - GRAPHE DES TRANSITIONS DIRECTES D'UN AUTOMATE ASYNCHRONE ET REDUCTION DE CET AUTOMATE

V-1 - Tableau d'états non réduit

Un tableau d'états d'automate asynchrone non réduit possède la propriété suivante : il existe un état stable et un seul dans une ligne de ce tableau. D'autre part, au cours de l'écriture de ce tableau, on s'impose un codage booléen des variables d'entrée tel que 2 valeurs successives des valeurs d'entrée ne diffèrent que d'une composante. On en déduit la propriété suivante :

PROPRIETE

Le graphe des transitions directes d'un automate asynchrone non réduit ayant un codage acceptable des variables d'entrée est bipartite.

Il suffit de remarquer que toute arête d'un cycle correspond à une variation des valeurs des variables d'entrée.

V-2 - Réduction des tableaux d'états

Il est intéressant de distinguer 2 types d'équivalence entre les états d'un automate asynchrone non réduit.

a) L'équivalence entre 2 états ayant même valeurs d'entrées (apparaissant dans la même colonne du tableau d'états).

b) L'équivalence entre 2 états ayant des valeurs d'entrée différentes.

En effet, après réduction du nombre des états par l'équivalence du premier type, le tableau d'états ne perd pas les propriétés mentionnées au paragraphe V-1. Ce qui n'est plus obligatoirement vrai pour l'équivalence du 2ème type. (La réduction de ce 2ème type est plus spécifiquement appelée "fusionnement" des lignes d'un tableau d'états).

La propriété d'être bipartite étant nécessaire en vue du traitement ultérieur (codage de ces graphes), il convient de ne procéder qu'aux réductions du premier type.

PRELIMINAIRES AUX CHAPITRES III et IV

Dans ces préliminaires aux chapitres III et IV, nous définirons essentiellement les représentations de graphes adoptées. Nous définirons, dans un premier temps, la représentation choisie d'un n-cube et d'un sous-graphe partiel d'un graphe d'un n-cube. Nous définirons dans un deuxième temps l'immersion cherchée d'un graphe connexe dans un n-cube, et enfin, nous définirons la représentation choisie d'un graphe connexe.

I - GRAPHE D'UN n-CUBE

I-1 - Représentation habituelle

Un n-cube est, en général, considéré comme la représentation des 2^n valeurs d'un vecteur booléen à n composantes.
Le graphe $C_n(X,\Gamma)$, où X est l'ensemble des sommets, Γ la relation d'adjacence, associé au n-cube est défini comme suit :

- à chaque valeur du vecteur booléen est associé bijectivement un sommet de X.

- $x,x' \in X$ et $\Gamma(x,x')$ <=> les valeurs du vecteur booléen associées à x et x' diffèrent d'une composante et d'une seule.

Propriétés de ce graphe

- C'est un graphe bipartite et homogène ; les 2^n sommets sont de degré n .

- Etant donné un graphe $C_n(X,\Gamma)$, il existe $2^n.n!$ façons différentes d'associer à l'ensemble des sommets X les valeurs du vecteur booléen à n composantes. A un sommet quelconque du graphe est associé le vecteur (0,0,...).
Il existe n! façons d'associer aux n sommets adjacents les n valeurs adjacentes de ce vecteur.
La valeur du vecteur booléen associée à un sommet du graphe est appelé coordonnées de ce sommet.
L'ensemble des affectations de coordonnées aux sommets de $C_n(X,\Gamma)$ défini ci-dessus est appelé ensemble de représentations équivalentes du n-cube.

I-2 - Représentations relatives

I-2-1 - Indice de direction d'une arête d'un n-cube

Soit une représentation habituelle d'un n-cube. Nous dirons qu'une arête de ce graphe est de direction ρ si les vecteurs booléens associés aux extrémités diffèrent par la ρ ième composante.

I-2-2 - Ensemble caractéristique d'indices de direction d'un couple de sommets

Soient 2 sommets x et x' d'une représentation habituelle d'un n-cube. Associons à chaque arête son indice de direction. On note $\varepsilon_c(x,x') = \{\alpha_1,\ldots,\alpha_i\}$ l'ensemble des indices de direction figurant en nombre impair sur une chaîne (x,x').

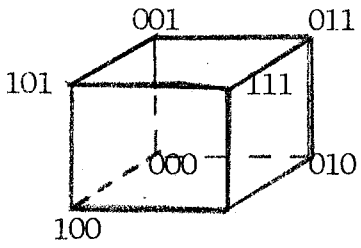
- $\varepsilon_c(x,x')$ ne dépend pas de la chaîne considérée. Cet ensemble peut se définir à partir des coordonnées de x et x' : les composantes du vecteur disjonction des vecteurs booléens associés à x et x' valent 1 dans les positions $\alpha_1,\alpha_2,\ldots,\alpha_i$.

- $|\varepsilon_c(x,x')|$ est la distance de x et x' sur le n-cube. Cette distance est égale au nombre de composantes dont diffèrent les coordonnées de x et x'.
$\varepsilon_c(x,x')$ sera appelé ensemble caractéristique d'indices de direction du couple de sommets (x,x').

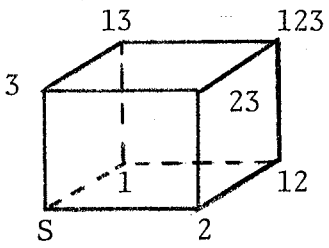
I-2-3 - Coordonnées relatives des sommets d'un n-cube

Soit S un sommet quelconque du n-cube (le graphe du n-cube étant un graphe régulier ce choix n'implique aucune restriction). Associons à chaque sommet x l'ensemble $\mathcal{C}_c(S,x) = \{\alpha_1,\ldots,\alpha_i\}$ appelé coordonnées relatives de x par rapport à S.

Exemple :



peut être représenté par



I-2-4 - Relation d'adjacence en terme de coordonnées relatives

Cette relation s'écrit :

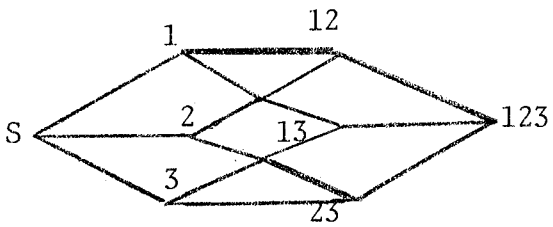
$$x,x' \in X \text{ et } \Gamma(x,x') \iff |\mathcal{C}_c(S,x) \oplus \mathcal{C}_c(S,x')| = 1$$

$\oplus$ étant la disjonction ensembliste habituelle.

I-2-5 - Représentation en treillis de Boole

La représentation relative précédente peut aussi être considérée comme le diagramme d'un treillis de Boole dont S est le minimum. Ce diagramme comporte n+1 niveaux, le ième niveau comportant C_n^i éléments, appelés éléments de rang i .

Exemple :



rappelons que ce treillis est isomorphe au treillis des sous-ensembles de l'ensemble de n éléments ; le sous-ensemble attaché à un sommet étant les coordonnées relatives par rapport à S. La relation d'ordre se traduit par

$$x < x' \iff \varepsilon_c(S,x) \subset \varepsilon_c(S,x')$$

1-2-6 - Représentations équivalentes du n-cube

A partir d'une représentation relative d'un n-cube, on génère les $2^n.n!$ représentations équivalentes par les 2 opérations suivantes :

α) On prend successivement tous les sommets du graphe comme sommet de référence S.

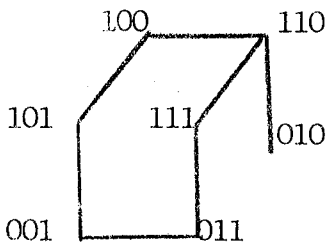
β) Pour un sommet S choisi, on permute entre eux les n indices de directions qui constituent les coordonnées relatives des sommets de rang 1.

II - SOUS-GRAPHE PARTIEL CONNEXE D'UN n-CUBE

II-1 - Représentation habituelle

Considérons un sous-graphe partiel connexe de $C_n(X,\Gamma)$
Les vecteurs booléens associés satisfont à la relation suivante qui n'est plus cette fois, une relation d'équivalence.
$x,x' \in X$ et $\Gamma(x,x')$ => les vecteurs associés diffèrent d'une composante et d'une seule.

Exemple :



II-2 - Représentations relatives

En choisissant un sommet S arbitraire du sous-graphe partiel connexe, la remarque précédente se traduit par la relation

$$x,x' \in X \text{ et } \Gamma(x,x') \Rightarrow |\varepsilon_c(S,x) \oplus \varepsilon_c(S,x')| = 1$$

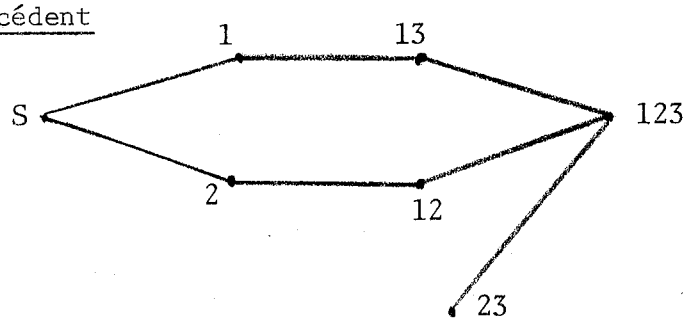
Les représentations équivalentes sont obtenues comme précédemment (opérations α et β du § I.2.6).

II-3 - Représentations ordonnées

Un sommet arbitraire S étant considéré comme minimum, on peut faire correspondre le diagramme d'un ensemble ordonné avec la relation

$$x \lessdot x' \text{ (x' couvre x)} \Leftrightarrow [\Gamma(x,x') \text{ et } \varepsilon_c(S,x) \subset \varepsilon_c(S,x')]$$

Exemple précédent

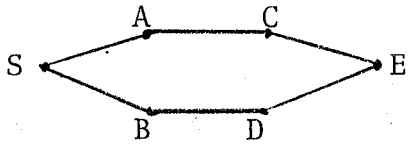


Cet ensemble ordonné n'est plus en général un treillis.

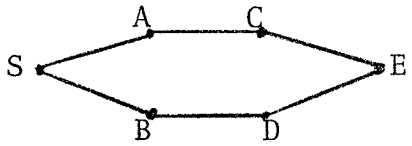
Remarques

α) Considérons un sous-graphe partiel de $C_n(X,\Gamma)$, sur lequel ne figurent pas les coordonnées des sommets x. On peut alors faire correspondre en général plusieurs diagrammes d'ensembles ordonnés.

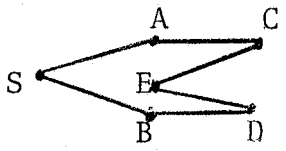
Exemple : Au graphe



peut correspondre, par exemple,



ou



ce problème sera repris dans le chapitre suivant.

β) En d'autres termes un graphe $C_n(X,\Gamma)$ définit biunivoquement un n-cube. Les différentes affectations de coordonnées possibles aux sommets ont été définies comme équivalentes. A un sous-graphe partiel de $C_n(X,\Gamma)$ peut être affectée, en général, une famille d'ensembles de coordonnées sur laquelle on peut définir des classes d'équivalentes distinctes.

III - DEFINITIONS GENERALES DE L'IMMERSION ETUDIEE

III-1 - Soit un graphe connexe $G(X,\Gamma)$ et le graphe $C_n(X',\Gamma')$ du n-cube. Immerger ce graphe G dans un n-cube revient à chercher une application injective de X dans X' respectant la relation d'adjacence.

Soit $x \in X$

l'application I cherchée est telle que

$$x,x' \in X \qquad x \neq x' \iff I(x) \neq I(x')$$

$$\Gamma(x,x') \Rightarrow \Gamma'(I(x), I(x'))$$

Plus simplement on cherche un sous-graphe partiel de $C_n(X',\Gamma')$ isomorphe à $G(X,\Gamma)$.

III-2 - Immersion en terme de coordonnées relatives

Soit S un sommet arbitrairement distingué de X.
L'immersion précédente peut alors être définie comme la recherche d'une application

$$x \to \varepsilon_c(S,x) \quad \text{telle que}$$

$$x,x' \in X, \quad x \neq x' \iff \varepsilon_c(S,x) \neq \varepsilon_c(S,x')$$

$$\Gamma(x,x') \Rightarrow \quad |\varepsilon_c(S,x) \ominus \varepsilon_c(S,x')| = 1$$

IV - REPRESENTATION DES GRAPHES CONNEXES

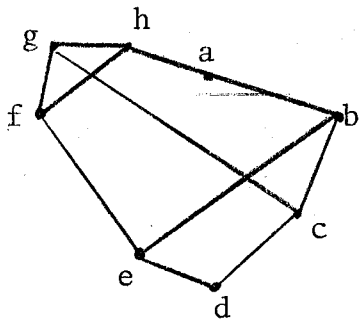
IV-1 - Représentation en couches associée à un graphe connexe

Soit un graphe connexe $G(X,\Gamma)$. Soit un sommet S de X arbitrairement choisi. Ce sommet sera appelé sommet source. Considérons la partition de X en une suite ordonnée de classes $C_1,C_2,\ldots,C_i,\ldots$ telle que $x \in C_i \iff d(S,x) = i$; $d(S,x)$ est la distance (S,x) au sens habituel à savoir la longueur de la plus courte chaîne (S,x) de G .
C_i est appelé ième couche de la représentation.
On vérifie facilement que nous avons bien une relation d'équivalence
$x_i \equiv x_j \iff d(S,x_i) = d(S,x_j)$.

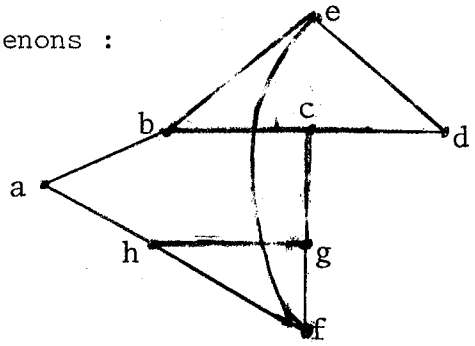
Le graphe sera alors représenté en faisant figurer sur des verticales successives les sommets appartenant respectivement à $C_1,C_2,\ldots$ Sur une même verticale, les sommets d'une même couche sont représentés dans un ordre arbitraire.
Cette construction est tout-à-fait semblable à la construction de l'arbre des distances au sommet S du graphe.

Exemple

Soit le graphe



si nous prenons a comme sommet source nous obtenons :



C_1 = (b,h)

C_2 = (e,c,g,f)

C_3 = (d)

avec la représentation suivante :

IV-2 - Propriétés d'une représentation en couches associée à un graphe connexe

S étant le sommet source, nous avons les propriétés suivantes :

. Pour tout $x \in C_i$, il existe une chaîne $(S,\alpha_1,\alpha_2,\ldots,\alpha_{i-1},x)$ où $\alpha_q \in C_q$ $(q < i)$

. Toute arête a ses extrémités, soit dans la même couche, soit dans des couches voisines.

. Tout cycle de longueur impaire contiendra au moins une arête dont les extrémités appartiennent à une même couche ; dans l'exemple précédent, il existe 4 cycles de longueur impaire : ebahF, cbahg, ghF, eFgcd.

. Un graphe est bipartite si et seulement si, pour un sommet S choisi arbitrairement, la représentation en couches correspondante ne contient pas d'arête dont les extrémités sont dans une même couche. (Théorème de König).

IV-3 - Remarque et applications

. pour supprimer les cycles de longueur impaire, il suffit d'ajouter un sommet supplémentaire sur toute arête dont les extrémités appartiennent à une même couche.

. en prenant tous les sommets du graphe comme sommet source, on obtient chaque fois une solution, c'est-à-dire un ensemble d'arêtes à dédoubler pour supprimer les cycles de longueur impaire.

CHAPITRE- III

ALGORITHME D'IMMERSION D'UN GRAPHE CONNEXE DANS UN n-CUBE

Soit un graphe connexe $G(X,\Gamma)$ que nous cherchons à immerger dans un n-cube. Un sommet source S étant arbitrairement choisi, ce graphe sera représenté en couches. Si dans cette représentation il existe une arête ayant ses extrémités dans une même couche, nous pouvons conclure à la non existence d'une solution cherchée. En effet (§ III.2 des préliminaires) $G(X,\Gamma)$ est non bipartite et ne peut être isomorphe à un sous-graphe partiel d'un n-cube (§II.1 des préliminaires). Supposons éliminés de tels graphes.

Nous cherchons les immersions de $G(X,\Gamma)$ en termes de coordonnées relatives, le sommet source S étant choisi sommet référence pour les coordonnées relatives.

Les immersions possibles, non équivalentes, de $G(X,\Gamma)$ seront générées en procédant couche par couche. Plus précisément, étant donnée une immersion possible pour les j premières couches, nous génèrerons les immersions possibles pour la (j+1)ème couche.

Dans un deuxième temps, nous en déduirons un algorithme pratique cherchant une solution possible.

I - GENERATION DE TOUTES LES IMMERSIONS POSSIBLES

I-1 - Rappel

Soient $1,2,\ldots n$ les n indices de direction du cube considéré. Soit un sommet x, dont l'ensemble des coordonnées relatives $\mathcal{E}_c(S,x)$ est fixé. Si x' est un sommet adjacent à x dans $G(X,\Gamma)$, l'ensemble des solutions possibles pour $\mathcal{E}_c(S,x')$ est l'ensemble $\{ \mathcal{E}_c(S,x) \oplus k\}$ où k décrit les valeurs $1,2,\ldots,n$. Cette famille de solutions possibles pour x' en tenant compte de la relation d'adjacence avec x sera notée $P(x'|x)$.

Exemple $n = 4$

$$\mathcal{E}_c(S,x) = \{1,2\}$$

$$\Gamma(x,x') \Rightarrow P(x'|x) = [\{2\}, \{1\}, \{1,2,3\}, \{1,2,4\}]$$

Notations

Pour des raisons de simplicité, nous noterons $\mathcal{E}_c(S,x) = \{\alpha_1,\ldots,\alpha_i\}$ par la concaténation $\alpha_1\ldots\alpha_i$ et appelerons un tel ensemble, coordonnées de x. Dans l'exemple précédent, nous aurons donc $P(x'|x) = \{2,1,123,124\}$ comme ensemble de coordonnées possibles pour x'.

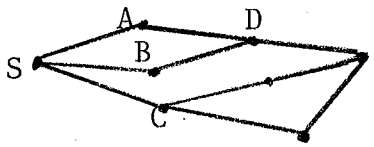
I-2 - Généralisation

Soit un sommet $x \in X$; soient $x_1,\ldots,x_n$ les sommets de G, adjacents à x, dont les coordonnées sont fixées. Alors l'ensemble des coordonnées possibles pour x, en tenant compte de sa relation d'adjacence avec $x_1,\ldots,x_n$ sera

$$P(x|x_1,\ldots,x_n) = P(x|x_1) \cap P(x|x_2) \ldots \cap P(x|x_n)$$

Le symbole $\cap$ est l'intersection ensembliste habituelle, un élément de $P(x|x_1)$ est une solution possible, rappelons le, pour $\mathcal{E}_c(S,x)$

exemple : n = 4



Supposons que $\mathcal{E}_c(S,A) = 1$

$\mathcal{E}_c(S,B) = 2$

$\mathcal{E}_c(S,C) = 3$

$P(D|A) = \{12,13,14\}$ en omettant de cet ensemble l'élément vide correspondant aux coordonnées de S.

$P(D|B) = \{12,23,24\}$

$P(D|A,B) = \{12,13,14\} \cap \{12,23,24\} = \{12\}$

Remarques

. $P(x|x_1)$ est de cardinalité égale à n.

. $P(x|x_1,x_2)$ est vide ou contient 2 éléments. En effet, si une solution existe, les 2 sommets correspondants sont respectivement la borne supérieure et la borne inférieure de x_1 et x_2 en terme de treillis de Boole.

. $P(x|x_1,x_2,\ldots,x_m)$ avec $m > 2$ contient, au plus, un élément. Le sommet correspondant est alors la borne supérieure des éléments $x_1,\ldots x_m$. Cet élément est unique.

. Les éléments de $P(x|x_1\ldots)$ sont des ensembles d'indices de direction ; 2 éléments distincts de P, p_i et p_j , s'ils existent, considérés comme des ensembles d'indices de direction sont tels que $|\{p_i \Theta p_j\}| = 2$; en d'autres termes, p_i et p_j diffèrent par 2 indices de direction. Le couple (p_i,p_j) peut être de 2 types :

Soit $\begin{cases} p_i = \alpha_1 \ldots \alpha_k \; a \\ p_j = \alpha_1 \ldots \alpha_k \; b \end{cases}$ soit $\begin{cases} p_i = \alpha_1 \ldots \alpha_k \\ p_j = \alpha_1 \ldots \alpha_k \; ab \end{cases}$

où α_i , a, b sont des indices de direction ; le sous-ensemble $\alpha_1 \ldots \alpha_k$ peut être vide.

Exemple :

$$\varepsilon_c(S,x_1) = 1 \qquad P(x|x_1,x_2) = \{\phi,12\}$$

$$\varepsilon_c(S,x_2) = 2$$

I-3 - Algorithme d'immersion

I-3-1 - Immersion de la première couche

Le nombre de sommets de la première couche est supposé inférieur ou égal à n. On affecte respectivement $1,2,\ldots,j,\ldots$ comme coordonnées à ces sommets.

I-3-2 - Immersion de la jème couche

Soit une immersion donnée des (j-1) premières couches. Soit x un sommet de la jème couche et x_1 , $x_2,\ldots$ les sommets de la (j-1)ème couche adjacents à x. $P(x|x_1,x_2\ldots)$ est l'ensemble des coordonnées possibles pour x. Notons $P(x|x_1,x_2\ldots) = \{p_1,p_2,\ldots p_r\}$ cet ensemble. Associons à x la somme des variables $x_{p_1}+x_{p_2}+\ldots$
On associera de même au sommet y de la couche C_j la somme des variables $y_{p_1'}+y_{p_2'}$ ... Les immersions de la jème couche sont alors données par le produit des sommes relatives à chaque élément de la couche

$$\prod_{x,y\ldots\in C_j} (x_{p_1}+x_{p_2}+\ldots)(y_{p_1'}+y_{p_2'}+\ldots)(\ldots$$

Ce produit a la propriété habituelle de distributivité par rapport à la somme et possède en outre la propriété suivante :

Soient x et j, 2 sommets distincts de C_j

$$x_{p_i} \; y_{p_j} = 0 \qquad \text{si} \quad p_i \equiv p_j \qquad (1)$$

(injectivité de l'immersion ; p_i et p_j sont des sous-ensembles de l'ensemble des indices de direction).

En posant $X = \{x_{p_1}, x_{p_2} \ldots\}$ $\quad Y = \{y_{p_1'}, y_{p_2'} \ldots\}$ $\quad Z = \ldots.$

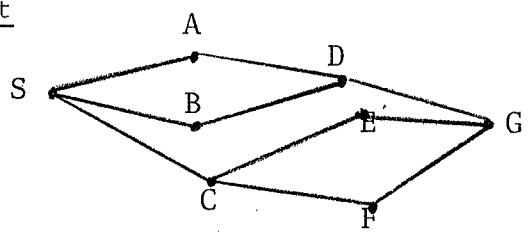
On peut encore dire que les solutions seront l'ensemble des éléments non nuls (règle (1)) de $X \times Y \times Z$ ...

I-3-3 - Immersion des j premières couches

Pour une immersion donnée des (j-1) premières couches, on fait le produit π de cette immersion par les immersions possibles de la jème couche.

Nous avons donc un ensemble de choix ayant la structure classique d'une arborescence.

Exemple complet



immersion de la première couche : $A_1B_2C_3$

immersion de la 2ème couche :

$$P(D|A,B) = \{12,13,14\} \cap \{12,23,24\} = \{12\}$$

$$P(E|C) = \{31,32,34\}$$

$$P(F|C) = \{31,32,34\}$$

les immersions de la 2ème couche sont données par

$$D_{12}(E_{31}+E_{32}+E_{43})(F_{31}+F_{32}+F_{34})$$

soit 6 immersions

$$A_1B_2C_3D_{12}E_{31}F_{32}+A_1B_2C_3D_{12}E_{31}F_{34}$$
$$+ \ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

immersion de la 3ème couche

Appelons $C_1, C_2, \ldots C_6$ les 6 immersions trouvées des 2 premières couches.

Immersion C_1 :
On trouve : $P(G|D,E,F) = \{2,1,123,124\} \cap \{3,123,134\} \cap \{123,3,2,234\}$
$= \{123\}$

Immersion C_2 :
On trouve $P(G|D,E,F) = \{2,123,124\} \cap \{3,123,134\} \cap \{134,234,4,3\}$
= ensemble vide
l'intersection étant vide signifie qu'il n'y a pas de codage possible pour G.

Immersion C_3 :
On trouve $P(G|D,E,F) = \{123\}$

Les immersions C_4, C_5, C_6 ne donnent aucune solution pour G.

immersions du graphe

Nous trouvons donc 2 solutions, sous forme de la somme :

$$A_1B_2C_3D_{12}E_{31}F_{32}G_{123} +$$
$$A_1B_2C_3D_{12}E_{32}F_{31}G_{123}$$

I-3-4 - Existence

Si aucune solution n'est trouvée, le graphe n'est pas immergeable dans le n-cube considéré.

II - ELIMINATION DES IMMERSIONS EQUIVALENTES

II-1 - Rappel

Nous avons vu que 2 immersions équivalentes, en termes de coordonnées relatives se déduisaient l'une de l'autre par les 2 opérations suivantes :

α) en changeant de sommet référence S

β) en permutant entre eux les indices de direction, pour un sommet référence S donné.

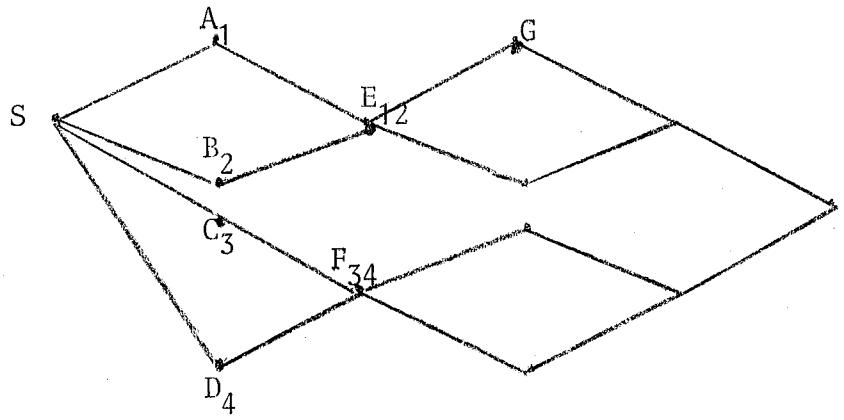
On en conclut :

II-2 - Proposition 1

Soit n *la dimension du cube dans lequel on réalise les immersions. Soit* N_1 *le nombre de sommets de la couche* C_1 *du graphe* $G(X,\Gamma)$ *considéré. Si* $N_1 = n$ *l'ensemble des immersions générées par l'algorithme précédent ne contient pas d'éléments équivalents.*

Il suffit de remarquer que le sommet S a été fixé dans la représentation en couches de $G(X,\Gamma)$ et que les n indices de direction ont été associés, bijectivement, d'une façon unique aux éléments de C_1 ; les permutations de ces indices sont donc éliminées.

Exemple :



Les 2 choix pour $\varepsilon_c(S,G)$ à savoir 123 et 124 donnent 2 immersions non équivalentes, car les indices 3 et 4 ne peuvent être permutés dans C_1.

II-3 - $N_1 < N$

Au cours de l'algorithme précédent, considérons l'ensemble des immersions possibles pour les j-1 premiers sommets (produit effectué des sommes relatives à ces sommets).

Nous allons montrer comment, à partir d'une immersion donnée des j-1 premiers sommets, on génère un ensemble d'immersions non équivalentes pour les j sommets.

Soit x_j, le jème sommet que nous voulons immerger et supposons donnée une immersion des sommets S, $x_1, x_2, \ldots, x_{j-1}$. Soit $\pi_j = \{(1,\ldots,k), (k+1,\ldots,n)\}$ la partition des indices en 2 blocs telle que

- les indices du premier bloc sont les indices de direction déjà utilisés pour l'immersion des sommets $x_1,\ldots,x_{j-1}$,

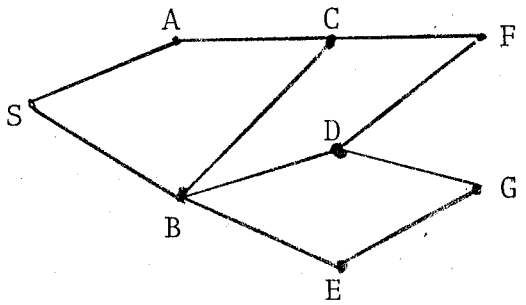
-les indices du 2ème bloc sont les indices non encore utilisés.

PROPOSITION 2

> *Si au cours de l'immersion de* x_j *nous remplaçons l'expression* $P(x_j|x_r)$ *par* $P^*(x_j|x_r) = \{\mathcal{E}_c(S,x_r) \ominus q\}$ *où* q *décrit maintenant le sous-ensemble* 1,2,...,k,k+1, *nous ne génèrerons pas d'immersions équivalentes pour les* j *sommets à partir de l'immersion considérée des* (j-1) *premiers sommets.*

En effet, l'expression $P^*(x_j\ x_r)$ évite de générer les immersions équivalentes obtenues par permutation des indices k+1,k+2,...,n.

Exemple : n = 4



l'algorithme initial donne

$$S_\emptyset A_1 B_2 C_{12} (D_{23}+D_{24})$$

et nous amènera aux 2 immersions équivalentes aux permutations de 3 et 4 près

$$S_{\emptyset}A_1B_2C_{12}(D_{23}+D_{24})(E_{23}+E_{24}) =$$

$$S_{\emptyset}A_1B_2C_{12}D_{23}E_{24}G_{234} +$$

$$S_{\emptyset}A_1B_2C_{12}D_{24}E_{23}G_{234}$$

Le théorème 2 nous donne $\pi_0 = \{12,34\}$
d'où

$$P^*(D|B) = 23$$

soit l'immersion unique

$$S_{\emptyset}A_1B_2C_{12}D_{23}E_{24}G_{234}.$$

II-4 - Conclusion

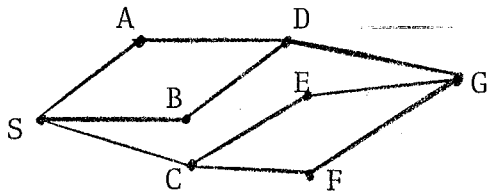
Si, au cours de l'algorithme d'immersion, on tient compte de la proposition 2, l'ensemble des immersions générées ne contient pas d'éléments équivalents.

III - RECHERCHE D'UNE IMMERSION POSSIBLE

III-1 - Il est intéressant en pratique, de chercher non pas toutes les immersions possibles mais de savoir le plus rapidement possible si le graphe est immergeable et de déterminer une immersion possible.

A partir du même algorithme, on recherche le premier terme complet de la somme précédente, donnant toutes les immersions. Cela suppose la mémorisation à chaque étape des choix possibles (noeud de l'arête de l'arborescence des solutions). Il faut se reporter au dernier choix effectué (ou au dernier noeud rencontré de l'arborescence des solutions) en présence d'un terme ne donnant pas de solution et recommencer le calcul. A chaque noeud ne seront considérés que les choix non équivalents.

Exemple



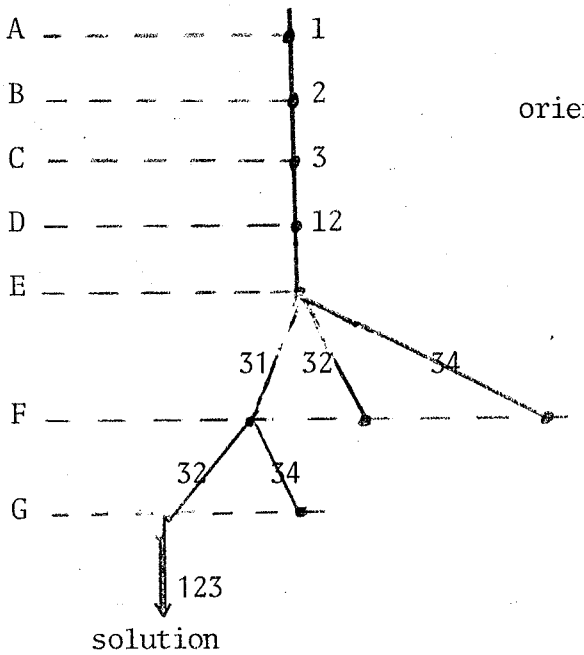
Nous obtenons

$S_{\emptyset}A_1B_2C_3D_{12}(E_{31}+E_{32}+E_{34})$ choix mémorisé

$S_{\emptyset}A_1B_2C_3D_{12}E_{31}(F_{32}+F_{34})$ choix mémorisé

$S_{\emptyset}A_1B_2C_3D_{12}E_{31}F_{32}G_{123}$ calcul arrêté

l'arborescence des choix pourrait se représenter de la façon suivante :



orientation du haut vers le bas

A chaque sommet du graphe est associé un niveau de l'arborescence, comprenant une ou plusieurs images. Sur les arcs issus d'un sommet donné, figure les différents choix possibles non équivalents pour les coordonnées du sommet correspondant à ce niveau. En cas d'impossibilité, il faut remonter au dernier noeud d'où partent des chemins non encore explorés. Le demi-degré extérieur d'un sommet x est égal au nombre d'éléments de $P(x|x_1\ldots)$.

III-2 - Accélération du processus

Nous allons examiner dans quels cas nous avons retour en arrière au dernier choix effectué. Dans chaque cas nous chercherons une accélération du processus qui consiste à retourner non pas au dernier choix effectué mais au choix antérieur pouvant conduire à une solution.

Supposons qu'au cours de la recherche d'une immersion possible, un sommet x soit impossible à immerger. Nous avons 2 cas d'impossibilité :

III-2-1 - $P(x|x_1,\ldots,x_m)$ est vide ; (dans ce cas $n \geq 2$)

Le processus de retour en arrière aux choix précédents doit se faire à partir de x_m , x_m étant le dernier sommet immergé, adjacent à x. Tout choix postérieur à celui-ci ne change rien à l'ensemble $P(x|x_1,\ldots,x_m)$.
Cette remarque accélère considérablement le processus dans le cas de graphes importants.

III-2-2 - $P(x|x_1,\ldots,x_m)$ est non vide mais ses éléments ont déjà utilisés pour l'immersion de sommets considérés avant x. Rappelons que

α) Si $m > 2$, cet ensemble contient un élément et un seul

β) Si $m = 2$ cet ensemble contient 2 éléments.

γ) Si $m = 1$ cet ensemble a n éléments.

Soient $(x_{j_1}, x_{j_2}, \ldots)$ les sommets antérieurement codés dont les coordonnées sont les éléments de $P(x|x_1 \ldots)$.
Le processus de retour en arrière doit se faire à partir du dernier sommet immergé parmi les sommets $(x_{j_1}, x_{j_2}, \ldots, x_1, x_2, \ldots)$
Tout choix postérieur laissera nul le produit π (condition d'injectivité).

En pratique, cette deuxième remarque est facilement utilisable et efficace pour $m \geq 2$.
$P(x|x_1 \ldots)$ contient alors un ou 2 éléments.

III-2-2 - L'algorithme de recherche d'une immersion possible accéléré par la première remarque et par la deuxième remarque dans le cas $m \geq 2$ a été programmé de façon élémentaire sur ordinateur et donne des résultats acceptables pour des graphes d'une trentaine de sommets.(voir chapitre Programme).

IV - CAS D'UN ARBRE

La recherche des immersions d'un arbre, au moyen de l'algorithme précédent, est très simple. Dans une structure en couches donnée, tout sommet est adjacent à un sommet et un seul de la couche précédente. L'ensemble des coordonnées possibles d'un sommet x est donc de la forme $P(x|x')$ et a n éléments.
La condition d'injectivité seule élimine certains termes (produit π nul).

On en déduit qu'un arbre est toujours immergeable dans un n-cube en prenant assez grand. Le nombre d'immersions possibles non équivalentes dans un n-cube est, en général très important. Cet ensemble d'immersions sera donné par l'expression très simple

$$\prod_{x \in X^\circ} P^*(x|x')$$

où x' est le sommet adjacent à x de la couche précédente et X° est l'ensemble des sommets à partir de la 2ème couche.

V - RECHERCHE DE L'ISOMORPHIE ENTRE UN GRAPHE DONNE ET UN n-CUBE

Ce problème, très simple, peut être résolu par l'application de conditions nécessaires successives, procédé employé usuellement dans la recherche de l'isomorphie de 2 graphes (UNGER [32], STEEN [18]).

Il semble plus intéressant ici d'utiliser l'algorithme de recherche d'une immersion précédemment décrite.

Au cours de l'application de cet algorithme, on peut conclure à l'isomorphie cherchée si et seulement si

- la ième couche d'une représentation en couches (S quelconque) à C_n^i éléments.
- $P(x|x_1\ldots)$ a un et seul élément pour tout $x \in X$.
- Le produit π de cet élément et de la solution unique trouvée pour les sommets antérieurement codés est non nul. Autrement dit, à chaque étape nous aurons une solution et une seule.

Ce procédé est donc très efficace.

VI - DIMENSION DU n-CUBE CONSIDERE

VI-1 - Borne inférieure de n

Soit d_M le degré maximal d'un graphe $G(X,\Gamma)$ dont nous voulons réaliser l'immersion.

Soit $q = \lceil \log_2 n_S \rceil$ où n_S est le nombre de sommets de G. Il est clair que $G(X,\Gamma)$ ne peut être isomorphe à un sous-graphe partiel d'un n-cube où $n < \max(q,d_M)$.

Les essais d'immersion par l'algorithme ci-dessus doivent donc se faire à partir de $n_o = \max(q,d_M)$ et la solution trouvée, si elle existe, sera une immersion dans un cube de dimension minimale.

VI-2 - Borne supérieure de n

Il s'agit de déterminer le nombre d'essais à effectuer avant de pouvoir déclarer qu'un graphe est non immergeable dans un n cube, quel que soit n.

Pour un graphe donné, on cherche n_{MAX} , tel que si le graphe n'est pas immergeable dans le cube de dimension n_{MAX} , il ne sera pas immergeable dans un cube de dimension supérieure.

Remarque :

Etant donné un graphe, on peut se demander quel est le nombre maximal d'indices de direction différents pouvant figurer sur ce graphe après immersion. Si n'_{MAX} est ce nombre, toute immersion dans un cube de dimension supérieure à n'_{MAX} n'utilisera que n'_{MAX} indices de direction différents. Si nous trouvons une borne supérieure de n'_{MAX} , ce sera à fortiori une borne supérieure de n_{MAX}.

VI-2-1 - Proposition 3

Soient x *chaines de* p *arêtes* ($p \geq 3$) *n'ayant en commun que leurs extérmités* A *et* B. *La borne supérieure du nombre d'indices de direction différents utilisés pour coder ce graphe est atteinte quand* A *et* B *sont à une distance minimale sur le cube. Cette borne supérieure est égale à*

$$\begin{cases} \text{Max}\left[p,\ 1+\frac{x(p-1)}{2}\right] & \text{si p impair} \\ \text{Max}\left[p,\ 2+\frac{x(p-2)}{2}\right] & \text{si p pair} \end{cases} \qquad (1)$$

Remarque : si $p = 2$, alors $x \leq 2$ pour un graphe immergeable et la borne supérieure cherchée est 2.

Démonstration

si $x = 1$ la borne supérieure cherchée est p.

Supposons $x > 1$.

a) A et B sont mis à distance p sur le cube ($x \leq p$).

Nous aurons p indices différents.

b) A et B sont mis à distance $p-2$ ($p-2 \geq 1$).

Nous aurons, au plus $(p-2)+x$ indices.

c) A et B sont mis à distance $(p-q)$ sur le cube ; $(p-q) \geq 1$.

Nous aurons, au plus, $(p-q)+\frac{q}{2}x$ indices de direction différents.

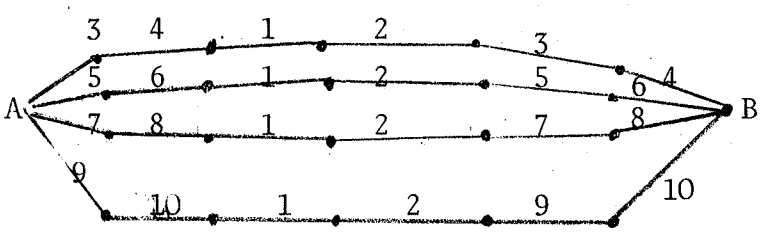
En effet $|\mathcal{E}_c(A,B)| = p-q$; tout indice n'appartenant pas à $\mathcal{E}_c(A,B)$ doit figurer, au moins, 2 fois sur une chaine.

Nous aurons donc $p+\frac{q}{2}(x-2)$, au plus, indices différents.

Comme $x \geq 2$, cette expression est maximale quand q est maximal, donc si A et B sont à distance minimale sur le cube.
si p est pair la borne supérieure de q est p-2 ;
si p est impair la borne supérieure de q est p-1 .

En remplaçant q par ces expressions, on trouve les limites énoncées.

Exemple x = 4 p = 6 distance 2



On peut mettre, au plus, 10 indices de direction différents.

Corollaire

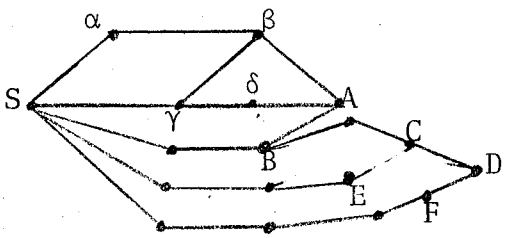
Soient x chaînes ($x \geq 2$) de p arêtes n'ayant en commun que leurs extrémités A et B. Si $\mathcal{E}_c(A,B)$ est déterminé, nous pourrons introduire, au plus, $\frac{x(p-1)}{2}$ (p impair) ou $\frac{x(p-2)}{2}$ (p pair) indices de direction n'appartenant pas $\mathcal{E}_c(A,B)$ en codant ces x chaines. Les termes 1 ou 2 ne sont plus à considérer.
En effet, nous nous plaçons dans le cas où $|\mathcal{E}_c(A,B)| = 1$ (p impair) ou $|\mathcal{E}_c(A,B)| = 2$ (p pair); si $x \leq 2$, aucun nouvel indice ne pourra être introduit.

VI-2-3 - APPLICATIONS: Recherche de n'_{MAX} ; minimisation de n'_{MAX}

On cherche une couverture des arêtes de $G(X,\Gamma)$ par un ensemble de chaînes simples (chaque arête appartenant à une chaîne et une seule). A toute chaîne ou tout ensemble de chaînes de même longueur entre 2 sommets appartenant à cette couverture, on associera le nombre maximal d'indices de direction différents pouvant y figurer. La somme de ces nombres sera une limite supérieure de n'_{MAX}.
Il est clair que cette évaluation dépend de la couverture choisie et même de l'ordre dans lequel les chaînes ou ensembles de chaînes sont considérés. En effet le corollaire énoncé ci-dessus s'applique ainsi :

> Soit une chaîne A α B de la couverture considérée.
> S'il existe une chaîne A α' B distinct de la précédente dont toutes les arêtes appartiennent déjà à la couverture on associe à A α' B le terme $\frac{p-1}{2}$ ou $\frac{p-2}{2}$ (suppression du terme 1 ou 2).
> En effet, après considération de la chaîne A α B, $\varepsilon_c(A,B)$ est fixé.

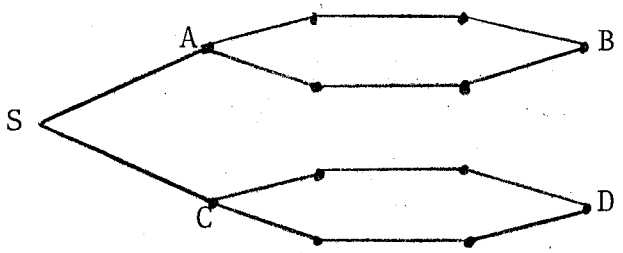
Exemple 1 :
Soit le graphe



On associe aux 2 chaînes (SγβA) et (SBA) 3 indices différents ;
on associe à la chaîne (B,C,E,S) 2 indices différents ;
on associe à la chaîne (C,D,F,S) 2 indices différents ;
(remarque précédente).

La remarque précédente nous permet d'affirmer qu'aucun nouvel indice ne pourra être introduit sur S α β ou γ δ A.
Soit $n'_{MAX} = 7$.

Exemple 2 :



On associe aux 2 chaînes A,B 3 indices différents ;
on associe aux 2 chaînes C,D 3 indices différents.

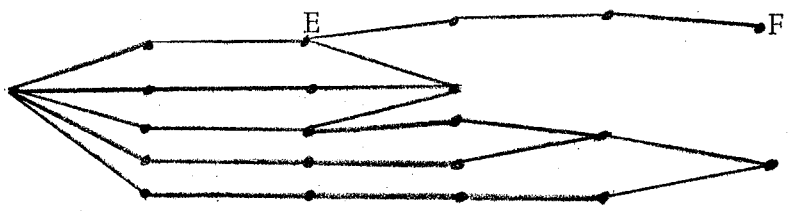
On peut, cette fois, associer 2 indices différents à la chaîne (ASC), qui ne remplit plus la condition de la remarque précédente.

Soit $n'_{MAX} = 8$ qui est le diamètre du graphe

Chaîne pendante

Considérons une chaîne dont une extrémité est de degré 1, une extrémité de degré quelconque, les sommets restants étant de degré 2.

Exemple : La chaîne EF dans le graphe suivant



le calcul précédent nous fait associer un nombre d'indices différents égal à la longueur de cette chaîne pendante.

On peut remarquer que de telles chaînes n'appartiennent à aucun cycle, et la seule contrainte d'immersion sera la contrainte d'injectivité ou d'encombrement sur le cube considéré. On affectera à une telle chaîne, un hyperplan de dimension suffisante ou un nombre d'indices α tel que $\alpha = \lceil \log_2 \ell \rceil$ si ℓ est la longueur de la chaîne (où le nombre de sommets à considérer de cette chaîne).

Exemple précédent

On associe un seul indice à la chaîne EF.
Au graphe complet est alors associé une grandeur n^* telle que $n_{MAX} \leq n^* \leq n'_{MAX}$.

CHAPITRE - IV

ETUDE ALGEBRIQUE DE L'IMMERSION D'UN GRAPHE BIPARTITE DANS UN n-CUBE

I - GRAPHE ET SOUS-GRAPHE PARTIEL D'UN n-CUBE

I-1- GRAPHE D'UN n-CUBE

Nous allons nous intéresser spécialement à la représentation du n-cube par un treillis de Boole (définie au chapitre précédent).

Rappelons les propriétés suivantes :

I-1 - Cette représentation ordonnée comprend $n+1$ niveaux. Le niveau 0 étant constitué par le minimum S. Il existe C_n^i élément dans le niveau i ($i \leq n$) appelés éléments de rang i.

I-2 - Tout élément de rang j s'écrit d'une façon unique comme union irrédondante de C_j^i ($i,j \leq n$) et ($i < j$) éléments de rang i; à 2 éléments distincts, de même rang, correspondent 2 écritures distinctes.

I-3 - La section commençante d'un élément de rang j est isomorphe à un (n-j)-cube ; la section finissante d'un tel élément est isomorphe à un j-cube. On en déduit, par exemple, que un élément de rang i, a i prédécesseurs (éléments adjacents de rang i-1) et n-i successeurs (éléments adjacents de rang i+1).

Remarque

La définition de la section commençante d'un élément d'un ensemble ordonné est la définition habituelle, à savoir, l'ensemble des éléments y, tel que $y \geq x$.

I-2 - Sous-graphe partiel d'un n-cube

Nous avons vu qu'un sous-graphe partiel d'un n-cube, auquel ne sont pas associées les coordonnées des sommets, admet, en général plusieurs représentations ordonnées. On choisira de même un sommet fixé S comme minimum de ces représentations ordonnées. A chaque représentation ordonnée peut alors être associé l'ensemble des coordonnées relatives non équivalentes.

I-3 - Application au problème de l'immersion

Soit un graphe bipartite connexe dont on recherche l'isomorphie avec un sous-graphe partiel d'un n-cube. Un sommet S étant arbitrairement choisi dans le graphe, on associe l'ensemble des représentations ordonnées de ce graphe, ayant un minimum S. Etant donné une représentation ordonnée du graphe, il sera facile de rechercher l'isomorphie avec un sous-graphe partiel d'un treillis de Boole.
Nous étudierons le problème de l'immersion en 2 temps :

a) étude des représentations ordonnées d'un graphe bipartite ayant un minimum.

b) étude de l'immersion d'une représentation ordonnée dans un treillis de Boole.

Le premier problème, se heurtera, à des difficultés combinatoires classiques. Il ne s'agira donc pas en un premier temps de générer toutes les représentations ordonnées mais de rechercher pour chaque représentation trouvée, l'isomorphie cherchée.

II - ENSEMBLE DES REPRESENTATIONS ORDONNEES D'UN GRAPHE BIPARTITE

II-1 - Partitions de X en niveaux

Soit un graphe bipartite $G(X,\Gamma)$ et un sommet S arbitrairement choisi dans X ; soit une partition de l'ensemble des sommets X en une suite de blocs $C_0, C_1, \ldots, C_i, \ldots C_q$ telle que :

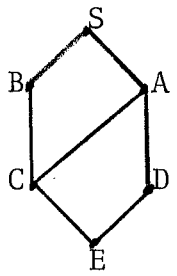
a) C_0 contient le sommet unique S

b) $\Gamma(x,x')$ => x,x' appartiennent à des blocs adjacents de la suite précédente.

S' étant fixé, nous noterons $\mathcal{P}_S$, l'ensemble des partitions de X remplissant ces 2 conditions.

Exemple

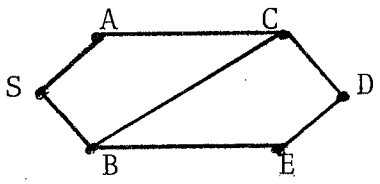
Soit $G(X,\Gamma)$



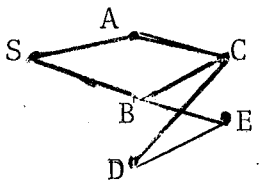
$\mathcal{P}_S = \{P_1, P_2\}$

avec

$P_1 = (S, AB, CE, D)$



$P_2 = (S, ABD, CE)$



II-2 - Représentations ordonnées d'un graphe bipartite

A chaque élément de $\mathcal{P}_S$, on associe une représentation ordonnée du graphe. Les éléments de $C_0, C_1 \ldots C_i \ldots$ sont représentés sur des verticales ou niveaux successifs. Un sommet C_i sera appelé élément de rang i.
Deux sommets adjacents appartiennent à 2 niveaux successifs. Orientations chaque arête dans cette représentation de gauche à droite et soit la relation d'ordre définie par $x < x' \iff$ il existe un chemin (x,x').
A chaque élément de $\mathcal{P}_S$ correspond ainsi une représentation ordonnée du graphe bipartite. Un algorithme simple donnera ultérieurement l'ensemble des représentations ordonnées ayant un minimum d'un graphe bipartite donné. Il est clair que nous avons ainsi associé toutes les représentations ordonnées de $G(X,\Gamma)$ ayant un minimum S .

III - IMMERSION D'UNE REPRESENTATION ORDONNEE D'UN GRAPHE BIPARTITE DANS UN TREILLIS DE BOOLE A n-ATOMES

III-1 - Définition

Etant donnée une représentation ordonnée d'un graphe bipartite $G(X,\Gamma)$ nous cherchons un sous-graphe partiel d'un treillis de Boole isomorphe à cette représentation.
En utilisant comme précédemment les coordonnées relatives par rapport à un sommet minimum $S \in X$, la recherche de cette immersion sera la recherche de l'application injective : $x \to \mathcal{E}_c(S,x)$ pour tout $x \in X$ $(x \neq S)$ telle que

a) $\Gamma(x,x') \Rightarrow |\{ \mathcal{E}_c(Sx) \oplus \mathcal{E}_c(S,x')\}| = 1$

b) $x < x'$ dans la représentation ordonnée de $G(X,\Gamma)$
$\Rightarrow \mathcal{E}_c(S,x) \subset \mathcal{E}_c(S,x')$.

Conséquences

On en déduit immédiatement que si x est de rang i alors $| \mathcal{E}_c(S,x) | = i$.

III-2 - Conditions nécessaires d'immersion d'une représentation ordonnée d'un graphe bipartite dans un treillis de Boole.

Nomenclature : nous appelerons représentation ordonnée d'un graphe bipartite plus brièvement graphe bipartite ordonné ; de même la représentation d'un n-cube par un treillis de Boole à n atomes sera appelé plus simplement n-cube.

III-2-1 - Condition nécessaire 1

Pour qu'un graphe bipartite ordonné soit immergeable dans un n-cube, il faut que :

- le nombre d'éléments de rang i, N_i , vérifie $N_i \leq C_n^i$
- un élément de rang i ait, au plus, (n-i) successeurs (éléments adjacents de rang i+1) et i prédécesseurs (éléments adjacents de rang i-1).

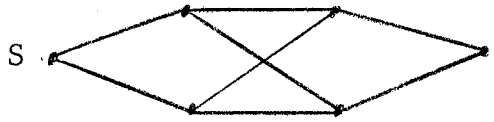
Remarque

La condition nécessaire portant sur le nombre de prédécesseurs d'un éléments est indépendante de la dimension n du cube.
Cette condition se déduit directement de l'étude de la structure d'un treillis de Boole.

III-2-2 - Condition nécessaire 2 (structure de treillis affaibli)

Pour qu'un graphe bipartite ordonné, soit immergeable dans un n-cube, il faut que 2 éléments distincts de rang i, aient, au plus, un majorant (minorant) de rang i+1 (i-1).
Si cette condition n'est pas vérifiée, une complétion en treillis sera impossible.

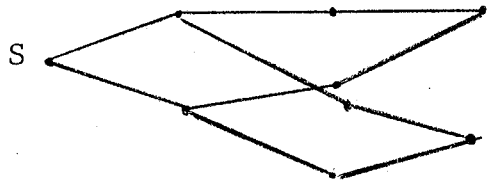
Exemple : graphe non immergeable dans un n-cube.



Remarque

- cette condition nécessaire est indépendante de n.
- rappel : il n'existe pas nécessairement une borne supérieure pour tout couple d'éléments d'un graphe immergeable dans un n-cube.

Exemple : graphe immergeable dans le 4-cube.



III-2-4 - Condition nécessaire 3

Si un graphe bipartite ordonné est immergeable dans le n-cube alors :

- la section commençante d'un élément de rang j est immergeable dans le (n-j) cube.
- la section finissante d'un élément de rang j est immergeable dans le j-cube.

Remarque

La 2ème partie est indépendante de n.

On peut donc réappliquer la condition nécessaire 1 (comptage sur les sous-ensembles définis ici).

Conséquences

De la condition nécessaire 3, on peut déduire :

- Soient 2 sommets immergés de rang j ; si d est la distance de ces 2 sommets sur l'hypercube $(d \leq 2j)$, alors l'intersection des sections finissantes de ces 2 éléments est immergeable dans le $(j - \frac{d}{2})$ -cube comprenant S. Si $d = 2j$, cette intersection doit se réduire à S.

  Il suffit de remarquer que les coordonnées relatives de ces 2 sommets ont des indices $(\alpha_1 \ldots \alpha_{j-\frac{d}{2}})$ en commun. L'intersection de leurs sections finissantes sera immergeable dans le cube $(\alpha_1 \ldots \alpha_{j-\frac{d}{2}})$

Cette condition 3 peut être renforcée.

Condition nécessaire 3 bis

Définition préliminaire

Considérons la section finissante d'un élément x de rang j. Si 2 éléments distincts de rang k $(k < j)$ appartiennent à cette section finissante et ont une borne supérieure de rang $k+1$, ajoutons cette borne supérieure à la section finissante de x. On appelera section finissante complétée de x, l'ensemble obtenu en complétant, de toutes les façons possibles, par l'opération définie ici, la section finissante de x.

Condition nécessaire 3 bis

Si un graphe bipartite ordonné est immergeable dans un n-cube, alors la section finissante complétée d'un élément de rang j est immergeable dans le j-cube.

Démonstration

Supposons le graphe immergé ; soient $\mathcal{E}_c(S,x_j) = \{\alpha_1 \ldots \alpha_j\}$ les coordonnées relatives d'un sommet x_j de rang j.
Soient x_{k_1}, x_{k_2} 2 sommets de rang k ($k < j$) et appartenant à la section finissante de x_j

$$x_{k_1} < x_j \Rightarrow \mathcal{E}_c(S,x_{k_1}) \subset \mathcal{E}_c(S,x_j)$$

$$x_{k_2} < x_j \Rightarrow \mathcal{E}_c(S,x_{k_2}) \subset \mathcal{E}_c(S,x_j) \ ;$$

si la borne supérieure de x_{k_1} et x_{k_2} existe et est de rang k+1, alors :

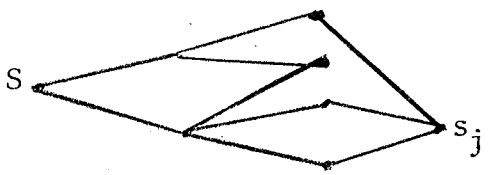
$$\mathcal{E}_c(S,x_{k_1} \cup x_{k_2}) = \mathcal{E}_c(S,x_{k_1}) \cup \mathcal{E}_c(S,x_{k_2}) \subseteq \mathcal{E}_c(S,x_j)$$

Ce sommet utilisera donc les seuls indices de direction $\{\alpha_1 \ldots \alpha_j\}$.
Il appartient donc par définition au j-cube défini par $(\alpha_1 \ldots, \alpha_j)$.

Remarque

Cette condition nécessaire et indépendante de n .

Exemple : soit le graphe bipartite ordonné :



La section finissante de s_j est immergeable dans le 3-cube mais la section finissante complétée ne l'est plus (nombre d'éléments de rang 2 trop grand).

Ce graphe bipartite ordonné n'est donc pas immergeable dans un n-cube, quel que soit n.

III-2-4 - Condition nécessaire 4

Si un graphe bipartite ordonné est immergeable dans un n-cube alors pour tout couple (x_k,x_m) $(k < m)$ où x_k et x_m sont respectivement de rang k et m, l'intersection de la section commençante de x_k et de la section finissante de x_m est immergeable dans un (m-k)-cube.

Démonstration

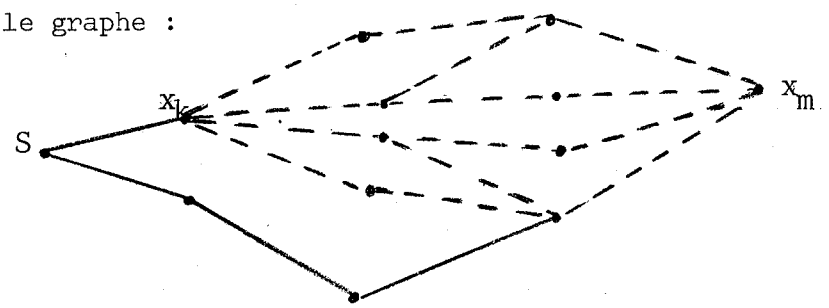
Supposons un tel graphe immergé ;

$$\left.\begin{array}{l} x_i < x_m \\ \\ x_i > x_k \end{array}\right\} \Rightarrow x_i \in \text{chemin } (x_k,x_m)$$

Un chemin (x_k,x_m) a m-k arêtes et les seuls indices de direction figurant sur un chemin (x_k,x_m) sont $\{\alpha_1,\ldots,\alpha_{m-k}\}$. L'ensemble de ces chemins est donc immergeable dans le (m-k)-cube défini par $\{\alpha_1,\ldots,\alpha_{m-k}\}$.

Exemple :

Soit le graphe :



Si on considère le couple (x_k,x_m) , le sous-graphe partiel défini précédemment (en pointillé) n'est pas immergeable dans le 3-cube ; le graphe considéré ne sera pas immergeable dans un n-cube, quelque soit n.

III-2-5 - Il est possible d'énoncer des conditions nécessaires, serrant le problème de plus en plus près. On peut par exemple rechercher des conditions nécessaires, déduites de la distributivité : "Un élément de rang j est union, au plus, de C_j^i éléments de rang i" ; en pratique, il est plus simple de vérifier ces conditions nécessaires plus restrictives en associant aux sommets leurs coordonnées relatives, par l'algorithme suivant.

III-3 - Algorithme d'immersion d'une représentation ordonnée d'un graphe bipartite dans un n-cube (n fixé)

On proposera un algorithme très efficace recherchant une immersion possible. On peut appliquer, au préalable, ou non les conditions nécessaires précédentes. Un algorithme plus complet comportant l'application préalable de ces conditions nécessaires sera donné ultérieurement.

III-3-1 - Algorithme d'immersion d'une représentation ordonnée d'un graphe bipartite dans un n-cube

On opèrera comme précédemment par niveaux successifs, les éléments d'un même niveau étant considérés dans un ordre fixé arbitrairement.

De la définition de l'immersion d'une représentation ordonnée (§ III.1 de ce chapitre), on tire les 3 conséquences : la règle R et les 2 types d'impossibilités de codage.

α) Règle R

Si un sommet x a été codé $(\alpha_1 \ldots \alpha_m)$, on assignera à tout sommet x' appartenant à la section commençante de x, $(\alpha_1 \ldots \alpha_m)$ comme sous-ensemble de ses coordonnées relatives.

Impossibilités de codage

β) si au cours de l'immersion, il existe 2 sommets x et x' , de rang k, tels que

$$|\mathcal{E}_c(S,x)| = |\mathcal{E}_c(S,x')| = k$$

et

$$\mathcal{E}_c(S,x) = \mathcal{E}_c(S,x'),$$

nous sommes en présence d'une impossibilité de codage (injectivité non respectée) ; remarquons que cette impossibilité peut être réalisée au cours de l'immersion de sommets de rang inférieur à k.

γ) soit x un sommet de rang k, si au cours de l'immmersion de sommets de rang inférieur à k $|\mathcal{E}_c(S,x)| > k$,
il y a impossibilité de codage.

Enoncé de l'algorithme

α) on affecte 1,2,... comme coordonnées relatives aux éléments de rang 1 (le nombre de ces éléments est supposé inférieur à n ; la dimension du cube sera discutée ultérieurement).
On applique la règle R ; s'il y a impossibilité de codage, le graphe n'est pas immergeable dans un n-cube.

β) Supposons codés les (j-1) premiers niveaux.
On considère les éléments du jème niveau (dans un ordre fixé, qui sera l'ordre d'introduction en machine). Soit x l'élément considéré.

- si le code d'un élément est déterminé ($|\mathcal{E}_c(S,x)| = j$), on passe à l'élément suivant.

- si $|\mathcal{E}_c(S,x)| < j$, ($\mathcal{E}_c(S,x)$ peut être vide), nous avons plusieurs choix possibles pour le codage de x. Ces différentes solutions sont obtenues en complétant $\mathcal{E}_c(S,x)$ en un ensemble de j indices (ou en générant cet ensemble si $\mathcal{E}_c(S,x)$ est vide). Les indices générés sont considérés dans un ordre lexicographique (l'ordre des entiers 1,2,...). S'il existe une solution qui, après application de la règle R, n'entraîne pas d'impossibilité de codage, on passe au sommet suivant.

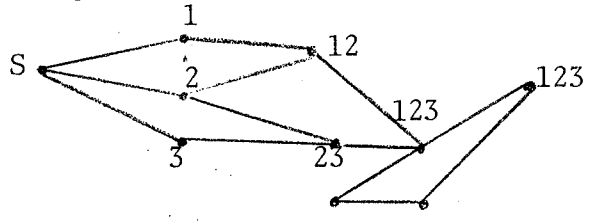
- s'il y a impossibilité de codage, après épuisement de toutes les possibilités de codage de x, on retourne au dernier sommet, pour lequel tous les codages possibles n'ont pas été considérés (les choix possibles sont mémorisés pour chaque sommet).
  Soit y ce sommet. On remet à jour (par application de la règle R), les contraintes sur les sections commençante des éléments dont le code est conservé (sommets codés avant y). On recommencera l'étape β.

- si, après épuisement de toutes les possibilités de codage pour chaque sommet, aucune solution n'est trouvée, le graphe n'est pas immergeable dans le cube considéré.

III-3-2 - Exemple

Soit le graphe bipartite ordonné que nous voulons immerger dans le 4-cube.

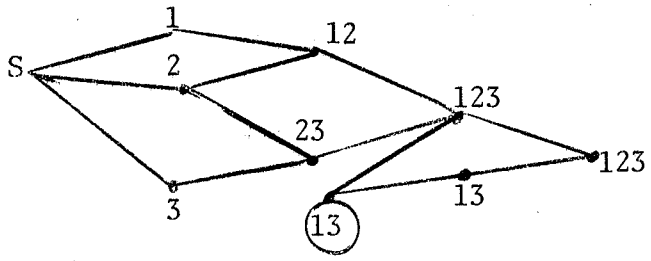
α) Codage du premier niveau avec les contraintes sur les sections commençantesde ces éléments.



A chaque sommet figure soit l'ensemble des coordonnées relatives de ce sommet, soit un sous-ensemble de cet ensemble.

β) Sommets du 2ème niveau

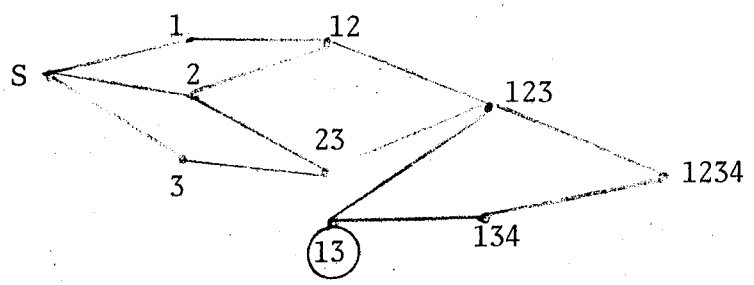
Les 2 premiers sommets sont codés ; le 3ème sommet de ce niveau sera codé 13 (le code 12 étant déjà utilisé, conduit à une impossibilité de codage) ; un choix sera indiqué par un code encerclé ;



les contraintes, portant sur le 2ème sommet du 4ème niveau, sont indiquées.

γ) Sommets du 3ème niveau

Le seul choix ne conduisant pas une impossibilité pour le 2ème sommet sera 134 d'où le codage final.



III-3-3 - Cet algorithme programmé en ALGOL sur IBM 7044 s'avère très rapide et très efficace. Des temps de calcul seront donnés dans l'appendice consacré aux programmes.

III-4 - Dimension du cube

III-4-1 - Borne inférieure

Soit d_M le degré maximal du graphe considéré.
Soit $q = \lceil \log_2 n_s \rceil$ où n_s est le nombre de sommets.
Soit r le plus petit entier tel que

$$\forall j, \text{ nombre d'éléments de rang } j : N_j \leq C_r^j$$

On prendra comme borne inférieure $n_o = \max(d_M, q, r)$.

III-4-2 - Borne supérieure

On cherchera, comme précédemment, une limite supérieure du nombre d'indices de direction distincts que l'on peut introduire au cours du codage du graphe ; cette limite étant à fortiori, une limite de la dimension du cube dans lequel on fera des essais d'immersions du graphe. L'évaluation d'une limite supérieure est, ici, rapide.

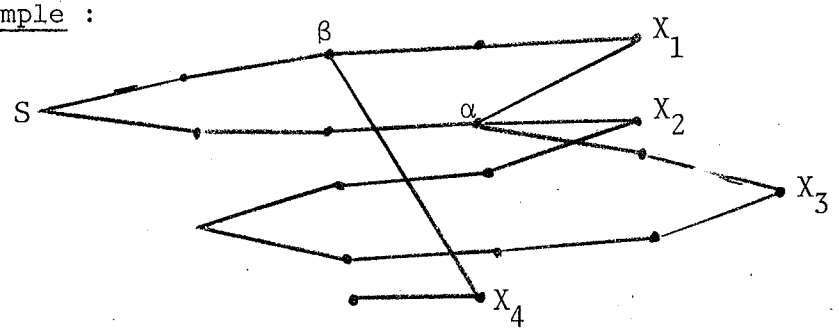
α) Soient $X_1 \ldots, X_j$ l'ensemble des sommets, éléments maximaux du graphe ordonné. Soient $R_1, \ldots, R_j$, respectivement le rang de ces sommets. Soit $N_1 = R_1 + \ldots + R_j$, la somme des rangs de ces sommets. Il est clair que N_1 constitue une limite supérieure du nombre d'indices de direction distincts pouvant apparaître au cours du codage. En effet, rappelons que dans un graphe ordonné, le rang d'un sommet est égal au nombre d'indices de direction figurant dans l'ensemble des coordonnées relatives de ce sommet. Dans l'évaluation de N_1, on suppose que les sections finissantes de X_j et X_k $(k \neq j)$ n'ont pas d'éléments communs. La limite supérieure du nombre d'indicesde direction distincts, pouvant apparaître au cours du codage, peut donc être diminuée comme suit :

β) On considère successivement les termes de N_1.

. Le terme R_1 est conservé.

. Le terme R_i : on considère l'ensemble E des sommets de la section finissante de X_i qui ont une intersection ensembliste non vide avec l'ensemble des sections finissante de $X_1 \ldots X_{i-1}$.
Si l'ensemble E est vide, on passe au terme suivant. Soit R_{M_i} la valeur maximale prise par le rang dans cet ensemble E.
On remplacera dans la somme N_1 le terme R_i par $(R_i - R_{M_i})$; cela revient à dire que dans le codage de X_i, les R_{M_i} premiers indices de direction sont déjà choisis et seuls les $(R_i - R_{M_i})$ derniers indices peuvent être choisis arbitrairement. Ce terme constitue donc une limite supérieure du nombre d'indices que l'on peut introduire en codant X_i. On procèdera ainsi pour tous les termes de la somme N_1 ; on obtient ainsi une nouvelle limite $N_o \leq N_1$.

Exemple :



Les éléments maximaux sont $\{X_1, X_2, X_3, X_4\}$

α) $N_1 = R_1 + R_2 + R_3 + R_4$

$= 4+4+5+3 = 16$

R_2 est remplacé par $R_2 - 3 = 1$, le rang de α étant 3.

En effet α est le sommet "le plus à droite" appartenant à la fois à la section finissante de X_1 et X_2.

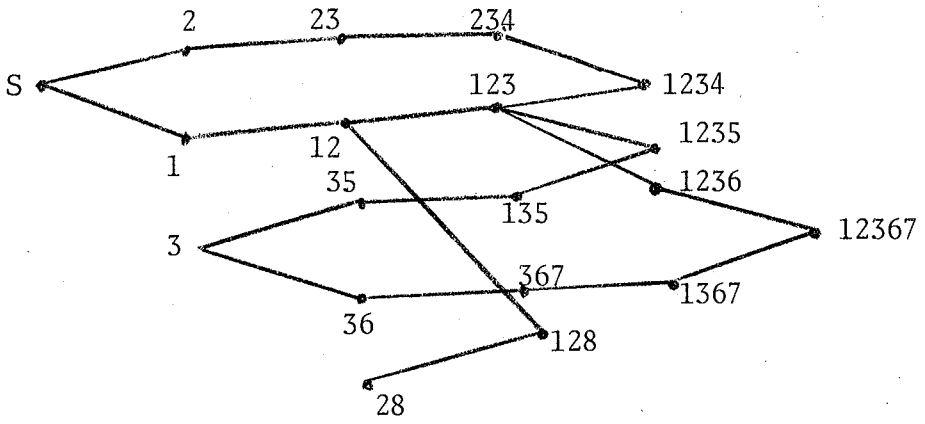
R_3 est remplacé par R_3-3 , le rang de α étant 3.

R_4 est remplacé par R_4-2 , le rang de β étant 2.

Soit $N_o = 4+1+2+1 = 8$.

Cette limite supérieure est donc considérablement diminuée et devient très utilisable.

Donnons un codage utilisant effectivement 8 indices de directions différentes :



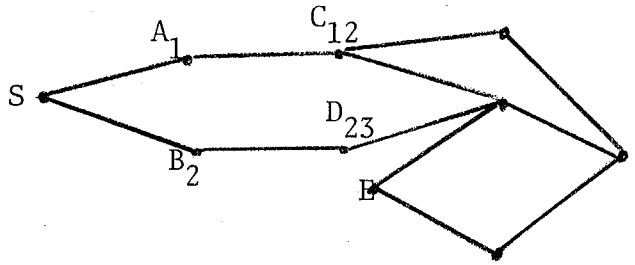
III-5 - Immersions équivalentes d'une représentation ordonnée

Nous opérons comme précédemment. Etant donnée une immersion des (j-1) premiers sommets, nous ne considèrerons que l'ensemble des immersions non équivalentes pour le jème sommet.

Considérons la partition $\pi_j = \{(\alpha_1,\ldots,\alpha_k), (\alpha_{k+1},\ldots,\alpha_n)\}$ de l'ensemble des indices de direction. Rappelons que $(\alpha_1,\ldots,\alpha_k)$ est l'ensemble des indices de direction figurant, au moins une fois, dans les

coordonnées relatives des sommets $x_1,\ldots x_{j-1}$. Dans l'ensemble des coordonnées possibles pour x_j, on définira des classes d'équivalence : 2 coordonnées relatives étant équivalentes, si elles se déduisent l'une de l'autre par permutation des indices $\alpha_{k+1},\ldots,\alpha_n$. On réduira donc l'ensemble des coordonnées possibles pour x_j à l'ensemble quotient ; le représentant choisi étant, par exemple, le premier dans un ordre lexicographique des indices de direction 1,2,...n.

Exemple : n = 5.



ensemble de coordonnées possibles pour E : 13, 14, 15, 24, 25, 45.
classes d'équivalence : (13), (14,15) (24,25), (45).
ensemble quotient constituant les différents choix considérés pour E :
13, 14, 24, 45.

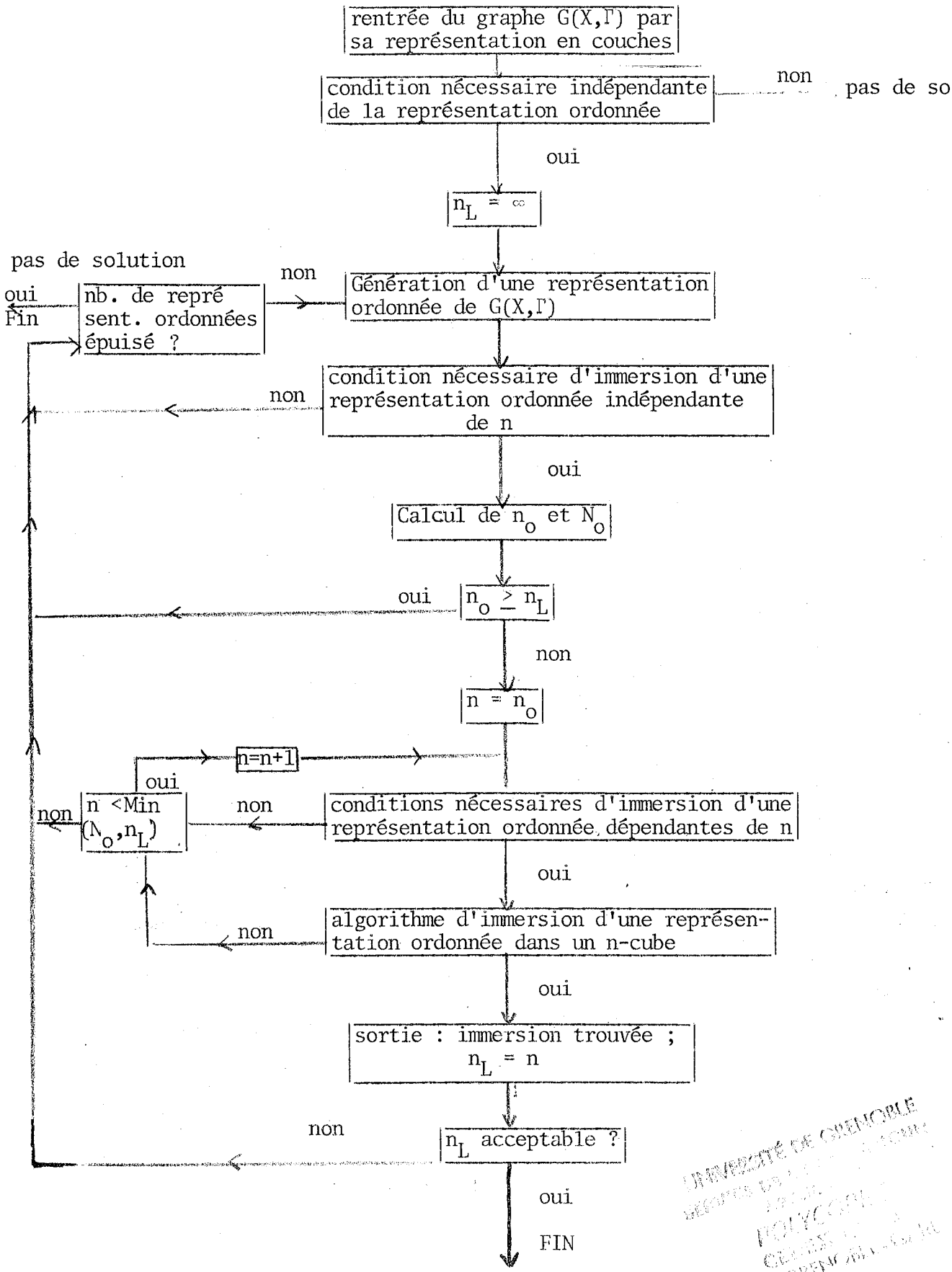
IV - APPLICATION AU PROBLEME GENERAL DE L'IMMERSION

IV-1 - Présentation générale de l'algorithme

Soit un graphe bipartite connexe $G(X,\Gamma)$ donné par sa représentation en couches. On lui appliquera dans un premier temps une condition nécessaire d'immersion indépendante de n et de la représentation ordonnée considérée (cf § IV-4). On génère successivement les représentations ordonnées de $G(X,\Gamma)$ (§ IV.3) ; à chaque représentation générée, sont appliquées dans un premier temps les conditions nécessaires d'immersion indépendantes de n (§IV-4).

Les essais d'immersion se feront ensuite entre la borne inférieure n_o et la borne supérieure N_o définies au chapitre précédent.
Si une solution est trouvée, la dimension correspondante n_L est mémorisée.
Si n_L est jugée satisfaisante, l'algorithme est arrêté, sinon on ne recherchera ultérieurement que des immersions dans un cube de dimension inférieure à n_L. Il existe donc la possibilité d'un compromis entre le temps d'obtention d'une solution et la minimalisation de la dimension du cube. Si on s'intéresse à une immersion dans un cube de dimension minimale, "n_L acceptable" doit être remplacé par "$n_L = n_o$?".

IV-2 - Organigramme de recherche d'une immersion possible d'un graphe bipartite dans un n-cube.



IV-3 - Génération des représentations ordonnées de G(X,Γ)

Générer une représentation ordonnée de G(X,Γ) ou une partition de X (cf § II.1 de ce chapitre) revient à associer à chaque sommet x un rang R(x) tel que

α) $R(S) = 0, \quad R(x) \neq 0$ pour tout $x \in X$ et $\neq S$

β) $\left.\begin{array}{l} R(x) = \alpha \\ \Gamma(x,x') \end{array}\right\} \Rightarrow R(x')$ est égal à $\alpha+1$ ou $\alpha-1$.

On note l'ensemble des rangs possibles pour x' compte tenu de sa relation d'adjacence avec x par $E_R(x'|x) = \{\alpha+1, \alpha-1\}$.
Et nous aurons

$$E_R(x|x_1,x_2,\ldots) = E_R(x|x_1) \cap E_R(x|x_2) \cap \ldots$$

si x est adjacent aux sommets $x_1, x_2, \ldots$
Cet ensemble a donc, au plus, 2 éléments; d'où l'algorithme de génération de ces représentations : nous considérons un élément particulier de $\mathfrak{S}_S$ dont la représentation est la structure en couches définie au chapitre 3 ; nous procédons couche par couche à partir de cette structure particulière et développons, comme précédemment, l'algorithme combinatoire suivant : étant donnée une solution pour les j premières couches (à chaque sommet a été affecté un rang), on cherche les solutions possibles pour la $(j+1)^\circ$ couche.

- Couches C_0, C_1, C_2

  S étant choisi comme sommet unique de rang 0, il existe une seule solution pour les sommets de ces 3 couches

  $R(x) = 1$ pour tout $x \in C_1$

  $R(x) = 2$ pour tout $x \in C_2$

- couche j

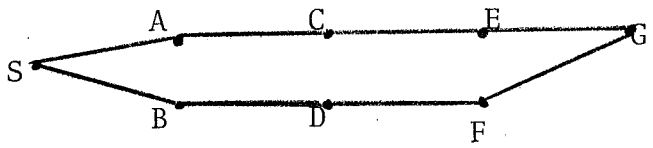
Si un sommet x de la couche j est adjacent aux sommets $x_1, x_2, \ldots$ de la couche (j-1), $E_R(x|x_1,x_2,\ldots)$ est l'ensemble des rangs possibles pour x. Si cet ensemble n'est pas vide, associons au sommet x la somme des variables $x_{R_1}+x_{R_2}$ où R_1 et R_2 sont les 2 rangs possibles pour x. Les solutions pour la jème couche sont alors données par le produit $\prod_{x,y..\in C_j} (x_{R_1}+x_{R_2})(y_{R'_1}+y_{R'_2})\ldots$

Ce produit est distributif par rapport à la somme précédente.

Si $X = \{x_{R_1}, x_{R_2}\}$, $Y = \{y_{R'_1}, y_{R'_2}\}\ldots$ les différentes solutions sont tous les éléments de $X \times Y \times \ldots$

Exemple :

Soit le graphe dont la structure en couches est



En faisant figurer le rang en indice du sommet, on obtient

2 premières couches

$$A_1B_1C_2D_2$$

3 premières couches

$A_1B_1C_2D_2(E_1+E_3)(F_1+F_3)$ soit4 termes.

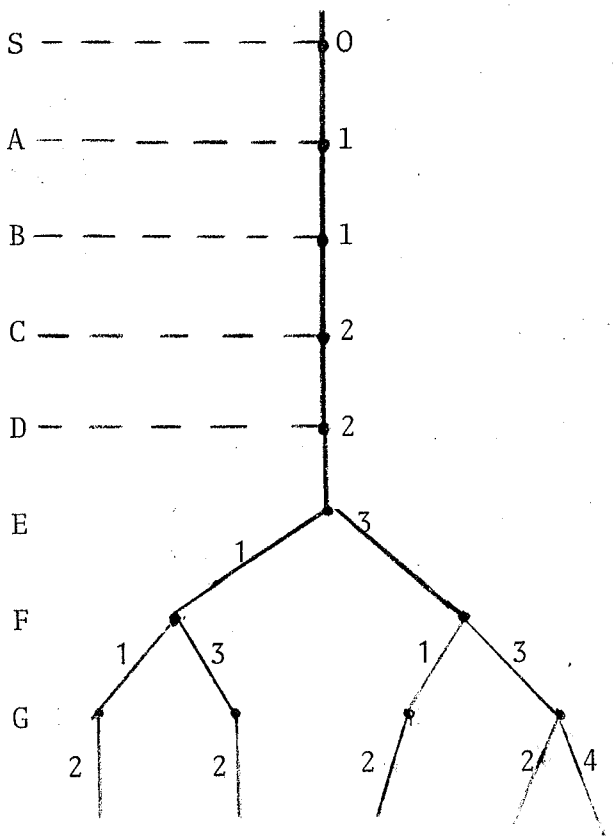
Pour chacun de ces 4 termes on obtient les solutions suivantes pour le graphe

$A_1B_1C_2D_2E_1F_1G_2$

$A_1B_1C_2D_2E_3F_1G_2$

$A_1B_1C_2D_2E_1F_3G_2$

$A_1B_1C_2D_2E_3F_3(G_2+G_4)$ soit 5 solutions que l'on peut représenter par l'arborescence des choix avec les mêmes conventions que dans le chapitre précédent.



Comme précédemment, on génère une solution possible (un chemin complet) et on retourne au dernier noeud de l'arborescence origine d'un arc non exploré si l'algorithme d'immersion requiert une autre représentation.

Remarque 1

Il est évident que le nombre total de représentations ordonnées d'un graphe bipartite est très important. Un graphe de 34 sommets a été étudié par un programme ALGOL récursif de 720 unités syntaxiques. Cette étude a été arrêtée après obtention de 3000 solutions.

Remarque 2

La difficulté de l'algorithme étudié étant dûe au grand nombre de représentations ordonnées d'un graphe bipartite, il est intéressant de rechercher des conditions nécessaires s'appliquant à toutes les représentations ordonnées et quelque soit n.

IV-4 - Classification des conditions nécessaires

IV-4-1 - Conditions nécessaires indépendantes de la représentation ordonnée choisie

Si dans un graphe bipartite $G(X,\Gamma)$, il existe un sous graphe partiel isomorphe au graphe  aucune représentation ordonnée de $G(X,\Gamma)$ ne sera isomorphe à un sous-graphe partiel d'un treillis de Boole. En effet, dans toute représentation ordonnée, ce sous-graphe partiel aura comme diagramme

Les 2 premiers diagrammes ne remplissent pas la condition nécessaire 2 (structure de treillis affaiblie), le 3ème ne remplit pas la condition nécessaire 4.

Il sera donc intéressant d'éliminer des graphes comprenant de tels sous-graphes partiels ; l'application préalable de cette condition évite pour de tels graphes la génération de toutes les représentations ordonnées.

IV-4-2 - Conditions nécessaires d'immersion d'une représentation ordonnée indépendantes de n (rappel)

Ces conditions nécessaires énumérées précédemment sont :

. *Condition nécessaire 1 partielle* : un élément de rang i doit avoir, au plus, i prédécesseurs.

. *Condition nécessaire 2* : structure de treillis affaiblie.

. *Condition nécessaire 3 partielle* : La section finissante d'un élément de rang j doit être immergeante dans le j-cube, ce qui permet l'application de la condition nécessaire 1 sur cette section finissante : comptage des éléments d'un rang donné, nombre de précédesseurs, nombre de successeurs d'un élément dans cette section. Cette section finissante peut être complétée comme indiquée dans la condition nécessaire 3 bis.

. *Condition nécessaire 4* : Elle permet l'application de la condition nécessaire 1 complétée aux sous-graphes partiels définis.

IV-4-3 - Conditions nécessaires d'immersion d'une représentation ordonnée dépendantes de n

. *Condition nécessaire 1 partielle* : Tout élément de rang i doit avoir n-i successeurs.

. *Condition nécessaire 3 partielle* : La section commençante d'un élément de rang i doit être immergeable dans le n-i cube d'où application de la condition nécessaire 1 à cette section.

CHAPITRE - V

METHODE PROPOSEE DE CODAGE DES AUTOMATES ASYNCHRONES

PRINCIPE GENERAL

Les étapes de cette méthode sont les suivantes :

a) Le fonctionnement de l'automate considéré est traduit par un graphe.

b) On extrait de ce graphe un graphe partiel immergeable dans le n_o -cube ce qui suppose un choix préalable de n_o. On considère une immersion possible (à chaque état de l'automate est alors associé son image sur le n_o-cube ou son "code").

c) On considère la contrainte impérative de codage (chap. 1 §III) qui n'est pas en général respectée pour l'ensemble de l'automate. On déterminera donc, dans cette étape, l'ensemble des sommets supplémentaires associé à l'automate.

Ces étapes seront successivement considérées et les critiques et amendements possibles examinés dans une deuxième partie.

I - TRADUCTION DU FONCTIONNEMENT D'UN AUTOMATE ASYNCHRONE PAR UN GRAPHE

I-1 - Famille de graphes associée à un automate asynchrone

L'ensemble de ces graphes classés suivant un intérêt décroissant pour le codage a été étudié au chapitre II. Le nombre de ces graphes est important et il est impossible, en pratique, de générer tous ces graphes en vue de rechercher une isomorphie éventuelle à un sous-graphe partiel d'un n_o-cube ou l'extraction d'un graphe partiel réalisant cette isomorphie.

Cas particulier : automate à 2^q états ; recherche d'un codage à q composantes

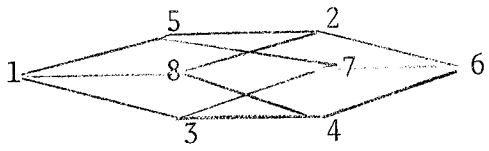
Soit un automate à 2^q états. La recherche d'un codage à q composantes de la variable interne se fera en générant l'ensemble des graphes de fonctionnement attaché à cet automate dans l'ordre décrit au chapitre II.

Pour chacun de ces graphes, on recherche l'isomorphie avec le q-cube. Nous avons vu (chap. III) que cette recherche est rapide. Le processus est arrêté si un graphe remplit cette condition, il s'agit alors d'un codage à nombre minimal de composantes de la variable interne.

Exemple Haring [12]

	E_1	E_2	E_3	E_4
1	1	2	5	3
2	4	2	5	8
3	1	6	7	3
4	4	2	7	3
5	1	6	5	3
6	4	6	5	8
7	4	6	7	8
8	1	2	7	8

un des graphes de fonctionnement obtenu est le suivant, isomorphe au 3-cube



Dans le cas général, il ne sera pas possible, à fortiori, de rechercher dans cette famille de graphes, ceux ayant un nombre minimal d'arêtes. De plus, une adjonction éventuelle de sommets supplémentaires diminue l'intérêt de ces éléments.

I-2 - Choix d'un graphe : graphe des transitions directes pondéré

Nous chercherons, dans un premier temps, au cours de cette méthode de codage, à réaliser un nombre maximal de transitions directes de l'automate. Nous considèrerons donc le graphe des transitions directes, obtenu directement à partir du tableau de fonctionnement de l'automate. Ce graphe sera pondéré comme indiqué au chapitre II. Rappelons que ce graphe possède une propriété importante : si le tableau des états de l'automate n'a pas été réduit parfusionnement, ce graphe est bipartite.

II - EXTRACTION D'UN GRAPHE PARTIEL IMMERGEABLE DANS LE n_o-CUBE

Cette extraction se fera en 2 temps

a) choix d'un arbre complet de poids maximal ; évaluation de n_o.

b) complétion de cet arbre.

II-1 - Arbre complet

Un arbre est un graphe toujours immergeable dans un n-cube. Il suffit de prendre n assez grand (chapitre III). On choisira un arbre complet de poids maximal du graphe précédent. En effet, la pondération du graphe initial traduit l'intérêt des arêtes du graphe. Une arête est d'autant plus intéressante qu'elle réalise un plus grand nombre de transitions (pour des entrées différentes). Soit n_o la dimension minimale d'un cube dans lequel cet arbre soit immergeable.

Remarque : minimisation de n_o

Nous avons vu que la borne inférieure de la dimension du cube dans lequel on essaie d'immerger un graphe $G(X,\Gamma)$ est $\max(\lceil \log_2 |X| \rceil, d_M)$ où d_M est le degré maximal de G. On peut donc chercher à minimiser cette borne inférieure en recherchant un arbre complet de degré maximal $d_M = \lceil \log_2 |X| \rceil$.
Soit $q = \lceil \log_2 |X| \rceil$. En pratique, en vue d'une solution complète de codage dans un q-cube, il est intéressant de limiter le degré de l'arbre à q-1. En effet, soit un sommet de degré supérieur à q dans le graphe initial. Par l'arbre complet, nous réaliserons q-1 transitions directes. Le $q^{ième}$ sommet adjacent non utilisé du q-cube pourra servir pour les transitions restantes non réalisées si ces transitions se font pour des valeurs d'entrée différentes.

Algorithme pratique

La détermination d'un arbre complet de poids maximal [Lobermann [19]] se fera par l'algorithme pratique suivant :

- Le graphe initial est entré sous forme d'une liste d'arêtes rangées dans un ordre d'intérêt décroissant ; à chaque arête est associé un ensemble de 2 sommets ; au début de l'algorithme, ces 2 sommets sont les 2 extrémités de l'arête :

arête	sommets associés	
a_1	x_{1_1}	x_{1_2}
⋮	⋮	⋮
a_i	x_{i_1}	x_{i_2}
⋮		

- les arêtes sont sélectionnées dans cet ordre pour l'arbre cherché ; une arête a_i étant choisie, le sommet figurant en x_{i_2} est remplacé par le sommet figurant en x_{i_1}, pour toutes les arêtes non encore considérées, où apparait le sommet figurant en x_{i_2}. Une arête est non choisie ou éliminée si les 2 sommets associés sont identiques (fermeture d'un cycle).

Remarque

On peut, au cours de l'algorithme précédent, mémoriser après chaque choix d'arête, le degré des sommets du graphe, et par suite, limiter ce degré. Une solution n'existera pas toujours.

II-2 - Complétion de cet arbre

On considère les arêtes du graphe n'appartenant pas à l'arbre complet choisi dans l'étape précédente ; ces arêtes sont considérées dans un ordre d'intérêt décroissant. Une arête est rajoutée si, après adjonction de cette arête, le graphe partiel obtenu est immergeable dans le n_o-cube. On teste dans un premier temps si le graphe partiel est bipartite et dans l'affirmative on applique un algorithme d'immersion dans le n_o-cube.

II-3 - Immersion du graphe partiel ainsi obtenu

La dernière immersion trouvée (test de la dernière arête rajoutée) est considérée. On peut donc remarquer qu'il s'agit d'une immersion arbitraire

(celle trouvée par l'algorithme du chapitre 3), non nécessairement optimale pour l'étape suivante. Il est à remarquer que très fréquemment dans les exemples traités, il existe une seule immersion possible. En effet, l'arbre complet initial est complété par fermeture de cycles et le nombre des immersions possibles non équivalentes se réduit.
A la fin de cette étape est donc associée à chaque sommet x (ou état de l'automate), son image $I(x)$ sur le cube (ou code de l'état)

III - RESPECT DE LA CONTRAINTE IMPERATIVE, RECHERCHE DE SOMMETS SUPPLEMENTAIRES

Dans cette recherche, on considèrera les critères suivants dans l'ordre indiqué :

- Minimalité de la dimension du cube ; une solution est d'abord cherchée dans le n_o-cube (n_o étant défini dans l'étape précédente).

- Dans un deuxième temps, les temps de transition seront considérés ; chaque état d'un bloc est relié à l'état stable en utilisant un nombre minimal d'états intermédiaires. Il semble en effet plus important de minimiser chaque transition (performance en temps de l'automate) que de minimiser la cardinalité de l'ensemble des états supplémentaires associés à un bloc donné.

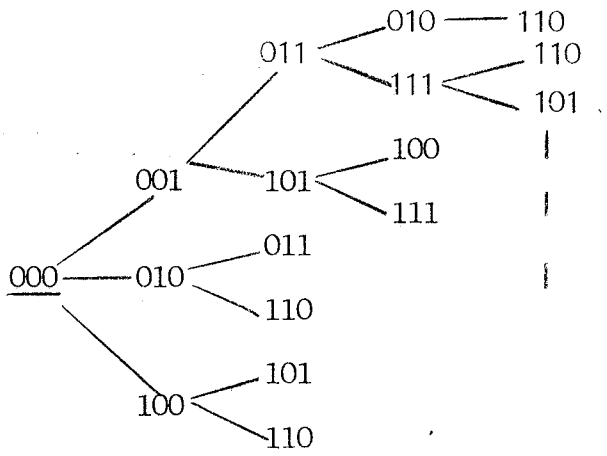
Rappelons que cette recherche se fait en considérant séparément chaque entrée. Nous avons vu, en effet, que la contrainte impérative de codage s'exprime pour une entrée E_i donnée. Soit E_i une valeur d'entrée et $\pi_i = (B_{i_1},\ldots,B_{i_j},\ldots B_{i_q})$ la partition associée à E_i. On associe au bloc B_{i_j} de π_i une arborescence sur le n_o-cube ayant comme racine l'image de l'état stable et nous donnant l'ensemble des solutions possibles pour relier l'image des états du bloc à l'image de l'état stable correspondant suivant les critères énoncés. Cette arborescence est définie et construite comme suit :

III-1 - Arborescence associée à un bloc B_{i_j} de π_i

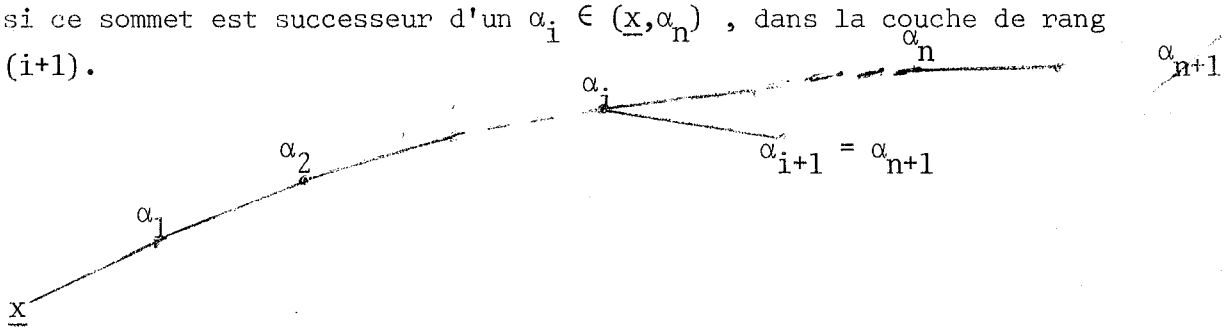
Soit $\underline{s}$ l'état stable de B_{i_j}, soit $\underline{x}$ l'image de $\underline{s}$ dans le n_o-cube. On génère une arborescence de racine $\underline{x}$ de chemins d'origine $\underline{x}$ sur le n-cube (initialement $n = n_o$). Cette arborescence est représentée par sa structure en couches (sommet source $\underline{x}$) et construite couche par couche en respectant les règles suivantes :

a) non fermeture de cycles : on ne fera pas figurer comme successeur d'un sommet, un élément qui est déjà ancêtre de ce sommet.
Cette seule restriction ferait générer l'arborescence de tous les chemins du n-cube issus de $\underline{x}$ et comporterait n-1 couches.

Exemple $n = 3$ $x = 000$



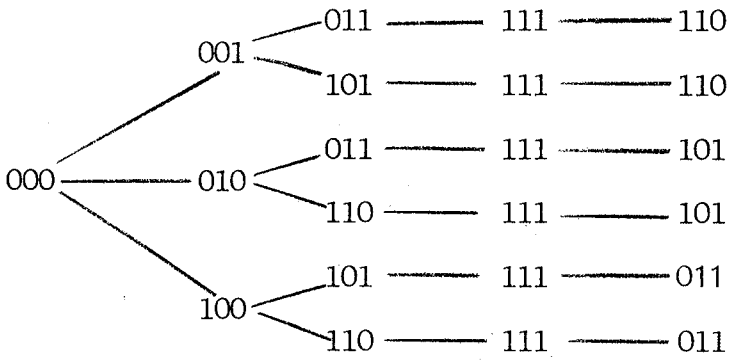
b) Simplification dûe au 2ème critère
Soit un sommet α_n de cette arborescence appartenant à la nième couche et soit $(\underline{x}, \alpha_1, \ldots, \alpha_i, \ldots, \alpha_n)$ le chemin $(\underline{x}, \alpha_n)$ de l'arborescence. Le sommet α_{n+1} successeur possible de α_n dans la couche n+1 ne sera pas considéré si ce sommet est successeur d'un $\alpha_i \in (\underline{x}, \alpha_n)$, dans la couche de rang (i+1).



Il suffit de remarquer que l'ensemble des descendants de ce sommet figurant en couche n+1 est inclus dans l'ensemble des descendants de ce sommet figurant en couche i+1 . Tout chemin $(\underline{x},\alpha_{n+1},\ldots,\lambda)$ de l'arborescence sera multiple du chemin $(\underline{x},\alpha_{i+1},..\lambda)$ de l'arborescence et ne sera jamais considéré dans cette étude.

Exemple n = 3 $\quad \underline{x} = 000$

l'arborescence précédente est alors réduite à



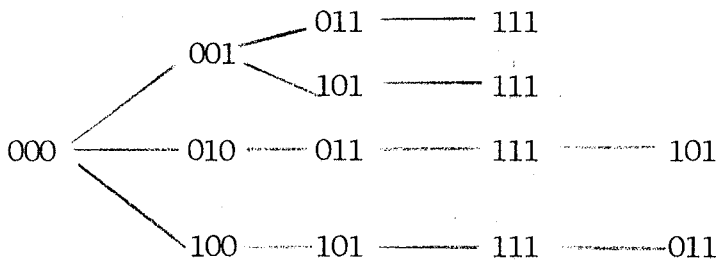
c) Respect de la contrainte impérative de codage

Si on rencontre un sommet appartenant à $(I(B_{i_\ell})$ $(1 \leq \ell \leq q$, et $\ell \neq j)$ alors le sommet est supprimé et la construction du chemin correspondant arrêtée.

(Rappel : $I(B_{i_j}) \cap I(B_{i_k}) = \emptyset$ pour $j \neq k$)

Exemple : n = 3 $\quad \underline{x} = 000 \quad$ si $110 \in I(B_{i_\ell}) \quad \ell \neq j$

alors l'arborescence associée à B_{i_j} se réduit à :



d) Arrêt de la construction de cette arborescence

Cette construction est arrêtée soit

α) après épuisement des possibilités,

β) soit après une couche donnée si tous les éléments de $I(B_{i_j})$ distincts de $\underline{x}$, figurent au moins une fois dans l'arborescence. En effet, le deuxième critère nous fera considérer que l'ensemble des chemins de longueur minimale entre l'image de l'état stable et l'image d'un état donné du bloc B_{i_j} .Dans le cas α, la dimension du cube est augmentée d'une unité et la construction des arborescences B_{i_1}, B_{i_2} ... reprise.

Nous nous supposons donc placés dans le cas β .

III-2 - Fonction attachée à l'arborescence des chemins associée à un bloc B_{i_j}.

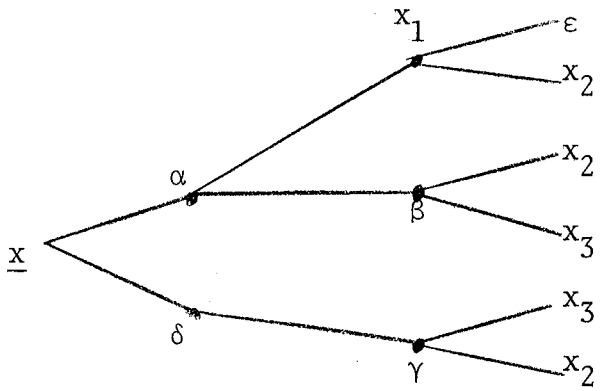
Soit x un sommet $\in B_{i_j}$ et disctinct de $\underline{x}$; considérons l'ensemble des chemins $(\underline{x},x)$ de l'arborescence, de longueur minimale. Chacun de ces chemins sera donné par la concaténation considérée comme un produit des sommets supplémentaires (distincts de $I(B_{i_j})$) y figurant. On associe à x la somme S_x de ces différents chemins. La somme S_x traduit, donc, les différentes possibilités de relier x à $\underline{x}$. On associe à l'arborescence le produit des sommes relatives à tous les sommets du bloc : soit

$F_{i_j} = \prod_{x \in I(B_{i_j})} S(x)$ où les opérations (sur l'ensemble des sommets) ont les propriétés suivantes :

- associativité et commutativité de la somme et du produit.
- distributivité du produit par rapport à la somme
- $\alpha\,\alpha = \alpha$
- $\alpha+\alpha\beta = \alpha$ ou $\Gamma+\Gamma\beta = \Gamma$ où Γ est la concaténation d'un ensemble de sommets.

Exemple :

Soit $\{\underline{x},x_1,x_2,x_3\}$ le bloc considéré et l'arborescence associée la suivante :



$S_{x_1} = \alpha$

$S_{x_2} = \alpha+\alpha\beta+\delta\gamma = \alpha+\delta\gamma$

$S_{x_3} = \alpha\beta+\delta\gamma$

$F = S_{x_1}.S_{x_2}.S_{x_3} = \alpha(\alpha+\delta\gamma)(\alpha\beta+\delta\gamma)$

$= (\alpha+\alpha\delta\gamma)(\alpha\beta+\delta\gamma)$

$= \alpha(\alpha\beta+\delta\gamma) = \alpha\beta+\alpha\delta\gamma$

Ensemble de solutions associé à B_{i_j}.

L'ensemble des solutions pour B_{i_j} est l'ensemble des termes de la somme F_{i_j}. Nous noterons $\mathcal{F}_{i_j}$ cet ensemble. Un élément de $\mathcal{F}_{i_j}$ est donc un ensemble de sommets supplémentaires pouvant être affectés à B_{i_j}.

Exemple précédent : $\mathcal{F}_{i_j} = \{\alpha\beta,\alpha\delta\gamma\}$.

III-3 - Solutions pour une entrée E_i

L'ensemble des solutions possibles pour une entrée E_i (de partition associée $(B_{i_1},\ldots,B_{i_j},\ldots,B_{i_q})$) est l'ensemble des suites $(f_1,\ldots,f_j,\ldots,f_q)$ remplissant les 2 conditions:

. $f_j \in S_{i_j}$ (solution pour le jème bloc B_{i_j})

. f_i et f_j $(i \neq j)$ n'ont pas de sommets communs.

Si cet ensemble est vide, c'est-à-dire si une telle suite n'existe pas, la dimension du cube doit être augmentée d'une unité et l'étude pour cette entrée reprise.

III-4 - Solutions globales pour l'automate

Les contraintes de codage étant indépendantes pour 2 entrées données, la solution globale consiste en la juxtaposition des solutions obtenues pour chaque entrée, les différentes solutions présentant le même intérêt pour le codage. On peut alors soit choisir arbitrairement, soit tenir compte d'autres critères comme la simplicité de l'équation de l'automate obtenue en utilisant ces différents codages.

III-5 - Solution accélérée et programmée

Considérons une entrée E_i ; un algorithme accéléré consiste à choisir arbitrairement une solution de cardinalité minimale pour B_{i_1} , puis à choisir de même une solution de cardinalité minimale pour B_{i_2} disjointe de la précédente etc... En cas d'impossibilité, la dimension du cube est augmentée d'une unité et le bloc étudié est reconsidéré, les solutions pour les blocs précédents étant maintenues (la nouvelle composante étant supposée nulle).
Cette solution est programmée sans difficulté ; il convient de rentrer l'arborescence étudiée sous une forme adéquate permettant d'éviter les problèmes d'encombrement.

IV - EXEMPLES TRAITES PAR ORDINATEUR

IV-1 - Exemple 1 tiré de Hazeltine [14]

Le fonctionnement de l'automate considéré se traduit par le tableau suivant :

	E_1	E_2	E_3	E_4
1	2	5	<u>1</u>	4
2	<u>2</u>	3	4	<u>2</u>
3	4	<u>3</u>	1	<u>3</u>
4	<u>4</u>	8	<u>4</u>	<u>4</u>
5	6	<u>5</u>	4	2
6	<u>6</u>	3	7	4
7	8	5	<u>7</u>	4
8	<u>8</u>	<u>8</u>	1	3

les partitions $\{\pi_i\}$ sont : $\pi_1 = (12,34,56,78)$

$\pi_2 = (236,157,48)$

$\pi_3 = (138,245,67)$

$\pi_4 = (25,38,1467)$

L'ordre des blocs choisi est déterminé par l'ordre des états stables y figurant (ordre des entiers naturels).
Dans cet exemple n_o sera pris égal à 4 (le nombre de sommets étant 8, $[\log_2 n_s] = 3$). En effet, la présence des arêtes obligatoires 34, 48, 38 suffit à exclure la possibilité d'une isomorphie entre un graphe de fonctionnement et un 3-cube.

Graphe pondéré des transitions directes

On prendra comme coefficients de pondération $\alpha = 3$, $\beta = 2$, $\gamma = 1$.

Ce graphe comprend 19 arêtes considérées dans l'ordre suivant (cet ordre est arbitraire entre arêtes de même poids).

arête	poids
6,7	4
2,5	4
3,8	4
1,2	3
3,4	3
5,6	3
7,8	3
4,8	3
1,5	2

arête	poids
7,5	2
2,3	2
3,6	2
1,3	2
1,8	2
2,4	2
4,5	2
1,4	2
4,6	2
4,7	2

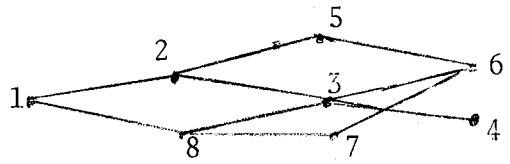
Choix d'un arbre complet de poids maximal

Les arêtes étant considérées dans l'ordre précédent, on trouve l'arbre complet de poids maximal suivant (dans ce cas particulier il s'agit d'une chaîne)

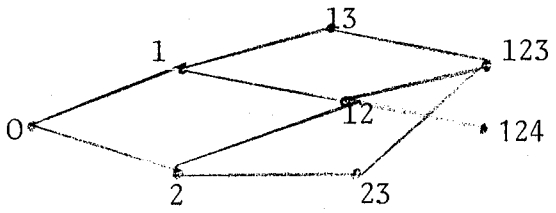
Complétion de cet arbre

Les arêtes suivantes de la liste initiale peuvent être rajoutées :
(2,3) (3,6) (1,8).

Le graphe partiel obtenu est alors le suivant :



L'adjonction de ces arêtes a été testée par l'algorithme du chapitre 3, et l'immersion trouvée, après adjonction de la dernière arête, par cet algorithme est donnée en terme de coordonnées relatives ($n_o = 4$)



ou en terme d'équivalents décimaux

états ou sommets	1	2	3	4	5	6	7	8
CODE	0	1	3	11	5	7	6	2

Adjonction de sommets supplémentaires

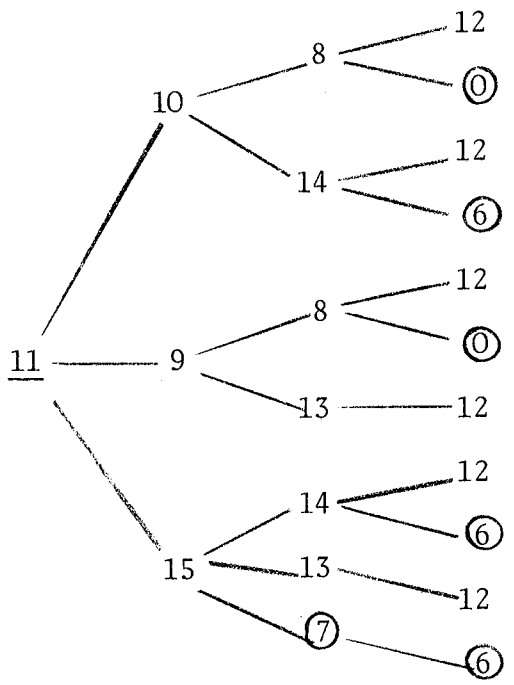
L'application de l'algorithme du 6 III.5 nous donne le résultat suivant :

blocs	E_1	E_2	E_3	E_4
B_1	-	-	-	-
B_2	-	4	9	-
B_3	-	10	-	10,3,15
B_4	-			

les sommets supplémentaires sont donnés par leurs équivalents décimaux.

Donnons à titre d'exemple l'arborescence associée au bloc

$B_{4_3} = 1\underline{4}67$



les sommets du cube encerclés, donnés en équivalents décimaux, sont images d'états du bloc B_{4_3}.

possibilités de relier $\underline{4}$ à 1 (de code 11 et 0) : 10.8+9.8

" " " $\underline{4}$ à 6 (de code 11 et 7) : 15

" " " $\underline{4}$ à 7 (de code 11 et 6) : 10.14+15.14+15 = 15+10.14

fonction associée à ce bloc : (10.8+9.8)(15)(15+10.14)

= 15(10.8+9.8) = 15.10.8+15.9.8

Une des 2 solutions est donc choisie soit 15,10,8.

Temps de calcul total (à l'aide de programmes précisés en annexe : 27").

Remarques

Le sommet supplémentaire de code 10 est utilisé pour 2 entrées différentes E_2 et E_4 .

Nous avons gagné 2 composantes par rapport à la méthode d'Hazeltine.

IV-2 - Exemple II tiré de TRACEY [30]

	E_1	E_2	E_3	E_4
a	a	c	d	c
b	a	f	c	b
c	f	c	c	c
d	-	d	d	b
e	a	d	c	e
f	f	f	-	e

les partitions associées sont :

E_1 : a be, fc

E_2 : ca, de, fb

E_3 : cbe, da

E_4 : bd, ca, ef

résultats

a) CODE états

a	b	c	d	e	f
0	4	1	2	3	5

b) Sommets supplémentaires

blocs	E_1	E_2	E_3	E_4
B_{i_1}	2	-	5	6
B_{i_2}	-	-	-	-
B_{i_3}		-		7

Remarques

Les sommets supplémentaires utilisés pour les entrées E_1 et E_3 sont images d'états n'ayant pas de successeurs définis pour cette entrée.

Temps de calcul : total 7"

IV-3 - Exemple 3 tiré de Morpurgo [9]

	1	2	3	4	5	6	7	8	9
a	a	c	e	f	a	a	b	m	m
b	a	b	c	b	b	c	b	b	b
c	c	c	c	c	b	c	c	b	b
d	f	d	d	g	f	a	d	g	d
e	e	e	e	e	1	e	f	m	e
f	f	e	d	f	f	1	f	g	m
g	f	d	h	g	h	g	d	g	h
h	h	b	h	b	h	g	h	h	h
1	1	e	m	m	1	1	1	1	m
m	1	m	m	m	1	m	1	m	m

Les partitions associées sont :

E_1 : ab, Fdg, 1m, c, e, h

E_2 : bh, ca, dg, eF1, m

E_3 : cb, dF, ea, hg, m1

E_4 : bh, c, e, fa, gd,m1

E_5 : a, bc, Fd, hg, 1em

E_6 : ad, cb, e, gh, 1f, m

E_7 : ba, c, dg, Fe, h, 1m

E_8 : bc, gdF, h, 1, mae

E_9 : bc, d, e, hg, mafl

Résultats

code

sommets	a	b	c	d	e	f	g	h	1	m
CODE	0	1	5	3	10	2	11	9	6	4

Sommets supplémentaires

blocs	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9
1	-	-	-	-	-	8,12 13,15 7	-	-	-
2	-	13,12 8	-	-	-	-	-	-	-
3	-	-	8	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-	-
5	-	-	-	-	14	-	-	8	-
6	-			-		-	-		

Temps de calcul : 25"

IV-4 - Exemple 4 tiré de FLORINE [6]

	000	100	010	001
1	1	19	10	11
2	2	12	10	20
3	3	13	10	9
4	4	14	10	9
5	5	15	10	22
6	6	16	10	9
7	7	17	10	9
8	8	18	21	9
9	3	12	-	20
10	4	13	-	-
11	5	14	-	-
12	6	15	-	22
13	7	16	-	-
14	2	17	-	-
15	7	18	-	-
16	7	19	-	-

Les partitions associées sont :

E_1 : {1 ; 2, 14 ; 3,9 ; 4, 10 ; 5, 11 ; 6, 12 ; 7, 13, 15, 16 ; 8}

E_2 : {9, 2 ; 10, 3 ; 11, 4 ; 12, 5 ; 13, 6 ; 14, 7 ; 15, 8 ; 16, 1}

E_3 : {1, 2, 3, 4, 5, 6, 7 ; 8}

E_4 : {1 ; 8, 3, 4, 6, 7 ; 9, 2 ; 12, 5}

Résultats

Code	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	0	6	12	15	9	10	3	5	4	14	13	8	11	2	7	1

Blocs	E_1	E_2	E_3	E_4
1	-	-	2,4,1,7	-
2	-	-	-	13,7,2
3	-	-		-
4	-	-		-
5	-	-		
6	-	-		
7	-	-		
8	-	-		

Remarque

Les états supplémentaires sont tous des états de l'automate n'ayant pas de sucesseur défini pour l'entrée considérée.

Temps de calcul 1' 21"

IV-5 - Exemple 5 tiré de NASLIN [24]

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
1	1	4	5	7	-	-	-	-
2	2	4	6	7	10	-	-	-
3	3	4	5	8	9	-	-	-
4	-	4	-	-	-	-	-	-
5	3	-	5	-	-	-	15	-
6	2	-	6	-	-	-	16	-
7	2	-	-	7	-	-	-	-
8	3	-	-	8	-	-	-	-
9	-	-	-	-	9	13	15	18
10	-	-	-	-	10	14	16	17
11	2	-	-	-	11	13	-	-
12	3	-	-	-	12	14	-	-
13	-	-	-	-	9	13	-	-
14	-	-	-	-	10	14	-	-
15	-	-	-	-	9	-	15	-
16	-	-	3	-	12	-	16	-
17	-	-	-	-	10	-	-	17
18	-	-	-	7	11	-	-	18

Les partitions associées sont :

E_1 : 1 ; 2, 6, 7, 11 ; 3, 5, 8, 12

E_2 : 4, 1, 2, 3

E_3 : 5, 1, 3 ; 6, 2

E_4 : 7, 1, 2, 18 ; 8, 3

E_5 : 9, 3, 13, 15 ; 10, 2, 14, 17 ; 11, 18 ; 12, 16

E_6 : 13, 9, 11 ; 14, 10, 12

E_7 : 15, 5, 9 ; 16, 6, 10

E_8 : 17, 10 ; 18, 9

Résultats

a) code

états	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
code	0	5	3	1	2	7	4	11	14	21	13	19	15	17	6	23	20	12

b) Sommets supplémentaires

Un seul sommet supplémentaire pour le premier bloc de la partition associée à E_5 : le sommet de code 11, soit le code d'un état n'ayant pas de successeur défini pour E_5.

c) temps de calcul global 1' 11".

IV-6 - Exemple 6 tiré de FLORINE [6]

	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
1	1	4	2	5	3	6	7	8
2	1	4	2	5	3	6	7	8
3	1	4	2	5	3	6	7	8
4	9	4	-	5	-	6	13	8
5	9	4	2	5	3	6	13	8
6	9	4	2	5	3	6	13	8
7	1	4	11	14	12	15	7	8
8	9	4	-	14	-	15	13	8
9	9	10	2	5	3	6	13	16
10	1	10	-	5	-	6	7	16
11	9	10	11	14	12	15	13	16
12	9	10	11	14	12	15	13	16
13	9	10	11	14	12	15	13	16
14	1	10	11	14	12	15	7	16
15	1	10	11	14	12	15	7	16
16	1	10	-	14	-	15	7	16

Les partitions associées sont :

E_1 : 1, 2, 3, 7, 10, 14, 15, 16 ; 9, 4, 5, 6, 8, 11, 12, 13

E_2 : 4, 1, 2, 3, 5, 6, 7, 8 ; 10, 9, 11, 12, 13, 14, 15, 16

E_3 : 2, 1, 3, 5, 6, 9 ; 11, 7, 12, 13, 14, 15

E_4 : 5, 1, 2, 3, 4, 6, 9, 10 ; 14, 7, 8, 11, 12, 13, 15, 16

E_5 : 3, 1, 2, 5, 6, 9 ; 12, 7, 11, 13, 14, 15

E_6 : 6, 1, 2, 3, 4, 5, 9, 10 ; 15, 7, 8, 11, 12, 13, 14, 16

E_7 : 7, 1, 2, 3, 10, 14, 15, 16 ; 13, 4, 5, 6, 8, 9, 11, 12

E_8 : 8, 1, 2, 3, 4, 5, 6, 7 ; 16, 9, 10, 11, 12, 13, 14, 15

Résultats

a) Code

Sommets	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Code	0	1	2	7	3	19	4	23	11	9	14	10	15	12	8	13

b) Sommets supplémentaires

Les sommets supplémentaires suivants sont donnés :

entrée E_4 : 2ème bloc sommet supplémentaire de code 31

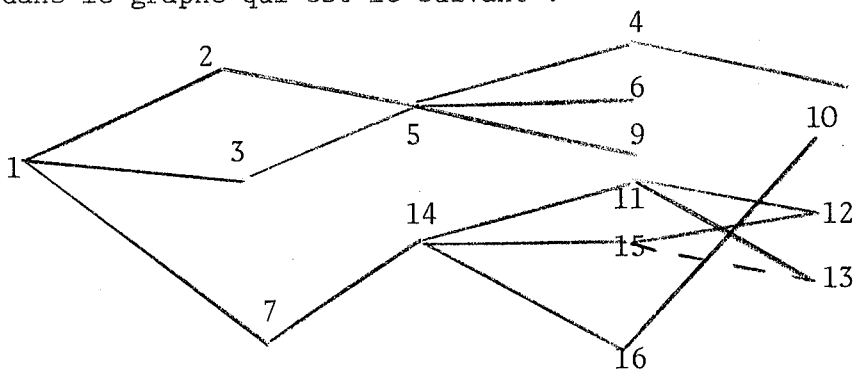
" E_6 : 2ème bloc " " " " 31

c) temps de calcul global 15'

Remarque sur ce temps de calcul

Le graphe des transitions directes comporte 76 arêtes. L'arbre complet comprenant 15 arêtes, on teste dans la première partie l'adjonction de 61 arêtes en 11' 52".

La suppression d'une arête impossible, que l'on détecterait manuellement ramène ce temps de calcul à 2' 9" . Il s'agit de l'arête (13,15), testée dans le graphe qui est le suivant :



L'algorithme d'immersion (chapitre III) détecte que ce graphe n'est pas immergeable dans le 5-cube en un temps relativement long (10') . Le graphe comporte en effet peu de cycles et de nombreuses possibilités sont testées. On envisage 2 solutions :

a) Une visualisation des arêtes testées sur écran ; une impossibilité du type rencontré ici, dans un graphe, par ailleurs très pauvre est facilement détectée.

b) Application préliminaire de la condition nécessaire développée dans l'algorithme du chapitre IV, détectant de telles structures ("condition nécessaire et indépendante de la représentation ordonnée choisie"). Cette détection ne sera un gain de temps par rapport à l'application directe de l'algorithme du chapitre III que si le graphe est pauvre en arêtes et si cette structure est présente dans des couches de rang élevé.

V - CRITIQUES ET AMENDEMENTS DE CETTE METHODE

Cette méthode appliquée aux exemples traités par d'autres méthodes donne des résultats meilleurs en temps d'obtention, en dimension de la variable interne ; les exemples pratiques de dimension importante se traduisent en général par des tableaux d'états relativement vides et peuvent alors être traités par cette méthode en des temps acceptables.

En théorie, certains points apparaissent génants. Nous allons citer 2 de ces points.

V-1 - Nous avons vu que la méthode proposée consiste à partir du graphe des transitions directes, d'en extraire un arbre complet ; cet arbre étant immergeable dans le n_o-cube, onajoute des arêtes à cet arbre si le graphe obtenu reste isomorphe a un sous-graphe partiel d'un n_o-cube.

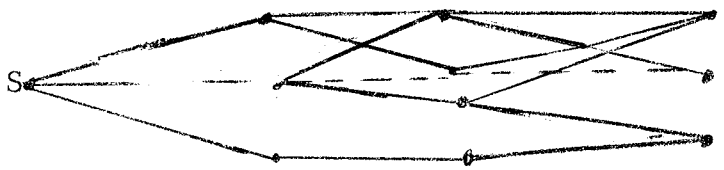
On peut penser partir du graphe des transitions directes et chercher un algorithme supprimant des arêtes au fur et à mesure des impossibilités d'immersion dans le n_o-cube ; n_o étant choisi au départ.

Ceci pourrait se faire en 2 temps :

- le sommet de référence S étant arbitrairement choisi, on supprime dans la structure en couches correspondante toute arête ayant ses extrémités dans une même couche.
  Cette étape est supprimée dans le cas d'un graphe bipartite ; ce qui est effectivement le cas dans le graphe des transitions directes que nous avons choisi.

- soit n_o la dimension du cube choisie ; si au cours de l'application de l'algorithme d'immersion, un sommet ne peut être codé, quelque soit le codage des sommets considérés préalablement, on supprime successivement les arêtes reliant ce sommet à un sommet codé antérieurement , jusqu'à obtention d'une solution. Si la dernière arête de cet ensemble ne peut être maintenue, la dimension n_o est augmentée de 1 .

Exemple :



Soit $n_o = \lceil \log_2 n_s \rceil = 4$

Le procédé précédent est très rapide et amène la suppression des 2 arêtes hachurées.

Néanmoins, la suppression des arêtes de l'ensemble considéré se fait dans un ordre arbitraire.

V-2 - Ayant extrait un graphe partiel du graphe des transitions directes, il semble arbitraire de considérer la première immersion obtenue qui ne sera donc pas forcément optimale pour l'étape suivante (recherche des sommets supplémentaires). Comme il est impossible en pratique de générer et de considérer toutes les immersions possibles, on peut penser tenir compte au cours de l'algorithme d'immersion des contraintes de l'étape suivante.

Soit un sommet x que nous voulons coder, au cours de l'algorithme d'immersion du graphe. Considérons l'ensemble des sommets adjacents à x dans le graphe des transitions directes et soit $\{y_1,\ldots,y_i,\ldots\}$ les éléments de cet ensemble codés avant x. Considérons les différents choix non équivalents pour le codage de x, non plus dans un ordre lexicographique mais en considérant dans un ordre croissant la quantité $\sum_{y_i} d(x,y_i)$ où d est la distance (x,y_i) sur le cube. Cette immersion avec contrainte semble moins arbitraire, mais il faut remarquer qu'il existera pas nécessairement un chemin direct (x,y_i) sur le cube pouvant être affecté à cette transition.

De plus, comme nous l'avons fait remarquer, les exemples traités ont montré qu'il existait souvent une seule immersion du graphe complété.

CHAPITRE - VI

FORMES DES EQUATIONS DE LA VARIABLE INTERNE APRES CODAGE D'UN AUTOMATE ASYNCHRONE

Ce problème est d'une importance fondamentale, en pratique. En effet, il est souhaitable que la réalisation technologique de ces fonctions puisse se faire par modules simples et identiques.

Nous rappelerons dans un premier temps les structures algébriques définies dans Hartmanis permettant l'approche de cette question. Nous montrerons dans le cas du codage classique de LIU, les formes systématiques des équations de la variable interne.

I - DEFINITIONS

I-1 - Rappelons (cf. Chapitre I) qu'un automate a été défini par le quintuplet $M(S,E,O,\delta,\lambda)$;
Si l'application $\delta : S\times E \overset{\delta}{\rightarrow} S$ est définie sur tout $S\times E$, nous parlerons d'automate complet ou complètement défini.

I-2 - Sous-automate relatif à une entrée E_j

Soit un automate $M(S,E,O,\delta,\lambda)$;
l'automate $M'(S,E_j,O,\delta_{E_j},\lambda)$ où
. E_j est une valeur particulière de E
. δ_{E_j} est la restriction de l'application δ
au sous-ensemble $S\times E$: $S\times E_j \overset{\delta_{E_j}}{\rightarrow} S$
est appelé sous-automate de M relatif à l'entrée E_j.

I-3 - Automate partiel

On appelle automate partiel attaché à un automate (S,E,O,δ,λ), la donnée du triplet (S,E,δ) ; en effet, nous serons souvent amenés à ne pas prendre en considération la fonction de sortie.

I-4 - Homomorphisme d'automates partiels

Un homomorphisme d'un automate partiel (S_1,E,δ_1) dans un automate partiel (S_2,E,δ_2) est une application $\varphi: S_1 \rightarrow S_2$ telle que $\varphi(\delta_1(s_1,e)) = \delta_2(\varphi(s_1),e)$ pour tout $s_1 \in S_1$ et $e \in E$.

- si l'application φ est surjective, l'homomorphisme est dit surjectif et A_2 est alors un automate partiel homomorphe à A_1 par φ .

- si l'application φ est bijective, A_2 est un automate partiel isomorphe à A_1.

II - STRUCTURES ALGEBRIQUES UTILISEES DANS LE CAS D'AUTOMATES COMPLETS

Les structures citées sont définies par Hartmanis [13] et rappelées brièvement ici. Ces structures sont relatives à l'ensemble des partitions de l'ensemble des états d'un automate ; ces partitions se confondent avec les équivalences sur cet ensemble, réticulé pour l'ordre suivant :

Soient 2 telles partitions π et τ . Nous avons :

$$\pi \leq \tau \quad \text{si } x \equiv y(\pi) \Rightarrow x \equiv y(\tau).$$

On notera $\pi+\tau$ et $\pi.\tau$ respectivement la borne supérieure et la borne inférieure de π et τ. On notera 0, la partie triviale pour laquelle chaque bloc est composé d'un seul état, et I la partition composée d'un seul bloc.

II-1 - Couple de partitions permis

C'est un ensemble ordonné de partitions (π,π') de S tel que

$$s \equiv t(\pi) \Rightarrow \delta(s,e) \equiv \delta(t,e)\,(\pi') \quad \text{pour tout } e \in E.$$

II-2 - Homomorphisme et couple de partitions permis

Soit A_2 un automate partiel homomorphe à A_1 par l'homomorphisme φ ; alors l'équivalence π sur S_1 définie par $s \equiv t(\pi)$ si $\varphi(s) = \varphi(t)$ est telle que (π,π) est un couple de partitions permis pour A_1.
En effet :

$$(\delta_1(s,e)) = \delta_2(\varphi(s),e) = \delta_2(\varphi(t),e) = \varphi(\delta_1(t,e))$$

pour tout $e \in E$.

PROPRIETE

Soient 2 couples de partitions permis (π,π'), (τ,τ') *alors* $(\pi.\tau,\ \pi'.\tau')$ *et* $(\pi+\tau,\ \pi+\tau')$ *sont des couples de partitions permis.*

II-3 - Couple de partitions Mm

II-3-1 - Opérateurs m et M

Soit une partition π de S ; on définit

$D_1(\pi) = \{\pi' ;\ (\pi,\pi')$ est un couple de partitions permis$\}$

$D_2(\pi) = \{\pi;\ (\pi',\pi)$ est un couple de partitions permis$\}$;

Les 2 opérateurs m , M sont définis par

$$m(\pi) = \prod_{\pi' \in D_1(\pi)} \pi'$$

$$M(\pi) = \sum_{\pi \in D_2(\pi)} \pi'$$

II-3-2 - Remarques

- $D_1(\pi)$ et $D_2(\pi)$ ne sont jamais vides ; en effet
  la partition $O \in D_2(\pi)$
  la partition $I \in D_1(\pi)$

- $(\pi,m(\pi))$ et $(M(\pi),\pi)$ sont des couples de partitions permis.

- $y \geq m(x) \iff (x,y)$ est un couple de partitions permis
  $x \leq M(y) \iff (x,y)$ est un couple de partitions permis.

II-3-3 - Définition d'un couple de partition Mm

Un couple de partition (x,y) sera un couple de partitions Mm si $y = m(x)$ et $x = M(y)$. Compte tenu des remarques précédentes, un tel couple est un couple permis.

II-4 - Partition relative à une entrée E_j

Cette partition définie au chapitre I, peut être définie de la manière suivante pour un automate complet : la partition relative à une entrée E_j d'un automate est la partition π_j telle que $(\pi_j,0)$ soit un couple de partitions Mm pour le sous-automate relatif à E_j .

Notation : On notera $(\pi,\pi')_{E_j}$ un couple de partitions du sous-automate relatif à E_j d'un automate complet.

II-5 -Théorème sur l'écoulement de l'information (Hartmanis).

II-5-1 - Définitions préliminaires

Nous dirons qu'une composante y_i d'une variable booléenne générale Y est codée suivant une partition de 2 blocs (B_1,B_2) d'un ensemble d'états si $Y_i = 0$ pour les états d'un des 2 blocs, $Y_i = 1$ pour les états du 2ème bloc.

Généralisation.

Soit π une partition d'un ensemble d'états en q blocs.

Soit $r = \lceil \log_2 q \rceil$. Nous dirons que r composantes d'une variable booléenne générale Y sont codées suivant π si

- ces r composantes ont la même valeur pour les états d'un même bloc de π
- à 2 blocs différents de π correspondent 2 valeurs distinctes de l'ensemble des r composantes.

Remarque sur les notations

Nous avons noté (cf. chapitre I), la variable interne (variable booléenne générale) au temps t par Y, et cette même variable au temps t-1 par y. Un sous-ensemble de composantes de Y, qui peut se réduire à une seule composante, sera notée Y_i ou y_i suivant l'instant considéré.

II-5-2 - Théorème sur l'écoulement de l'information

Soit un automate complet donné ; la composante Y_i de la variable interne Y est codée suivant une partition τ_i ; soit P un sous-ensemble de l'ensemble des composantes de Y ;
Y_i sera une fonction des composantes de P, seules, si et seulement si

$$\prod_P (\tau_j) \leq M(\tau_i)$$

ou encore

$$(\prod_P \tau_j, \tau_i)$$

est un couple de partitions permis.

III - METHODE DE LIU (cas des automates complets)

III-1 - Enoncé de la méthode

Soit un automate asynchrone complet a n *entrées* $(E_1,\ldots,E_i,\ldots,E_n)$.
Soit π_i *la partition relative à l'entrée* E_i.
Soient q_i *le nombre de blocs de* π_i *et* $r_i = \lceil \log_2 q_i \rceil$.
Soit Y_i *la variable booléenne de* r_i *composantes codées suivant* π_i;
Si $\prod_{i=1}^{n} \pi_i = 0$ *, alors la variable interne constituée par la concaténation* $(Y_1,Y_2,\ldots,Y_n)$ *est codée d'une façon acceptable pour l'automate considéré.*

Démonstration

Il suffit de remarquer, en se reportant à la contrainte impérative énoncée au chapitre I, que le sous-graphe partiel G_{i_j} est l'hyperplan $Y_i = \alpha_j$ si α_j est la valeur de Y_i spécifique au bloc B_{i_j} de π_i.

La condition $\prod_{i=1}^{n} \pi_i = 0$ assure simplement un code différent pour chaque état.

Si ce produit n'est pas nul, il convient d'ajouter une partition à nombre minimal de blocs annulant ce produit.

Exemple Haring [12]

	00	01	11	10
1	①	2	5	3
2	4	②	5	8
3	1	6	7	③
4	④	2	7	3
5	1	6	⑤	3
6	4	⑥	5	8
7	4	6	⑦	8
8	1	2	7	⑧

π_1 = (1358, 2467)

π_2 = (1248, 3567)

π_3 = (1256, 3478)

π_4 = (1345, 2678)

Il existe donc un codage à 4 composantes

	Y_1	Y_2	Y_3	Y_4
1	0	0	0	0
2	1	0	0	1
3	0	1	1	0
4	1	0	1	0
5	0	1	0	0
6	1	1	0	1
7	1	1	1	1
8	0	0	1	1

Les équations obtenues étant :

$$Y_1 = E_1 y_1 + E_2 + E_3 y_3$$
$$Y_2 = E_2 y_2 + E_3 + E_4 y_4'$$
$$Y_3 = E_1 y_1 + E_3 y_3 + E_4$$
$$Y_4 = E_2 + E_3 y_3 + E_4 y_4$$

III-2 - Théorème 1

Soit un automate asynchrone complet ; soit $\{\pi_i\}$ *l'ensemble des partitions associées aux différentes entrées ;*
soit Y *la variable interne codée par la méthode précédente ;*
soit Y_j *les composantes de* Y *codées suivant* $\pi_j \in \{\pi_i\}$,
Y_j *peut s'écrire sous la forme*

$$Y_j = E_1 \tilde{y}_1 + E_2 \tilde{y}_2 + \ldots + E_n \tilde{y}_n$$

où $\tilde{y}_i$ *représente une fonction des seules composantes de* y_i *;*
$\tilde{y}_i = 0$ ou 1, y_i, y_i' si y_i *est réduit à une seule composante.*

Démonstration

a) Considérons le sous-automate relatif à l'entrée E_j. Y_j est codée suivant π_j, c'est-à-dire $(\pi_j, 0)_{E_j}$ est un couple de partitions Mm. Ceci entraîne que (π_j, π_j) est un couple de partitions permis. Le théorème sur l'écoulement de l'information nous permet alors d'affirmer que pour ce sous-automate : $Y_j = E_j\, \tilde{y}_j$.

b) Considérons le sous-automate relatif à une entrée E_k avec $k \neq j$. Pour ce sous-automate $(\pi_k, 0)_{E_k}$ est un couple de partitions Mm et à fortiori (π_k, π_j) est un couple de partitions permis. Ce même théorème nous permet d'affirmer que $Y_j = E_k\, \tilde{y}_k$.

Exemple 1

On vérifie que dans l'exemple du paragraphe précédent, les équations ont la forme annoncée.

Exemple 2

Soit l'automate

	E_1	E_2	E_3
1	2	3	①
2	②	4	3
3	5	③	③
4	5	④	3
5	⑤	⑤	1

les partitions sont :

$\pi_1 = 12, 345$

$\pi_2 = 13, 24, 5$

$\pi_3 = 15, 234$

Y_1 sera codée suivant π_1

$Y_2 = (Y_2^*, Y_3^*)$ sera codée suivant π_2

Y_3 sera codée suivant π_3

Soit le codage

états	Y_1	Y_2 (Y_2^*, Y_3^*)	Y_3
1	0	0 0	0
2	0	0 1	1
3	1	0 0	1
4	1	0 1	1
5	1	1 1	0

et les équations peuvent s'écrire :

$$\begin{cases} Y_1 = E_1 y_1 + E_2 + E_3 y_3 \\ Y_2^* = E_1 y_1 + E_2 y_2^* \\ Y_3^* = E_1 \quad + E_2 y_3^* \\ Y_4 = E_1 y_1' + E_2 y_2^{*\prime} + E_3 y_3 \end{cases}$$

Remarques

a) ces équations ne sont pas uniques ; il existe plusieurs codages de Y_i suivant π_i.

b) le nombre des états de l'automate est, en général, différent de 2^q
Les valeurs non utilisées de Y correspondent à des états supplémentaires dont les successeurs sont choisis de façon à conserver la forme des équations de Y . L'expression "Y peut s'écrire" peut s'énoncer "Il existe un surisomorphe de l'automate donné dont les équations sont de la forme ..."

IV - METHODE DE LIU ETENDUE AUX AUTOMATES INCOMPLETS

IV-1 - Soit un automate asynchrone incomplet ; π_j , partition relative à E_j est maintenant une partition de S_{E_j} ($S_{E_j} \subseteq S$) cf. Chap. I.

L'énoncé du codage de LIU (6 III.1) reste inchangé dans ce cas. En effet, les états n'ayant pas de successeur pour E_j n'engendrent aucune contrainte impérative. Les composantes Y_j des états de $S-S_{E_j}$ ne sont pas définies au cours du codage.

Exemple

l'automate

	E_1	E_2	E_3
1	2	3	①
2	②	4	3
3	5	③	③
4	5	④	3
5	⑤	-	1

peut être codé

	Y_1	Y_2	Y_3
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1
5	1	-	0

En particulier les états de $(S-S_{E_j})$ n'ayant pas de successeurs définis pour E_j , peuvent être inclus dans un bloc quelconque de π_j. Cette remarque donne lieu aux 2 règles de simplification énoncées par LIU.

α) Soit π_j' une partition obtenue à partir d'une partition incomplète π_j par application de la remarque précédente. Si π_j' est une partition figurant déjà dans l'ensemble $\{\pi_i\}$, appelée π_k, on dit que π_k couvre π_j ; π_j est alors supprimé de l'ensemble $\{\pi_i\}$.

Exemple

$\pi_j = \{12, 345\}$ $\qquad$ $S = \{1,2,3,4,5,6\}$

$\pi_k = \{126,345\}$

β) si 2 ou plusieurs partitions incomplète peuvent être remplacées par une partition unique par la remarque précédente, cette partition est appelée intersection des 2 partitions initiales.

Exemple :

$$S = \{a,b,c,\ldots,m\}$$

$$\pi_1 : \{aF,\ b1,\ ceg,\ dhm\}$$

$$\pi_2 : \{aFj,\ b1k,\ ceg,\ dhm\}$$

peuvent être remplacées par (aFj, b1k, ceg, dhm).

IV-2 - Forme des équations dans le cas d'automates incomplets

PROPOSITION 1

Dans le cas d'automates incomplets, l'application de la méthode de LIU permet de trouver des équations de la variable interne conformes au théorème 1.
Il suffit de choisir les successeurs non définis de manière à conserver une telle forme.

PROPOSITION 2

Dans le cas d'automates incomplets où le nombre de partition $\{\pi_i\}$ a été réduit par application de la règle de simplification, l'application de la méthode de LIU permet encore de trouver des équations de la variable interne conformes au théorème 1.

Démonstration

Soient π'_k une partition intersection des partitions $\pi_{k_1}, \pi_{k_2}, \ldots$
π'_k est obtenue en ajoutant aux blocs de π_{k_i} des éléments dont les successeurs sont non définis pour cette entrée.

a) π'_k est une partition de S.

Il suffit d'attribuer des valeurs aux successeurs non définis tels que $(\pi';0)_{E_{k_i}}$ soit un couple de partitions permis, ce qui est toujours possible.

b) π'_k est une partition de $S' \subseteq S$

Dans un premier temps, les successeurs de S' sont déterminés comme précédemment en considérant la partition de S'. Les états de S-S' n'ont leur successeur défini pour aucun E_{k_i} . Dans le codage de la composante Y_k, ces états sont rattachés à un bloc quelconque de π'_k. Il est intéressant dans ce choix de tenir compte du produit $\prod_{i=1}^{n} \pi_i$ qui, rappelons-le, doit être nul.

Exemple tiré de [ ] (cas a)

	E_1	E_2	E_3	E_4
a	①	⑤	11	15
b	②	7	12	⑬
c	③	5	-,9	15
d	④	7	-,10	13
e	3	⑥	⑨	16
f	1	⑦	11	-,13
g	3	⑧	9	-,14
h	4	8	⑩	⑭
j	-,1	6	⑪	16
k	-,2	8	⑫	14
1	2	-,5	12	⑮
m	4	-,6	10	⑯

π_1 = {af ; bl ; ceg ; dhm}

π_2 = {ac ; bd ; ej ; ghk}

π_3 = {afj ; bkl ; eg ; hm}

π_4 = {acl ; bd ; ejm ; hk}

π_1 et π_3 sont inférieures à la partition $\pi_1' =$ {afj ; bkl, ceg ; dhm}

π_2 et π_4 sont inférieures à la partition $\pi_2' =$ {acl ; bdf ; ejm ; ghk}

Nous sommes alors ramenés à l'automate complet obtenu en remplaçant les tirets par les états indiqués dans le tableau.

Si on code π_1' par $\chi_1 = (y_1, y_2) = 00, 01, 11, 10$
pour les blocs énumérés de π_1'
Si on code π_2' par $\chi_2 = (y_3, y_4) = 00, 01, 11, 10$
pour les blocs énumérés de π_2' , on obtient les équations :

$$\chi_1 \begin{cases} Y_1 = (E_1+E_3)y_1+(E_2+E_4)y_3 \\ Y_2 = (E_1+E_3)y_2+E_2y_3+E_4y_3' \end{cases}$$

$$\chi_2 \begin{cases} Y_3 = E_2y_3+E_3+E_4y_3 \\ Y_4 = E_1(y_1 \oplus y_2)+E_2y_4+E_3(y_1 \oplus y_2)'+E_4y_4. \end{cases}$$

V - CONCLUSION

Si d'on cherche une synthèse rapide d'un automate asynchrone, à l'aide de circuits simples, nous disposons essentiellement de 2 méthodes : le codage de LIU et le codage utilisant une composante de la variable interne par ligne. Si l'automate possède un grand nombre de valeurs d'entrées la méthode de LIU s'avèrera moins intéressante. Il convient d'évaluer rapidement une borne supérieure du nombre de composantes de la variable interne pour cette méthode.

Quand aux méthodes de codage plus élaborées tendant à minimiser le nombre de composantes de la variable interne (méthode du chapitre V) ou à minimiser les temps de transitions (méthode TRACEY [ ]), il semble très difficile d'introduire lors de choix possibles, ce critère de simplicité des équations.

CHAPITRE - VII

RECHERCHE DE L'ISOMORPHIE DE 2 GRAPHES

INTRODUCTION

Les méthodes actuellement proposées (essentiellement UNGER, STEEN) pour rechercher l'isomorphie de 2 graphes, consiste à étudier

. Les structures des 2 graphes ("fonctions de structure").

. A faire correspondre à chaque sommet suivant la structure étudiée (nombre de descendants d'ordre successif, nombre de boucles etc...) un nombre ou une suite de nombres ("fonctions intrinsèques" ou "fonctions nodales").

. A répartir les sommets en classes suivant les résultats en tenant compte des renseignements de l'étape précédente (intersection de partitions).

. A comparer les résultats ainsi obtenus pour les 2 graphes.

. A pratiquer des essais successifs (appelés hypothèses) si les fonctions de structure étudiées ne permettent pas de conclure (procédure ENUM (UNGER)).

Nous proposons une méthode plus souple en plaçant éventuellement, une hypothèse de niveau 1 dans une étape quasi initiale, ce qui nous permet l'emploi de fonctions de structure à la fois plus élaborées et plus simples à établir et nous amène à des conclusions plus rapide que les autres méthodes.

I - RAPPEL DE DEFINITIONS

I-1 - Isomorphismes

Soient 2 graphes $G(X,\Gamma)$, $G'(X',\Gamma')$; G et G' sont isomorphes s'il existe une correspondance biunivoque entre X et X' respectant les relations Γ et Γ' ; c'est-à-dire

. pour $x,y \in X$, $x' \in X'$

$$[x <-> x' \text{ et } \Gamma(x,y)] \Rightarrow [\exists y' \in X' \text{ tel que } y <-> y' \text{ et } \Gamma'(x',y')]$$

. et pour $x \in X$, $x',y' \in X'$

$$[x <-> x' \text{ et } \Gamma'(x',y')] \Rightarrow [\exists y \in X \text{ tel que } y <-> y' \text{ et } \Gamma(x,y)]$$

Rappel

Comme précédemment $\Gamma(x,y)$ signifie que les sommets x et y de G sont reliés par une arête de G, G étant supposé être un graphe non marqué.

I-2 - Automorphismes

Un automorphisme d'un graphe est un isomorphisme de ce graphe sur lui-même.
Les automorphismes d'un graphe forment un groupe ; c'est le sous-groupe du groupe des permutations des sommets du graphe respectant la relation Γ.

I-3 - Isomorphismes et automorphismes

Tout isomorphisme entre 2 graphes, se construit en faisant le produit d'un isomorphisme donné entre les 2 graphes par un automorphisme quelconque de l'un des 2 graphes.

I-4 - Classes d'équivalence entre les sommets définis par le groupe d'automorphismes H d'un graphe

2 sommets x, y d'un graphe sont dits H-équivalents s'il existe un automorphisme de H qui fasse correspondre x et y.
Si 2 graphes sont isomorphes, les classes d'équivalence définies ici sont isomorphes.

Cas particulier : S-équivalence

2 sommets x et y de X sont S-équivalents si la permutation de x et y, tous les autres sommets étant invariants, constitue un élément de H.

II - ISOMORPHISME DE 2 GRAPHES

Nous les supposerons dans un premier temps connexes et non marqués. En effet, l'étude de graphes non connexes nous amènera à étudier les différentes composantes connexes, l'étude des graphes marqués (graphes orientés, pondérés, etc...) facilitera les méthodes employées par un supplément d'information.

II-1 - Représentation d'un graphe G(X,Γ) ; Automorphismes de ce graphe

Nous adopterons une représentation déjà employée précédemment et qui donne très rapidement certains renseignements sur la structure du graphe.

II-1-1 - Structure en couches associée à un sommet $S \in X$

S étant un sommet distingué dans X, nous associons à $G(X,\Gamma)$, la représentation en couches $\{C_o,\ldots C_i,\ldots\}$ définie par

$$S \in C_o$$

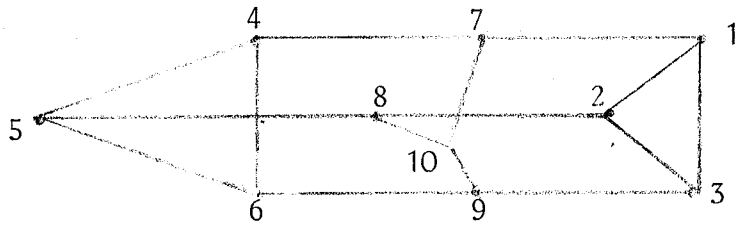
$$x \in C_i \iff d(S,x) = i$$

d étant la distance au sens habituel (S,x) dans G.
Dans cette preprésentation, chaque sommet peut être relié à des sommets de la couche précédente, à des sommets de la même couche, à des sommets de la couche suivante.

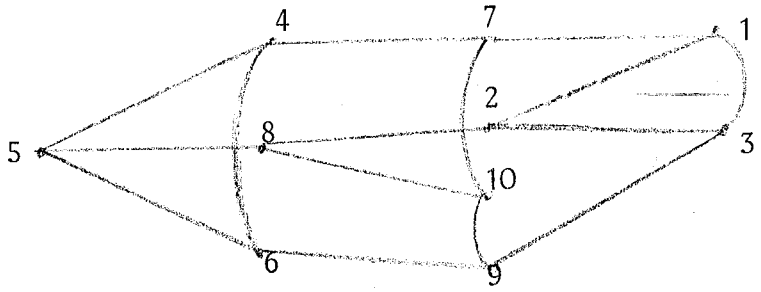
Triplets associés aux sommets {X} dans une représentation en couches

A chaque sommet on associe une suite de 3 nombres (α ,β , γ) où α est le nombre de liaisons avec la couche précédente, β est le nombre de liaisons avec la même couche, γ est le nombre de liaisons avec la couche suivante.

Exemple
Soit le graphe suivant (Vilcoq [33]) dont tous les sommets sont de degré 3



Si 5 est choisi comme sommet S, nous avons la structure et les triplets suivants :



4 : (1,1,1)	7 : (1,1,1)	1 : (2,1,0)
6 : (1,1,1)	9 : (1,1,1)	3 : (2,1,0)
8 : (1,0,2)	2 : (1,0,2)	
	10 : (1,2,0)	

Remarques

- Si tous les triplets sont de la forme (α,0,β) le graphe est bipartite (cf. Chap. 3).
- A des sommets de même degré dans le graphe initial peuvent être affectés des triplets différents.

II-1-2 - Automorphismes de G laissant S invariant ; conditions nécessaires et partitions

Soit H le groupe des automorphismes de S. Le sous-groupe de H appliquant S sur lui-même sera noté H_S . Nous avons les propriétés suivantes : (conditions nécessaires pour que 2 sommets appartiennent à une même H_S-classe)

C.N.1.

x, x' $\in$ X

si x <-> x'par un automorphisme de H_S alors x et x' appartiennent à une même couche (de la représentation associée à S).

C.N.2.

si $x \xrightarrow[h \in H_S]{} x'$ alors les triplets associés à x et x' sont les mêmes.

A chaque condition nécessaire ainsi énoncée, correspond une partition de X. La condition nécessaire 1 donne la partition de X en couches. La condition nécessaire 2 peut affiner cette partition, exemple précédent :

C_o : 5

C_1 : (4,6) (8)

C_2 : (7,9) (2) (10)

C_3 : (1,3)

A l'application de plusieurs conditions nécessaires correspond l'intersection des partitions correspondantes.
Nous appelerons $\{P_o, P_1, \ldots, P_i, \ldots\}$ les partitions des sommets des couches $\{C_o, C_1, \ldots, C_i, \ldots\}$ obtenues, soit après application de conditions nécessaires, soit après hypothèse arbitraire.

Signification de ces partitions

Nous supposons que 2 éléments de la ième couche, appartenant à 2 classes distinctes de P_i ont été distingués entre eux, soit par des propriétés différentes (condition nécessaire), soit arbitrairement (par hypothèse).

Conséquence de ces partitions

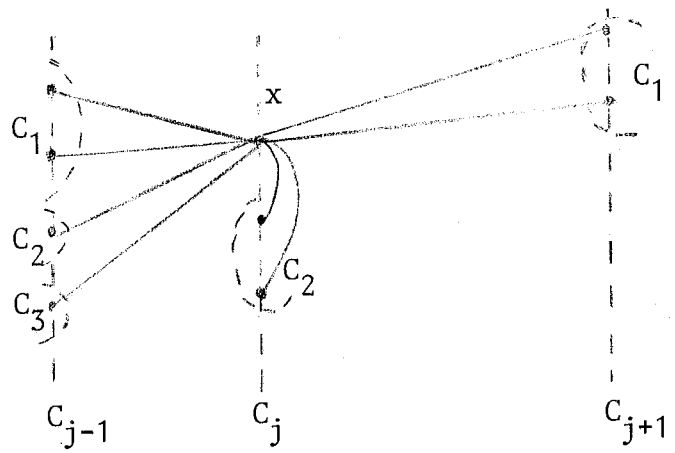
Supposons que 2 sommets x et x' appartiennent à 2 classes distinctes de P_i (ont pu être différenciés par une certaine propriété). Cette distinction engendrera des distinctions dans l'ensemble des sommets adjacents à x ou x' (voisinage de rang 1, de rang 2, etc...) et peut engendrer un ensemble de partitions $P'_o, \ldots, P'_i, \ldots$ plus fines que les précédentes. Nous allons donc étudier un algorithme rapide donnant à partir d'un ensemble de partitions $\{P_i\}$, un ensemble de partitions $\{P'_i\}$ (avec $P'_i \leq P_i$) obtenu en étudiant les conséquences des partitions $\{P_i\}$ au voisinage d'ordre 1, 2,... jusqu'à stationnarité.

II-1-3 - Algorithme d'affinement d'un ensemble de partitions $\{P_i\}$ jusqu'à stationnarité

α) Soit un ensemble de partitions $\{P_i\}$ initiales. Numérotons les classes de chacune de ces partitions ; $C_o, C_1, C_2, \ldots$ A chaque classe, correspond, par exemple, un triplet défini au paragraphe précédent.

β) Associons à chaque sommet d'une couche une suite de 3 monômes (m_1, m_2, m_3) Le monôme m_1 sera constitué par la concaténation des classes de sommets de la couche précédente, adjacents à ce sommet. Le monôme m_2 traduira de même les liaisons avec les éléments d'une même couche, le monôme m_3 avec les éléments de la couche suivante.

Exemple



à x sont associés les 3 monômes $(C_1C_1C_2C_3, C_2C_2, C_1C_1)$.
On définit ainsi une nouvelle partition $P'_j \leq P_j$ par la relation : 2 éléments sont dans une même classe si et seulement si les suites de 3 monômes (m_1, m_2, m_3) sont les mêmes pour ces 2 éléments.

γ) La partition relative à une couche, ayant été affinée par le processus précédent, les partitions relatives aux couches voisines sont examinées.

δ) Si aucun affinement n'est plus possible, par le processus précédent, les dernières partitions trouvées constituent l'ensemble $\{P_i^!\}$.

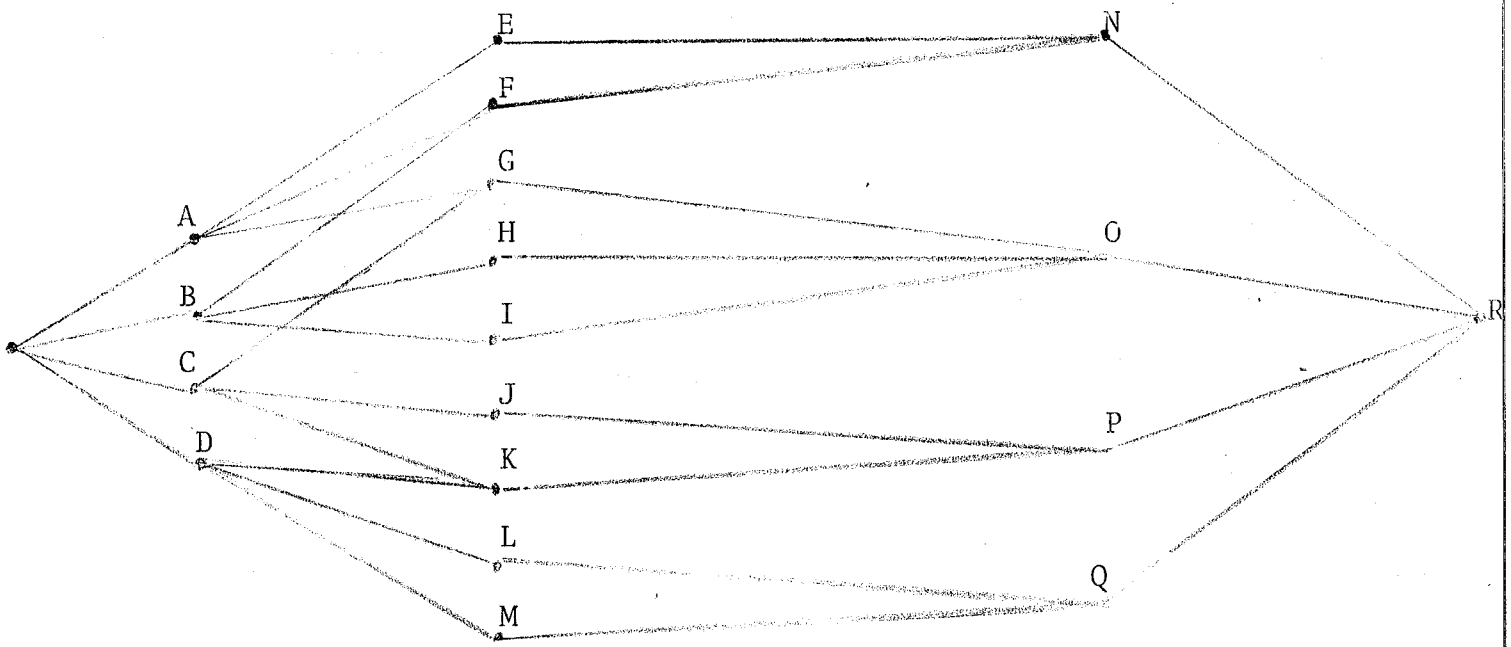
Signification des partitions $\{P_i^!\}$

Elles traduisent les distinctions les plus fines que l'on peut faire, à partir des distinctions traduites par les partitions initiales, par étude des voisinages successifs.

II-1-4 - Exemples d'application de cet algorithme ; analyse des résultats obtenus

Dans les 2 exemples traités, les partitions initiales sont les partitions générées par les conditions nécessaires 1 et 2 : partitions par couche selon le triplet défini.

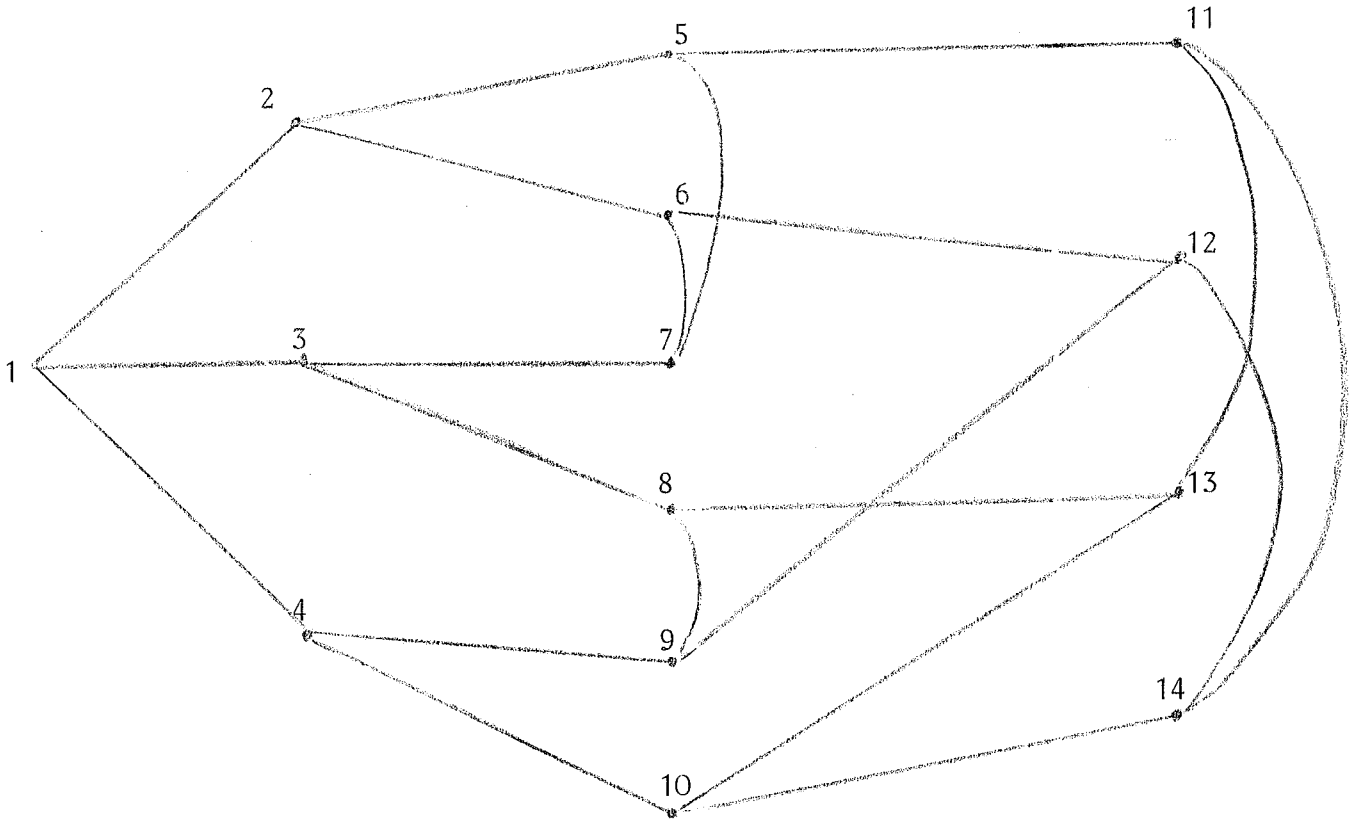
Exemple 1



	couche 1	couche 2	couche 3	couche 4
Partition initiale	ABCD C_1	EHIJLM, FGK C_1 , C_2	NPQ, O C_1 , C_2	R C_1
monômes associés	A : $C_1C_2C_2$ B : $C_1C_1C_2$ C : $C_1C_2C_2$ D : $C_1C_1C_2$			
nouvelle partition	AC, BD C_1, C_2			
monômes associés		E : C_1, C_1 H : C_2, C_2 I : C_2, C_2 J : C_1, C_1 L : C_2, C_1 M : C_2, C_1 F : C_1C_2, C_1 G : C_1C_1, C_2 K : C_1C_2, C_1		
nouvelle partition		EJ, HI, LM, FK, G C_1, C_2, C_3, C_4, C_5		
monômes associés	A : $C_1C_4C_5$ B : $C_2C_2C_4$ C : $C_1C_4C_5$ D : $C_3C_3C_4$		N : C_1C_4, C_1 P : C_1C_4, C_1 Q : C_3C_3, C_1 O : $C_2C_2C_5$, C_1	
nouvelles partitions	AC, B, D C_1, C_2, C_3		NP, Q, O C_1, C_2, C_3	

	couche 1	couche 2	couche 3	couche 4
monômes associés		E : C_1, C_1 H : C_2, C_3 I : C_2, C_3 J : C_1 C_1 L : C_3, C_2 M : C_3, C_2 F : C_1C_2, C_1 G : C_1C_1, C_3 K : C_1C_3, C_1		R : C_1
nouvelle partition		EJ, HI, LM, F, G, K C_1, C_2, C_3, C_4, C_5, C_6		
monômes associés	A : $C_1C_4C_5$ B : $C_2C_2C_4$ C : $C_1C_5C_6$ D : $C_3C_3C_6$		N : C_1C_4, C_1 P : C_1C_6, C_1 Q : C_3C_3, C_1 O : $C_2C_2C_5$, C_1	
nouvelle partition	A, B, C, D C_1, C_2, C_3, C_4		N, P, Q, O C_1 C_2 C_3 C_4	
monômes associés		E : C_1, C_1 H : C_2, C_4 I : C_2, C_4 J : C_3, C_2 L : C_4, C_3 M : C_4, C_3 F : C_1C_2, C_1 G : C_1C_3, C_4 K : C_3C_4, C_2		
nouvelle partition résultat	A, B, C, D	E, J, IH, LM, F, G, K	N, P, Q, O	R

Exemple 2



	couche 1	couche 2	couche 3
partition initiale	(2,3,4) C_1	(5,6,8,9) (7) (10) C_1 C_2 C_3	(11,14) (12, 13) C_1 C_2
monômes associés	2 : C_1C_1 3 : C_1C_2 4 : C_1C_3		
nouvelle partition	(2) (3) (4) C_1 C_2 C_3		
monômes associés		5 : C_1, C_2, C_1 6 : C_1, C_2, C_2 8 : C_2, C_1, C_2 9 : C_3, C_1, C_2 7 : C_2, C_1C_1, $\emptyset$ 10: C_3, $\emptyset$, C_1C_2	
nouvelle partition		(5) (6) (8) (9) (7) (10) C_1 C_2 C_3 C_4 C_5 C_6	
monômes associés			11 : C_1, C_1C_2 14 : C_6, C_1C_2 12 : C_2C_4, C_1 13 : C_3C_6, C_1
nouvelle partition			(11) (14) (12) (13)

Remarques

- Dans l'exemple 1, les classes trouvées sont les S_S -classes ; on peut donc affirmer que les distinctions les plus fines ont été opérées pour caractériser ce graphe.

- Chaque classe finale est caractérisée par une suite de 3 monômes. Il est important, lorsque l'on opèrera sur 2 graphes, que cette suite de 3 monômes soit définie biunivoquement à partir des classes des partitions initiales ; ce qui permettra d'établir une correspondance biunivoque entre les classes finales à partir de la correspondance biunivoque entre les classes initiales.

Il suffira pour cela de respecter la règle suivante :
Soit une partition P s'affinant en une partition $P' < P$.
En numérotant les classes de P', on respectera
. l'ordre des classes de P : seront d'abord considérés les sommets de la première classe de P etc...
. au cours de l'éclatement d'une telle classe, on respectera un ordre lexicographique des variables $(C_1, C_2, \ldots, C_n)$ figurant dans la suite des 3 monômes.

Exemple 1

La partition initiale de la 2ème couche :
EHIJLM, FGK donnera à l'étape suivante :
C_1 C_2
la partition en classes ordonnées comme suit :
EJ, LM, HI, G, FK ; en effet LM est caractérisé par C_2, C_1
C_1 C_2 C_3 C_4 C_5 HI " " " C_2, C_2

Exemple 2

Cet ordre est respecté.

En fait, à chaque étape (nouvelle partition à partir des monômes associés), nous avons une écriture unique des nouvelles classes et une caractérisation unique de ces classes. En opérant sur 2 graphes, la correspondance biunivoque possible pourra être vérifiée à chaque pas (à partir de la correspondance initiale).

II-2 - Isomorphie de 2 graphes

Considérons 2 graphes G et G' dont nous recherchons l'isomorphie éventuelle. La recherche proposée se fera par les étapes suivantes.

A - ETUDE PREALABLE DES 2 GRAPHES

Cette étude rapide comprendra

- le dénombrement des sommets, des arêtes, l'établissement de la partition de l'ensemble des sommets suivant la valeur de leur degré (généralisation aux graphes marqués tiendra compte des demi-degrés etc...) et des boucles pouvant éventuellement figurer dans le graphe. Dans les cas les plus défavorables, cette partition se réduira à une seule classe.
  Si les mêmes résultats sont trouvés pour G et G' l'étape suivante est considérée (les classes se correspondent biunivoquement, chaque classe étant caractérisée par les quantités telles que : degrés, nombre de boucles).

  A cette étude préalable, peut être substitué une étude plus poussée, (proposée par UNGER, STEEN, VILCOQ), le résultat final étant toujours une correspondance biunivoque entre les classes des partitions de l'ensemble des sommets de G et G' . Une étude rapide et grossière nous semble plus rentable pour la méthode proposée.

B - DEUXIEME ETAPE

Considérons une classe C, ayant un nombre minimal d'éléments, de la partition précédemment trouvée de l'ensemble des sommets de G et l'image C' de cette classe dans G'.

Soit S un sommet de C ;

* choisissons un sommet S' dans C' ; nous cherchons si S' peut être image de S par une isomorphie cherchée. Pour cela, étudions les structures en couches associées à S et S' dans G et G' .

α) Ces structures en couches doivent avoir

. même nombre de couches

. même nombre d'éléments dans une couche

. même partition initiale des éléments d'une couche suivant la valeur du triplet associé. On numérotera de la même façon les classes correspondantes de G et G' ayant même valeur de ce triplet.

Si ces conditions ne sont pas remplies, retour à * (nouveau choix de S').

β) Les partitions initiales sont affinées à l'aide de l'algorithme énoncé au paragraphe II-1-3.

S'il n'y a pas correspondance biunivoque entre les partitions obtenues à chaque étape (même ensemble de monômes caractéristiques pour les classes obtenues), retour à * (choix de S'). Supposons cette correspondance possible.

Si les classes ainsi obtenues se réduisent à un seul sommet, l'isomorphie cherchée est établie. Dans le cas contraire, il s'agit soit de H_S-classes maximales de plus d'un élément, soit d'une partition supérieure à cette partition en H_S-classes maximales.

γ) Hypothèses d'ordre supérieur : identification progressive des classes de plus d'un élément.

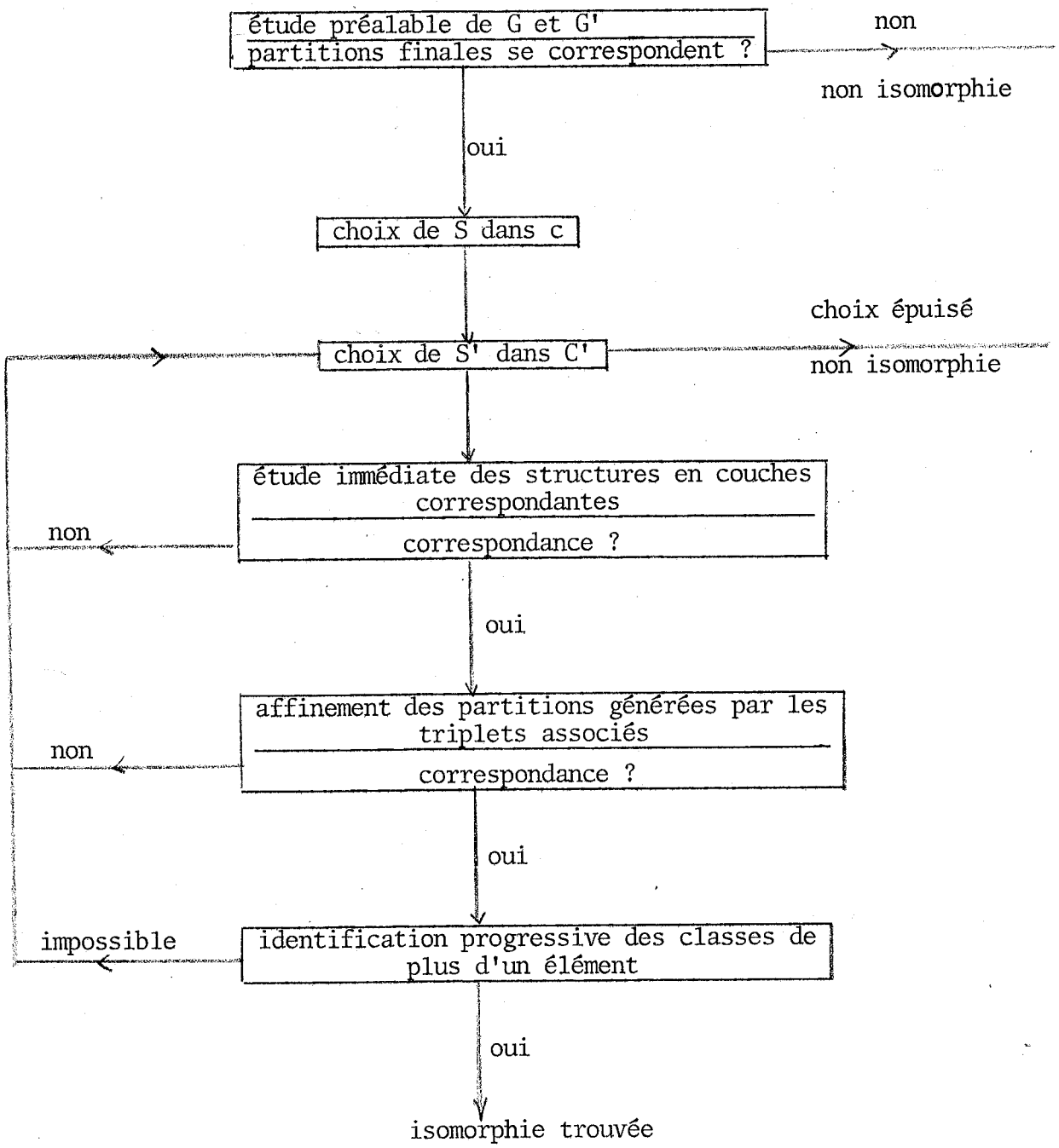
On considère dans la 1ère couche (puis 2ème etc...) une classe de plus d'un élément dans G , et la classe image dans G'.

** Soient x et x' 2 éléments de ces 2 classes, que nous essayons de mettre en correspondance.
Ce choix correspond à un affinement des partitions dans G et G' (x et x' constituent une classe unique dans G et G' numérotée de la même façon dans G et G'). Ces partitions sont affinées par l'algorithme du § II-1-3. Si les classes finales se correspondent biunivoquement nous avons

- soit résultat final cherché si ces classes se réduisent à un seul élément
- soit passage à l'hypothèse suivante (2 éléments sont mis en correspondance dans 2 classes correspondantes).

Si les classes finales ne se correspondent pas, retour à l'hypothèse précédente : nouveau choix de x et x' ou retour à 2 classes précédemment considérées pour lesquelles les hypothèses possibles n'auront pas été épuisées.
Si les hypothèses de l'étape γ sont épuisées sans avoir trouvée une correspondance biunivoque cherchée, retour au choix de S' (*).

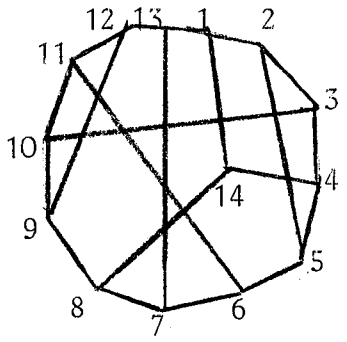
D'où l'organigramme suivant

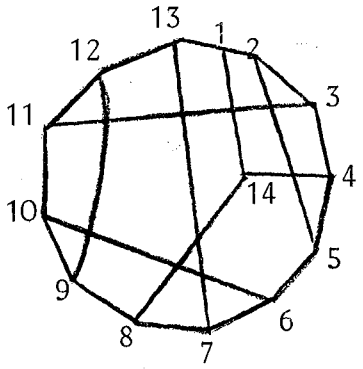

étude préalable de G et G'
partitions finales se correspondent ?
non
non isomorphie
oui
choix de S dans c
choix épuisé
choix de S' dans C'
non isomorphie
étude immédiate des structures en couches correspondantes
correspondance ?
non
oui
affinement des partitions générées par les triplets associés
correspondance ?
non
oui
identification progressive des classes de plus d'un élément
impossible
oui
isomorphie trouvée

II-3 - Exemples

Exemple 1 (STEEN) [28]

Soient les 2 graphes suivants dont nous recherchons l'isomorphie

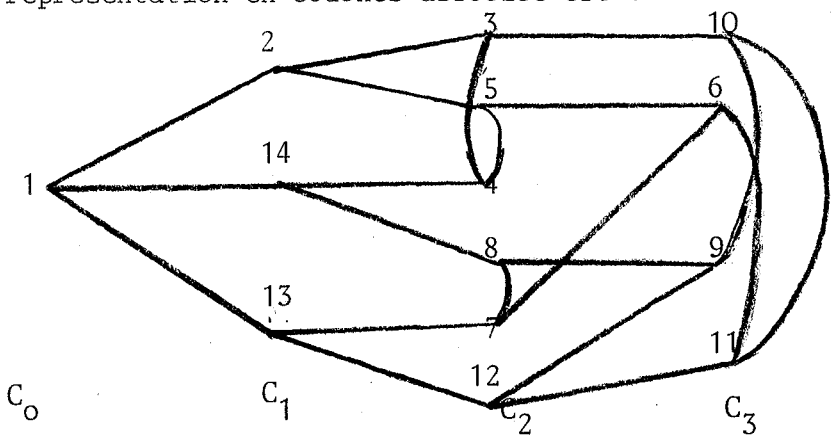




G et G'sont homogènes, chaque sommet étant de degré 3

Choisissons le sommet 1 de G comme sommet S

La représentation en couches associée est :



et la partition

des sommets suivant la valeur du triplet :

(2,14,13)	(3,5,7,8)	(4)	(12)	(10,11)	(6,9)
(1,0,2)	(1,1,1)	(1,2,0)	(1,0,2)	(1,2,0)	(2,1,0)

Choix de S'

. Le sommet 1 de G' considéré comme sommet S' donne une représentation en couches comprenant 5 couches.

. Les sommets 2, 3, 4, 5 de G' donnent une couche C_2 comprenant 5 éléments (au lieu de 6) dans la structure en couches correspondante.

. Le sommet 6 donne le même nombre d'éléments par couches mais une répartition différente suivant la valeur du triplet dans la 2ème couche.

. Le sommet 7 de G' donne une 2ème couche à 5 éléments.

. Le sommet 8 donne une représentation en couches avec même nombre d'éléments par couches et même partition initiale.

Affinement des partitions initiales

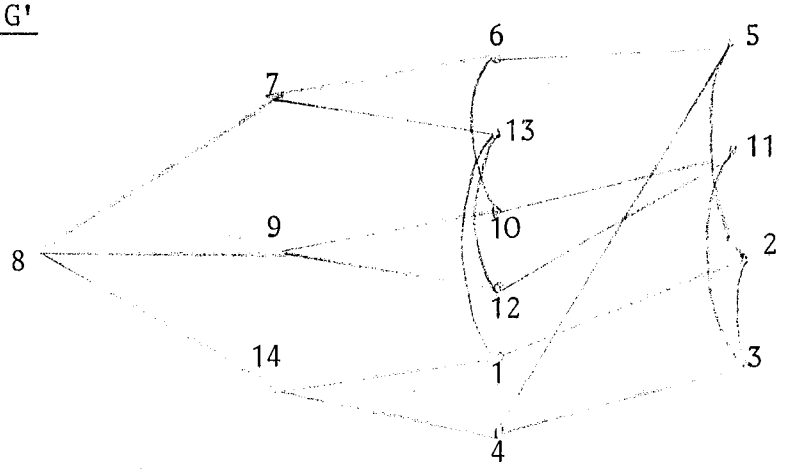
Graphe G

	P_o : (2, 13,14) C_1	(3,5,8,7) (4) (12) C_1 C_2 C_3	(10,11) (6,9) C_1 C_2
	2 : Ø; Ø, C_1C_1 13: Ø, Ø, C_1C_3 14: Ø, Ø, C_1C_2		
nouvelle partition	(2) (14) (13) C_1 C_2 C_3		
		3 : C_1, C_2, C_1 5 : C_1, C_2, C_2 7 : C_3, C_1, C_2 8 : C_2, C_1, C_2 4 : C_2, C_1C_1, Ø 12 : C_3, Ø, C_1C_2	
nouvelle partition		(3) (5) (8) (7) (4) (12) C_1 C_2 C_3 C_4 C_5 C_6	

			10 : C_1, C_1C_2, $\emptyset$ 11 : C_6, C_1C_2, $\emptyset$ 6 : C_2C_4, C_1, $\emptyset$ 9 : C_3C_6, C_1, $\emptyset$
nouvelle partition			(10) C_1 (11) C_3 (6) C_3 (9) C_4

L'ordre lexicographique défini précédemment a été respecté dans l'écriture des nouvelles classes.

Graphe G'



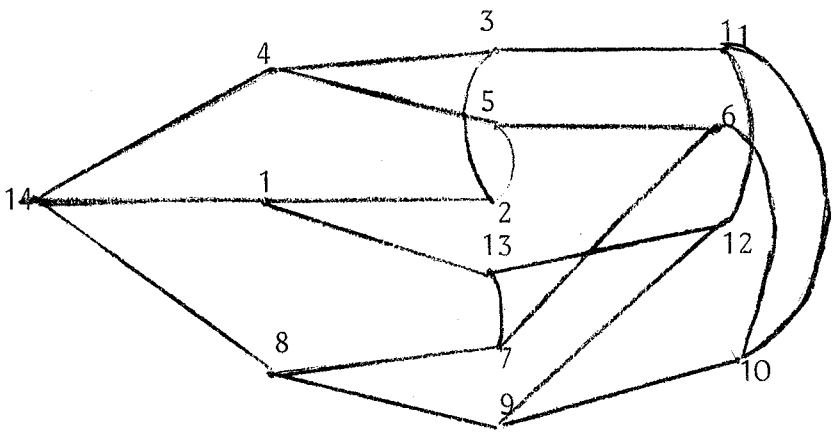
(les classes initiales sont numérotées comme dans G suivant la valeur du triplet)

(7,9,14) C_1	(6,10,12,1) (13) (4) C_1 C_2 C_3	(2,3) (5,11) C_1 C_2
7 : C_1C_2, Ø 9 : C_1C_1, Ø 14 : C_1C_3, Ø		
(9) (7) (14) C_1 C_2 C_3		
	6 : C_2, C_1, C_2 10 : C_1, C_1, C_2 12 : C_1, C_2, C_2 1 : C_3, C_2, C_1	

les suites de monômes associées à ces 4 sommets ne sont pas les mêmes que dans G ; correspondance impossible : retour au choix de S'.

- Les sommets 9, 10, 11, 12 donnent un nombre de sommets en couches C_2 égal à 5
- Le sommet 13 donne un même nombre de sommets par couche mais unepartition initiale suivant la valeur du triplet différent.
- Sommet 14 de G' même nombre d'éléments par couche, même partition initiale.

Affinement de cette partition



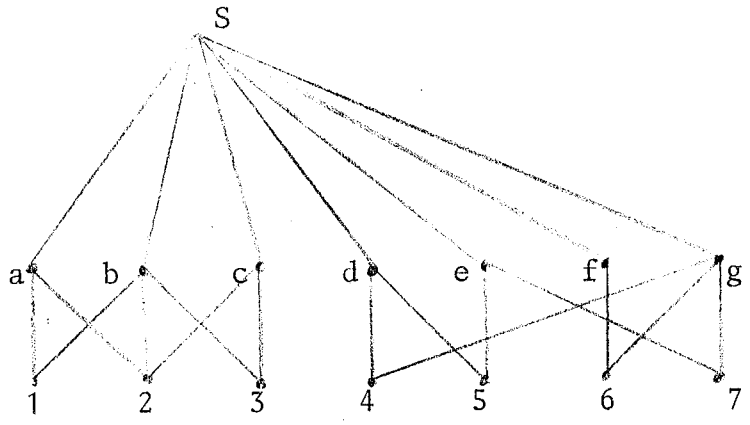
(1,4,8) C_1	(13,3,5,7) (2) (9) C_1 C_2 C_3	(11,10) (6,12) C_1 C_2
1 : C_1C_2 4 : C_1C_1 8 : C_1C_3 (4) (1) (8) C_1 C_2 C_3		
	13 : C_2, C_1, C_2 3 : C_1, C_2, C_1 5 : C_1, C_2, C_2 7 : C_3, C_1, C_2 2 : C_2, C_1C_1, $\emptyset$ 9 : C_3, $\emptyset$, C_1C_2	
	(3) (5) (13) (7) (2) (9) C_1 C_2 C_3 C_4 C_5 C_6	
		11 : C_1, C_1C_2, $\emptyset$ 10 : C_6, C_1C_2, $\emptyset$ 6 : C_2C_4, C_1, $\emptyset$ 12 : C_3C_6, C_1, $\emptyset$
		(11) (10) (6) (12) C_1 C_2 C_3 C_4

Les monômes caractéristiques sont les mêmes d'où l'isomorphie cherchée :

G	1	2	14	13	3	5	8	7	4	12	10	11	6	9
G'	14	4	1	8	3	5	13	7	2	9	11	10	6	12

Nous nous plaçons donc comme dans STEEN dans le cas le plus défavorable où le sommet S' satisfaisant est testé en dernier lieu. Les procédés de comptage sur les structures en couches correspondantes et la partition initiale ne laissent subsister que 2 sommets (8 et 14) de G'. Le sommet 8 est éliminé par l'algorithme d'affinement. Cet algorithme donne la solution pour 14.

Exemple 2



Graphe G

Soit 2 graphes G et G' ayant même nombre de sommets, d'arêtes et même partition de l'ensemble des sommets suivant la valeur du degré. Supposons ces 2 graphes isomorphes et cherchons le nombre maximal d'étapes nécessaires pour reconnaître cette isomorphie. Le choix de S et S' est ici évident.

L'étude immédiate des structures en couches donne une partition en classes identiques aux couches et ne permet pas d'affinement possible. L'étape 1 est donc immédiatement abordée.

Pour comprendre ce qui se passera par application de l'algorithme, étudions un tel graphe.

a) Nous pouvons considérer comme système de générateurs de H_S :

α) [abc] [123]
β) [dF] [46] [57]
γ) [eg] [45] [67]
δ) [deFg] [4567]
ε) [a] [b] ... [1] [2] ...

les H_S classes maximales étant

(abc) (deFg)
(123) (4567)

b) Distinguons un sommet parmi les sommets de la première couche ; aux 7 possibilités correspondent les 7 partitions suivantes, qui, après affinement donnent les résultats suivants :

		Couche 1	Couche 2
P_a = (a,bcdeFg)	→	a,bc,deFg	12,3,4567
P_b = (b,acdeFg)	→	b,ac,deFg	13,2,4567
P_c = (c,abdeFg)	→	c,ab,deFg	23,1,4567
P_d = (d,abcdFg)	→	abc,d,eg,F	123,45,67
P_e = (e,abcdFg)	→	abc,dF,eg	123,46,57
P_f = (f,abcdeg)	→	abc,d,eg,f	123,45,67
P_g = (g,abcdef)	→	abc,dF,ef	123,46,57

Application de l'algorithme

- Supposons que le sommet a soit considéré dans G.
  Si dans G', nous essayons de mettre en correspondance les sommets images de d, e, f, g nous obtenons des partitions affinées non comparables.
  Seuls les sommets images de a, b, c donneront mêmes partitions affinées.

L'hypothèse suivante portant sur la classe bc sera toujours satisfaisante et permettra de mettre en correspondance les sommets a, b, c, 1, 2, 3.

- Classes restantes

Par le même raisonnement que précédemment, nous aurons 2 images possibles pour d (sommets correspondant à d ou f). Une image étant choisie, toutes les hypothèses suivantes (classe eg) sont satisfaisantes et amènent à une isomorphie cherchée.

II-4 - Comparaison avec les autres méthodes

- L'écriture des triplets associés à un sommet, revient à considérer comme fonction de structure les liaisons d'un sommet à distance i de S avec les sommets à distance i, i-1, i+1 de S. La fonction intrinsèque (STEEN) ou fonction nodale (UNGER) associée n'a pas été traduite par un seul nombre (difficulté d'une correspondance biunivoque évoquée par Unger) mais par des suites de nombres. Néanmoins, l'introduction de ces suites de nombres en machine doit être résolue.

- L'écriture de monômes caractéristiques revient à considérer la même fonction de structure. La fonction intrinsèque ou nodale tient compte des partitions précédentes des couches voisines (intersection de partitions chez STEEN) et se traduira de même par des suites de nombres ou de classes.

- Le processus d'affinement qui consiste à épuiser les conséquences des distinctions faites sur une couche, revient à considérer comme fonction de structure les voisinages successifs des classes d'une partition en tenant compte à chaque pas des informations précédentes (intersection de partitions ou "affinement" de la partition).

La méthode proposée se différencie des précédentes par

- une hypothèse initiale de niveau 1 (STEEN) ou une correspondance évidente (S et S') sans avoir épuisé les autres moyens d'investigation.

- ce choix de S et S' permet alors l'emploi de fonctions de structures beaucoup plus élaborées et de fonctions nodales faciles à établir, donnant des conclusions rapides. En particulier, nous évitons les fonctions "ancestors" et "descendants" spécialement longues à calculer pour $N \geq 4$ (Unger) ; seule, la recherche de "shells" identiques aux couches subsiste et pour certains sommets seulement.

- les hypothèses finales (essais successifs avec back-tracking, "procédure ENUM" chez Unger) sont diminuées par rapport aux autres méthodes et souvent supprimées.

En conclusion, pour tous les exemples traités, le gain en nombre d'opérations et en temps est très net. A titre d'exemple, nous traitons, à nouveau, l'exemple 1 par la méthode de Unger.

Exemple 1 traité par la méthode de UNGER

1) Entrée des graphes G et G' par la liste des successeurs de chaque sommet

partition PNPL (suivant les degrés des sommets)

1,2,.....,14 (1,2,.....,14)

G G'

2) N = 2

. Nombre de successeurs de chaque sommet à la distance 2 ("fonction ancestor, descendants")

G : 1(7) 2(6) 3(6) 4(6) 5(6) 6(7) 7(7) 8(7) 9(6) 10(6) 11(6) 12(6)...

G': 1(7) 2(6) 3(6) .....

partitions obtenues (les classes sont mises en correspondance)

PNPL : 1,6,7,8,13,14 (1,6,7,8,13,14) 2,3,4,5,9,10,11,12(2,3,4,5,9,10,11,12)

G G'

. Nombre de sommets à distance minimale 2 ("shell")

G : 1(6) 2(5) 3(5) .....

G': 1(6) 2(5) .....

PNPL : inchangé

. Nombre de circuits de longueur 2 : néant.

. Fonctions extend

- les classes de G et G' sont numérotées ; chaque sommet est représenté par sa classe ; à savoir

G	1(1) 2(2) 3(2) 4(2) 5(2) .....
G'	1(1) 2(2) 3(2) 5(2) 6(1)

fonction extend (somme des numéros des sommets adjacents)

G	1(4) 2(5) 3(6) 4(5) .....
G'	1(4) 2(5) 3(6) 4(5) .....

d'où partition 'PNPL" :

1,8,13,14(1,8,13,14)2,4,5,6,9,11,12(2,4,5,6,9,10,12)3,10(3,11)7(7)

3) N = 3

Fonction ancestor et descendant (très longues à calculer)

G : 1(12) 2(10) 3(9) 4(9) 5(9) 6(9) .....

G': 1(11) 2(9) 3(9) 4(10) 5(9) 6(9) .....

partition résultante

1,13(8,14) 8(13)14(1) 2,12(4,9) 4,5,6,9,11(2,5,6,10,12) 3(3)...

1 2 3

les classes sont numérotées dans cet ordre ; aux sommets correspondent alors les nombres suivants :

G	1(1) 2(4) 3(6) 4(5) 5(5) 6(5) 7(8) 8(2) 9(5) 10(7) 11(5) .....
G'	1(3) 2(5) 3(6) 4(4) 5(5) 6(5) 7(8) 8(1) 9(4) 10(5) 11(7) .....

Fonctions EXTEND

G	1(8) 2(12) 3(16) 4(14) 5(14) 6(18) 7(8) .....
G'	1(8) 2(14) 3(16) 4(12) 5(14) 6(18) 7(8) .....

partition résultante

1(14) 13(8) 8(13) 14(1) 2(4) 12(9) 4,5(2,5) 6(6) ...

fonction shell nombre de sommets à distance minimale 3

G	1(4) 2(4) .....4(4) 5(4) .....
G'	... 2(4) ..... 5(4) .....

n'affine plus la partition (ne permet pas de distinguer 4 et 5 de G).

Fonction circuit → pas de résultat.

4) Hypothèse

2 hypothèses possibles

l'hypothèse 4(5) 5(2) donne des fonctions EXTEND incompatibles pour G et G'

l'hypothèse 4(2) 5(5) donne la solution.

En conclusion, nous avons des recherches de structure beaucoup plus longues (recherche de tous les descendants d'ordre n, pour tous les sommets, des descendants à distance minimale n, de tous les circuits de longueur n), et nous aboutissons, néanmoins, à une obligation de choix.

Le même exemple, traité par la méthode STEEN étudie 14 hypothèses avec établissement de 55 partitions.

CHAPITRE - VIII

EXTENSION AUX GRAPHES MARQUES : ISOMORPHIE DE 2 GRAPHES MARQUES

La méthode reste la même que pour les graphes non marqués. L'ensemble des marques conduit à des comparaisons plus fastidieuses (comparaison de suites de nombres) mais permet de conclure plus rapidement grâce à des suppléments d'information.

I - DEFINITIONS

I-1 - Graphe marqué

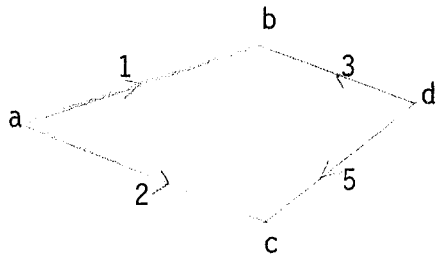
Un graphe marqué est un graphe dans lequel à chaque liaison entre 2 sommmets est associé un ensemble de caractéristiques ou marques (sens, pondération etc...).

I-2 - Caractérisation des liaisons

L'ensemble des caractéristiques d'un graphe est ordonné préalablement (ex : on considèrera d'abord le sens, puis telle pondération etc...). Aux n états possibles d'une caractéristique donnée sont associés n nombres entiers distincts, l'absence de cette caractéristique est traduite par le symbole φ .

Soit un graphe G ; soit p le nombre de caractéristiques associées à G ; à chaque liaison peut être associée une suite de p symboles (nombres ou symbole φ) $\alpha_1 \ldots \alpha_i \ldots \alpha_p$ où α_i traduit l'état ou l'absence de la ième caractéristique. Cette suite est appelée suite caractéristique de la liaison.

Exemple :



Nous considérons les 2 caractéristiques dans l'ordre suivant : sens, poids. Codons ces caractéristiques de la façon suivante :

sens : gauche droite : 1
droite gauche : 2

pondération : entiers indiqués sur la figure.

A l'arête (a,b) est alors associé le doublet (1,1)
A l'arête (c,d) est alors associé le doublet (2,5).

I-3 - Rangement lexicographique d'un ensemble de suites caractéristiques

Considérons 2 suites caractéristiques distinctes $S = (\alpha_1, \ldots, \alpha_p)$ et $S' = (\alpha'_1, \ldots, \alpha'_p)$. Dans l'ensemble ordonné des caractéristiques du graphe, considérons la première caractéristique dont diffèrent ces 2 suites ; soient α_i et α'_i les symboles de cette caractéristique dans S et S'.

Si $\alpha_i < \alpha'_i$ (nombres entiers) ou $\alpha_i = \varphi$ S sera rangée avant S'

Si $\alpha'_i = \varphi$ où $\alpha'_i < \alpha_i$ S' sera rangée avant S.

Un ensemble de suites caractéristiques peut être rangé suivant cette règle. Ce rangement ne sera pas unique s'il existe des suites identiques. Un tel rangement sera appelé rangement lexicographique des suites.

I-4 - Structures en couches, partition initiale, rangement des sommets à l'intérieur d'une couche

I-4-1 - La structure en couches d'un graphe marqué, par rapport à un sommet S est établie comme précédemment sans tenir compte des marques. Un sommet donné, aura α liaisons avec la couche précédente

β liaisons avec la même couche

γ liaisons avec la couche suivante.

I-4-2 - La partition initiale d'une structure en couches est définie comme suit :

2 sommets d'une couche appartiennent au même bloc de la partition de l'ensemble des sommets de cette couche si et seulement si

a) les triplets associés α,β,γ sont les mêmes

b) l'ensemble des suites caractéristiques des α liaisons (respectivement β,γ) est le même pour les 2 sommets.

I-4-3 - Rangement des sommets à l'intérieur d'une couche

Pour faciliter la comparaison de 2 graphes, les sommets à l'intérieur d'une couche sont rangés d'après les règles suivantes :

a) dans un ordre lexicographique des triplets (α,β,γ) associés (même définition que § I.3).

b) si ces triplets sont les mêmes, on considèrera un rangement lexicographique des α suites caractéristiques des liaisons avec la couche précédente.

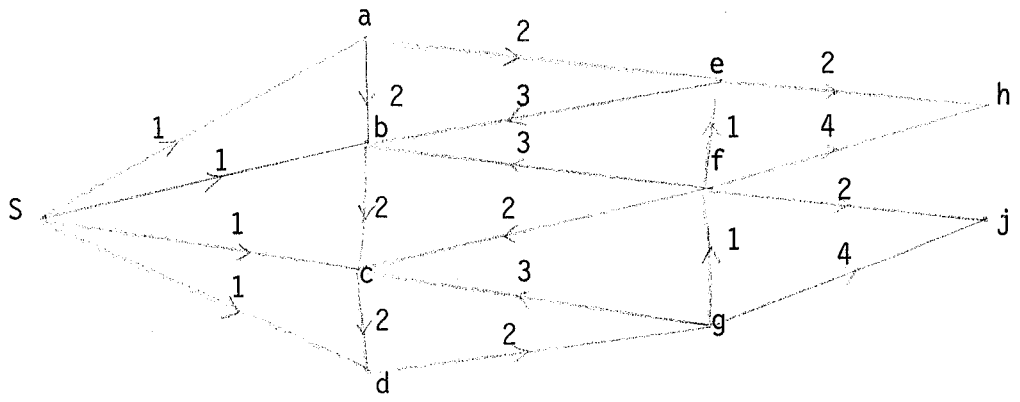
c) Si les caractéristiques a et b sont les mêmes on considère un rangement lexicographique de β suites caractéristiques des liaisons avec la même couche.

d) Si les caractéristiques a.b.c sont les mêmes, on considère les γ suites caractéristiques.

e) L'ordre des sommets à l'intérieur d'une classe de la partition initiale est arbitraire.

Exemple 1

Soit le graphe



Considérons les caractéristiques dans l'ordre : sens, pondération ; le sens étant codé comme suit :

arc aboutissant au sommet considéré : 1
arc partant du sommet considéré : 2

Sommets		α	β	γ
Couche C_o	S	4 | 1,1 1,1 2,1 2,1	0	0
C_1	a	1(1,1)	1(2,2)	1(2,2)
	d	1(2,1)	1(1,2)	1(2,2)
	b	1(1,1)	2 | 1,2 2,2	2 | 1,3 1,3
	c	1(2,1)	2 | 1,2 2,2	2 | 1,2 1,3
C_2	e	2 | 1,2 2,3	1(1,1)	1(2,2)
	g	2 | 1,2 2,3	1(2,1)	1(2,4)
	f	2 | 2,2 2,3	2 | 1,1 2,1	2 | 2,2 2,4
C_3	h	2 | 1,2 1,4	0	0
	i	2 | 1,2 1,4	0	0

d'où la partition initiale S

(a) (d) (b) (c)
(e) (g) (f)
(h_i)

I-4-1 - Affinement des partitions initiales

A chaque liaison est attribuée une caractéristique supplémentaire : la classe du sommet constituant l'autre extrémité. Une liaison est alors caractérisée par une suite de p+1 nombres. Le processus d'affinement est conduit comme précédemment, à partir de la partition initiale en tenant compte de cette nouvelle caractéristique, jusqu'à stationnarité.

Exemple précédent

partition initiale

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(hi)
C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_1

Considérons la dernière couche :

$$h : 2 \left| \begin{matrix} 1, 2, C_1 \\ 1, 4, C_2 \end{matrix} \right.$$

$$i : 2 \left| \begin{matrix} 1, 2, C_2 \\ 1, 4, C_3 \end{matrix} \right.$$

La partition triviale est alors trouvée.

II - RECHERCHE DE L'ISOMORPHIE DE 2 GRAPHES MARQUES

L'organigramme reste le même que précédemment :

α) L'étude préliminaire consistera à compter le nombre de sommets, de liaisons, à établir la partition initiale suivant le degré des sommets (sans tenir compte des marques).
Néanmoins, en présence d'une marque ne prenant que 2 états possibles (ex : sens), il est facile d'en tenir compte à cette étape et d'affiner cette partition préliminaire (partition suivant les demi-degrés). Le résultat étant le même pour les 2 graphes on procède au choix de S et S' dans 2 classes correspondantes ayant un nombre minimal d'éléments.

β) Pour un choix de S et S' , les structures en couches sont construites ;

- on compare le nombre de couches, le nombre d'éléments par couches dans les 2 graphes.

- on établit pour les 2 graphes, les partitions des sommets d'une couche suivant la valeur des triplets α,β,γ. Pour faciliter la comparaison, on pourra ranger les blocs d'une telle partition dans l'ordre lexicographique de la suite de nombres (α,β,γ) (cf. § I-3).

- on affine les partitions précédentes en considérant pour un bloc donné les α(puis β,γ) suites caractéristiques ; on rangera, également, en vue de faciliter la comparaison entre les 2 graphes ces liaisons dans l'ordre lexicographique des suites caractéristiques (cf. § I-3).

Une correspondance impossible dans l'une de ces étapes renvoie au choix de S et S'.
A la fin de cette étape, les partitions initiales sont établies et les classes des 2 graphes sont mises en correspondance.

γ) Affinement des partitions initiales

On procède à l'affinement des partitions initiales pour les 2 graphes, les comparaisons portant cette fois sur des suites caractéristiques de p+1 termes.

Ordre des blocs au cours d'un affinement de partitions :

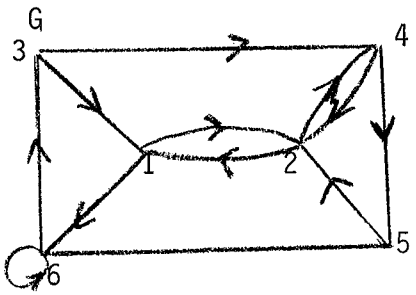
- Comme précédemment, l'ordre des blocs de la partition de départ est respecté.

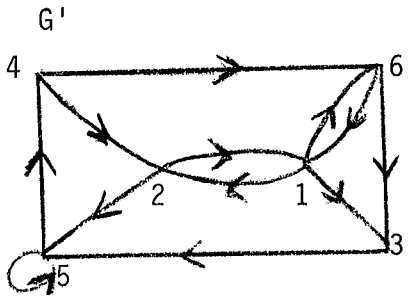
- Au cours de l'éclatement d'un bloc, on rangera selon l'ordre lexicographique des monômes associés ; un monôme associé à un sommet est constitué par la suite des $(p+1)^{è}$ caractéristiques des liaisons, les liaisons étant rangées comme indiqué au cours de l'établissement des partitions initiales.

Une correspondance impossible renvoie au ~~choix~~ de S et S' ; si la correspondance sommet à sommet n'est pas trouvée, l'étape des hypothèses est abordée.

δ) Les hypothèses se font, comme pour les graphes non marqués, en mettant en correspondance arbitraire 2 sommets appartenant à des classes de plus d'un élément.

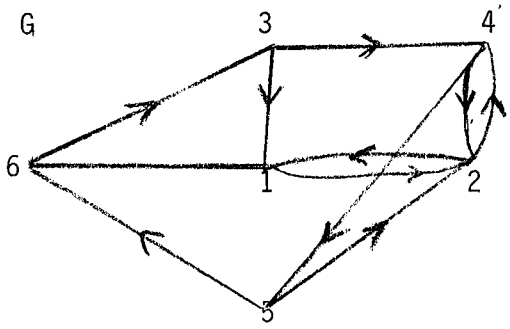
Exemple 2 (tiré de STEEN)

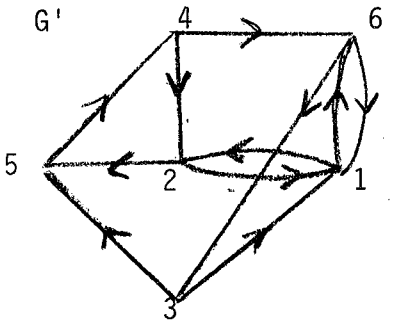




. Choix de S et S' : évident ici, par la présence d'une boucle pour les sommets 5 et 6

. Représentation en couches : (il est inutile de faire figurer les boucles)





partitions initiales le sens est codé comme dans l'exemple 1)

			α	β	γ
Graphe G	couche C_o	sommet 6 :	3 | 1 | 1 | 2	0	0
	couche C_1	sommet 5 :	1(2)	0	2 | 1 | 2
		sommet 3 :	1(1)	1(2)	1(2)
		sommet 1 :	1(2)	1(1)	2 | 1 | 2
	couche C_2	sommet 4 :	2 | 1 | 2	2 | 1 | 2	0
		sommet 2 :	3 | 1 | 1 | 2	2 | 1 | 2	0
Graphe G'	couche C_o	sommet 5 :	3 | 1 | 1 | 2	0	0
		sommet 3 :	1(2)	0	2 | 1 | 2
		sommet 4 :	1(1)	1(2)	1(2)
		sommet 2 :	1(2)	1(1)	2 | 1 | 2
		sommet 6 :	2 | 1 | 2	2 | 1 | 2	0
		sommet 1 :	3 | 1 | 1 | 2	2 | 1 | 2	0

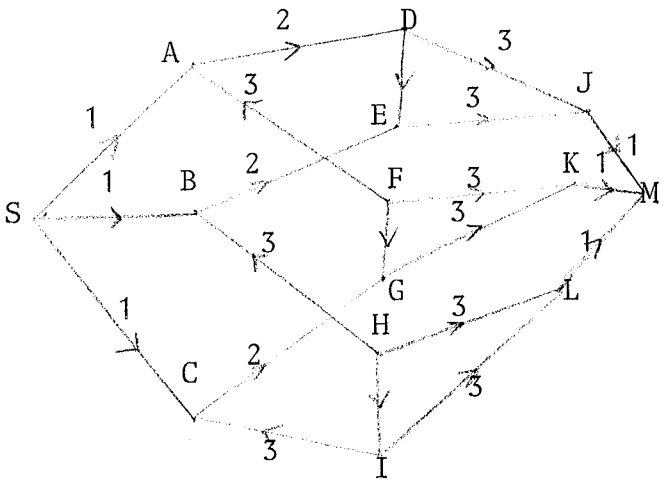
Ces partitions initiales (partitions triviales) donnent immédiatement l'isomorphie cherchée, à savoir

G	6	3	1	5	4	2
G'	5	4	2	3	6	1

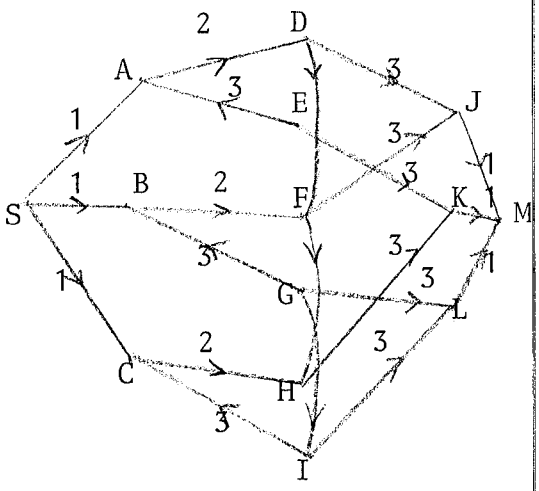
Exemple 3

Soient les 2 graphes G et G'

Graphe G



Graphe G'



- les sommets S sont facilement distingués dans G et G' et mis en correspondance (ce sont les seuls sommets de degré extérieur 3, les liaisons correspondants étant de poids 1).
- les structures en couches de G et G' de sommet référence S ont même nombre de couches, même nombre d'éléments par couches.

- Etablissement de partitions initiales

sommets	α	β	γ	
S	$3\left\vert\begin{matrix}(2,1)\\(2,1)\\(2,1)\end{matrix}\right.$	0	0	
A	$1(1,1)C_1$	0	$2\left\vert\begin{matrix}1,3,C_4\\2,2,C_2\end{matrix}\right.$	
B	$1(1,1)C_1$	0	$2\left\vert\begin{matrix}1,3,C_4\\2,2,C_1\end{matrix}\right.$	→ C,B,A $c_1c_2c_3$
C	$1(1,1)C_1$	0	$2\left\vert\begin{matrix}1,3,C_3\\2,2,C_1\end{matrix}\right.$	
E	$1(1,2)C_3$	$1(1,\emptyset)C_2$	$1(2,3)C_1$	
G	$1(1,2)C_1$	$1(1,\emptyset)C_4$	$1(2,3)C_1$	
D	$1(1,2)C_3$	$1(2,\emptyset)C_1$	$1(2,3)C_1$	→ G,E,D,I,H,F
I	$1(2,3)C_1$	$1(1,\emptyset)C_4$	$1(2,3)C_1$	$c_1\ c_2c_3c_4c_5c_6$
F	$1(2,3)C_3$	$1(2,\emptyset)C_1$	$1(2,3)C_1$	
H	$1(2,3)C_2$	$1(2,\emptyset)C_3$	$1(2,3)C_1$	
J	$2\left\vert\begin{matrix}(1,3)C_2\\(1,3)C_3\end{matrix}\right.$	0	$1(2,1)$	
K	$2\left\vert\begin{matrix}(1,3)C_1\\(1,3)C_6\end{matrix}\right.$	0	$1(2,1)$	→ K,J,L $c_1c_2c_3$
L	$2\left\vert\begin{matrix}(1,3)C_4\\(1,3)C_5\end{matrix}\right.$	0	$1(2,1)$	
M	$3\left\vert\begin{matrix}(1,1)\\(1,1)\\(1,1)\end{matrix}\right.$	0	0	

En tenant compte de l'ordre indiqué, la partition initiale est :

C_o	C_1	C_2	C_3	C_4
S,	A B C,	EG, D, I, FH,	JKL,	M
c_1	c_1	c_1 c_2 c_3 c_4	c_1	c_1

Graphe G'

Sommets	α	β	γ	
S	$3\left\vert\begin{matrix}(2,1)\\(2,1)\\(2,1)\end{matrix}\right.$	0	0	
A	$1(1,1)C_1$	0	$2\left\vert\begin{matrix}1,3\ C_4\\2,2\ C_2\end{matrix}\right.$	
B	$1(1,1)C_1$	0	$2\left\vert\begin{matrix}1,3\ C_4\\2,2\ C_1\end{matrix}\right.$	→ C,B,A $c_1c_2c_3$
C	$1(1,1)C_1$	0	$2\left\vert\begin{matrix}1,3\ C_3\\2,2\ C_1\end{matrix}\right.$	
F	$1(1,2)C_2$	$1(1,\emptyset)C_2$	$1(2,3)C_1$	
H	$1(1,2)C_1$	$1(1,\emptyset)C_4$	$1(2,3)C_1$	
D	$1(1,2)C_3$	$1(2,\emptyset)C_1$	$1(2,3)C_1$	→ H,F,D,I,G,E
I	$(1,2,3)C_1$	$1(1,\emptyset)C_4$	$1(2,3)C_1$	$c_1c_2c_3c_4c_5c_6$
E	$1(2,3)C_3$	$1(2,\emptyset)C_1$	$1(2,3)C_1$	
G	$1(2,3)C_2$	$1(2,\emptyset)C_3$	$1(2,3)C_1$	
J	$2\left\vert\begin{matrix}(1,3)C_2\\(1,3)C_3\end{matrix}\right.$	0	$1(2,1)$	
K	$2\left\vert\begin{matrix}(1,3)C_1\\(1,3)C_6\end{matrix}\right.$	0	$1(2,1)$	→ K,J,L $c_1c_2c_3$
L	$2\left\vert\begin{matrix}(1,3)C_4\\(1,3)C_5\end{matrix}\right.$	0	$1(2,1)$	
M	$3\left\vert\begin{matrix}(1,1)\\(1,1)\\(1,1)\end{matrix}\right.$			

d'où la partition initiale :

C_o	C_1	C_2	C_3	C_4
S,	ABC,	FH, D, I, EG,	JKL,	M
c_1	c_1	c_1 c_2 c_3 c_4	c_1	c_1

Les partitions initiales des 2 graphes se correspondent, les ensembles de suites caractéristiques étant les mêmes.

Affinement des partitions initiales

On se sert des tableaux précédents en portant une 3ème caractéristique, la classe du sommet constituant l'autre extrémité.

Graphe G (voir tableau)

- en opérant sur la première couche, on obtient la partition C,B,A ; l'ordre des blocs étant donné par les monômes C_3C_1, C_4C_1, C_4C_2
- on obtient alors pour la deuxième couche la partition G,E,D,I,H,F ;
- pour la troisième couche K,J,L.

Graphe G'

- la première couche donne la partition C,B,A ; la correspondance entre G et G' est possible (mêmes monômes).
- la deuxième couche donne la partition H,F,D,I,G,E
- la troisième couche la partition K,J,L d'où l'isomorphie

G	C	B	A	G	E	D	I	H	F	K	J	L
G'	C	B	A	H	F	D	I	G	E	K	J	L

A N N E X E

PROGRAMMES

I - REMARQUE GENERALE

L'ensemble de ces programmes a été écrit en ALGOL et les temps de passage indiqués sont des temps de passage sur I.B.M. 7044. Il convient d'accorder une valeur relative à ces temps de passage. Ces programmes n'ayant pas été écrits par des spécialistes de programmation, il est clair que ces temps peuvent être diminués de façon considérable. Ces programmes ont simplement été écrits en vue d'avoir une première idée de la maniabilité des algorithmes proposés et de la dimension des exemples que l'on peut espérer traiter ainsi. En vue d'applications industrielles, il conviendrait de reprendre les différentes solutions proposées et d'écrire de nouveaux programmes. Nous joindrons néanmoins les listings des 2 algorithmes d'immersion (graphe bipartite quelconque et graphe bipartite ordonné) à titre d'exemple.

II - PROGRAMME D'IMMERSION D'UN GRAPHE BIPARTITE CONVEXE DANS UN n-CUBE

II-1 Un programme de 1500 unités syntaxiques ALGOL réalise l'algorithme proposé au chapitre III.

Les temps de passage restent très réduits jusqu'à l'immersion dans le 5-cube ($\simeq$ 25 sommets) mais augmentent très rapidement ensuite. De plus, ces temps d'immersion sont très variables pour des graphes ayant sensiblement le même nombre de sommets d'arêtes et dépendant du nombre d'essais nécessaires dans l'algorithme précédent.

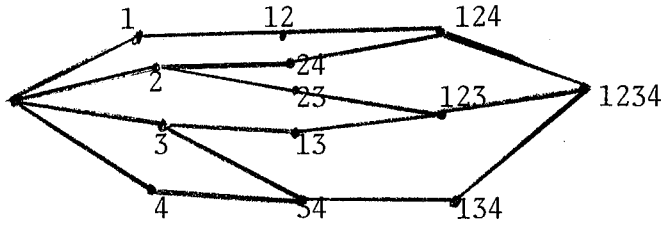
II-2 Entrées, Sorties

On considère la structure en couches associée à un sommet référence S .
L'ordre des sommets est arbitrairement fixé à l'intérieur d'une couche.
Ses sommets sont considérés couche par couche. Le graphe est rentré par sa matrice d'adjacence, les sommets étant considérés dans l'ordre ci-dessus.
Le code d'un sommet, est donné, à la sortie par son équivalent décimal.

II-3 - Résultats

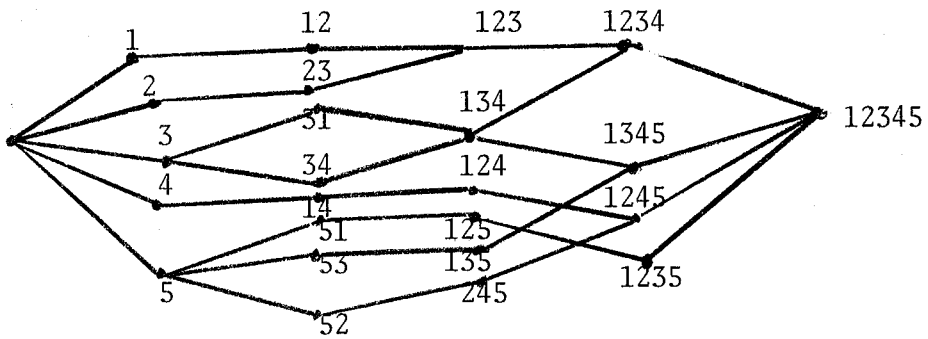
a) Graphe et code trouvé par le programme

14 sommets, 18 arêtes, n=4



temps de compilation et de calcul : 1'5"
temps de calcul : 2"

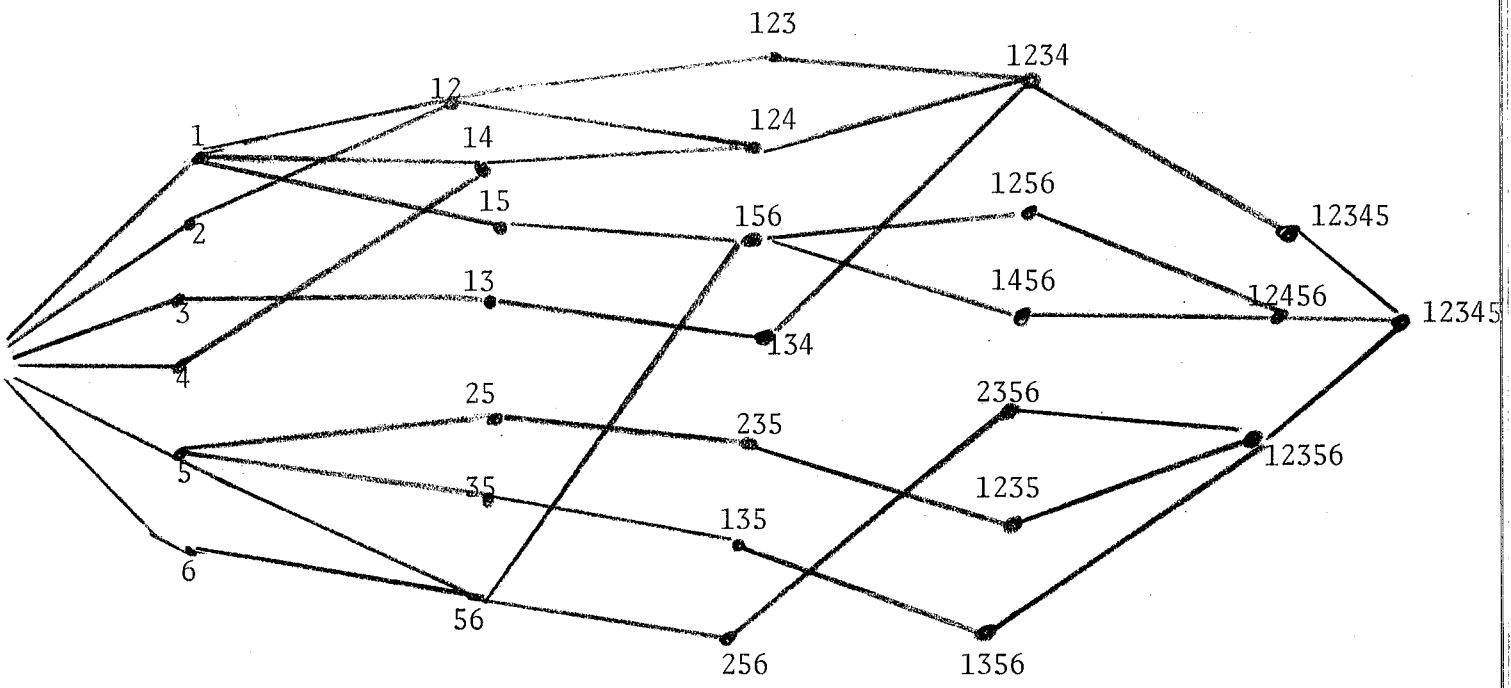
b) 25 sommets, 32 arêtes, n=5



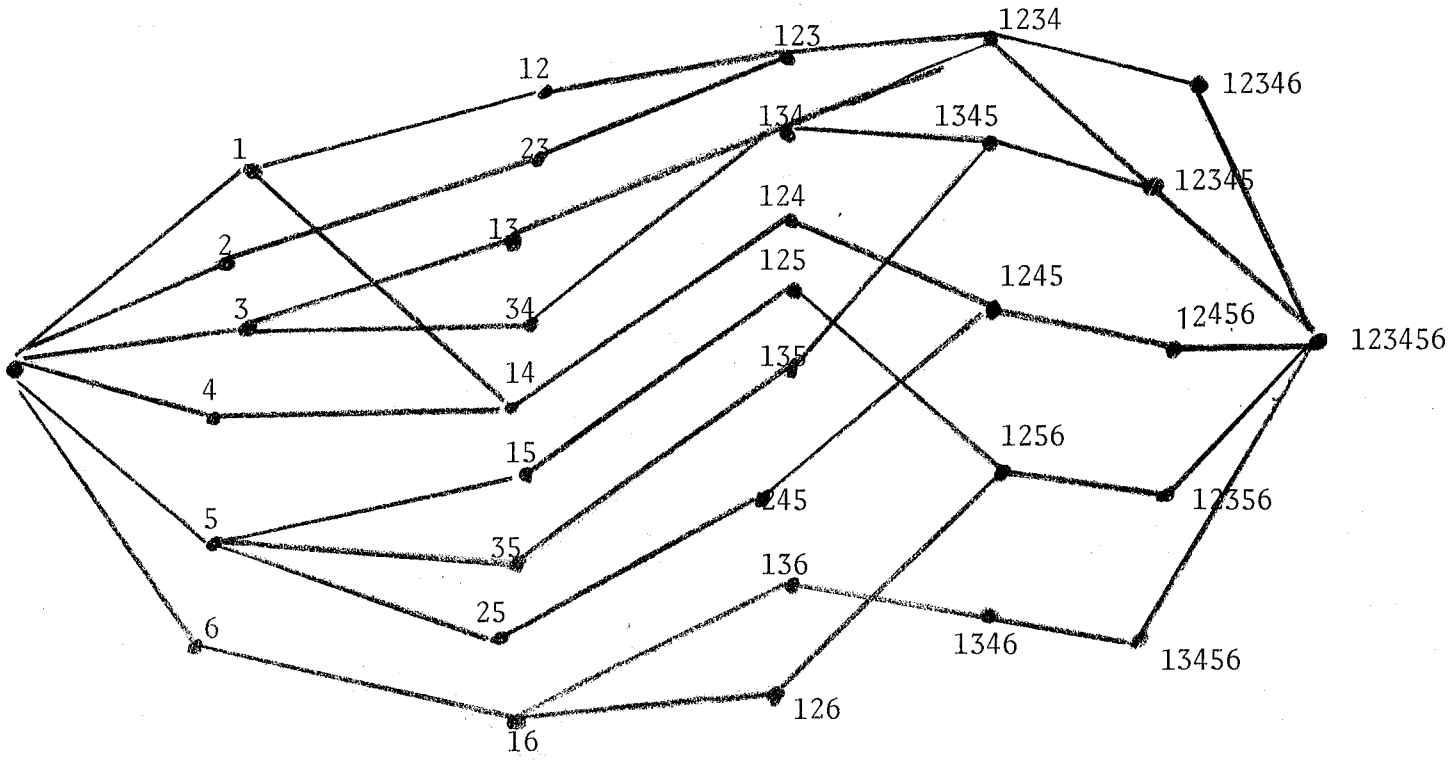
temps de calcul 16"

c) 31 sommets, 42 arêtes, n=6

temps de calcul 22"



d) 35 sommets, 46 arêtes, n=6

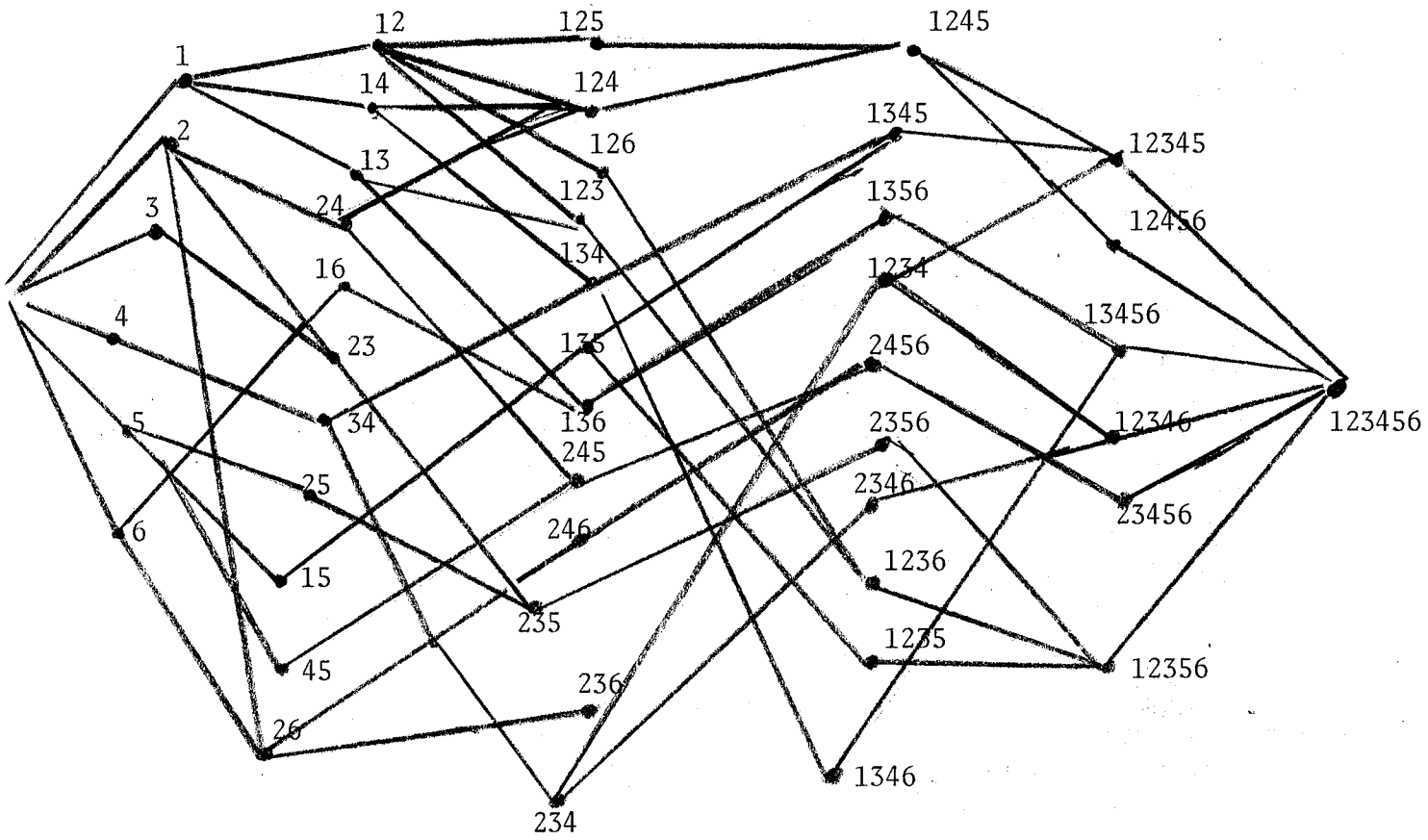


temps de calcul 4' 50"

On notera donc la grande différence des temps de calcul sur ces 2 exemples comportant sensiblement les mêmes nombres d'arêtes, de sommets et immergeables dans le même cube.
En appliquant manuellement l'algorithme aux 2 exemples, on remarque en effet un nombre d'essais nécessaires très supérieur dans le 2ème cas.

e) Ce dernier exemple montre qu'en augmentant le nombre d'arêtes (le nombre de sommets restant du même ordre) on peut s'attendre à une diminution du temps de calcul (moins de liberté dans le codage).

47 sommets, 71 arêtes, n=6 temps de calcul 1'25"



II-4 - Listing du programme correspondant

```
'DEBUT' 'ENTIER' I,J,NS,NC,N :: N est la dimension du cube considéré ;
        LIRE(NS,NC,N) :: N est le nombre de sommets, NC le nombre de couches
                         du graphe ;
   'DEBUT' 'ENTIER' 'TABLEAU' MATRADJ.(1:NS,1:NS), ce tableau est la matrice
                                                   d'adjacence du graphe ;
        NEC.(1:NC).,NOC.(1:NS).,COD.(0:NS). ,
        MASQUE.(1:36). :: le tableau NEC donne le nombre d'éléments dans une couche ;
                                  NOC le numéro de couche des sommets ;
        'POUR' I:=1 'PAS' 1 'A' NS 'FAIRE'
        'POUR' J:=1 'PAS' 1 'A' NS 'FAIRE' MATRADJ.(I,J).:=EDONNEE::
        'POUR' I:=1 'PAS' 1 'A' NC 'FAIRE' NEC.(I).:=EDONNEE ::
        'POUR' I:=1 'PAS' 1 'A' NS 'FAIRE' NOC.(I).:=EDONNEE ::
        MASQUE.(1).:=1 ::
        'POUR' I:=2 'PAS' 1 'A' 35 'FAIRE'
        MASQUE.(I).:=2*MASQUE.(I-1). ::
        MASQUE.(36).:=-0 ::
        'DEBUT'
'PROCEDURE' IMMERSGRAPH(MATADJ,NBS,NBEC,NOC,COD,N) ::
'ENTIER' 'TABLEAU' MATADJ,NBEC,NOC,COD :: 'ENTIER' NBS,NBC,N ::
'DEBUT' 'ENTIER' I,J,K,X,U,Z,COMPT,W,U1,U2,C1,C2,C3 ::
'BOOLEEN' BOOL1,BOOL2 ::
   'ENTIER' 'TABLEAU' TADJ.(1:NBS,1:N).,TAP.(1:NBS).,CHOIX.(1:NBS,0:2).,
   TV.(1:3). ,  SOM.(0:NBS).    ::
   'PROCEDURE' ADJ(A,T,X) :: 'ENTIER' A,X :: 'ENTIER' 'TABLEAU' T ::
      'DEBUT' 'ENTIER' I ::
      'POUR' I:=1 'PAS' 1 'A' N 'FAIRE'
        T.(X,I).:=DJ(A,MASQUE(I).)
      'FIN' ::
   'PROCEDURE' VERIFCOD(T,X,U,B1) :: 'ENTIER' X,U ::
         'ENTIER' 'TABLEAU' T :: 'BOOLEEN B1 ::
      'DEBUT' 'ENTIER' I,J ::
      B1:= 'VRAI' ::
      'POUR' I:=0 'PAS' 1 'A' X-1 'FAIRE'
      'DEBUT' 'SI' T.(I).=t.(X). 'ALORS' 'DEBUT' B1:= 'FAUX' ::
        U:=I 'FIN' 'FIN'
      'FIN' ::
   'PROCEDURE' INTERTABL(P,TP,B2,V) ::
      'ENTIER' P :: 'ENTIER' 'TABLEAU' TP,V :: 'BOOLEEN' B2 ::
      'DEBUT' 'ENTIER' I,J,K,C1,C2,C3 ::
      C1:=0 :: C2:=0 ::
      INT12:
        'POUR' I:=1 'PAS' 1 'A' N 'FAIRE'
        'POUR' J:=1 'PAS' 1 'A' N 'FAIRE'
'DEBUT' 'SI' TADJ.(TP.(1).,I).=TADJ.(TP.(2).,J).'ALORS'
        'DEBUT' C1:=C1+1    ::
V.(C1).:=TADJ.(TP.(1).,I).'FIN'
        'FIN' ::
      'SI' C1=0 'ALORS' 'DEBUT' B2:='FAUX' :: 'ALLERA' IMPOSS 'FIN' ::
      'SI' P=2 'ALORS' 'ALLERA' SORTIE ::
      INT23:
        'POUR' I:=1 'PAS' 1 'A' N 'FAIRE'
```

```
'DEBUT' 'SI' TADJ.(TP.(3).,I).=V.(1). 'ALORS'
        'DEBUT' C2:=1 :: V.(3).:=V.(1). :: 'ALLERA' SOR23 'FIN' ::
'SI'  TADJ.(TP.(3).,I).= V.(2). 'ALORS'
        'DEBUT' C2:=1 :: V.(3).:=V.(2). :: 'ALLERA' SOR23 'FIN' ::
        'FIN' ::
        SOR23: 'SI' C2=0 'ALORS' 'DEBUT' B2:= 'FAUX' ::
        'ALLERA' IMPOSS 'FIN' ::
        'SI' P=3 'ALORS' 'ALLERA' SORTIE ::
        C3:=3
      INT4N:
'POUR'I:=4'PAS'1'A'P'FAIRE'
'DEBUT''POUR'J:=1'PAS'1'A'N'FAIRE'
'DEBUT' 'SI' TADJ.(TP.(I).,J).= V.(3).'ALORS' 'ALLERA' IBOUCL 'FIN'::
B2:='FAUX'::'ALLERA'IMPOSS::
IBOUCL:'FIN'::
      SORTIE: B2:= 'VRAI' ::
      IMPOSS:
      'FIN' ::
      COD.(0).:=0   ::
      'POUR' K:=1 'PAS' 1 'A' NEC.(1). 'FAIRE'
      'DEBUT' COD.(K).:=MASQUE.(K). ::
         ADJ(COD.(K).,TADJ,K) 'FIN' ::
      SOM.(0).:=1   ::
      'POUR' I:=1 'PAS' 1 'A' NBC 'FAIRE' SOM.(I).:=SOM.(I-1).+
         NBEC.(I). ::
X:=SOM.(1).::
      TRAVAIL:
         'POUR' I:=X 'PAS' 1 'A' NBS 'FAIRE'
         'DEBUT'
ECRIRE(''I'',I)::
         COMPT:=0 ::
         'POUR' J:=I+1 'PAS' 1 'A' NBS 'FAIRE' CHOIX.(J,0).:=0 ::
         'POUR' K:=SOM.(NOC.(I).-2).   'PAS' 1 'A'(SOM.(NOC.(I).-1).-1)
         'FAIRE'
'DEBUT' 'SI' MATADJ.(I,K).=1 'ALORS' 'DEBUT' COMPT:=COMPT+1 ::
         TAP.(COMPT).:=K 'FIN'
ECRIRE(''K'',K)::
         'FIN' ::
         'SI' COMPT=1 'ALORS'
         'DEBUT' 'POUR' K:=( 'SI' CHOIX.(I,0).=0 'ALORS' 1 'SINON'
         CHOIX.(I,0).) 'PAS' 1 'A' N 'FAIRE'
         'DEBUT' COD.(I).:=TADJ.(TAP.(1).,K).::
ECRIRE(''COD'',COD.(I).)::
         VERIFCOD(COD,I,U,BOOL1) ::
         'SI' 'NON' BOO1 'ALORS' 'ALLERA' KBOUCLE ::
         CHOIX.(I,0).:=K ::
         ADJ(COD.(I).,TADJ,I) :: 'ALLERA' CHF1 ::
         KBOUCLE: 'FIN' ::
         'SI' I 'NONEG' NBEC.(1).+1  'ALORS' 'DEBUT' W:=I-1 ::
```

```
ECRIRE(''W'',''CHOIXPREC'',W)::
          'ALLERA' CHOIXPREC 'FIN'::
          'ALLERA' CODIMP ::
          CPF1: 'ALLERA' SUITE
          'FIN' ::
          INTERTABL(COMPT,TAP,BOOL2,TV) ::
          'SI' 'NON' BOOL2 'ALORS' 'DEBUT' W:=TAP.(COMPT). ::
ECRIRE(''W'',''CHOIXPREC'',W)::
          'ALLERA' CHOIXPREC 'FIN'::
          'SI' COMPT=2 'ALORS'
          'DEBUT' CHOIX.(I,1).:=TV.(1). ::
          CHOIX.(I,2).:=TV.(2). ::
          COD.(I).:=TV.(1). ::
ECRIRE(''COD'',COD.(I).)::
          VERIFCOD(COD,I,U1,BOOL1) ::
          'SI' BOOL1 'ALORS' 'DEBUT' CHOIX.(I,0).:=100 ::
          ADJ(COD.(I).,TADJ,I) ::
          'ALLERA' CHF2 'FIN' ::
          COD.(I).:=TV.(2). ::
ECRIRE(''COD'',COD.(I).)::
          VERIFCOD(COD,I,U2,BOOL1) ::
          'SI' BOOL1 'ALORS' 'DEBUT' CHOIX.(I,0).:=200 ::
          ADJ(COD.(I).,TADJ,I) ::
          'ALLERA' CHF2 'FIN' ::
          'SI' U1 'SUPER' U2 'ALORS' K:=U1 'SINON' K:=U2 ::
          'SI' K 'SUPER' TAP.(COMPT). 'ALORS' W:=K
          'SINON' W:=TAP.(COMPT). ::
ECRIRE(''W'',''CHOIXPREC'',W)::
          'ALLERA' CHOIXPREC ::
          CHF2: 'ALLERA' SUITE
          'FIN' ::
          COD.(I).:=TV.(3). ::
ECRIRE(''COD'',COD.(I).)::
          VERIFCOD(COD,I,U,BOOL1) ::
          'SI' 'NON' BOOL1 'ALORS' 'DEBUT'
          'SI' U 'SUPER' TAP.(COMPT). 'ALORS' W:=U 'SINON'
          W:=TAP.(COMPT). ::
ECRIRE(''W'',''CHOIXPREC'',W)::
          'ALLERA' CHOIXPREC 'FIN' ::
          ADJ(COD.(I).,TADJ,I) :: 'ALLERA' SUITE ::
          CHOIXPREC:
          'POUR' K:=W 'PAS' -1 'A' NBEC.(1).+1 'FAIRE'
          'DEBUT' 'SI' CHOIX.(K,0).=0 'ALORS' 'ALLERA' K2BOUCLE ::
          'SI' CHOIX.(K,0).=200 'ALORS' 'ALLERA' K2BOUCLE ::
          'SI' CHOIX.(K,0).=100 'ALORS' 'DEBUT' CHOIX.(K,0).:=200 ::
          COD.(K).:=CHOIX.(K,2). ::
```

```
ECRIRE(''COD'',COD.(I).)::
        VERIFCOD(COD,K,U,BOOL1) ::
        'SI' BOOL1 'ALORS' 'DEBUT' ADJ(COD.(K).,TADJ,K) ::
        'SI' K 'NONEG' NBS 'ALORS' 'DEBUT' CHOIX.(K+1,O).:=O ::
        X:=K+1 :: 'ALLERA' TRAVAIL 'FIN'
        'ALLERA' ECRITURE 'FIN' ::
        'ALLERA' K2BOUCLE 'FIN' ::
        'SI' CHOIX.(K,O).=N 'ALORS' 'ALLERA' K2BOUCLE::
        CHOIX.(K,O).:=CHOIX.(K,O).+1 :: X:=K ::
        'ALLERA' TRAVAIL ::
        K2BOUCLE: 'FIN' ::
        CODIMP: ECRIRE(''CODAGE IMP'') :: 'ALLERA' TERMIN ::
        SUITE: 'FIN' ::
        ECRITURE: 'POUR' I:=1 'PAS' 1 'A' NBS 'FAIRE'
        ECRIRE(COD.(I).) ::    tableau des codes des sommets
        TERMIN: 'FIN' ::
        IMMERSGRAPH(MATRADJ,NS,NC,NEC,NOC,COD,N) ::
        'FIN'
        'FIN'
U.S.1627 "FIN' ::
FALGOL
```

III - IMMERSION D'UNE REPRESENTATION ORDONNEE D'UN GRAPHE BIPARTITE

III-1 - Entrée-Sortie

Le graphe est rentré, comme précédemment, par sa matrice d'adjacence, l'ordre des sommets étant déterminé de la même façon. En vue de réduire l'encombrement de cette matrice, chaque élément de cette matrice sera le contenu d'un bit d'une mémoire; de même, à la sortie, le code d'un sommet s sera donné par le contenu d'une mémoire dans laquelle le $i^{ème}$ bit vaut 1 si $i \in \varepsilon_c(S,s)$, S étant le sommet de référence.

III-2 - Ce programme comprend 2.200 unités syntaxiques ALGOL et

est compilé sur IBM 7044 en 1' 20".

Les temps de calcul sont très acceptables ; le graphe, exemple e) du paragraphe précédent immergé en 4' 50", est immergé, cette fois, muni de la relation d'ordre en 1' 25". On peut donc facilement immerger des graphes bipartites ordonnés d'une cinquantaine de sommets.

III-3 - Listing du programme correspondant

```
'DEBUT' 'ENTIER' NS,NC,N :: les symboles sont les mêmes que dans le programme précédent.
'PROCEDURE' RCH ::
'COMMENTAIRE' RETOUR A LA LIGNE ::
                                                                    'CODE'
       TRA     ZIMAX+1
       EXTERN  JOBOU
ZJOBOU SXA     ZLISTE+1,2
       SXD     ZLISTE+1,4
       PXD     ,4
       ADD     ZI
       STD     ZI
       CLA     ZIMAX
       SUB     ZI
       TPL     ZJOBLI
       PXD     ,4
       ADD     ZBLANC+1
       STD     ZI
ZJOBLA CALL    JOBOU(ZBLANC)
ZJOBLI CALL    JOBOU(ZLISTE)
       TRA     1,1
ZBLANC PZE     1
       PZE     *+1,1,1
       BCI     9,
       BCI     9,
       BCI     4,
ZLISTE PZE     1
       MZE     **,1,**
ZI     BCI     1,022000
ZIMAX  BCI     1,022000
       CLA     =H022000
       STO     ZI
'FOCDE' ::
'PROCEDURE' SORCHAINE(I,J,X) :: 'VALEUR' I,J :: 'ENTIER' I,J,X ::
'COMMENTAIRE' ECRIT UN TEXTE DE I SYMBOLES DONT LE PREMIER EST DANS
LA MEMOIRE X EN POSITION J    . LE TEXTE S'IMPRIME A LA SUITE DES
TEXTES PRECEDENTS , S'IL Y A SUFFISEMENT DE PLACE. AUCUN ESPACE N'EST
PREVU .    ::
                                                                    'CODE'
       TSX     P2-3,4
       LXA     F2+1,4
       CLA     F2+2
       LAS     3
       SAC     ZLISTE+1,,3
       TSX     ZJOBOU,1
```

```
'FCODE' ::
   LIRE(NS,NC,N) ::
   'DEBUT' 'ENTIER' C,P,R,I,J,K,L,M1,M2 :: 'BOOLEEN' B ::
   'ENTIER' 'TABLEAU' NEC.(1:NC).,MASQUE.(0:36).,NB.(0:NC).,
   NOC.(0:NS).,COMB .(1:N).,SOMME.(0:N).,TK.(1:N).,RANGNB.(1:2**N). ::
   MASQUE.(1).:=1 :: MASQUE.(36).:=-0 ::
   'POUR' I:=2 'PAS' 1 'A' 35 'FAIRE'
   MASQUE.(I).:=2*MASQUE.(I-1). ::
   'POUR' I:=1 'PAS' 1 'A' NC 'FAIRE' NEC.(I).:=EDONNEE ::
   P:=(NS'/'37)+1 :: R:=NS-36*(P-1) :: 'SI' NS=36*P 'ALORS' R:=0 ::
   NB.(0).:=0 :: NOC.(0).:=0 ::
   'POUR' I:=1 'PAS' 1 'A' NC 'FAIRE'
   NB.(I).:=NB.(I-1).+NEC.(I). ::
   'POUR' I:=1 'PAS' 1 'A' NC 'FAIRE'
   'POUR' J:=(NB.(I-1).+1) 'PAS' 1 'A' NB.(I). 'FAIRE' NOC.(J).:=I ::
'POUR'I:=1'PAS'1'A'NS'FAIRE'ECRIRE (NOC.(I).)::
COMB.(1).:=N::
   'POUR' I:=2 'PAS' 1 'A' N 'FAIRE'
   'DEBUT' C:=N ::
      'POUR' J:=2 'PAS' 1 'A' I 'FAIRE' C:=C*(N-(J-1))/J ::
      COMB.(I).:=C    ::ECRIRE(C)::
   'FIN' ::
   SOMME.(0).:=0 ::
   'POUR' I:=1 'PAS' 1 'A' (N-1) 'FAIRE'
   SOMME.(I).:=SOMME.(I-1).+COMB.(I). ::
   'POUR' I:=1 'PAS' 1 'A' N 'FAIRE' TK.(I).:=1 ::
'POUR'I:=1 'PAS'1 'A'  (2**N)-1
'FAIRE' 'DEBUT' J:=POIDS(I)::
   RANGNB.(SOMME.(J-1).+TK.(J).).:=I::
     TK.(J).:=TK.(J).+1
     'FIN' ::
'POUR'I:=1 'PAS'1 'A' (2**N)-1
'FAIRE' ECRIRE (RANGNB.(I).)::
   'DEBUT' 'PROCEDURE' ETABLISC(DN,SC) :: 'ENTIER' 'TABLEAU' DN,SC ::
      'DEBUT' 'ENTIER' I,J,K,L :: procédure recherchant la section commençante
                                  des sommets d'un graphe bipartite ordonné ;
      'POUR' I:=1 'PAS' 1 'A' NS 'FAIRE'
      'POUR' J:=1 'PAS' 1 'A' P 'FAIRE' SC.(I,J).:=DN.(I,J). ::
      'POUR' I:=1 'PAS' 1 'A' NS 'FAIRE'
      'POUR' J:=1 'PAS' 1 'A' P 'FAIRE'
      'POUR' K:=1 'PAS' 1 'A' 36 'FAIRE'
      'DEBUT' 'SI' BIT(SC.(I,J).,K)=1 'ALORS'
         'DEBUT' 'POUR' L:=1 'PAS' 1 'A' P 'FAIRE'
      SC.(I,L).:=OU(SC.(I,L).,DN.(36*(J-1)+K,L).) 'FIN'
      'FIN'
      'FIN' ::
```

```
      'PROCEDURE' COMPLETSC(CD,SC,X) :: 'ENTIER' X ::
      'ENTIER' 'TABLEAU' CD,SC ::
      'DEBUT' 'ENTIER' I,J ::
         'POUR' I:=1 'PAS' 1 'A' P 'FAIRE'
         'POUR' J:=1 'PAS' 1 'A' 36 'FAIRE'
   'DEBUT''SI'BIT(SC.(X,I).,J)=1'ALORS'
      COD.(36*(I-J).:=OU(COD.(36*(I-1-+J).,COD.(X).)
         'FIN'
      'FIN' ::
      'PROCEDURE' IMPCOD(COD,B) :: 'ENTIER' 'TABLEAU' COD ::
      'BOOLEEN' B ::
      'DEBUT' 'ENTIER' I,J ::
      B:= 'VRAI' ::
      'POUR' I:=(NEC.(1).+1) 'PAS' 1 'A' NS 'FAIRE'
      'DEBUT' 'SI' POIDS(COD.(I).) 'SUPER' NOC.(I). 'ALORS' 'DEBUT'
      B:= 'FAUX' ::
      'ALLERA' SORPRO 'FIN' ::
         'FIN' ::
      'POUR' I:=(NEC.(1).+1) 'PAS' 1 'A' (NS-1) 'FAIRE'
'POUR' J:=I+1 'PAS'1 'A'NS 'FAIRE'
      'DEBUT' 'SI' ((NOC.(I).=NOC.(J).) 'ET' (POIDS(COD.(I).)=NOC.(I).)
      'ET' (POIDS(COD.(J).)=NOC.(J).) 'ET' (COD.(I).=COD.(J).))
         'ALORS' B:= 'FAUX' ::
      'FIN'::
      SORPRO: 'FIN' ::
      'ENTIER' 'TABLEAU' DONNEE.(1:NS,1:P).,SECTCOM.(1:NS,1:P).,
      COD.(1:NS).,CHOIX.(1:NS). ::
         'POUR' I:=1 'PAS' 1 'A' NS 'FAIRE'
      'DEBUT' 'POUR' J:=1 'PAS' 1 'A' (P-1) 'FAIRE'
         'POUR' K:=1 'PAS' 1 'A' 36 'FAIRE'
      'DEBUT' 'SI' EDONNEE=1 'ALORS' DONNEE(I,J).:=OU(DONNEE.(I,J).,
      MASQUE.(K).) 'FIN' ::
         'POUR' K:=1 'PAS' 1 'A' R 'FAIRE'
         'DEBUT' 'SI' EDONNEE=1 'ALORS' DONNEE.(I,P).:=OU
      (DONNEE.(I,P). ,MASQUE.(K).) 'FIN' ::
      'FIN' ::
         ETABLSC(DONNEE,SECTCOM) ::
         'POUR' I:=1 'PAS' 1 'A' NS 'FAIRE'
         'POUR' J:=1 'PAS' 1 'A' P 'FAIRE'
      'DEBUT' 'POUR' K:=1 'PAS' 1 'A' 36 'FAIRE'
         SORCHAINE(1,6,BIT(SECTCOM.(I,J). ,K))::
         RCH
         'FIN' ::
         'POUR' I:=1 'PAS' 1 'A' NEC.(1). 'FAIRE'
         'DEBUT' COD.(I).:=MASQUE.(I). ::
         COMPLETSC(COD,SECTCOM,I) ::
         IMPCOD(COD,B) ::
         'SI' 'NON' B 'ALORS' 'ALLERA' IMPOSS ::
         'FIN' ::
         'POUR' I:=1 'PAS' 1 'A' NS 'FAIRE'
      'DEBUT' 'POUR' K:=1 'PAS' 1 'A' 36 'FAIRE'
   SORCHAINE(1,6,BIT(COD.(I   ).,K))::
```

```
RCH'FIN'::
         M1:=NEC.(1).+1 ::
       'POUR' L:=M1 'PAS' 1 'A' NS 'FAIRE' CHOIX.(L).:=1 ::
          INIT: 'POUR' I:=M1 'PAS' 1 'A' NS 'FAIRE'
          'DEBUT' K:=NOC.(I). ::
          'SI' POIDS(COD.(I).)=K 'ALORS' 'ALLERA' SORTIE ::
          'SI' COD.(I).=0 'ALORS'
'DEBUT''POUR'
               J:=(SOMME.(K-1).+CHOIX.(I).) 'PAS' 1 'A' SOMME.(K).
       'FAIRE'
       'DEBUT' COD.(I).:=RANGNB.(J). ::
ECRIRE(COD.(I).)::
          COMPLETSC(COD,SECTCOM,I) ::
          IMPCOD(COD,B) ::
    ECRIRE(B) ::
       'SI' B 'ALORS' 'DEBUT' CHOIX.(I).:=J-SOMME.(K-1). ::
ECRIRE(CHOIX.(I).)::
          'ALLERA' SORTIE 'FIN' ::
'POUR'L:=(I+1)'PAS'1'A'NS'FAIRE'COD.(L).:=0::
'POUR'L:=1'PAS'1'A'(I-1)'FAIRE' COMPLETSC(COD,SECTCOM,L)::
       'SI' J=SOMME.(K). 'ALORS'
       'DEBUT' M1:=I-1 ::
       'ALLERA' RENVOI 'FIN' ::
          'FIN'
    'FIN'::
       'POUR' K:=CHOIX.(I). 'PAS' 1 'A' N 'FAIRE'
          'DEBUT' 'SI' BIT(COD.(I).,K) =0 'ALORS'
          'DEBUT' COD.(I).:=OU(MASQUE.(K).,COD.(I).) ::
ECRIRE(COD.(I).)::
          COMPLETSC(COD,SECTCOM,I) ::
          IMPCOD(COD,B) ::
    ECRIRE(B)::
          'SI' B 'ALORS' 'DEBUT' CHOIX.(I).:=K ::
ECRIRE(CHOIX.(I).)::
          'ALLERA' SORTIE 'FIN' ::
    COD.(I).:=DJ(MASQUE.(K).,COD.(I).)::
'POUR'L:=(I+1)'PAS'1'A'NS'FAIRE'COD.(L).:=0::
'POUR'L:=1'PAS'1'A'(I-1)'FAIRE' COMPLETSC(COD,SECTCOM,L)::
          'FIN'
          'FIN' ::
    M1:=(I-1)::
          RENVOI:
          'SI' M1=NEC.(1). 'ALORS' 'ALLERA' IMPOSS ::
          'POUR' L:=(M1+1) 'PAS' 1 'A' NS 'FAIRE' CHOIX.(L).:=1 ::
          'POUR' L:=M1 'PAS' 1 'A' NS 'FAIRE' COD.(L).:=0 ::
          'POUR' L:=1 'PAS' 1 'A' (M1-1) 'FAIRE'
          COMPLETSC(COD,SECTCOM,L) ::
```

```
          'SI' POIDS(COD.(M1)).)=NOC.(M1). 'ALORS'
          'DEBUT' M1:=M1-1 ::
          'ALLERA' RENVOI
          'FIN' ::
       'SI' ((COD.(M1).=0) 'ET' CHOIX.(M1).=SOMME.(POIDS (M1) ).)
          'ALORS' 'DEBUT' M1:=M1-1 :: 'ALLERA' RENVOI 'FIN' ::
       'SI' CHOIX.(M1).=N 'ALORS' 'DEBUT' M1:=M1-1 ::
          'ALLERA' RENVOI 'FIN' ::
    CHOIX.(M1). := CHOIX.(M1). +1 ::
          'ALLERA' INIT ::
          SORTIE: 'FIN' ::
       'POUR' I:=1 'PAS' 1 'A' NS 'FAIRE'
          'DEBUT' 'POUR' K:=1 'PAS' 1 'A' 36 'FAIRE'
          SORCHAINE(1,6,BIT(COD.(I). ,K))::
          RCH ::
          'FIN' ::
          'ALLERA' FINAL ::
          IMPOSS: ECRIRE(''CODAGE IMP'') ::
       FINAL: 'FIN'
U.S.2208 FIN'
'FIN' ::
FALGOL
```

IV - RECHERCHE D'UN ARBRE COMPLET DE POIDS MAXIMAL ; RECHERCHE DES SOMMETS SUPPLEMENTAIRES

IV-1 - Recherche d'un arbre complet de poids maximal

L'algorithme utilisé a été explicité et se programme sans aucune difficulté ; un programme de 300 unités syntaxiques recherche en quelques secondes un arbre complet de poids maximal pour un graphe d'une vingtaine de sommets.

IV-2 - Recherche des sommets supplémentaires

La difficulté de ce programme (2000 unités syntaxiques) est liée dans certains exemples à des problèmes d'encombrement de l'arborescence des chemins. Cet encombrement est proportionnel à la quantité $\frac{n-n_s}{n}$ où n est la dimension du cube, n_s le nombre de sommets du bloc ; cette quantité nous donne, en effet, la proportion des sommets inoccupés du cube.
Des performances ont été données au cours des exemples traités.

BIBLIOGRAPHIE

1) D.B. ARMSTRONG — "A programmed algorithm for assigning internal codes to sequential machines". I.E.E.E. Trans. on E.C août 1962 pp. 466-472.

2) D.B. ARMSTRONG — "On the efficient assignment of internal codes to sequential machines". Octobre 1962 pp. 611-622.

3) C. BERGE — "Théorie des graphes et ses applications". Dunod Paris 1963 pp. 260.

4) S.H. CALDWELL — "Switching circuits and logical design" New-York ; Wiley, 1958.

5) T.A. DOLOTTA et E.J. Mc CLUSKEY — "The coding of internal states of sequential circuits". I.E.E.E. Trans. on E.C octobre 1964 - pp. 549-562.

6) J. FLORINE — "Automatisme à séquences et commandes numériques". Dunod Paris 1969.

7) A. FRIEDES — "State reduction and state assignments for asynchronous machines". Ph. D. dissertation, Columbia University, 1967.

8) A.D. FRIEDMAN — "Universal single transition time asynchronous state assignments". I.E.E.E. Conf. Record of eight annual symposium on switching and automata theory, Austin, Texas, october 18-20, 1967.

9) GALIMBERTI, MORPURGO — "A method for internal variable assignment in asynchronous sequential switching networks" dans 17)

11) B. GERACE — "A universal state assignment method for pulse input asynchronous sequential circuits". I.F.A.C. Symposium on hazard and race phenomena in switching circuits ; Bucharest 6-10 october 1968.

12) D.R. HARING — "Sequential circuit synthesis state assignment aspects". M.I.T PRESS 1966 - pp. 340.

13) J. HARTMANIS, R.E. STEARNS — "Algebraic structure theory of sequential machines englewood". Cliffs, N.J. : Prentice Hall 1966.

14) B. HAZELTINE — "Encoding of asynchronous sequential circuits". I.E.E.E. Trans. on E.C octobre 1965 pp. 727-729.

15) D.A. HUFFMAN — "A study of the memory requirements of sequential switching circuits". Research Lab. of electronics M.I.T. Cambridge Tech. Rep. n° 293 : avril 1955.

16) J. KUNTZMANN — "Algèbre de Boole". Dunod 1965 - pp. 310.

17) J. KUNTZMANN, P. NASLIN — "Algèbre de Boole et machines logiques". Dunod 1967 - pp. 310.

18) C.N. LIU — "A state variable assignment method for asynchronous sequential swtiching circuits". J. A.C.M. Vol. 10 pp. 209-226 ; avril 1963.

19) H. LOBERMAN, A. WEINBERGER — "Formal procedures for connecting terminals with a minimal total wire length". J. A.C.M. June 1957, pp. 428-433.

20) E.J. Mc CLUSKEY — "Introduction to the theory of switching networks". New York Mc Graw Hill 1965.

21) E.J. Mc CLUSKEY, S.H. UNGER — "A note on the number of internal variable assignments for sequential switching circuits". I.R.E. Trans. on E.C Vol EC-8 - pp. 439-440.

22) G.K. MAKI
J.H. TRACEY
"State assignment selection in asynchronous sequential circuits".
I.E.E.E. Trans. on Computers Vol. C-19 n° 7 juillet 1970.

23) G.K. MAKI
J.H. TRACEY
"Generation of design equations in asynchronous sequential circuits".
I.E.E.E. Trans on Computers Vol. C-18 pp. 467-472 (mai 1969).

24) P. NASLIN
"Circuits logiques et automatismes à séquence".
DUNOD (1965).

25) J. RIORDAN
"An introduction to combinatorial analysis".
New-York Wiley (1958).

26) G. SAUCIER
"Encoding of asynchronous sequential networks".
I.E.E.E. Trans. on E.C vol. 16 n° 3 pp. 365-369.

27) G. SAUCIER
"Algorithme pratique de codage des automates asynchrones".
I.F.A.C. Symposium : "Aléas de continuité et de simultanéité dans les circuits de commutation".
Bucarest 6-10 octobre 1968.

28) J.P. STEEN
"Algorithme de recherche d'un isomorphisme entre 2 graphes".
Thèse de 3ème Cycle, Mathématiques Appliquées, Lille 1968.

29) H.C. TORNG
"An algorithm for finding secondary assignments of synchronous sequential circuits".
I.E.E.E. Trans. on computers mai 1968, pp. 461-469.

30) J.H. TRACEY
"Internal state assignments for asynchronous sequential machines".
I.E.E.E. Trans. on E-C, vol. 15, août 1966.

31) S.H. UNGER
"A row assignment for delay free realizations of flow tables without essential hazards".
I.E.E.E. Trans. on E.C Vol. 17 n° 2 pp. 146-151.

32) S.H. UNGER "GIT - A heuristic program for testing pairs of directed line graphs for isomorphism". Communications of the A.C.M., vol. 7, n° 1 (janvier 1964) pp. 26-34.

33) M. VILCOQ "Recherche de l'isomorphie de 2 graphes". Séminaire I.M.A.Grenoble janvier 1970.

VU

Grenoble, le

Le Président de la Thèse

VU

Grenoble, le

Le Doyen de la Faculté des Sciences

VU, et permis d'imprimer

Le Recteur de l'Académie de GRENOBLE