Conception et évaluation des performances d'un système informatique de commande

Yves Sagot

N° d'ordre : 574

Année 1977

THÈSE

présentée

DEVANT L'UNIVERSITÉ PAUL SABATIER DE TOULOUSE

pour l'obtention

du GRADE de DOCTEUR INGÉNIEUR

par

Yves SAGOT

Ingénieur ENICA

CONCEPTION ET ÉVALUATION DES PERFORMANCES D'UN SYSTÈME INFORMATIQUE DE COMMANDE

Soutenue le 13 Avril 1977, devant la Commission d'Examen :

MM.	R. BEAUFILS	*Président*
	A. COSTES	*Examinateurs*
	J.C. LAPRIE	
	R. MORTIER	
	D. POTIER	
	B. SEMPÉ	

A Alain COSTES

A Jean-Claude LAPRIE

A Christian MARROT

A mes camarades de l'Equipe

"Automatismes Numériques"

en témoignage d'amitié.

AVANT-PROPOS

Le travail que nous présentons dans ce mémoire a été effectué au sein de l'Equipe "Automatismes Numériques" du Laboratoire d'Automatique et d'Analyse des Systèmes du Centre National de la Recherche Scientifique.

Nous tenons à exprimer notre profonde gratitude à :

- *Monsieur R. BEAUFILS, Professeur à l'Université Paul Sabatier de Toulouse,*
- *Monsieur A. COSTES, Maître de Conférences à l'Institut National Polytechnique de Toulouse,*
- *Monsieur J.C. LAPRIE, Chargé de Recherche au C.N.R.S., Responsable de l'Equipe "Automatismes Numériques" du L.A.A.S.,*
- *Monsieur R. MORTIER, Chef du Département Techniques Numériques à la S.F.E.N.A.,*
- *Monsieur D. POTIER, Responsable Scientifique au LABORIA,*
- *Monsieur B. SEMPE, Directeur de la Société OPTION,*

qui ont bien voulu être membres du jury de soutenance de cette thèse.

Nous ne saurions dire ici tout ce que nous devons à Alain COSTES depuis qu'il nous a accueilli dans son Equipe ; il nous a accordé sa confiance et son aide en toutes circonstances.

Nous sommes très reconnaissant envers Jean-Claude LAPRIE qui nous a donné tous les conseils nécessaires pour mener cette étude.

Ce travail n'aurait pas été possible sans l'aide du Centre National d'Etudes Spatiales qui nous a accordé une bourse d'étude. Que cet Organisme, et plus particulièrement MM. LE STANG, LAVERLOCHERE et ASTOR, trouve ici l'expression de tous mes remerciements.

Nous voulons aussi exprimer nos plus vifs remerciements à tous ceux qui nous ont aidé dans la réalisation de ce travail :

- *J. ARLAT, J. BARRETO LEITE, C. BEOUNES, F. CEREJA, Y. CROUZET, G. JUANOLE, C. LANDRAULT, A. LESTRADE-CARBONNEL, D. POWELL, membres de l'Equipe "Automatismes Numériques",*
- *J. PENAVAYRE et C. MARROT, Secrétaire et Ingénieur de la Division "Structures des Systèmes de Commande Automatique",*
- *R. ZITTEL, D. DAURAT et R. LORTAL du Service "Documentation-Publications".*

SOMMAIRE

CHAPITRE I

CONTEXTE GENERAL DE L'ETUDE

INTRODUCTION

-

Le traitement automatique de l'information met en jeu un grand nombre de disciplines dans un champ d'application très voisin de l'organisation qu'elle soit scientifique, industrielle ou administrative.

Pour notre part, nous nous sommes essentiellement intéressé au traitement automatique de l'information dans la conduite en temps réel de processus industriels, avioniques,... par des systèmes que nous appellerons désormais "systèmes informatiques de commande".

I.1. GENERALITES SUR LES SYSTEMES INFORMATIQUES DE COMMANDE

I.1.1. Les entités d'un système informatique de commande

Sans préjuger de la complexité d'une application particulière, on peut considérer que les entités qui interviennent dans la définition d'un système informatique de commande se répartissent entre les cinq ensembles indiqués sur le tableau de la figure I.1.

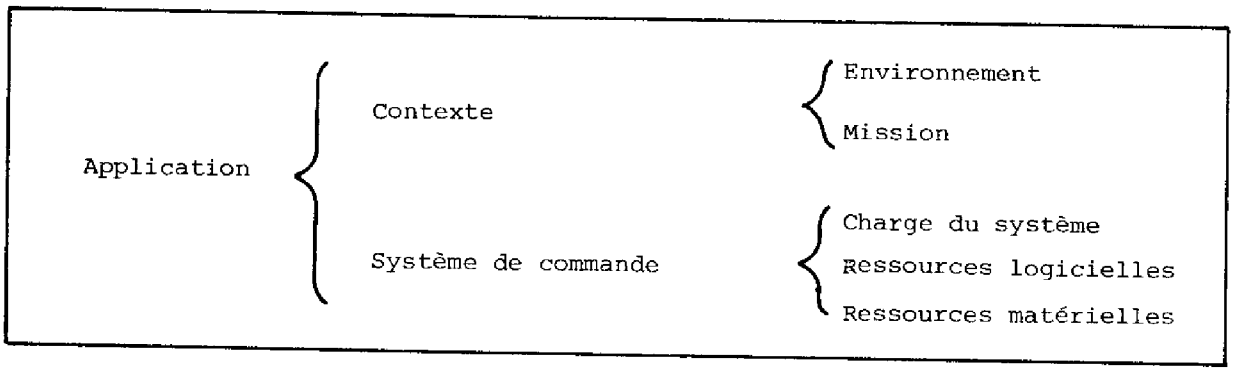


FIGURE I.1.

Le but de ce paragraphe est de présenter ces entités et de les situer les unes par rapport aux autres.

I.1.1.1. L'environnement

L'environnement regroupe tous les éléments de l'application qui entourent le système de commande ; il s'agit :

- du système physique auquel est rattaché le système de commande et qui est caractérisé par ses actions, son évolution, les moyens d'intervention qu'il offre,...

- de l'opérateur humain, avec sa facilité d'action et de décision,
- du milieu extérieur en général, qui peut intervenir sous la forme de variations, de perturbations,...

I.1.1.2. La mission

La mission qui recouvre l'ensemble des actions que doit effectuer le système de commande, constitue le cahier des charges de l'application. La nature des actions et la manière dont elles doivent être exécutées dépendent de l'environnement -actions à réaliser, relations de dépendance et de priorité entre ces actions- et se traduisent au niveau du système de commande par des contraintes sur ses performances :

- fonctionnelles : elles regroupent la nature des actions à engager et des caractéristiques temporelles comme les temps de prise en compte, d'exécution,...
- opérationnelles : elles regroupent les différentes composantes de la sûreté de fonctionnement : disponibilité, fiabilité, sécurité, maintenabilité, tolérance aux fautes,...

Pour conserver une description simple de l'application, nous attribuerons également à la mission des contraintes qui ne proviennent pas directement de l'environnement de l'application envisagée, mais de considérations plus générales, d'ordre économique ou purement pratique, se rapportant à la totalité de la classe des applications d'un même type.

Ces contraintes affectent :

- la facilité d'emploi (limitation de l'intervention de l'homme, transparence de la structure au programmeur,...),
- le caractère évolutif du système (adaptabilité, souplesse),
- le coût (coût global, coût du stockage, coût des transmissions,...).

I.1.1.3. La charge du système

La charge du système est l'expression de la mission au niveau du système de commande :

- les actions de la mission sont matérialisées par des programmes qui cons-

tituent les tâches que le système de commande doit réaliser,

- les contraintes de la mission se répercutent, au niveau de la charge du système, sous la forme de règles de coordination des tâches.

I.1.1.4. Les ressources logicielles

Les ressources logicielles sont les possibilités fonctionnelles du système de commande : elles permettent l'énoncé des tâches.

La complexité de cette notion est due au fait qu'il est possible de réaliser différents découpages des tâches en fonctions logiques qui correspondent à des niveaux de description des tâches plus ou moins abstraits. Pour notre part, nous avons attribué cette notion aux plus petites fonctions du système de commande que l'on désire prendre en compte.

Les contraintes générales de la mission se traduisent, au niveau des ressources logicielles, par des règles d'allocation des fonctions logiques aux différentes tâches.

I.1.1.5. Les ressources matérielles

Le problème du niveau de description où l'on définit les ressources matérielles se pose de la même manière que pour les ressources logicielles ; nous avons défini une ressource matérielle comme l'ensemble des éléments matériels nécessaires à la réalisation de la ressource logicielle correspondante.

La nature des éléments matériels est donc déterminée par le choix des fonctions logiques, et les performances de ces éléments sont fixées pour les contraintes propres à l'application envisagée.

Le regroupement ou le découpage en modules physiquement distincts des différents éléments matériels définis procède, outre des considérations de nature et de performances que nous venons de citer, de la prise en compte de contraintes plus générales de la mission, qui caractérisent l'ensemble du domaine de l'application envisagée. Ces contraintes générales se traduisent également par des règles d'utilisation des éléments matériels du système de commande.

I.1.1.6. Récapitulation

Nous rappelons sur le tableau de la figure I.2., les différents éléments que nous venons de présenter.

Environnement	Système physique Opérateur humain Milieu extérieur	Caractéristiques et évolution
Mission	Actions	Contraintes de l'application Contraintes générales
Charge du système	Tâches	Coordination des tâches
Ressources logicielles	Fonctions élémentaires	Allocation des ressouces
Ressources matérielles	Eléments de la structure	Règles d'attribution des moyens

FIGURE I.2.

Les liaisons entre les éléments de ces différents niveaux sont très nombreuses, mais ne sont pas toutes considérées en bloc lors de la conception d'un système informatique ; elles sont prises en compte par groupes qui diffèrent en fonction de la phase de la conception dans laquelle on se trouve.

I.1.2. Les quatre phases de la conception d'un système

I.1.2.1. Première phase

La première phase est exclusivement concernée par les deux premiers ensembles que nous avons regroupés sous le nom de contexte de l'application (figure I.3.).

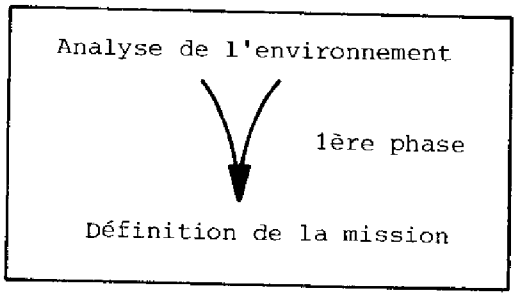


FIGURE I.3.

Cette phase regroupe trois activités et conduit à la définition de la mission :

- acquisition d'une bonne connaissance de la nature et de l'évolution de l'environnement pour aboutir à la définition des actions que doit effectuer le système,
- obtention de renseignements sur la dépendance entre ces actions, leur importance relative, leur durée, la fréquence nécessaire de leur exécution, leur priorité, l'urgence de leur prise en compte qui déterminent ce que nous avons appelé les contraintes propres à l'application,
- énoncé des contraintes générales imputables au type de l'application envisagée.

On est ainsi ramené au schéma de la figure I.4.

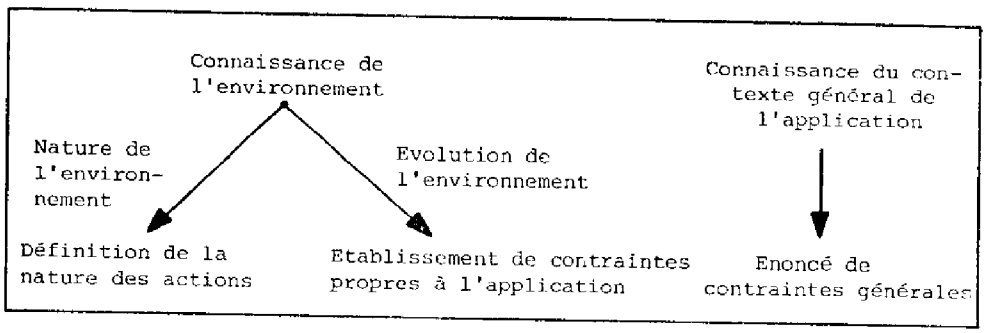


FIGURE I.4.

I.1.2.2. Deuxième phase

La deuxième phase, qui aboutit au choix d'une structure, consiste à définir les éléments du système de commande.

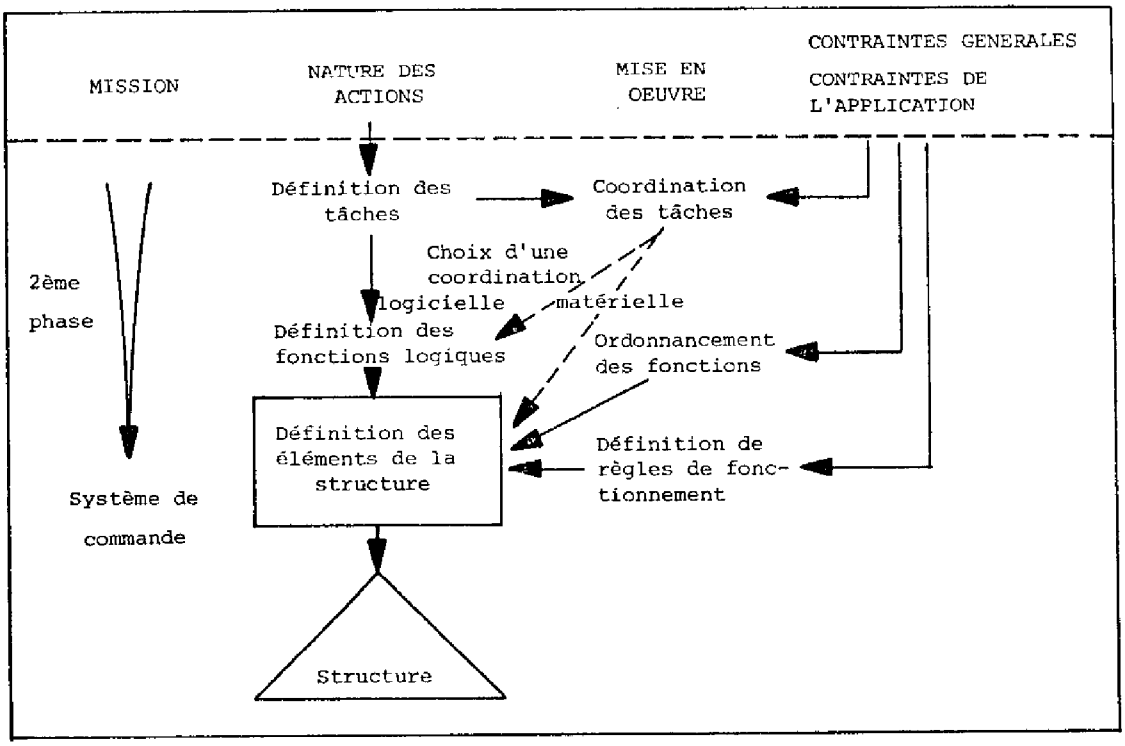


FIGURE I.5.

Les étapes successives de cette phase permettent d'accéder, à la suite du choix à chacun des niveaux, à de nombreuses informations sur les ressources matérielles nécessaires pour remplir la mission : ceci est concrétisé par la convergence des chemins fléchés apparaissant sur le schéma de la figure I.5.

Ces éléments doivent permettre soit la définition d'une nouvelle structure, soit le choix, parmi le matériel existant, d'une structure se rapprochant suffisamment des résultats obtenus.

Il convient de noter qu'à ce niveau, il est indispensable de posséder une expérience du matériel que l'on est susceptible d'employer, tant sur le plan de la nature des éléments (fonctions et rapidité des modules) que sur le plan de l'architecture des systèmes.

La démarche que nous venons de présenter ne constitue évidemment qu'un cliché simplifié d'une procédure vers laquelle il faut tendre car, en réalité, il existe toujours une boucle de retour qui se traduit par une remise en question de choix effectués aux niveaux supérieurs à partir de la structure qui s'est détachée de la première approche. Une bonne connaissance du matériel est ici très efficace, car elle permet d'aiguiller, dès le début, les décisions vers des solutions réalisables.

I.1.2.3. Troisième phase

La troisième phase consiste à comparer les différentes solutions possibles et à évaluer les performances que l'on peut attendre des systèmes retenus, pour juger de leur faculté de remplir la mission fixée.

La plupart des études menées dans le cadre de l'évaluation des performances des systèmes se limitent à l'observation de leur comportement en fonction d'un seul type de composantes (temps, sûreté de fonctionnement) ; certaines regroupent plusieurs composantes dans une même analyse :

- performances fonctionnelles, performances temporelles, coût,
- performances fonctionnelles, performances temporelles et performances en regard de la sûreté de fonctionnement pour des systèmes pour lesquels un fonctionnement dégradé est envisagé.

I.1.2.4. Quatrième phase

La quatrième phase de la conception d'un système de commande consiste à vérifier et à préciser les performances calculées lors de la phase précédente en étudiant le comportement du système retenu face à une représentation de l'environnement de plus en plus précise.

Cette dernière étape est indispensable car les résultats approchés obtenus dans la troisième phase peuvent ne pas prendre en compte certains phénomènes tels que des phénomènes de surcharge locale et masquer éventuellement un blocage du système.

De plus, les hypothèses simplificatrices faites au niveau de la description du système de commande et de l'environnement pour obtenir des évaluations des performances, malgré la complexité du problème, peuvent s'éloigner trop du comportement réel du système ou de l'environnement et conduire à des résultats non conformes avec la réalité.

I.2. PRESENTATION DU MEMOIRE DE THESE

Ce mémoire constitue un développement de la démarche de conception d'un système que nous venons de présenter.

Lorsque l'on se préoccupe d'un certain type d'application, défini essentiellement par un créneau de performance et par la nature dominante des actions à effectuer, on dispose en général de suffisamment d'informations pour figer les caractéristiques essentielles du système de commande nécessaire (architecture et principe de fonctionnement).

I.2.1. Première partie

La première partie de ce mémoire est concernée par les phases 1 et 2 de la conception d'un système.

Le type d'application visé est caractérisé par :

- une complexité moyenne (performances dans la gamme de celles des systèmes de la microinformatique),
- une grande activité de communication avec le milieu extérieur,
- la réalisation de nombreuses fonctions de l'automatique.

Notre propos est de cerner les relations qui existent entre les contraintes générales fixées pour ce type d'application et de présenter comment nous les avons répercutées sur la définition du système de commande.

Cette analyse des fonctions que doit réaliser le système de commande conduit, parallèlement à une remise en question des structures classiques, à une architecture originale d'un système multiopérateurs, obtenue par un découpage matériel qui reproduit la différence de nature des actions élémentaires du système.

L'architecture de cette structure présente des points communs avec les structures réparties ou les structures parallèles, mais son fonctionnement est avant tout celui d'une structure monolithique classique, caractérisée par la simplicité due à la nature séquentielle de son fonctionnement.

Les choix fondamentaux qui ont conduit à ce type de structure ont été faits en 1971 dans le cadre du projet A.S.M.O.D.E.E. [COS.73] - [LAP.74].

Ce projet a donné lieu à trois réalisations :

- A.S.M.O.D.E.E. 01 [ALA.72],
- A.S.M.O.C.A.S. [COS.73],
- A.S.M.O.D.E.E. 02 [LAN.75].

La structure obtenue est particulièrement souple et répond aux performances nécessaires à l'application envisagée par l'utilisation de modules fonctionnels homogènes à l'application, tant en performances qu'en nombre.

Ces modules constituent des composants de l'automatique au même titre que les microprocesseurs constituent de nouveaux outils pour l'informatique.

I.2.2. Seconde partie

La seconde partie traite de l'évaluation des performances du système de commande retenu.

Nous utilisons à cette fin des modèles mathématiques et plus particulièrement des descriptions par chaînes de MARKOV.

La maquette réalisée n'ayant pas conduit à des applications industrielles, nous n'avons pas pu réaliser la quatrième phase de la conception du système, c'est-à-dire valider les résultats obtenus par une simulation du comportement du système dans son environnement réel ou simulé.

C'est pour cette raison que nous insisterons essentiellement dans cette seconde partie sur les possibilités ou les difficultés de prise en compte de tout un ensemble de contraintes réalisant ainsi une étude essentiellement qualitative des performances temporelles de systèmes multi-opérateurs.

1ère PARTIE

CONCEPTION ET RÉALISATION

D'UN SYSTÈME DE COMMANDE MULTIOPÉRATEUR

-

CHAPITRE II

CONCEPTION D'UN SYSTEME DE COMMANDE :

A.S.M.O.D.E.E. 02

-

II.1. PARTICULARITES DES SYSTEMES INFORMATIQUES POUR LA COMMANDE AUTOMATIQUE

L'insertion de systèmes informatiques dans la conduite en temps réel de processus se traduit par des activités à caractère automatique et informatique.

Nous donnons dans le tableau de la figure II.1. des exemples de fonctions de l'automatique et de l'informatique.

FONCTIONS DE L'AUTOMATIQUE	FONCTIONS DE L'INFORMATIQUE
Régulation Filtrage Conversions(A/D) et (D/A) Conversions (série/parallèle) et(parallèle/série)	Manipulations de données pour l'établissement de bilans Manipulations de fichiers

FIGURE II.1.

L'importance relative de ces deux types d'activités, dont nous allons analyser les implications, est essentiellement liée à l'autorité qui est confiée au système de commande, ou, si l'on préfère, aux différences de degré et de niveau de l'intervention du système de commande et de l'homme.

On peut distinguer, dans ce cadre, deux catégories principales de systèmes selon qu'il y a intervention de l'homme ou non :

- la pleine autorité est confiée au système lorsque l'opérateur humain n'intervient pas en fonctionnement normal ; les fonctions utilisées par le système de commande sont alors à caractère essentiellement automatique,
- l'autorité conférée au système est plus ou moins limitée lorsque l'intervention de l'homme se situe ou non au niveau de la prise de décision ; dans ce cas, le système de commande exécute, parallèlement aux fonctions de l'automatique qui ont suscité son emploi, des activités d'ordre informatique à des fins statistiques par exemple.

II.1.1. Particularités des actions de la commande de processus

Outre la nature des fonctions propres à l'automatique, certaines caractéristiques, d'ordre plus général, particularisent la conduite de processus.

II.1.1.1. Particularités des tâches

- prédominance de certains types d'action et, en particulier, grande activité de communication avec le milieu extérieur,
- stabilité des programmes : la mission d'un système de commande est souvent figée et le programme de l'application est peu susceptible d'être modifié une fois qu'il est mis au point,
- fort déterminisme et lien entre les actions : les actions de l'automatique se distinguent des tâches informatiques qui sont plus indépendantes et dont l'application est souvent de nature aléatoire.

Le tableau de la figure II.2. montre certains traits caractéristiques des deux types de tâche.

TACHES AUTOMATIQUES	TACHES INFORMATIQUES
Durées courtes et figées	Durées longues et aléatoires
Apparitions fréquentes et périodiques	Apparitionspeu fréquentes et aléatoires
Urgentes	Peu urgentes
Liées les unes aux autres	Indépendantes

FIGURE II.2.

REMARQUE

Les tâches dont la fréquence de répétition est faible en regard de celle de l'ensemble des autres tâches de l'application sont en général considérées comme des tâches dont la demande se fait de manière aléatoire ; ceci permet d'éviter la perte de temps qu'entrainerait une scrutation programmée systématique d'un registre indiquant l'état de cette tâche.

II.1.1.2. Particularités des données

La commande de processus est souvent particularisée par un petit nombre de données dont la durée de vie est courte et qui subissent un grand nombre de transformations.

Ces données sont en général directement liées aux grandeurs physiques de l'application -lecteur de points de mesure, émission d'une valeur de commande- et sont à la base de la grande activité de communication qui existe avec le milieu extérieur.

II.1.2. Contraintes sur le système de commande

Les contraintes qui interviennent sur les systèmes dans ce type d'application sont nombreuses ; nous les situons dans la figure II.3. par rapport à trois contraintes fondamentales qui recouvrent, à notre avis, toutes les autres.

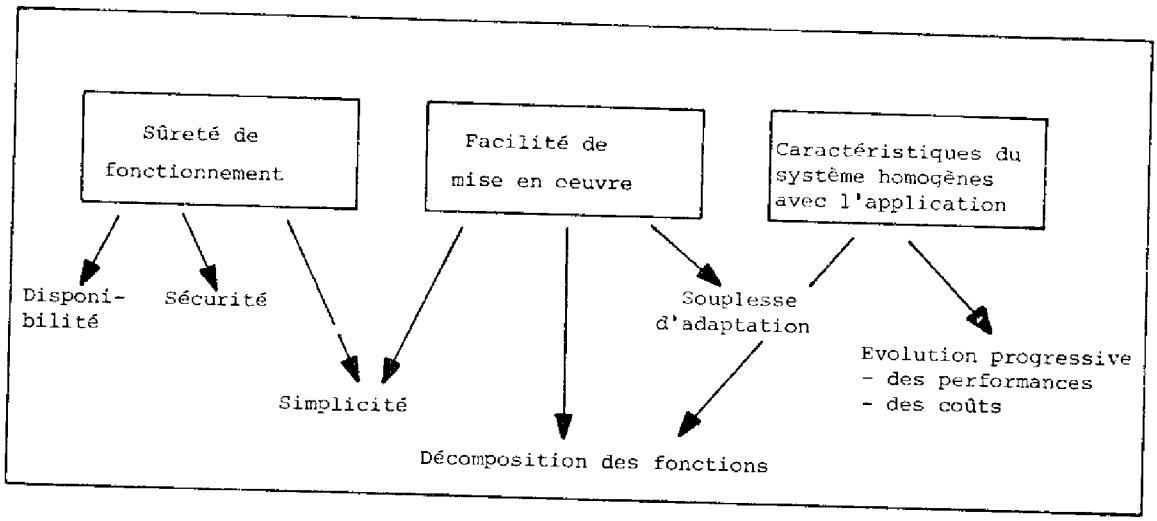


FIGURE II.3.

II.1.2.1. Une grande disponibilité

Un matériel qui justifie l'emploi d'organes numériques de commande pour l'amélioration des performances est en général trop onéreux pour que l'on puisse se permettre de le garder dans un état inopérant, à la suite de l'apparition d'un cas de non fonctionnement au système de commande.

II.1.2.2. Une grande sécurité

A un autre niveau, le coût des installations commandées par le système numérique peut être tel que cette autre composante de la sûreté de fonctionnement devienne un critère impératif : c'est-à-dire qu'il faut se prémunir de pannes dites catastrophiques qui endommageraient les installations.

II.1.2.3. Une simplicité du fonctionnement et de l'architecture

La simplicité du fonctionnement et de l'architecture d'un système est nécessaire à l'obtention de bonnes performances en ce qui concerne la sûreté de fonctionnement ; elle est indispensable également pour une mise en oeuvre facile du système par des automaticiens qui doivent consacrer leur attention aux tâches propres à l'application, sans trop se préoccuper de tâches de gestion de ressources. Le système doit constituer une boîte noire dont la programmation ne nécessite pas une connaissance approfondie de la structure.

II.1.2.4. Une décomposition naturelle des fonctions

Outre les facilités de compréhension du système qu'elle entraîne, une telle décomposition permet une mise au point progressive de la commande réalisée en s'occupant des fonctions locales les unes après les autres.

II.1.2.5. Une souplesse d'adaptation

L'adaptation d'un système à une application donnée se fait d'une manière aisée lorsque la création des moyens matériels et logiciels spécifiques de cette application n'entraîne pas de répercussion sur toutes les parties déjà existantes du système.

II.1.2.6. Une évolution progressive du système

Une caractéristique essentielle de la conduite de processus étant que le coût de la commande est généralement faible devant celui du système à commander, les performances du système doivent se limiter aux performances exigées et rester homogènes avec la complexité de la mission.

Il est nécessaire pour cela que le matériel et le logiciel soient en mesure de croître par petits incréments.

II.2. LA DEMARCHE SUIVIE

L'analyse des contraintes générales de la conduite de processus permet de figer certains éléments des systèmes de commande, et d'obtenir ainsi des structures adaptées à toute une gamme d'applications.

L'optique que l'on poursuit est de déterminer les caractéristiques essentielles d'un système répondant aux contraintes que nous avons énoncées, et pour lequel seule une partie de la structure reste à concevoir lorsque l'on étudie une nouvelle application ; cette partie spécifique est alors ajoutée à un matériel déjà existant à la manière d'un mécano. Une telle démarche permet de parvenir rapidement à un système opérationnel, ce qui est un facteur déterminant dans le coût général du sous-ensemble de commande.

Le but de ce paragraphe est de présenter comment nous avons fait intervenir les différentes contraintes dans la conception d'un système de commande. La politique suivie est marquée par :

- une analyse des différentes actions à effectuer au niveau de chaque élément d'information, indépendante de toute considération d'architecture,
- la prise en compte de l'évolution technologique qui a bouleversé les règles qui présidaient à l'élaboration de systèmes.

Le développement extraordinaire des composants de l'électronique se traduit à la fois par la réalisation de fonctions très évoluées de nature variée et par un prix qui devient faible. Ce phénomène permet d'envisager la conception d'un système sous un angle nouveau et amène deux conséquences :

- il n'est plus nécessaire de rechercher systématiquement une optimisation de l'utilisation des éléments et ceci peut avoir de très fortes répercussions sur l'architecture qui découle alors plus directement d'autres contraintes (sûreté de fonctionnement, simplicité,...),

- les possibilités de stockage apportées par les mémoires à semiconducteurs font que l'ensemble des programmes qui constituent la "charge du système" peut résider de manière permanente dans la mémoire de programme, sans nécessiter la gestion d'une mémoire de masse.

La figure II.4., qui reprend les contraintes données sur la figure II.3., regroupe les principaux choix effectués et présente la démarche qui a conduit à la conception de l'automate A.S.M.O.D.E.E. 02.

REMARQUE

Nous nous sommes intéressé uniquement aux performances temporelles des systèmes de commande dont tous les éléments sont en état de bon fonctionnement, et nous n'avons pas cherché dans cette étude, à analyser l'impact des contraintes introduites par les composantes de la sûreté de fonctionnement.

II.3. PRISE EN COMPTE DES PRINCIPAUX CHOIX

II.3.1. Structure monoprocesseur ; flot d'instruction unique

Le choix d'un système de commande ayant un organe de décision unique est fondamental car il fixe une limite supérieure au niveau des performances envisageables. Il répond à la contrainte de simplicité des systèmes automatiques.

Les trois niveaux de définition du système de commande, tels que nous les avons présentés dans l'introduction de ce mémoire, sont identifiés sur le tableau de la figure II.5.

Charge du système	Tâches : programmes de l'application
Ressources logicielles	Fonctions élémentaires : instructions
Ressources matérielles	Eléments de la structure : éléments réalisant les instructions

FIGURE II.5.

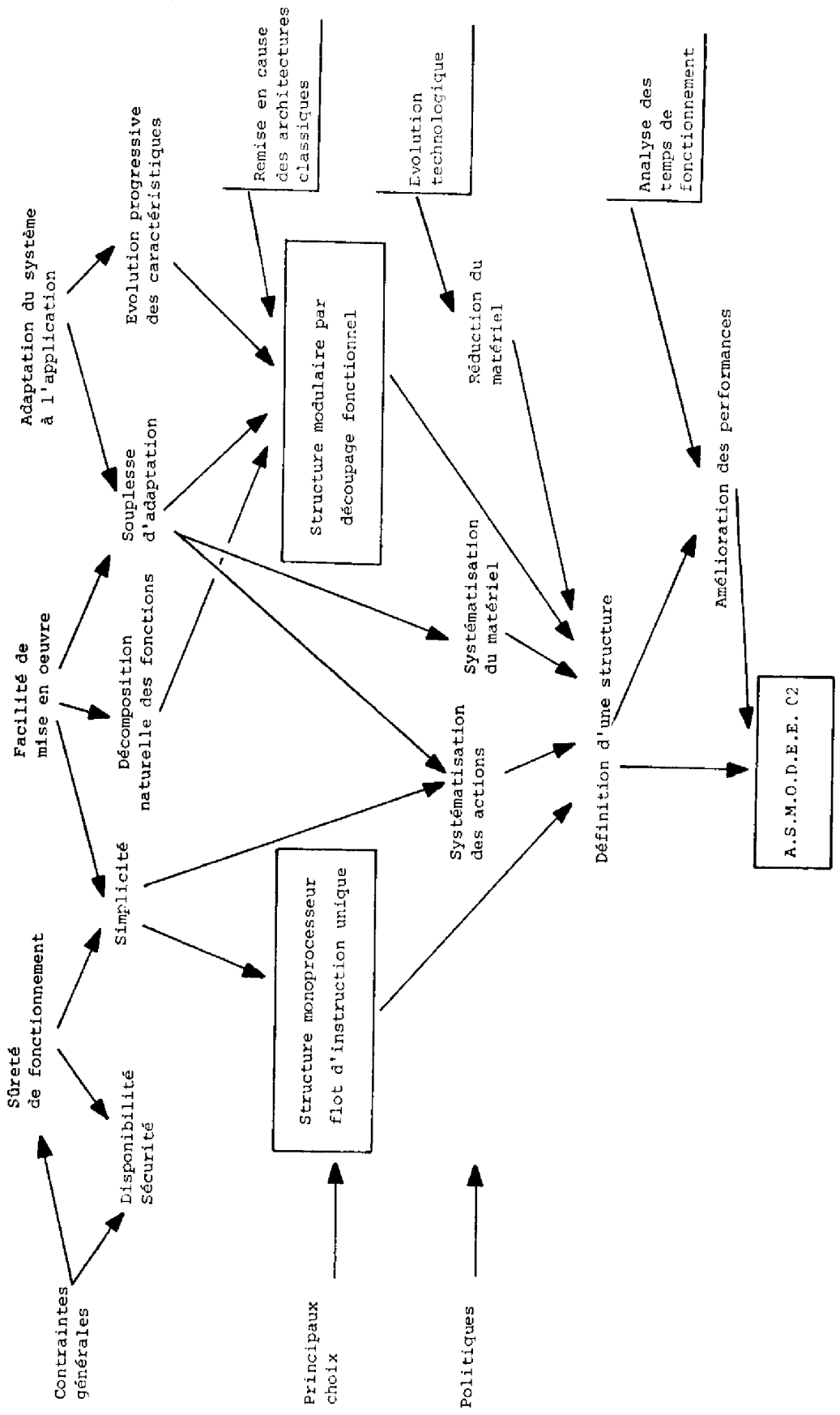

Contraintes générales
Sûreté de fonctionnement
Disponibilité Sécurité
Facilité de mise en oeuvre
Simplicité
Décomposition naturelle des fonctions
Adaptation du système à l'application
Souplesse d'adaptation
Evolution progressive des caractéristiques
Principaux choix
Structure monoprocesseur flot d'instruction unique
Politiques
Systématisation des actions
Structure modulaire par découpage fonctionnel
Remise en cause des architectures classiques
Systématisation du matériel
Evolution technologique
Réduction du matériel
Définition d'une structure
Analyse des temps de fonctionnement
Amélioration des performances
A.S.M.O.D.E.E. C2

FIGURE II.4.

La raison pour laquelle nous avons limité les possibilités de la machine à la scrutation d'une seule instruction à la fois est essentiellement liée à la simplicité de fonctionnement ; en effet, la complexité des algorithmes d'exploitation des ressources, propre au multitraitement, [BRUN.74] - [LAG.76] - [WUL.74] , est contraignante et s'oppose plus au caractère évolutif dont nous voulons doter le système que les difficultés soulevées par la complexité du matériel.

II.3.2. Structure modulaire obtenue par un découpage fonctionnel

II.3.2.1. Originalité de cette approche

Une analyse des différentes actions élémentaires d'un système informatique permet de poursuivre très loin dans la voie d'une décomposition naturelle des fonctions, ouverte par la décomposition des fonctions de l'automatique.

L'activité que l'on peut qualifier d'efficace dans un système de commande concerne la génération des commandes, la saisie et la transformation de données. Mais cette activité s'accompagne nécessairement d'actions "parasites" comme :

- le décodage des instructions,
- la sélection de l'instruction suivante.

Nous avons voulu tenir compte de cette différence de nature qui existe au niveau des fonctions élémentaires réalisées au sein d'un calculateur, et la reproduire dans la constitution d'un système de commande.

Nous proposons ainsi une structure que nous qualifions de purement fonctionnelle pour insister sur la répartition des éléments en trois ensembles, en fonction de l'action qu'ils exercent sur les instructions :

- l'enchaînement des instructions,
- le stockage des instructions,
- l'exécution des instructions.

Ceci conduit donc à distinguer les trois types d'information suivants :

- les adresses,
- les instructions,
- les données,

et, par voie de conséquences, les ressources matérielles qui leur sont affectées :

- séparation du stockage des données et du stockage des instructions par exemple,
- séparation des moyens de communication propres à chaque type d'information .

Cette approche est résumée sur le schéma de la figure II.6.

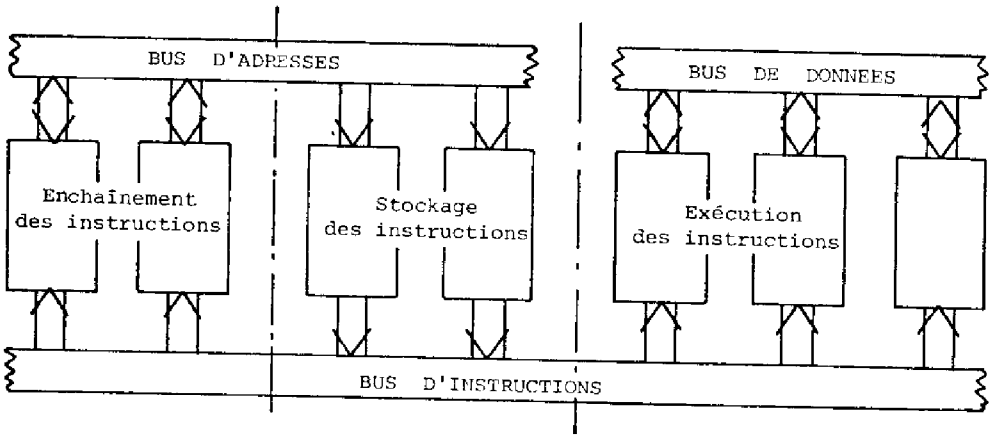


FIGURE II.6.

Bien qu'il ne soit ni possible, ni avantageux de maintenir intégralement la distinction fonctionnelle que nous avons faite apparaître à cause :

- de sauts sur test qui relient le chemin des données au chemin des adresses,
- des regroupements que nous ferons dans le but de réduire le matériel,

le système obtenu gardera l'originalité de cette décomposition que nous appellerons fonctionnelle dans la suite.

II.3.2.2. Conséquences d'un tel découpage

Le choix d'un découpage fonctionnel répond à plusieurs des contraintes propres au domaine de la conduite de processus.

La création d'autant de modules que de fonctions à exécuter permet de définir un système dont les caractéristiques sont directement liées à la complexité de l'application ; il est ainsi possible d'accroître les performances du système par petits incréments pour répondre à de nouveaux besoins posés soit par une augmentation de la charge initiale, soit par une utilisation ultérieure pour une application différente.

Cette évolution se fait d'une manière souple par le choix d'une structure où le regroupement des différents modules fonctionnels se fait de manière très simple.

Enfin, un tel découpage permet de tenir compte de l'importance des liaisons avec le milieu extérieur sur deux plans distincts :

- les organes d'entrée-sortie peuvent tenir une place importante dans ce type de structure,
- le milieu extérieur a la possibilité d'intervenir de manière aléatoire sur le système, pour demander l'exécution d'une tâche lorsque certains événements surviennent.

II.3.3. Constitution du système

La nature, le nombre et la complexité des modules de chacun des trois ensembles du système de commande sont susceptibles d'évoluer ; cependant, certains blocs sont indispensables et ils constituent une unité de base du système qui est elle-même modulaire et très réduite.

II.3.3.1. L'ensemble d'enchaînement des instructions

Il est constitué de plusieurs briques qui matérialisent les trois niveaux de description du système.

.L'organisation des tâches dans un contexte de multiprogrammation très fréquent dans la conduite de processus est réalisée par un opérateur de gestion des interruptions. Le choix d'une solution centralisée permet de lui conférer une complexité en rapport avec celle du problème posé par l'application. L'organe de gestion des interruptions scrute systématiquement l'état des demandes d'interruption et décide de l'exécution des programmes indépendants, de manière imbriquée dans le temps, en fonction des priorités affectées à chacune des tâches.

.L'organisation des instructions est réalisée par un opérateur compteur ordinal dont le rôle est de décoder les instructions concernant les adresses et d'effectuer tous les calculs correspondants.

.L'organisation du fonctionnement du système est assurée par un opérateur pilote qui génère l'ensemble des signaux nécessaires aux actions élémentaires du système de commande.

II.3.3.2. L'ensemble de stockage des instructions

Il est constitué de modules de faible capacité mémoire. Ces mémoires à semiconducteur sont des mémoires mortes ou reprogrammables dans la version définitive du système ; ceci est rendu possible par la "stabilité" du programme dans la majorité des applications.

II.3.3.3. L'ensemble d'exécution des instructions

L'ensemble d'exécution des instructions regroupe deux types d'opérateurs :

- ceux qui effectuent un traitement des données, de manière interne,
- ceux qui sont directement en liaison avec les interfaces réalisant les commandes ou les acquisitions des données.

II.3.3.4. Les différents modules

Nous avons regroupé sur le tableau de la figure II.7. différents modules qui sont classés en fonction de leur nature.

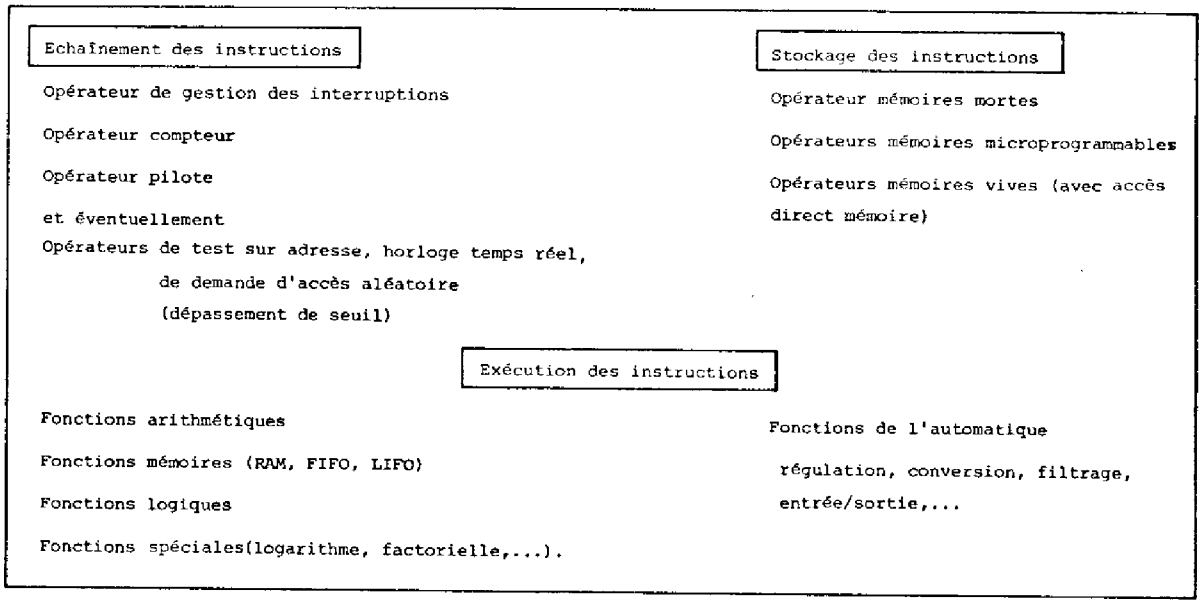


FIGURE II.7.

II.3.4. Les différentes politiques

II.3.4.1. La réduction du matériel

Deux points ont joué un rôle important dans la recherche d'une réduction du matériel :

- le choix de la technologie,
- l'emploi d'un bus généralisé unique.

L'emploi de la technique CMOS contribue largement à l'obtention d'un volume réduit en raison de ses :

- propriétés bidirectionnelles qui simplifient les problèmes d'interface avec le bus,
- hautes possibilités d'intégration qui permettent la création de fonctions évoluées sous un faible volume.

Malgré l'implication, en première analyse, de trois bus dans un découpage fonctionnel d'une structure, nous avons choisi de limiter la taille du réseau de connexion qui constitue le "point chaud" d'une structure modulaire. Pour ceci, nous avons adopté un bus généralisé unique, utilisé selon la technique du temps partagé, pour véhiculer les différents types d'informations du système (adresses, instructions, données et demandes d'interruption).

La minimisation du nombre des fils du bus entraîne une réduction de la quantité de matériel par une diminution de la taille de l'interface et des circuits de décodage.

L'unicité du bus modifie l'organisation du système ; son architecture est représentée sur le schéma de la figure II.8.

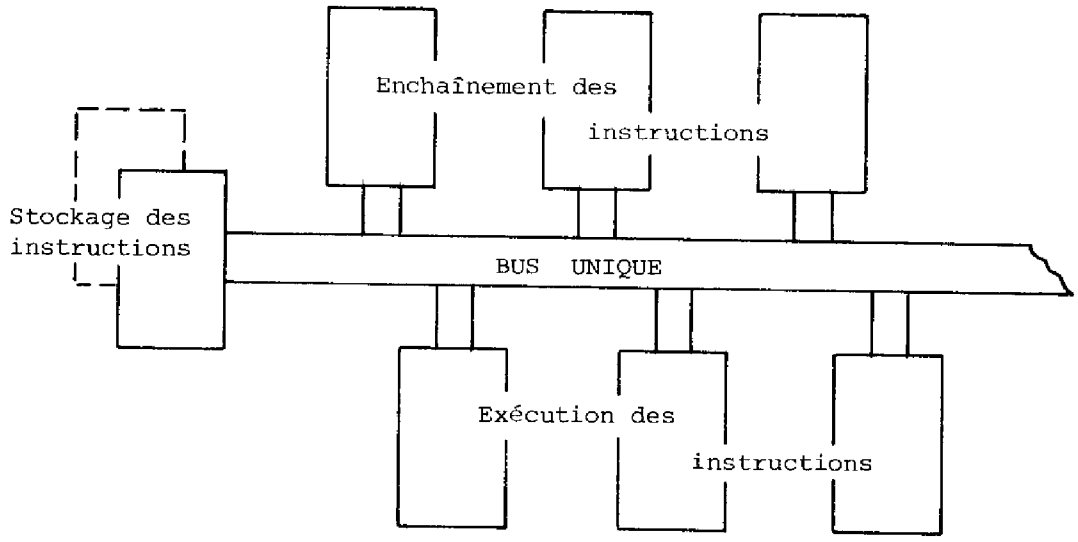


FIGURE II.8.

II.3.4.2. Systématisation du matériel

Parallèlement à la possibilité de faire varier la quantité de matériel, la souplesse conférée aux ressources matérielles contribue au caractère évolutif du système. Cette souplesse est donnée aux différents éléments par l'utilisation quasi-systématique de mémoires :

- dans le décodage, au niveau de chaque module ; ceci permet de systématiser l'interface des modules avec le bus généralisé unique et de créer ainsi un sous-ensemble identique sur chaque module,

- dans les modules fonctionnels, pour une spécialisation des traitements par l'écriture des mémoires de microprogramme,
- dans le séquenceur qui génère les différents états de la machine.

II.3.4.3. Systématisation des actions de la machine

Dans cette structure décentralisée où chaque opérateur joue le même rôle et où les circuits d'interface sont identiques, il est facile d'obtenir une systématisation du fonctionnement du système.

Une instruction réalise systématiquement, en un cycle machine, le transfert d'une donnée de l'opérateur où elle se trouve et que l'on appelle opérateur source, vers un opérateur dans lequel elle sera éventuellement traitée puis stockée, appelé opérateur destination. Le format des instructions est alors unique, composé de deux champs d'adresse qui définissent les codes des deux opérateurs.

Un cycle machine est lui-même constitué d'une succession d'états qui durent le temps d'un transfert entre deux modules. Ces échanges élémentaires sont dirigés par le sous-ensemble de gestion des instructions et sont réalisés par les interfaces communs à tous les opérateurs ; le traitement des données, qui peut être microprogrammé, est effectué dans la partie propre à chaque opérateur.

II.3.5. Définition d'une structure

On peut à présent regrouper les choix effectués sur un tableau (figure II.9) conforme à celui du paragraphe I.1.2.2. (figure I.5.) qui présente la démarche à suivre pour concevoir un système :

- la colonne "nature des actions" caractérise les choix pariculiers qu'il faut faire lorsque l'on a une mission donnée à accomplir,
- la colonne "mise en oeuvre" est propre à la structure A.S.M.O.D.E.E. et définie à priori pour toute une classe d'applications.

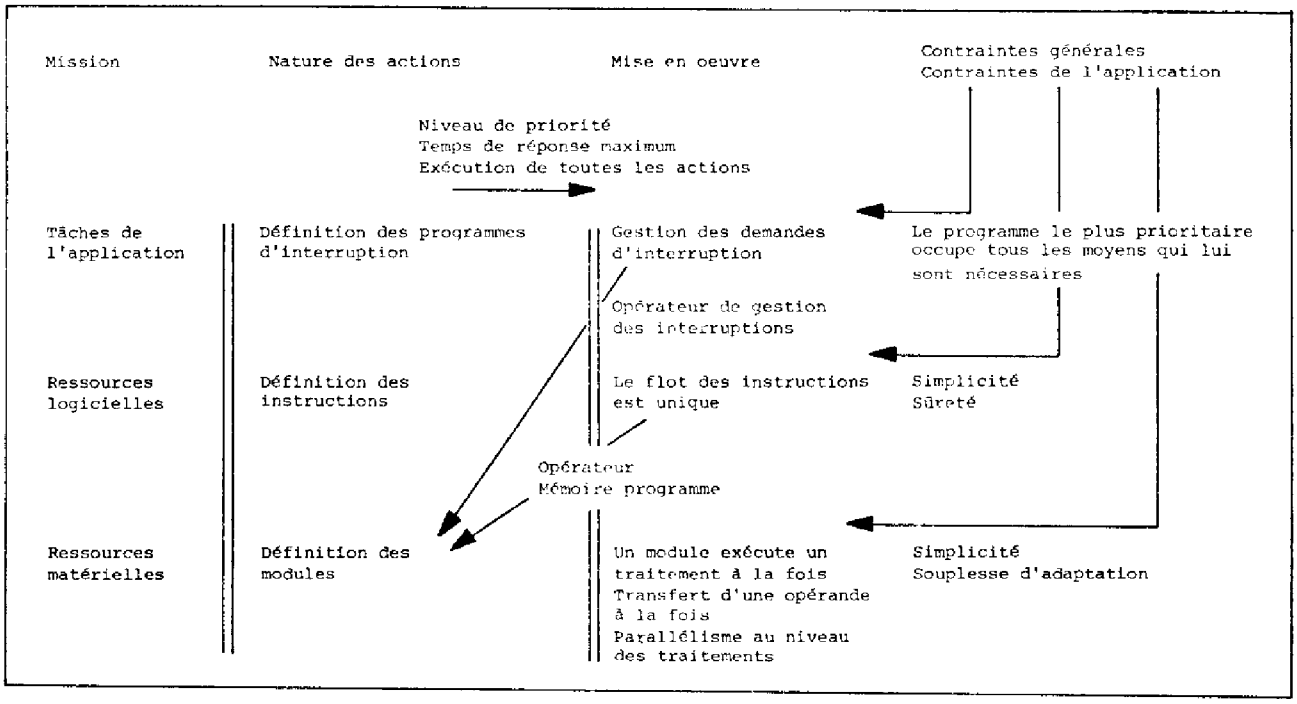


FIGURE II.9.

La dernière règle de mise en oeuvre apparaissant sur le tableau de la figure II.9. -parallélisme au niveau des traitements- fait l'objet du paragraphe suivant.

II.4. ANALYSE DES TEMPS DE FONCTIONNEMENT

II.4.1. Conséquences des choix et politiques sur les temps de fonctionnement

II.4.1.1. Comparaison avec une structure classique

Un système classique a une architecture organisée autour de deux pôles :

- la mémoire, qui réalise toutes les fonctions de stockage,
- l'unité centrale qui est le seul organe apte à décoder et à exécuter les instructions.

Dans le cas de structures classiques, aux performances moyennes et au fonctionnement essentiellement séquentiel, le fonctionnement se résume aux deux successions d'actions que nous présentons sur les figures II.10. et II.11., et qui couvrent un ou plusieurs cyles machine.

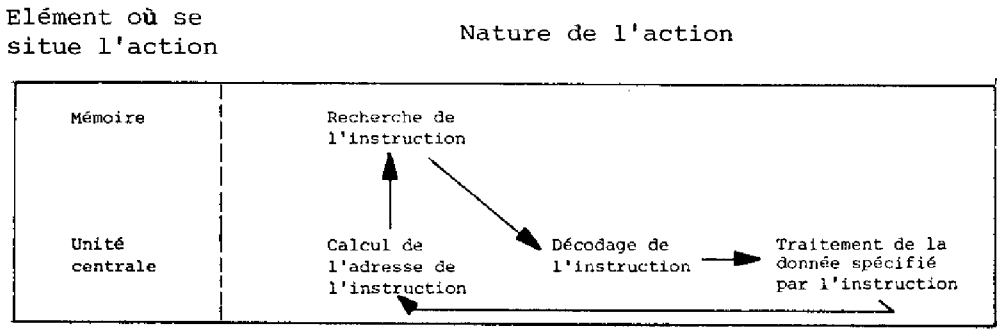


FIGURE II.10. Traitement d'une donnée rangée dans l'unité centrale

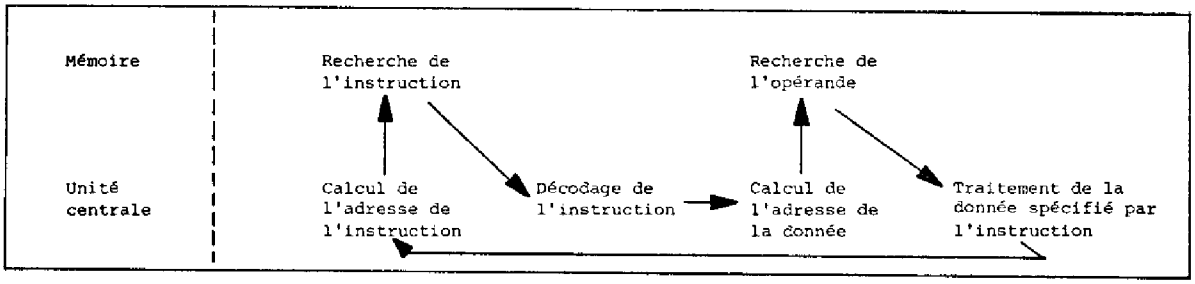


FIGURE II.11. Recherche d'une donnée en mémoire, puis traitement dans l'unité centrale

Une représentation analogue du fonctionnement de la structure modulaire fonctionnelle que nous présentons conduit au schéma de la figure II.12.

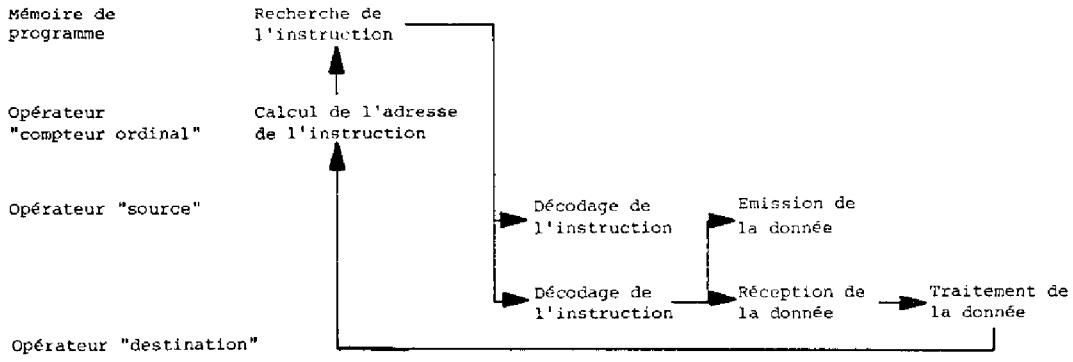


FIGURE II.12.

A priori, un tel système, qui réalise cinq actions élémentaires selon un ordre matérialisé par les flèches de la figure II.12., n'est pas pénalisé sur le plan du temps d'exécution par rapport aux architectures classiques qui réalisent soit quatre, soit six actions élémentaires comme cela est indiqué sur les figures II.10. et II.11. Il faut en réalité faire une analyse plus précise pour pouvoir porter un jugement : tel est le but du paragraphe suivant.

II.4.1.2. Les temps de fonctionnement

Perte de temps due à une réduction du matériel

Pour les deux types de structure, l'utilisation de moyens communs (le bus de communication par exemple) s'accompagne d'une diminution de la vitesse de fonctionnement. Il y a deux raisons à ce phénomène :

- la première, qui est la plus apparente, tient au fait que lorsqu'il n'y a pas une identité fortuite des formats, l'utilisation d'un moyen de communication unique pour des informations de type différent entraîne, dans un souci de compromis évident, la nécessité d'un transfert en plusieurs bytes des informations les plus longues,
- la seconde découle de la technique du multiplexage temporel : en effet, outre la nécessité de couches logiques supplémentaires comme des étages de puissance en sortie et des étages de protection en entrée des circuits utilisés, l'emploi de bus multiplexés nécessite l'introduction de fonctions logiques supplémentaires dans les liaisons, comme l'indique la figure II.13.

Perte de temps due à la systématisation

Les choix effectués dans un but de systématisation du matériel ou du fonctionnement de la structure ont des conséquences sur les temps d'exécution, soit directes, lorsqu'on affecte la même durée à toutes les actions élémentaires de la machine, soit indirectes lorsque l'on décide d'utiliser des mémoires pour simplifier le codage ou pour introduire un niveau de programmation.

Ce phénomène est de plus accentué par le fait que l'on introduit un ralentissement des transferts entre les deux premiers niveaux, dans le but de réduire la quantité de matériel ; ceci est représenté schématiquement sur la figure II.14.

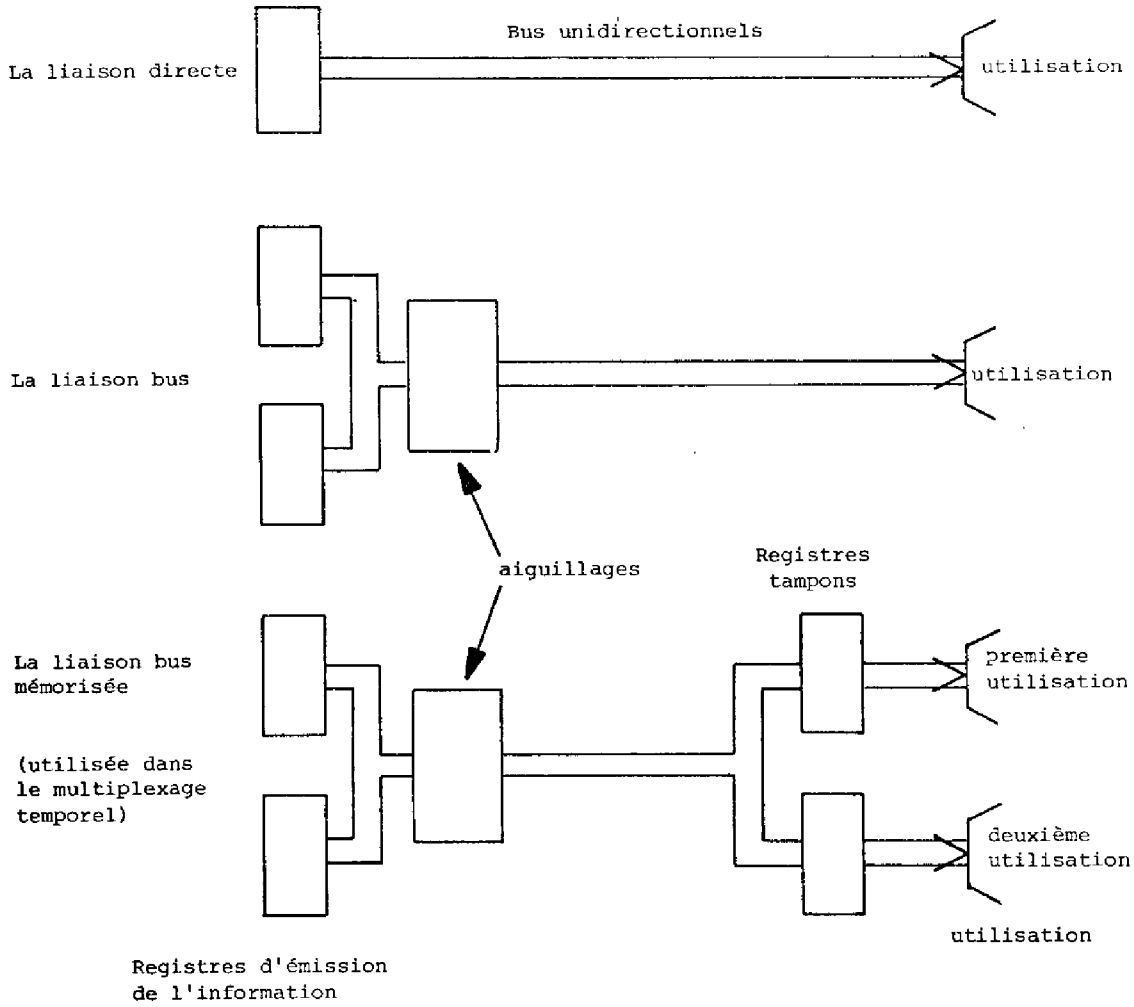


FIGURE II.13.

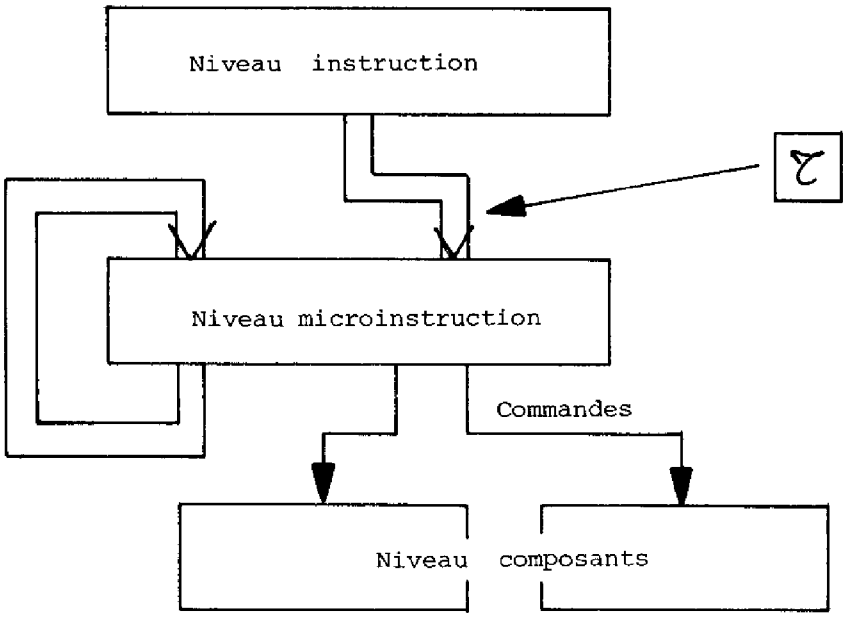


FIGURE II.14.

II.4.2. Recherche d'une plus grande vitesse

Le principe de la transmission par bytes,l'utilisation d'un bus multiplexé, le décodage effectué par mémoires, la systématisation de la longueur d'un état et surtout la microprogrammation des traitements constituent cinq raisons qui, ajoutées au choix d'une technologie lente, conduisent à des temps de fonctionnement importants.

La recherche d'un gain de vitesse se limite à une amélioration des performances d'une structure dont l'organisation logique est figée : en effet, cette dernière phase de la conception d'un système de commande ne doit pas remettre en cause les principaux choix qui ont été effectués durant les phases précédentes.

La succession des actions élémentaires que doit faire le systè-système de commande auquel nous sommes parvenu est rappelée sur la figure II.15.

L'analyse de l'organisation de cette liste d'actions -ordre d'apparition et emplacement où elles se situent- nous permet de déterminer les principes permettant un gain de performance qui sont les plus facilement exploitables dans le cadre de ce projet.

II.4.2.1. Les différents principes permettant une plus grande vitesse

Nous avons regroupé sur le tableau de la figure II.16. les politiques que l'on peut généralement retenir pour obtenir des structures fonctionnant plus rapidement.

Le découpage fonctionnel de la structure présentée se prête particulièrement bien à l'introduction d'un parallélisme implicite, c'est-à-dire d'une réalisation en parallèle de certaines actions du système qui n'est pas décidée par le programmeur.

Les procédés correspondant à cette méthode sont développés dans le paragraphe suivant.

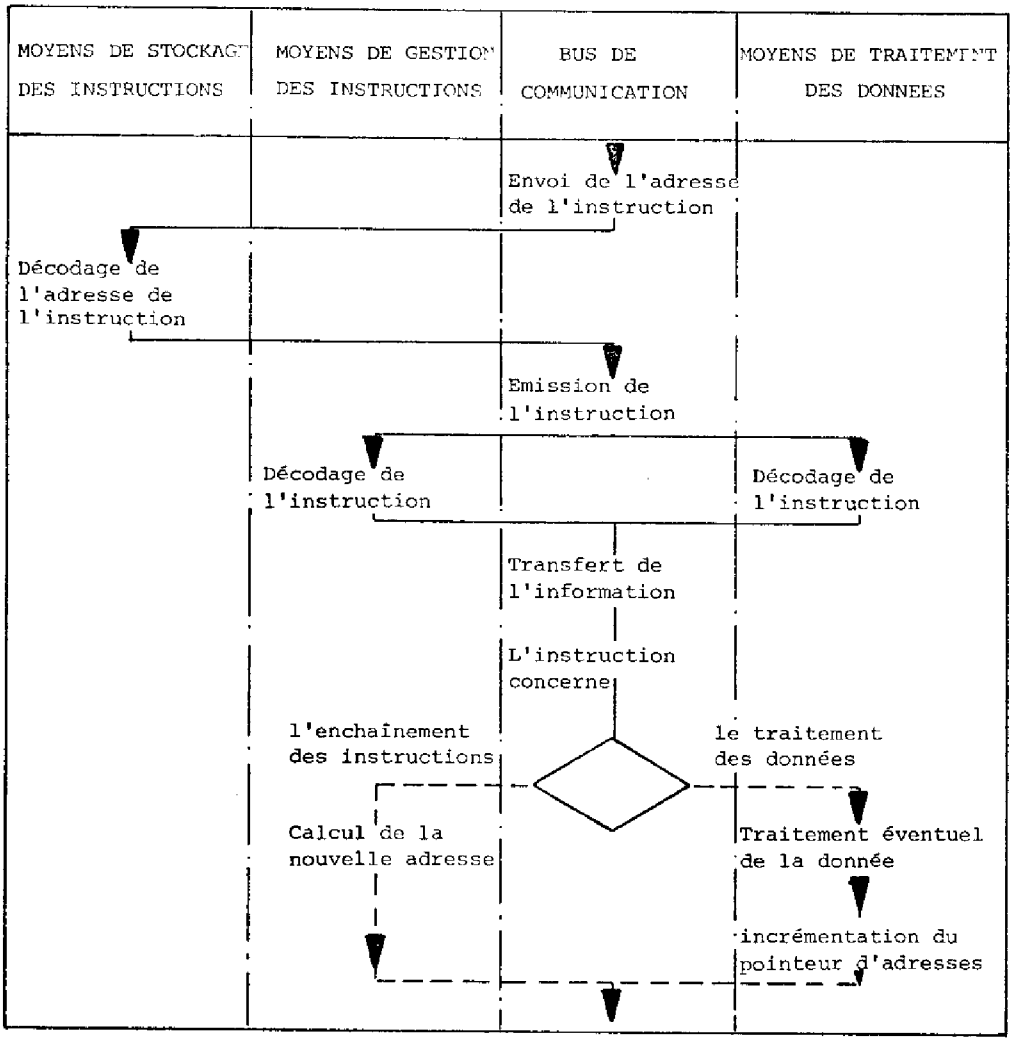


FIGURE II.15.

Choix d'instructions plus performantes	Augmentation du nombre de champ d'adresses. Utilisation de fonctions plus performantes grâce à la microprogrammation.
Suppression de certains transferts	Recherche d'un découpage logique permettant la suppression de certains transferts (incrémentation du pointeur d'adresses dans l'ensemble de stockage des instructions par exemple). Augmentation de la longueur des transferts pour réduire la proportion de temps nécessaire à leur initialisation (recherche de plusieurs instructions ou de plusieurs données à la fois).
L'introduction d'un parallélisme explicite	(On appelle ainsi un parallélisme au niveau des traitements qui apparaît de manière explicite dans les structures : grandeurs vectorielles).
Les méthodes d'anticipation : le parallélisme implicite	Sélection anticipée de l'adresse suivante. Introduction d'un parallélisme au niveau des traitements structures multi-opérateurs. Fonctionnement en mode pipe-line.

FIGURE II.16.

II.4.2.2. Les méthodes d'anticipation

Sélection anticipée de l'adresse suivante

Le schéma de la figure II.15. donné au paragraphe précédent indique la présence d'une action élémentaire "incrémentation du pointeur d'adresse" dans le cas où l'instruction concerne le traitement des données.

L'existence d'un compteur ordinal spécialisé pour exécuter cette action permet d'effectuer celle-ci d'une manière anticipée à un moment où les moyens de gestion des instructions sont disponibles.

Lorsque l'instruction concerne un calcul d'adresse, l'action anticipée est annulée et il faut attendre la fin du calcul propre à l'instruction en cours pour connaître l'adresse de la nouvelle instruction.

Si l'instruction porte sur les données, l'incrémentation anticipée est validée et la prochaine instruction traitée est celle qui suit immédiatement dans l'écriture du programme.

Cette procédure est très employée car la majorité des calculateurs possède un pointeur d'adresse spécialisé qui peut être incrémenté de manière indépendante. L'intérêt de cette méthode réside dans le fait que dans 85 % des cas, les instructions se suivent dans l'ordre dans lequel elles sont écrites dans le programme [HUG.72].

La figure II.17. permet de montrer les transformations apportées par cette politique, la comparaison devant être effectuée avec la figure II.15.

Parallélisme au niveau des traitements : structure multi-opérateurs

Le test apparaissant sur le schéma de la figure II.17. souligne la différence entre les deux classes d'instructions (données, adresses). Cette distinction s'accompagne de contraintes qui ne sont pas identiques : il est nécessaire que l'exécution d'une instruction de gestion de programme soit terminée pour connaître l'adresse de l'instruction suivante, alors qu'il ne faut pas systématiquement connaître le résultat d'un calcul sur une donnée pour enchaîner avec les instructions en séquence.

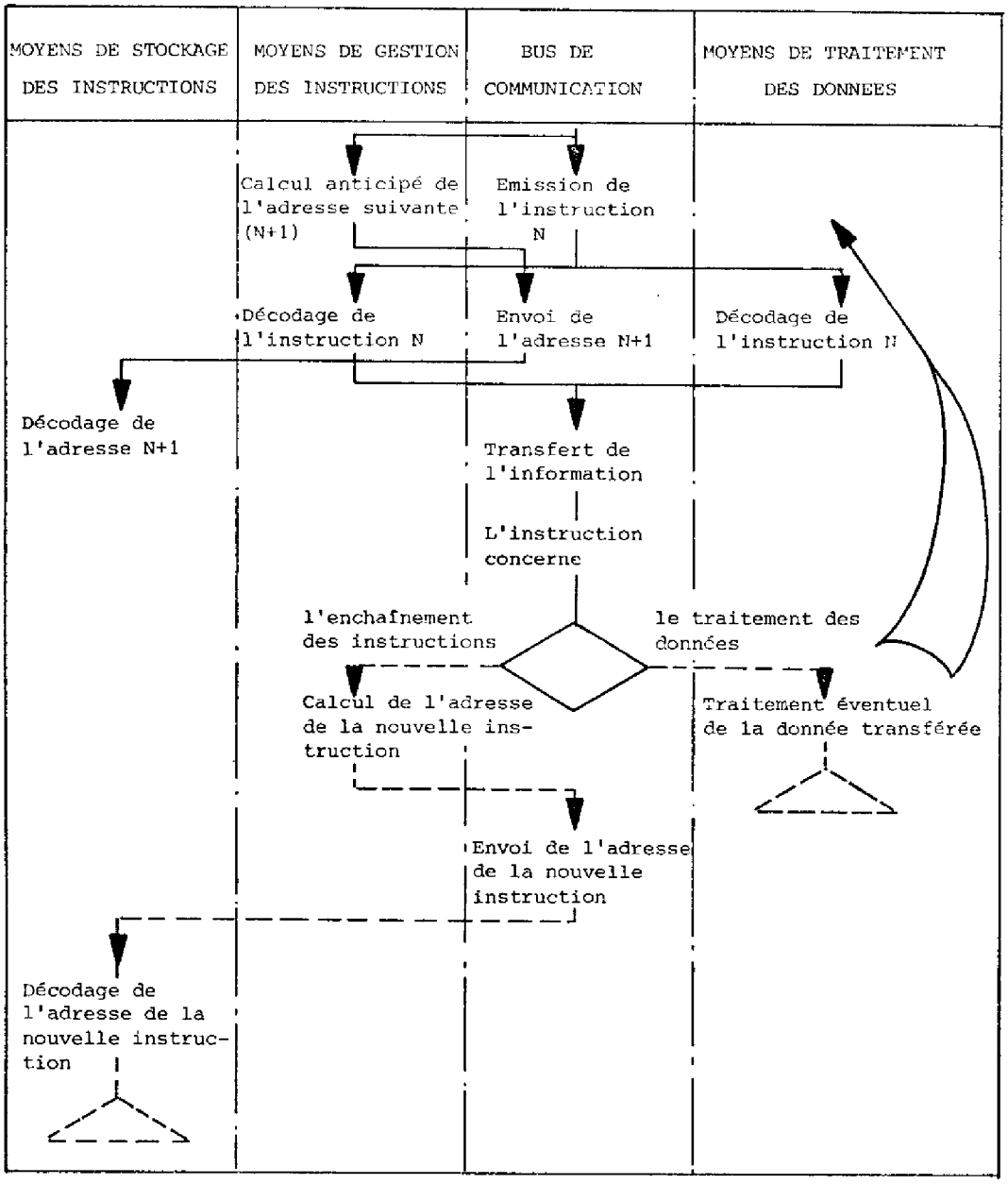


FIGURE II.17.

La durée des traitements, qui peut être longue et variable à cause de la microprogrammation, ne s'accorde pas avec la systématisation de la longueur des états. Cette raison, alliée à l'absence éventuelle de contraintes temporelles ou matérielles entre des instructions en séquence,

conduit à adopter un traitement des données qui s'exécute indépendamment du reste de la machine qui poursuit ses actions en parallèle, à chaque fois que cela est possible. Cette action est indiquée sur la figure II.17.

Le parallélisme ainsi introduit nécessite un fonctionnement par appel-réponse ou par test d'un bit de verrouillage, chaque opérateur appelé devant informer l'organe d'enchaînement des instructions de sa disponibilité. Le flot des instructions est interrompu, lorsqu'il y a référence à un module occupé, jusqu'à ce que celui-ci se libère.

La volonté de ne pas nuire à la simplicité du système par la création de files d'attente et de dispositifs de gestion des priorités en cas de conflits d'accès limite le parallélisme aux cas de "bon fonctionnement" où les opérateurs appelés sont disponibles. Ce parallélisme est appelé implicite parce qu'il se fait de manière fortuite, sans qu'il soit décidé par le programmeur. Nous indiquerons au paragraphe III.2., comment il est possible de favoriser le recouvrement des traitements en analysant de manière automatique les dépendances entre les instructions d'un programme.

Fonctionnement pipe-line

La possibilité de découper une action en segments d'égale durée et d'exécuter ces portions d'action dans des ressources indépendantes permet de faire fonctionner un système selon le mode pipe-line.

Chacun des transferts entre les modules constitue un état de la machine et regroupe :

- le temps de préparation de l'information (décodage et émission),
- le temps de la communication (traversée des circuits d'interface),
- le temps d'enregistrement dans le module récepteur.

Il est possible de diviser cet état en plusieurs segments et d'envisager un recouvrement de deux états successifs, au niveau de l'exécution de ces segments, dans le but d'obtenir une meilleure occupation du bus de communication, point chaud de ce type de structure.

Les trois politiques énoncées dans ce paragraphe recherchent un recouvrement des actions élémentaires du système pour augmenter le débit des instructions. Le gain des performances amené par un fonctionnement pipe-line systématique au niveau des transferts entre les modules (les états de la machine) est directement mesurable. Celui apporté par le mode pseudo pipe-line conféré à cette structure est difficile à apprécier et fait l'objet de l'étude que nous présentons dans la seconde partie de ce mémoire.

CHAPITRE III

REALISATION, PROGRAMMATION

D'A.S.M.O.D.E.E. 02

-

III.1. PRESENTATION DE LA MAQUETTE D'A.S.M.O.D.E.E. 02

Les principes retenus dans le chapitre II ont été intégralement suivis dans la conception du prototype A.S.M.O.D.E.E. 02 qui a été achevé en 1975. Le but de ce paragraphe est d'énumérer les principales caractéristiques de ce système de commande.

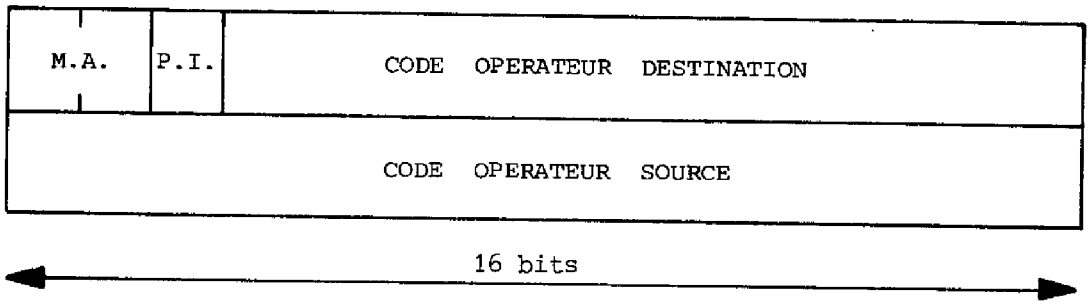
III.1.1. Présentation générale du système

La structure d'A.S.M.O.D.E.E. 02 est conforme au schéma de la figure II.8. Tous les modules fonctionnels sont reliés entre-eux par le bus généralisé de seize fils, multiplexé dans le temps, qui supporte tous les transferts d'information :

- adresses,
- instructions,
- données,
- demandes d'interruption .

Les signaux de contrôle et les alimentations sont véhiculés par un ensemble de dix fils.

Le format constant des instructions comporte deux champs principaux qui sont constitués des codes des opérateurs source et destination, et de trois bits de contrôle dont deux indiquent le mode d'adressage et le troisième l'autorisation d'interruption de l'instruction. Ce format est représenté sur la figure III.1.



M.A. = mode d'adressage
P.I. = autorisation d'interruption

FIGURE III.1.

Nous avons retenu les quatre modes d'adressage suivants :

- direct source - direct destination,
- immédiat source - direct destination,
- indirect source - direct destination,
- direct source - indirect destination.

Le prototype, dont le schéma de principe est représenté sur la figure III.2. comprend, outre les différents opérateurs d'enchaînement des instructions et de stockage des programmes, un opérateur arithmétique et logique, une mémoire de travail et un opérateur d'interface (console de commande et de visualisation).

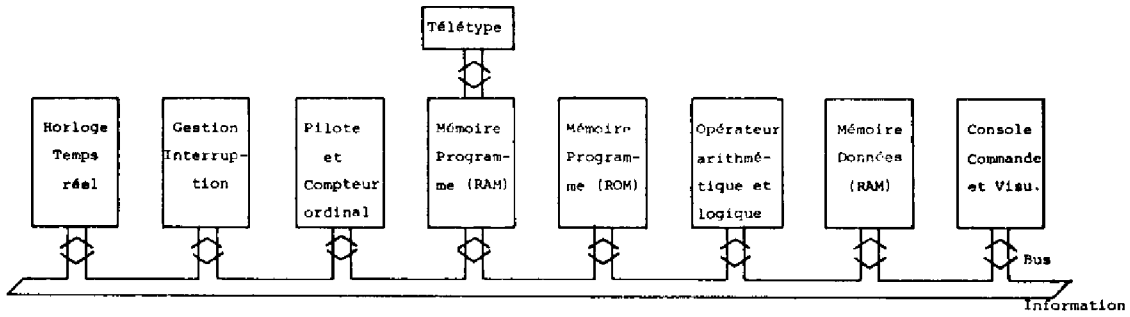


FIGURE III.2. Schéma de principe de la maquette A.S.M.O.D.E.E. 02

Les différents types d'instruction mis en oeuvre au niveau de la maquette sont répertoriés sur le tableau de la figure III.3. : ils correspondent aux codes "destination" des opérateurs développés.

Nous ne mentionnerons pas les instructions faisant référence à la mémoire de données ou à la console de commande et de visualisation ; pour celles-ci, chaque mot mémoire ou chaque porte d'entrée ou de sortie correspond à un code opératoire.

Instructions portant sur les adresses

	Saut inconditionnel	Saut sur test
Forçage du pointeur d'adresse à une adresse quelconque	Chaque saut peut être effectué avec ou sans sauvegarde automatique de l'ancienne adresse	
Addition / Soustraction d'un nombre quelconque à la valeur du pointeur d'adresse		

Deux codes "source " permettent de sortir

- le contenu du pointeur d'adresse,
- le contenu du registre de sauvegarde.

Instructions portant sur les données exécutées dans l'opérateur arithmétique et logique

(L'opérateur arithmétique et logique possède deux accumulateurs de 16 bits : A et B).

		Opérations sur 16 bits	Opérations sur 32 bits
Opérations d'écriture	Chargement de A	$y \rightarrow A$	$y \rightarrow A$ $0 \rightarrow B$
	Chargement de B	$y \rightarrow B$	$y \rightarrow B$ $0 \rightarrow A$
Opérations arithmétiques	Addition N fois	$(A) + Ny \rightarrow A$	$(B)+Ny \rightarrow B, (A)+retenue \rightarrow A$
	Soustraction N fois	$(A) + Ny \rightarrow A$	$(B)-Ny \rightarrow B, (A)-report \rightarrow A$
	Incrémentation N fois	$y + N \times 2^{-15} \rightarrow A$	$y+Nx2^{-31} \rightarrow B, (A)+retenue \rightarrow A$
	Décrémentation N fois	$y - N \times 2^{-15} \rightarrow A$	$y-Nx2^{-31} \rightarrow B, (A)-report \rightarrow A$
	Complément à 2	$c_2^t(y) \rightarrow A$	$c_2^t((A),y) \rightarrow A,B$
	Multiplication		$(A)xy \rightarrow A$(fort poids), $\rightarrow B$(faible poids)
	Division		$(A),(B):y \rightarrow A$(reste), $\rightarrow B$(quotient)
Opérateurs de décalage	Décalage N fois de (A)	droit avec écrite de "0"	gauche avec "0" ou "1" en fonction de (B)
	Décalage de (A)	circulaire droit numérique droit	
Opérations logiques	ET	$(A) . y \rightarrow A$	
	OU	$(A) + y \rightarrow A$	
	OU EXIL	$(A) \oplus y \rightarrow A$	

Deux codes sources permettent de sortir les contenus des accumulateurs A et B.

FIGURE III.3.

III.1.1.1. Les interruptions

Le système de gestion des interruptions adopté est centralisé. Toutes les demandes d'interruption sont transmises, à raison de une par fil du bus de communication, à l'opérateur de gestion des interruptions pendant un état systématique de scrutation. Cette solution centralisée permet une plus grande souplesse du système par l'adoption d'un opérateur de gestion des interruptions simple (priorités fixes précablées) ou sophistiqué (allocation dynamique des priorités), en fonction de l'application envisagée. Cet opérateur sélectionne les différentes tâches à accomplir en fonction de leur degré de priorité.

Les tâches sont identifiées par la source du signal d'interruption, à laquelle est affectée une priorité fixe ou variable.

A tout moment, l'ensemble des moyens nécessaires à l'exécution de la tâche la plus prioritaire demandée lui est allouée ; la multiprogrammation est réduite de ce fait à l'exécution de la tâche la plus prioritaire demandée, indépendamment de la disponibilité des moyens.

III.1.1.2. L'enchaînement des instructions

Le compteur ordinal assure la gestion des instructions. Grâce à l'utilisation de circuits évolués, il offre de nombreuses possibilités de calculs d'adresses : sauts inconditionnels ou sur test, absolus ou relatifs, à toute adresse de la mémoire de programme ; les tests peuvent être effectués dans n'importe quel opérateur.

III.1.2. Fonctionnement d'A.S.M.O.D.E.E. 02

Toute instruction du programme correspond systématiquement au transfert d'une information, exécuté en un cycle machine, qui peut être suivi ou non par un traitement microprogrammé de cette information ; ce dernier est mené parallèlement avec les transferts suivants lorsqu'il s'agit d'une donnée du système.

On distingue ainsi deux modes de fonctionnement pour chacun des opérateurs :

- le mode liaison qui assure les échanges avec le reste du système,
- le mode autonome, chargé du traitement des données.

La figure III.4. présente le principe des opérateurs d'A.S.M.O.D.E.E. 02 et fait apparaître la distinction entre les éléments affectés par chacun des deux modes de fonctionnement.

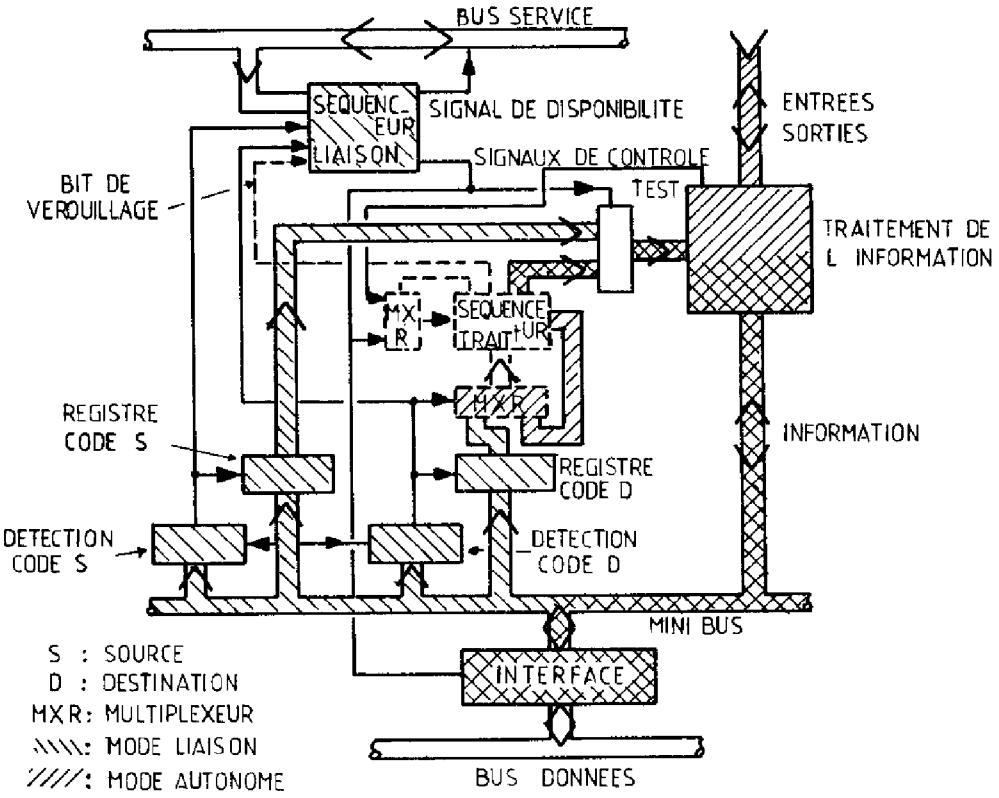


FIGURE III.4.

III.1.2.1. Le mode liaison

L'émission et la réception des informations sont organisées par le séquenceur de liaison de chaque module, qui reçoivent deux sortes de signaux :

- des signaux de synchronisation transmis par le bus de service,
- des signaux provenant du décodage des instructions, dans chaque opérateur.

Les signaux de synchronisation sont créés dans l'opérateur pilote par le séquenceur de la machine qui rythme les transferts de manière synchrone.

Ces signaux contiennent les états de la machine qui indiquent la nature de l'information occupant le bus de communication et deux signaux d'horloge sans recouvrement, qui permettent la validation des transferts.

La possibilité d'un traitement parallèle impose au séquenceur de liaison de chaque opérateur nommé d'émettre un signal indiquant son état de disponibilité.

Si un ou plusieurs des opérateurs appelés par l'instruction en cours ne sont pas disponibles, le séquenceur du système se met dans une boucle d'attente qu'il ne pourra quitter qu'à la réception d'une interruption ou à la libération de tous les opérateurs concernés.

Le cycle machine adopté est indiqué sur la figure III.5.

REMARQUES

On peut changer le cycle machine en modifiant la programmation des mémoires :

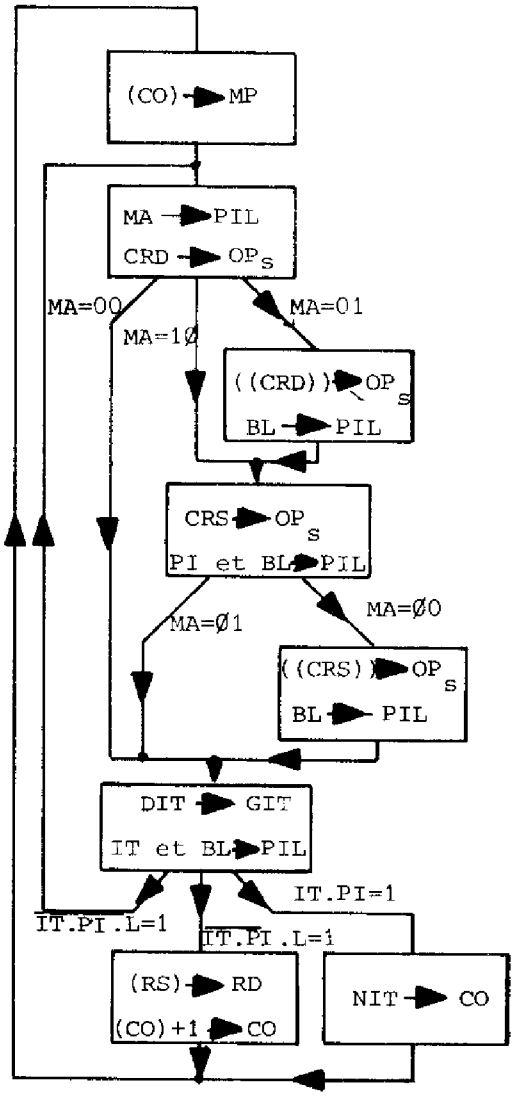
- dans les séquenceurs de liaison de tous les opérateurs,
- dans le séquenceur du pilote.

Par rapport au développement présenté au chapitre II, on note :

- l'introduction de modes d'adressage, qui permettent une programmation plus souple,
- l'absence d'un fonctionnement en mode pipe-line par un recouvrement des actions élémentaires à l'intérieur d'un cycle machine.

III.1.2.2. Le mode autonome

Lorsqu'une instruction indique le traitement d'une donnée, le séquenceur de liaison de l'opérateur destination, après avoir participé à l'échange de la donnée, délègue le contrôle de l'opérateur au séquenceur de traitement ; ce dernier isole alors le module du bus de communication pour ne rendre la main qu'après l'exécution de la totalité du traitement.



BL : Bit libre, transmis au pilote par l'opérateur concerné par l'état précédent du cycle :
- BL=0 : opérateur occupé
- BL=1 : opérateur libre

CO : compteur ordinal

CRD : code registre destination

CRS : code registre source

DIT : demandes d'interruption véhiculées par le bus information

GIT : opérateur de gestion des interruptions

IT : demande d'interruption adressée par le GIT au pilote : IT=0 ou 1 suivant qu'il n'y a pas ou qu'il y a une tâche plus prioritaire que celle en cours d'exécution

L : fonction ET des bits libres des opérateurs concernés par le cycle

MA : bits de mode d'adressage

00 immédiat source-direct destination
01 direct source-indirect destination
10 indirect source-direct destination
11 direct source-direct destination

MP : opérateur mémoire programme

NIT : numéro de l'interruption

OP_S : tous les opérateurs à l'exception du pilote et de la mémoire programme

PIL : opérateur pilote

RD : registre destination

RS : registre source

(X) contenu de X

((X)) contenu du registre adressé par X

Ø=0 ou 1

FIGURE III.5.

REMARQUE

La technique de microprogrammation utilisée pour l'opérateur arithmétique et logique est totalement horizontale : le calcul de l'adresse de la microfonction suivante et la réalisation de la microfonction en cours

sont effectués en parallèle. La mémoire de microfonction utilise 256 mots de 32 bits. La sélection de la microinstruction suivante se fait par incrémentation ou par saut (inconditionnel ou sur test) à une adresse quelconque de la mémoire de microprogramme ; huit des trente-deux bits de chaque mot sont réservés à l'écriture de cette adresse.

III.2. RECHERCHE D'UN PARALLELISME DES TRAITEMENTS PAR UNE MODIFICATION DE L'ECRITURE DES PROGRAMMES

III.2.1. Les problèmes généraux de la recherche d'un parallélisme des traitements

La structure à caractère généralisé que l'on a obtenue diffère des autres machines multiopérateur , le plus souvent spécialisées, car elle fonctionne en mode pseudo-pipe-line seulement lorsqu'il n'y a pas de conflit d'accès, alors que l'on n'a fait aucune hypothèse sur :

- la durée des transferts élémentaires,
- le nombre et les temps d'exécution des opérateurs.

La simplicité de fonctionnement que nous avons voulu attribuer à A.S.M.O.D.E.E. 02 s'oppose à une recherche d'un parallélisme des traitements pendant le fonctionnement du système, à cause de la complexité du matériel et des programmes d'exploitation que cela entraînerait. Nous nous proposons, pour améliorer les performances de la structure, de favoriser l'introduction du parallélisme, en reportant sa recherche au niveau de la conception du programme.

Il convient de noter que la nature essentiellement séquentielle d'un programme est plus le fait des possibilités qu'a l'homme de s'exprimer ou même de raisonner sous la forme d'un enchaînement unique d'idées -ce phénomène est renforcé par la nature monoprocesseur de la majorité des systèmes- que le fait des relations de dépendances amenées par l'application qui laissent quelque latitude à l'ordre dans lequel doivent être exécutées les instructions.

Lorsque l'on envisage une recherche de la forme optimale d'écriture d'un programme, pour une machine donnée, on se heurte à trois problèmes importants :

- la démarche nécessite un gros travail du programmeur qui doit transcrire toutes les dépendances temporelles qui existent entre les actions élémentaires,
- l'introduction du parallélisme laissée à l'intuition du programmeur nécessite de sa part une parfaite connaissance du fonctionnement de la machine et des temps d'exécution de chaque opération ; cela va à l'encontre du désir de plus en plus vif que la machine soit transparente à l'utilisateur,
- les travaux effectués pour écrire des programmes à structure maillée n'ont pas conduit à de grands résultats.

Pour ces raisons, nous avons limité notre action à une amélioration de l'ordre dans lequel sont écrites les instructions du programme. Cette action est effectuée de manière automatique par un programme non résident à partir d'un programme qui a été écrit d'une manière conventionnelle.

Cette amélioration est faite dans le sens d'une minimisation du temps global d'exécution d'un programme.

Un tel objectif cadre avec le fait que sur le plan du logiciel comme du matériel, on recherche une solution qui soit très adaptée à chaque application envisagée ; comme nous avons indiqué que nous faisions l'hypothèse d'un programme d'application court et stable (peu susceptible de grand changement après sa mise au point), on peut envisager de passer un temps assez long à une amélioration de la forme du programme.

Cette recherche doit être faite de manière automatique, car elle incombe à un automaticien qui, pour des raisons de confiance dans le logiciel ou de pure simplicité, doit consacrer toute son attention à l'analyse de la mission et aux problèmes fonctionnels de l'écriture des programmes de l'application.

La méthode utilisée,qui réalise une amélioration locale de la forme d'un programme,est présentée dans le paragraphe III.2.2.

III.2.2. Amélioration de la forme du programme

Le point essentiel à considérer dans le développement de cette méthode est amené par le découpage en modules fonctionnels du système de commande : en effet, celui-ci a pour conséquence que tous les problèmes de dépendances temporelles s'expriment par des conflits d'accès potentiels aux modules fonctionnels, conflits qui apparaissent de manière explicite dans l'énoncé des programmes.

En fait, l'analyse des deux champs des instructions -code de l'opérateur source et code de l'opérateur destination- met en évidence les deux origines des dépendances temporelles ; il s'agit :

- du chemin des données (dépendances fonctionnelles qui seules nous intéressent),
- du chemin des fonctions (dépendances matérielles qui traduisent l'occupation des moyens dans A.S.M.O.D.E.E. et créent des contraintes supplémentaires que l'on voudrait ne pas prendre en compte).

En raison de la difficulté -voire l'impossibilité- qu'il y a à discerner les dépendances d'origine matérielle, nous les exploiterons comme les dépendances fonctionnelles, malgré la réduction des possibilités de parallélisme qu'elles entraînent.

L'existence de deux catégories d'instruction limite également les possibilités de parallélisme dans l'exécution d'un programme ; leur différence de nature se traduit, en effet, par deux modes de fonctionnement distincts :

- on n'interrompt pas la scrutation du flot d'instructions dans le cas d'instructions de traitement des données,
- on attend la fin du calcul d'adresse pour effectuer un saut inconditionnel ou sur test dans le cas d'une instruction de gestion des adresses.

Nous nous préoccuperons dans une première phase des segments de programmes constitués des instructions de traitement de données qui sont comprises entre deux instructions de gestion des adresses ; nous indiquerons ensuite au paragraphe III.2.3. les possibilités d'extension des résultats à l'ensemble d'un programme.

Nous rappelons les principaux traits de fonctionnement d'A.S.M.O.D.E.E. 02 que nous aurons à prendre en compte dans cette étude :

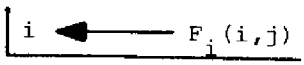
- A.S.M.O.D.E.E. 02 est constitué de blocs fonctionnels reliés entre-eux par un bus unique,
- chaque bloc est caractérisé par un accumulateur unique et un ensemble de fonctions qu'il est capable d'effectuer,
- l'initialisation d'une instruction dure un temps (τ) constant,
- le traitement de la donnée s'effectue par la suite en un temps ($n\tau$),
- une instruction est initialisée à chaque temps τ, tant qu'il n'y a pas de conflit.

III.2.2.1. Enoncé du problème

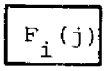
Le formalisme utilisé pour énoncer le problème des dépendances entre les instructions est présenté dans [KEL.75] ; soient :

- i l'accumulateur de l'opérateur destination,
- j l'accumulateur de l'opérateur source,
- F_i la fonction à réaliser sur le contenu de (j) et le contenu de (i).

On représente les instructions d'A.S.M.O.D.E.E. sous la forme :



ou, avec une notation simplifiée :



Enoncé du problème

On recherche une exécution en parallèle des segments de programme qui soit équivalente à l'exécution séquentielle du programme initial.

L'équivalence est définie comme suit :
les séquences des contenus des registres sont les mêmes que dans l'exécution séquentielle initiale.

Définition des conflits

Soit NUM le numéro d'une instruction dans le programme initial ; on définit la source d'information $S[NUM] = j$ et on définit la destination de l'information $D[NUM] = i$ pour l'instruction NUM : $F_i(j)$

Si (NUM 1) et (NUM 2) sont les numéros de deux instructions quelconques du programme, il y a conflit entre (NUM 1) et (NUM 2) si et seulement si l'une ou plusieurs des trois relations suivantes sont vérifiées :

$D(NUM\ 1) \cap S(NUM\ 2) \neq \emptyset$

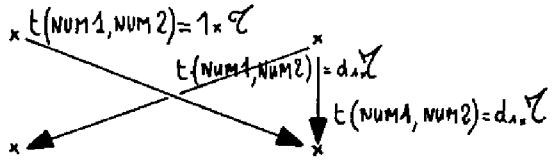
$D(NUM\ 2) \cap S(NUM\ 1) \neq \emptyset$

$D(NUM\ 1) \cap D(NUM\ 2) \neq \emptyset$

$\emptyset$ étant l'ensemble vide.

Dans les autres cas, (NUM 1) et (NUM 2) ne sont pas en conflit et ceci est une condition nécessaire et suffisante pour que (NUM 1) et (NUM 2) puissent s'exécuter simultanément (au sens défini ci-dessus).

Soit t(NUM 1, NUM 2) le temps minimum séparant les instructions (NUM 1) et (NUM 2) (avec NUM 1 < NUM 2) ; lorsqu'il y a conflit, on a :

NUMERO DE L'INSTRUCTION	OPERATEUR SOURCE OPERATEUR DESTINATION
NUM 1 NUM 2	

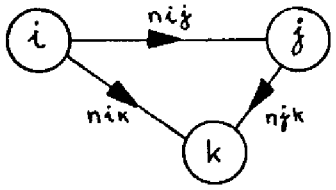
$d_1 \times \tau$ étant le temps nécessaire à l'exécution du traitement de l'instruction NUM 1.

III.2.2.2. Méthode utilisée

La réorganisation du programme constitue un problème d'ordonnancement [BEL.74] qui passe par les étapes suivantes :

1. Détermination du graphe de précédence, écrit à partir de la matrice obtenue en indiquant la valeur maximale du temps minimum séparant toute instruction des instructions qui suivent dans le programme initial, avec lesquelles il y a conflit.

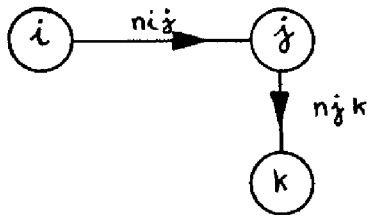
2. Réduction du graphe par transitivité qui conduit au graphe d'ordonnancement, sur lequel on rajoute les deux sommets "début" et "fin".
La règle de réduction est donnée sur l'exemple suivant :



La transitivité permet de supprimer l'arc n_{ik} si et seulement si la relation suivante est vérifiée :

$$n_{ij} + n_{jk} \geqslant n_{ik}$$

on a alors un schéma équivalent :



3. Introduction des paires disjonctives entre tous les noeuds du graphe qui ne sont pas reliés directement ou indirectement par les relations d'ordre du graphe d'ordonnancement.

Ces paires disjonctives matérialisent la contrainte de l'unicité du flot d'instruction : une seule instruction peut être initialisée à chaque unité de temps. Le temps affecté à ces paires disjonctives est donc égal à $(1 \times \tau)$. On obtient le graphe disjonctif.

4. Arbitrage (sélection d'une des deux flèches d'une paire disjonctive) du graphe disjonctif par l'application d'un algorithme SEP (séparation et évaluation progressive) de l'exploration implicite au cas de l'ordonnancement avec limitation de moyens.

On obtient le graphe conjonctif qui donne immédiatement la séquence optimale des instructions pour une structure donnée.

Présentation de l'algorithme utilisé dans la phase d'arbitrage

On définit pour tout sommet (i) du graphe disjonctif :

v_i = date de début au plus tôt de l'instruction numéro i,

v'_i = date de début au plus tard de l'instruction numéro i.

Le chemin critique est caractérisé par $v_i = v'_i \quad \forall i$

Soit d_{ij} l'intervalle de temps minimum entre le début de la tâche i et le début de la tâche j.

Les paires disjonctives reliant les sommets du graphe disjonctif qui ne sont pas déjà reliés par relation de transitivité, on remplace une paire disjonctive par un arc d_{ij}, lorsque l'on a arbitré un conflit ; (v_j) prend alors une valeur v_j^*) telle que :

- si $v_j^* > v'_j$, la longueur (L) du chemin critique augmente au moins de $v_j^* - v'_j = \boxed{v_i + d_{ij} - v'_j = \Delta_{ij}}$
- si $v_j^* \leqslant v'_j$, (L) n'augmente pas.

On a donc les relations suivantes pour l'augmentation ΔL de la longueur du chemin critique :

- si $\Delta_{ij} > 0 \qquad \Delta L \geqslant \Delta_{ij}$
- si $\Delta_{ij} \leqslant 0 \qquad \Delta L = 0$

Choix de l'arc (rs) à introduire :

1. On calcule les valeurs de (Δ_{ij}) pour tous les arcs disjonctifs.
2. On choisit la paire (rs) comme suit :

$$|\Delta rs - \Delta sr| = \max_{i,j} |\Delta_{ij} - \Delta_{ji}|$$

i et j étant les sommets d'une paire disjonctive.

3. On détermine l'arc (rs) comme suit :

$r \rightarrow s$ tel que $\Delta rs \leqslant \Delta sr$

Après chaque sélection, on recherche le nouveau chemin critique et on réajuste les dates au plus tôt et au plus tard où cela s'avère nécessaire ; on supprime de la liste des conflits qui restent à arbitrer tous ceux qui deviennent arbitrés par relation de transitivité ; on s'intéresse enfin aux paires disjonctives restantes.

REMARQUE

Dans le cas particulier où ($d_{ij}=d_{ji}$ $\forall$ i,j), la procédure adoptée revient à comparer la position dans le temps des milieux des segments {date au plus tôt - date au plus tard} pour les deux sommets d'une paire, et à retenir ceux qui sont les plus éloignés, arbitrés dans le sens du milieu le plus tôt vers le milieu le plus tard.

Lorsque l'on a arbitré tout le graphe, on obtient la date au plus tôt de la dernière instruction à exécuter,qui correspond à la longueur du chemin critique. Il faut ensuite remonter dans l'arbre de division tracé et étudier tous les chemins possibles, que l'on abandonne dès que l'on trouve une valeur de date au plus tôt supérieure à la longueur du plus court chemin critique trouvé. Le temps minimum nécessaire pour l'exécution du programme est obtenu à partir de la longueur du dernier chemin critique retenu -lorsque l'on a parcouru tout l'arbre de décision- en ajoutant le temps de traitement de la dernière instruction.

Comme l'existence de dépendances matérielles crée des contraintes plus fortes que celles qui sont strictement nécessaires, nous n'avons pas réalisé la dernière phase de cette méthode qui consiste à regarder

tous les chemins possibles pour arriver à un optimum. Nous nous sommes restreint à une amélioration de la forme du programme obtenue en limitant la procédure décrite à l'arbitrage de toutes les paires disjonctives, ne remettant en cause les choix effectués que lorsque la longueur du dernier chemin critique retenu est dépassée.

III.2.2.3. Relations de dépendance moins contraignantes

Nous sommes amené, pour obtenir une représentation plus juste des contraintes et donner un intérêt plus grand à la méthode développée (i.e. obtenir un gain de temps plus important), à distinguer deux types de fonctions exécutées dans les opérateurs "destination" :

- les fonctions qui utilisent le contenu de l'accumulateur "destination" comme opérande :

  (a)+(b) ⟶ a : instructions du type $F_{i1}(j)$

- les fonctions qui initialisent le contenu de l'accumulateur à une nouvelle valeur :

  Remise à zéro, chargement de (a) : instructions du type $F_{io}(j)$.

Les dépendances deviennent alors moins contraignantes.

On recherche les segments de programmes destinés par deux apparitions successives d'une instruction du type $F_{io}(j)$, pour un même accumulateur (i) et la nouvelle règle d'équivalence est plus souple :

La séquence des contenus de l'accumulateur (i) ne doit pas être modifiée à l'intérieur du segment de programme défini, par la réécriture du programme.

Les modifications à apporter à la procédure présentée dans le paragraphe III.2.2.2. concernent les phases 1 et 3.

Dans l'établissement de la matrice P et du graphe de précédence, on interrompt la scrutation des instructions suivantes dès que l'on rencontre un conflit avec une instruction du type $F_{io}(j)$. On effectue un changement de notation, appelant i'_1, i'_2,... l'opérateur i après la première, la deuxième ... apparition d'une instruction $F_{io}(j)$.

Avant l'introduction des paires disjonctives, on explicite le fait qu'une séquence des contenus des registres doit être le même pour tout registre i'_k entre la kième et la (K+1)ième apparition de $F_{io}(j)$.

Ceci est effectué en reliant chaque instruction du segment i'_k où apparaît le registre i'_k à l'ensemble des débuts des autres segments i'_ℓ c'est-à-dire à l'ensemble des instructions $F_{io}(j)$. Cette liaison est affectée du temps nécessaire à l'exécution de l'instruction dont part l'arc considéré. On utilise ainsi des "segments disjonctifs" qui s'ajoutent aux paires disjonctives existantes, et qui sont arbitrés selon les mêmes règles.

III.2.2.4. Exemple d'application

(Calcul de l'expression $(a+b)^2+\sqrt{a}+\sqrt{b/a}$)

Considérons un programme dont la forme initiale est donnée sur la figure III.6.

NUMERO DE L'INSTRUCTION	ECRITURE DE L'INSTRUCTION		NOM DE L'ACCUMULATEUR "DESTINATION"	CONTENU DE CET ACCUMULATEUR EN FIN D'INSTRUCTION
	Source	Destination		
0	4	1	Σ	a
1	5	1	Σ	a + b
2	1	2	Π	a + b
3	2	2	Π	$(a + b)^2$
4	2	1	Σ	$(a + b)^2$
5	4	3	$\sqrt{}$	$\sqrt{a}$
6	3	1	Σ	$(a + b)^2 + \sqrt{a}$
7	5	2	Π	b
8	4	2	Π	b/a
9	2	3	$\sqrt{}$	$\sqrt{b/a}$
10	3	1	Σ	$(a+b)^2+\sqrt{a}+\sqrt{b/a}$

FIGURE III.6.

La figure III.7. permet d'apprécier le gain apporté par les deux procédures présentées aux paragraphes III.2.2.2. et III.2.2.3. par rapport à cette forme initiale.

III.2.3. Introduction des sauts

Nous admettons dans ce paragraphe qu'il n'existe pas de contrainte sur l'utilisation de la mémoire, due par exemple à la réservation de certaines zones de la mémoire.

On peut obtenir une amélioration de la forme de tout un programme en appliquant la méthode présentée dans le paragraphe précédent à chacun des blocs de ce programme, délimité par deux instructions portant sur la gestion des adresses.

Les temps d'exécution des instructions portant sur les adresses sont faibles et par suite, les opérateurs actifs n'ont pas tous le temps de se libérer pendant le transfert et le traitement propres à cette instruction. Si l'on veut tenir compte de ce fait, il faut considérer le problème d'une manière globale, car le découpage en blocs utilisé dans la première méthode peut amener une erreur importante due à la fréquence des instructions de gestion d'adresse (20 à 35 % des instructions d'un programme).

Deux politiques peuvent être envisagées :

- remanier le programme dans son ensemble,
- conserver le découpage du programme en blocs, réalisé par l'emplacement des instructions portant sur les adresses.

Compte tenu des deux observations suivantes, nous ne retiendrons que la seconde politique :

1. Les méthodes permettant de minimiser le temps moyen d'exécution d'un programme en fonction de la probabilité de passer dans chacune des branches d'un algorithme sont complexes, surtout si l'on envisage la possibilité de déplacer ou même de répéter certaines instructions en dehors de la branche à laquelle elles appartiennent.
2. Dans des applications temps réel, seul le temps maximum est contraignant et il ne faut pas minimiser le temps moyen d'exécution d'un algorithme, mais le temps d'exécution de la branche la plus longue.

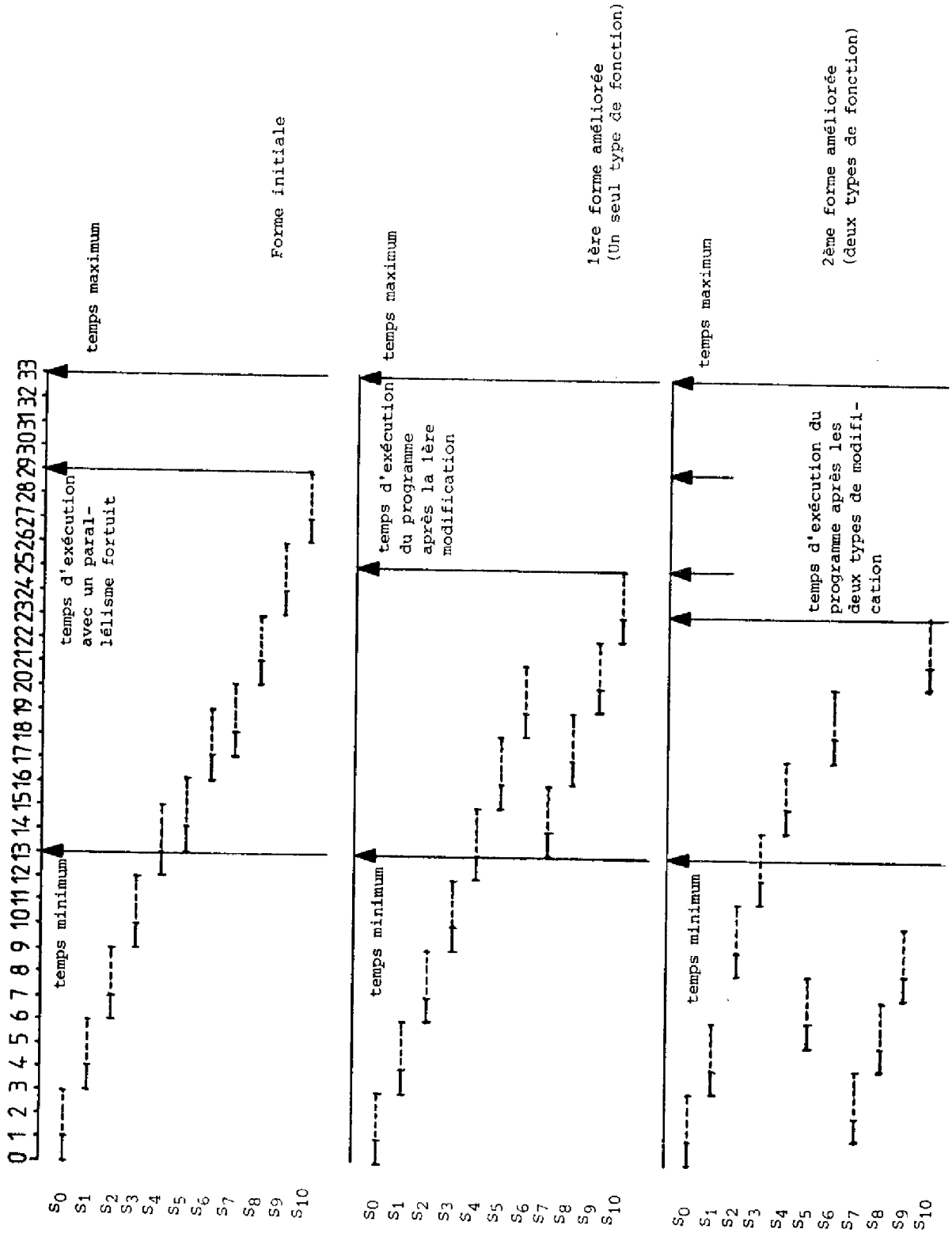

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
Forme initiale
temps maximum
temps d'exécution avec un paral-lélisme fortuit
temps minimum
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
1ère forme améliorée
(Un seul type de fonction)
temps maximum
temps d'exécution du programme après la 1ère modification
temps minimum
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
2ème forme améliorée
(deux types de fonction)
temps maximum
temps d'exécution du programme après les deux types de modifi-cation
temps minimum
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10

FIGURE III.7.

L'optique consiste alors à rechercher les branches les plus longues en cas de sauts sur test, écrire les instructions de saut dans les branches les plus courtes à chaque fois que cela est possible, puis minimiser le temps d'exécution du chemin le plus long du programme. Enfin, il faut minimiser le temps d'exécution de chacune des branches que l'on a laissées de côté.

Dans cette minimisation, aucune instruction d'un bloc n'est déplacée en dehors de ce bloc. Deux types de liaison viennent s'ajouter, par rapport à l'étude de chaque bloc pris indépendamment :

- les liaisons dues à l'introduction de l'instruction d'adressage, qui durent un temps multiple ou non de τ ,
- les liaisons entre deux blocs consécutifs, dues à l'utilisation des mêmes registres.

2ÈME PARTIE

EVALUATION DES PERFORMANCES D'UN SYSTÈME DE COMMANDE

-

CHAPITRE IV

MODELISATION DU COMPORTEMENT EN VUE DE L'EVALUATION DES PERFORMANCES

-

IV.1. GENERALITES SUR LA MODELISATION

IV.1.1. Objet de la modélisation

IV.1.1.1. Outil d'aide à la conception

Lors de la conception d'un système, on est fréquemment amené à comparer les différentes solutions envisageables, et il est nécessaire, pour effectuer un choix, d'être en mesure de porter un jugement sur chacune des possibilités en présence.

Lorsque l'on s'intéresse à l'évolution du système dans le temps, on peut avoir à mesurer différentes grandeurs telles que :

- le débit, qui correspond au nombre de fonctions élémentaires exécutées par unité de temps,
- le rendement, qui est une caractéristique individuelle de chaque élément de la structure et qui regroupe différentes composantes : taux d'oisiveté, taux de dépassement dû à la gestion des ressources, taux d'activité au bénéfice des processus,
- les différents temps caractéristiques : temps d'exécution (des tâches, des instructions), temps de réponse (prise en compte de la demande d'activation d'une tâche)...

La comparaison entre les performances de deux systèmes n'étant intéressante que dans le cadre de l'application envisagée, il est important de posséder un modèle qui décrive l'évolution du système dans son environnement et dans lequel interviennent des paramètres caractéristiques :

- du fonctionnement du système de commande, qui est connu dans une situation donnée,
- du comportement de l'environnement, qui comprend à la fois des composantes déterministes et des composantes aléatoires.

Cinq paramètres particularisent le fonctionnement dans le temps d'un système formé d'ensembles interactifs ; ces paramètres sont présentés sur la figure IV.1. selon un schéma donné dans [BUZ.71].

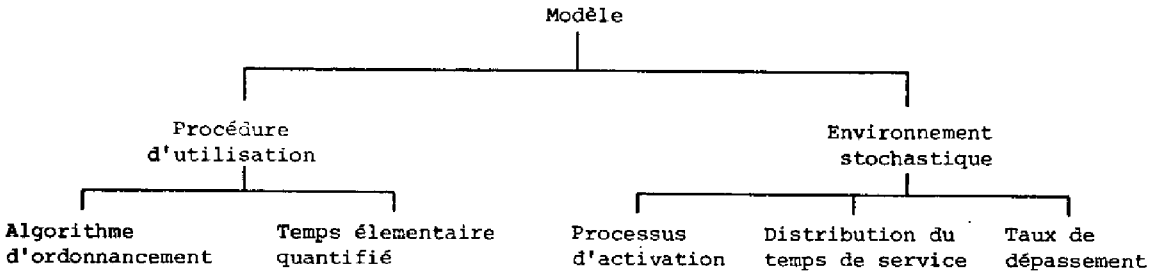


FIGURE IV.1.

En reportant ces paramètres aux entités de l'application que nous avons définies au chapitre I, on est conduit au schéma de la figure IV.2.

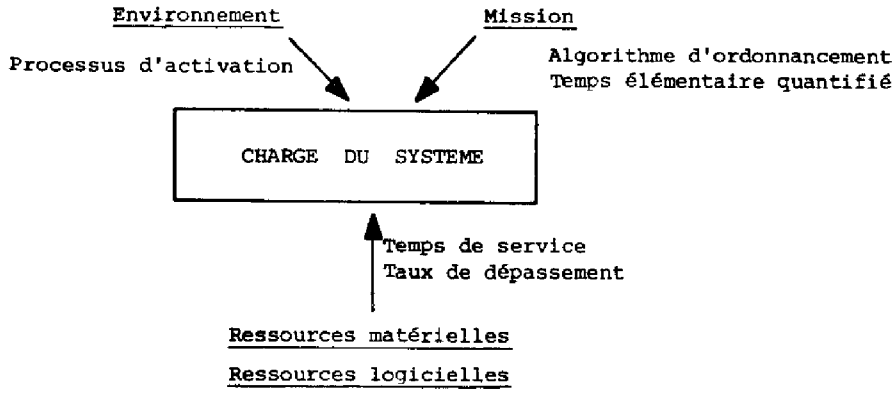


FIGURE IV.2.

Un modèle de description du comportement d'un système doit constituer un outil d'aide à la conception qui répond à deux objectifs :

- prévoir les performances du système et déterminer ainsi ses chances de succès pour la mission,
- permettre une étude de sensibilité aux paramètres.

C'est essentiellement dans cette seconde optique que nous avons développé un modèle de représentation d'A.S.M.O.D.E.E. 02, afin de pouvoir chiffrer le gain des performances amené par certaines modifications.

Dans le cas particulier de ce type de structures où l'architecture et le mode de fonctionnement sont définis à priori pour toute une

classe d'applications, les choix qui restent à effectuer, à chaque nouvelle application, portent sur :

- la nature et le nombre des fonctions élémentaires à développer,
- les temps alloués à l'exécution des fonctions élémentaires,
- le temps attribué au transfert d'une information entre deux modules.

IV.1.1.2. Régime permanent

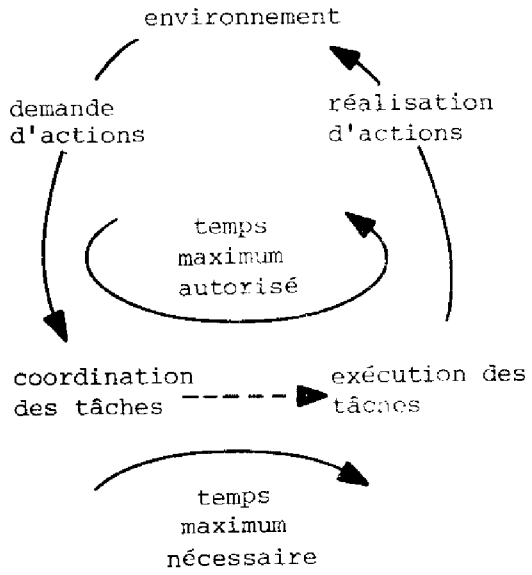
Dans cette optique d'aide à la conception, on désire établir facilement les performances du système proposé, sans avoir à effectuer un long dépouillement de résultats statistiques provenant d'une simulation. Pour permettre une exploitation rapide du modèle, on est amené à émettre des hypothèses simplificatrices au niveau de la description du fonctionnement du système et du comportement de l'environnement.

On obtient ainsi un modèle de représentation simple mais approché, dont le but est de fournir une estimation des performances que l'on peut attendre. A ce titre, on se limitera à la description d'une phase de la mission qui se situe assez loin de la mise en route du système ; on peut admettre que, dans ces conditions, l'environnement n'évolue plus de manière importante et qu'il y a une constance des actions que le système doit effectuer (régulation autour du point nominal du fonctionnement par exemple). On se préoccupe donc dans cette étude du fonctionnement "en régime permanent" du système, c'est-à-dire, de son comportement dans un environnement stabilisé.

IV.1.2. Recherche d'un modèle de description

L'optique dans laquelle doit être construit le modèle est de répondre à la question simple suivante : le système de commande est-il apte à exécuter la mission dans le temps imparti ?

On peut exprimer ceci en faisant apparaître les contraintes temporelles à respecter sur le schéma général de description de l'application représentée sur la figure IV.3.



Description au
niveau de la mission

FIGURE IV.3.

La boucle formée par les liaisons entre les éléments, se refermant sur l'environnement, est caractéristique du temps réel et responsable de l'une des principales difficultés de la modélisation : celle du manque d'homogénéité des grandeurs à prendre en compte. En effet, les actions mises en jeu dans cette boucle se traduisent par l'existence de plusieurs "pôles" dans le spectre des fréquences ou des durées, qui caractérisent différents niveaux d'abstraction dans la représentation du système, et qui particularisent également la classe de l'application :

- le rebouclage au niveau de l'environnement crée le déterminisme des systèmes de commande automatique,
- on peut considérer une boucle ouverte pour des systèmes informatiques.

Une description du système au niveau de la mission, qu'on appellera description du comportement de la charge du système, ne permet pas de prendre en compte de manière fine les particularités du fonctionnement du système de commande. Si l'on veut connaître avec assez de précision le temps d'exécution des tâches, ce qui est un des tests recherchés lors de la conception d'un système, des hypothèses générales, comme un temps de service exponentiel, ne sont pas satisfaisantes, et il est nécessaire de développer la relation qui existe entre la coordination et l'exécution des tâches, en faisant intervenir les niveaux de définition inférieurs, du système de commande.

Le diagramme de la figure IV.3. est ainsi remplacé par celui de la figure IV.4.

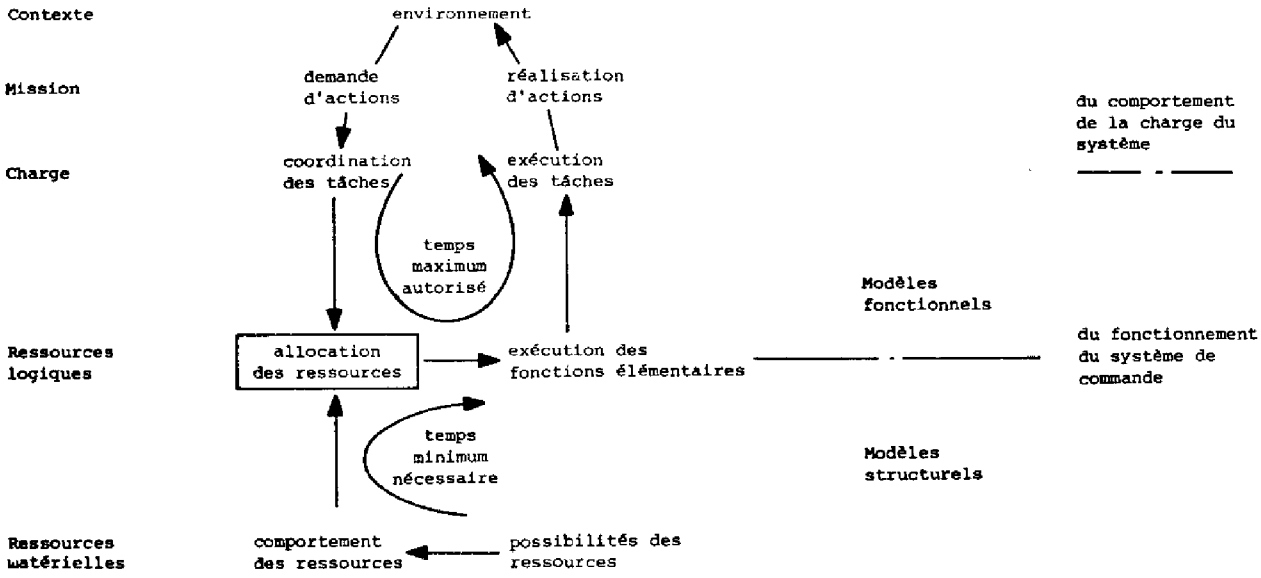


FIGURE IV.4.

Dans la suite, nous appellerons :

- "modèles fonctionnels" les modèles qui représentent essentiellement la charge du système,
- "modèles structurels" les modèles qui décrivent l'évolution du matériel.

Ces deux types de modèle correspondent à deux niveaux de description distincts :

- les modèles fonctionnels constituent une description du comportement du système par l'expression de l'évolution de la charge (succession des instructions),
- les modèles structurels décrivent le comportement du système par l'introduction des données fonctionnelles du premier type de modèle dans la représentation de l'évolution du matériel.

Des travaux sont actuellement en cours à l'Université du Michigan [MEY.76] pour obtenir une description formelle de la mission, qui soit compatible avec les modèles de représentation des systèmes de commande. Cette étude, qui passe par la définition de niveaux intermédiaires entre

la mission et le système de commande, n'est pas concernée par la modélisation du comportement des systèmes informatiques mais par l'établissement des relations qui lient les descriptions propres à chacun des niveaux définis.

Nous retiendrons la philosophie de cette méthode qui consiste à procéder à une analyse descendante des différents niveaux de description de l'application.

IV.1.3. Le problème des dépendances

Il existe de fortes relations de dépendance entre les différentes entités d'une application (mission et système). Les dépendances qui peuvent être d'origine soit matérielle, soit fonctionnelle ou logique, se distinguent en fonction du niveau où elles apparaissent.

IV.1.3.1. Niveau de la charge du système

Les dépendances fonctionnelles sont de deux types :

- dépendances directes entre les tâches, car elles apparaissent explicitement dans l'énoncé des tâches, donc, dans l'écriture du programme,
- dépendances indirectes lorsque les tâches, qui se présentent sous la forme de programmes indépendants, peuvent exercer une influence sur les autres tâches par une modification qu'elles entraînent sur le comportement de l'environnement (demande d'activation d'une tâche par exemple).

Les dépendances matérielles sont dues au contexte de multiprogrammation qui attribue un moyen unique (le système de commande) aux différentes tâches.

IV.1.3.2. Niveau du système de commande

Les dépendances temporelles qui existent au niveau des fonctions logiques se traduisent par la nature séquentielle de l'exécution des fonctions élémentaires qui apparaît dans la forme du programme.

La nature séquentielle des fonctions élémentaires peut être due :

- à une dépendance du type fonctionnel, c'est-à-dire à la logique du programme (cas d'une action qui nécessite le résultat d'une opération exécutée auparavant),
- à une dépendance de type matériel due au partage, pour des raisons économiques évidentes,d'une ressource commune.

IV.1.3.3. Dépendances entre les deux niveaux

La présence d'interactions entre les deux niveaux de description du système est due à la boucle reliant les éléments du système de commande aux éléments de la mission. On peut les résumer de la manière suivante :

- le système de commande intervient directement dans les temps d'exécution des tâches, puisqu'il les réalise,
- l'imbrication des exécutions des tâches privilégie par moment certaines fonctions élémentaires de la machine : lors d'un changement de tâches, il y a par exemple une forte activité de communication avec la mémoire, due à la sauvegarde ou à la constitution d'un contexte.

La durée de cette activité et l'intervalle entre deux de ses apparitions constituent un exemple de deux pôles distincts dans le spectre des temps de fonctionnement du système ; nous avons indiqué que cette dispersion est l'une des principales difficultés de l'établissement d'un modèle.

IV.2. DEMARCHE SUIVIE POUR SIMPLIFIER LA MODELISATION

IV.2.1. Deux étapes pour l'évaluation des performances d'un système

IV.2.1.1. Suppression de l'influence du niveau mission sur le niveau système

Pour s'affranchir du problème soulevé par la non homogénéité des grandeurs à inclure dans un même modèle de représentation, nous faisons l'hypothèse simplificatrice suivante :

on admet que la constitution des tâches est figée et ne subit pas de modifications locales amenées par l'allocation dynamique des tâches propre au contexte de multiprogrammation.

Cette hypothèse permet d'évaluer les performances d'un système en deux étapes distinctes se rapportant :

- au niveau du système de commande,
- au niveau de la mission.

L'étape relative au niveau du système de commande consiste en une description du fonctionnement du système de commande, qui permet d'évaluer les temps nécessaires à l'exécution des fonctions élémentaires, dans le cadre d'hypothèses qui caractérisent chaque tâche de manière figée.

L'étape relative au niveau de la mission est un développement d'un modèle de comportement de la charge du système, certaines composantes de ce modèle étant les paramètres évalués dans la première étape. Le modèle permet alors de déterminer les performances globales du système, dans le cadre de l'application envisagée, et d'évaluer les chances de succès de la mission.

IV.2.1.2. Application à A.S.M.O.D.E.E. 02

La multiprogrammation est extrêmement simple dans le cas de la structure A.S.M.O.D.E.E. 02 et se traduit par une sauvegarde du contenu de tous les registres nécessaires à une tâche, lorsque celle-ci est activée, puis une restitution de ces informations qui constitue les dernières instructions de cette tâche.

Ces actions de sauvegarde et de la restitution -successions de transferts avec la mémoire- durent un temps constant caractéristique de la tâche considérée et se produisent une fois à chaque exécution de celle-ci.

On divise ainsi facilement la tâche en deux parties qui correspondent :

- aux instructions propres aux actions de l'application,
- aux instructions "système" (sauvegarde et restitution de contexte).

La caractérisation de la tâche que l'on fera au paragraphe IV.2.2. ne concerne que la partie propre à l'application, qui est la seule à intervenir dans la première étape (description du fonctionnement au niveau du système de commande).

Les instructions "système" ne sont introduites que dans la seconde étape sous la forme d'un taux de dépassement propre à chaque tâche, qui est ajouté au temps moyen d'exécution de la tâche -résultat de la première étape- lors de l'étude du comportement de la mission.

Les deux niveaux de description étant dans le cas d'A.S.M.O.D.E.E. 02 :

- la gestion des programmes d'interruption (coordination des tâches au niveau de la mission),
- la gestion des instructions (allocation des ressources logicielles au niveau du système de commande),

on admet donc que les demandes de changement de tâches sont sans effet sur les probabilités de référence aux instructions du programme correspondant.

L'essentiel de notre travail porte sur la première étape de cette démarche, c'est-à-dire sur la description du fonctionnement du système de commande. La grandeur que nous cherchons à évaluer est le débit moyen d'exécution des instructions, ou, ce qui revient au même, le temps moyen d'exécution de chaque instruction dans le cadre d'une tâche d'un type donné.

La figure IV.5.a. fait apparaître de manière nette le rôle primordial joué par l'allocation des ressources qui se trouve au point de conflit entre les contraintes de la mission et les possibilités du système : pour cette raison, la recherche d'un modèle de représentation du fonctionnement du système de commande s'attache à décrire cette question des ressources.

La figure IV.5.b. traduit ces mêmes contraintes dans le cas particulier d'A.S.M.O.D.E.E. 02.

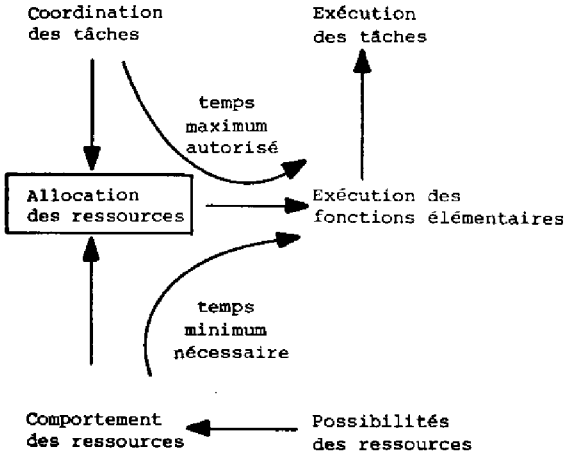


FIGURE IV.5.a.

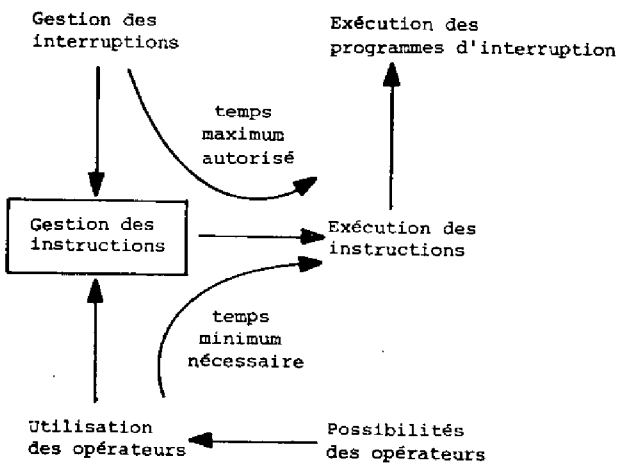


FIGURE IV.5.b.

IV.2.1.3. Représentativité du débit d'exécution des instructions

Le débit d'exécution des instructions est la grandeur que l'on cherche à évaluer dès que l'on s'intéresse à l'architecture d'un système de commande ou de calcul, au niveau des blocs de traitement et de mémorisation, parce qu'elle est significative de la puissance absolue du système.

D'une manière générale, les études portant sur l'évaluation des performances à ce niveau de définition du système s'arrêtent à la fin de ce que nous avons appelé "la première étape" et n'insistent pas sur le fait que la valeur essentielle d'un système est liée plus à sa capacité de traiter l'ensemble des instructions propres aux programmes d'application, qu'au temps moyen d'exécution d'une instruction qui peut appartenir aussi bien aux programmes d'application qu'au programme moniteur de gestion des ressources [LAL.75].

IV.2.2. Caractérisation des tâches

IV.2.2.1. Le nombre moyen de références à chaque instruction

Le nombre de références à chaque instruction du répertoire d'un système, pendant le déroulement d'un programme, intervient directement dans le temps d'exécution de celui-ci.

L'existence de différents types de distribution, qui ont été mis en évidence sur de gros systèmes informatiques par des statistiques effectuées à la suite de nombreuses observations de leur comportement réel, fait apparaître l'influence de la nature de la tâche sur la quantité de références à chaque fonction : problèmes spécifiques, compilation ou même des programmes d'application de nature différente [AGR.75]-[LEN.75]-[WIL.73]. A notre connaissance, de telles études sont rares dans le cas de petits systèmes.

IV.2.2.2. L'ordre d'apparition des instructions

Aux difficultés qu'il y a de caractériser une tâche par des nombres moyens de référence à chaque instruction s'ajoute le problème, propre aux structures permettant un parallélisme des traitements, de la connaissance de l'ordre dans lequel se suivent les instructions [BAT.76]-[SPI.76] : nous avons, en effet, montré au paragraphe III.2., l'importance que revêt la forme du programme sur son temps d'exécution dans le cas d'une machine pouvant fonctionner en mode pseudo-pipe-line.

L'analyse des conflits d'accès à des moyens communs est une problème trop spécifique pour que des résultats sur de gros systèmes (conflit d'accès à la mémoire) puissent être utilisés dans cette étude.

En fonction des deux difficultés que nous venons d'énoncer qui sont propres à toute évaluation des performances mais qui se posent avec plus d'acuité dans le cas particulier que nous présentons, nous avons préféré présenter un modèle "qualitatif" du fonctionnement d'une telle structure, en insistant beaucoup plus sur les méthodes possibles de prise en compte de caractéristiques de la mission ou du système que sur l'exploitation du modèle dans des cas précis. Nous procéderons pour cela à une analyse des extensions possibles et des limitations de modèles simples.

IV.2.2.3. Caractérisation des tâches : modèle de description

L'hypothèse fondamentale que nous formulons, au niveau de la caractérisation des tâches est la suivante :

si on considère la succession des événements constitués par le choix d'une instruction à chaque pas de programme, on admet que ce choix procède d'un tirage indépendant constant, c'est-à-dire que les probabilités de référence à chaque instruction du répertoire restent constantes tout au long du programme.

Cette hypothèse, très fréquemment adoptée pour sa simplicité, donne de bons résultats lorsque les éléments de la structure sont banalisés, mais perd beaucoup de sa valeur lorqu'elle est constituée de modules spécialisés comme c'est le cas dans la structure à opérateurs fonctionnels que nous étudions.

Pour cette raison, nous étudierons au paragraphe VI.3., la possibilité de remplacer cette hypothèse trop restrictive par une description plus réaliste, en introduisant des phénomènes de localité dans le modèle de base.

IV.3. L'OUTIL MATHEMATIQUE

IV.3.1. Choix des chaînes de MARKOV

Les modèles mathématiques les plus fréquemment développés pour la prévision des performances de systèmes informatiques sont les modèles généraux de la théorie des files d'attente et les chaînes de MARKOV.

D'une manière générale, on peut considérer que ces deux outils sont complémentaires car le niveau de description des systèmes que l'on envisage privilégie l'un ou l'autre.

.De nombreuses études ont conduit au développement de modèles généraux de phénomènes d'attente qui représentent le fonctionnement de systèmes informatiques dans lesquels les éléments de base sont constitués par une tâche ou un groupe d'instructions. Ces modèles de description sont facilement exploitables lorsque le comportement des variables du système peut être assimilé à des phénomènes aléatoires comme des processus d'arrivée de POISSON, une distribution du temps de service exponentiel, etc.

.A l'opposé, la majorité des études portant sur une analyse fine du comportement d'un système, au niveau du cycle machine,où intervient un fort déterminisme, utilise les chaînes de MARKOV.

En plus de cette considération générale de niveau de description, trois points particuliers nous ont amené à utiliser une représentation par chaîne de MARKOV :

- un point important qui s'oppose à l'utilisation des modèles conventionnels de la théorie des files d'attente est lié au fait que, dans le cas d'A.S.M.O.D.E.E., les opérateurs ne sont pas systématiquement réutilisés pour une nouvelle action dès que l'action en cours s'achève,
- l'utilisation de chaînes de MARKOV permet une approche "par le bus" qui consiste à prendre en compte l'ensemble des contraintes qui doivent intervenir dans le modèle et à effectuer à postériori des hypothèses simplificatrices pour exploiter le modèle obtenu,en général trop lourd ; on peut considérer que lorsque le plus grand commun diviseur de toutes les grandeurs que l'on fait intervenir dans le modèle est petit devant ces grandeurs, le modèle est inexploitable,
- enfin, l'hypothèse markovienne est respectée puisque le seul aléa dans le comportement du système provient du décodage des instructions, et que l'on a fait l'hypothèse de pas de programmes indépendants.

IV.3.2. Rappels sur les chaînes de MARKOV

Nous nous sommes efforcé d'utiliser les définitions et les notations d'un seul auteur (KEMENY) dans toute la présentation de ce mémoire : les définitions et les résultats dont nous aurons besoin sont rappelés dans ce paragraphe et se trouvent dans [KEM.59]. Ils sont dépouillés de toute notion de temps, ce que nous avons essayé de faire apparaître par le choix d'un vocabulaire général, malgré l'emploi le plus courant de termes provenant de l'observation dans le temps de l'évolution des phénomènes.

IV.3.2.1. Rappel des définitions

- un ensemble ergodique d'états est un ensemble dans lequel tout état peut être atteint à partir d'un état quelconque, et qu'on ne peut plus quitter une fois atteint,

- un ensemble transitoire d'états est un ensemble dans lequel tout état peut être atteint à partir d'un état quelconque, et qui peut être quitté,
- un état ergodique est un élément d'un ensemble ergodique,
- un état transitoire est un élément d'un ensemble transitoire,
- un état absorbant est un état que l'on ne peut plus quitter après y être entré (i.e., c'est un ensemble ergodique constitué d'un état unique).

Nous nous préoccuperons exclusivement des phénomènes que l'on observe de manière discontinue, aux différentes valeurs d'une variable discrète, et qui se trouvent, à chaque observation, dans un nombre fini d'états.

Les observations successives du phénomène sont numérotées de zéro à l'infini à partir d'une étape initiale et constituent une suite d'événements que nous appellerons des tirages par analogie avec des processus indépendants.

.Chaînes de MARKOV

Soient $x_i(n)$ la probabilité que le système soit dans l'état s_i au nième tirage et $p_{ij}(n)$ la probabilité conditionnelle que, étant en s_i au nième tirage, le système se trouve en s_j au (n+1)ième tirage.

Le vecteur $\mathbb{X}(n)$ est le vecteur des probabilités des états

$$\mathbb{X}(n)_{1,r} = \left[x_1(n), \ldots, x_r(n) \right]_{1,r}$$

La matrice $\mathbb{P}(n)$ des probabilités des transitions est la matrice stochastique

$$\mathbb{P}_{(n)_{r,r}} = \begin{bmatrix} & & j & & \\ & & \vdots & & \\ i & \cdots & p_{ij}(n) & \cdots & \\ & & & & r \times r \end{bmatrix}$$

(n+1)ième tirage

nième tirage

Les termes $p_{ij}(n)$ vérifient les relations suivantes :

$$\begin{cases} 0 \leqslant p_{ij}(n) \leqslant 1 & \forall i,j \\ \sum_{j=1}^{j=r} p_{ij}(n) = 1 & \forall i \end{cases}$$

Le fonctionnement du système, décrit par la suite des états qu'il prend au fur et à mesure des observations, se comporte comme une chaîne de MARKOV lorsque son évolution vérifie l'équation d'état suivante :

$$\mathbb{X}_{(n+1)} = \mathbb{X}_{(n)} \mathbb{P}$$

L'état du système au tirage (n+1) ne dépend que de son état au tirage (n) (ceci constitue l'hypothèse markovienne).

L'évolution du système est parfaitement déterminé lorsque l'on connaît les composantes de la matrice $\mathbb{P}$ et le vecteur d'état initial $\mathbb{X}_{(o)}$.

.Chaînes de MARKOV ergodiques

Nous ne considérons que les chaînes de MARKOV ergodiques irréductibles [COR.75] ; celles-ci ont un ensemble ergodique unique. En effet, la présence de plusieurs ensembles ergodiques peut être ramenée à l'étude d'autant de chaînes de MARKOV indépendantes les unes des autres.

Les chaînes de MARKOV ergodiques irréductibles sont des chaînes dans lesquelles il est possible d'aller de tout état à n'importe quel autre état ; les seules que nous utiliserons sont les chaînes de MARKOV régulières que l'on peut définir par la propriété suivante : quel que soit l'état de départ dans une chaîne de MARKOV régulière,on peut se trouver, après un nombre d'étapes suffisant, dans n'importe quel état ; c'est-à-dire que tous les éléments de $\mathbb{P}^n$ sont strictement positifs ($0 < p_{ij} \leqslant 1$) pour une valeur assez grande de n.

.Chaînes de MARKOV ayant des états transitoires

Nous nous servirons essentiellement des chaînes absorbantes caractérisées par le fait que tous les ensembles ergodiques ne possèdent qu'un élément. La probabilité que le système soit dans l'un de ces états absorbants tend vers 1 et, une fois que le système se trouve dans l'un de ces états, il ne peut plus en sortir.

IV.3.2.2. Regroupement d'états

La dimension du modèle constituant la principale limite à l'exploitation des chaînes de MARKOV, nous avons procédé sytématiquement à la recherche d'une réduction du nombre des états.

Il est possible de réduire le nombre des états d'un modèle par la recherche, dans la matrice des probabilités des transitions, des éléments satisfaisants des conditions nécessaires et suffisantes permettant de regrouper des états en plusieurs ensembles disjoints qui réalisent une partition de l'ensemble des états possibles.

Des programmes connus MARCA [STE.76] qui utilisent ces propriétés, permettent une analyse directe des chaînes de MARKOV mais nous avons préféré opérer des réductions des matrices des probabilités de transition à partir de l'analyse du système ; en effet, compte tenu de notre objectif réduit qui est de mesurer un temps moyen passé dans un sous-ensemble donné d'états sans quitter celui-ci, nous avons intérêt à tenir compte de la nature des états pour obtenir un modèle de faible dimension, facilement exploitable. Pour cela, il est nécessaire de procéder à une nouvelle définition des états ; la présentation de cette méthode fait l'objet du paragraphe IV.3.3.

IV.3.2.3. Applications des chaînes ergodiques

Un processus décrit par une chaîne de MARKOV ergodique est caractérisé par le fait qu'il évolue, après une phase transitoire, vers un régime permanent, ce qui se traduit par le fait que le vecteur des probabilités d'état tend vers un vecteur limite. Les composantes a_i de ce vecteur sont les probabilités constantes que le système se trouve dans les états s_i lorsque le régime permanent est atteint.

$$\lim_{n \to \infty} \mathcal{X}(n) = \mathcal{X}(\infty) = \alpha \qquad \text{avec} \quad \alpha = [a_1, \ldots, a_r]$$

Le vecteur α des probabilités limite vérifie le système d'équations suivant :

$$\begin{cases} \alpha_{1,r} \; \mathbb{P}_{r,r} = \alpha_{1,r} \\ \alpha_{1,r} \; \xi_{r,1} = 1 \end{cases} \qquad \text{avec} \quad \xi_{r,1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

Nous utilisons une résolution directe du système linéaire pour la recherche des composantes du vecteur limite α .

Si on appelle $\mathbb{D}$ la matrice obtenue en remplaçant la première colonne de la matrice $(\mathbb{P} - \mathbb{I})$ par des 1, et $\mathbb{B}$ la matrice $[10\ldots0]_{1,r}$, le système que l'on doit résoudre est le suivant :

$$\alpha_{1,r} \; \mathbb{D}_{r,r} = \mathbb{B}_{1,r}$$

L'application de la théorie des chaînes de MARKOV permet d'obtenir des renseignements très intéressants sur l'évolution du système en régime transitoire ou en régime permanent, comme :

- le nombre moyen d'étapes nécessaires pour atteindre s_j pour la première fois à partir de s_i,
- la variance du nombre d'étapes nécessaires pour atteindre s_j à partir de s_i,
- le nombre moyen des tirages où l'on se trouve dans l'état s_j dans les n premiers tirages.

IV.3.2.4. Applications des chaînes absorbantes

Une chaîne absorbante peut toujours être définie par une matrice de la forme :

$$\mathbb{P} = \left(\begin{array}{c|c} \mathbb{I} & \mathbb{O} \\ \hline \mathbb{R} & \mathbb{Q} \end{array} \right) \begin{matrix} r-s \\ s \end{matrix}$$

$$\begin{matrix} r-s & s \end{matrix}$$

La matrice $(\mathbb{I} - \mathbb{Q})$ a un inverse $\mathbb{N}$ que l'on appelle la matrice fondamentale de la chaîne de MARKOV absorbante :

$$\mathbb{N} = (\mathbb{I} - \mathbb{Q})^{-1} = \mathbb{I} + \mathbb{Q} + \mathbb{Q}^2 + \cdots = \sum_{k=0}^{k=\infty} \mathbb{Q}^k$$

Soient T l'ensemble formé par les s états transitoires de la chaîne et $\tilde{T}$ l'ensemble des états absorbants. La matrice fondamentale d'une chaîne de MARKOV absorbante permet de parvenir aux résultats suivants :

- les éléments n_{ij} de la matrice $\mathbb{N}$ représentent le nombre moyen d'étapes, en comptant l'étape initiale i, que le processus passera dans l'état j avant d'être absorbé, s_i et s_j appartenant à T,
- le vecteur $\mathcal{C} = \mathbb{N} * \begin{bmatrix} 1 \\ 1 \end{bmatrix}_i$ a s composantes indiquant le nombre moyen d'étapes passées dans les états transitoires de T avant d'être absorbé à partir de chacun des états de T,
- les éléments b_{ij} de la matrice $\mathbb{B} = \mathbb{N}\mathbb{R}$ représentent les probabilités que le processus, partant de l'état transitoire $s_i (\in T)$ termine dans l'état absorbant $s_j (\in \tilde{T})$.

IV.3.2.5. Création d'une chaîne ergodique réduite par la théorie des chaînes absorbantes

On ne s'intéresse qu'aux s premiers états d'une chaîne ergodique à r états dont la matrice de probabilité de transition est la suivante :

$$\mathbb{P} = \begin{array}{c} \\ s \\ \tilde{s} \end{array} \begin{array}{c} \begin{array}{cc} s & \tilde{s} \end{array} \\ \left[\begin{array}{c|c} \mathbb{T} & \mathbb{W} \\ \hline \mathbb{R} & \mathbb{Q} \end{array}\right] \\ \begin{array}{cc} s & r-s \end{array} \end{array} \begin{array}{l} \\ \updownarrow s \\ \updownarrow r-s \end{array}$$

On construit une nouvelle chaîne ne contenant que les s états qui nous intéressent de la manière suivante :

- une seule étape du nouveau processus correspond, dans le processus initial, à la transition en une ou plusieurs étapes entre deux états quelconques de S ; les nouvelles probabilités de transition entre s_i et s_j de S s'obtiennent en calculant la probabilité que, partant de $s_i \in S$ dans le processus initial, le prochain état que l'on atteint dans S soit s_j.

Les propriétés des chaînes de MARKOV absorbantes permettent d'écrire la matrice de transition de la nouvelle chaîne ; soit $\overline{\mathbb{P}}$ cette nouvelle matrice :

$$\overline{\mathbb{P}} = \mathbb{T} + \mathbb{U}\left(\mathbb{I} - \mathbb{Q}\right)^{-1}\mathbb{R}$$

Dans cette expression, la matrice $\mathbb{T}$ correspond aux probabilités que, parti de S, ou soit encore dans S au tirage suivant ; $\mathbb{U}$ traduit le passage de S à $\tilde{S}$ lors du premier tirage ; $(\mathbb{I} - \mathbb{Q})^{-1}$ correspond au nombre de tirages où l'on reste dans $\tilde{S}$; enfin, $\mathbb{R}$ est la possibilité de revenir dans S à partir de $\tilde{S}$.

Ceci revient à rendre tous les éléments de S absorbants.

La nouvelle chaîne définie par cette matrice est aussi ergodique ; on obtient le vecteur limite des probabilités $\overline{\alpha}$ par la résolution du système linéaire suivant :

$$\begin{cases} \overline{\alpha}_{1,S}\ \overline{\mathbb{P}}_{s,s} = \alpha_{1,S} \\ \overline{\alpha}_{1,S}\ \mathcal{S}_{s,s} = 1 \end{cases}$$

Les composantes du vecteur $\overline{\alpha}$ obtenu sont égales aux s premières composantes du vecteur α de la première chaîne, normalisées à 1.

IV.3.3. Introduction de la notion du temps

Ce paragraphe constitue une analyse des possibilités d'ajouter la notion de temps aux propriétés purement probabilistes des chaînes de MARKOV que nous avons présentées au paragraphe IV.3.2. Nous décrivons et comparons plusieurs méthodes dans le cadre d'un objectif unique qui est d'évaluer le temps moyen θ moy qui sépare l'apparition d'un état quelconque d'un sous-ensemble S de l'arrivée du prochain état appartenant à ce même sous-ensemble. Ceci peut se représenter de la manière résumée sur la figure IV.6.

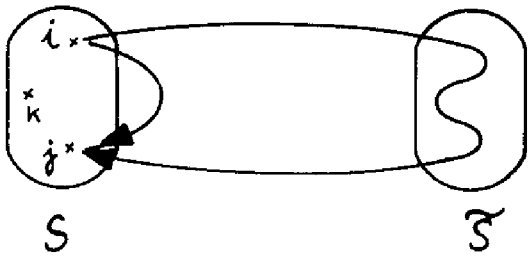


FIGURE IV.6.

Or, des résultats classiques comme le temps moyen passé entre les apparitions de deux états quelconques choisis parmi les r états que peuvent prendre le système [DEN.69] ne sont pas directement utilisables (figure IV.7.).

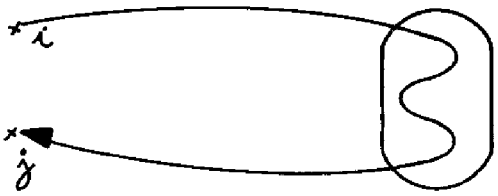


FIGURE IV.7.

Considérons l'exemple simple d'une chaîne de MARKOV à trois états ; la matrice des probabilités de transition qui lui est associée peut être représentée par le graphe des états de la figure IV.8.

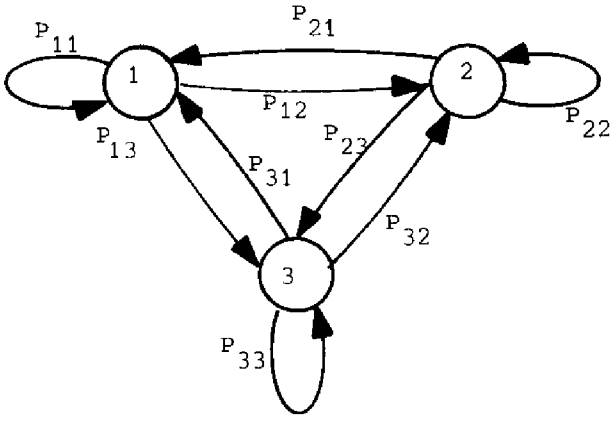


FIGURE IV.8.

On peut interpréter ce graphe, de la manière la plus générale, comme étant un cliché du système effectué au moment de son observation, dans lequel les flèches de transition constituent une relation entre ce cliché et le cliché suivant.

On peut introduire la notion de temps sur ce diagramme de deux manières :

- d'une manière que nous appellerons implicite, en considérant que la structure reste figée dans l'état défini au moment du cliché, jusqu'au cliché suivant, et que la transition se fait de manière instantanée,
- d'une manière appelée explicite en considérant au contraire que la structure se trouve dans l'état enregistré par le cliché à l'instant de l'observation, et qu'elle évolue jusqu'à son nouvel état pendant tout le temps que dure la transition.

Ces deux interprétations semblent complémentaires dans la mesure où la première peut représenter un système pour lequel la longueur des états est grande devant les temps de transition des circuits et la seconde représente mieux un système où l'on stocke un résultat dans un registre, après un délai consécutif à l'établissement d'une fonction combinatoire de plusieurs variables.

IV.3.3.1. Introduction implicite du temps

Interprétation des composantes du vecteur limite

Les termes (a_i) sont les probabilités que, lorsque l'on effectue une observation en régime permanent, le système se trouve dans les états (s_i). Dans une chaîne de MARKOV régulière, lorsque l'on effectue un grand nombre de tirages, les termes a_i représentent aussi le rapport du nombre de fois que le système se trouve dans l'état s_i sur le nombre total d'observations.

$$a_i = \frac{\text{nb. moyen d'apparitions de } s_i}{\text{nb. total d'observations}}$$

D'une manière équivalente, la valeur moyenne du nombre de tirages séparant deux apparitions successives du même état (s_i) est :

$$\boxed{\mathcal{T}_{ii} = \frac{1}{a_i}}$$

Ce résultat peut être généralisé à l'ensemble des états du sous-ensemble S : la probabilité que l'on se trouve dans l'un quelconque des s états de S est $\sum_{i=1}^{i=s} a_i$. Par suite, l'espérance mathématique du nombre d'étapes séparant un état de S de l'apparition du prochain état de S est :

$$\boxed{\mathcal{T}_{moy} = \frac{1}{\sum_{i=1}^{i=s} a_i}}$$

Première méthode de calcul du temps moyen Θ_{moy}

Si les tirages sont effectués à des intervalles de temps $\mathcal{T}$ constants, les numéros des tirages constituent un étalonnage du temps, et les quantités calculées prennent une signification évidente :

- $\mathcal{T}_{ii} = \frac{\mathcal{T}}{a_i}$ = temps séparant deux arrivées successives de (s_i),
- $\mathcal{T}_{moy} = \frac{\mathcal{T}}{\sum_{i=1}^{i=s} a_i}$ = temps moyen séparant l'apparition de deux états consécutifs de S.

Ceci constitue donc une première méthode pour calculer le résultat cherché :

$$\boxed{\mathcal{T}_{moy} = \Theta_{moy} = \frac{\mathcal{T}}{\sum_{s_i \in S} a_i}}$$

Calcul de Θ_{moy} avec une chaîne absorbante : 2ème méthode

Nous avons montré au paragraphe IV.3.2.5. comment on peut créer une nouvelle chaîne incluant uniquement les états de S ; cette chaîne n'est plus,sauf cas spécial, une succession de tirages qui s'effectuent à

intervalles de temps réguliers et ne comprend plus toutes les informations concernant l'évolution du système lorsqu'il se trouve dans les états de $\tilde{S}$.

Dans cette méthode de calcul de θ_{moy}, nous allons exploiter certains résultats concernant les états de la nouvelle chaîne et certaines données de la chaîne initiale.

Si on rend les états de S absorbants, on obtient une nouvelle chaîne dans laquelle les états de $\tilde{S}$ sont transitoires. Cette nouvelle chaîne décrit le fonctionnement du système de la même manière que la chaîne initiale tant que l'on reste en dehors des états absorbants de S et correspond encore à une succession de tirages qui sont effectués à intervalles de temps τ constants.

On en déduit donc la signification du vecteur τ présenté au paragraphe IV.3.2.4. :

$$\tau_{r-s,1} = N_{(r-s,r-s)} \; \xi_{(r-s,1)} \qquad \text{avec} \quad \xi_{(r-s,1)} = \begin{bmatrix} 1 \\ \\ 1 \end{bmatrix}$$

ses composantes correspondent au temps moyen passé dans $\tilde{S}$ à partir des états de $\tilde{S}$, avant d'être absorbé.

Partant d'un état de S, les probabilités d'arriver dans les états $\tilde{S}$ sont données par

$$\overline{\alpha}_{(1,s)} \times U_{(s,r-s)}$$

et par suite, le temps moyen recherché θ_{moy} est donné par le cheminement suivant : à la fin du premier intervalle τ, parti de S, on se retrouve soit en S, soit en $\tilde{S}$; connaissant la probabilité d'arriver dans chacun des états de $\tilde{S}$ et le temps passé dans $\tilde{S}$ avant de retourner dans S, on a :

$$\theta_{moy} = 1 + \alpha_{(1,s)} \; U_{(s,r-s)} \; N_{(r-s,r-s)} \; \xi_{(r-s,1)}$$

Au niveau de l'exploitation, les deux méthodes proposées ont un inconvénient majeur qui est lié à l'importance du volume et du temps de calcul dès que le nombre d'états devient grand. On note toutefois que même si la capacité de stockage en mémoire est identique dans les deux méthodes,

car il est indispensable de connaître les éléments des matrices Π , U , R et Φ , la seconde méthode est plus performante : il suffit d'effectuer l'inversion d'une matrice (s,s) pour le calcul de $\bar{\alpha}$ et l'inversion d'une matrice (r-s,r-s) pour le calcul de N , au lieu de l'inversion d'une matrice (r,r) pour le calcul du vecteur $\bar{\alpha}$ dans la première méthode.

IV.3.3.2. Introduction explicite du temps

La seconde interprétation donnée au graphe des états consiste à attribuer des valeurs de transition τ_{ij} à la chaîne de MARKOV. Dans ce cas, elle se rapproche davantage de la théorie des graphes car on préfèrera, pour représenter cette chaîne, utiliser un diagramme de fluence obtenu en juxtaposant les informations données par plusieurs clichés successifs effectués à intervalles de temps τ constants (figure IV.9.).

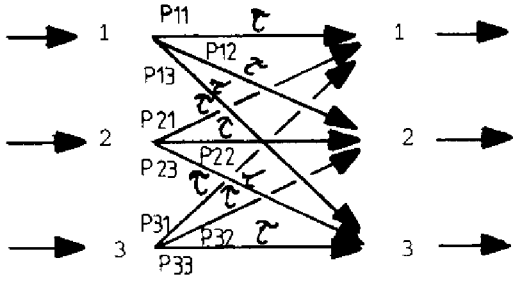
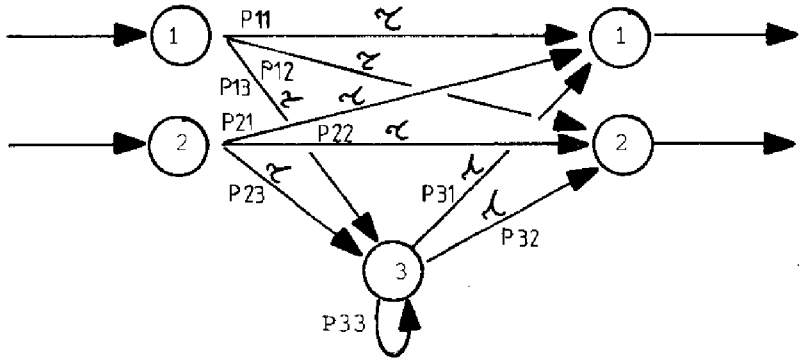


FIGURE IV.9.

Tous les résultats concernant les probabilités qui ont été établis au paragraphe IV.3.2. avant l'introduction de la notion de temps restent valables pour cette représentation. La chaîne que l'on obtient, si l'on ne se préoccupe pas de tous les chemins du diagramme de fluence (certains états du graphe des états), est une chaîne de MARKOV dont la matrice des transitions et le vecteur limite se calculent comme indiqué au paragraphe IV.3.2.5.

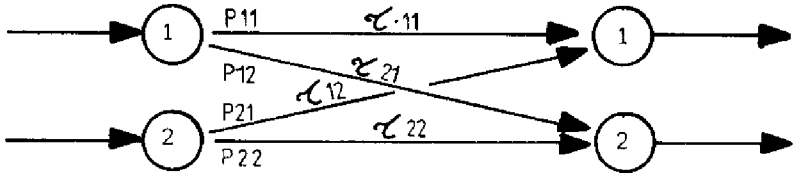
A titre d'exemple, si on s'intéresse aux états 1 et 2 de la chaîne à trois états de la figure IV.8., on effectue les modifications conduisant à la figure IV.10.



Nouveau tirage p Nouveau tirage p+1

FIGURE IV.10.

En remarquant que les nouveaux tirages ne s'effectuent plus à temps constant, on aboutit finalement à la représentation donnée par la figure IV.11. par suppression de l'état 3.



tirage p tirage p+1

FIGURE IV.11.

Cette élimination des états que l'on veut supprimer s'appelle la transformation de "Star Mesh"[BEI.71] : il s'agit d'un procédé itératif de suppression des noeuds qui obéit aux lois de la théorie des graphes.

Nous avons donné au paragraphe IV.3.2.5. le produit matriciel permettant de supprimer plusieurs noeuds à la fois pour obtenir la nouvelle matrice de transition :

$$\overline{\mathbb{P}} = \mathbb{T} + \mathbb{U}\left(\mathbb{I} - \mathbb{Q}\right)^{-1}\mathbb{R}$$

L'opération correspondante en ce qui concerne les temps de transition est beaucoup plus complexe, comme le montrent les expressions de $\overline{p_{ij}}$ et $\overline{\tau_{ij}}$ dans le cas de la suppression d'un seul noeud présentée sur la figure IV.12.

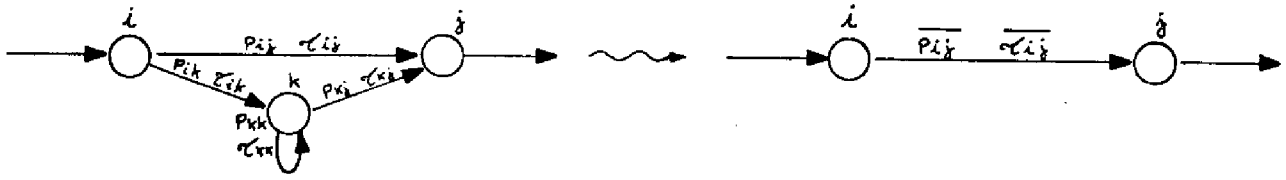


FIGURE IV.12.

Quelle que soit la méthode de suppression des noeuds -point par point ou globale-, on est amené à la construction d'une nouvelle chaîne de MARKOV, avec valeurs de transition [KAV.70] ,définie par deux matrices :

- la matrice $\overline{\Pi}$ des probabilités des transitions,
- la matrice $\overline{\overline{\Pi}}$ des temps de transition.

Connaissant les probabilités et les temps de transition entre deux états quelconques de la nouvelle classe, on obtient le temps moyen pour passer d'un sous-ensemble A_i quelconque de S à un sous-ensemble quelconque A_j de S en écrivant que le temps moyen global est la somme des temps moyens :

- les termes $\overline{a_i}$ étant les probabilités limites de se trouver dans les états s_i de S, sachant que l'on est dans S, le temps moyen τ_{A_i,A_j} passé entre ces deux sous-ensembles est :

$$\tau_{A_i,A_j} = \frac{1}{\sum_{S_i \in A_i} \overline{a_i}} \sum_{s_i \in A_i} \left[\frac{\overline{a_i}}{\sum_{S_j \in A_j} \overline{p_{ij}}} \sum \overline{p_{ij}}\ \overline{\tau_{ij}} \right]$$

Pour le résultat θ_{moy} qui nous intéresse, on a les relations

$$A_i = A_j = S$$

$$\sum_{S_i \in A_i} \overline{a_i} = \sum_{S_j \in A_j} p_{ij} = 1$$

Par suite :

$$\theta_{moy} = \sum_{s_i \in S} \overline{a_i} \left(\sum_{s_j \in S} \overline{p_{ij}} \ \overline{\tau_{ij}} \right) = \sum_{i,j} \overline{a_i} \ \overline{p_{ij}} \ \overline{\tau_{ij}}$$

C'est cette troisième méthode que nous utiliserons pour calculer θ_{moy} dans la majorité des cas, car :

- elle conduit à un volume de traitement plus réduit que les autres,
- il est souvent aussi facile d'écrire directement les matrices $\overline{\mathbb{P}}$ et $\mathbb{T}$ que la matrice $\mathbb{P}$,
- dans le cas particulier d'un temps de service exponentiel, la matrice $\mathbb{T}$ est simple à écrire, et on peut obtenir $\overline{\mathbb{P}}$ à partir de $\mathbb{P}$ par le produit matriciel indiqué au paragraphe IV.3.2.5.

L'identité des résultats des deuxième et troisième méthodes est montrée en annexe ; ces deux méthodes correspondent à l'étude de chemins différents dans le graphe de fluence.

IV.4. CARACTERISTIQUES DU MODELE DE DESCRIPTION

La forme modulaire de la mémoire et la diminution du prix des organes de traitement sont deux facteurs importants qui ont conduit au développement de nouvelles architectures pour lesquelles le souci d'optimiser les moyens de traitement n'est plus primordial ; on trouve ainsi des structures où plusieurs processeurs partagent une mémoire commune. L'évaluation des performances de ce type de systèmes a donné lieu à de nombreuses études :

- BANDHARKAR [BAN.73] - [BAN.75], reprenant les travaux de STRECKER [STR.70] présente une étude très complète sur les conflits d'accès aux différents modules mémoire du système multi-miniprocesseur "cmmp" développé à l'Université de CARNEGIE-MELLON. Il calcule, en utilisant des chaînes de MARKOV le débit d'exécution des instructions dans le cadre d'une structure où le nombre de processeurs et le nombre de modules mémoire peuvent varier.
  Cette étude est menée dans les trois cas envisageables, en fonction des valeurs relatives des temps de cycle t_p des processeurs et t_m des mémoires

- $t_p = t_m$,
- $t_p > t_m$,
- $t_p < t_m$.

- SASTRY [SAS.75] étudie le même phénomène à l'Université du MINNESOTA mais en distinguant deux origines à l'oisiveté des processeurs :
  .disponibilité des moyens,
  .disponibilité des opérandes,
  qui correspondent aux flux des instructions et des données ; il utilise également les chaînes de MARKOV.
- Les résultats de RAVINDRAN [RAV.73] de l'Université de STANFORD, utilisés par REYLING [REY.74] et par OHMORI [OHM.74], sont motivés par l'apparition sur le marché de circuits LSI.
  RAVINDRAN développe un modèle stochastique simple dans lequel $t_p = 2\ t_m$ et OHMORI, dans le cadre du projet MICS, s'intéresse au cas où t_p devient grand (de l'ordre de 5 à 10 fois) par rapport à t_m, à la suite de la création de fonctions microprogrammées.

Malgré l'analogie du schéma de représentation d'A.S.M.O.D.E.E. 02 avec les systèmes multiprocesseurs, nous avons été amené à développer un modèle propre pour évaluer les performances de cette structure caractérisée par deux traits de fonctionnement :

- un taux d'occupation des organes de traitement qui peut être faible,
- une optimisation de l'utilisation de la mémoire de programme et du bus de communication qui est limitée, par un souci primordial d'une grande simplicité, à un recouvrement des instructions lorsque cela est possible.

Ce mode de fonctionnement transpose au niveau des processeurs les conflits qui se situent pour le multitraitement au niveau de l'accès à la mémoire ; d'une manière paradoxale, nous retrouverons donc des analogies avec les travaux de BANDHARKAR dans l'hypothèse ($t_p < t_m$), dans le cas particulier d'une structure ayant un processeur unique et plusieurs mémoires. Le fonctionnement des structures satisfaisant la relation ($t_p < t_m$) n'est cependant étudié que de manière approchée dans cette étude très vaste où le nombre

des processeurs peut aller jusqu'à seize, à cause des difficultés d'exploitation auxquelles conduirait la dimension du modèle correspondant.

Ce cas est aussi étudié par COCHI [COC.74] qui fait intervenir la présence d'instructions de saut .

On retrouve également des éléments communs avec des études portant sur l'utilisation de mémoires de masse et en particulier, avec l'étude de RYAN [RYA.74] qui développe un modèle de MARKOV traduisant le fonctionnement d'un système où l'exécution d'un programme débute dès le début de son chargement en mémoire centrale.

En ce qui concerne le chemin des données, la structure décrite correspond au schéma 7 de la figure IV.13 qui rappelle la taxonomie de ANDERSON [AND.75] ; les structures qui y sont représentées sont cataloguées en fonction de la stratégie et du contrôle des transferts entre les modules et sur ce plan, A.S.M.O.D.E.E. 02 sera représenté par le schéma 4, à cause de l'organe centralisé qui commande l'accès au bus.

Nous allons étudier les performances des structures représentées par le schéma 4 de la figure IV.13. dont le comportement est analogue à celui d'A.S.M.O.D.E.E. 02, c'est-à-dire, caractérisé par les points suivants :

- les éléments de la structure peuvent se trouver dans l'un des trois modes suivants :
  .mode repos,
  .mode de liaison avec le reste du système,
  .mode autonome pendant la durée de leur traitement,
- l'accès au moyen de communication est accordé pour un temps α constant aux éléments du système qui sont sélectionnés, lorsqu'ils sont disponibles,
- en cas d'indisponibilité d'un des éléments désignés, la recherche des actions suivantes est retardée jusqu'à ce que l'action en attente puisse être initialisée.

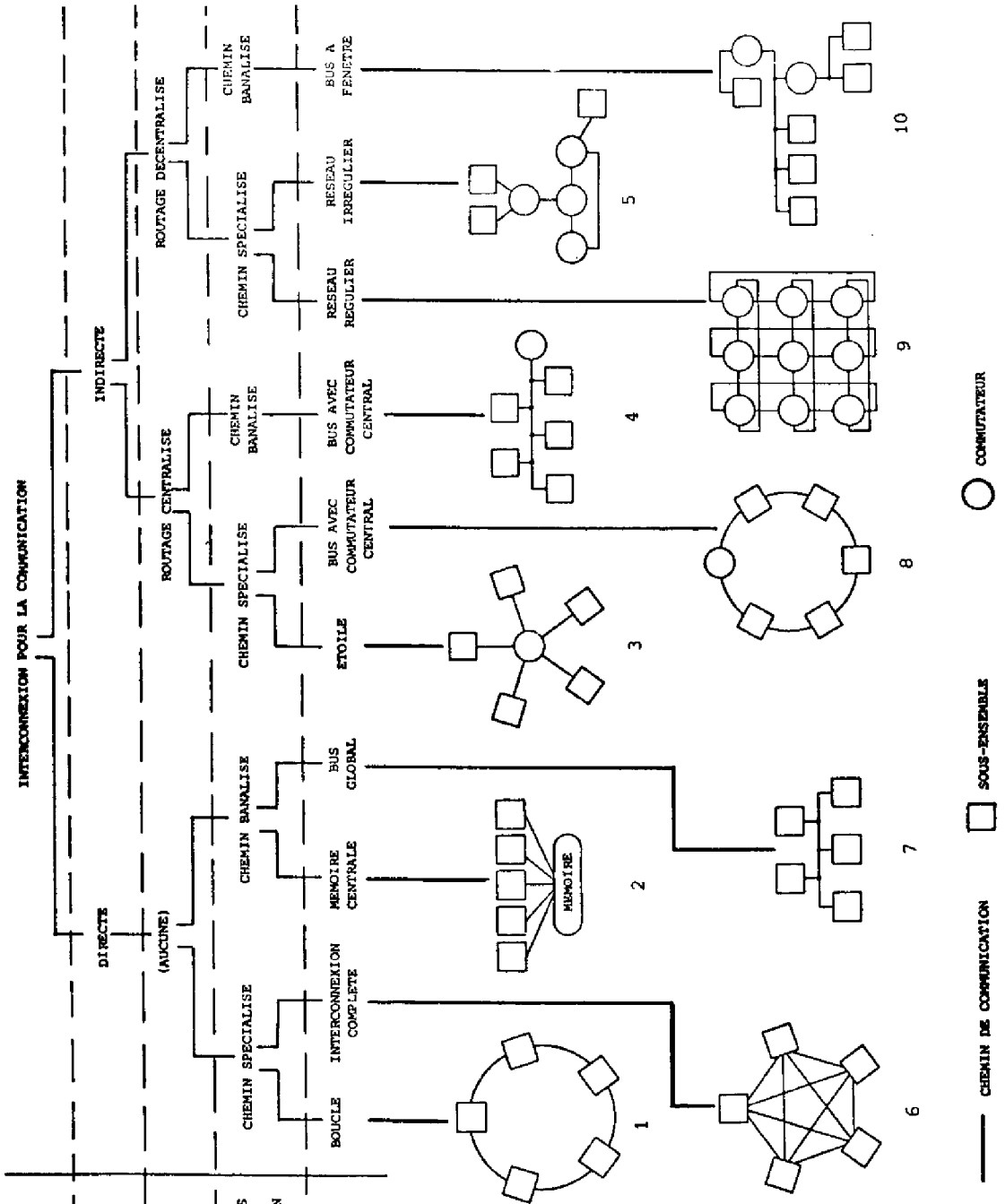

INTERCONNEXION POUR LA COMMUNICATION
STRATEGIE DES ECHANGES
METHODE DE CONTROLE DES ECHANGES
STRUCTURE DES CHEMINS DE COMMUNICATION
ARCHITECTURE DU SYSTEME
DIRECTE
INDIRECTE
(AUCUNE)
ROUTAGE CENTRALISE
ROUTAGE DECENTRALISE
CHEMIN SPECIALISE
CHEMIN BANALISE
CHEMIN SPECIALISE
CHEMIN BANALISE
CHEMIN SPECIALISE
CHEMIN BANALISE
BOUCLE
INTERCONNEXION COMPLETE
MEMOIRE CENTRALE
BUS GLOBAL
ETOILE
BUS AVEC COMMUTATEUR CENTRAL
BUS AVEC COMMUTATEUR CENTRAL
RESEAU REGULIER
RESEAU IRREGULIER
BUS A FENETRE
MEMOIRE
1
2
3
4
5
6
7
8
9
10
CHEMIN DE COMMUNICATION
SOUS-ENSEMBLE
COMMUTATEUR

FIGURE IV.13.

Dans l'optique de regarder les difficultés de prise en compte d'hypothèses variées pour le fonctionnement du système de commande et la caractérisation des tâches, nous avons été amené à développer des modèles qui ne rentrent pas directement dans le cadre de la description d'A.S.M.O.D.E.E. 02.

C'est ainsi que nous développerons un modèle intermédiaire de description, moins pour l'intérêt propre des systèmes représentés par ce modèle que pour les deux avantages suivants :

- ce modèle simple correspond à la démarche naturelle qui consiste à introduire pas à pas les différentes contraintes du système décrit, et il permet de présenter plus facilement la méthode de prise en compte des variations des caractéristiques matérielles du système,
- il revêt une grande importance dans le cadre des structures à bus avec un organe central de contrôle, car ce modèle intermédiaire et le modèle final d'A.S.M.O.D.E.E. 02 correspondent aux deux cas extrêmes où :
  .les communications directes entre les éléments du système ne sont pas possibles,
  .les communications entre modules sont très fortes.

La transposition des résultats obtenus au cas d'A.S.M.O.D.E.E. 02 nous permettra de mesurer l'influence des dépendances introduites par le chemin des données -communications entre opérateurs- qui se superposent aux dépendances créées par le chemin des fonctions (évaluées dans le modèle intermédiaire).

CHAPITRE V

MODELE D'UNE STRUCTURE MULTIOPERATEUR SANS ECHANGE DIRECT ENTRE LES MODULES

-

Nous avons indiqué précédemment que nous considérions le modèle décrit dans ce chapitre comme une étape intermédiaire servant de support à l'étude de l'influence des paramètres fonctionnels et temporels du système développé.

Nous présentons successivement dans ce chapitre :

- la recherche d'une description du comportement du système par un modèle de représentation de l'enchaînement des instructions,
- le passage progressif à un modèle structurel en donnant une description de plus en plus fine de l'évolution du matériel,
- la construction du modèle structurel,
- la réduction de la dimension de ce modèle structurel,
- son exploitation dans le cas de structures symétriques.

La symétrie des structures envisagées est une hypothèse que nous ferons plusieurs fois ; on admet :

- la symétrie matérielle : tous les opérateurs ont le même temps de traitement,
- la symétrie fonctionnelle : il y a équiprobabilité de sélection de chacun des opérateurs.

Ces hypothèses viennent s'ajouter aux hypothèses de caractérisation des tâches que nous avons faites dans le chapitre IV.

Malgré notre objectif de procéder à une étude essentiellement qualitative, les problèmes de dénombrement revêtent une place importante à cause de l'intérêt qu'ils ont au niveau de l'exploitation des modèles.

V.1. MODELE FONCTIONNEL POUR UNE STRUCTURE SYMETRIQUE

Nous avons indiqué au paragraphe IV.1.2. que les paramètres intervenant dans un modèle sont de deux types :

- les paramètres fonctionnels, représentatifs des tâches à exécuter et attribués aux numéros des instructions dans le déroulement du programme,

- les paramètres temporels propres à l'exécution des instructions par le système de commande.

L'intérêt du modèle fonctionnel d'un système obtenu en décrivant son fonctionnement par le comportement du programme provient de la possibilité de modifier facilement les hypothèses caractérisant les tâches à exécuter ; le système de commande intervient dans ce type de modèle uniquement par les temps d'exécution des instructions du programme.

Bien que l'analyse des dernières instructions du programme donne des informations sur le temps d'attente nécessaire à l'exécution de toute nouvelle instruction, une recherche précise de ce temps d'attente nécessite parfois la connaissance d'un très grand nombre de pas précédents du programme allant jusqu'à la liste de toutes les instructions passées depuis le début du programme.

Pour éviter d'avoir des modèles lourds à manier, on a le choix entre deux types d'approximation :

- émettre des hypothèses globales de temps d'occupation,
- introduire dans le modèle fonctionnel des données caractérisant l'état des ressources matérielles du système de commande et tendre ainsi vers l'établissement d'un modèle structurel.

C'est la seconde solution que nous avons retenue pour élaborer les modèles fonctionnels que nous allons présenter dans ce paragraphe.

V.1.1. Modèle fonctionnel

Le modèle fonctionnel développé sera présenté sur un exemple que nous utiliserons à plusieurs reprises dans ce chapitre : il s'agit d'un système qui vérifie les hypothèses de symétries matérielle et logicielle que nous avons faites :

- temps de traitement des opérateurs : $T = 3 \times \tau$,
- nombre d'opérateurs : $N = 5$,
- probabilité de référence : $P = \frac{1}{N} = 0,2$ (pour chaque opérateur).

V.1.1.1. Description par le comportement du programme

Considérons l'exécution d'un programme, matérialisée par le nom des opérateurs tirés à chaque pas du programme et dont la séquence est donnée sur la figure V.1.

Numéro du pas du programme	0, 1, 2,..., N-6, N-5, N-4, N-3, N-2, N-1, N
Opérateur sélectionné	5, 2, 5,..., 3, 4, 3, 3, 1, 3, 5

FIGURE V.1.

La méthode qui consiste à décrire le fonctionnement du système en représentant le comportement du programme par les p derniers opérateurs nommés présente :

- des avantages au niveau de la matrice des probabilités de transition entre états, qui s'écrit facilement et peut être modifiée si l'on veut introduire un phénomène de localité traduisant l'influence d'une instruction sur les instructions immédiatement en séquence,
- des inconvénients, en particulier pour les durées de transition entre états qui sont difficiles à calculer et, dans certains cas, ne peuvent être connuesavec certitude.

A titre d'exemple,si l'on s'intéresse aux cinq derniers opérateurs nommés, on a dans l'exemple choisi les résultats donnés par la figure V.2.

Etat actuel	Nom de l'opérateur référencé par l'instruction N+1	Probabilité de cette référence	Nouvel état	Temps de transition nécessaire
3, 3, 1, 3, 5	1	1/5	3, 1, 3, 5, 1	1 τ
	2	1/5	3, 1, 3, 5, 2	1 τ
	3	1/5	3, 1, 3, 5, 3	3 τ
	4	1/5	3, 1, 3, 5, 4	1 τ
	5	1/5	3, 1, 3, 5, 5	4 τ

FIGURE V.2.

Au contraire, si on part, à titre d'exemple, du nouvel état (3,1,3,5,1) sans connaître l'état dont il est issu, on ne peut pas faire les mêmes conclusions; on obtient alors lesrésultats donnés par la figure V.3.

Etat de la machine à l'instruction N	Opérateur référencé par l'instruction N+1	Probabilité de cette référence	Nom du nouvel état	Temps de transition
3, 1, 3, 5, 1	1	1/5	1, 3, 5, 1, 1	4 τ
	2	1/5	1, 3, 5, 1, 2	1 τ
	3	1/5	1, 3, 5, 1, 3	1 τ ou 2τ
	4	1/5	1, 3, 5, 1, 4	1 τ
	5	1/5	1, 3, 5, 1, 5	2 τ ou 3τ

FIGURE V.3.

Cette différence provient du fait que dans le premier cas, la répétition de l'appel à l'opérateur 3, aux instructions N-4 et N-3, réinitialise l'ensemble des éléments de la structure à leur état de repos, alors que dans le second cas, il y a une incertitude due au fait que l'on ne sait pas dans quel état se trouvent les opérateurs 3 et 5 quand ils sont appelés par les instructions N-1 et N.

Pour lever cette incertitude qui peut se propager dans certains cas tout au long du programme, il faudrait inclure dans la définition des états d'un tel modèle fonctionnel tous les opérateurs qui ont été tirés depuis le premier pas du programme. Ceci conduit à un nombre et à une longueur des états qui sont prohibitifs.

De plus, une telle description qui fait intervenir l'état initial est contraire à l'hypothèse d'ergodicité que nous avons faite : étude du système en régime permanent.

C'est pour cette raison que l'on est conduit à faire des hypothèses simplificatrices donnant un modèle ayant un nombre et une longueur d'états réduits ; on peut par exemple :

- prendre la valeur moyenne des différents temps de transition possibles,
- admettre que la structure est au repos au moment où le premier tirage cité dans l'énoncé d'un état intervient.

V.1.1.2. Nombre d'états du modèle fonctionnel

C'est la seconde hypothèse que nous avons retenue pour construire un modèle fonctionnel exploitable.

Le nombre NB des états du modèle de description du comportement du programme dépend à la fois du nombre (p) de pas de programme enregistré et du nombre (N) d'opérateurs de la structure, selon la formule :

$$NB = N^p$$

ou, en simplifiant par le nombre de cas identiques, à une permutation près du nom des opérateurs :

$$NB = N^{p-1}$$

La progression du nombre d'états avec (p) est donc extrêmement rapide et le problème consiste alors à faire un compromis entre la précision des résultats recherchée et les moyens informatiques qu'il est nécessaire de mettre en oeuvre pour les obtenir.

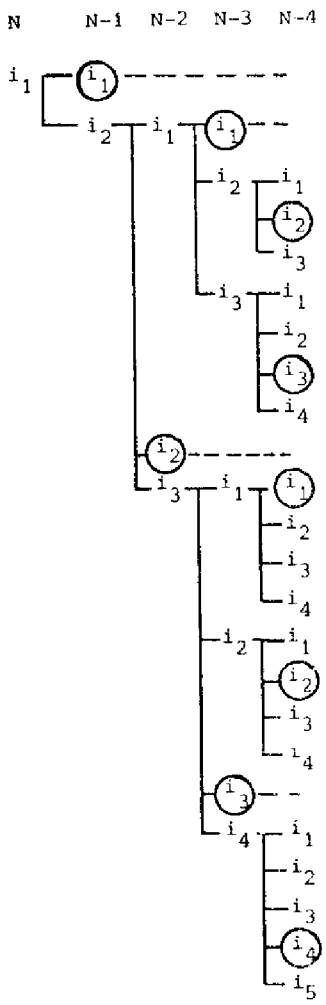
V.1.2. Introduction de données matérielles dans le modèle fonctionnel

V.1.2.1. Réduction du nombre d'états : deuxième modèle

Si l'on n'est pas obligé de mémoriser les derniers pas du programme pour des raisons fonctionnelles -et c'est le cas si on n'introduit pas de dépendances locales- on peut réduire le nombre des états du modèle de description en introduisant certaines particularités du comportement de la machine dues au fait que le flot des instructions s'interrompt lorsqu'il y a conflit d'accès à une ressource : dans l'exemple choisi, si on ne se préoccupe pas des opérateurs appelés avant une répétition -parce que deux références successives à un même moyen donnent le temps à toutes les ressources de se libérer- on est conduit à considérer les états de la figure V.4.

En partant du dernier pas de programme et en scrutant les instructions précédentes jusqu'au premier conflit rencontré :

En partant du dernier conflit et en scrutant les pas de programme suivants, jusqu'à l'instruction N :



Le passage d'un tableau à l'autre se fait en numérotant les mêmes états à partir de la droite ou de la gauche : nous avons gardé sur une même ligne les deux notations d'un même état.

N	N-1	N-2	N-3	N-4
.	.	.	.	.
i_1	0			
i_1	i_2	i_1	0	
i_1	i_2	i_1	i_2	i_1
i_2	i_1	i_2	i_1	0
i_3	i_2	i_3	i_2	i_1
i_1	i_3	i_1	i_2	i_1
i_3	i_1	i_3	i_2	i_1
i_2	i_3	i_2	i_1	
i_3	i_4	i_3	i_2	i_1
i_2	i_1	0		
i_1	i_3	i_2	i_1	0
i_2	i_1	i_3	i_2	i_1
i_2	i_3	i_1	i_2	i_1
i_2	i_4	i_3	i_2	i_1
i_1	i_2	i_3	i_2	i_1
i_3	i_1	i_2	i_1	0
i_3	i_2	i_1	i_2	i_1
i_4	i_2	i_3	i_2	i_1
i_3	i_2	i_1	0	
i_1	i_4	i_3	i_2	i_1
i_4	i_1	i_3	i_2	i_1
i_4	i_3	i_1	i_2	i_1
i_4	i_3	i_2	i_1	0
i_5	i_4	i_3	i_2	i_1

FIGURE V.4.

On peut facilement majorer le nombre des états du modèle, en fonction du nombre p de pas de programme pris en compte, par l'expression :

$$1+1\ [1+2\ [1+3\ [1+4\ [1+4\ [1+4\ - - - -\quad < 1 + \frac{1-4^p}{1-4}$$

$$p = 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \qquad \text{ou} \left(< 1 + \sum_{K=1}^{K=p-1} K! \text{ pour } p \leq N \right)$$

Le nombre d'états de ce modèle,qui est une fonction du nombre (p) de pas de programme enregistrés, peut être calculé de manière exacte en faisant apparaître le nombre (a) d'opérateurs différents apparaissant dans les chaînes d'instructions que représentent les états.

L'énumération se fait directement avec la seconde notation, en déduisant le nombre des chaînes ayant a_j éléments distincts en considérant (p+1) pas de programme si on connaît les mêmes informations pour (p) pas de programme:

$$a_j(p+1) = a_{j-1}(p) + (j-1)\, a_j(p)$$

avec

$a_j(p+1)$ = nombre de chaînes ayant j opérateurs distincts en considérant (p+1) pas de programme,

$a_{j-1}(p)$ = nombre de chaînes ayant j-1 opérateurs distincts en considérant (p) pas de programme,

$(j-1)a_j(p)$ = nombre de chaînes ayant j opérateurs distincts dans p pas de programme auquel on rajoute l'un des j opérateurs déjà présents dans la chaîne, à l'exception du dernier opérateur appelé, qui donnerait un conflit, donc une chaîne déjà existante.

Les résultats ainsi obtenus sont donnés sur la figure V.5.

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	Nombre d'états
N	1	0	0	0	0	0	0	1 → 1
N-1	1	1	0	0	0	0	0	2 → 2
N-2	1	2	1	0	0	0	0	4 → 4
N-3	1	3	4	1	0	0	0	9 → 9
N-4	1	4	11	7	1	0	0	24 → 24
N-5	1	5	26	32	11	1	0	76 → 75
N-6	1	6	57	122	76	16	1	279 → 262

FIGURE V.5.

V.1.2.2. *Exemple d'application N=5 T=3 x* τ

L'exploitation de ce modèle pour calculer le temps moyen d'exécution d'une instruction se fait en appliquant les résultats des chaînes de MARKOV avec valeurs de transition : c'est la troisième méthode que nous avons développée au chapitre IV.

La démarche consiste à :

- établir la matrice $\mathbb{P}$,
- calculer le temps moyen θ_i de transition à la sortie de chaque état i,
- résoudre le système linéaire $\begin{cases} \alpha \mathbb{P} = \alpha \\ \alpha \xi = 1 \end{cases}$
- déduire le temps $T_{moy} = \sum_i \alpha_i \theta_i$.

Nous développons cette méthode sur le cas de la structure symétrique déjà utilisée :

- N=5 opérateurs,
- T=3 τ comme temps de traitement autonome suivant un temps τ de communication,
- $P_i = P = \frac{1}{5}$ probabilité de référence à chaque opérateur,

ceci pour un nombre (p=4) de pas intervenant dans la constitution des états.

Soit (i_3, i_1, i_2, i_1) la dénomination de l'un des états du système selon la notation de droite de la figure V.4., c'est-à-dire, en utilisant une appellation contractée des opérateurs :

Numéro de l'instruction	N, N-1, N-2, N-3
Opérateur sélectionné	3, 1, 2, 1

FIGURE V.6.

Quel que soit l'état du système à cet instant, les cinq "événements" qui peuvent se produire sont la sélection par l'instruction de numéro (N+1) de l'un des cinq opérateurs constituant le système.

Dans le cas de l'état (3,1,2,1), les états que l'on atteint et le temps mis pour y parvenir dépendent de l'opérateur nommé par (N+1) (fig.V.7).

Nom de l'opérateur de l'instruction (N+1)		Nom du nouvel état	Temps de la transition
ancien état (3,1,2,1)	1	1,3,1,2 ≡ 2,3,2,1	3 x τ
	2	2,3,1,2 ≡ 1,3,2,1	1 x τ
	3	3,3,1,2 ≡ 1,	4 x τ
	4	4,3,1,2 ≡ 4,3,2,1	1 x τ
	5	5,3,1,2 ≡ 4,3,2,1	1 x τ

Temps moyen : $\theta = 2\tau$

FIGURE V.7.

On calcule les coefficients de la matrice des probabilités de transition entre états comme il est indiqué sur la figure V.7. pour tous les états qui interviennent : on obtient le tableau de la figure V.8. sur laquelle apparaît également le temps moyen mis pour sortir de chaque état.

Nom des états \ Probabilités de référence		P_1	P_2	P_3	P_4	P_5	θ_i
1	(1)	$(1)_4$	$(2)_1$	$(2)_1$	$(2)_1$	$(2)_1$	1,6 x τ
21	(2)	$(3)_3$	$(1)_4$	$(4)_1$	$(4)_1$	$(4)_1$	2 x τ
121	(3)	$(1)_4$	$(5)_1$	$(6)_1$	$(6)_1$	$(6)_1$	1,6 x τ
321	(4)	$(7)_2$	$(8)_3$	$(1)_4$	$(9)_1$	$(9)_1$	2,2 x τ
2121	(5)	$(5)_3$	$(1)_4$	$(6)_1$	$(6)_1$	$(6)_1$	2 x τ
3121	(6)	$(8)_3$	$(7)_1$	$(1)_4$	$(9)_1$	$(9)_1$	2 x τ
1321	(7)	$(1)_4$	$(7)_1$	$(8)_2$	$(9)_1$	$(9)_1$	1,8 x τ
2321	(8)	$(6)_1$	$(1)_4$	$(5)_1$	$(6)_1$	$(6)_1$	2 x τ
4321	(9)	$(9)_1$	$(7)_2$	$(8)_3$	$(1)_4$	$(9)_1$	2,2 x τ

FIGURE V.8.

$$\mathbb{P} = \frac{1}{5}\begin{bmatrix} 1 & 4 & & & & & & & \\ 1 & & 1 & 3 & & & & & \\ 1 & & & & 1 & 3 & & & \\ 1 & & & & & & 1 & 1 & 2 \\ 1 & & & & 1 & 3 & & & \\ 1 & & & & & & 1 & 1 & 2 \\ 1 & & & & & & 1 & 1 & 2 \\ 1 & & & & 1 & 3 & & & \\ 1 & & & & & & 1 & 1 & 2 \end{bmatrix}$$

P(1)	= α_1	= 0,200	θ_1	=	1,6
P(21)	= α_2	= 0,160	θ_2	=	2
P(121)	= α_3	= 0,032	θ_3	=	1,6
P(321)	= α_4	= 0,096	θ_4	=	2,2
P(2121)	= α_5	= 0,032	θ_5	=	2
P(3121)	= α_6	= 0,096	θ_6	=	2
P(1321)	= α_7	= 0,096	θ_7	=	1,8
P(2321)	= α_8	= 0,096	θ_8	=	2
P(4321)	= α_9	= 0,192	θ_9	=	2,2

$$T_{moy} = 1{,}907\,\tau$$

V.1.2.3. Convergence de la méthode

Le temps moyen calculé par le modèle dépend du nombre de pas de programme (p) pris en compte dans la désignation d'un état. Les résultats pour l'exemple ci-dessus sont rassemblés dans le tableau de la figure V.9.

(P)	1	2	3	4
T moy :	1,600 x τ	1,920 x τ	1,952 x τ	1,907 x τ

FIGURE V.9.

Nous avons donné ces résultats numériques dans le but de montrer qu'ils n'évoluent pas de manière uniforme : en effet, toutes les valeurs moyennes θ_i des temps de transition possibles lorsque la durée du traitement est de 3 x τ apparaissent explicitement à partir de p=4 ; il faut noter que les probabilités d'être dans chacun des états représentatifs

du système ne sont pas les valeurs exactes des coefficients qu'il faudrait leur attribuer pour le calcul du temps moyen exact.

Cependant, l'incertitude des résultats diminue avec p et on peut se rapprocher de la valeur exacte de T_{moy} ($T_{moy\ p=\infty} = 1{,}92 \times \tau$) en faisant croître p. Cette convergence se montre en analysant la matrice des probabilités des transitions entre états. La première des deux notations des états présentées sur la figure V.4. est la plus adaptée pour cette analyse.

Considérons la partition des états du système :

- A regroupe l'ensemble des états représentés par des chaînes de moins de p éléments,
- B regroupe l'ensemble des états représentés par des chaînes ayant exactement p éléments.

On peut représenter la matrice des probabilités de transition et le graphe des transitions associé comme cela apparaît sur la figure V.10.

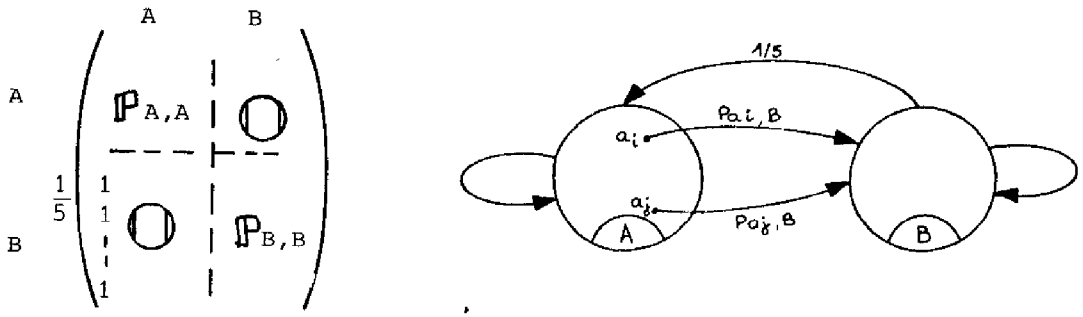


FIGURE V.10.

Le fait de partitionner l'ensemble (B) en deux nouveaux sous-ensembles ne modifie pas les probabilités de transition de chacun des éléments de (A) vers l'ensemble (B), ni la probabilité de la transition de B vers A.

Soit B',B" la partition de B :

- {B'} = états où serait apparu un conflit entre les instructions (N-(p-1)) et (N-p),
- {B"} = états où l'introduction d'une instruction supplémentaire n'entrainerait pas un nouveau conflit.

Le fait d'étendre la chaîne de MARKOV initiale par le développement de l'un des groupes d'états donne une nouvelle chaîne de MARKOV. Les probabilitésd'être dans les états de la nouvelle chaîne qui ont été repris sans changement par rapport à la première chaîne ne sont pas modifiées (cette extension constitue la démarche inverse du regroupement d'états présenté au paragraphe IV.3.2.2.). La matrice et le graphe de transition sont donnés sur la figure V.11.

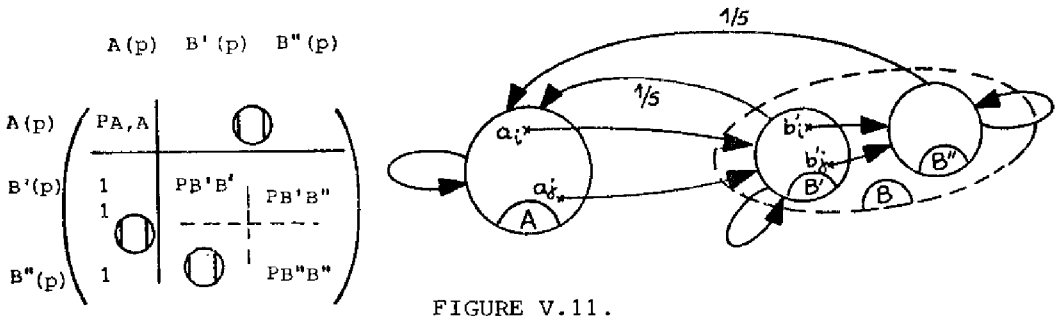


FIGURE V.11.

On procède alors au regroupement de A et B' en un seul ensemble comme le montre la figure V.12.

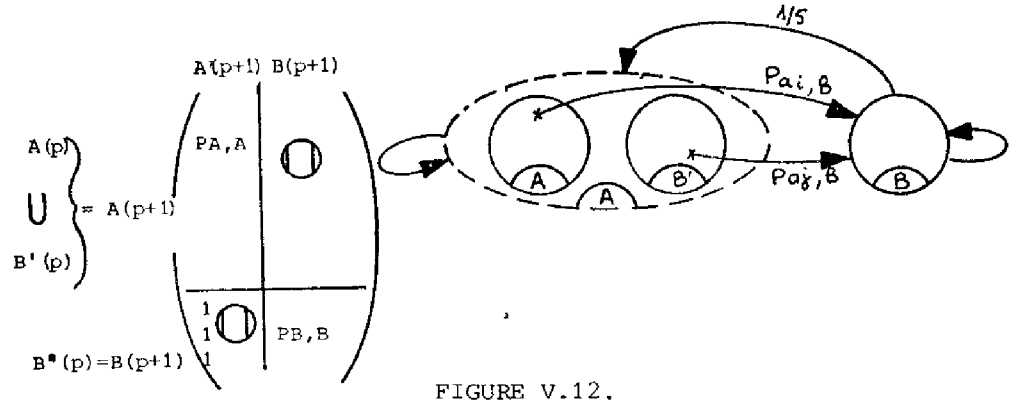


FIGURE V.12.

Comme il n'y a pas de retour possible de B" vers B', la probabilité de se trouver dans les ensembles B(p) successifs diminue quand p croît, puisque les probabilités a'_i ne sont pas toutes nulles. Or, ces différents ensembles B(p) regroupent tous les états pour lesquels les temps de transition ne sont pas connus avec certitude, mais appartiennent à un lot fini de temps possibles. La valeur de l'incertitude diminue donc quand p croît.

V.1.3. *Passage à un modèle structurel*

Parti d'un modèle fonctionnel, nous avons introduit une réduction du nombre des états par la prise en compte de caractéristiques du système en regroupant certains états car, du point de vue du comportement du système, ils sont équivalents, c'est-à-dire que les probabilités de transition et le temps moyen de transition hors de ces états sont identiques.

Si on poursuit cette démarche en regroupant des états, seulement parce qu'ils ont le même temps moyen de transition, on obtient un modèle que nous appelons structurel.

Appliqué à l'exemple retenu (N=5, T=3 α), les classes d'équivalence sont définies comme suit (figure V.13.).

Etats i	θ_i	Nouveaux états i'	θ'_i
(1) 1	8/5	1 (1') = 1	8/5
(2) 21	10/5	1231 (2') = 1321	9/5
(3) 121	8/5	12 (3') = 21	10/5
(4) 321	11/5	123 (4') = 321	11/5
(5) 2121	10/5		
(6) 3121	10/5		
(7) 1321	9/5		
(8) 2321	8/5		
(9) 4321	11/5		

FIGURE V.13.

En effet, le fonctionnement du système est tel que les seuls cas que l'on peut rencontrer sont ceux donnés par la figure V.14.

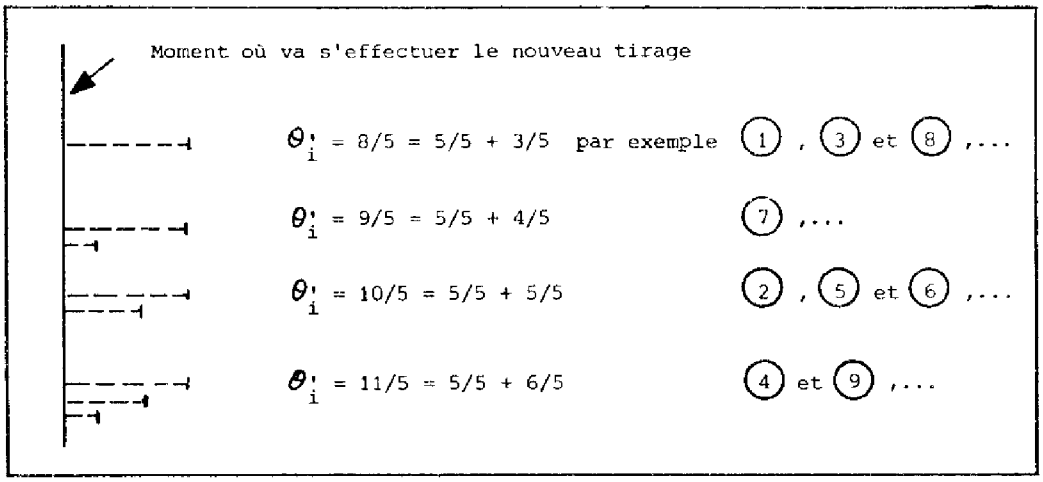


FIGURE V.14.

Bien que la dénomination des états se retrouve dans l'ancienne chaîne puisqu'ils constituent un des éléments d'une classe d'équivalence, il est essentiel de noter que leur signification n'est pas la même.

Par exemple, l'état (3') de la nouvelle chaîne sur la figure V.13. regroupe tous les états de la chaîne fonctionnelle caractérisés par un temps de transfert $\theta'_3 = 2 \times \tau$, c'est-à-dire, les états 2, 5 et 6.

Pour l'exemple choisi, on est conduit à la matrice des probabilités de transition suivante :

	P_1	P_2	P_3	P_4	P_5
1'	1'	3'	3'	3'	3'
2'	1'	1'	4'	4'	4'
3'	2'	1'	1'	4'	4'
4'	1'	3'	2'	3'	5'

$$\Rightarrow \begin{pmatrix} (1) \\ (1321) \\ (21) \\ (321) \end{pmatrix} = \begin{pmatrix} 1/5 & 1/5 & 2/5 & 2/5 \\ 0 & 1/5 & 0 & 1/5 \\ 4/5 & 3/5 & 0 & 0 \\ 0 & 0 & 3/5 & 2/5 \end{pmatrix} \begin{pmatrix} 1 \\ 1321 \\ 21 \\ 321 \end{pmatrix}$$

La résolution de ce système donne les résultats exacts suivants:

P(1) = 0,3208 $\quad \theta_{(1)} = 8/5$

P(1321) = 0,0755 $\quad \theta_{(1321)} = 9/5$

P(21) = 0,3019 $\quad \theta_{(21)} = 10/5$

P(321) = 0,3019 $\quad \theta_{(321)} = 11/5$

$$T_{moy} = 1.917 \times \tau$$

On remarque, par exemple, que si la machine se trouve dans l'état 1321 et que l'instruction suivante sélectionne l'opérateur (2) l'état d'arrivée est différent selon que l'on utilise le premier ou le second modèle fonctionnel :

1321 $\xrightarrow{P_2}$ 1321 dans le premier modèle fonctionnel,

$\xrightarrow{P_2}$ 21 dans le nouveau modèle..

Le nouveau modèle étant obtenu en désignant les états pour leurs caractéristiques temporelles et non fonctionnelles sera appelé un modèle structurel.

C'est ce type de modèle que nous développons par la suite, en partant directement d'une description du comportement du système matériel.

V.2. *MODELE STRUCTUREL*

V.2.1. *Représentation du fonctionnement d'un opérateur*

Chacun des états de la structure peut se trouver dans l'un des quatre modes de fonctionnement suivants :

- mode repos : l'opérateur est disponible à chaque cycle de la machine,
- mode liaison : l'opérateur effectue un accès au bus, parce qu'il était disponible lorsqu'il a été appelé,
- mode autonome : l'opérateur effectue, d'une manière autonome, un traitement pendant un temps T donné,
- mode conflit : l'opérateur termine l'exécution du traitement en cours pendant que la nouvelle instruction qui nécessite cet opérateur attend la libération de celui-ci pour être initialisée.

L'opérateur évolue entre ces différents modes de la manière indiquée sur la figure V.15.

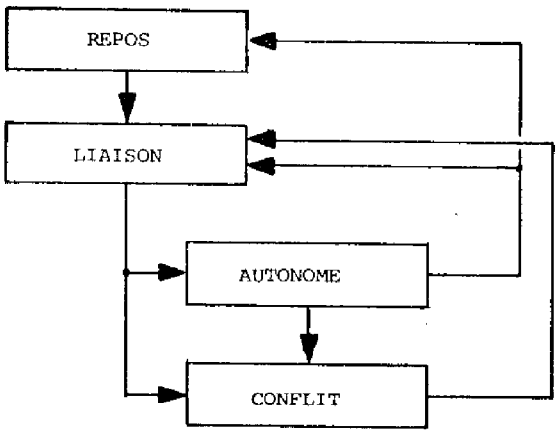


FIGURE V.15.

Nous avons choisi de représenter l'évolution que peut suivre un opérateur par un modèle dans lequel tous les états durent le temps unitaire τ car, si l'on veut indiquer dans quelle phase du traitement se trouve un opérateur par le nom du seul état dans lequel il se trouve à l'instant d'observation (hypothèse markovienne), il est nécessaire de décomposer les états caractérisant les modes autonome et de conflit en plusieurs états.

On est ainsi conduit au schéma de représentation de la figure V.16. (temps de traitement $T = K \times \tau$).

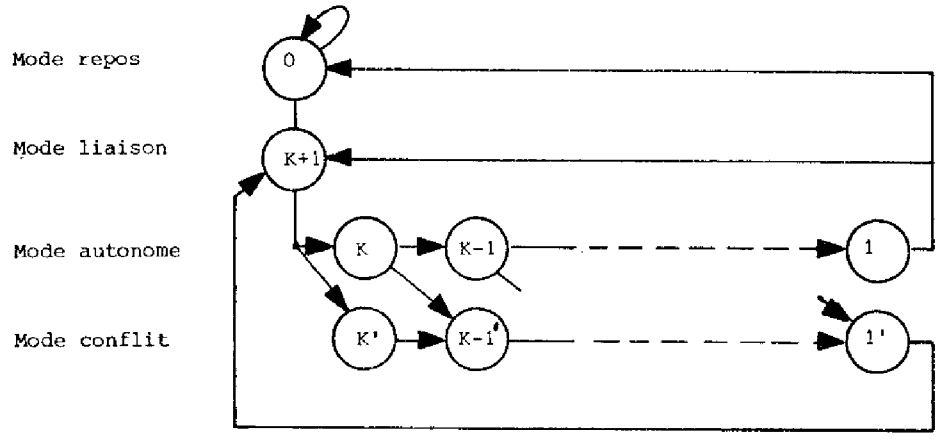


FIGURE V.16.

Les transitions entre les états procèdent de trois notions de nature différente :

- le temps, introduit directement dans le modèle par le choix des états : il y a nécessairement une transition à la fin de chaque état qui dure un temps τ ,
- les données fonctionnelles rapportées au niveau de chaque opérateur par la probabilité conditionnelle qu'il soit sélectionné lorsqu'une instruction est initialisée,
- l'influence du reste du système dont dépend l'initialisation de la nouvelle instruction.

V.2.2. *Représentation du fonctionnement du système*

V.2.2.1. *Présentation du premier modèle structurel*

Les interactions entre les opérateurs de la structure peuvent être exprimées sur un modèle où l'état de la structure est entièrement déterminé, à chaque cycle machine, par un N uplé ; celui-ci est formé par la juxtaposition des états dans lesquels se trouvent les N opérateurs de la structure, à l'instant t (multiple de τ) de l'observation du système.

Un état du système est donc représenté par le vecteur d'état suivant :

$$\eta t = \{ \eta_1(t) , \eta_2(t) , \text{----} , \eta_N(t) \}$$

et son fonctionnement en régime permanent est décrit par une chaîne de MARKOV définie par la matrice des probabilités des transitions entre deux états successifs de la machine $\eta(t)$ et $\eta(t+1)$.

Or, connaissant la valeur des états dans lesquels se trouve chaque opérateur de la structure, on sait exprimer cette matrice des probabilités de transition, en fonction de probabilités constantes p_i de référence à chaque opérateur i.

V.2.2.2. *Nombre d'états du modèle*

Nous nous limitons au calcul d'un majorant du nombre d'états du modèle, cette grandeur n'étant pas nécessaire pour calculer le temps moyen d'exécution d'une instruction, mais seulement pour évaluer la complexité de l'exploitation du modèle.

Tout état comprenant nécessairement :

- un opérateur et un seul en mode de liaison ou de conflit,
- l'ensemble des autres opérateurs dans le mode repos ou le mode autonome,

un majorant du nombre des états est donné par l'expression :

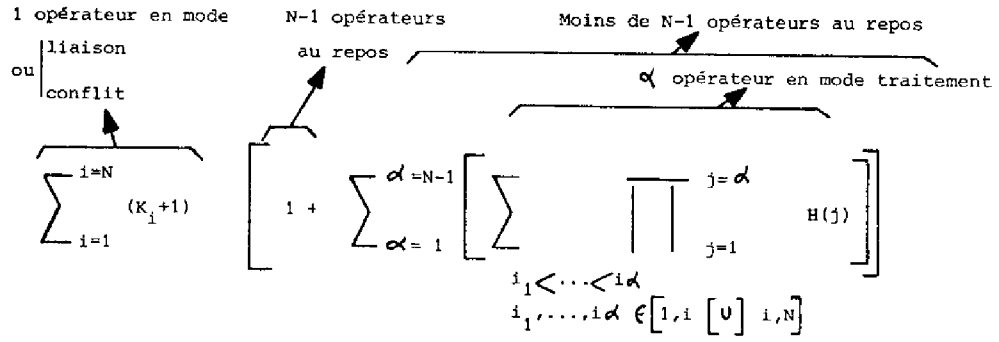
$$Maj\,(NB) = N \prod_{i=1}^{i=N} (i+1)$$

En effet, il y a N positions possibles pour l'opérateur étant dans l'état de liaison où les (K_i) états de conflit , les N-1 autres opérateurs se trouvant dans l'état de repos où les (K_i) états du mode autonome, soit :

$$\sum_{i=1}^{i=N} (K_i+1) \times \prod_{\substack{j\neq i \\ j\in[1,N]}} (K_j+1) = N \prod_{(i\in 1,N)} (K_i+1)$$

Si on veut obtenir un majorant plus fin du nombre des états possibles, il faut tenir compte du fait que, une seule instruction pouvant être initialisée à la fois, deux opérateurs ne peuvent pas se trouver dans la même phase de leur traitement respectif.

Si on positionne les opérateurs dans l'écriture des états de la structure par ordre croissant de leur temps de traitement, un majorant assez proche du nombre d'états possibles est donné par l'expression :

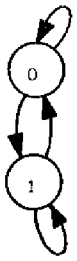


V.2.2.3. *Exemple d'application*

On développe ce modèle sur le cas simple d'une structure à deux opérateurs :

O_{p1} : $T_1 = 0$ $\qquad$ P_1 = probabilité de référence à O_{p1}

O_{p_2} : $T_2 = 2 \times \mathcal{C}$ $\qquad$ P_2 = probabilité de référence à O_{p2}



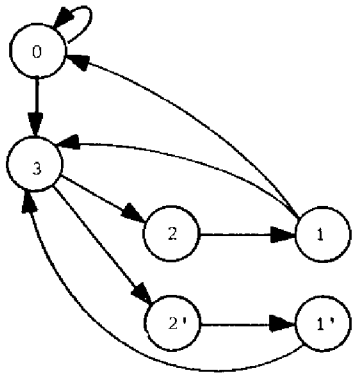


FIGURE V.17.

Les différents états possibles du système sont les suivants :

(O_{p1}, O_{p2})

0 , 3

0 , 2'

0 , 1'

1 , 0

1 , 2

1 , 1

Il y a six états ce qui, dans ce cas très simple, correspond exactement aux deux majorants calculés par les formules du paragraphe précédent.

La matrice des probabilités de transition est obtenue de la manière suivante :

$$\mathbb{P} = \begin{array}{c} \\ 0,3 \\ 0,2' \\ 1,0 \\ 1,1' \\ 1,2 \\ 1,1 \end{array} \begin{array}{c} \begin{array}{cccccc} 0,3 & 0,2' & 1,0 & 1,1' & 1,2 & 1,1 \end{array} \\ \begin{bmatrix} 0 & P_2 & 0 & 0 & P_1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ P_2 & 0 & 0 & P_1 & 0 & 0 \\ 0 & 0 & P_2 & 0 & 0 & P_1 \\ P_2 & 0 & 0 & P_1 & 0 & 0 \end{bmatrix} \end{array}$$

nouveaux états

anciens états

La résolution de ce système donne le vecteur des probabilités limites suivant :

$$\alpha_1 = \frac{P_2}{3P_2 + P_1^3}$$

$$\alpha_2 = \frac{P_2^2}{3P_2 + P_1^3}$$

$$\alpha_3 = \frac{P_2^2(1 + P_1)}{3P_2 + P_1^3}$$

$$\alpha_4 = \frac{P_1^3}{3P_2 + P_1^3}$$

$$\alpha_5 = \frac{P_1 P_2}{3P_2 + P_1^3}$$

$$\alpha_6 = \frac{P_1^2 P_2}{3P_2 + P_1^3}$$

Par suite :

$$\theta_{moy} = \frac{1}{\alpha_1 + \alpha_4 + \alpha_5 + \alpha_6} = 3\ (1 - P_1) + P_1^3$$

V.2.3. *Réduction du nombre d'états par changement du mode de description*

V.2.3.1. *Présentation d'un second modèle structurel*

La transformation que l'on exécute, au niveau de la description du fonctionnement du système, consiste à n'observer l'état du système qu'aux instants où il y a initialisation d'une nouvelle instruction. Cette transformation qui correspond au passage d'une chaîne de MARKOV à une chaîne de MARKOV plus réduite mais avec valeurs de transition, comme nous l'avons présenté au chapitre IV, est rendue possible car on connaît tous les temps des transitions apparaissant sur le nouveau schéma de représentation d'un opérateur, sur lequel on a supprimé (K + 1) états.

Pour T = 3 x τ par exemple, on obtient le graphe de la figure V.18.b. déduit du graphe de la figure V.18.a.

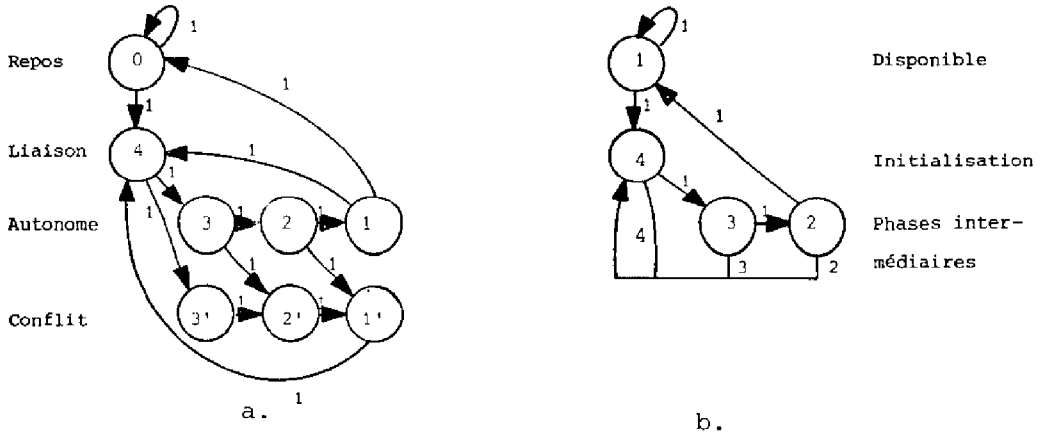


FIGURE V.18.

Cette transformation consiste à dépouiller les états de la notion de temps, les transitions ne traduisant plus que :

- les contraintes fonctionnelles,
- les interactions avec le reste du système.

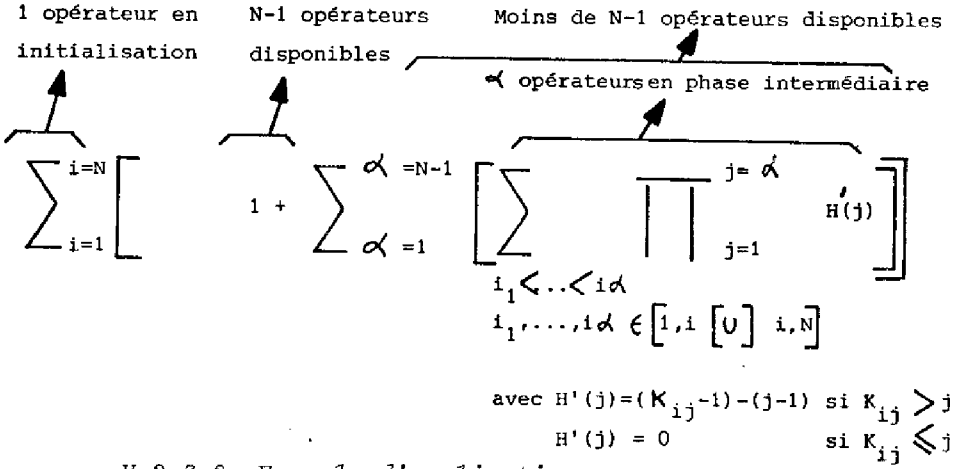
La notion de temps est reportée au niveau de durées affectées aux transitions.

V.2.3.2. *Nombre d'états du second modèle*

Dans cette nouvelle description du système, le nombre d'états est majoré par :

$$\sum_{i=1}^{i=N} \left(\prod_{j \neq i} K'_j \right) \qquad \text{avec} \qquad K'_j = \min \left[1, K_j \right]$$

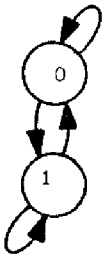
En tenant compte de manière plus précise des particularités du fonctionnement du système, on est conduit à un majorant très proche donné par l'expression suivante :



V.2.3.2. *Exemple d'application*

On reprend le même exemple qu'au paragraphe V.2.2.3.

La représentation des deux opérateurs donnée sur la figure V.19. devient celle de la figure V.20. (On ne gagne rien à regrouper les états (0) et (1) du premier opérateur).



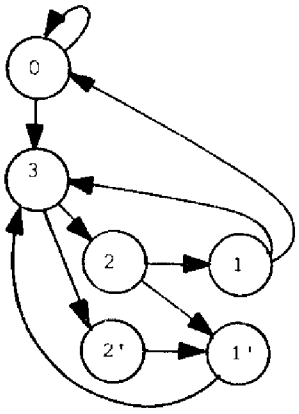
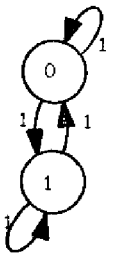


FIGURE V.19.



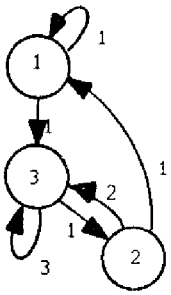


FIGURE V.20.

Le nombre d'états est 3 (0,3 ; 1,1 ; 1,2) ; il correspond aussi, dans ce cas, aux majorants calculés avec les formules ci-dessus.

La matrice $\mathbb{P}$ des probabilités de transition est déterminée à partir du tableau de la figure V.21.

Etat \ Opérateur appelé		O_{p1}	O_{p2}	Temps moyen
(1)	0,3	$(3)_1$	$(1)_3$	$\theta_1 = (P_1 x 1 + P_2 x 3) x \gamma$
(2)	1,1	$(2)_1$	$(1)_1$	$\theta_2 = (P_1 x 1 + P_2 x 1) x \gamma$
(3)	1,2	$(2)_1$	$(1)_2$	$\theta_3 = (P_1 x 1 + P_2 x 2) x \gamma$

FIGURE V.21.

$$\mathbb{P} = \begin{bmatrix} P_2 & 0 & P_1 \\ P_2 & P_1 & 0 \\ P_2 & P_1 & 0 \end{bmatrix}$$

Le vecteur des probabilités limites et les temps de transition conduisent à la même expression pour le temps moyen T_{moy}.

$P(0,3) = \alpha_1 = P_2$ $\quad \theta_1 = (P_1 \times 1 + P_2 \times 3) \times \gamma$

$P(1,1) = \alpha_2 = P_1^2$ $\quad \theta_2 = 1 \times \gamma$

$P(1,2) = \alpha_3 = P_1 P_2$ $\quad \theta_3 = (P_1 \times 1 + P_2 \times 2) \times \gamma$

$$T_{moy} = 3(1-P_1) + P_1^3$$

V.2.4. *Hypothèse de la symétrie de la structure*

En raison du très grand nombre d'états auquel on parvient pour la description du fonctionnement de la structure très générale que nous venons de présenter, nous ne donnerons de résultats numériques que pour les structures que nous avons qualifiées de symétrique (symétrie matérielle et symétrie fonctionnelle).

V.2.4.1. *Nombre d'états résultant*

Cette symétrie de la structure permet une réduction considérable du nombre d'états que peut prendre la machine.

Par exemple, à la suite de l'hypothèse de la symétrie matérielle (tous les opérateurs ont le même temps de traitement), on est conduit au nombre d'états suivant (pour $N > K$) :

- Chaîne de MARKOV sans valeur de transition :

$$Maj(NB)_1 = N \times (K+1) \left\{ \sum_{\alpha=0}^{\alpha=K} C_{N-1}^{\alpha} \quad A_K^{\alpha} \right\}$$

- Chaîne de MARKOV avec valeurs de transition :

$$Maj(NB)_2 = N \times \left\{ \sum_{\alpha=0}^{\alpha=K-1} C_{N-1}^{\alpha} \quad A_{K-1}^{\alpha} \right\}$$

Si l'on admet l'hypothèse de la symétrie fonctionnelle (équiprobabilité d'appel) en plus de la symétrie matérielle, l'ordre dans lequel les opérateurs sont cités dans la dénomination des états du système et le nombre des états devient :

$$N \rightsquigarrow 1 \qquad A_K^{\alpha} \rightsquigarrow \frac{A_K^{\alpha}}{(K-\alpha)!} = C_K^{\alpha}$$

$$C_{N-1}^{\alpha} \rightsquigarrow 1$$

Suppression des considérations de position. Suppression des considérations de permutation

- Chaîne de MARKOV sans valeur de transition :

$$\text{Maj (NB)}_1 = (K+1) \sum_{\alpha=0}^{\alpha=K} C_K^{\alpha} = (K+1)\ 2^K$$

- Chaîne de MARKOV avec valeurs de transition :

$$\text{Maj (NB)}_2 = \sum_{\alpha=0}^{\alpha=K-1} C_K^{\alpha} = 2^{K-1} .$$

V.2.4.2. *Exemple d'application*

Nous développons le second modèle de représentation structurelle dans l'exemple des 5 opérateurs qui ont un temps de traitement de 3τ et une probabilité de référence $\frac{1}{5}$ à chaque pas de programme.

Le nombre des états représentatifs de la structure, de 32 pour le premier modèle structurel, passe à 4 pour le second modèle.

Ils sont numérotés :

	1	2	3	4	5
(1) =	4	1	1	1	1
(2) =	4	2	1	1	1
(3) =	4	3	1	1	1
(4) =	4	3	2	1	1

Les nombres apparaissant dans la désignation des états du système correspondent au temps que mettront les opérateurs pour être disponibles.

	P_1	P_2	P_3	P_4	P_5	Temps moyens
(1) 4 1 1 1 1	$(1)_4$	$(3)_1$	$(3)_1$	$(3)_1$	$(3)_1$	$\theta_1 = \frac{1}{5}\,4 + \frac{4}{5} \times 1 = 1,6$
(2) 4 2 1 1 1	$(1)_4$	$(2)_2$	$(3)_1$	$(3)_1$	$(3)_1$	$\theta_2 = \frac{1}{5}\,4 + \frac{1}{5}\,2 + \frac{3}{5}\,1 = 1,8$
(3) 4 3 1 1 1	$(1)_4$	$(1)_3$	$(4)_1$	$(4)_1$	$(4)_1$	$\theta_3 = \frac{1}{5}\,4 + \frac{1}{5}\,3 + \frac{3}{5}\,1 = 2,0$
(4) 4 3 2 1 1	$(1)_4$	$(1)_3$	$(2)_2$	$(4)_1$	$(4)_1$	$\theta_4 = \frac{1}{5}\,4 + \frac{1}{5}\,2 + \frac{1}{5}\,3 + \frac{2}{5}\,1 = 2,2$

FIGURE V.22.

$$\mathbb{P} = \frac{1}{5}\begin{bmatrix} 1 & 0 & 4 & 0 \\ 1 & 1 & 3 & 0 \\ 2 & 0 & 0 & 3 \\ 2 & 1 & 0 & 2 \end{bmatrix}$$

La matrice $\mathbb{P}$ des probabilités de transition et les temps de transition sont bien ceux que l'on avait trouvés au paragraphe V.1.3.

Bien que la caractérisation des états provienne de deux regroupements distincts :

- par identité des temps d'exécution dans le premier cas,
- en ne gardant que certains états significatifs d'une description du système par les états des opérateurs,

les deux modèles sont identiques comme cela est montré sur la figure V.23.

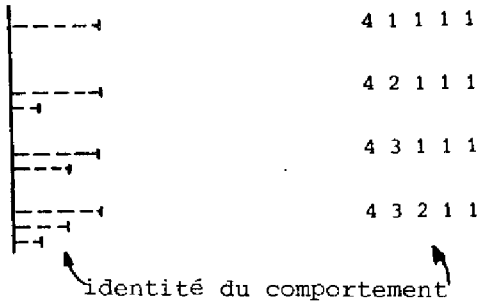


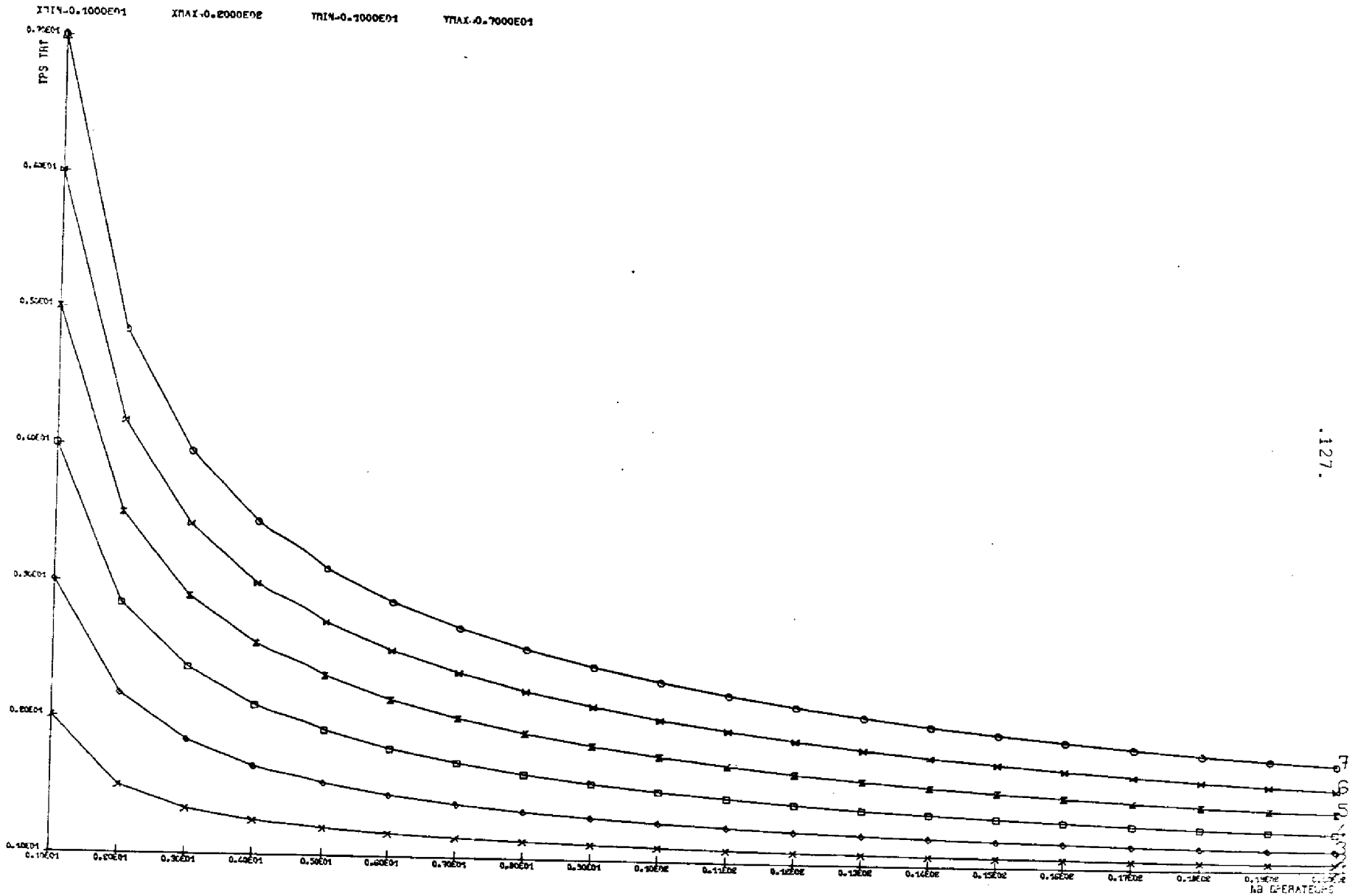
FIGURE V.23.

V.3. *VARIATION DES ELEMENTS MATERIELS D'UN SYSTEME DANS UN MODELE STRUCTUREL*

V.3.1. *Temps moyen d'exécution d'une instruction pour différentes valeurs de N et de T*

Nous faisons apparaître sur le graphique de la figure V.24. les différentes valeurs du temps d'exécution d'une instruction dans le cas de structures symétriques, lorsque nous faisons varier la valeur du nombre d'opérateurs, citéeen abscisse, entre 1 et 20 et du temps de traitement, constant sur chaque courbe du faisceau, entre 0 et 7.

Le point mis en évidence sur le faisceau de courbes de la figure V.24. correspond au cas d'une structure de 5 opérateurs ayant un temps de traitement de 3 x τ , que l'on a développé à titre d'exemple dans les paragraphes précédents.



PROBABILITE D APPEL
DES OPERATEURS

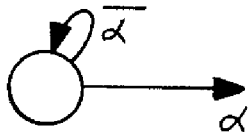
FIGURE V.24.

V.3.2. *Distribution exponentielle du temps de service*

Soit (α) la probabilité qu'un opérateur qui se trouve dans un état donné de durée (τ) quitte cet état à la fin de l'intervalle de temps considéré,

soit (β) = $\bar{\alpha}$ = 1- α la probabilité qu'il soit encore dans le même état à l'intervalle de temps suivant.

Un temps de service exponentiel, de valeur moyenne $\frac{1}{\alpha}$ est représenté de la manière suivante :



Aussi le fonctionnement d'un opérateur peut-il être décrit par le schéma de la figure V.25. dans lequel les termes i_j sont les probabilités que l'opérateur i soit appelé alors qu'il se trouve dans l'état j.

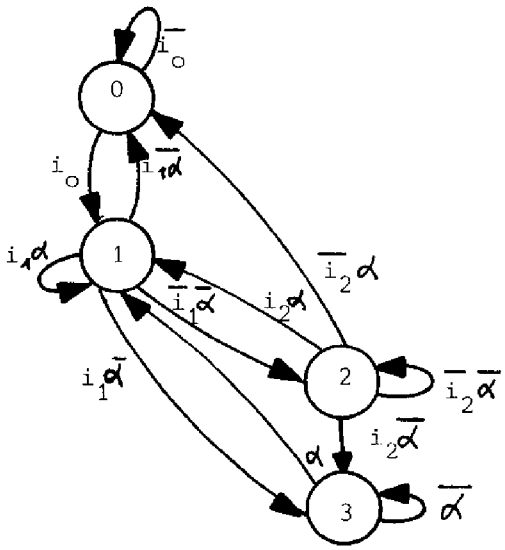


FIGURE V.25.

V.3.2.1. *Résolution avec la chaîne sans valeur de transition*

a. Nombre d'états

Une structure de N opérateurs est représentée par $\boxed{N \times 2^N}$ états et ce nombre est réduit à $\boxed{2 \times N}$ si la structure est symétrique.

b. Construction de la matrice des probabilités de transition

Un état du système est caractérisé par :

- la présence dans son énoncé d'un état (1) ou d'un état (3),
- le nombre (A) d'opérateurs parmi les N-1 restants qui sont en cours de traitement.

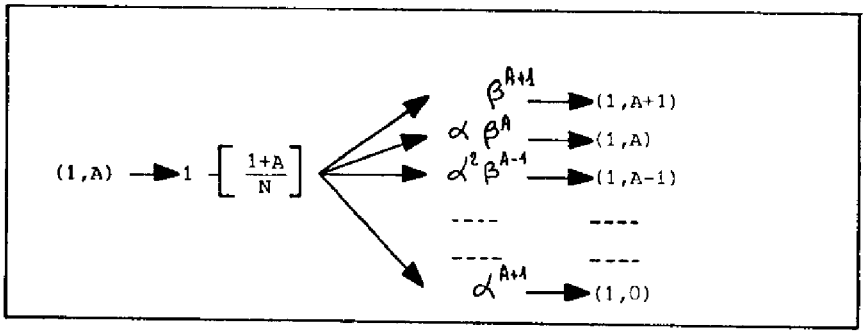
Considérons l'état que l'on peut représenter par le doublet (1,A) signifiant que : 1 opérateur se trouve en mode de communication,

.A opérateurs se trouvent en mode autonome.

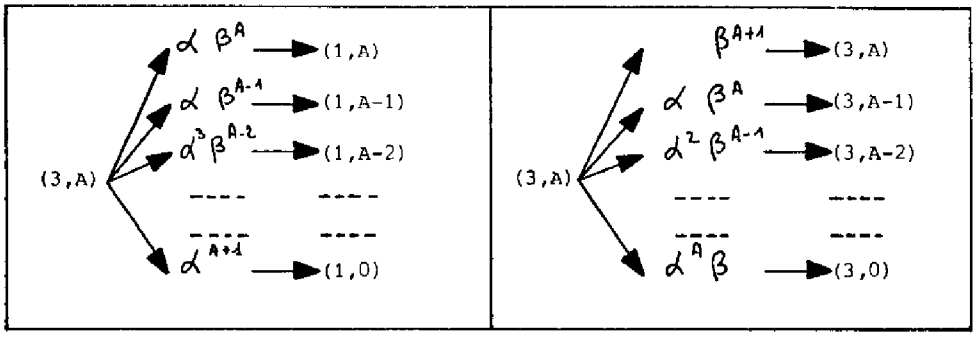
- Si l'un de ces (1+A) opérateurs est appelé, c'est-à-dire, avec une probabilité de $\frac{A+1}{N}$, on se trouve dans l'un des états suivants :

Nouvel état	Probabilité conditionnelle d'y arriver	Nouvel état	Probabilité conditionnelle d'y arriver
1,A	$\alpha^1 \beta^A$	3,A	β^{A+1}
1,A-1	$\alpha^2 \beta^{A-1}$	3,A-1	$\alpha \beta^A$
1,A-2	$\alpha^3 \beta^{A-2}$	3,A-2	$\alpha \beta^{A-1}$
----	----	----	----
1,0	α^{A+1}	3,0	$\alpha^A \beta$
si l'opérateur appelé se libère		si l'opérateur appelé reste occupé	

- Si l'un des N-(1+A) opérateurs restants est appelé avec une probabilité de $1-\frac{(1+A)}{N}$, les transitions possibles sont différentes :



En partant de l'état représenté par (3,A), on trouve, avec des probabilités absolues (il n'y a pas d'émission de nouvelle instruction) :



Si l'on numérote d'abord les états (1,A) pour les valeurs croissantes de A, puis les états (3,A) aussi pour des valeurs croissantes de A, le terme générique (I,3) de la matrice des probabilités de transition est donnée par :

	(1,A)	(3,A)
(1,A)	I	II
(I,3)	III	IV

Dans le quadrant I

$$(I,3) = \frac{1}{N}\left[(N-I)\,C_{I}^{I-J+1} + I\,C_{I-1}^{J-1}\right]\alpha^{I-J+1}\,\beta^{J-1}$$

Dans le quadrant II :

$$(I,3) = \frac{1}{N}\left[I\,C_{I-1}^{J-N-1}\right]\alpha^{I-J+N}\,\beta^{J-N}$$

Dans le quadrant III :

$$(I,3) = C_{I-N-1}^{J-1}\,\alpha^{I-N-J+1}\,\beta^{J-1}$$

Dans le quadrant IV :

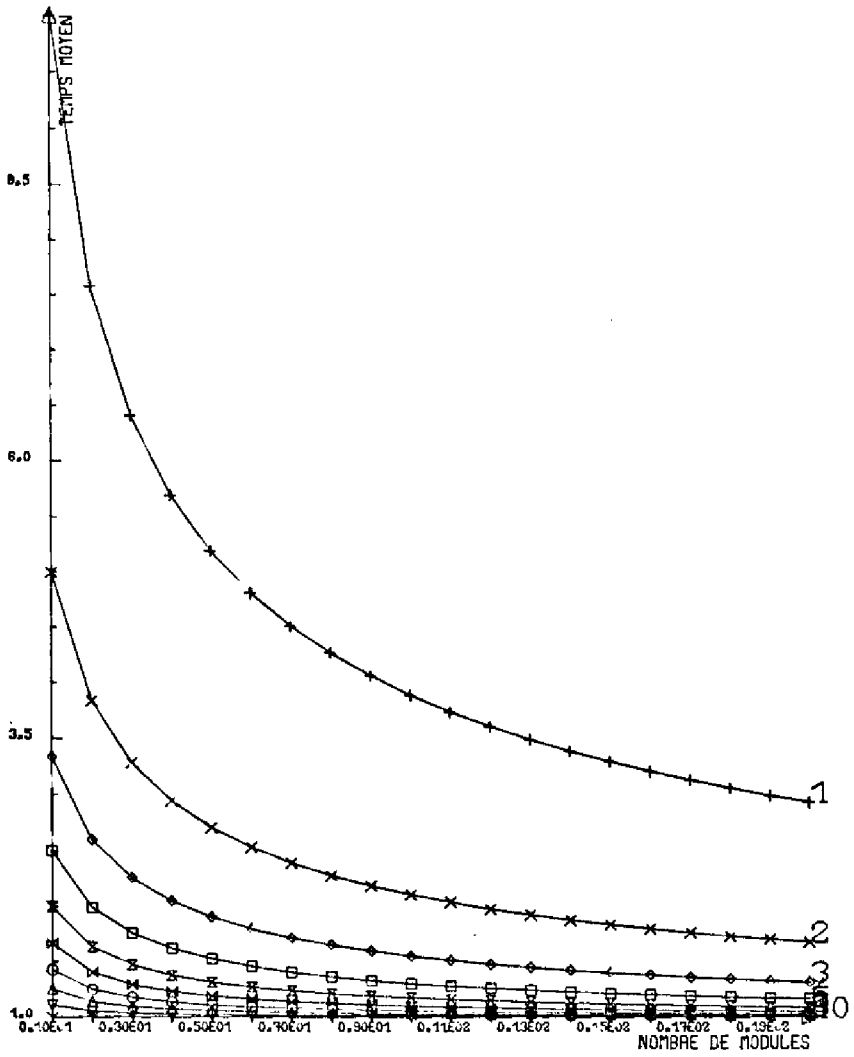
$$(I,3) = C_{I-N-I}^{J-N-1}\,\alpha^{I-J}\,\beta^{J-N}$$

c. A titre d'exemple, la matrice de transition pour 3 opérateurs est donnée ci-dessous :

	100	120	122	300	320	322
100	α	$\frac{2}{3}\beta$	0	$\frac{1}{3}\beta$	0	0
120	α^2	$\frac{4}{3}\alpha\beta$	$\frac{1}{3}\beta^2$	$\frac{2}{3}\alpha\beta$	$\frac{2}{3}\beta^2$	0
122	α^3	$\frac{6}{3}\alpha^2\beta$	$\alpha\beta^2$	$\alpha^2\beta$	$2\alpha\beta^2$	β^3
300	α	0	0	β	0	0
320	α^2	$\alpha\beta$	0	$\alpha\beta$	β^2	0
322	α^3	$2\alpha^2\beta$	$\alpha\beta^2$	$\alpha^2\beta$	$\alpha\beta^2$	β^3

d. Résultats

Le développement de ces calculs, pour des valeurs de (N) variant de 1 à 20 et de (α) comprises entre 0,1 et 0,9 par incréments de 0,1, donne le faisceau de courbes de la figure V.26.



PROBABILITE D'APPEL CONSTANTE
TEMPS DE SERVICE EXPONENTIEL

FIGURE V.26.

V.3.2.2. Résolution utilisant une chaîne de MARKOV avec valeurs de transition

Une particularité du temps de service exponentiel est que le temps moyen d'attente, lorsqu'un opérateur est appelé pendant qu'il est en cours de traitement, est égal à $\frac{1}{\alpha}$ (avec α = probabilité de quitter l'état).

Le fonctionnement d'un opérateur peut être représenté par le schéma de la figure V.27. :

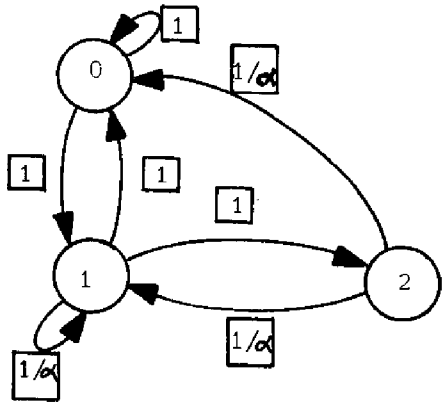


FIGURE V.27.

ou même, puisque le temps de traitement peut être nul par le schéma de la figure V.28.

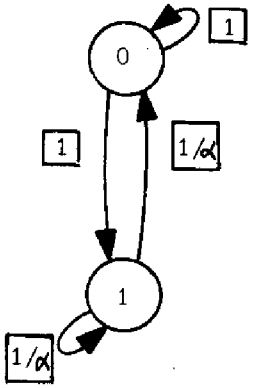


FIGURE V.28.

Nous retenons la seconde représentation bien que les deux descriptions du fonctionnement d'un opérateur conduisent au même nombre d'états du système.

Les temps moyens de sortie des états étant faciles à déterminer (1 si l'opérateur appelé est disponible, $1/\alpha$ s'il est occupé), on peut facilement construire la matrice Π des temps de transition, propre à la troisième méthode présentée au chapitre IV.3. Pour calculer la matrice des probabilités de transition de cette chaîne de MARKOV avec valeursde transition,

on peut soit développer les calculs de la convolution directement, soit utiliser le produit matriciel donnant $\overline{\mathbb{P}}$.

Les résultats présentés sur la figure V.26. ont cependant été calculés avec la première méthode.

CHAPITRE VI

EVALUATION DES PERFORMANCES DU SYSTEME MULTIOPERATEUR A.S.M.O.D.E.E. 02

Nous étudions dans ce dernier chapitre, les performances du système de commande que nous avons présenté dans la première partie.

L'évaluation des performances d'A.S.M.O.D.E.E. 02 est effectuée en deux étapes, conformément à la démarche présentée au chapitre IV :

- nous établissons, dans une première étape, le débit moyen des instructions pour un programme d'interruption d'un type donné ; nous utilisons pour ce calcul un modèle analogue à celui développé dans le chapitre V (paragraphe V.2.3.1.),
- nous nous limitons pour la seconde étape à la première phase de l'étude des performances globales d'A.S.M.O.D.E.E. 02 (au niveau de la mission), c'est-à-dire une description de l'enchaînement des programmes d'interruption.

La majeure partie de ce chapitre concerne donc la première étape et se présente sous la forme d'une succession de modèles correspondant à :

- différentes caractérisations des programmes,
- différents degrés d'évolution des modules.

- Le premier modèle correspond au cas simple des structures symétriques pour lesquelles la probabilité de référence et le temps de traitement propres à chaque opérateur sont identiques et constants. Cette probabilité de référence est prise égale au rapport entre le nombre de références aux différents opérateurs et le nombre total d'instructions du programme.

- Les modèles suivants correspondent à une description plus proche du fonctionnement réel du système de commande :

- le second modèle introduit la possibilité de deux types d'opérateurs qui diffèrent par la durée de leur traitement,
- le troisième modèle tient compte d'une caractéristique fonctionnelle des opérateurs. Ceux-ci ont en effet un comportement différent selon qu'ils ont été nommés en source ou en destination lors de leur dernier appel.

- Enfin, l'introduction d'un phénomène de localité permet de s'éloigner de l'hypothèse de l'indépendance des pas du programme et conduit à une caractérisation plus vraisemblable du comportement du programme :

  - lorsqu'un programme est écrit par une personne ignorant les particularités du fonctionnement d'A.S.M.O.D.E.E. 02, le phénomène de localité se traduit par l'influence de la dernière fonction nommée sur la nouvelle instruction,

  - lorsque l'écriture du programme a été reprise de manière automatique dans le but de minimiser son temps global d'exécution, le phénomène se traduit par une plus forte probabilité de référence aux opérateurs disponibles.

Ces deux cas font l'objet des quatrième et cinquième modèles développés.

VI.1. MODELE APPROCHE D'A.S.M.O.D.E.E. 02

VI.1.1. Hypothèses simplificatrices

Le modèle de description d'A.S.M.O.D.E.E. 02 qui représente le déroulement des instructions est effectué à partir des hypothèses suivantes :

- la succession des échanges élémentaires nécessaires au transfert d'une donnée constitue un cycle machine qui dure un temps τ constant, pris comme unité de temps du modèle,

- le test des informations de verrouillage s'effectue à la fin de chaque cycle, ce qui permet d'exprimer la durée du traitement à l'intérieur de chaque opérateur sous la forme d'un multiple du temps unitaire τ,

- lorsqu'un opérateur nommé par une instruction est en cours d'exécution d'un traitement, le transfert de la nouvelle donnée est répété par cycles entiers successifs, jusqu'à ce que l'on observe, à la fin d'un cycle, que les deux opérateurs qui participent au transfert sont libres ; le transfert suivant peut alors être validé,

- le cycle reste inchangé lorsqu'il y a demande d'interruption,

- seules les instructions affectant les données sont prises en compte, laissant ainsi de côté les instructions affectant les calculs d'adresse ; celles-ci, évaluées entre 20 et 35 % en fonction des applications [RAV.73]-[TAY.72]-[FLY.74], ne se prêtent pas au parallélisme des traitements et entraînent nécessairement une interruption du flot des instructions,
- seul le mode d'adressage direct est employé.

Nous comparons sur la figure VI.1. le cycle réel d'A.S.M.O.D.E.E. 02 (figure VI.1.a.) et le cycle obtenu à la suite des hypothèses simplificatrices que nous venons d'énoncer (figure VI.1.b.).

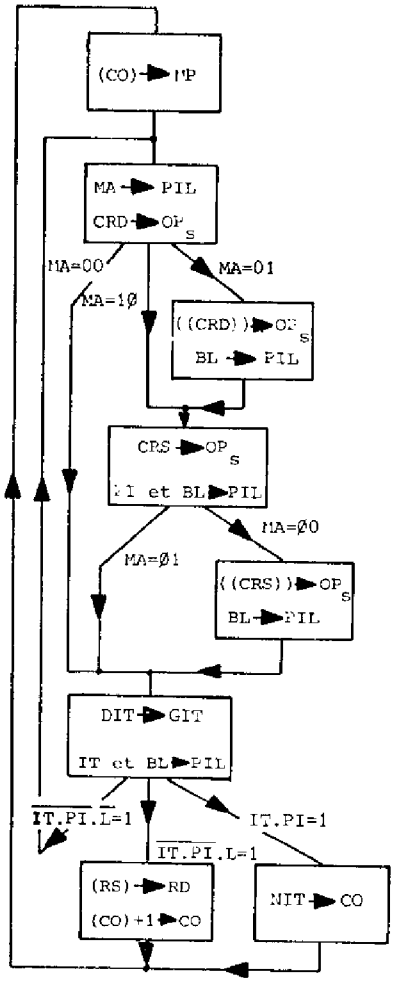


FIGURE VI.1.a.

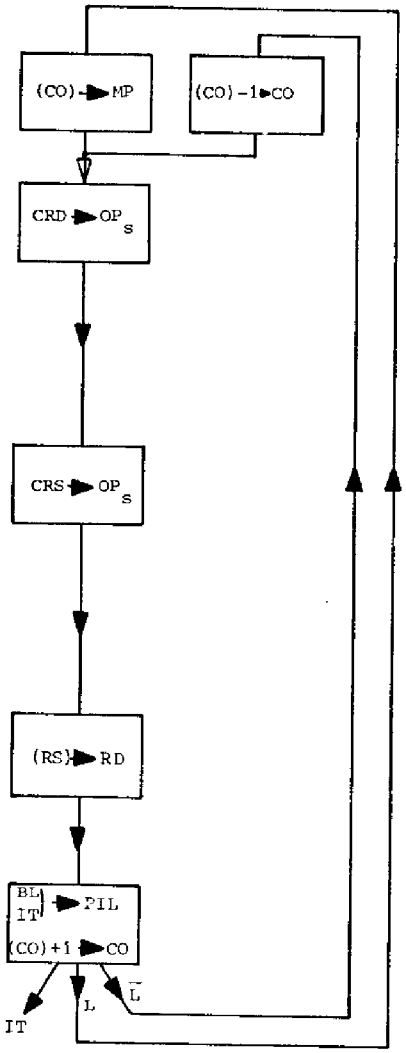


FIGURE VI.1.b.

VI.1.2. Description du fonctionnement d'un opérateur

La description de l'évolution d'un opérateur par son état pendant le dernier intervalle τ conduit à distinguer deux états de transfert selon que l'opérateur est appelé en destination ou en source, car le transfert peut être suivi d'un traitement dans le premier cas, alors que dans l'autre cas, l'opérateur est immédiatement disponible pour une autre utilisation. Comme un opérateur appelé à la fois en source et en destination par une même instruction se comporte dans le temps de la même manière qu'un opérateur appelé seulement en destination, nous ne considèrerons que deux types d'évolution possibles pour chaque opérateur ce qui apparaît sur le diagramme des états de la figure VI.2.

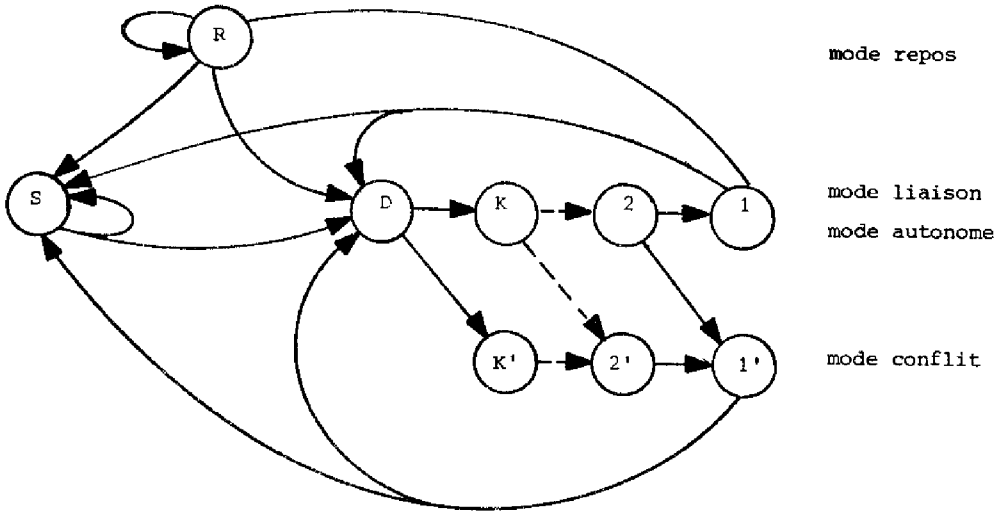


FIGURE VI.2.

Si l'on dépouille les états de leur signification fonctionnelle pour ne garder que la notion du temps nécessaire pour que l'opérateur puisse participer à un nouveau transfert, on est conduit au graphe avec valeurs de transition de la figure VI.3. dans lequel le nouvel état (1) regroupe les états (R), (S), (1) et (1') de la représentation de la figure VI.2.

La désignation de chacun des états correspond au nombre d'intervalles τ que doit attendre l'opérateur lorsqu'il est appelé par une nouvelle instruction.

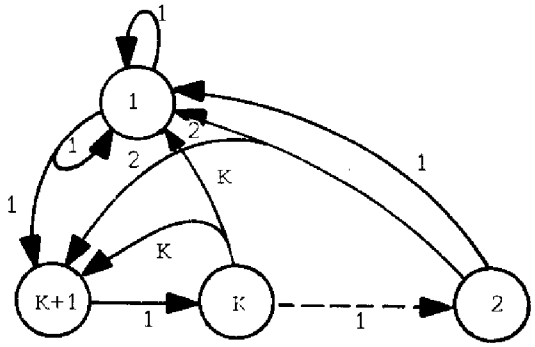


FIGURE VI.3.

VI.1.3. Description du fonctionnement du système

Le fonctionnement du système est décrit par une chaîne de MARKOV avec valeurs de transition dont les états correspondent à la juxtaposition des états dans lesquels se trouvent les opérateurs de la structure.

En représentant le fonctionnement de chaque opérateur de la structure conformément au diagramme de la figure VI.3., le nombre d'états globaux que peut prendre la structure est le même que dans le modèle du chapitre V où un seul opérateur est appelé par instruction, soit, pour une structure symétrique :

$$NB = 2^{K-1}$$

Ce nombre d'états, dans lequel K représente le nombre d'intervalles τ que dure le traitement ($T = K\,\tau$), est en fait un majorant qui n'est atteint que pour un nombre d'opérateurs N supérieur à K.

L'hypothèse générale de l'indépendance des instructions en fonction des différents pas du programme doit être précisée :

les tirages des opérateurs source et destination sont indépendants mais correspondent aux mêmes probabilités de référence, constantes dans le temps.

Soient :

- $P_{ij}(N)$ la probabilité que l'instruction de numéro (N) fasse référence à l'opérateur (i) en source et (j) en destination,
- $P_{s,i}(N)$ la probabilité que l'opérateur (i) soit appelé en source,
- $P_{d,j}(N)$ la probabilité que l'opérateur (j) soit appelé en destination.

Ces grandeurs sont liées par les relations :

$$P_{s,i}(N) = P_{d,j}(N) = \frac{1}{N} = c^{te} \qquad \forall i,j,\ N$$

$$P_{ij}(N) = P_{s,i}(N) \times P_{d,j}(N) = \frac{1}{N^2} = c^{te}$$

Le comportement du système est connu lorsque l'on a construit la matrice des probabilités de transition entre états. La méthode de construction de cette matrice est exposée sur l'exemple suivant :

Exemple d'application : structure symétrique Nombre d'opérateurs N = 5

Temps de traitement T = 3 x τ

Le tableau de la figure VI.4. qui fait apparaître le nom du nouvel état et le temps nécessaire à la transition lorsqu'une instruction désigne l'opérateur destination et l'opérateur source, permet de calculer les termes de la matrice des probabilités des transitions et le temps moyen nécessaire pour quitter chacun des états. On utilise pour cela la troisième méthode présentée au paragraphe IV.3.

Dans cette figure :

- les opérateurs sont numérotés de 1 à 5, dans la dénomination de l'état, de la gauche vers la droite,
- la probabilité d'apparition de chacune des instructions (i,j) est égale à $(\frac{1}{N^2})$ soit (0,04).

Op_D / Op_S nommés par l'instruction	Noms des anciens états : 41111	43111	42111	43211
1 1	41111 4	41111 4	41111 4	41111 4
1 2	41111 4	41111 4	41111 4	41111 4
1 3	41111 4	41111 4	41111 4	41111 4
1 4	41111 4	41111 4	41111 4	41111 4
1 5	41111 4	41111 4	41111 4	41111 4
2 1	41111 4	41111 4	41111 4	41111 4
2 2	43111 1	41111 3	42111 2	41111 4
2 3	43111 1	41111 3	42111 2	41111 3
2 4	43111 1	41111 3	42111 2	41111 3
2 5	43111 1	41111 3	42111 2	41111 3
3 1	41111 4	41111 4	41111 4	41111 3
3 2	43111 1	41111 3	42111 2	41111 4
3 3	43111 1	43211 1	43111 1	41111 3
3 4	43111 1	43211 1	43111 1	42111 2
3 5	43111 1	43211 1	43111 1	42111 2
4 1	41111 4	41111 4	41111 4	42111 2
4 2	43111 1	41111 3	42111 2	41111 4
4 3	43111 1	43211 1	43111 1	41111 3
4 4	43111 1	43211 1	43111 1	42111 2
4 5	43111 1	43211 1	43111 1	43211 1
5 1	41111 4	41111 4	41111 4	43211 1
5 2	43111 1	41111 3	42111 2	41111 4
5 3	43111 1	43211 1	43111 1	41111 3
5 4	43111 1	43211 1	43111 1	42111 2
5 5	43111 1	43211 1	43111 1	43211 1

FIGURE VI.4.

La matrice $\mathbb{P}$ des probabilités de transition est :

$$\mathbb{P} = \begin{bmatrix} 0.36 & 0.64 & 0.0 & 0.0 \\ 0.64 & 0.0 & 0.0 & 0.36 \\ 0.36 & 0.36 & 0.28 & 0.0 \\ 0.64 & 0.0 & 0.20 & 0.16 \end{bmatrix}$$

et on a :

Nom de l'état	Probabilité stationnaire	Temps moyen
41111	0.491443	2.079998
43111	0.328607	2.639997
42111	0.039120	2.359998
43211	0.140831	2.839997

TEMPS MOYEN = 2.382 x τ

VI.1.4. Variations du temps moyen d'exécution en fonction de paramètres matériels

La méthode décrite sur l'exemple ci-dessus, appliquée à des structures ayant :

- un nombre N d'opérateurs variant entre 1 et 20,
- un temps de traitement, identique pour tous les opérateurs, variant entre 0 et 6 x τ,

dans le cadre de l'hypothèse ($P_{ij} = \frac{1}{N^2}$), conduit au faisceau de courbes de la figure VI.5.

VI.2. SUPPRESSION DE L'HYPOTHESE DE LA SYMETRIE DE LA STRUCTURE

VI.2.1. Dissymétrie matérielle

VI.2.1.1. Structure à deux pôles

Malgré les techniques de microprogrammation qui permettent l'introduction de fonctions évoluées, certaines phases du programme ne peuvent pas être découpées en modules répétés un nombre de fois suffisant pour justifier la création d'une nouvelle fonction évoluée. Aussi existe-t-il des actions élémentaires qui doivent constituer à elles seules une instruction dont le temps d'exécution est plus petit que celui des fonctions évoluées.

Nous considérons qu'il est intéressant de développer un modèle ayant deux pôles, qui correspond au cas fréquent de structures ayant deux types d'instruction se distinguant par leur temps d'exécution :

- les instructions courtes qui durent un temps inférieur ou égal à la durée du transfert, et qui sont disponibles à la fin du cycle machine suivant, au moment où le bit de verrouillage est susceptible d'être testé ; il s'agit d'instructions comme les additions, l'écriture ou la lecture en mémoire, etc.,
- les instructions longues, qui durent toutes le même temps exprimé sous la forme d'un multiple de τ : instructions de multiplication, division, racine carrée,...

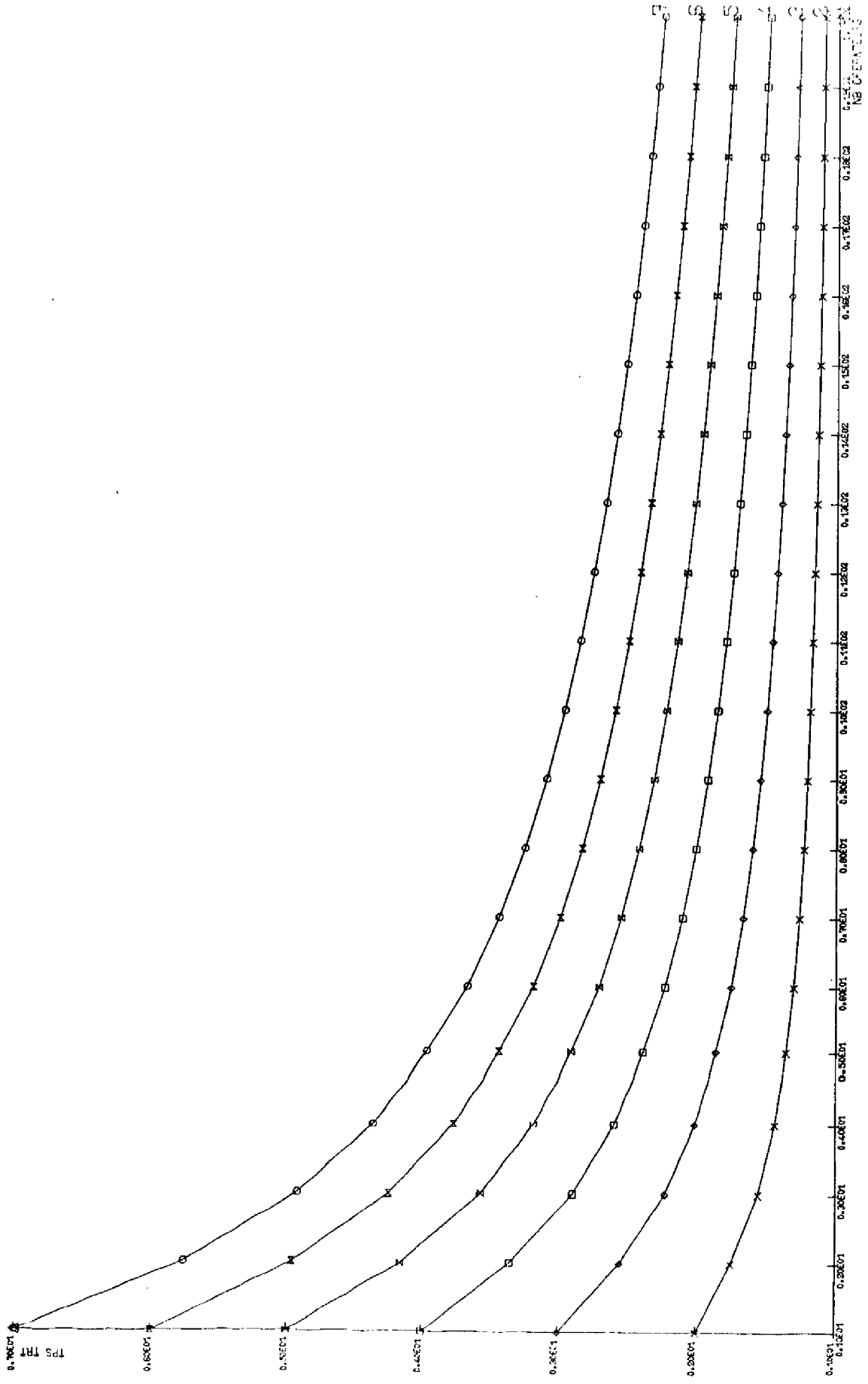

PROBABILITE D APPEL CONSTANTE
DEUX OPERATEURS PAR INSTRUCTION
TPS TRT
NB OPERATEURS

FIGURE VI.5.

VI.2.1.2. Nombre d'états du système

Soient :

- NCOUR le nombre d'opérateurs simples ($T \leqslant \tau$),
- NLONG le nombre d'opérateurs évolués ($T = K \times \tau$) ;

le nombre NB d'états que peut prendre le système est majoré par :

$NB = 2^K$ (cette valeur est atteinte pour NLONG $\geqslant$ K).

VI.2.1.3. Application

Les résultats présentés sur la figure VI.6. font apparaître la variation du temps moyen d'exécution d'une instruction en fonction de la dissymétrie de la structure : les courbes correspondent à des valeurs constantes de NCOUR, variant de 1 à N.

VI.2.2. Dissymétrie fonctionnelle

Dans cette structure où, pour chaque instruction, l'opérateur destination représente la fonction à effectuer sur une donnée fournie par l'opérateur source, le rôle joué par les opérateurs évolue avec les pas du programme. On peut introduire cette différence fonctionnelle dans le modèle de description si on mémorise la nature (source ou destination) du dernier appel à chacun des opérateurs.

Dans la suite de ce paragraphe, nous appellerons un opérateur "opérateur source" ou "opérateur destination" suivant qu'il a été cité comme source ou comme destination lors de sa dernière référence.

Nous avons vu au paragraphe VI.1. qu'un opérateur appelé à la fois en source et en destination par une instruction se comporte sur le plan temporel comme un opérateur appelé uniquement en destination. Il en est de même d'un point de vue fonctionnel car un opérateur qui vient de recevoir la donnée qu'il doit traiter ne se préoccupe plus de l'origine de cette donnée. Dans la suite du programme, l'opérateur est caractérisé par la dernière fonction exécutée ; sa nature est donc celle d'un "opérateur destination".

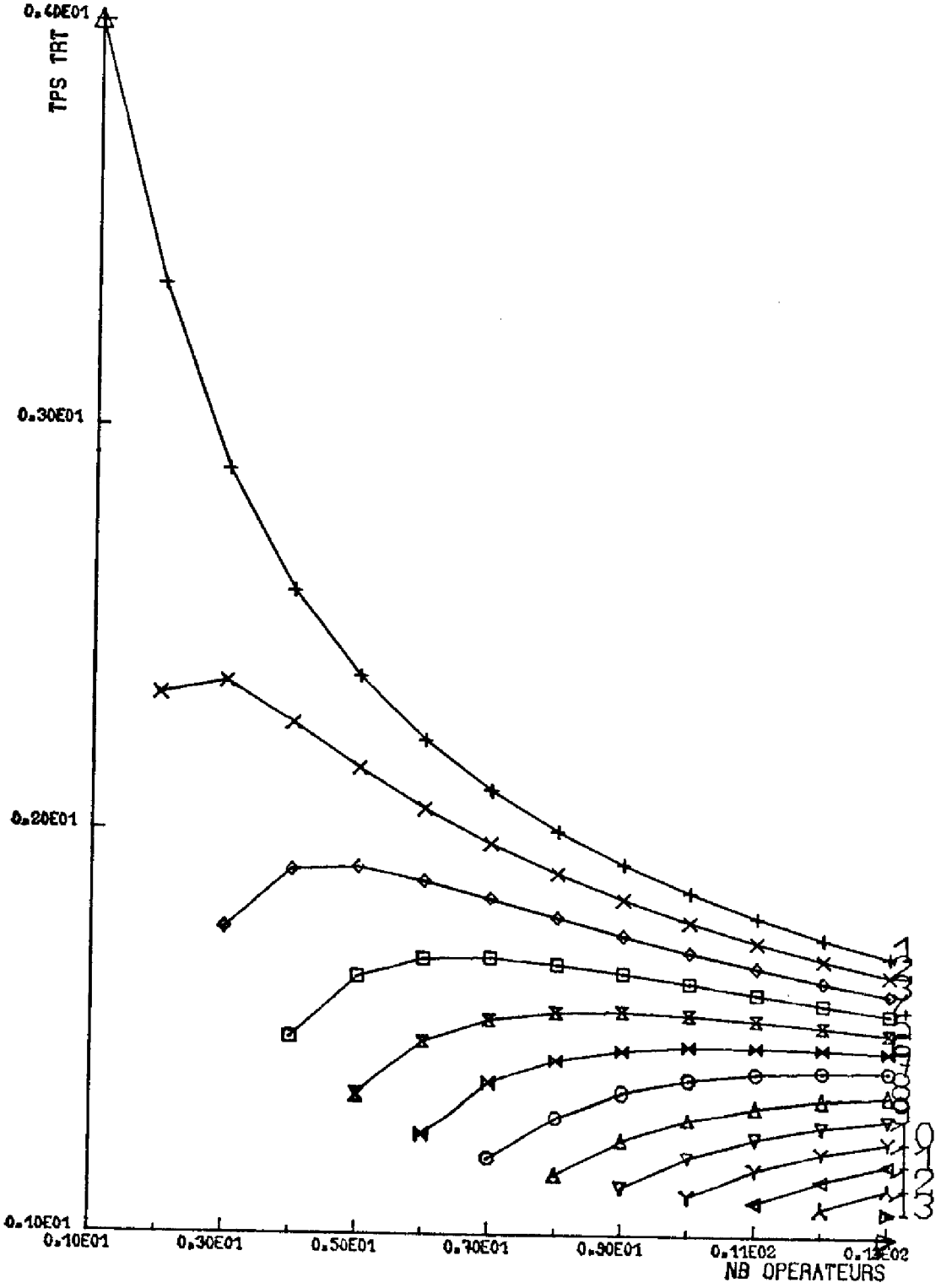

0.40E01
TPS TRT
0.30E01
0.20E01
0.10E01
0.10E01
0.30E01
0.50E01
0.70E01
0.90E01
0.11E02
0.13E02
NB OPERATEURS
1
2
3
4
5
6
7
8
9
10
11
12
13

FIGURE VI.6.

VI.2.2.1. Représentation du fonctionnement d'un opérateur

Le diagramme de la figure VI.7. représente le fonctionnement d'opérateurs dont le temps de traitement vaut T = 3 x τ , lorsque l'on tient compte de leur nature aux instants considérés.

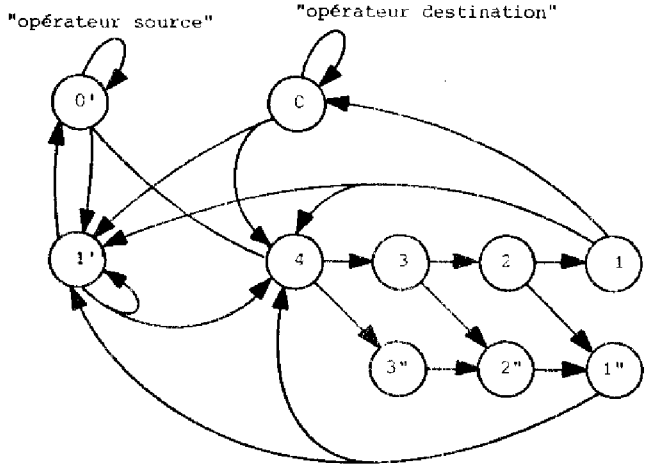


FIGURE VI.7.

Si l'on introduit des temps de transition entre états, on aboutit au diagramme de la figure VI.8.

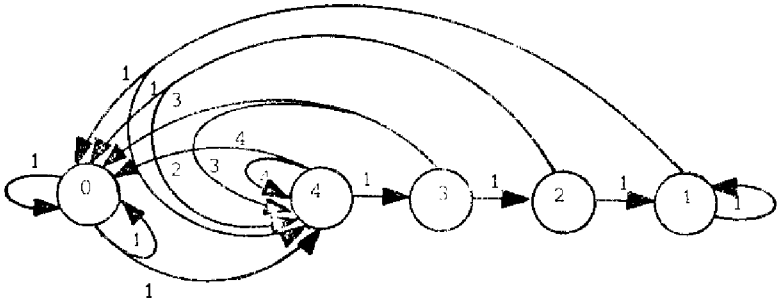


FIGURE VI.8.

VI.2.2.3. Nombre d'états

Le nombre NB des états représentatifs du système s'obtient à partir du nombre d'états du système où un seul opérateur est appelé par instruction en considérant que tout opérateur disponible à la fin du cyle considéré peut se trouver dans deux états :

- état (1) s'il était "destination",
- état (0) s'il était "source".

Un majorant du nombre des états est :

$$NB = \sum_{\alpha=0}^{\alpha=K-1} C_{K-1}^{\alpha}(N-\alpha) = 2^{K-2}\left[2N - K + 1\right]$$

Dans cette expression,

- $(N-\alpha)$ provient des opérateurs dans les états (0) ou (1);
- C_{K-1}^{α} provient des opérateurs dans les états autres que (0), (1) et (K+1).

Ce majorant est atteint pour les valeurs de N supérieures à K (nombre d'intervalles que dure le traitement).

VI.2.2.4. Recherche des probabilités d'appel des opérateurs

La distinction fonctionnelle entre les opérateurs "source" et les opérateurs "destination" au moment considéré permet de prendre en compte le phénomène suivant : dans une structure qui provient d'un découpage fonctionnel d'un système, il est tout à fait vraisemblable que si l'on demande l'exécution d'une fonction, on sera amené ultérieurement à faire référence à son résultat. Ceci entraîne qu'un opérateur "destination" a plus de chances d'être appelé par une nouvelle instruction qu'un opérateur "source" ; nous admettons par ailleurs que quelle que soit la nature de l'opérateur qui sera sélectionné, il y a équiprobabilité qu'il soit appelé en source et en destination.

Soient :

- PS la probabilité d'appel de l'un quelconque des NS opérateurs qui sont "source",
- PD la probabilité d'appel de l'un quelconque des ND opérateurs qui sont "destination",

les valeurs de NS et ND étant déterminées à l'instant où l'on considère le système, c'est-à-dire, pour un état de la structure.

Pour tout état du système, on connaît NS et ND qui vérifient :

$$NS + ND = N$$

On admet d'autre part que l'on connaît le rapport :

$$DF = \frac{PD}{PS} \geqslant 1$$

constant dans le temps, qui caractérise la dissymétrie fonctionnelle : un opérateur qui a effectué une opération a DF fois plus de chances d'être appelé qu'un opérateur qui a déjà fourni son résultat.

La détermination de PS et PD se fait en considérant qu'une instruction désigne une nouvelle source et une nouvelle destination, soit, puisqu'on a indiqué que ceci faisait l'objet de deux tirages indépendants identiques :

$$NS \times PS + ND \times PD = 1,$$

Il vient donc :

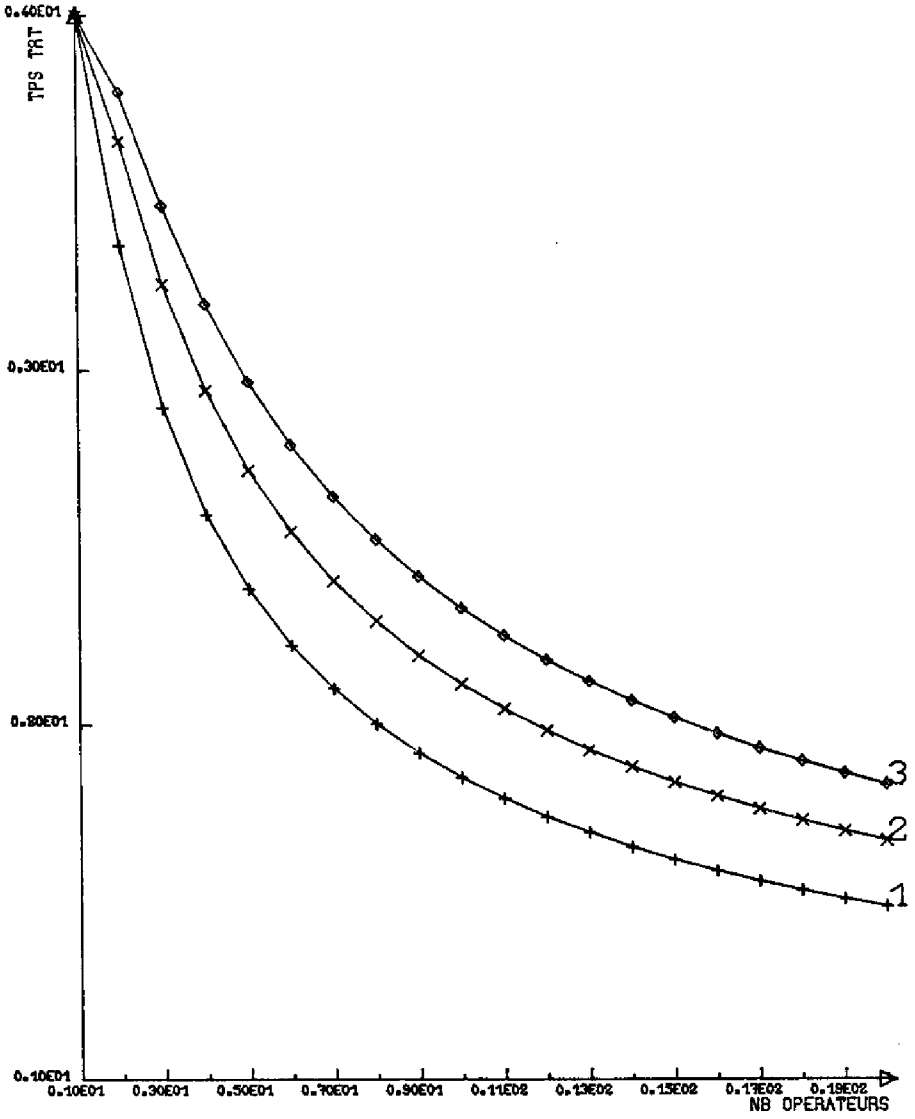
$$PS = \frac{1}{NS + DF(N-NS)}$$

$$PD = \frac{DF}{NS + DF(N-NS)}$$

La figure VI.9. fait apparaître la variation du temps moyen entre deux instructions pour les différentes valeurs de DF :

- Courbe 1 : DF = 1
- Courbe 2 : DF = 2
- Courbe 3 : DF = 3

Les calculs ont été faits sur l'exemple de structure $N = 5$, $T = 3 \times \gamma$.



DISSYMETRIE FONCTIONNELLE
DEUX OPERATEURS PAR INSTRUCTION

FIGURE VI.9.

VI.3. REMISE EN CAUSE DE L'HYPOTHESE DE L'INDEPENDANCE DES PAS DE PROGRAMME

VI.3.1. Introduction d'un phénomène de localité

VI.3.1.1. Dépendance des instructions

Si l'hypothèse de l'indépendance des instructions d'une séquence amène des résultats valables pour des structures multiprocesseur banalisés, elle est contestable dans le cas d'une structure à opérateurs spécialisés.

En effet, la constitution des programmes est généralement telle que le système connaît, pendant leur déroulement, de fortes activités d'entrée/sortie, de traitement arithmétique, etc., qui s'enchaînent par phases successives. Ceci traduit le fait que les traitements exécutés influent sur la nature des instructions qui suivent et fait apparaître le rôle du chemin des fonctions dans la dépendance entre les instructions. L'influence du chemin des données est pourtout plus importante et vient s'ajouter à celle du chemin des fonctions.

Nous avons déjà fait apparaître dans le modèle précédent que la demande d'exécution d'un traitement sur une donnée entraîne la présence, dans la suite du programme, d'une instruction utilisant le résultat obtenu (ceci correspond au chemin des données). Nous voulons rajouter la nôtion de temps amenée par le chemin des fonctions en introduisant dans le modèle un phénomène de localité dû au fait que le résultat obtenu sera utilisé rapidement.

Il existe donc un influence des instructions sur celles qui suivent immédiatement dans le déroulement du programme.

VI.3.1.2. Nouvelle caractérisation d'un programme

Le lot des instructions exécutables par la machine est de dimension finie ; aussi ne peut-on pas considérer l'apparition de chaque fonction, dans la succession des événements que constitue l'enchaînement des pas de programme, comme un phénomène aléatoire indépendant.

En effet, un pas du programme correspondant systématiquement à la sélection d'une instruction, on ne peut pas admettre que l'apparition de chaque fonction dans le déroulement du programme constitue un processus de POISSON par exemple.

Nous remplaçons les tirages indépendants des opérateurs par des probabilités de référence qui tiennent compte des p derniers opérateurs "destination" cités. Le comportement du programme est alors décrit par une chaîne de MARKOV dont les états correspondent à la juxtaposition des noms de ces p opérateurs, comme dans le modèle fonctionnel développé au paragraphe V.1.1.1. .

Dans le cas où la détermination d'une instruction dépend uniquement de la fonction appelée précédente, le comportement du programme est représenté par la matrice stochastique suivante :

anciens états \ nouveaux états	O_{p1}	O_{p2} ----- O_{pN}
O_{p1}	P_0+P_1	P_0 ----- P_0
O_{p2}	P_0	P_0+P_1 ---- P_0
O_{pN}	P_0	P_0 ----- P_0+P_1

$(O_{p1}),\ldots,(O_{pN})$, noms des N' opérateurs de la structure, sont ici les noms des états de la chaîne de MARKOV qui décrit le programme. Son fonctionnement peut se résumer ainsi : la probabilité de référence d'un opérateur provient de la superposition :

- d'une probabilité constant P_0,
- d'un terme traduisant l'influence locale du dernier opérateur nommé P_1.

On a :

$$P_r(O_{pi},t+1) = P_0 + P_i \times P_r(O_{pi},t)$$

avec :

$$N P_0 + P_1 = 1$$

VI.3.2. Mise en oeuvre de cette nouvelle caractérisation des tâches

Nous disposons d'un modèle de description de la structure, sous la forme d'une chaîne de MARKOV avec valeurs de transition. Les états de cette classe stipulent dans quelle phase d'activité se trouve chaque opérateur.

Le problème consiste donc à ramener au niveau de ce modèle les informations sur le comportement du programme, données par une chaîne de MARKOV sans valeur de transition, ce qui revient à déduire de la chaîne fonctionnelle la valeur des probabilités de transition de la chaîne structurelle pour en étudier le régime permanent.

Cette opération a déjà été effectuée implicitement dans le cas des tirages indépendants et a donné lieu aux résultats présentés au paragraphe VI.1.4.

VI.3.2.1. Influence de la dernière fonction nommée

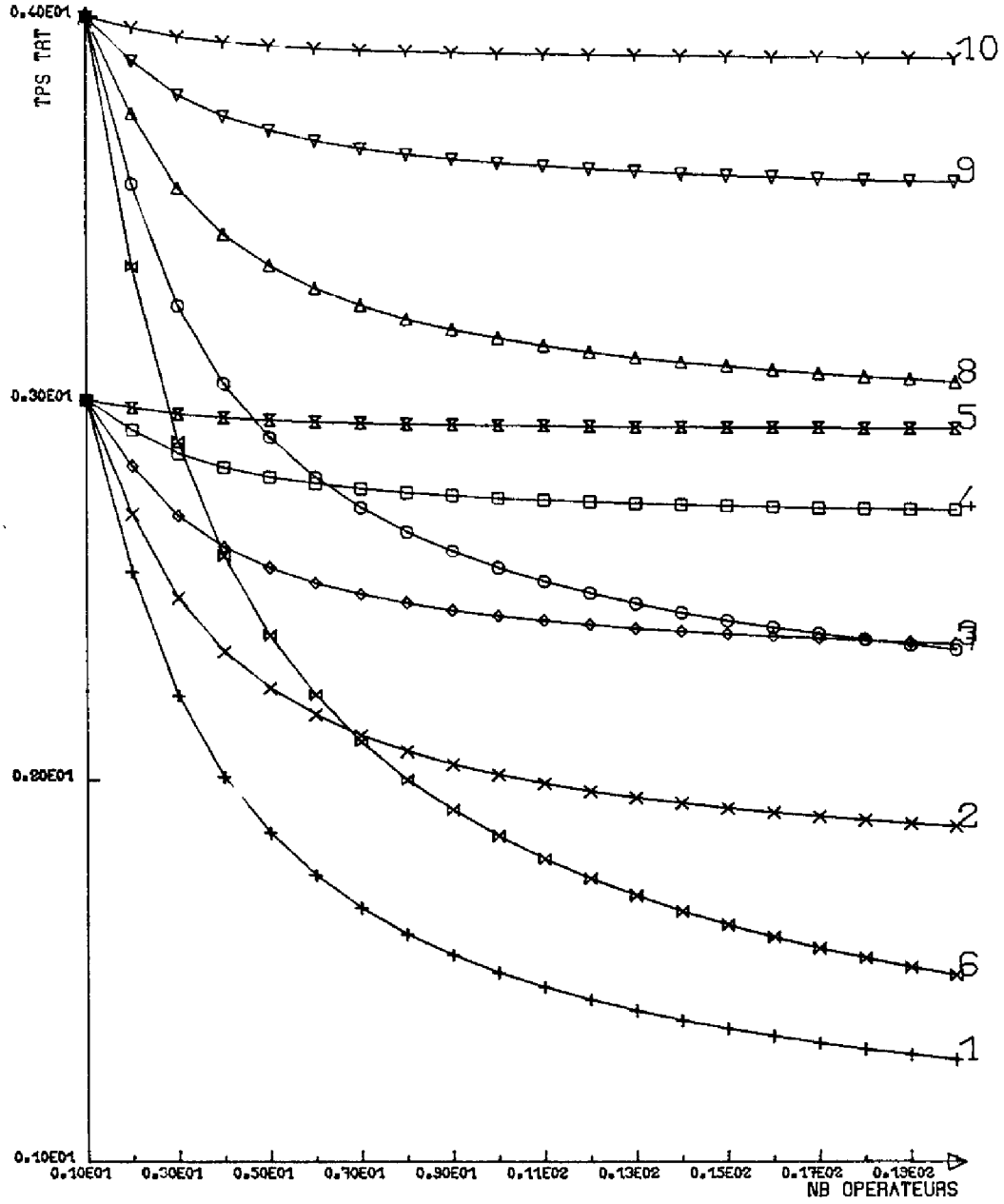
Puisque la dénomination des états dans le modèle structurel permet de connaître quel est le dernier opérateur nommé (celui qui réalise le transfert appelé (K+1)), la matrice des probabilités de transition est facile à établir.

Dans l'exemple choisi de 5 opérateurs dont le temps de traitement dure $T = 3 \times \tau$, le dernier opérateur appelé est indiqué par le nombre (4) dans la dénomination de l'état du système ; la matrice $\mathbb{P}$ des probabilités des transitions et la matrice T des temps de transition se déduisent du tableau de la figure VI.4., établi au paragraphe VI.1.3. pour le cas de structures symétriques.

La figure VI.10. montre le temps moyen entre deux instructions pour différentes valeurs du temps de traitement ($T = K \tau$) et de l'influence locale (P_1) en fonction du tableau suivant :

Numéro de la courbe

K \ P_1	0	0.2	0.4	0.6	0.8
2	1	2	3	4	5
3	6	7	8	9	10



INFLUENCE FONCTION PRECEDENTE
DEUX OPERATEURS PAR INSTRUCTION

FIGURE VI.10.

VI.3.2.2. Influence des deux dernières fonctions nommées

Il n'est pas possible d'utiliser le même modèle lorsque l'influence d'une instruction s'étend sur les deux pas de programme suivants ou plus. En effet, la dénomination des états du modèle structurel n'apporte pas assez d'informations fonctionnelles : on ne sait pas, par exemple, déterminer avec certitude le nom de l'opérateur appelé par l'avant dernière instruction dans le cas du premier état de la figure VI.11.

	Nom des opérateurs O_{p1}	O_{p2}	O_{p3}	O_{p4}	O_{p5}	Dernier opérateur appelé	Avant-dernier opérateur appelé
①	4	1	1	1	1	O_{p1}	*
②	4	2	1	1	1	O_{p1}	O_{p2}
③	4	3	1	1	1	O_{p1}	O_{p2}
④	4	3	2	1	1	O_{p1}	O_{p2}

Nom des états du système

FIGURE VI.11.

Pour représenter l'évolution de ce système lorsque le choix d'une instruction dépend de deux dernières fonctions appelées, il est donc nécessaire de dédoubler l'état (1).

Ceci, au même titre que les différentes étapes du passage d'un modèle fonctionnel à un modèle structurel, au paragraphe V.1., met l'accent sur l'opposition entre les deux descriptions d'un système :

- description par le comportement des opérateurs de la structure,
- description par l'évolution des instructions du programme.

L'emploi d'un modèle structurel exact est possible lorsque le comportement du programme est simple (tirage indépendant ou chaîne de MARKOV à un opérateur).

L'utilisation de modèles fonctionnels approchés est nécessaire lorsque la description du programme devient plus complexe ; de plus, l'incertitude liée à cette méthode de représentation diminue lorsque l'influence locale entre les instructions devient forte.

VI.3.2.3. Caractérisation de tâches dont la forme est optimisée

L'optimisation de la forme d'un programme, pour une machine donnée, affecte la séquence de déroulement des instructions de telle sorte que la probabilité de référence est plus grande pour les opérateurs disponibles que pour ceux qui sont en cours d'exécution.

Cette procédure ne nécessite pas la connaissance de l'ordre d'appel des derniers opérateurs nommés, puisqu'elle se réfère aux instructions à venir et que la sélection s'effectue en fonction du moindre temps d'attente. Le temps d'attente étant indiqué pour chaque opérateur dans la constitution d'un état du système, on pense obtenir une bonne description du comportement du programme en caractérisant les tâches par les probabilités de référence indiquées sur le tableau de la figure VI.12. Cette caractérisation des tâches conduit aux résultats de la figure VI.13.

Nom des opérateurs classés par ordre décroissant du temps d'attente	$O_{p1}\ O_{p2} \cdots O_{pN}$
Probabilité de référence à ces opérateurs ($\alpha = \frac{2}{N(N+1)}$)	$\alpha \quad 2\alpha \ldots N \times \alpha$

FIGURE VI.12.

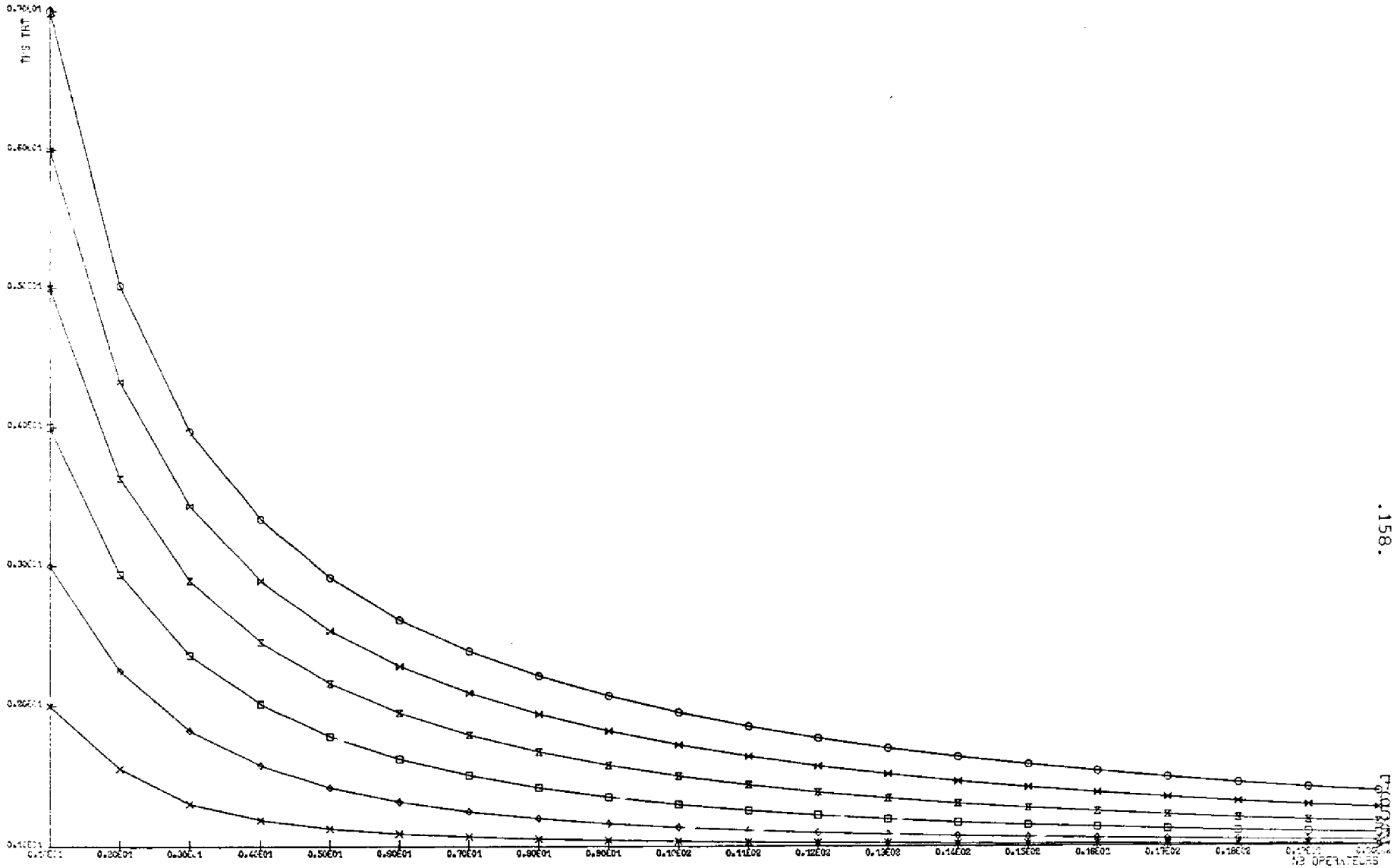
VI.4. SYSTEME D'INTERRUPTION D'A.S.M.O.D.E.E. 02 : DESCRIPTION AU NIVEAU DE LA CHARGE DU SYSTEME

VI.4.1. Introduction

Nous n'avons pas développé les deux modèles de représentation du système d'interruption d'A.S.M.O.D.E.E. 02 sur des exemples numériques parce que le plus souvent, le fonctionnement réel ne correspond pas à l'un des deux cas particuliers présentés et conduit à un modèle inexploitable.

Ceci donne lieu à deux remarques importantes :

- un système qui a un comportement déterministe (temps d'exécution constants) et des paramètres descriptifs qui ne sont pas homogènes d'un point de vue quantitatif est très difficile à représenter par un modèle mathématique simple,



AMELIORATION FORME PROGRAMME
DEUX OPERATEURS PAR INSTRUCTION

FIGURE VI.13.

- les chaînes de MARKOV sont peu adaptées pour une description d'un système au niveau de la mission alors que la théorie des files d'attente fournit un grand nombre de résultats en ce qui concerne les performances de systèmes à ce niveau de description.

Le système des interruptions d'A.S.M.O.D.E.E. 02 correspond à seize niveaux hiérarchiques distincts. Les programmes qui correspondent à ces niveaux sont numérotés de 1 (niveau le plus élevé) à 16 (niveau le moins prioritaire).

Chacun des programmes d'interruption peut se trouver dans l'un des trois états suivants :

- (A_c) : actif, c'est-à-dire qu'il utilise tous les moyens du système,
- (A_t) : en attente, l'exécution du programme est demandée mais différée,
- (R) : au repos, l'exécution du programme n'est pas demandée.

Les demandes d'exécution des programmes n'étant prises en compte que dans l'état de repos et la multiprogrammation dans A.S.M.O.D.E.E. 02 se limitant à l'occupation de tout le système par le programme le plus prioritaire demandé, les différentes configurations que peut prendre le système sont indiquées sur le tableau de la figure VI.14.

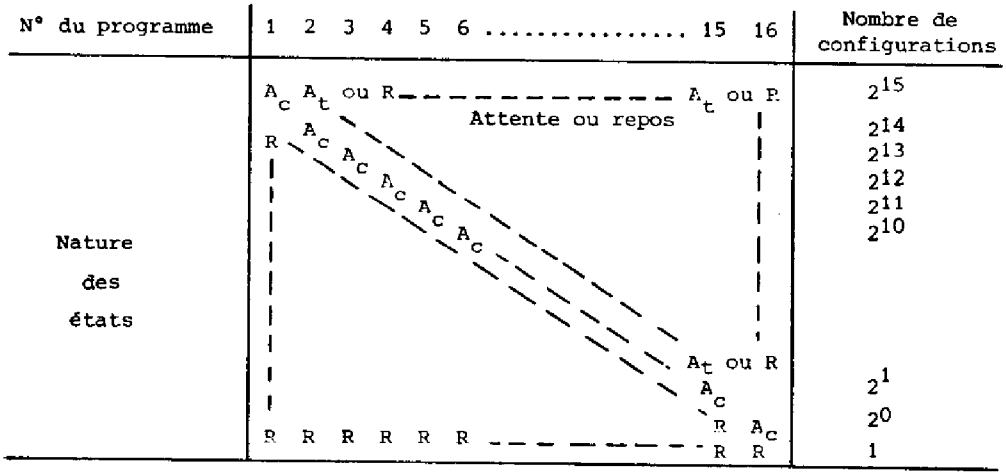


FIGURE VI.14.

Le système peut donc se trouver dans 2^{16} configurations différentes.

Nous admettrons que les probabilités de demande d'exécution des programmes d'interruption sont constantes dans le temps mais qu'elles ne sont prises en compte que lorsque les programmes concernés sont dans l'état de repos (c'est le principe qui a été développé sur la maquette).

VI.4.2. Temps d'exécution des tâches fixes

Lorsque le temps d'exécution de l'ensemble des instructions d'une tâche est le même à chaque réalisation de cette tâche, le comportement du système, au niveau de la mission, peut être décrit par une chaîne de MARKOV.

Comme les changements de configuration du système peuvent apparaître à chaque cycle machine, il est nécessaire de faire une description du système à ce niveau là. Néanmoins, dans les cas particuliers où les durées d'exécution de chaque programme (considéré seul) sont identiques ou possèdent un Plus Grand Commun Diviseur qui est du même ordre de grandeur que ces durées, on peut prendre ce PGCD comme unité de temps du modèle ; dans la mesure où on ne s'intéresse qu'au débit moyen d'exécution des programmes, on peut considérer que les transitions entre états ne peuvent se produire qu'à la fin des unités de temps ainsi définies.

L'exemple donné sur la figure VI.15. décrit le comportement du système obtenu, à la suite de cette simplification :

(On considère que tous les programmes durent le même temps pris comme unité de temps sur le diagramme).

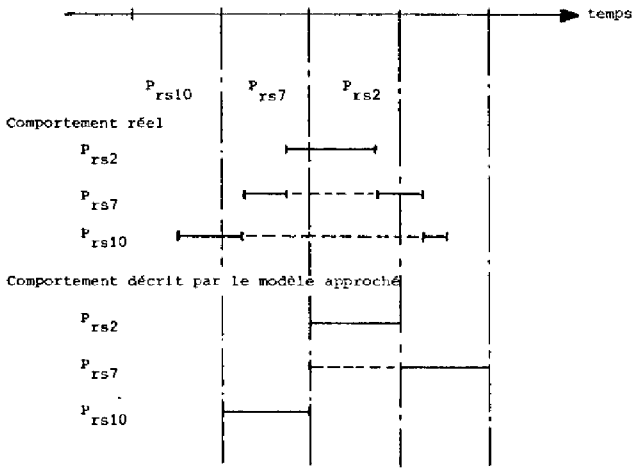


FIGURE VI.15.

Le temps d'exécution global est le même.

VI.4.3. Temps d'exécution des tâches exponentiellement distribuées

On peut représenter facilement le comportement du système au niveau de la mission par une chaîne de MARKOV dont les états ont une durée correspondant au temps de cycle du système, lorsqu'on peut admettre que les temps d'exécution des programmes d'interruption sont exponentiellement distribués. On est alors conduit à un modèle présentant des analogies avec celui développé au paragraphe V.3.2. et dans lequel la notion de temps passé avant de quitter un état n'apparaît plus explicitement, mais est remplacée par une probabilité de transition hors de cet état.

VI.5. PRINCIPAUX RESULTATS

La caractérisation des tâches tient une place prépondérante dans le chapitre VI, alors que les intervalles de définition des paramètres structurels n'ont fait l'objet d'aucun commentaire. Ceci tient au fait que nous avons porté notre effort en premier lieu sur la prévision des performances de la maquette d'A.S.M.O.D.E.E. 02, la réalisation fixant un cadre à l'étude entreprise sur des systèmes multiopérateur , avec, comme variations possibles :

- nombre d'opérateurs N = 1 à 20,
- temps de traitement T = 1 x τ à 7 x τ.

De cette étude, nous pouvons dégager les résultats suivants :

1. Les figures VI.5., VI.10. ou VI.13. en fonction de l'adaptation de l'écriture du programme aux particularités du fonctionnement d'A.S.M.O.D.E.E. 02 permettent de quantifier l'intérêt qu'il y a à introduire un parallélisme par une exécution anticipée des instructions immédiatement en séquence.

En effet, à chaque temps d'exécution d'une fonction -identique que celle-ci soit réalisée dans l'unité centrale d'un calculateur ou dans un module fonctionnel d'A.S.M.O.D.E.E. 02- correspond une courbe d'un faisceau qui fait apparaître le nombre d'opérateurs fonctionnels à partir duquel la structure modulaire devient avantageuse, malgré la perte de temps due au choix d'une transmission par caractères et l'adoption de circuits de décodage lents.

2. Le modèle présentant une dissymétrie matérielle permet de prévoir les performances d'un système multiopérateur ayant deux types de fonction qui se différencient par leur temps d'exécution. La dissymétrie est mesurée par le rapport des durées entre les instructions longues et les instructions courtes. Ce rapport vaut 3 ou plus exactement 5 si on exclue les temps de cycle de recherche des instructions, dans le cas du calculateur embarqué A.R.M.M.S. développé au "Marshall Space Flight Center". Il est calculé avec le temps de la multiplication et le temps de l'addition. Bien que nous ayons pris des rapports du même ordre de grandeur, leur origine est différente : en effet, le temps de multiplication (160 μ s) et le temps d'addition (8 μ s) sont dans le rapport 20 pour A.S.M.O.D.E.E. 02. Les temps qui interviennent dans le rapport utilisé correspondent à la durée des traitements longs et au temps unitaire τ séparant les tests de disponibilité des opérateurs (τ = 25 μ s).

3. La comparaison des courbes V.24. et VI.5. permet d'apprécier l'influence des conflits créés par le chemin des données sur les performances du système.

Plus globalement, ces résultats rapportés à des systèmes interconnectés par liaison bus donnent des informations sur la quantité de conflits d'accès au bus de communication en fonction de l'importance des dépendances entre modules.

CONCLUSION

L'analyse des actions élémentaires de systèmes informatiques de commande et des contraintes de la conduite de processus en temps réel nous a conduit, dans la première partie de ce mémoire, au développement d'une structure multiopérateur originale qui couvre un créneau de performances important situé entre les systèmes monoprocesseur classiques et les systèmes multiprocesseur .

Nous avons donné au système développé un caractère évolutif en amenant au niveau du matériel les distinctions fonctionnelles qui apparaissent dans la constitution d'un système de commande. Cette démarche a abouti à une structure modulaire traduisant un découpage par fonctions, dont une manifestation essentielle consiste en une dissociation totale des actions et des moyens de gestion par rapport aux organes et aux traitements propres à la conduite du processus.

Bien qu'une démarche analogue se retrouve dans plusieurs études qui ont donné lieu à des réalisations, nous apportons, par la structure multiopérateur proposée pour la commande en temps réel, une originalité due à l'introduction d'un critère supplémentaire, celui de la simplicité.

En ce qui concerne le fonctionnement, cette simplicité se traduit :

- au niveau de la mission, par une gestion des tâches en fonction de leur priorité,
- au niveau des instructions, par la gestion de la pile que constitue la mémoire morte selon le mode "F.I.F.O.".

La réalisation obtenue correspond à une structure hiérarchique maître-esclave simple ; l'organe spécifique de gestion de la mission (opérateur de gestion des interruptions) et les organes de gestion des moyens (mémoire programme et opérateur pilote) constituent les opérateurs maîtres, les éléments esclaves étant les opérateurs d'exécution des traitements.

Nous avons examiné à postériori les possibilités amenées par l'introduction d'un parallélisme des traitements dans un tel système. Ce parallélisme se limite au recouvrement des instructions en séquence pour éviter d'alourdir considérablement le programme d'exploitation, et correspond à une

amélioration de la forme des programmes réalisée avant son implantation sur le site. Nous proposons cette démarche comme une alternative aux structures parallèles vraies qui traitent simultanément des instructions de programmes indépendants mais qui nécessitent des programmes d'exploitation complexes.

La détermination de la nature des fonctions réalisées par les opérateurs de traitement soulève le problème de l'incompatibilité entre plusieurs contraintes qui affectent notamment l'aspect système (découpage fonctionnel) et l'aspect économique (bonne utilisation du matériel) ; nous avons donné priorité, dans toute cette étude, à l'aspect système, malgré les répétitions d'éléments matériels que cette solution entraîne.

La maquette réalisée, bien que répondant à la démarche annoncée par sa découpe en fonctions spécifiques, a fait l'objet de quelques concessions qui masquent légèrement notre objectif d'un découpage purement fonctionnel ; ces concessions concernent :

- le choix d'un bus unique à cause de la considérable réduction du matériel que cela apporte dans une structure modulaire,
- le choix de regroupements d'origine matérielle plus que fonctionnelle pour d'une part l'opérateur mémoire de programme qui stocke des instructions concernant les données aussi bien que les adresses, d'autre part l'opérateur arithmétique et logique.

En ce qui concerne l'opérateur arithmétique et logique, nous nous sommes limité à l'utilisation des possibilités actuelles des circuits en nous attachant à montrer la compatibilité entre le traitement autonome microprogrammé et la liaison avec le reste du système.

L'impact du degré d'évolution des fonctions réalisées par les opérateurs sur les performances du système fait l'objet de la modélisation présentée dans la seconde partie.

Les deux problèmes essentiels rencontrés dans cette partie de l'étude sont le propre de toute évaluation des performances de systèmes informatiques [WIL.72] :

- la caractérisation des tâches,
- la description du système.

Dans le cas particulier de la commande de processus en temps réel, les différents niveaux de description d'un système que TEOREY présente sous la forme du tableau de la figure VII.1. rappelé dans [LER.76, p.11] sont très liés.

Niveau	Occurrence d'événements
Description fonctionnelle grossière	50 à 500 ms
Description fonctionnelle détaillée	1 à 10 ms
Instruction machine	1 à 10 μs

FIGURE VII.1.

Nous avons, pour notre part, abordé le problème de la manière résumée sur le tableau de la figure VII.2.

Niveau	Occurrence d'événements
Description au niveau de la mission	$\sim$ 100 ms
Description au niveau d'instructions évoluées	$<$ 1 ms

FIGURE VII.2.

En effet, la création matérielle des fonctions de l'automatique amène un lien direct entre les deux niveaux du bas du tableau de la figure VII.1. que nous développons dans un unique modèle qui prend en compte :

- l'évolution déterministe du matériel,
- la matrice aléatoire de l'occurrence des fonctions évoluées.

Nous avons de plus indiqué comment nous pouvions obtenir une description de l'évolution du système au niveau des fonctions évoluées indépendantes de la description de la mission dans le cas de la multiprogrammation très simple d'A.S.M.O.D.E.E. 02.

Il convient de noter que les hypothèses retenues pour simplifier la représentation du système :

- hypothèse d'homogénéité de la structure,
- hypothèse d'un appel programmé des fonctions,

ont été effectuées pour la modélisation de la maquette d'A.S.M.O.D.E.E. 02 et que leur domaine de validité ne s'étend pas forcément au cas de fonctions plus évoluées.

Pour tenir compte de ce phénomène et donner plus de généralité au modèle développé, nous avons porté une attention toute particulière dans le chapitre VI aux possibilités d'introduire différentes caractérisations des programmes.

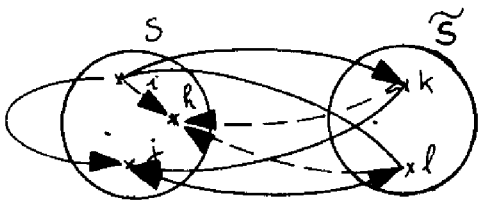
Dans le cadre des hypothèses effectuées -attribution d'un moyen de communication unique à différents modules d'un système pendant un quantum de temps fixe- les résultats obtenus par comparaison d'une structure où un seul opérateur est appelé à une structure où deux opérateurs sont appelés sont interprétés comme les différences existant entre des systèmes faiblement liés ou aux actions très dépendantes.

Nous tenons enfin à dire que nous avons essayé de conduire cette étude avec le souci de la plus grande rigueur possible, mais que nous n'avons pas pu aborder deux points qui nous semblent fondamentaux pour les recherches menées dans ce domaine :

- caractériser les tâches de manières systématique : la caractérisation du comportement des programmes constitue en effet l'élément de base nécessaire à toute évaluation des performances ; la nature et le nombre des renseignements requis nécessitent l'introduction "d'espions" dans des systèmes déjà opérationnels pour lesquels on ne peut espérer aucun bénéfice immédiat,

- confronter les difficultés et les résultats de modélisation menés dans des optiques différentes : études des performances temporelles, de la sûreté de fonctionnement, du coût,... pour obtenir une image réelle des performances des systèmes actuels qui s'orientent de plus en plus vers des possibilités de fonctionnement dégradé.

ANNEXE



Les quantités barrées indiquent les grandeurs concernant le passage par les états de $\tilde{S}$ à l'exclusion de tout état de S.

$$\overline{\tau_{ij}} = \frac{p_{ij}\,\tau_{ij} + \sum_k p_{ik}\,b_{kj}\,(\tau_{ik} + \overline{\tau_{kj}})}{p_{ij} + \sum_k p_{ik}\,b_{kj}}$$

$$\theta_{moy} = \sum_i \overline{a_i}\,\overline{\tau_i} = \sum_i \overline{a_i}\left(\sum_j \overline{p_{ij}}\,\overline{\tau_{ij}}\right)$$

$$\overline{p_{ij}} = p_{ij} + \sum_k p_{ik}\,b_{kj}$$

$$\theta_{moy} = \sum_i \overline{a_i}\left[\sum_j \left\{p_{ij}\,\tau_{ij} + \sum_k p_{ik}\,b_{kj}\,(\tau_{ik} + \overline{\tau_{kj}})\right\}\right]$$

$$= \sum_i \overline{a_i}\left(\sum_j p_{ij}\,\tau_{ij} + \sum_{j,k} p_{ik}\,b_{kj}\,\tau_{ik} + \sum_{j,k} p_{ik}\,b_{kj}\,\overline{\tau_{kj}}\right)$$

$$\sum_{j,k} p_{ik}\,b_{kj}\,\tau_{ik} = \sum_k p_{ik}\,\tau_{ik}\left(\sum_j b_{kj}\right) = \sum_k p_{ik}\,\tau_{ik}$$

$$= 1$$

$$\sum_{j,k} p_{ik}\,b_{kj}\,\overline{\tau_{kj}} = \sum_k p_{ik}\left(\sum_j b_{kj}\,\overline{\tau_{kj}}\right) = \sum_k p_{ik}\,\overline{\tau_k}$$

$$= \overline{\tau_k}$$

$$\theta_{moy} = \sum_i \overline{a_i}\left(\sum_{j\in S} p_{ij}\,\tau_{ij} + \sum_{k\in\tilde{S}} p_{ik}\,\tau_{ik} + \sum_{k\in\tilde{S}} p_{ik}\,\overline{\tau_k}\right)$$

Dans notre cas : $\tau_{ij} = \tau_{ik} = 1$

$$\theta_{moy} = \sum_i \overline{a_i}\left(\underbrace{\sum_{j\in S} p_{ij} + \sum_{k\in\tilde{S}} p_{ik}}_{=1} + \sum_{k\in\tilde{S}} p_{ik}\,\overline{\tau_k}\right)$$

$$\theta_{moy} = \sum_i \overline{a_i}\left(1 + \sum_{k\in\tilde{S}} p_{ik}\,\overline{\tau_k}\right)$$

$$= \sum_i \overline{a_i} + \sum_i \overline{a_i}\left(\sum_{k\in\tilde{S}} p_{ik}\,\overline{\tau_k}\right)$$

$$= 1 + \sum_i \overline{a_i}\left(\sum_{k\in\tilde{S}} p_{ik}\,\overline{\tau_k}\right)$$

BIBLIOGRAPHIE

[AGR.75] A.K. AGRAWALA, R.M. BRYANT, J.M. MOHR, "An approach to the workload characterization problem", Technical Report n°TR410, University of Maryland, September 1975.

[ALA.72] A. ALABAU MUNOZ, "Contribution à la conception d'organes numériques de traitement de l'information, Projet A.S.M.O.D.E.E.", Thèse de Docteur-Ingénieur, Université Paul Sabatier, Toulouse, 22 Décembre 1972.

[AND.75] G.A. ANDERSON, E.D. JENSEN, "Computer interconnection structures : Taxonomy, characteristics and examples", ACM Computing Surveys, vol.7, n°4, December 1975, pp.197-213.

[BAT.76] A. BATSON, "Program behavior at the symbolic level", Computer, November 1976, pp.21-26.

[BEI,71] B. BEIZER, "The architecture and engineering of digital computer complexes", Plenum Press, Tome 1, pp.358-366.

[BEL.74] G. BEL, J.B. CAVAILLE, J. DELMAS, "Optimisation dans les graphes, flux et ordonnancement", Cours de l'E.N.S.A.E., Toulouse, 1974.

[BHA.73] D.P. BHANDARKAR, "Analytic models for memory interference in multiprocessor computer systems", Ph.D. dissertation, Carnegie-Mellon University, Pittsburg, Penn., September 1973.

[BHA.75] D.P. BHANDARKAR, "Analysis of memory interferences in multiprocessors", IEEE Trans. on Computer, vol.C-24, n°9, September 1975, pp.897-908.

[BRU.74] C. BRUNET, "Etude d'une structure biprocesseur banalisé", Thèse de Docteur-Ingénieur, Institut National Polytechnique de Grenoble, 7 Mars 1974.

[BUZ.71] J.P. BUZEN, "Queuing network models of multiprogramming", Ph.D. dissertation, Harvard University, August 1971.

[COC.74] B. COCHI, "Modeling aspects of computer architecture", Computer architectures and networks : modeling and evaluation, Ed. by E. GELEMBE and R. MAHL, North Holland/American, August 1974.

[COR.75] M. CORRAZA, "Techniques mathématiques de la fiabilité prévisionnelle", Edition Cepadues, Toulouse, 1975.

[COS.73.a.] A. COSTES, J.C. LAPRIE, "Contribution to the design of specialisable processors : the A.S.M.O.D.E.E. project", COMPCON 73, Palo Alto, Calif., March 1973.

[COS.73.a.] A. COSTES, J.C. LAPRIE, "Projet A.S.M.O.D.E.E.", Journées de l'I.R.I.A. sur les structures résultant d'un groupement de processeurs, Grenoble, 22-23 Novembre 1973.

[DEN.69] M. DENIS-PAPIN, A. KAUFMAN, "Cours de calcul matriciel appliqué", Edition Albin Michel, 1969, pp.321-344.

[FLY.74] M.J. FLYNN, "Trends and problems in computer organization", Information Processing 74, North Holland Publishing Company, 1974, pp.3-10.

[HUG.72] HUGHES AIRCRAFT CO., "Design of a modular digital computer system", Contract NAS 827929, Internal Report, 1972.

[KAU.70] A. KAUFMANN, "Méthodes et modèles de la recherche opérationnelle", DUNOD Editeur, 1970, Tomes 1 et 2.

[KEL.75] R.M. KELLER, "Look-ahed processors", ACM Computing Surveys, vol.7, n°4, December 1975, pp.177-195.

[KEM.59] J.G. KEMENY, J.L. SNELL, "Finite MARKOV chaîns", D. Van Nostrand Company Inc., Princetown, 1959.

[LAG.76] M.L. LAGIER, "Conception et réalisation d'un moniteur temps réel pour un système multiprocesseur", Thèse de Docteur-Ingénieur, University de Nancy I, Décembre 1976.

[LAL.75] J.H. LALA, "Performance evaluation of a multiprocessor in real-time environment", Ph.D. dissertation, The Charles Stark Draper Laboratory, 1975.

[LAN.75] C. LANDRAULT, Y. SAGOT, C. BEOUNES, "A.S.M.O.D.E.E. 02 : an evolutive modular processor", Digital Process, vol.1, n°4, November 1975, pp.319-327.

[LAP.74] J.C. LAPRIE, A. COSTES, C. LANDRAULT, A.L. BOGADO, "Projet A.S.M.O.D.E.E.", Journées d'Etudes sur les Calculateurs Numériques Embarqués , Toulouse, 10-12 Juin 1974, pp.97-113.

[LEN,75] J. LENFANT, P. BURGEVIN, "Empirical data on program behaviour", Internal Computing Symposium, Antibes, France, June 1975.

[LER.76] J. LEROUDIER, M. PARENT, "Quelques aspects de la modélisation des systèmes informatiques par simulation à éléments discrets", RAIRO Informatique, vol.10, n°1, Janvier 1976, pp.5-26.

[MEY.76] J.F. MEYER, "A model hierarchy for evaluating the effectiveness of computing systems", IIIème Congrès National de Fiabilité, Perros-Guirec, France, Septembre 1976, pp.539-555.

[OHM.74] K. OHMORI et al, "MICS-a multimicroprocessor system", Information Processing 74, North Holland Publishing Company, 1974, pp.98-102.

[RAV.73] V.K. RAVINDRAN, T. THOMAS, "Characterization of multiple microprocessor networks", COMPCON 73, Palo Alto, Calif., March 1973, pp.133-137.

[REY.74] G. REYLING, "Performance and control of microprocessor systems", Computer Design, 1 March 1974, pp.81-86.

[RYA.74] T.A. RYAN, B. RYAN, E.G. COFFMAN, "MARKOV models of parallelism in loading and execution of a single process", Computer architectures and networks ; Modelling and evaluation, Ed. by E. GELEMBE and R. MAHL, North Holland/American, August 1974.

[SAS.75] K.V. SASTRY, R.Y. KAIN, "On the performance of certain multi-processor computer organizations", IEEE Trans. on Computer, vol.C-24, n°11, November 1975, pp.1066-1074.

[SPI.76] J. SPIRN, "Distance string models for program behavior", Computer, November 1976, pp.14-20.

[STE.76] W.J. STEWART, "MARCA : MARKOV chaîn analyser", IRISA, Publication Interne N°45, Juin 1976.

[STR.70] W.D. STRECKER, "Analysis of the instruction execution rate in certain computer structures", Ph.D. dissertation, Carnegie-Mellon University, Pittsburgh, Penn., September 1970.

[TAY.72] D.S. TAYLOR et al, "Memory interface simulator : a computer design aid", NASA CR 128996, 6 Octobre 1972.

[WIL.72] T.G. WILLIAMS et al, "Computer system simulation and analysis", NASA CR 61376, 6 Mars 1972.

[WIL.73] T.G. WILLIAMS et al, "Optimum spaceborne computer system design by simulation", Conference AGARD n°114.

[WUL.74] W. WULF et al, "HYDRA : the Kernel of a multiprocessor operating system", Communication of the ACM, June 1974, vol.17, n°6, pp.337-345.

TABLE DES MATIERES

CHAPITRE V. MODÈLE D'UNE STRUCTURE MULTIOPÉRATEUR SANS ÉCHANGE DIRECT ENTRE LES MODULES 99
