Etude et amélioration d'un algorithme d'allocation d'espace sur disques

Jean-Paul Machefaux

THESE

présentée à

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

par

pour obtenir le grade de

Docteur de troisième cycle

«INFORMATIQUE»

Jean-Paul MACHEFAUX

Etude et amélioration d'un algorithme d'allocation d'espace sur disques.

Thèse soutenue le 22 mai 1974 devant la Commission d'Examen :

Président : L. BOLLIET

Examinateurs : S. KRAKOWIAK
M. TERRENOIRE

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

LISTE DES PROFESSEURS

Président : Monsieur Michel SOUTIF
Vice-Président : Monsieur Gabriel CAU

PROFESSEURS TITULAIRES

MM.	ANGLES D'AURIAC Paul	Mécanique des fluides
	ARNAUD Georges	Clinique des maladies infectieuses
	ARNAUD Paul	Chimie
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale
	BENOIT Jean	Radioélectricité
	BERNARD Alain	Mathématiques Pures
	BESSON Jean	Electrochimie
	BEZES Henri	Chirurgie générale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Pathologie médicale
	BONNIER Etienne	Electrochimie Electrométallurgie
	BOUCHERLE André	Chimie et Toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques Appliquées
	BRAVARD Yves	Géographie
	BRISSONNEAU Pierre	Physique du solide
	BUYLE-BODIN Maurice	Electronique
	CABANAC Jean	Pathologie chirurgicale
	CABANEL Jean	Clinique rhumatologique et hydrologie
	CALAS François	Anatomie
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et Toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Oto-Rhino-Laryngologie
	CHATEAU Robert	Thérapeutique
	CHENE Marcel	Chimie papetière
	COEUR André	Pharmacie chimique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie Pathologique
	CRAYA Antoine	Mécanique

Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de Dermatologie et Syphiligraphie
	FAU René	Clinique neuro-psychiatrique
	FELICI Noël	Electrostatique
	GAGNAIRE Didier	Chimie physique
	GALLISSOT François	Mathématiques Pures
	GALVANI Octave	Mathématiques Pures
	GASTINEL Noël	Analyse numérique
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques Pures
	GIRAUD Pierre	Géologie
	KLEIN Joseph	Mathématiques Pures
Mme	KOFLER Lucie	Botanique et Physilogie végétale
MM.	KOSZUL Jean-Louis	Mathématiques Pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre-Jean	Mathématiques appliquées
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LOUP Jean	Géographie
Mle	LUTZ Elisabeth	Mathématiques Pures
MM.	MALGRANGE Bernard	Mathématiques Pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Seméiologie médicale
	MASSEPORT Jean	Géographie
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NEEL Louis	Physique du solide
	OZENDA Paul	Botanique
	PAUTHENET René	Electrotechnique
	PAYAN Jean-Jacques	Mathématiques Pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET René	Servomécanismes
	PILLET Emile	Physique industrielle
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REULOS René	Physique industrielle
	RINALDI Renaud	Physique
	ROGET Jean	Clinique de pédiatrie et de puériculture
	SANTON Lucien	Mécanique
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SILBERT Robert	Mécanique des fluides

MM.	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLAND François	Zoologie
	VALENTIN Jacques	Physique nucléaire
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
M.	VERAIN André	Physique
Mme	VEYRET Germaine	Géographie
MM.	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM.	BULLEMER Bernhard	Physique
	HANO JUN-ICHI	Mathématiques Pures
	STEPHENS Michaël	Mathématiques appliquées

PROFESSEURS SANS CHAIRE

MM.	BEAUDOING André	Pédiatrie
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BONNETAIN Lucien	Chimie minérale
Mme	BONNIER Jane	Chimie générale
MM.	CARLIER Georges	Biologie végétale
	COHEN Joseph	Electrotechnique
	COUMES André	Radioélectricité
	DEPASSEL Roger	Mécanique des fluides
	DEPORTES Charles	Chimie minérale
	GAUTHIER Yves	Sciences biologiques
	GAVEND Michel	Pharmacologie
	GERMAIN Jean-Pierre	Mécanique
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	HACQUES Gérard	Calcul numérique
	JANIN Bernard	Géographie
Mme	KAHANE Josette	Physique
MM.	MULLER Jean-Michel	Thérapeutique
	PERRIAUX Jean-Jacques	Géologie et Minéralogie
	POULOUJADOFF Michel	Electrotechnique
	REBECQ Jacques	Biologie (CUS)
	REVOL Michel	Urologie
	REYMOND Jean-Charles	Chirurgie générale
	ROBERT André	Chimie papetière
	DE ROUGEMONT Jacques	Neurochirurgie
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIBILLE Robert	Construction mécanique
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

Mle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBLARD Pierre	Dermatologie
	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Yves	Chimie
	BEGUIN Claude	Chimie organique
	BELORIZKY Elie	Physique
	BENZAKEN Claude	Mathématiques appliquées
	BILLET Jean	Géographie
	BLIMAN Samuel	Electronique (EIE)
	BLOCH Daniel	Electrotechnique
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BOUCHET Yves	Anatomie
	BOUVARD Maurice	Mécanique des fluides
	BRODEAU François	Mathématiques (IUT B)
	BRUGEL Lucien	Energétique
	BUISSON Roger	Physique
	BUTEL Jean	Orthopédie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CHIBON Pierre	Biologie animale
	COHEN-ADDAD Jean-Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie médicale
	CONTE René	Physique
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	DURAND Francis	Métallurgie
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	GENSAC Pierre	Botanique
	GIDON Maurice	Géologie
	GRIFFITHS Michaël	Mathématiques appliquées
	GROULADE Joseph	Biochimie médicale
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et Médecine préventive
	IDELMAN Simon	Physiologie animale
	IVANES Marcel	Electricité
	JALBERT Pierre	Histologie
	JOLY Jean-René	Mathématiques Pures
	JOUBERT Jean-Claude	Physique du solide
	JULLIEN Pierre	Mathématiques Pures
	KAHANE André	Physique générale
	KUHN Gérard	Physique
	LACOUME Jean-Louis	Physique
Mme	LAJZEROWICZ Jeannine	Physique
MM.	LANCIA Roland	Physique atomique
	LE JUNTER Noël	Electronique
	LEROY Philippe	Mathématiques
	LOISEAUX Jean-Marie	Physique nucléaire
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LUU DUC Cuong	Chimie organique
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et Médecine préventive
	MARECHAL Jean	Mécanique

MM.	MAYNARD Roger	Physique du solide
	MICHOULIER Jean	Physique (IUT A)
	MICOUD Max	Maladies infectieuses
	MOREAU René	Hydraulique (INP)
	NEGRE Robert	Mécanique
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B)
	PEFFEN René	Métallurgie
	PELMONT Jean	Physiologie animale
	PERRET Jean	Neurologie
	PERRIN Louis	Pathologie expérimentale
	PFISTER Jean-Claude	Physique du solide
	PHELIP Xavier	Rhumatologie
Mle	RIERY Yvette	Biologie animale
MM.	RACHAIL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RENAUD Maurice	Chimie
	RICHARD Lucien	Botanique
Mme	RINAUDO Marquerite	Chimie macromoléculaire
MM.	ROMIER Guy	Mathématiques (IUT B)
	SHOM Jean-Claude	Chimie générale
	STIEGLITZ Paul	Anesthésiologie
	STOEBNER Pierre	Anatomie pathologique
	VAN CUTSEM Bernard	Mathématiques appliquées
	VEILLON Gérard	Mathématiques appliquées (INP)
	VIALON Pierre	Géologie
	VOOG Robert	Médecine interne
	VROUSSOS Constantin	Radiologie
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES ASSOCIES

MM.	BOUDOURIS Georges	Radioélectricité
	CHEEKE John	Thermodynamique
	GOLDSCHMIDT Hubert	Mathématiques
	SIDNEY STUARD	Mathématiques Pures
	YACOUD Mahmoud	Médecine légale

CHARGES DE FONCTIONS DE MAITRES DE CONFERENCES

Mme	BERIEL Hélène	Physilogie
Mme	RENAUDET Jacqueline	Microbiologie

Fait le 30 mai 1972.

Je tiens à remercier Monsieur Louis BOLLIET qui a bien voulu diriger mon travail et qui m'a fait l'honneur d'être Président du Jury, ainsi que Messieurs TERRENOIRE et KRAKOWIAK qui ont accepté de faire partie de ce Jury.

Mes remerciements vont en particulier à Monsieur Claude BOKSENBAUM qui m'a guidé et conseillé tout au long de mon travail et à Monsieur Paul BERARD, dont la coopération et les remarques m'ont été très utiles.

Je n'oublie pas non plus ceux et celles qui ont contribué à la réalisation matérielle de cet ouvrage.

TABLE DES MATIERES

ETUDE ET AMELIORATION D'UN ALGORITHME D'ALLOCATION D'ESPACE SUR DISQUES

INTRODUCTION

Le développement récent, dans les années 1950-1970 des ordinateurs est dû en grande partie aux progrès dans la miniaturisation des composants électroniques. Les lampes ont été remplacées par les transistors, puis, plus récemment par les circuits monolithiques.

Les périphériques électromécaniques ont une "inertie" très grande par rapport à l'unité centrale ou à la mémoire centrale. En effet, alors que le temps de transfert d'une information élémentaire est sensiblement le même (de 1 à 10 µs), le temps de positionnement des périphériques reste élevé (de l'ordre de 10 ms) ce qui donne un rapport de 1 à 10^4 entre le temps d'accès à une information élémentaire, suivant qu'elle se trouve en mémoire centrale ou sur disque.

Les progrès déterminants dans l'utilisation maximale de l'unité centrale ont tous eu pour objectif de compenser cet écart.

Afin de réduire le temps d'attente de l'unité centrale, on a cherché à diminuer le nombre d'accès aux périphériques à l'aide d'un "tampon" (ou buffer) en mémoire centrale pouvant contenir plusieurs enregistrements. Lorsque les fichiers sont utilisés de manière séquentielle, c'est par exemple le cas de fichiers entrant ou sortant d'un compilateur, le tampon est très efficace puisqu'il contient les prochaines données utilisées.

La taille des tampons est limitée par la recherche d'un compromis entre la mémoire utilisée et le nombre d'entrées-sorties effectuées, qui influe sur le temps d'exécution.

Néanmoins, lorsque les accès aux fichiers sont indépendants, ce qui se produit dans le cas d'une utilisation conversationnelle ou d'une base de données en accès direct, le rôle du tampon est beaucoup moins efficace.

commencé par effectuer certaines opérations, en simultanéïté avec l'unité centrale, comme la lecture des cartes, ou l'impression des résultats.

Exemple :

Les "symbionts" (symbolic input-output), ou "parasites" d'EXEC 2 sur UNIVAC 1107 puis le "spooling" (simultaneous peripheral operations on line) des OS sur IBM 360.

Cette simultanéïté permet aussi d'améliorer le rendement des tampons : pendant qu'un tampon est lu, un autre est en cours d'écriture, et ce dernier pourra être lu, pendant que l'autre est en cours d'écriture. Cette méthode se généralise pour n tampons organisés d'une manière cyclique.

Avec les nouveaux dispositifs permettant la multiprogrammation, les opérations simultanées se sont généralisées.

Le sujet de cette thèse se limite à la réduction du nombre d'entrées-sorties dans le mécanisme d'allocation d'espace sur disques des systèmes SIRIS 7 et SIRIS 8 des ordinateurs CII 10070 et IRIS 80.

Trois méthodes se présentaient pour mener à bien cette étude : l'expérimentation sur le système réel, une étude analytique ou la simulation.

L'expérimentation sur le système réel imposait de modifier le système et risquait d'être fort coûteuse en temps machine. De plus, la méthode d'allocation a été changée au cours de ce travail, et cela aurait imposé de recommencer. Il aurait aussi fallu programmer en langage machine. Pour ces raisons, cette méthode n'a pas été utilisée.

L'obtention des renseignements désirés par la méthode analytique nous a semblé très complexe. Elle aurait entraîné une réduction des données d'entrées dont on n'était pas sûr qu'elle soit justifiée (N1). L'examen de ces données a confirmé cette hypothèse, comme on le verra plus tard.

Disposant de mesures prises sur le système réel, il nous était possible de construire un modèle de simulation de la gestion de l'espace sur disque, alimenté par ces données, et c'est la technique qui a été retenue.

L'élaboration du modèle sera faite en quatre phases (cf : (A1), (B3), (B4), (C1)

a) Formulation du problème

Il s'agit de définir le problème à résoudre. Les objectifs à atteindre sont déterminés et replacés dans un contexte général. Il faut en outre proposer les critères qui permettront de juger jusqu'à quel point les objectifs sont atteints (C1). Les critères sont une évaluation quantifiée des objectifs.
Lors de la définition des objectifs, il est souvent nécessaire de faire des hypothèses quant au fonctionnement du système que l'on veut étudier. C'est ici que la collecte des données du monde réel est utile, sinon indispensable, pour permettre de vérifier en partie les hypothèses émises dans un but de simplification. La non concordance des données du monde réel et des hypothèses entraînera une réévaluation des objectifs à remplir.

b) Construction du modèle et sa validation

Une fois en possession des objectifs que l'on veut atteindre, et des données du problème, il faut construire le modèle, qui est une simplification du phénomène réel. Cette retranscription schématisée de la réalité ne va pas sans quelque distorsion. Pour affirmer que le modèle représente correctement le comportement du système réel, il faut procéder à sa validation.
Celle-ci consiste à "comparer" le fonctionnement du système réel en présence d'une certaine demande, avec celui du modèle en présence de la même demande. La validation est faite sur un domaine précis, et c'est une étape importante, qui permet de passer à la phase suivante d'extrapolation, en fournissant des résultats significatifs.

La comparaison peut se limiter à l'obtention de résultats du modèle ayant le même ordre de grandeur que ceux du système réel. Mais la validation est parfois hasardeuse, par exemple si on ne dispose pas encore du système réel ; il faut alors procéder par analogie, en comparant le système à des systèmes semblables, ou procéder par intuition en dernier recours.

c) Extrapolation

C'est l'étape d'obtention et d'analyse des résultats de la simulation. On examine la réponse du modèle à la variation des conditions d'entrée (variables d'entrée) d'une part, des conditions de fonctionnement (paramètres du modèle) d'autre part.

Dans notre cas, nous nous sommes surtout intéressés à la variation des conditions de fonctionnement. Nous ne disposons pas d'assez de données relatives aux conditions d'entrée pour pouvoir prédire comment elles varieront. Les résultats de la simulation sont analysés, et on essaye de les expliquer à l'aide, par exemple, d'un modèle analytique ou déterministe très schématisé.

d) Conclusions

Les résultats de la simulation nous permettent alors de proposer certaines modifications à apporter au système réel et d'estimer quelles sont les valeurs des paramètres les plus efficaces.

Une dernière étape de vérification peut avoir lieu : une fois le système réel en fonctionnement, avec les modifications proposées par les résultats de la simulation, on compare les nouvelles conditions de fonctionnement avec les prévisions du modèle, pour examiner si elles concordent.

Le plan de cette étude est le suivant :

- Le mécanisme d'allocation d'espace sur disque de SIRIS 7 / SIRIS 8 est décrit et replacé dans le contexte général des autres systèmes afin de définir les objectifs à atteindre (Chapitre I)

- Les variables d'entrée du modèle, issues de mesures réelles faites sur SIRIS 7 sont analysées (Chapitre II).

- Le modèle de l'algorithme d'allocation est décrit ainsi que sa validation (Chapitre III).

Finalement les améliorations proposées sont modelées et des expériences de simulation les évaluant sont décrites (Chapitre IV et V).

CHAPITRE - I

DESCRIPTION DU SYSTEME D'ALLOCATION

I-0 - INTRODUCTION

L'allocation d'espace sur disque est un problème qui se présente dans tous les grands systèmes d'exploitation conçus de 1960 à 1970.

Exemples : OS, TSS, MULTICS, SIRIS 7 et SIRIS 8.

Ces systèmes utilisent généralement le concept de "fichiers" : les travaux à exécuter sont considérés comme des fichiers, et créés comme tels ; de plus, il utilisent et fabriquent d'autres fichiers. Le rôle du système d'exploitation est de gérer ces fichiers, de leur accorder les ressources nécessaires en temps et en espace. La tendance actuelle est de faire maintenir les fichiers, publics comme privés, par le système d'exploitation.

Sous les systèmes SIRIS 7/8, les fichiers peuvent résider sur des disques, soit dans un espace privé, soit dans un espace banalisé, commun au système et aux divers utilisateurs (disques système). Les demandes d'allocation ou de restitution d'espace sont centralisées sur un programme du système d'exploitation, qui effectue l'allocation et la libération de l'espace.

Le disque système est l'espace sur disque réservé aux demandes faites par le système d'exploitation (par exemple les symbionts d'entrée-sortie, les compilateurs, le moniteur de travaux) ou par des utilisateurs désirant un espace sur disque banalisé (non privé). Dans le cas d'une demande système (c'est à-dire adressée aux disques système), le choix du disque, et l'implantation des blocs alloués sont déterminés par le système d'allocation, dans un but d'optimisation alors qu'une demande privée précise le disque sur lequel les blocs devront être alloués.

La gestion des disques privés n'a pas la même exigence d'efficacité que celle des disques système, partagés entre tous les utilisateurs qui doivent être bien gérés pour jouer efficacement leurs rôles de distributeurs d'espace secondaire. La suite de ce travail ne concernera que la demande système.

Nous allons étudier différentes méthodes permettant de représenter et d'allouer cet espace ; l'allocation peut se faire de deux manières :

a) Pré-allocation

On accorde à l'avance aux demandeurs une zone bien déterminée. C'est ainsi que les composants du système d'exploitation, les bibliothèques de programme ont leur espace alloué.

Exemple : Le système OS : les fichiers de travail ont leur espace accordé, avant l'utilisation, au moyen des cartes de contrôle (carte "DD") (I5).

b) Allocation dynamique

On accorde l'espace au moment où on en a un besoin effectif.

Exemples :-Le système CP (Control Program) : les opérations de "spooling" permettent d'allouer dynamiquement des blocs de 800 octets (I2)

-Le système SIRIS 7/8 : la taille des fichiers peut varier dynamiquement par allocations ou restitutions de blocs multiples de un quantum. Un quantum vaut 8 K octets (S1).

Ces deux méthodes ne s'excluent pas mutuellement, et elles peuvent se combiner. On alloue fréquemment un certain espace à l'avance, et cet espace peut être agrandi, dynamiquement s'il s'avère insuffisant.

Exemple 1 : Dans le système OS, le paramètre des cartes de contrôle allouant de l'espace secondaire permet de préciser de combien on veut agrandir l'espace alloué, au cas où la demande serait insuffisante. On peut aussi faire une allocation dynamique à l'intérieur d'une allocation statique ou l'inverse.

Exemple 2 : Dans le système CP, les fichiers de "spooling" possèdent une zone

entre tous les utilisateurs. De même, chaque utilisateur CMS a une portion fixe de disque, mais ses fichiers l'occupent dynamiquement.

En général, on fera une allocation statique pour des fichiers à "longue durée de vie", et une allocation dynamique pour des fichiers "temporaires", par exemple des fichiers de travail, bien que la frontière entre le temporaire et le durable soit mal définie.
Une allocation statique est plus simple qu'une allocation dynamique mais conduit à une utilisation moins bonne de l'espace.
Il ne suffit pas de savoir quand allouer, il importe autant de savoir comment allouer. Les méthodes d'allocation sont liées à la représentation de l'espace sur disque et à l'usage que l'on veut faire de cet espace. On distinguera d'une part la représentation de la structure des fichiers (espace occupé), d'autre part la représentation de l'espace des disques (division entre espace libre et occupé). Les opérations à effectuer sur ces deux structures n'étant pas les mêmes.

Sur un fichier, on peut faire les opérations suivantes :

- créer un fichier
- changer le nom d'un fichier
- ajouter, changer ou détruire les informations
- recopier, trier, ou combiner plusieurs fichiers.

Si l'accès aux fichiers se fait uniquement par requête au système, celui-ci peut contrôler les accès en lecture (pour conserver le secret de l'information) ou en écriture (pour conserver l'invariance de l'information). En revanche, le système doit fournir les divers modes d'accès ou les diverses organisations souhaitées.

De ces opérations sur les fichiers, il résulte des sous-opérations sur l'espace des disques :

- allocation ou restitution d'espace
- restructuration de l'espace.

Un aspect important de l'allocation est sa fiabilité : en général, plus

I-1 - REPRESENTATION DE LA STRUCTURE DES FICHIERS

On représente généralement un ensemble de fichiers par un système arborescent (D1). Au sommet de l'arbre est situé le super-catalogue qui répertorie les fichiers existants ; il y a ensuite des fichiers catalogues de fichiers (exemple : le catalogue des membres d'un fichier partitionné) pour arriver au fichier lui-même ; les feuilles de l'arbre représentent l'information contenue dans le fichier. Nous ne nous intéressons ici qu'à la seule structure du fichier connue par le système d'exploitation. La description de cette structure sera plus ou moins 'fine', selon que l'utilisateur désire gérer lui-même l'espace qui lui est alloué, on s'en remet au système de gestion de fichiers.

Nous distinguons deux manières de représenter l'espace occupé par un fichier :

- soit par chaînage interne des blocs alloués.
- soit par une information d'implantation séparée du contenu des fichiers.

1°) - Représentation par chaînage interne

L'allocation est faite par blocs qui peuvent être de tailles différentes, et ces blocs sont chaînés entre eux. Un bloc plein n'est écrit sur disque qu'à la demande suivante lorsque l'adresse du bloc suivant est déterminée, ou à la fin du travail. L'inconvénient de cette méthode est que l'on doit relire tout un fichier pour le supprimer. Il faut aussi prévoir un mécanisme de protection des pointeurs si l'utilisateur a accès aux pointeurs.

Exemple : Dans le système CP le "spooling" se voit réserver une zone d'espace secondaire bien déterminée. A l'intérieur de cette zone, l'espace est alloué dynamiquement, à partir de trois têtes de listes correspondant aux lecteurs, perforateurs et imprimantes. L'allocation est faite par blocs de 800 octets. Pour chaque fichier, les blocs sont chaînés entre eux.

2°) - Représentation par une information d'implantation séparée

Pour chaque fichier il existe un descripteur,ou catalogue, de celui-ci. Il contient le nombre de blocs occupés, l'adresse et la taille de ces blocs.

a) - Représentation par une table de bits (technique du "bit-mapping")

L'espace est divisé en quanta égaux, et chaque quantum est représenté par un bit dans la table. Un bit à 1 indiquera que le bloc correspondant est occupé.

Exemple : Le système HASP (Houston Automatic Spooling Priority (H1)). C'est un système de "spooling" qui travaille sur le système OS, en remplaçant certaines de ses fonctions. Grâce à une table de bits d'allocation de l'espace secondaire, qui réside en mémoire centrale, le système HASP alloue dynamiquement l'espace secondaire. Cette table représente l'ensemble de tous les groupes de pistes (cylindres logiques) utilisables sur tous les "volumes" de spooling d'accès direct. Un cylindre est représenté par un bit. La valeur 1 indique que le cylindre est allouable ; la valeur 0 indique que le cylindre est déjà alloué, ou n'est pas allouable. Pour chaque travail, il existe 2 tables qui sont identiques à la première : il y en a une pour les fichiers d'entrée, et une pour les fichiers de sortie (imprimantes et perforateurs). Si un bit à la valeur 1 sur ces tables, cela veut dire que le cylindre correspondant appartient au travail en cours. Pour libérer l'espace d'un travail il suffit de faire l'union logique (opération OR) de ces 2 tables avec la table de l'espace total. L'allocation et la libération de l'espace se font sans aucune entrée-sortie.

Remarque :L'unité d'allocation de l'espace peut être inférieur au cylindre ; en effet, à chaque table d'allocation particulière est associé un mot qui indique la dernière piste allouée sur le dernier cylindre alloué. Lors d'une allocation, on commence par allouer les pistes encore libres sur ce cylindre, avant d'allouer un nouveau cylindre au fichier. Il existe par ailleurs un chaînage des blocs alloués, interne aux blocs. Ces tables de bits sont chargées d'effectuer l'allocation et la restitution de l'espace sur disque d'une manière rapide (voir Figure 1).

b) - Représentation par un catalogue des blocs occupés

C'est la méthode la plus utilisée. Elle permet avec une seule structure de représenter d'une part la structure du fichier, d'autre part l'emplacement physique des zones allouées. Le système de gestion de fichiers se charge de mettre à jour le catalogue, d'allouer et de restituer l'espace, d'effectuer les opérations d'entrée-sortie.

Exemple 1 : Dans le système OS chaque disque est représenté par sa table d'occupation qui réside sur le disque même. Cette table est décomposée en "blocs de contrôle" de fichiers (un pour chaque fichier) et il existe un bloc de contrôle pour tout l'espace libre du disque. Chaque bloc de contrôle est décomposé en "formats". A cause de la gestion dynamique de ces blocs de contrôle et formats, les formats sont chaînés entre eux. Il existe plusieurs types de formats : le format "4" décrit les caractéristiques du disque, la table d'occupation et les noms des fichiers catalogués sur ce disque. Le format "5" décrit l'espace libre du disque : on peut décrire jusqu'à 26 zones dans un format "5". Le format "1" décrit un fichier et les 3 premières zones occupées (voir Figure 2).

Exemple 2 : Dans les systèmes SIRIS 7/8, la structure d'un disque est décrite dans l'entête de ce disque. L'entête comprend, entre autres, un catalogue des tables des fichiers, et chaque label décrit, en particulier l'implantation du fichier sur le disque. C'est une énumération des zones occupées avec leur taille et leur adresse de début. Par construction un fichier est limité quant à son étendue et son fractionnement.

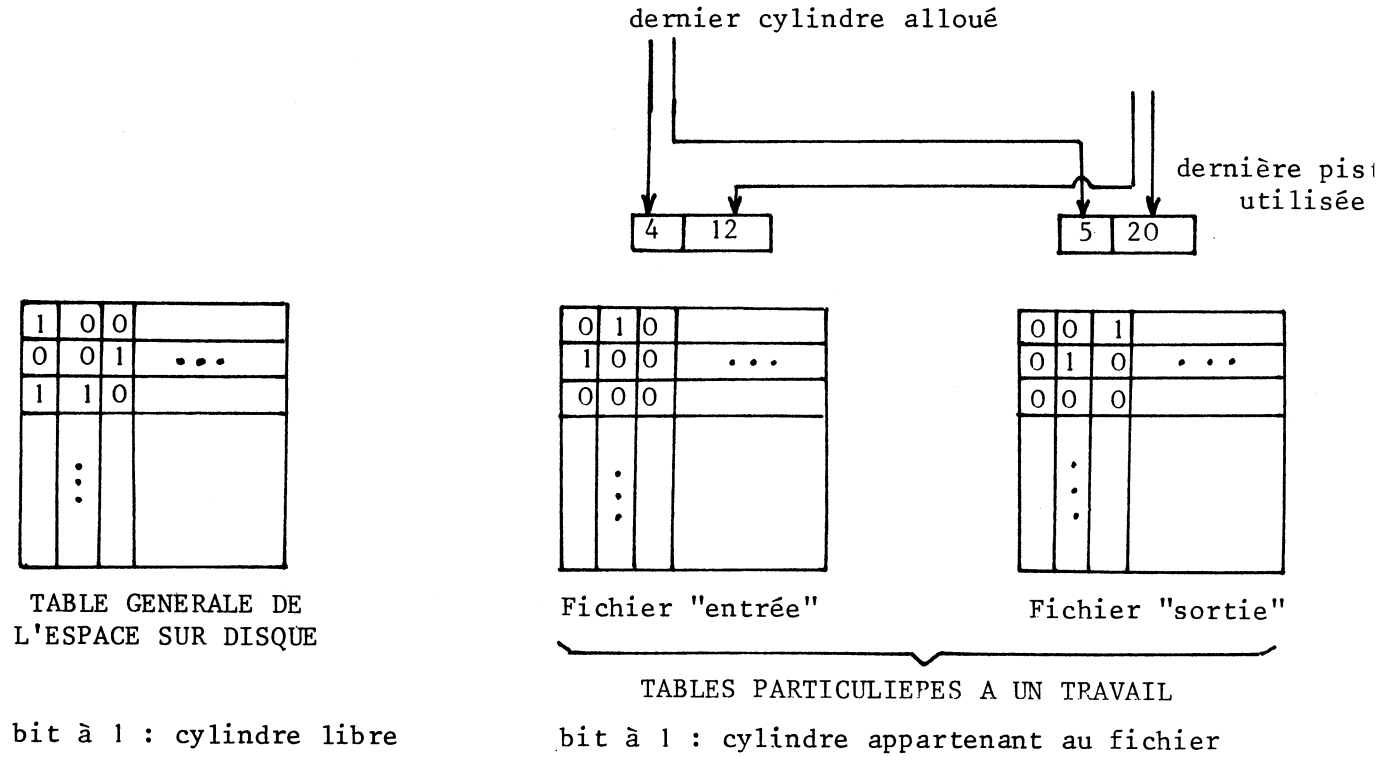


FIGURE 1 : Le système HASP : Tables d'occupation de l'espace sur disque, résidentes en mémoire centrale.

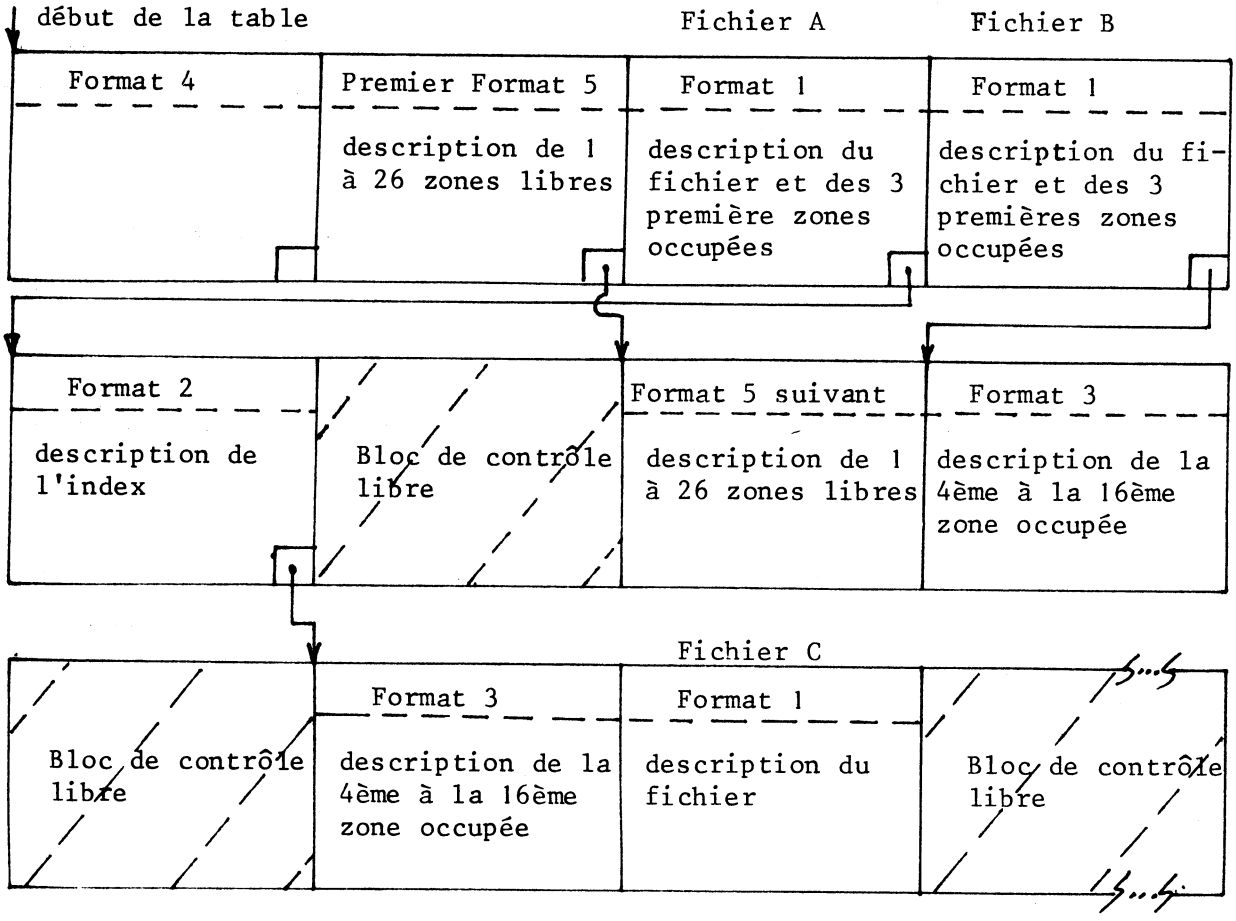


FIGURE 2 : Le système OS : tables d'occupation de l'espace sur disque, résidentes sur disque.

I-2 - REPRESENTATION DE L'ESPACE DES DISQUES

L'allocation d'espace sur disque doit répondre à quatre impératifs plus ou moins contradictoires :

- la rapidité de l'algorithme
- l'encombrement minimum, à la fois de l'algorithme et de ses tables de gestion associées
- la compacité de l'espace libre qui permet un meilleur remplissage en cas de "saturation"
- la sécurité.

Le premier impératif nous amène à structurer la représentation de l'espace libre, donc à augmenter l'encombrement ;afin d'accélérer la recherche d'une zone libre de taille donnée, on pourra disposer d'une série de pointeurs, ou d'une série de listes de zones libres pour chaque taille (méthode des listes orthogonales) (K1,K2,M1). On doit aussi amener les tables d'occupation en mémoire centrale, afin d'éviter des entrées sorties inutiles sur disque.

Le deuxième impératif nous amène à simplifier la représentation de l'espace libre (moins d'encombrement) et à amener les tables d'occupation sur disque, ou à taille égale la place est moins chère qu'en mémoire centrale.

Le troisième impératif qui dépend essentiellement de la méthode d'allocation (cf. I.4) se fait généralement au détriment de la rapidité.

Le quatrième impératif nous impose de mettre à jour assez fréquemment les tables sur disque ; l'idéal serait de les mettre à jour après chaque allocation ou restitution d'espace. Il n'y aurait pas alors de décalage entre la structure des tables sur disque et leur copie en mémoire centrale ; en cas de panne de courant par exemple, il ne serait pas utile alors de sauvegarder en catastrophe les tables d'occupation. Mais ces mises à jour obligent à faire de nombreuses entrées sorties pour lire et réécrire les tables, ce qui va à l'encontre du premier impératif.

1) - Représentation d'un espace de blocs de taille fixe

Ce cas se produit notamment lorsque le système d'exploitation est un système à pagination et où l'espace secondaire réservé à la pagination et la mémoire centrale sont divisés en pages de même dimension. La méthode la plus courante consiste alors à représenter l'espace secondaire par une table de bits, chaque bit représentant une page.

Exemple : Dans le système CP, la mémoire secondaire (tambours et disques) réservée au système CP est partagée en pages de 4 K octets ; il existe en mémoire centrale une table de bits représentant l'ensemble des pages, suivant la technique du bit mapping. L'allocation se fait en balayant la table pour trouver le premier bit de valeur zéro, qui est basculé à 1, indiquant ainsi que la page correspondante est allouée. La restitution de l'espace alloué à un travail (machine virtuelle) n'a lieu qu'à la fin du travail. Une allocation plus fine est possible si, entre étapes de travail (commandes) le système de pagination est prévenu (L1).

2) - Représentation d'un espace de blocs de longueur variable

Cette méthode permet une plus grande variété dans les demandes et autorise l'allocation de zones physiquement contigües. On retrouve les deux mêmes méthodes que pour représenter la structure des fichiers. Néanmoins, lorsque l'unité d'allocation est importante, on peut encore utiliser la méthode précédente.

a) - Représentation de l'espace disque par chaînage interne

Chaque bloc libre est chaîné au bloc suivant. Il est nécessaire de réserver de l'espace pour mettre le chaînage. Lorsque l'espace-disque est très "émietté", la recherche d'une zone de taille donnée risque d'être longue. Aussi plutôt que de disposer d'une seule tête de liste indiquant le premier bloc libre (par adresses disque croissantes), on pourra créer une série de liste pour chaque taille, ou pour les tailles les plus fréquemment allouées.

Exemple : Le système AED (cf. (R1)).

Ce système gère un espace de blocs libre en mémoire centrale mais son principe peut être appliqué à l'espace disque.

Tous les blocs libres d'une taille T sont chaînés en liste, et toutes les têtes de listes sont chaînées suivant les tailles croissantes (cf. Figure 3). Un autre chaînage indépendant relie les blocs par adresses croissantes afin de ressouder les blocs contigus, s'il y a lieu.

La recherche d'un bloc libre peut être accélérée si des pointeurs existent pour les tailles les plus fréquemment utilisées. Tous ces pointeurs sont logés dans les blocs libres eux-mêmes.

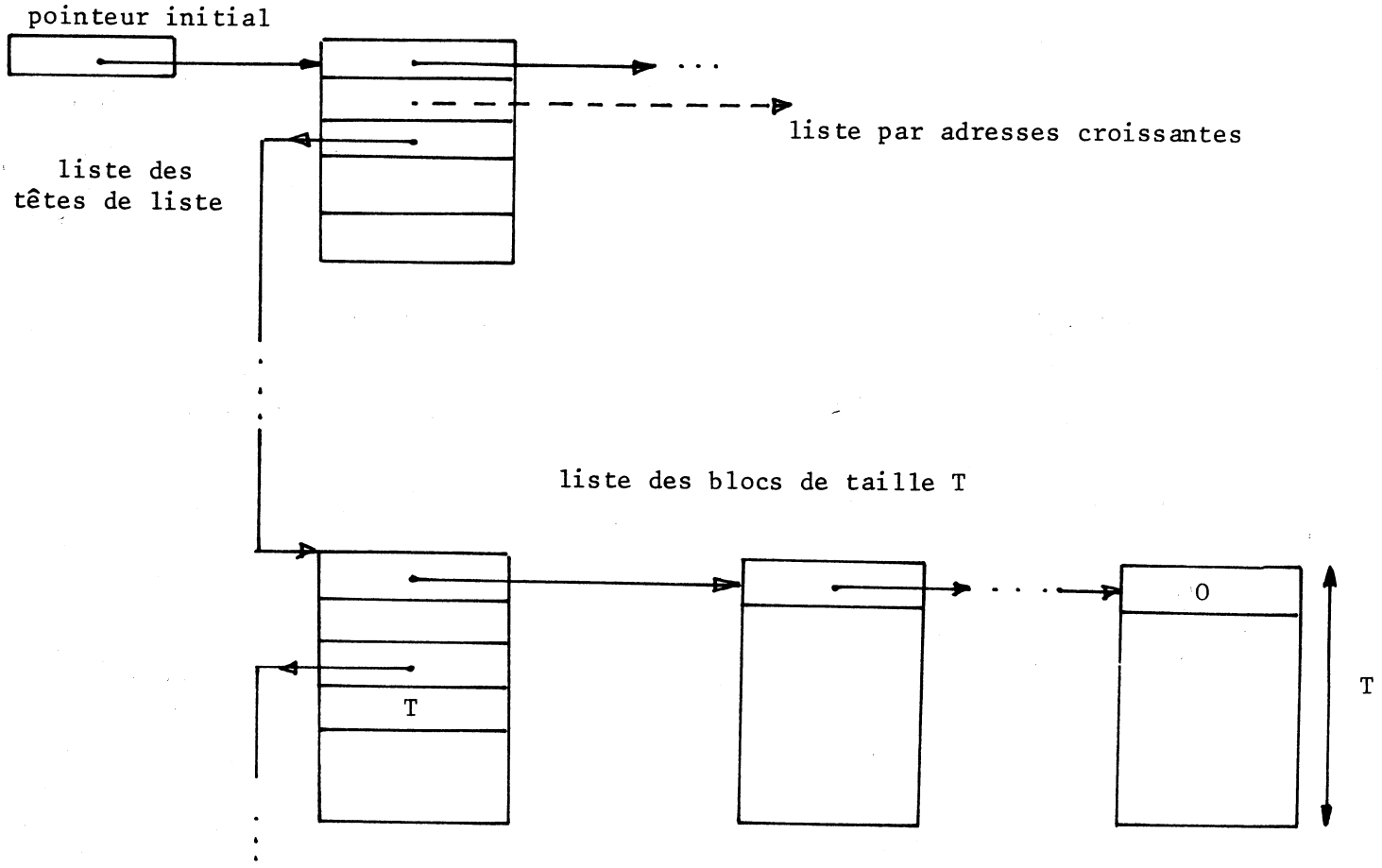


FIGURE 3 : Structure de l'espace libre du système AED

b) - Représentation par un catalogue

Exemple : Dans le système OS l'espace libre d'un disque est répertorié dans des formats, chaînés entre eux, pouvant contenir chacun de une à vingt six descriptions de zones libres. Les zones sont organisées suivant une structure de file ordonnée par adresses disque croissantes des zones. C'est en effet un point important si l'on veut restituer les zones libérées en les ressoudant aux zones libres contigües, d'une manière efficace.

A chaque allocation, ou presque, il est nécessaire de restructurer la file : si une zone contigüe entière est allouée, on retire cette zone de la file, et l'espace ainsi libéré est "bouché" par la translation des zones d'adresses supérieures.

On restitue les zones libérées en conservant la séquence croissante des adresses de début de zone. Pour voir s'il est possible de ressouder le bloc libéré aux zones libres proches, il suffit de comparer l'adresse de début du bloc libre postérieur. Si elles coïncident, on ressoude les zones ensemble. Il faut encore restructurer la file à chaque libération ou presque.

Nous consacrerons le paragraphe suivant à la représentation de l'espace sur disque sour SIRIS 7/8, car cette représentation combine plusieurs méthodes.

I-3 - STRUCTURE DE L'ESPACE-DISQUE SOUS SIRIS 7 ET SIRIS 8

L'espace disque a été divisé en blocs de 8 K octets, appelés "quanta". Deux manières extrêmes de procéder à la gestion de l'espace sur disque se présentaient :

i) Laisser les tables d'occupation en mémoire centrale et faire l'allocation sans accès aux disques.

ii) Ne rien laisser en mémoire centrale, et à chaque allocation, accéder au disque concerné pour amener sa table d'occupation, la modifier, puis la réécrire sur disque.

La seconde utilisation a été choisie, car l'économie de mémoire centrale réalisée semblait compenser le supplément d'accès aux disques en résultant,et car cette solution offre plus de sécurité que la première puisque chaque disque porte sa table d'occupation courante.

Cependant, cette solution, appelée mécanisme général, a été modifiée dans le sens de la première.

Un mécanisme particulier joue le rôle d'intermédiaire dans l'allocation de blocs d'un quantum, qui est très fréquente. Ce mécanisme répertorie, en mémoire centrale, un stock de tels blocs qui sont alloués sans accès aux disques.
C'est le mécanisme appelé "Quick-cell" (voir figure 5).

1) Tables du mécanisme général

a) La table d'occupation résidente sur disque :
L'espace sur disque réservé au système est partitionné ; chaque partition peut comprendre un ou plusieurs disques de même type physique, un disque n'appartenant qu'à une seule partition. Chaque disque est divisé en une à six portions ; une portion est une zone d'espace disque contigu, de taille variable, représentée par sa table d'occupation, qu'on appellera TABLEDISQUE.
Les "TABLEDISQUES" résident sur le disque, dans le quantum 0. L'espace pris par une de ces tables est de 1 K octets.

Tableau des types physiques de disques :

à tête fixe	octets/secteur	capacité	temps d'accès
D I A D	1024	5.4 Mo	20.3 ms
amovible			
DIAM	360	6.2 Mo	87.5 ms
DIMAS	1024	25 Mo	87.5 ms
MD 25	1024	25 Mo	47.5 ms
MD 50	1024	50 Mo	47.5 ms
MD 100	1024	100 Mo	38.3 ms

Figure 4

Ces tables décrivent l'espace de chacune des portions. Les zones libres et occupées sont représentées par des bits mis respectivement à zéro et un. Suivant les cas, le quantum représente 8, 16 ou 32 K octets.
Un bref rappel est nécessaire pour expliquer la valeur représentative du bit. Initialement les disques avaient une capacité limitée à 25 millions d'octets, il y avait une TABLEDISQUE par disque qui était l'image de 3072 quanta soit à peu près 25 millions d'octets. Avec la mise en service de nouveaux disques de 50 millions d'octets, et bientôt de 100 millions d'octets, la capacité d'accueil de ces tables est insuffisante. Aussi a-t-on décidé soit de mettre plusieurs tables par dique, soit d'admettre que le quantum représenterait 8, 16 ou 32 K octets suivant les tables.

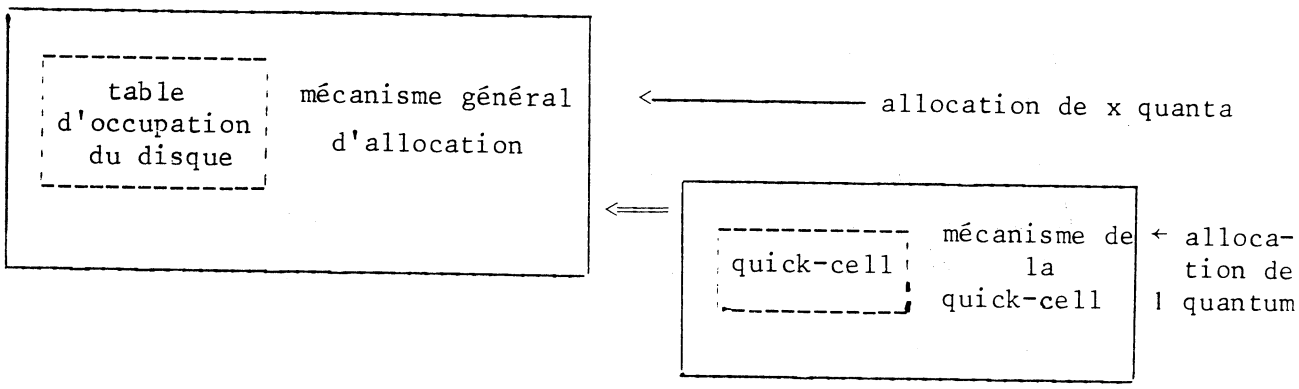
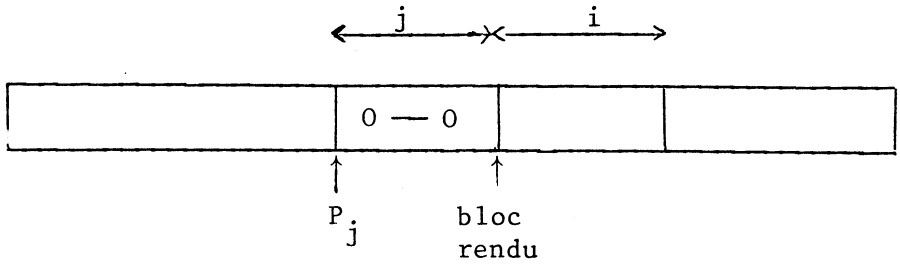


Figure 5
Mécanisme de la quick-cell

La table d'occupation d'un disque est un vecteur booléen où chaque bit représente 1 quantum. Chaque bloc libre de taille n quanta y est représenté par n zéros consécutifs.

Cette structure permet une fusion implicite, donc rapide, des zones libres contigües. Mais elle impose lors de l'allocation de parcourir cette table, ce qui peut être long, notamment si aucun bloc de la taille cherchée n'est présent. Aussi des index P_i limitent inférieurement ces recherches, pour les blocs de la taille i.

Pour que ces index indiquent exactement le premier bloc libre, il faut faire une recherche à la fois lors de l'allocation, ce qu'on doit faire de toutes façons, et éventuellement lors d'une fusion. En effet, lors de la restitution d'un bloc de taille i, immédiatement voisin du bloc pointé par P_j, la fusion entraîne la création



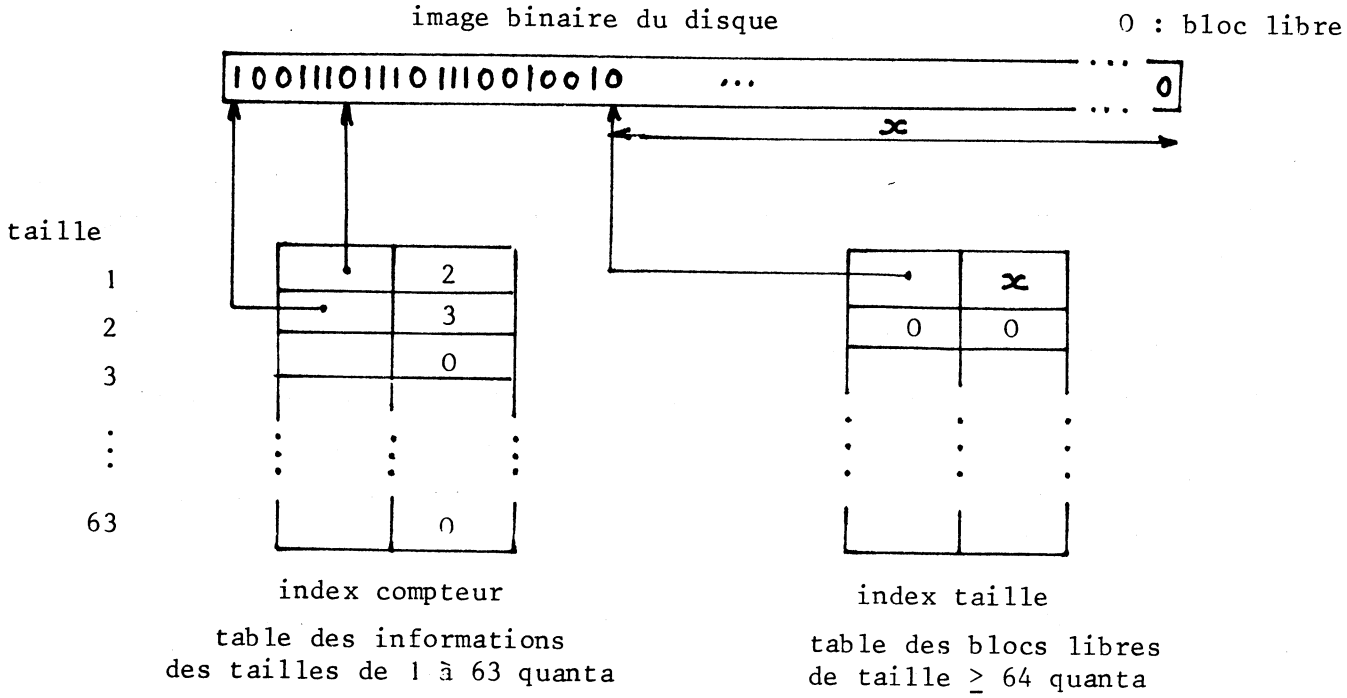
d'un bloc de taille i+j (P_{i+j} étant mis à jour par comparaison) mais la nouvell valeur de P_j doit faire l'objet d'une recherche.

L'algorithme de SIRIS 7/8 ne faisant pas une telle recherche, P_j est dans ce cas laissé tel quel et indique une <u>limite inférieure</u> du premier bloc de taille .

Afin de limiter supérieurement cette recherche, ce qui est important en cas de recherche infructueuse, pour chaque taille de blocs un compteur du nombre de blocs libres est maintenu.

Ces informations associées, index et compteur, ne concernent que les tailles inférieures à 64 quanta. Les autres blocs libres, plus rares sont répertoriés individuellement (index et taille) dans un tableau spécial.

On aboutit donc à la structure de la figure 6 :



Représentation de l'espace sur disque
(TABLEDISQUE)

FIGUPE 6

b) Les tables résidentes en mémoire centrale :

Le mécanisme d'allocation, avant toute lecture de table sur le disque, recherche la portion de la partition offrant le plus de chances de réaliser l'allocation dans de bonnes conditions. A cet effet, un résumé des TABLEDISQUES réside en mémoire centrale, résumé que l'on appellera TABLEMEMOIRE.

Pour chaque TABLEDISQUE, la TABLEMEMOIRE correspondante en mémoire centrale contient :

. L'indication de la taille du quantum représenté par le bit.
. Le nombre de quanta disponibles sur la portion (espace libre).
. L'adresse disque du début de la portion et de la TABLEDISQUE.
. Une table de 64 bits indiquant s'il existe des zones libres de la taille n quanta (n = 1 à 63) ou de taille supérieure à 63 quanta (bit de rang 64). Ces tailles sont à multiplier par 1, 2 ou 4 suivant la valeur précisée au début de la table.

Les TABLEMEMOIRES permettent de choisir la portion de disque où sera faite l'allocation et de lire la table d'occupation correspondante sur disque TABLEDISQUE.

2) Tables du mécanisme particulier ou tables d'allocation rapide "QUICK-CELL"

Afin d'éviter des entrées-sorties trop nombreuses et une mise à jour trop fréquente des tables, il existe une table d'allocation en mémoire centrale utilisée pour les demandes d'un quantum : la "quick-cell".

Cette table répertorie les adresses des quanta qui sont préalloués sur la table d'occupation du disque TABLEDISQUE. Lors d'une demande d'un quantum, un des quanta de la table quick-cell est alloué au demandeur, sans qu'il soit nécessaire de lire la TABLEDISQUE et de la mettre à jour. Cette table joue le rôle d'un tampon et évite de nombreuses entrées-sorties sur disque.

L'annexe 1 décrit complètement cette table.

I-4 - LES MECANISMES D'ALLOCATION ET DE RESTITUTION

I-4-1- Méthodes d'allocation

Nous allons présenter 2 méthodes générales d'allocation (K2).

1) Méthode FIRST-FIT

On alloue le premier bloc libre dont la taille est supérieure ou égale à celle demandée. C'est évidemment la méthode employée lorsque les blocs allouables sont touts de même taille.

Exemple : - le système HASP, où la mémoire secondaire est allouée en cyclindres,
- le système CP, où la mémoire secondaire est découpée en pages.

L'avantage de la méthode FIRST FIT est sa rapidité. Son inconvénient majeur est la perte de place qui en résulte lorsque les blocs allouables ne sont pas tous de la même taille. Une amélioration de cette méthode, dans ce cas là, consite à tronquer le premier bloc trouvé, supérieur à la demande, à la taille de la demande. Nous l'appellerons méthode FIRST-FIT améliorée.

2) Méthode BEST-FIT

On alloue parmi tous les blocs libres celui dont la taille se rapproche le "mieux", par valeur supérieure, de la taille demandée. Il est nécessaire de parcourir toutes les zones susceptibles de convenir à la demande pour faire la meilleure allocation. La perte de place de la méthode FIRST-FIT est contre-balancée par la perte de temps dûe au choix entre différentes zones dans la méthode BEST-FIT.

Dans le cas de la méthode BEST-FIT, on peut aussi tronquer la zone libre à la taille demandée. Si l'on cherche à comparer cette méthode avec la méthode FIRST-FIT améliorée, le facteur "perte de place" ne joue plus : la méthode BEST-FIT, au bout d'un certain temps de fonctionnement (régime stationnaire), entraîne la prolifération de petits blocs, ce qui peut être un avantage si l'on veut disposer de petits blocs ou un désavantage si l'on recherche de grands blocs contigus.

Dans certains systèmes, lorsqu'il n'existe pas de bloc libre supérieur ou égal à la demande, l'allocation peut être fractionnée en plusieurs blocs. L'utilisateur peut préciser le fractionnement maximum autorisé.

Le système d'allocation peut laisser l'utilisateur choisir la méthode qui lui convient le mieux.

Exemple : - le système OS/MVT (I1).

Il est laissé une grande liberté à l'utilisateur quant à la méthode d'allocation. Suivant les options choisies, on peut en particulier :

- allouer tout l'espace de la zone contiguë la plus grande du disque, si cette zone est supérieure ou égale à la demande. Ceci est utilisé lorsqu'on demande de l'espace en ne possédant qu'une borne inférieure.

- allouer l'espace exact demandé à partir de la première zone supérieure ou égale à la demande (méthode FIRST-FIT améliorée).

- construire une liste des zones libres ; si en la construisant on trouve une zone égale à la demande ou supérieure, on alloue cette zone qui doit être la plus proche en taille de la demande (méthode BEST-FIT) et on détruit la liste.

Si toutes les zones disponibles sont inférieures à la demande, on alloue le montant exact demandé, parmi les plus grandes zones trouvées. On peut préciser qu'on limite à 5 le nombre des zones discontiguës.

Exemple : - le système SIRIS 7/8

Nous avons vu que l'utilisateur a le choix entre deux méthodes : Si la taille de la demande est un paramètre calculé, l'utilisateur appelle le mécanisme général d'allocation (que nous appellerons programme GET QUANTA) en précisant la partition sur laquelle seront alloués les quanta, le nombre de quanta demandés et le fractionnement désiré.

L'allocation se fait sur l'espace de la partition indiquée, par blocs contigus. On alloue par ordre de préférence (méthode BEST-FIT améliorée) :

- un bloc de taille égale à la demande,
- si ce n'est pas possible, un bloc de taille supérieure que l'on tronquera à la bonne taille,
- en dernier lieu, plusieurs blocs de taille inférieure à la demande, tout en s'efforçant de minimiser le nombre de ces blocs.

Si l'utilisateur désire uniquement un quantum, il fait appel au mécanisme particulier. Le quantum demandé est alloué immédiatement en prenant son adresse dans la quick-cell. Lorsque le stock de quanta de la table quick-cell est vide, le mécanisme particulier (que nous nommerons programme GET QUANTUM) appelle le mécanisme général, qui remplit le stock à nouveau. La gestion de la quick-cell est décrite dans le paragraphe suivant.

I-4-2- Les mécanismes de restitution

Afin d'allouer l'espace, il existe une liste ou un stock d'espace libre qui est accordé aux demandeurs.

Lorsqu'on libère l'espace ainsi alloué, on peut envisager de ne pas le rattacher à la liste d'espace libre. Un mécanisme particulier, le "garbage collector" récupèrera cet espace lorsque la liste d'espace libre sera vide. Il est nécessaire de mettre un indicateur, par exemple en tête de chaque bloc, indiquant s'il est libre et quelle est sa taille.

Lorsque le chaînage des blocs alloués est interne au système et n'est pas communiqué à l'utilisateur (par exemple en LISP), un mécanisme supplémentaire de compactage mémoire permet de n'avoir qu'un seul bloc libre et il est associé au "garbage collector". Ce mécanisme n'est utilisable que dans le cas où la mémoire est à accès rapide car l'algorithme est très cher en temps machine.

Suivant l'organisation de la liste d'espace libre, on doit :

- soit simplement rattacher l'espace libre à une extrémité de la liste,
- soit rechercher (si on possède une relation d'ordre entre les éléments de la liste) où est situé l'élément restitué. Cette recherche se simplifie si l'adresse du bloc fournit la position dans la liste (cas d'une allocation de blocs égaux avec table séquentielle).
- ressouder le bloc libéré aux blocs contigus libres, s'ils existent. Cette soudure peut être limitée comme dans le cas des systèmes par paires ("buddy system") ou par nombres de Fibonacci (cf. (K2)) au détriment d'une certaine place gâchée.

Expliquons, par exemple, comment fonctionne le système par paires :

L'espace libre sur disque est structuré en zones libres, dont la taille est toujours égale à une puissance de 2. Soit 2^M la dimension totale de l'espace disque.

On alloue des zones de tailles 2^0, 2^1, 2^2, ..., 2^M. Un tableau de taille M+1 permet d'indiquer s'il existe un bloc de taille 2^p, ($0 \leq p \leq M$) et si oui quelle est son adresse. Plusieurs blocs de même taille sont chaînés entre eux.

L'allocation se fait en recherchant un bloc de la taille demandée 2^p ; s'il n'en existe pas, on prend un bloc de taille 2^{p+m} libre, que l'on sépare en deux m fois pour obtenir un bloc de taille 2^p. La libération d'un bloc de taille 2^p est simple et tient à la particularité de la dichotomie : deux blocs de taille 2^2 obtenus à partir d'un bloc de taille 2^{2+1} sont dits jumeaux ("buddy"). Si on connaît l'adresse A_1 de l'un et sa taille T, on connaît l'adresse A_2 du jumeau ($A_2 = A_1+T$ ou A_1-T). L'adresse d'un bloc est toujours un multiple de sa taille.

Donc si le bloc jumeau A_2 est libre, il est oté de la liste de taille 2^p et on crée un bloc de taille 2^{p+1} en ressoudant les jumeaux Si le bloc jumeau n'est pas libre, le bloc A est rangé dans la liste de sa taille. Cette procédure est récursive et s'applique au nouveau bloc libre de taille 2^{p+1}, s'il est créé.

I-4-3- Méthodes utilisées dans SIRIS 7/8

L'utilisateur fait appel aux programmes RELEASE QUANTUM et RELEASE QUANTA pour libérer les quanta alloués respectivement par GET QUANTUM et GET QUANTA.

1) Le programme RELEASE QUANTUM

L'ordre de libération précise l'adresse secteur du quantum à libérer. Si le quantum appartient à la portion quick-cell, il est restitué à la table de la quick-cell, lorsqu'elle n'est pas saturée. Si la table est saturée ou si le quantum n'appartient pas à la portion de la quick-cell, le programme appelle le programme RELEASE QUANTA et le quantum est restitué au disque.

2) Le programme RELEASE QUANTA

Ce programme permet de libérer les zones de plusieurs demandes. L'ordre de libération précise le disque et les adresses secteurs de début et de fin de chaque zone. Les zones doivent être classées par ordre croissant de disque et d'adresse secteur. Le programme lit la TABLEDISQUE de la portion

contenant la zone à libérer et la met à jour, ainsi que la TABLEMEMOIRE correspondante; La zone libérée est ressoudée aux zones libres immédiatement contiguës pour créer une zone libre plus vaste, si cela est possible. Enfin, on réécrit la TABLEDISQUE modifiée sur le disque.

Les algorithmes de ces 4 programmes sont décrits dans l'annexe 2.

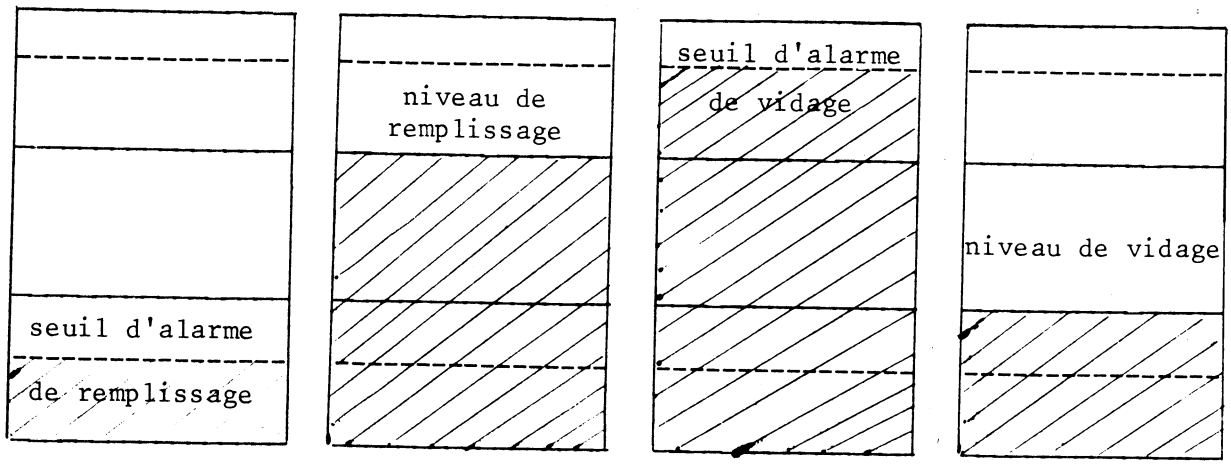
I-5 - GESTION DE LA QUICK-CELL

Définition :

La quick-cell se présente comme un réservoir de blocs d'un quantum qui alimente les demandes (GET QUANTUM) et reçoit les restitutions (RELEASE QUANTUM) ; lorsque la quick-cell devient vide (respectivement pleine), on doit faire un accès au disque pour la remplir (respectivement la vider) par l'intermédiaire du mécanisme général d'allocation.

Un raffinement de ce mécanisme a été introduit (voir figure 7).

On effectue des demandes anticipées de remplissage ou de vidage (appelées "alarmes") lorsque des seuils d'alarme sont atteints, avec l'espoir de faire ces opérations à l'occasion d'une autre allocation sur le même disque ; de telles opérations, qui économisent 2 accès au disque, seront appelées mouvements "marginaux", tandis que les autres, faites exclusivement pour la quick-cell, seront appelées mouvements "principaux". De plus, les remplissages et les vidages se font partiellement, jusqu'à des niveaux fixes. Ces niveaux doivent permettre d'éviter des vidages et remplissages intempestifs lorsqu'il y a alternance de demandes et restitutions et que la quick-cell est vide ou pleine.



Les tests sur la valeur des seuils se font dans les procédures GET QUANTA, pour le seuil bas SR, et RELEASE QUANTA, pour le seuil haut SV lors d'une demande sur la portion de la quick-cell.

Pour simplifer la gestion, le mécanisme de la quick-cell a été restreint à un seul disque, car les remplissages et vidages sont efficaces lorsqu'ils ne concernent qu'un disque. Néanmoins, il ne doit pas être difficile ni d'implanter plusieurs quick-cells (pour des tailles différentes) concernant un même disque, ni d'utiliser plusieurs quick-cells pour des disques différents.

CHAPITRE II

ANALYSE DES DONNEES

INTRODUCTION

Les mesures, dont nous n'allons donner ici que les principales caractéristiques, nous ont été fournies par le centre de calcul de la CII aux Clayes-sous-Bois. L'analyse complète en est faite dans la note interne CII (B3).

Ces mesures portent sur une période de fonctionnement de trois heures et demie d'un ensemble 10070 sous SIRIS 7, en traitement par lots. Elles décrivent les demandes et restitutions d'espace sur les disques "système" (un DIAD et un DIMAS), ainsi que l'état d'occupation des disques à intervalles fixes.

L'algorithme d'allocation, lors de la prise de mesures, correspondait à l'ancienne version et il avait en outre été modifié dans un but que nous ignorons : les quanta préalablement alloués à la quick-cell sont restitués au disque et non à la quick-cell. Cette modification ne change pas la demande, mais influe sur le résultat de l'allocation (état d'occupation du disque, remplissage de la quick-cell).

Nous avons relevé deux périodes durant lesquelles il y eut arrêt des mesures ; nous avons dû reconstituer les ordres d'allocation et de restitution manquants, pour fournir un ensemble cohérent de données au modèle. Sauf indication expresse, nous n'avons pas tenu compte de ces ordres dans nos statistiques.

Les points de mesure sont introduits dans le code des programmes d'allocation et de restitution, au moment de la génération du système. Les informations sont enregistrées sur bande magnétique, en binaire. Un programme écrit dans le langage d'assemblage du 10070 opère un pré-traitement sur ces données (traduction du binaire en EBCDIC, élimination des informations non utiles) et fournit des renseignements statistiques élémentaires.

II-1 - ASPECTS GENERAUX

L'utilité de ces mesures est double :

- injectées en entrée sur le modèle,
- l'analyse statistique de ces données permet d'examiner si l'algorithme d'allocation mis en oeuvre est adapté à ce type de demandes.

L'examen des données ne nous a pas permis de dégager des lois statistiques concernant la demande d'espace disque, écartant ainsi la méthode analytique pour résoudre les problèmes posés (C2).

Le tableau ci-dessous donne le relevé des mesures prises :

DISQUE CONCERNE	DIAD	DIMAS	TOUS
Type de demandes	nombre	nombre	nombre
GET QUANTUM	0	1686	1686
RELEASE QUANTUM	0	1686	1686
GET QUANTA	92	527	619
RELEASE QUANTA	14	376	390
			4381

Sur les 619 zones allouées grâce à la procédure GET QUANTA, 596 ont été rendues : la procédure RELEASE QUANTA n'a cependant été appelée que 390 fois, car elle permet de restituer plusieurs zones à la fois (gain d'E/S à l'aide d'un seul ordre).

$$\frac{\text{Nombre de GET QUANTUM}}{\text{Nombre de (GET QUANTUM + GET QUANTA)}} = 72\ \%$$

On remarquera que 72 % des allocations font appel à la procédure GET QUANTUM, ce qui justifie l'importance accordée à l'étude de cette procédure (utilisation de la quick-cell).

D'autre part, l'observation du taux d'occupation des disques nous indique que ces disques sont trop loin du seuil de saturation pour que l'émiettement de l'espace libre en petites zones discontiguës puisse être significatif ; nous définirons le taux moyen d'occupation (respectivement l'émiettement moyen)

d'un disque par l'intégrale dans le temps du taux d'occupation (respectivement de l'émiettement) à l'instant t, divisé par l'intervalle de temps référencé, où :

- taux d'occupation à l'instant t = $\frac{\text{place occupée sur le disque au temps t}}{\text{place totale sur le disque}}$

- l'émiettement à l'instant t = nombre de zones libres discontiguës à l'instant t.

Ainsi défini, le taux moyen d'occupation du disque DIAD est de 50 % et celui du disque DIMAS varie entre 3 et 8 % ; l'émiettement moyen est de 8,4, avec un émiettement instantané maximum de 23. Une étude par taille des zones libres indique une prédominance des tailles faibles, en effet 50 % des zones libres sont inférieures à 5 quanta.

II-2 - ORDRES GET ET RELEASE QUANTA

Les ordres GET QUANTA représentent 28 % des demandes d'allocation.

a) Répartition par taille

Le tableau ci-dessous nous donne la répartition par taille des ordres GET QUANTA.

TAILLE EN QUANTA	1	2	3	4	5	6	7	10	15	20	28	39	40
NOMBRE D'ORDRES GET	184	153	13	84	71	77	1	3	2	27	1	1	1
% par TAILLE	30	25	2	14	11	12	-	0,5	-	-	4	-	-

On s'aperçoit que 55 % des demandes sont de 1 et 2 quanta et 94 % des demandes sont inférieures à 7 quanta.

La répartition par taille des ordres RELEASE QUANTA reflète celle des ordres GET QUANTA.

b) Cadences d'arrivée

Les ordres GET QUANTA arrivent groupés par "paquets" dans le temps, comme le montre la figure 8. A l'intérieur de ces paquets, les ordres se suivent d'assez près : l'écart de temps entre 2 ordres GET QUANTA est inférieur à 10 secondes pour 86 % des ordres.

Il en est de même pour les ordres RELEASE QUANTA qui apparaissent encore plus groupés que les ordres GET QUANTA : l'écart entre 2 ordres RELEASE QUANTA est inférieur à 5 secondes pour 77 % des ordres (voir figure 9).

c) Temps de rétention

En moyenne une zone allouée est conservée 210 secondes (écart type 690 sec.). Les ordres étant groupés par tranches de 100 sec., la superposition des figures 8 (ordres GET QUANTA) et 9 (ordres RELEASE QUANTA) montre que les zones sont rendues dans la tranche de temps où elles ont été allouées.

Près de 40 % des quanta alloués par GET QUANTA sont conservés moins de 10 secondes.

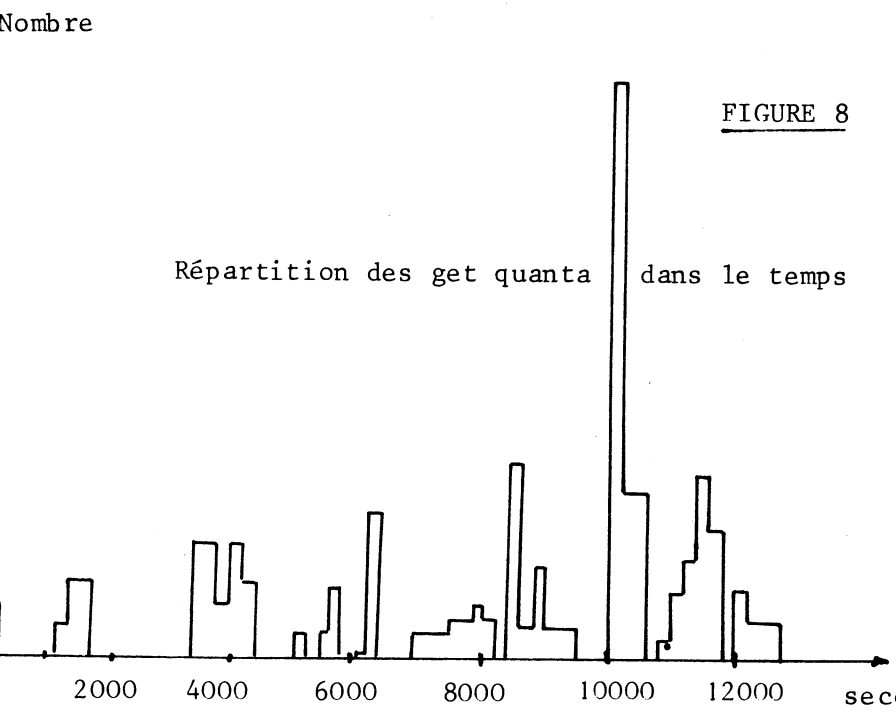


FIGURE 8

Répartition des get quanta dans le temps

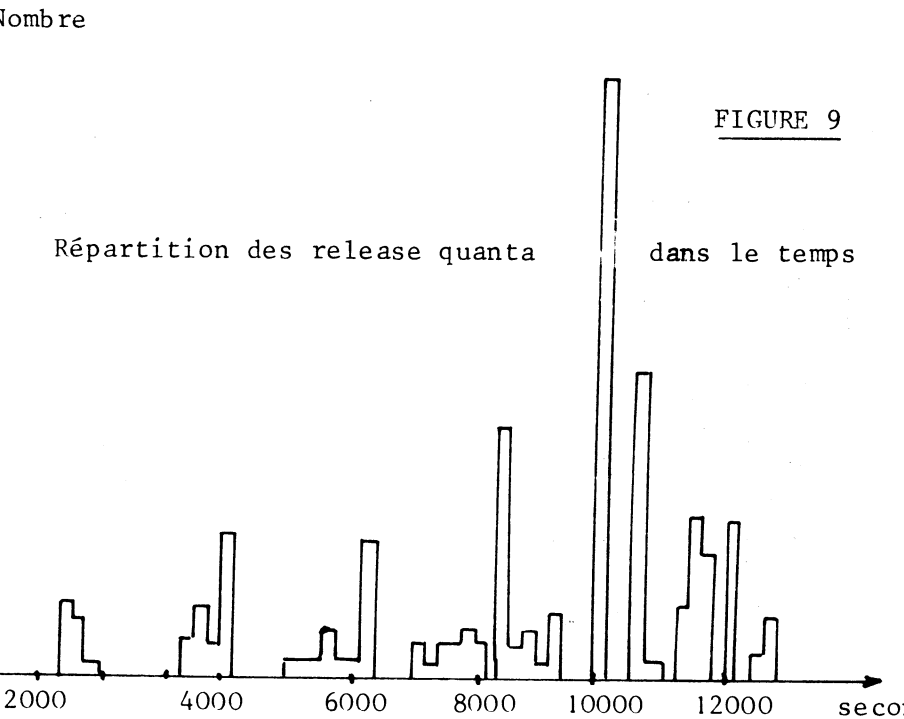


FIGURE 9

Répartition des release quanta dans le temps

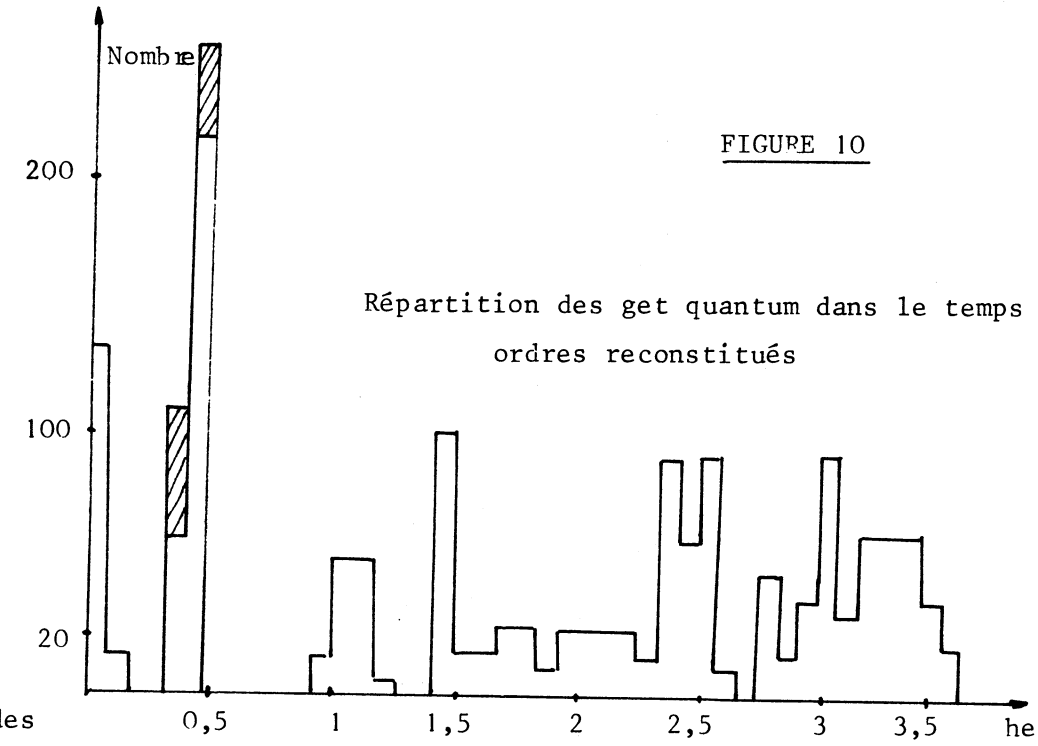


FIGURE 10

Répartition des get quantum dans le temps
ordres reconstitués

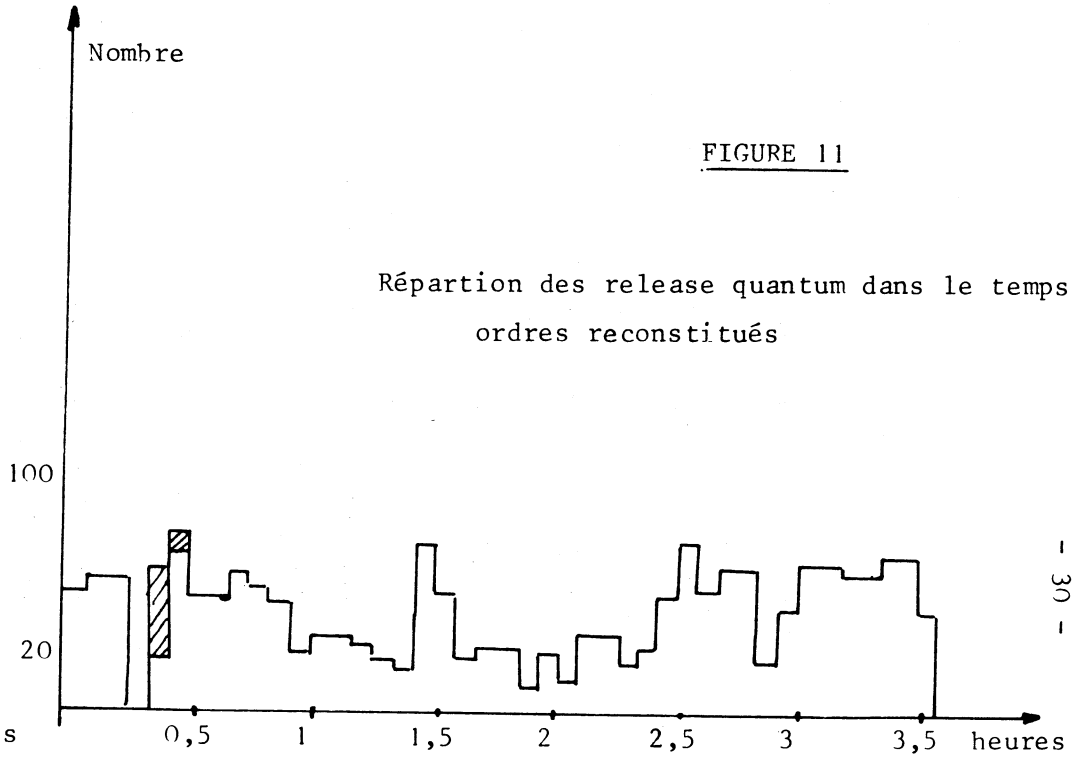


FIGURE 11

Répartion des release quantum dans le temps
ordres reconstitués

II-3 - ORDRES GET ET RELEASE QUANTUM

Nous rappelons que les ordres GET QUANTUM allouent un quantum appartenant à la quick-cell, mais que pour cette mesure les ordres RELEASE QUANTUM rendent directement le quantum au disque.

a) Cadences d'apparition

Les ordres GET QUANTUM (figure 10) apparaissent de façon très discontinue : ils ont tendance à arriver par paquets ; l'écart de temps entre deux GET QUANTUM est égal à 8 secondes en moyenne, mais plus de 60 % des écarts sont inférieurs à 2 secondes ; l'écart maximum est de 1594 secondes.

Les ordres RELEASE QUANTUM (figure 11) apparaissent de façon plus régulière : l'écart moyen est de 7, 8 secondes, avec un écart maximum de 312 secondes.

b) Temps de rétention

La figure 13 donne l'histogramme, en pourcentage, des temps de rétention d'un quantum par tranche de 100 secondes. Plus de 50 % des quanta sont libérés dans un laps de temps de 5 minutes, le temps de rétention moyen étant de 8 minutes 40 et le maximum de 2 heures.

Nous avons tenté de trouver une corrélation entre la date d'arrivée des ordres GET QUANTUM et le temps de rétention du quantum correspondant. Bien que sur l'ensemble de la période le coefficient de corrélation soit proche de zéro, il n'en est pas moins vrai qu'une relation linéaire existe entre ces deux variables, mais pour des groupes d'ordre GET QUANTUM limités dans le temps, comme le montre la figure 12.

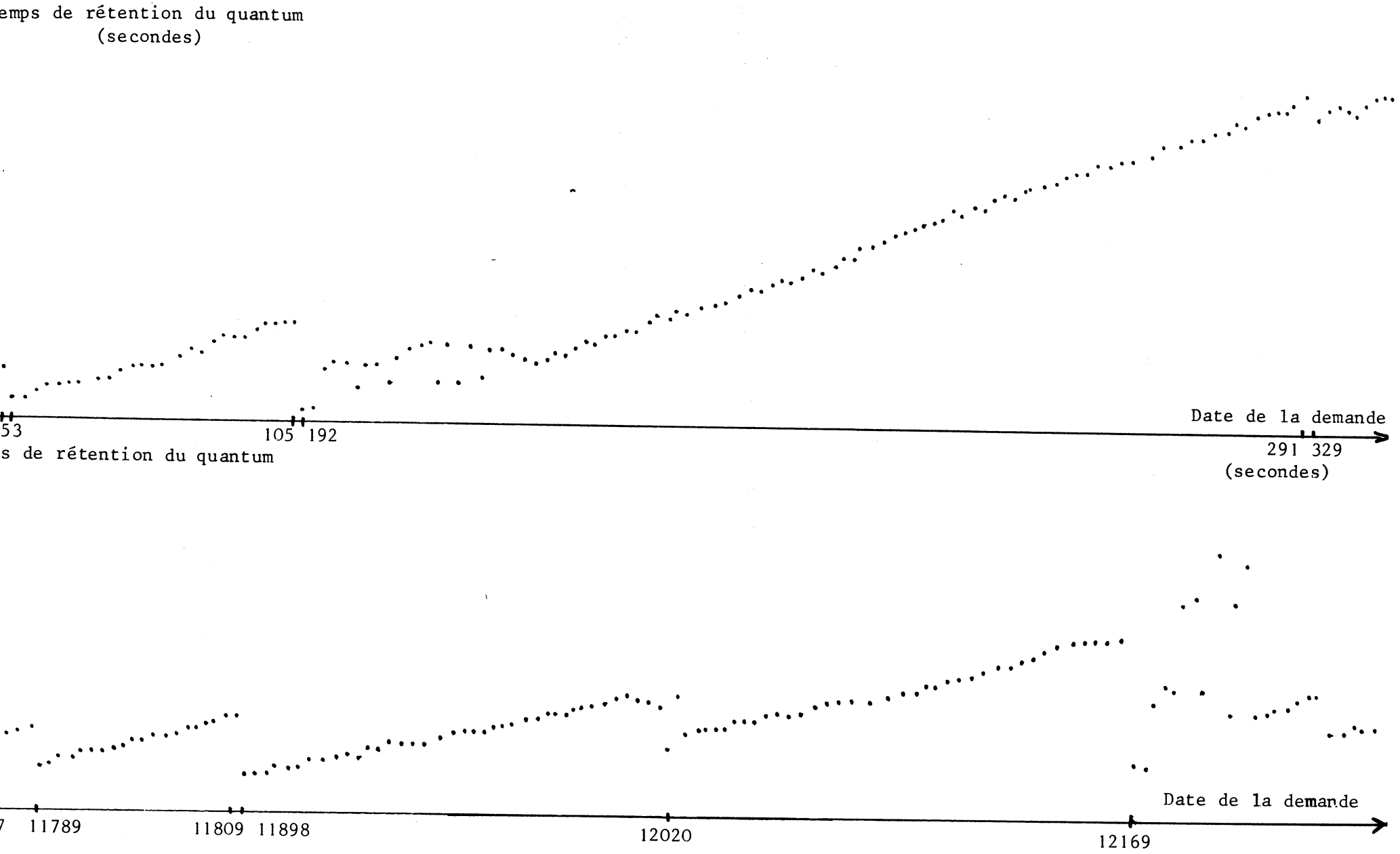


FIGURE 12 : Date d'apparition d'un ordre get quantum et temps de rétention correspondant (exemples pris au début et à la fin de la période de mesures)

Figure 13

Histogramme du temps de rétention d'un get quantum

Temps de rétention minimum : 3 secondes
" " " maximum : 7996 secondes
" " " moyen : 524 secondes

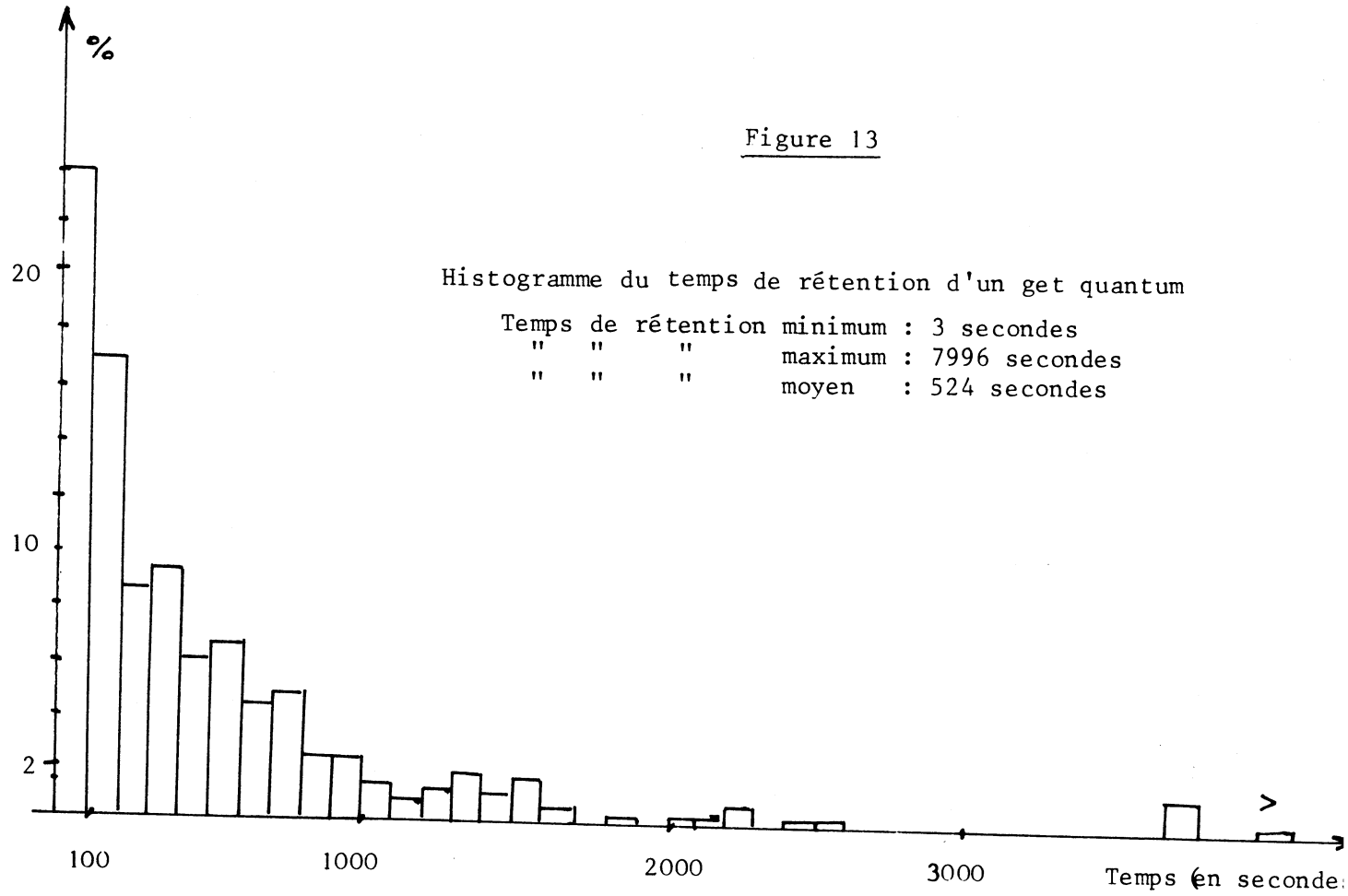
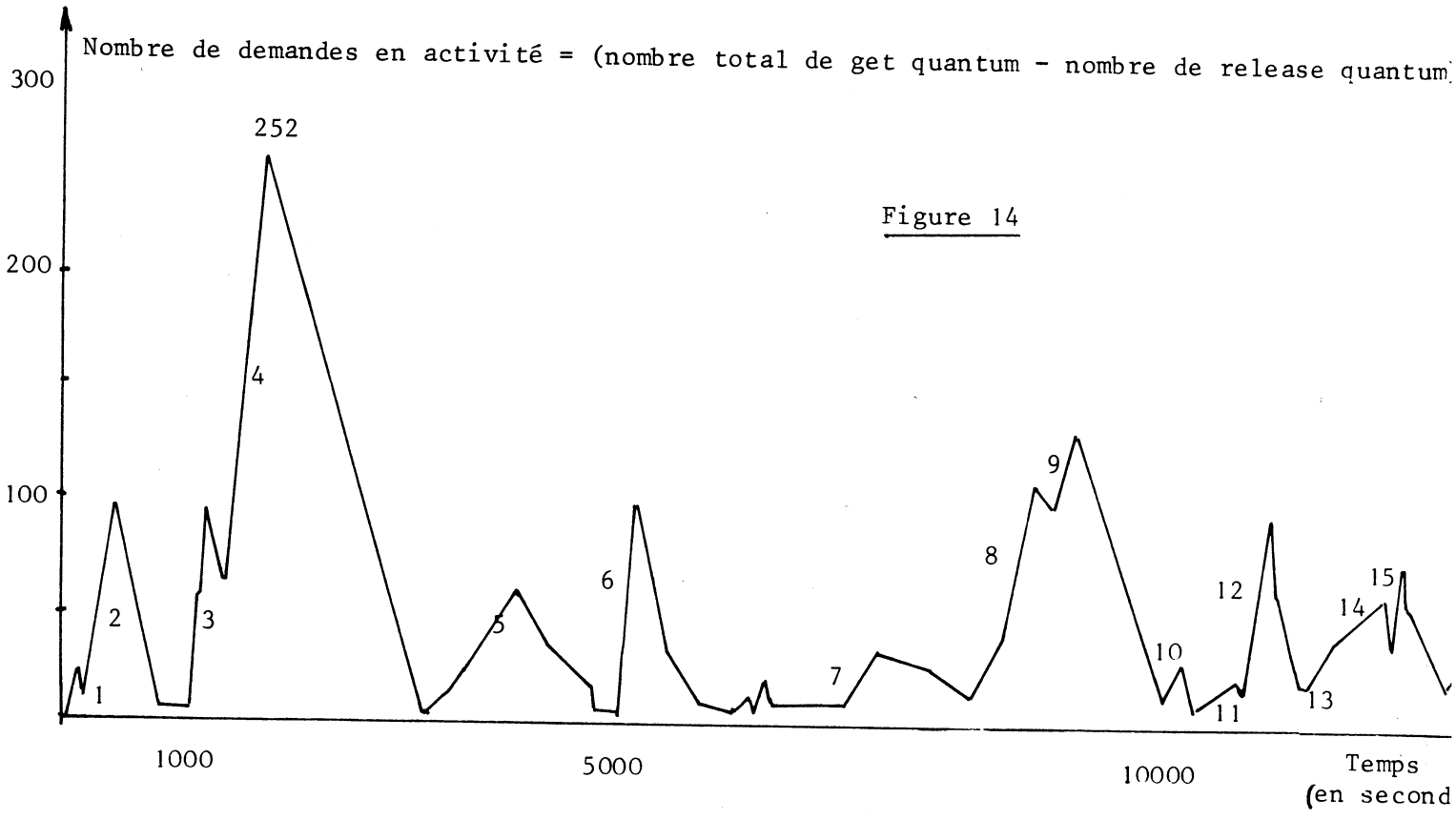


Figure 14



Une analyse statistique plus fine des ordres GET QUANTUM est donc nécessaire. Comme on l'a vu, ceux-ci arrivent par paquets assez denses, et à l'intérieur d'un paquet, les ordres sont à peu près uniformément répartis. La figure 12 représente le nombre de demandes en activité à l'instant t, sur l'ensemble de la période, c'est à dire le nombre total d'ordres GET QUANTUM, auquel on soustrait le nombre total d'ordres RELEASE QUANTUM, apparus jusqu'à l'instant t.

Aux paquets d'ordres GET QUANTUM correspondent les montées des "pics" de la figure 14, tandis que la restituion des quanta alloués au cours de la montée correspond à la descente des mêmes pics.

Les temps de rétention, pour un paquet (un pic sur la figure 12) des quanta alloués suivent une loi linéaire en général croissante*. (Voir figure 12)

Temps de rétention = A × date d'arrivée par rapport au début du paquet + m
(d'un quantum) (de l'ordre allouant ce quantum)

Ce qui signifie que les quantum sont utilisés dans l'ordre où ils sont demandés. Cette approximation est valable, avec un coefficient de corrélation égal à 0,83 en moyenne (moyenne non pondérée).

La pente A dépend du paquet envisagé et elle est liée à la densité D des ordres dans le paquet par la relation :

$A = kD + 1$ (coefficient de corrélation = 0, 68)

Le tableau de la figure 15 montre les intervalles numérotés de 1 à 15 où les ordres GET arrivent en paquets. Certains de ces paquets ont été décomposés pour éliminer une ou deux mesures qui perturbent les moyennes (c'est le cas, par exemple, de quanta non rendus). Les numéros 1 à 15 correspondent aux montées numérotées de la figure 14.

* Pour 6 blocs la pente A est négative, mais il s'agit de périodes d'activité presque nulles (peu d'ordres, faible densité).

Si nous avons trouvé une loi pour les temps de rétention à l'intérieur d'un paquet, nous n'en avons pas trouvé, par contre, pour les paquets permettant de calculer le nombre d'ordres dans un paquet ou le temps de rétention moyen.

II-4 - ETUDE DE LA QUICK-CELL

Avec les modifications apportées par la prise de mesures, on ne peut rien dire de significatif sur le remplissage de la quick-cell. Signalons, cependant, que la quick-cell a été remplie 34 fois avec cet algorithme.

Aussi, avons-nous dû créer une deuxième bande de mesures, où les ordres de restitution sont éliminés, mais où le temps de rétention est associé à l'ordre d'allocation correspondant. Cela nous a ainsi permis de changer l'algorithme d'allocation. La caractéristique invariante d'une demande est le temps de rétention et non l'adresse de la zone allouée qui dépend de l'algorithme et qui nous servait à établir la correspondance entre un ordre GET et l'ordre RELEASE correspondant.

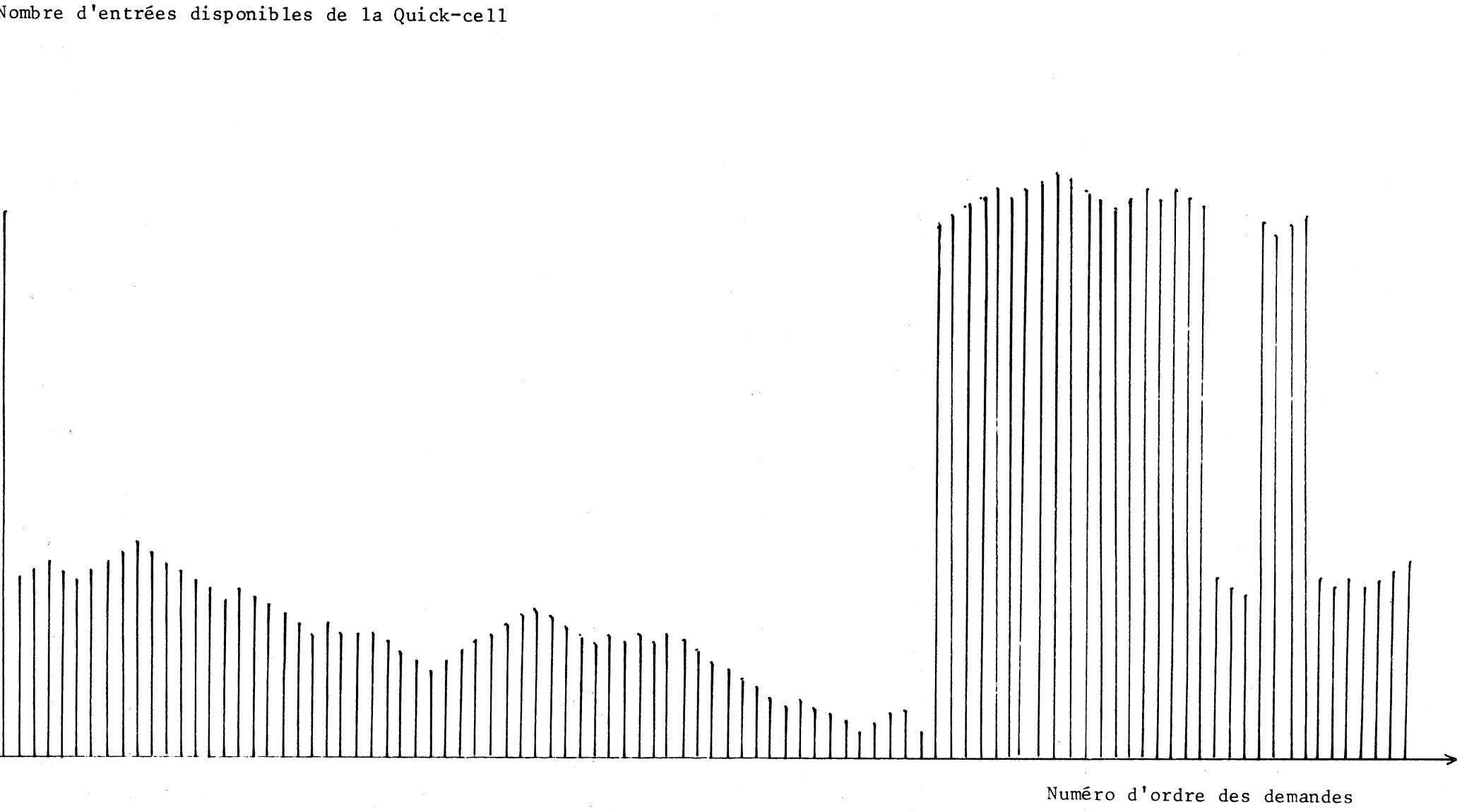
Avec l'algorithme réel, la quick-cell est remplie 18 fois et vidée 15 fois dont 2 fois par anticipation grâce aux seuils.

La figure 16 donne une partie caractéristique de la courbe représentant le nombre d'entrées disponibles (les quanta) dans la quick-cell, en fonction du temps. Nous avons uniquement représenté les instants où la quick-cell est modifiée. En 9 minutes, la quick-cell a été remplie, puis aussitôt vidée, 3 fois. On peut penser qu'une simple modification de la valeur des seuils peut supprimer ces remplissages et vidages intempestifs : le seuil de vidage SV est en effet égal au niveau de remplissage NR, de même que le niveau de vidage NV est égal au seuil de remplissage SR.

Le fait de donner des valeurs différentes à SV et NR d'une part, NV et SR d'autre part, ne peut empêcher ces opérations inutiles, mais ne peut que les décaler dans le temps. Ceci est dû à la très grande irrégularité de la distribution, comme on le verra au cours des expériences de simulation.

Intervalles		Durée (secondes)	Nb. de get	Densité	Temps moyen de rétention	Ecart-type	Pente A	Corrélations	Nb. de relea
0	105	105	34	0.32	83	40	0.7	0.53	10
192	291	99	101	1.02	267	151	5.0	0.90	22
330	336	6	6	1.00	527	9	3.1	0.90	3
1202	1314	112	112	1.00	307	193	5.4	0.87	26
1520	1673	153	147	0.96	616	299	3.3	0.51	35
1677	1716	39	28	0.72	1751	40	2.6	0.95	10
1719	1813	94	63	0.67	1425	103	3.8	0.90	28
3407	3458	51	9	0.18	198	59	2.8	0.92	0
3531	3659	128	21	0.18	67	43	1.1	0.89	14
3662	3764	102	18	0.18	204	56	1.2	0.61	11
3768	3819	51	16	0.31	307	84	2.3	0.59	5
3822	4073	251	46	0.18	544	187	1.6	0.52	22
5124	5226	102	98	0.96	207	110	5.3	0.73	24
5643	5764	121	12	0.10	76	24	-0.5	-0.35	9
5767	6282	515	33	0.06	60	30	-0.2	-0.81	35
6322	6564	242	19	0.08	75	64	0.8	0.93	19
6734	7003	269	15	0.06	50	25	-0.2	-0.87	14
7092	7393	301	29	0.10	129	78	0.6	0.68	19
7458	7878	420	41	0.10	126	37	-0.3	-0.95	43
7899	7997	98	10	0.26	155	42	2.7	0.73	4
7998	8141	143	11	0.07	132	21	-0.5	-0.99	8
8422	8437	15	19	1.27	242	109	27.5	0.95	3
8573	8741	168	78	0.46	375	105	2.8	0.67	15
8790	8803	13	8	0.61	500	6	1.2	0.38	0
8950	8955	5	8	1.60	517	10	6.7	0.98	1
8969	9032	63	8	0.13	210	49	1.9	0.91	11
9036	9156	120	50	0.42	695	40	0.9	0.92	29
9164	9205	41	17	0.42	372	15	1.0	0.99	8
9226	9237	11	7	0.64	399	5	1.4	0.90	1
9860	9918	58	12	0.21	698	8	-0.2	-0.59	17
9973	10051	78	17	0.22	753	44	1.3	0.63	19
10067	10197	130	31	0.24	62	22	0.3	0.61	27
10548	10723	175	37	0.21	47	23	0.3	0.65	29
10860	10889	29	80	2.76	170	1	13.7	0.98	8
11179	11230	51	13	0.25	51	11	0.5	0.62	12
11294	11307	13	13	1.00	42	23	4.3	0.88	3
11373	11503	130	34	0.26	68	26	0.6	0.79	19
11564	11582	18	14	0.78	63	16	2.5	0.97	4
11651	11728	77	30	0.39	83	31	1.4	0.96	14
11792	11809	17	20	1.18	108	35	4.1	0.97	3
11894	12020	126	41	0.33	124	39	1.2	0.98	25
12083	12169	86	46	0.53	203	47	1.4	0.87	17
12384	12423	39	15	0.38	152	12	3.7	0.96	7
12575	12615	40	17	0.43	81	15	1.2	0.92	8

FIGURE 16


Nombre d'entrées disponibles de la Quick-cell
Numéro d'ordre des demandes

CHAPITRE 3

LE MODELE DE L'ALLOCATION

INTRODUCTION

Pour résoudre les problèmes posés par l'allocation de l'espace sur les disques "système", nous avons bâti un modèle, écrit en langage SIMULA, de cette allocation.

a) - Le modèle possède en entrées des demandes et des restitutions d'espace sur le disque. Ce sont les mesures réelles prises sur le système SIRIS 7.

b) - Le modèle reproduit fidèlement les algorithmes d'allocation et de restitution. Pour préciser cette fidélité, disons que lors d'une demande de quanta, le modèle alloue ce que le système aurait réellement alloué.

c) - En sortie nous avons la trace de la structure du disque système et diverses statistiques permettant d'étudier la réponse du système aux variations des paramètres de l'allocation, ainsi que la réponse à différentes sortes de demandes.

Le premier problème posé pour bâtir le modèle est de savoir quel serait le langage que nous allions utiliser pour effectuer notre simulation.

III-1 - CHOIX D'UN LANGAGE DE SIMULATION

Le choix d'un langage de simulation repose sur trois considérations :

1) La difficulté majeure est de savoir comment représenter l'espace sur disque. Si on veut reprendre intégralement la représentation réelle, cela nécessite la reproduction des tables sur disque, soit une table de 3000 bits et trois tableaux permettant de repérer les zones libres.

D'une part, pour des raisons de rapidité d'exécution de la simulation, on est obligé de transférer ces tables en mémoire centrale ; d'autre part, on doit reprendre tout le code des mécanismes d'allocation, donc écrire le modèle

en langage d'assemblage, pour permettre de travailler au niveau du bit. L'encombrement de la mémoire n'est pas négligeable et la programmation n'est pas aisée.

Nous ne pouvons donc reprendre la même représentation de l'espace disque.

Quelle est la représentation choisie ?

Un premier examen des données (les mesures prises sur le système) nous a montré que l'espace libre du disque était très peu morcelé ; aussi plutôt qu'une structure statique de table, une structure dynamique de listes a été retenue : chaque élément de liste représente une zone libre sur une portion et chaque liste une portion de disque.

2) Pour faciliter l'écriture des programmes, leur compréhension, la facilité des corrections, il est nécessaire de prendre un langage de programmation évolué.

Le langage APL d'Iverson (14), un moment retenu, a été écarté à cause de l'impossibilité de lire ou d'écrire sur des fichiers : il n'existe pas encore de gestion de fichiers en APL, ceux-ci doivent résider en mémoire centrale.

Les langages FORTRAN ou PL1 ou ALGOL convenaient assez bien, le langage ALGOL par sa syntaxe claire et précise paraissait préférable.

3) Le modèle étant bâti pour simuler un mécanisme, un langage de simulation, permettant de manipuler aisément la notion du "temps", était nécessaire.

En résumé, nous avons besoin d'un langage de haut niveau, permettant de manier aisément une structure de listes et d'activer des processus suivant une chronologie déterminée. Le langage SIMULA, disponible sur le 10070 à l'Université de GRENOBLE, qui combine ces possibilités a été retenu (S3, S4).

III-2 - LE LANGAGE SIMULA

Notre propos n'est pas de décrire le langage SIMULA, mais d'indiquer quelles sont les caractéristiques de ce langage utiles à la construction de notre modèle.

Si l'écriture du modèle a été aisée, par contre, sa mise au point a été très délicate. En effet, la version, que nous avons du compilateur, est une version expérimentale, non encore "commercialisée" ; d'autre part, le compilateur était prévu pour une mémoire plus vaste, aussi a-t-on dû introduire des modifications pour pouvoir l'utiliser.

Un premier modèle n'a pas réussi à s'exécuter correctement et, au bout d'un mois, nous avons décidé de le remanier :

Nous avons éliminé certains cas très rares de l'algorithme ; il a enfin pu s'exécuter. Il s'agissait sans doute du mécanisme de "garbage collection", la récupération de la mémoire inutilisée, qui fonctionnait mal. Nous étions toujours, en effet, à la limite de l'espace mémoire disponible. Ces difficultés de mise au point mises à part, le langage SIMULA s'est révélé être un outil élégant pour l'écriture de modèles de simulation. Deux caractéristiques se détachent en SIMULA : le concept de classe, la notion de temps.

1) Le concept de classe

SIMULA, comme tout langage évolué, permet de manipuler des booléens, des entiers, des réels, des caractères et des tableaux. On peut également traiter des "objets" plus complexes, possédant une structure. Tout objet appartient à une classe d'objets.

Une déclaration de classe d'objets a la forme simplifiée suivante :

```
class  A  (P)  ;  S  ;
begin      D  ;
       I ;
end ;
```

A est le nom de la classe, P est une liste de paramètres formels, S la liste de leurs spécifications, D est une liste de déclarations et I représente la liste des instructions à exécuter.

Un objet est un exemplaire dynamique de classe ; il est généré par un générateur d'objets.

L'instruction "new A (paramètres)" engendre un exemplaire de la classe A et exécute les instructions de la classe.

Il faut remarquer que la liste des déclarations D peut contenir des déclarations de procédure.

La structure de l'objet, c'est à dire les paramètres P et les déclarations D, reste en mémoire centrale aussi longtemps qu'il est possible d'y accéder. C'est ainsi que les procédures de l'objet généré sont exécutables.

La concaténation de classes est un élément intéressant du langage. Une structure de données peut se déduire d'une autre structure de données par adjonctions d'un ou plusieurs attributs. On parle alors de sous-classe d'une classe donnée. Les classes et sous-classes s'organisent en arborescences. Soient les déclarations suivantes de la classe C1 et de la classe C2 :

```
class  C1  (P1)  ;  S1
begin  D1  ;
       I1  ;
inner  ;
F1
end  ;
```

```
C1  class  C2  (P2)  ;  S2  ;
    begin  D2  ;
           I2  ;
    end  ;
```

La génération d'un objet de la classe C2 entraîne la création d'une classe fictive qui est la concaténation des deux classes C1 et C2 d'après le schéma suivant :

```
class  C1  C2          (P1, P2)  ;  S1  ;  S2  ;
begin  D1  ;  D2  ;
       I1  ;
       I2  ;
       F1  ;
end  ;
```

Le symbole inner est un séparateur qui permet l'inclusion des instructions de la classe concaténée. Les instructions sont exécutées dans l'ordre I1, I2 puis F1. Cette concaténation se généralise aisément pour plusieurs classes. Elle permet de créer d'une manière modulaire des structures complexes, aussi bien de données que de programmes : c'est un processus de macro-définition.

La structure de listes est particulièrement représentative de ces possibilités ; il existe en SIMULA une classe système nommée SIMSET qui contient les classes HEAD et LINK, permettant de manipuler des listes circulaires à double chaînage, successeur-prédécesseur.

Une liste se compose d'une tête de liste, la classe HEAD (c'est le nom de la liste) et d'un ensemble de liens, la classe LINK qui constituent les éléments de la liste.

Des procédures de manipulation de liste, prédéfinies dans la classe SIMSET, permettent, entre autres, d'insérer ou d'ôter des éléments dans une liste ; soit, par exemple, la liste des abonnés au téléphone réf. (HEAD) ABONNES.

Une personne abonnée est définie par :

```
LINK class PERSONNE (NOM, NUMERO) ; text NOM ; integer NUMERO ; begin text
PROFESSION, ADRESSE ;
    null  $  il n'y a pas d'instruction à exécuter  $ ; end ;
```

Monsieur ZZZ désire s'abonner, on lui donne un numéro :

```
    réf. (PERSONNE) ZZZ ; ZZZ : - new PERSONNE ('ZZZ', 874561) ;
```

(: - est le symbole : = s'appliquant aux pointeurs ou références)

et on l'inscrit sur la liste des abonnés, à la fin de la liste :

```
    ZZZ . INTO (ABONNES) ;
```

la procèdure INTO attachée à la classe à laquelle ZZZ réfère (c'est à dire PERSONNE sous-classe de LINK) est appelée avec le paramètre ABONNES.

De même, Monsieur YYY est rangé devant Monsieur ZZZ :

```
    YYY . PRECEDE (ZZZ) ;
```

Le prédécesseur de YYY résilie son abonnement et quitte la liste : YYY. PRED.OUT;

Par suite d'une erreur de l'ordinateur, tous les abonnements sont résiliés :

ABONNES . CLEAR ;

On peut connaître le premier abonné de la liste, ABONNES . FIRST, le dernier, ABONNES . LAST, le nombre d'abonnés ABONNES . CARDINAL.

2) La représentation du flux du temps

Il existe deux méthodes générales pour représenter le temps dans un modèle de simulation (N2).

- La méthode à incréments de temps fixe : l'horloge de temps réel est mise à l'heure à intervalles réguliers et il faut rechercher les évènements qui doivent se produire à cet instant précis dans le modèle de simulation. Cet incrément doit être la quantité minimale séparant deux évènements.

- La méthode à incréments de temps variable : l'horloge de temps réel est mise à l'heure de l'évènement dont la réalisation est imminente.

L'horloge progresse par sauts d'ampleur variable pour ne s'arrêter qu'aux instants "où il se passe quelque chose" (S4). En SIMULA, un processus, (PROCESS) à activer à un instant donné (EVTIME), est rangé dans une file d'attente des évènements, appelée SQS (sequencing-set), ordonnée par EVTIME croissants. Chaque évènement est associé à un processus et un seul.

Si un processus est présent dans la SQS, on dit qu'il est actif, c'est à dire qu'il doit être activé à l'époque associée EVTIME, supérieure ou égale à la valeur courante de l'horloge (TIME). Lorsque le processus en cours de traitement (CURRENT) s'interrompt, il peut rester actif (REACTIVATE) ou devenir passif (PASSIVATE). C'est le processus qui le suit dans la SQS qui reprend son traitement au point où il était suspendu (l'horloge fait un saut jusqu'au nouveau EVTIME associé).

Les opérations relatives à la vie autonome d'un processus sont gérées par les instructions :

- HOLD (T) : le processus courant se suspend pour une durée T ; si aucun évènement n'est prévu entre temps, l'horloge fait un saut d'amplitude T sinon un autre processus prend le contrôle.
- PASSIVATE : le processus courant disparaît de la SQS et passe le contrôle au processus suivant dans la SQS - l'horloge avance.

Les interactions entre processus se font par :

- ACTIVATE X : le processus X est mis en activité à la tête de la SQS.
- REACTIVATE X DELAY T: le processus X est déjà actif, mais on déplace son temps de réalisation de T : EVTIME = EVTIME + T.

Un exemple d'utilisation de ces possibilités est donné dans le paragraphe suivant.

III-3 - LE MODELE DE L'ALLOCATION

Nous allons séparer en deux la description du modèle en présentant d'abord la représentation de l'espace sur disque, puis le mécanisme d'allocation d'espace.

III-3-1- La représentation de l'espace sur disque (voir figure 17)

Pour chaque portion d'un disque, il existe une liste d'éléments décrivant l'espace libre de la portion.

Chaque élément représente une zone libre contiguë de la portion. Cette liste appelée DISQUE est rangée par ordre croissant d'adresses sur le disque. Dans le cas des mesures fournies, il n'y a qu'une seule portion par disque, d'où le nom DISQUE de la liste des zones de la portion.

De plus, il existe un tableau L de 64 listes permettant de classer ces zones libres par taille croissante. La liste de rang i, L(i), contient des zones de taille i. Une liste L(i) est rangée par ordre d'adresse croissante. La liste L(64) contient des zones de taille supérieure ou égale à 64 quanta.

On peut ainsi représenter l'espace de la portion par une suite de listes L(i), i = 1 à 64, orthogonales à DISQUE.

Les éléments de la liste DISQUE sont appelés "trou primaire" TROUP :

```
LINK class TROUP

    begin integer N, AD : ref (TROUS) X ;
    X :- new TROUS (this TROUP) ;
    end ;
```

Les éléments de la liste L(i) sont appelés "trou secondaire" TROUS ils servent de référence aux trous primaires.

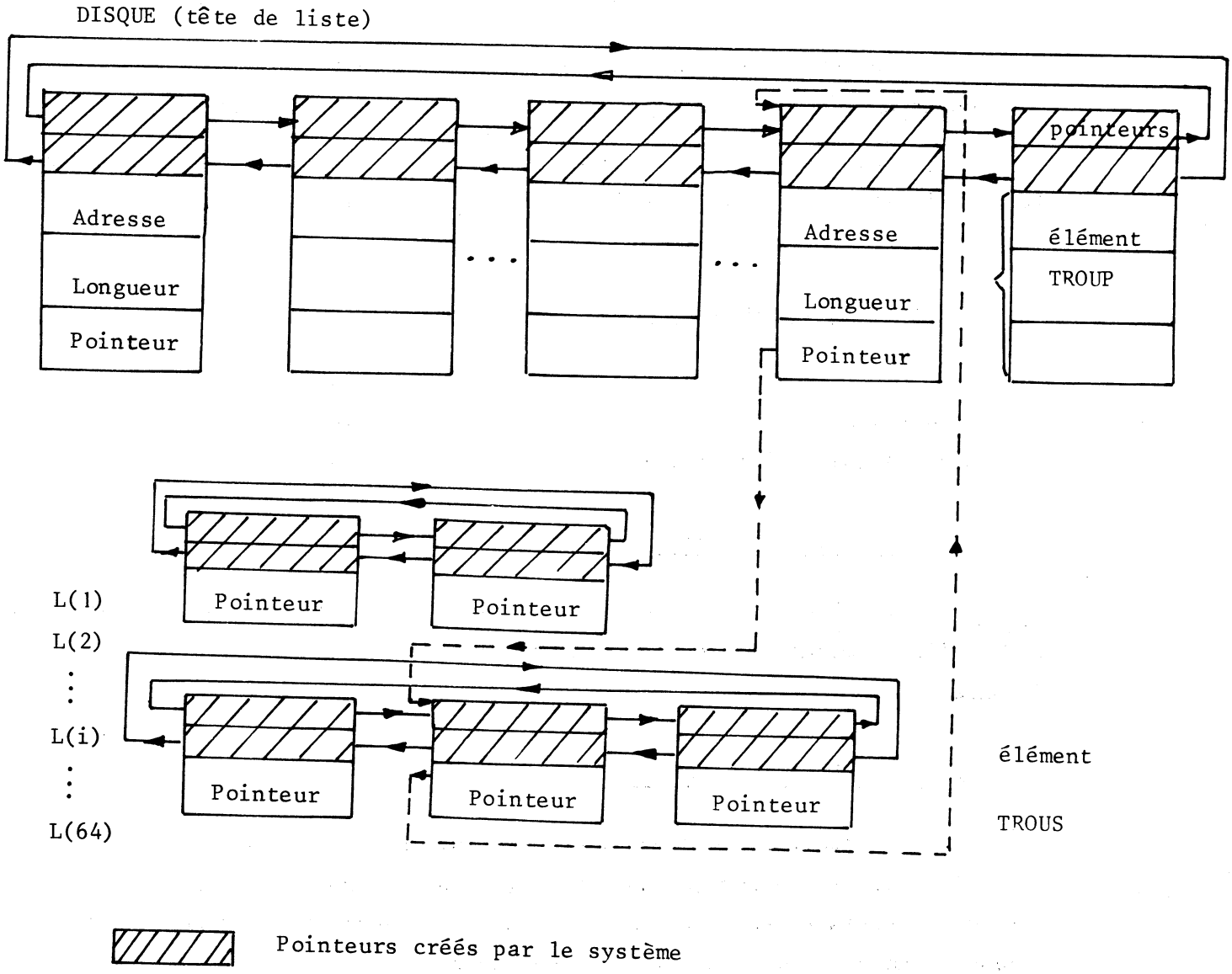
```
LINK class TROUS (Y) ; ref (TROUP) Y ; null.
```

Pour une zone d'espace sur le disque, un "trou", nous sommes obligé de définir deux liens ("trou primaire" et "trou secondaire") car un lien ("un trou", ici) ne peut appartenir à deux listes à la fois. C'est une contrainte du langage SIMULA.

Ces éléments sont préfixés par la classe LINK, ils en possèdent do les attributs, c'est à dire qu'on pourra facilement les insérer ou les ôter d'une liste.

Nous définirons donc une portion comme une classe, possédant un certain nombre d'attributs : une tête de liste DISQUE, 64 têtes de liste L(i) la place totale libre sur le disque PLACE, la taille de la portion NQMAX, ainsi que des procédures permettant de gérer l'espace libre de cette portion :

```
class PORTION
begin ref (head) DISQUE ;
      ref (head) array L (1:64) ;
      integer NBQBIT ; $ valeur du quantum sur cette portion
      integer PLACE ; $ place libre sur le disque $
      procedure OCCUPATION ; ...
      procedure EDIT ; ..., $ elle sert à imprimer l'état d'occupation
                              du disque
      procedure RESTITUTION ; ... ;
DISQUE :- new head ; $ création des têtes de listes, vides $
for I := 1 step 1 until 64 do L(I) :- new head ;
end ; $ fin de la classe portion $
```



Représentation de l'espace sur disque

FIGURE 17

La portion est créée vide ; l'état de la portion au départ de la simulation sera précisée dans la phase d'initialisation du modèle.

Les procédures OCCUPATION et RESTITUTION sont chargées de transformer l'état de la portion en insérant ou ôtant une zone à la portion.

III-3-2- Le mécanisme de l'allocation

Correspondants aux quatre procédures get quantum, get quanta, release quantum et release quanta, les modules suivants reproduisent la démarche logique de l'allocation et de la restitution.

1) Le module get quantum

Il alloue une entrée de la table quick-cell : si la table est vide, il appelle le module get quanta pour remplir la quick-cell et alloue le quantu demandé.

2) Le module release quantum

Il restitue le quantum demandé, non pas à la table quick-cell, comme le voudrait la logique du système, mais à la portion elle-même, car, lors des mesures, il existait une "trappe" qui branchait immédiatement le modul release quantum vers le module release quanta.

Ce module ainsi que le module suivant release quanta seront modifiés par la suite pour correspondre à l'algorithme réel (non modifié par la mesure)

3) Le module release quanta

Il restitue au disque les zones données en paramètres et, pour cela, met en oeuvre la procédure RESTITUTION, attribut de la classe PORTION.

4) Le module get quanta

Il alloue le nombre de quanta demandés sur la partition indiquée en paramètres. Mais il se réserve le choix de la portion sur laquelle sera faite l'allocation ; l'allocation effective est faite par la procédure OCCUPATION, attribut de la classe PORTION.

Ce modèle, décrit dans l'annexe 3, sert à vérifier que l'algorithme simulé correspond à l'algorithme mis en oeuvre lors de la mesure. Mais, comme nous l'avons précisé, les mesures ont porté sur l'algorithme modifié. Un modèle simplifié permet de reproduire l'algorithme réel, normal.

III-3-3- Le modèle simplifié

Ce modèle permet de ne pas tenir compte des modifications que comportait l'algorithme lors de la mesure. Beaucoup plus simplifié, il reproduit cependant l'algorithme réel dans les cas simples d'allocation. En particulier, l'allocation morcelée ou sur plusieurs portions n'a pas été traitée.

Les données ont dû être modifiées pour permettre de changer l'algorithme. Ainsi à chaque ordre GET est associé l'ordre RELEASE correspondant par l'intermédiaire du temps de rétention de la zone allouée.

C'est ici que le traitement du temps en SIMULA s'est révélé efficace. Les algorithmes RELEASE étant définis comme des processus (préfixés par la classe PROCESS), il suffit d'une instruction :

```
ACTIVATE  RELEASE  AT  TIMEX ;
```

pour que la procédure RELEASE soit exécutée à l'instant souhaité TIMEX.

La description schématisée suivante du modèle illustre cet aspect. Les mots clés et identificateurs sont en majuscules dans cette description.

1) Déclarations

```
SIMULATION BEGIN
PROCESS CLASS RELEASEQUANTUM (ADR) ; INTEGER ADR ;
BEGIN IF la quick-cell est pleine THEN ACTIVATE RELEASEQUANTA (ADR,1,UD2)
      ELSE on restitue le quantum d'adresse ADR à la quick-cell
END ;
PROCESS CLASS RELEASEQUANTA (ADR,N,UD) ; INTEGER ADR, N ; REF (PORTION) UD
BEGIN UD.RESTITUTION (ADR,N) ; $ restitution de N quanta d'adresse ADR au $
                               $ disque UD.$
      IF UD = UD2 $ c'est le disque allouant des quanta à la quick-cell $
         AND la quick-cell dépasse le seuil de vidage
         THEN on vide la quick-cell
END ;
REF (PORTION) UD, UD1, UD2 ;
    $ nous supposerons l'existence de 2 disques, UD1 et UD2, de type $
    $ différent et ne contenant qu'une portion ; les quanta de la $
    $ quick-cell sont alloués sur UD2 ; $
```

2) Mise en oeuvre du modèle

```
DEBUT    : $ initialisation des disques UD1 et UD2 ;
DEMANDE  : $ génération d'une demande $
    DATE := ININT ;      $ lecture de la date de la demande $
    HOLD (DATE-TIME) ;   $ tous les processus actifs ayant un EVTIME $
                         $ compris entre TIME et DATE seront exécutés $
                         $ avant la reprise de ce processus à l'instant DATE $
    ORDRE := ININT ;     $ indicateur d'ordre GET QUANTUM ou GET QUANTA $
    IF ORDRE = 1 THEN  GOTO GETQUANTUM ELSE GOTO GETQUANTA ;
GET QUANTUM :
    RETENTION := ININT ; $ lecture du temps de rétention $
    IF la quick-cell est vide THEN BEGIN INDEX:=2;N:=1;VIDE:=1;GOTO GTA END ;
       mise à jour de la quick-cell et allocation d'un quantum d'adresse ADR
```

```
     RESTI : ACTIVATE NEW RELEASEQUANTUM (ADR) DELAY RETENTION ;
                    $ ce processus est créé (NEW) et rangé dans la SQS $
     GOTO FIN DEMANDE ;

GET QUANTA :
     INDEX := ININT ;          $ lecture du type de disque $
     N := ININT ;              $ nombre de quanta demandés $
     RETENTION := ININT ;      $ lecture du temps de rétention $
     GTA : IF INDEX = 1 THEN  UD :- UD1  ELSE UD :- UD2 ;

allocation des N quanta demandés d'adresse de zone ADR
IF INDEX = 2 AND la quick-cell "sous-passe" le seuil de remplissage
THEN BEGIN remplissage de la quick-cell ; IF VIDE = 1 THEN GOTO RESTI ; END ;

ACTIVATE NEW RELEASEQUANTA (ADR,N,UD) DELAY RETENTION ;
                                $ ce processus est créé (NEW) et rangé dans la SQS ;
FIN DEMANDE :  IF  ce n'est pas la dernière demande  GOTO DEMANDE ;
     édition des résultats

END $ simulation $ ;
```

Tout le programme est préfixé par la classe SIMULATION qui permet d'accéder aux procédures de manipulation de l'horloge TIME.

III-4 - VALIDATION DU MODELE

Le premier modèle construit nous a permis de simuler l'algorithme mis en oeuvre lors de la prise de mesures. Possédant les adresses des zones alloués et l'état des disques à intervalles réguliers dans nos mesures, il nous a été possible, lors de l'exécution de la simulation, d'imprimer les adresses des zones et l'état des disques aux mêmes moments. La comparaison des mesures et des résultats de la simulation indique une parfaite concordance entre ceux-ci*. Ce modèle reproduit donc avec fidélité le système réel, du moins dans le domaine mis en oeuvre par l'algorithme avec les mesures fournies. Tous les cas de l'algorithme n'ont pas été testés au cours de la séance de mesures.

* Au début de nos séances de simulation, nous n'obtenions pas cette concordance. Après vérification, nous nous sommes aperçus que les mesures avaient été stoppées pendant 2 périodes, faussant la comparaison. A l'aide d'une part de l'état d'occupation des disques, avant et après les interruptions, d'autre part du modèle, nous avons reconstitué, ordre après ordre, les mesures man-

C'est ici qu'est soulevé le problème de la validation du modèle (N2) Il faut en premier lieu préciser les hypothèses de départ :

- Notre étude porte uniquement sur les demandes relatives aux disques "système".

- Tous les "cas de figure" de l'algorithme ne sont pas décrits par le modèle.

La validation ne sera effective que sur ce domaine ou sur un domaine inclus.

Il faut en deuxième lieu indiquer la précision avec laquelle nous voulons évaluer nos résultats.

Les critères choisis pour tester les performances de la gestion d'espace sur disque sont au nombre de 2 :

i - La place occupée par les tables décrivant l'espace sur disque et en mémoire.

ii - Le nombre d'entrées-sorties sur disque nécessaire à la lecture et la réécriture des tables.

Dans notre cas, la précision obtenue est totale car nous connaissons exactement la place occupée par les tables et le nombre d'entrées-sorties effectuées pendant la simulation.

Il n'en serait pas de même si nous avions choisi comme critère le temps d'unité centrale utilisé par l'algorithme pour effectuer l'allocation. Nous aurions dû nous contenter d'une approximation.

La validité du modèle étant acquise, nous pouvons passer à l'étape suivante qui est l'optimisation de l'allocation.

CHAPITRE IV

L'ALLOCATION RAPIDE : LA QUICK-CELL

Deux voies différentes ont été suivies pour étudier le problème de la quick-cell.

1°) A l'aide des mesures réelles faites sur le système, nous pouvons étudier grâce au modèle l'influence de la variation des seuils et de la taille de la quick-cell sur les performances.

2°) A l'aide d'un modèle déterministe de la quick-cell, nous essaierons de déduire certaines conséquences quant à la taille de la quick-cell et à la valeur des seuils, conséquences que nous pourrons comparer avec les résultats du modèle réel précédent.

La bande de mesures prises sur le système réel qui sert de base à toutes nos expériences comprend :

4275 allocations (demandes et restitutions) sur le disque de type "Dimas" dont 3372 vers la quick-cell (soit 79 %).

Il est essentiel, afin de situer le problème, de remarquer que ces demandes représenteraient, sans le mécanisme de la quick-cell, l'occupation (en temps d'accès et temps de transfert) de 2,7 % du temps du disque de type "Dimas" et de 1,1 % du temps de ce disque s'il était du type "DIAD"*.

Ces pourcentages indiquent les limites de gain qu'une quick-cell peut apporter à l'utilisation des disques. Il convient de noter que l'unité centrale peut être en attente pendant ces accès aux disques.

En outre, il est possible que ces taux soient plus importants sous une charge différente et rendent le problème plus aigu.

La quick-cell n'a fait, dans ces conditions, que 16 mouvements "principaux" (soient 32 accès) pour 3372 allocations d'un quantum, soit une

* Une lecture/écriture dure 40 ms pour 1 Diad et 100 ms pour 1 Dimas en ne comptant qu'un mouvement de bras pour chaque ensemble lecture puis écriture.

suppression de 99,5 % des accès correspondants et de 79 % du nombre total des accès au disque DIMAS.

Il est intéressant de déterminer l'influence des divers paramètres de la quick-cell : seuils et niveaux, taille.

IV-1 - LE MODELE DE L'ALGORITHME REEL

A) Rappelons ici rapidement les caractéristiques du mécanisme de la quick cell

1. Les allocations de blocs d'un quantum, lorsque le stock peut les servir ou les accueillir, ne nécessitent aucun accès au disque.

2. Une place en mémoire centrale est nécessaire pour répertorier le stock.

3. L'allocation est accélérée, car il n'y a pas de fusion des blocs du stock.

4. Les blocs du stock sont inutilisables pour les demandes de blocs de taille différente.

Les caractéristiques 1 et 3 sont à rechercher, les caractéristiques 2 et 4 sont à éviter lorsque l'on choisit la taille de la quick-cell et la valeur des seuils et de niveaux.

Nos expériences consistent à étudier l'inflence de ces paramètres sur le compromis entre les caractéristiques 1 et 2.

Notre problème a 5 paramètres dont nous indiquons, entre parenthèses les valeurs actuelles :

T taille de la quick-cell (78)

SV seuil de vidage (60)

SR seuil de remplissage (20)

NV niveau de vidage (20)

NR niveau de remplissage (60)

Il parait par ailleurs cohérent de maintenir entre eux la relation suivante :

$0 \leq SR \leq NV \leq NR \leq SV \leq T$

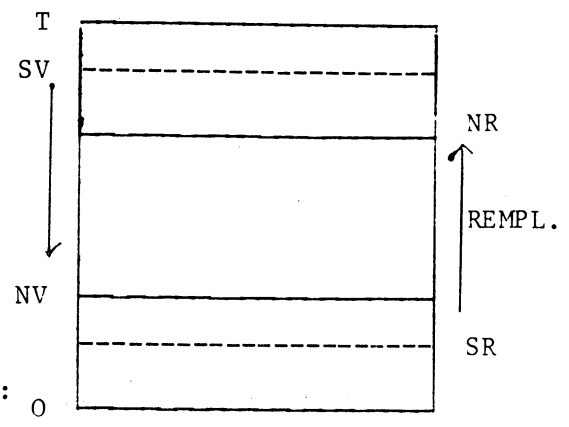


FIGURE 12

Etant donné le nombre de paramètres, il était nécessaire d'établir un plan d'expériences qui ne soit pas prohibitif en temps machine et qui cerne les caractéristiques du problème.

La taille de la quick-cell est le paramètre qui, a priori, semble avoir le plus d'influence sur les résultats : en effet, si la quick-cell passe de la taille 0 à la taille 260, supérieure au plus grand nombre de demandes à un instant donné, le nombre d'E/S (mouvements principaux) décroît de 3372 (le maximum) à 1 (le minimum). Nous décomposerons donc l'étude en deux parties : d'une part, l'influence des seuils et des niveaux, d'autre part l'influence de la taille. Les résultats numériques sont présentés dans l'annexe 4.

B) Influence des seuils et des niveaux

Nous appellerons mouvements les vidages ou remplissages de la quick-cell.

Nous avons fait pour T = 78 et T = 36 des expériences déterminant le nombre de mouvements principaux, puis de mouvements marginaux sans accès supplémentaire au disque en fonction de la valeur des deux seuils et des deux niveaux. L'histogramme de ces expériences est présenté en annexe 4. On constate que le nombre de mouvements principaux varie dans un intervalle étroit. Etudions cela pour la taille 78 (voir l'annexe 4 pour la taille T = 36).

1. Seuils égaux au niveaux

Posons X = NR (niveau de remplissage) = SV (seuil de vidage)
Y = NV (niveau de vidage = SR (seuil de remplissage)

C'est le cas de la quick-cell actuelle. Nous avons la relation $0 \leq Y \leq X \leq T$.

Le nombre de mouvements principaux (figure 19) varie peu, sauf lorsque X se rapproche de la borne 0 (augmentation du nombre de remplissages principaux), ou Y se rapproche de la borne T (augmentation du nombre de vidages principaux).

Le nombre de mouvements marginaux tend à croître, ce qui n'est pas surprenant lorsque X et Y sont proches l'un de l'autre (figure 20), alors que le nombre de mouvements principaux est peu sensible à cette approche (figure 21).

2. Seuils différents des niveaux : (cas général)

Notons D_1 = SV - NR l'écart entre le seuil de vidage et le niveau de remplissa

D_2 = NV - SR l'écart entre le niveau de vidage et le seuil de remplissa

Sur l'ensemble des expériences faites, le nombre de mouvements principaux possède une faible variation et n'est pas influencé par les écarts D_1 et D_2 (figure 23).

La figure 22 nous montre que le nombre de mouvements marginaux (coût en temps d'unité centrale) croît au fur et à mesure que les différences D_1 et D_2 tendent vers zéro.

Nous avons exprimé le nombre de mouvements marginaux NM par une expression de la forme :

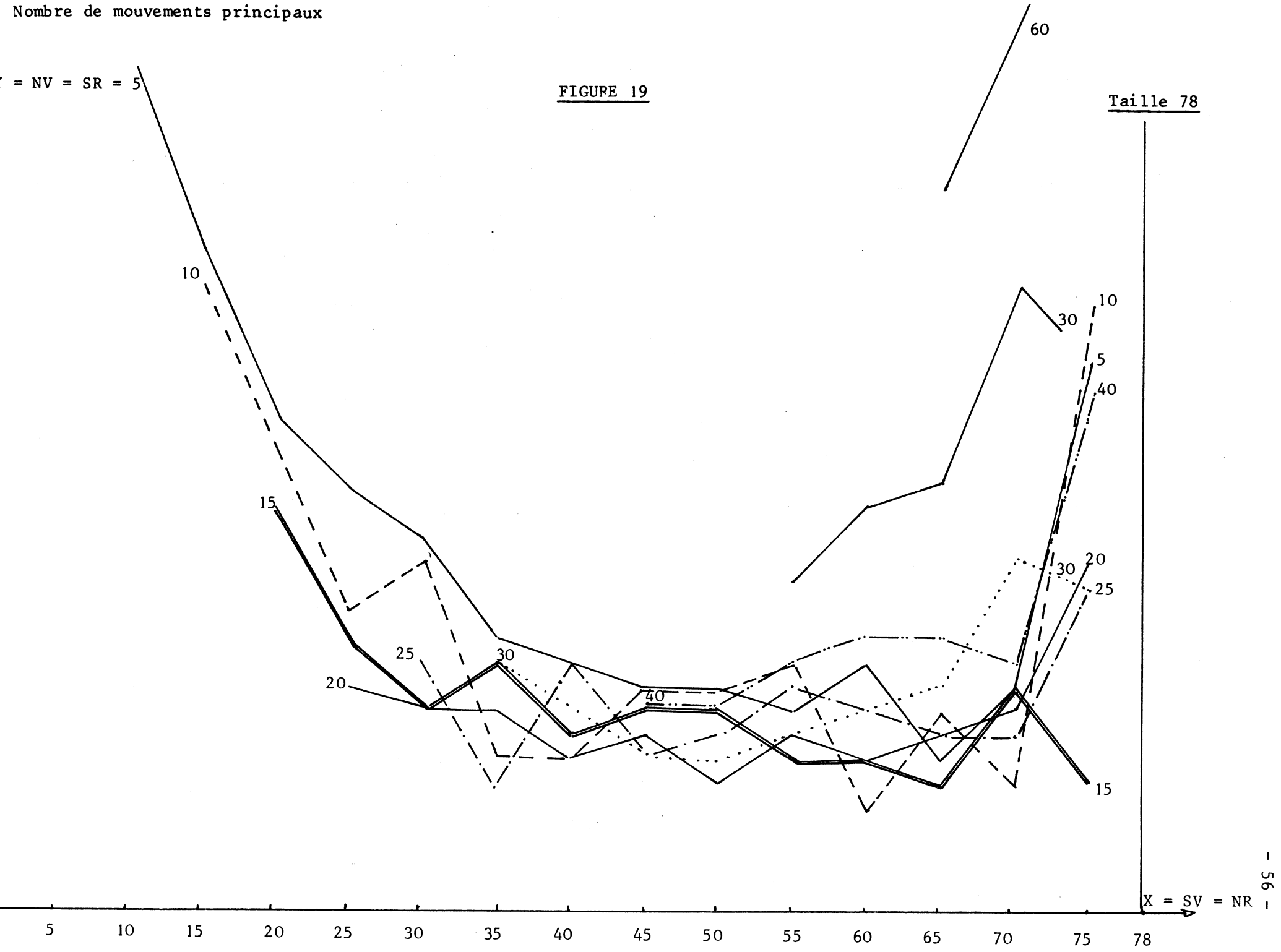
$NM = a_o + a_1 D_1 = a_2 D_2 + \varepsilon$, où a_o, a_1, a_2 sont des coefficients et ε une variab résiduelle. En choisissant a_o, a_1 et a_2 par la méthode des moindres carrés, on trouve un coefficient de corrélation de 0,6 entre NM et $a_o + a_1 D_1 + a_2 D_2$. Ce qui tend à montrer que D_1 et D_2 ont une influence notable sur le nombre de mouvements marginaux.

A titre d'exemple, pour la taille T = 78 le meilleur résultat a été obtenu avec les valeurs suivantes des seuils et niveaux : SR = 5, NV = 15, NR = 55, SV = 60.
Le nombre de mouvements marginaux est alors de 14 et le nombre de mouvements principaux de 12.

En conclusion, on peut s'interroger sur le rôle exact que jouent les seuils. En théorie, les mouvements marginaux, qui sont mis en oeuvre lorsqu'un seuil est dépassé, devraient remplacer les mouvements principaux dûs à un débordement de la quick-cell. Or le nombre de mouvements principaux est parfaitement insensible à la croissance des mouvements marginaux.

Nombre de mouvements principaux

FIGURE 19



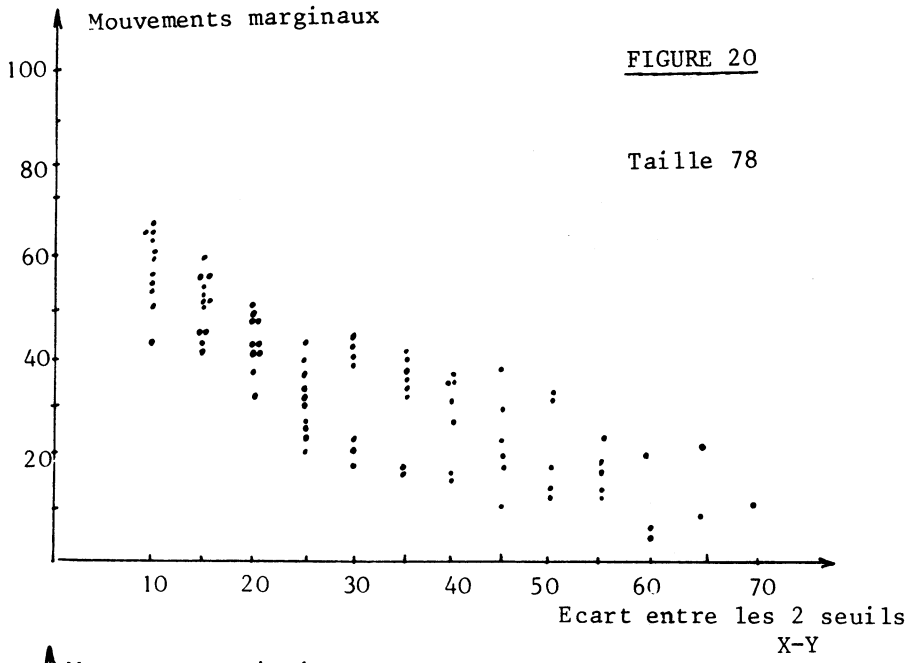


FIGURE 20

Taille 78

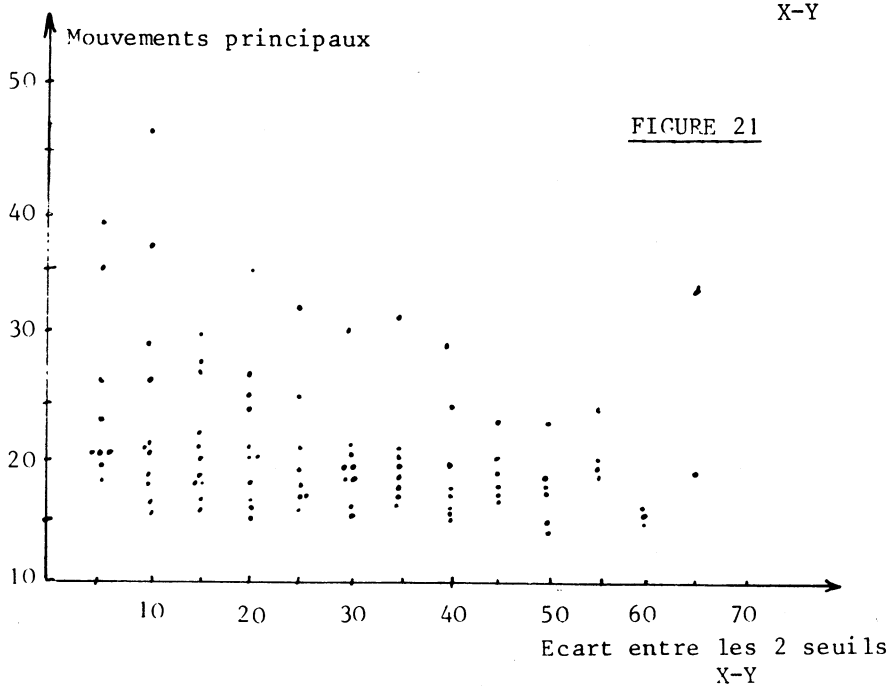


FIGURE 21

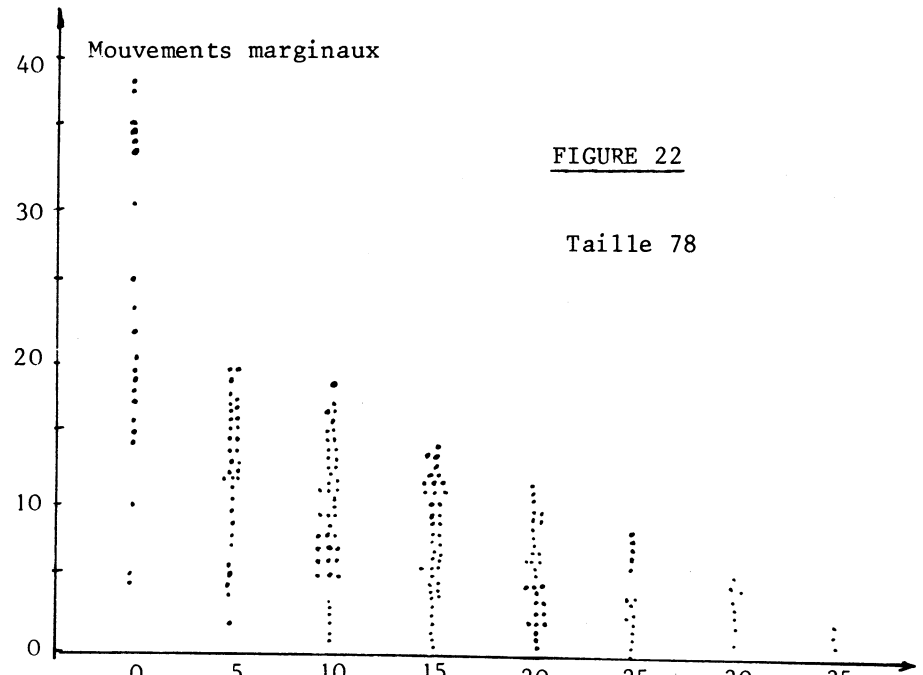


FIGURE 22

Taille 78

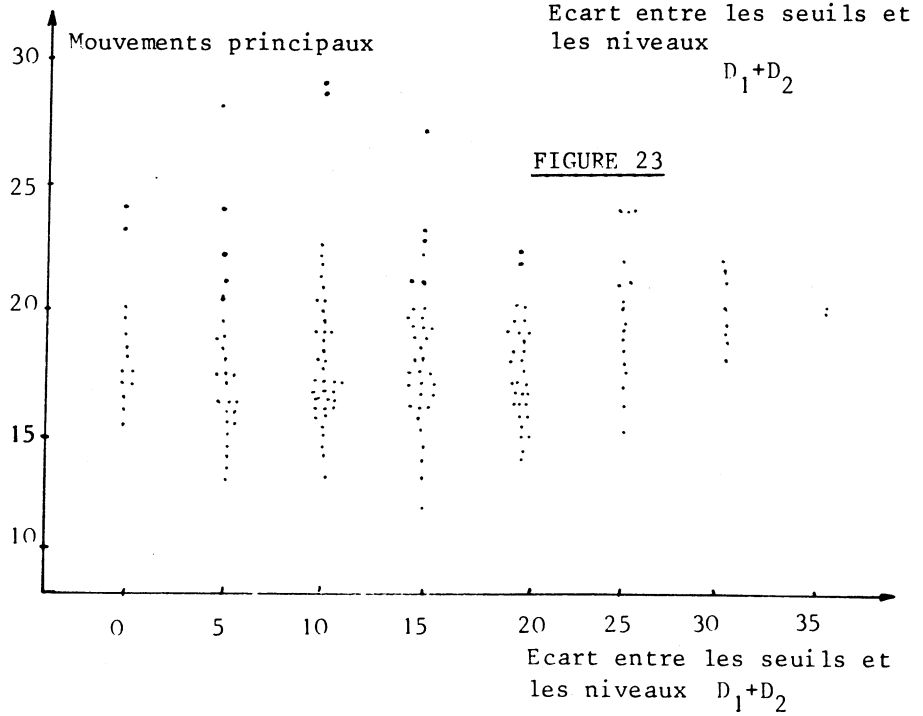


FIGURE 23

Une solution envisageable est de supprimer ces seuils d'alarme trop perfectionnés pour apporter une amélioration sensible au fonctionnement de la gestion d'espace sur disque. L'avantage serait de simplifier les programmes d'allocation et de restitution.

Dans le cas où l'utilisation de ces seuils serait conservée, il conviendrait de donner une valeur suffisante aux écarts D_1 et D_2 entre les niveaux et les seuils, pour réduire le nombre de mouvements marginaux ; quant aux niveaux, il suffit de ne pas leur donner une valeur trop proche des bornes o et T.

C) Influence de la taille

Nous avons fait varier la taille de la quick-cell, en choisissant des niveaux et des seuils arbitraires puisque la dispersion associée reste faible.

Nous présentons sur la figure 24 le nombre de mouvements principaux P en fonction de la taille T de la quick-cell. Les trois séries de mesures présentées correspondent à des modifications de la gestion de la quick-cell.

1°) Algorithme actuel : (mesures représentées par les signes "+")

L'aspect linéaire de la courbe obtenue, en échelle logarithmique, semble montrer la relation $P = \frac{C}{T^a}$, où a et C sont des constantes.

L'approximation graphique nous donne : $\boxed{P = \frac{3000}{T^{1,15}}}$

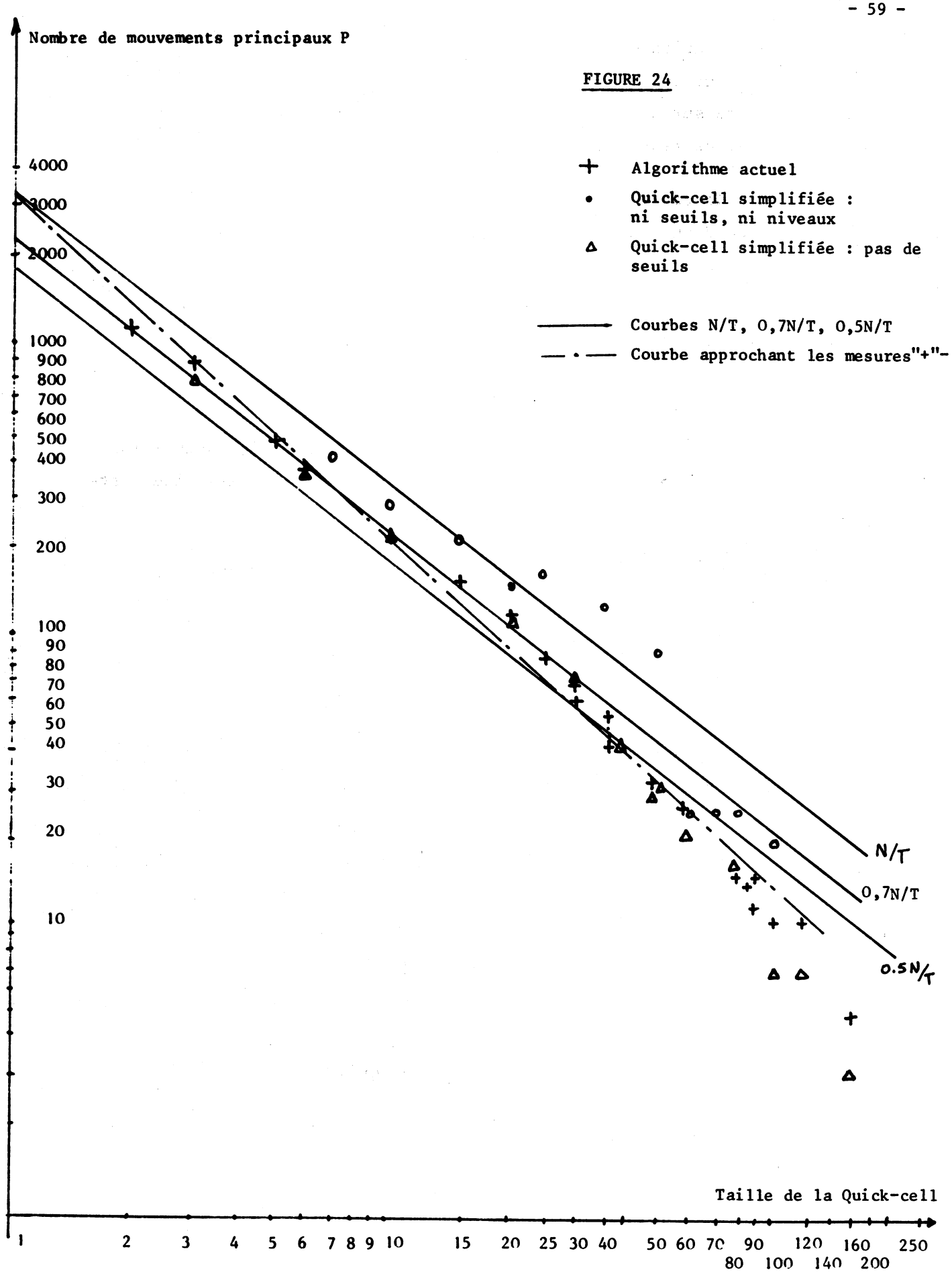
la formule est valable pour $T \in [2,120]$

2°) Suppression des seuils : (mesures représentées par les signes "Δ")

C'est l'hypothèse qui avait été envisagée dans le paragraphe précédent sur l'étude des seuils et des niveaux. Lorsque la quick-cell est vide, elle est remplie jusqu'au niveau NR. Lorsque la quick-cell est pleine, elle est vidée jusqu'au niveau NV ; mais il n'y a plus de test des seuils d'alarme dans les procédures get et release quanta, pour savoir si on doit remplir ou vider la quick-cell.

Les résultats obtenus sont meilleurs qu'avec la quick-cell actuelle, dans presque tous les cas.

FIGURE 24


Nombre de mouvements principaux P
4000
3000
2000
1000
900
800
700
600
500
400
300
200
100
90
80
70
60
50
40
30
20
10
+ Algorithme actuel
• Quick-cell simplifiée : ni seuils, ni niveaux
△ Quick-cell simplifiée : pas de seuils
Courbes N/T, 0,7N/T, 0,5N/T
Courbe approchant les mesures"+"-
N/T
0,7N/T
0.5N/T
Taille de la Quick-cell
1
2
3
4
5
6
7
8
9
10
15
20
25
30
40
50
60
70
80
90
100
120
140
160
200
250

3°) Suppression des seuils et des niveaux : (mesures représentées par les signes "O")

C'est un cas limite du cas précédent.

Le niveau NR devient égal à la taille T de la quick-cell et le niveau NV devient égal à zéro. On remplit et on vide entièrement la quick-cell. Ce sont les plus mauvais résultats obtenus. Les marges laissées dans le cas précédent (entre O et NV ou NR etT) servaient "d'amortisseurs".

Un ordre get quantum qui suivait un vidage de la quick-cell n'entraînait pas un remplissage immédiat, alors que c'est ce qui se produit ici et ce qui explique la dispersion plus grande des mesures par rapport à la droite approchant ces mesures (méthode des moindres carrés).

Les 3 séries de mesures peuvent être approchées par une relation de la forme :

$$P = \frac{\alpha N}{T^a}$$

où P représente le nombre de mouvements principaux par unité de temps

T la taille de la quick-cell

N le nombre d'ordres GET et RELEASE QUANTUM par unité de temps. α et a sont des constantes dépendant de l'algorithme et de la distribution des ordres.

La discussion de cette formule sera faite dans le paragraphe suivant. La taille est le facteur prépondérant des performances de la quick-cell et nous avons déjà précisé comment la quick-cell pouvait réduire le nombre d'entrées sorties sur disque jusqu'à 1. Les résultats numériques sont présentés dans l'annexe 5.

Le problème du choix de la taille nécessite la connaissance des coûts d'une place dans la quick-cell (C1) et d'une allocation supplémentaire, soient 2 accès au disque (C2).

Le coût (F) d'opération de l'algorithme est alors :

$$F(T,P) = C_1 T + C_2 P.$$

Dans le cas de l'algorithme actuel, $F(T,N) = C_1 T + C_2 \frac{3000}{T^{1,15}}$

Le minimum s'obtient pour $\boxed{T_o = k \left(\frac{C_2}{C_1}\right)^{1/2,15}}$ où $k = (1,15 \times 3000)^{1/2,15}$

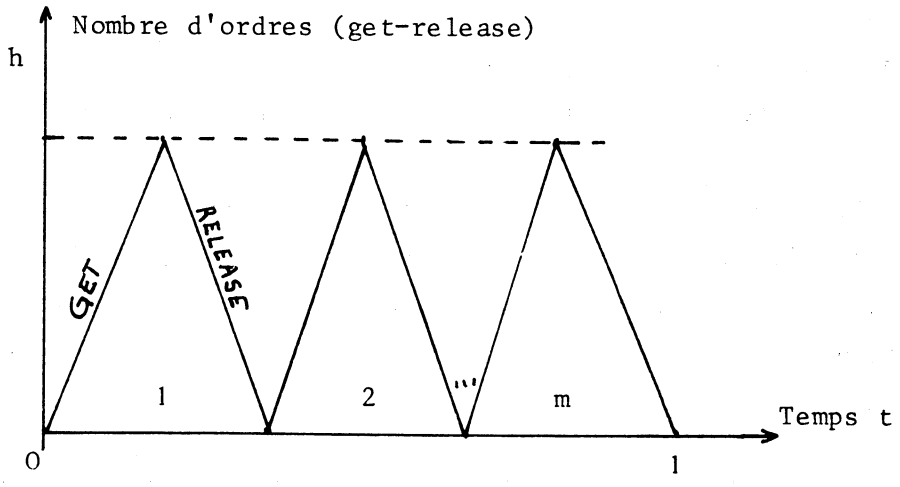
Il serait intéressant de savoir si un fonctionnement reste relativement proche de l'optimum. La quantité $\frac{F(T,P) - F(T_o,P_o)}{F(T_o,P_o)}$ peut servir à cet effet.

IV-2 - LE MODELE DETERMINISTE

Les résultats de la simulation nous fournissent des courbes qui sont intéressantes dans le sens où nous avons la performance du système face à une demande, mais ils n'expliquent pas le phénomène observé.

Reprenons l'étude la figure 14, que nous avons reproduite sur la figure 25 pour plus de commodité. Nous observons que les ordres get quantum arrivent par paquets et sont rendus par paquets, ce qui se traduit par des pics sur la figure.

Simplifions la distribution de la demande : soit la distribution suivante, les ordres GET et RELEASE QUANTUM.



Les ordres get quantum arrivent par paquets de h ordres (montée du pic), puis ils sont tous rendus (descente) et ce phénomène se répète m fois par unité de temps.

Il y a $N = 2\,m\,h$ ordres par unité de temps.

Etudions la réponse du système (le nombre de mouvements principaux P) face à cette distribution avec différents types de quick-cell.

IV-2-1- Quick-cell sans seuils, ni niveaux

Lorsque la quick-cell est pleine (respectivement vide) elle doit être vidée (respectivement remplie) et cela compte pour un mouvement principal. Calculons le nombre P de mouvements par unité de temps.

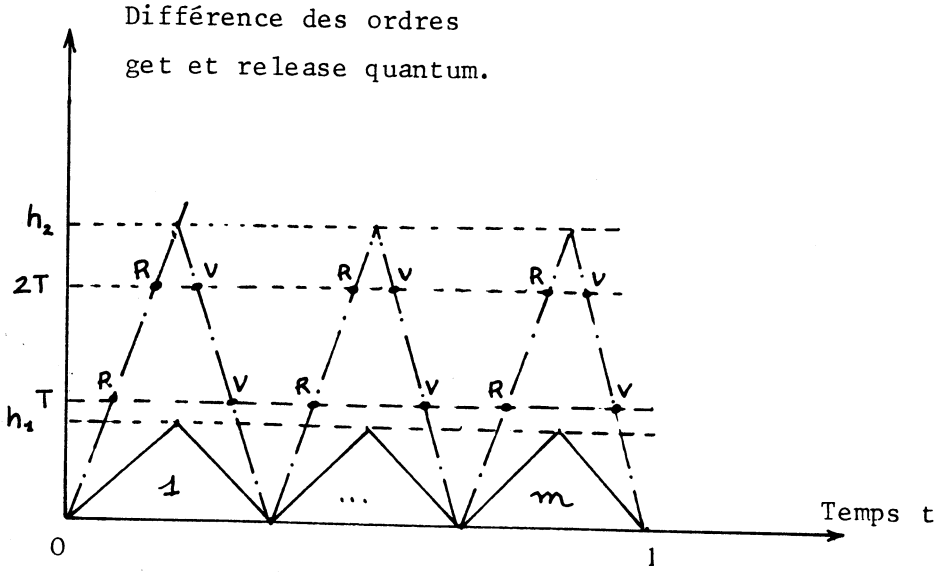
Soit T la taille de la quick-cell. Supposons qu'au temps t = 0 la quick-cell est pleine.
Elle sera vide au Tième ordre et remplie au T+1 ième ordre get quantum.

Posons $h = kT + \alpha$, avec $0 < \alpha \leq T$

Il y a k remplissages et autant de vidages par pic, donc 2k mouvements principaux par pic (voir le schéma qui suit).

D'où $P = 2k \times m$

Si k = 0 : $\boxed{P = 0}$, il n'y a aucun remplissage de la quick-cell, ni vidage ; ce cela tient au fait que $h \leq T$. La quick-cell absorbe toutes les variations de la demande.

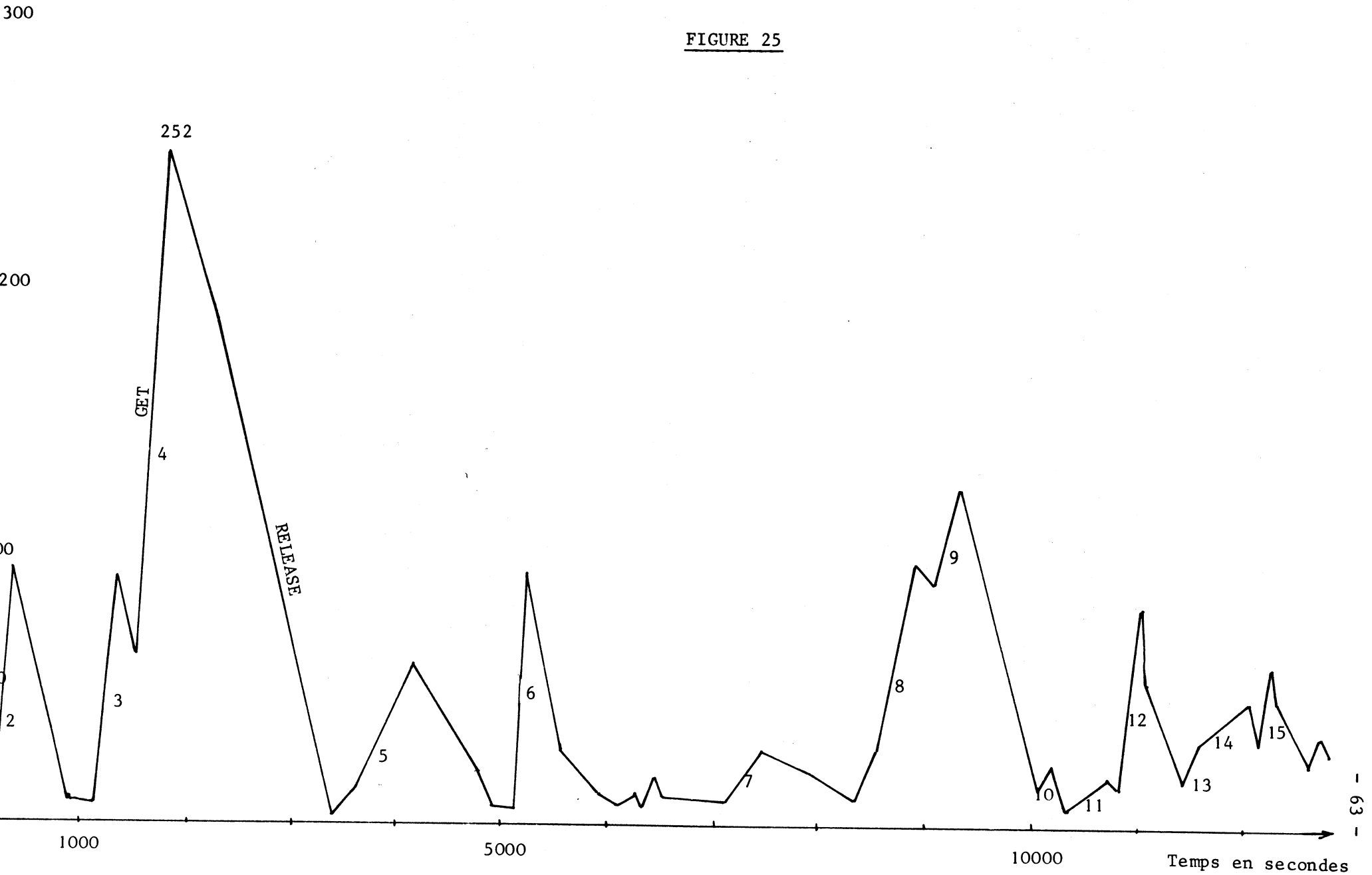


——— cas où $h = h_1 \leq T$

—·— cas où $h = h_2 > T$

Nombre de demandes - Nombre de restitutions

FIGURE 25



Si $k \neq 0$: $P = 2k \times m = 2m \times \frac{h-\alpha}{T} = \frac{2m\,h}{T} - \frac{2m\,\alpha}{T}$

$P = \frac{N}{T} - \frac{2m}{T}$; en moyenne α sera égal à $\frac{T}{2}$, d'où

$P \simeq \frac{N}{T} - \frac{2m}{T} \times \frac{T}{2} = \frac{N}{T} - m$; en général m est petit devant $\frac{N}{T}$

d'où $\boxed{P \simeq \frac{N}{T}}$ (formule 1)

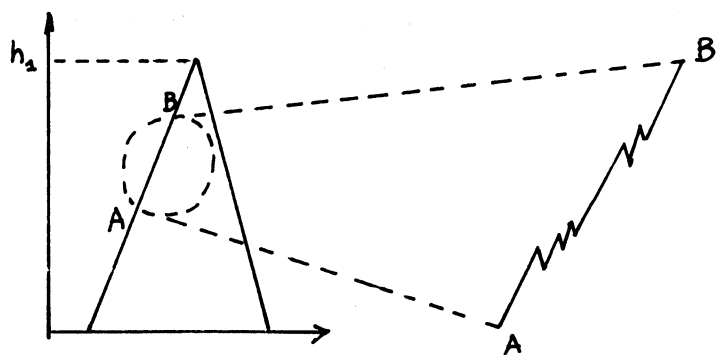
La quick-cell joue un rôle démultiplicateur : il y a un mouvement principal tous les T ordres.

Ce modèle simple suffit à expliquer pourquoi le nombre de mouvements principaux est nul si la taille de la quick-cell est supérieure à 252 (le remplissage initial n'étant pas compté) dans le cas de nos mesures.

Nous avons tracé sur la figure 22, en échelle logarithmique, la droite N/T ; sur ce même dessin sont représentées les mesures faites avec le modèle pour une telle quick-cell. Pour certaines mesures, le nombre pratique de mouvements P est supérieur au nombre théorique $\frac{N}{T}$, ce qui n'est pas normal, ce nombre $\frac{N}{T}$ étant <u>un maximum</u> (les petis pics sont absorbés par la quick-cell sans mouvements supplémentaires).

Mais cela s'explique par le fait que ce type de quick-cell est très instable : il suffit qu'un ordre get quantum suive un vidage de la quick-cell pour qu'aussitôt elle soit remplie, puis vidée si un ordre release suit, etc...

En réalité, il n'y a pas h ordres get quantum, suivis d'autant d'ordres release, mais $h_o = h_1 = h_2$ ordres get parmi lesquels s'intervalent h_2 ordres release quantum (ainsi sur la figure 23, le plus grand pic, de hauteur $h_1 = 252$ correspond à $h_o = 340$ ordres get quantum, moins $h_2 = 88$ ordres release quantum). Un "zoom" sur ce pic montrerait de très nombreuses variations :



Dans la montée du pic, il y a h_1/T remplissages ; il y a aussi h_2 ordres release d'où la probabilité $h_2/(h_1+h_2)$ pour qu'un ordre release suive un ordre get ; ce qui explique la probabilité non négligeable d'avoir un remplissage suivi d'un vidage.

Pour éviter ces mouvements inutiles, nous introduisons des niveaux de remplissage et de vidage dans la quick-cell.

Calculons d'abord le nombre de mouvements P, s'il n'y avait pas ces mouvements néfastes :

$P \simeq \frac{2m\ h_1}{T}$; posons $\alpha_1 = \frac{h_1}{h_o+h_2} = \frac{\text{le nombre d'ordres/get-release}}{\text{le nombre d'ordres (get-release)}} = \frac{h_1}{h}$

$P \simeq \frac{2m\ h\ \alpha_1}{T}$; or $N = 2m\ h$ (formule 2), d'où

$$\boxed{P \simeq \alpha_1 \frac{N}{T}}$$

avec $0 \leq \alpha_1 \leq 1$: ce nombre de mouvements est théorique, car il y a <u>en plus</u> les mouvements inutiles précédents.

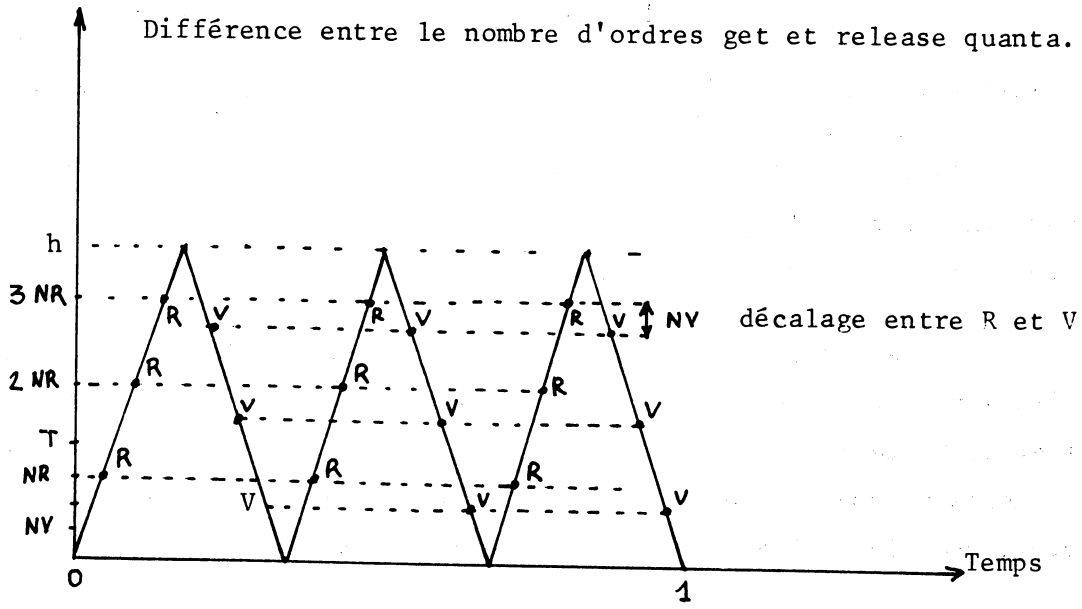
Pour l'ensemble des pics de la mesure $\boxed{\alpha_1 = 0,49}$

IV-2-2- <u>Quick-cell avec niveaux</u>

Soient NR et NV les niveaux de remplissage et de vidage qui satisfont à la relation : $0 \leq NV \leq T$.

Prenons les niveaux symétriques par rapport aux bornes 0 et T pour simplifier : $T - NR = NV$ et soit $\alpha_2 = \frac{NR}{T}$, avec $0 \leq \alpha_2 \leq 1$.

Par rapport au cas précédent, la quick-cell se comporte comme si elle avait une taille $T' = NR$ plus petite que T et les remplissages et les vidages sont décalés de $NV = T - NR$ ordres.



Posons $h = kNR + a$; il y a $2m\,k$ mouvements principaux par unité de temps ; d'où $P = 2m\,k = \frac{2m(h-a)}{NR} = \frac{N}{NR} - \frac{2m\,a}{NR}$; prenons en moyenne $a = \frac{NR}{2}$ $(0 < a \leq NR)$ alors $P \simeq \frac{N}{NR} - m$; m est petit devant $\frac{N}{NR}$, d'où

$$\boxed{P \simeq \frac{N}{\alpha_2\, T}} \qquad \text{(formule 3)}$$

Avec ce modèle de demande, cette quick-cell a un comportement moins bon que la précédente.

Reprenons le modèle amélioré de la demande : il y a $h_o = h_1 + h_2$ ordres get et h_2 ordres release dans une montée de pic, dont la hauteur $h = h_1$. Les ordres h_2 font subir à la droite de la montée des oscillations de faible amplitude (on suppose qu'elles sont inférieures à NV).

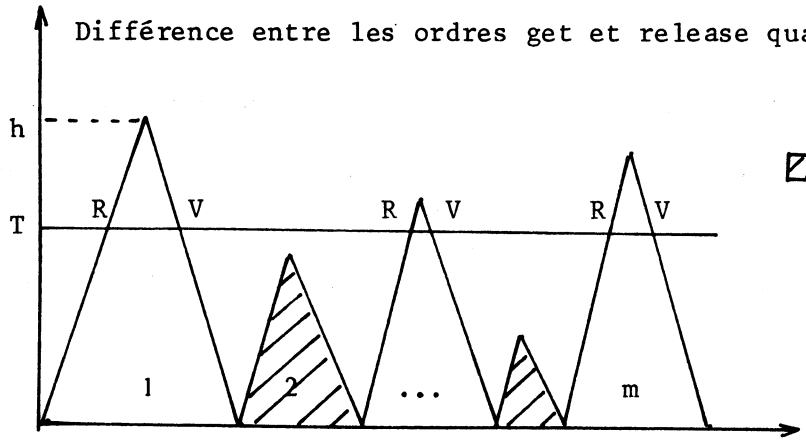
Dans ce cas, cette quick-cell absorbe ces variations et le nombre de mouvements principaux devient, en reprenant la formule 2 :

$$\boxed{P = \frac{\alpha_1\, N}{\alpha_2\, T}} \qquad \text{(formule 4)}$$

En donnant à α_2 une valeur assez grande pour que la quick-cell joue son rôle démultiplicateur ($\alpha_2 = 0{,}7$) ; mais pas trop grande pour que les fluctuations de faible amplitude soient absorbées, nous obtenons des performances supérieures à celles d'une quick-cell sans seuils, ni niveaux.

En donnant à α_1 la valeur trouvée pour les mesures fournies, la relation $P = \frac{\alpha_1 N}{\alpha_2} = \frac{0,49}{0,7} \frac{N}{T} = \frac{0,7\ N}{T}$ approche les résultats de la simulation (voir figure 22), du moins pour des valeurs de $T \leq 40$. Les performances obtenues par simulation (c'est à dire ce que l'on aurait obtenu en réalité) sont meilleure pour des valeurs de T supérieures à 40 que la formule $P = 0,7\ N/T$.

Ceci s'explique par le fait que le modèle de la demande choisi (les pics sont égaux entre eux) ne correspond pas à une demande réelle. Les pics sont de hauteurs différentes ; pour une taille T donnée, la quick-cell absorbera un certain nombre de pics sans mouvements principaux. Le nombre de pics absorbés croît avec la taille de la quick-cell et dépend de la distribution des pics.



Par suite, le nombre n d'ordres entraînant des mouvements principaux, décroît lorsque la taille T de la quick-cell augmente, ce que l'on peut exprimer par une relation de la forme $n = \alpha T^b$, $\alpha > 0$, $b < 0$; lorsque $T = 1$, $n = N$, par suite $\alpha = N$.

$$\boxed{n = NT^b, \quad b < 0}$$

Le nombre de mouvements principaux P est alors, en appliquant la formule 4 :

$$P = \frac{\alpha_1}{\alpha_2} \frac{NT^b}{T} = \frac{\alpha_1}{\alpha_2} \frac{N}{T^{(1-b)}}$$

b étant négatif, cette formule devient : $\boxed{P = \frac{\alpha_1}{\alpha_2} \frac{N}{T^a}, \quad a > 0}$ avec $a = 1-b$

Ceci explique la formule $P = \alpha \frac{N}{T^a}$ trouvée par approximation (méthode des moindres carrés et méthode graphique) lors des expériences de simulation sur la taille de la quick-cell.

Si on veut introduire les seuils d'alarme, il faut faire intervenir les ordres GET QUANTA et RELEASE QUANTA, car ces seuils sont testés dans ces procédures. Nous ne l'avons pas fait, car les expériences menées avec ce type de quick-cell (algorithme actuel) montrent que les ordres get quanta sont en trop petits nombres et mal répartis, pour avoir une influence notable sur le nombre de mouvements principaux.

L'approximation graphique des résultats de la simulation pour une telle quick-cell, rappelons-le, donne la relation $P = 3000/T^{1,15}$, que l'on peut rapprocher de la formule précédente $P = \frac{\alpha_1}{\alpha_2} \times \frac{N}{T^a}$: on obtient $a = 1,15$ et $\frac{\alpha_1}{\alpha_2} = \frac{3000}{3372} = 0,89$.

CHAPITRE V

GENERALISATION DE L'ALLOCATION RAPIDE

INTRODUCTION

Pour réduire les temps des algorithmes d'allocation et de restitution (temps des entrées-sorties), on envisage de ne plus essayer de ressouder les zones rendues aux zones libres du disque ; on garde en mémoire centrale une liste des zones rendues qui permet, lors d'une demande, une allocation rapide, sans mise à jour complexe de l'espace sur disque, ni entrées-sorties sur disque.

Cette liste est décomposée en sous-listes, chaque sous-liste conservant des zones de taille donnée. La longueur maximum de la sous-liste dépend de la fréquence de la demande pour cette taille, elle est nulle pour les fréquences trop faibles.

V-1 - EXTENSION DE LA QUICK-CELL A TOUTES LES ALLOCATIONS D'UN QUANTUM

Pour être servi par la quick-cell, il faut le demander explicitement ; or, parmie les allocations de taille quelconque n'appelant pas ce mécanisme, 197 sur 900 concernent des tailles d'un quantum.

En faisant prendre en compte ces demandes par la quick-cell, nous n'avons obtenu aucun mouvement principal supplémentaire, aussi bien avec une taille de quick-cell de 78 que de 36. Cette économie de 7,5 % des accès initiaux ne nécessite qu'un faible changement de programmation consistant à tester les demandes, d'où qu'elles proviennent, pour connaître la taille de la demande.

Rappelons que ce gain est aussi important que celui obtenu en faisant varier la quick-cell de 12 à 260.

V-2 - EXTENSION AUX TAILLES LES PLUS FREQUEMMENT UTILISEES

Nous pouvons utiliser le même mécanisme de quick-cell pour des allocations sur les tailles les plus fréquemment utilisées, dans notre cas 2, 3, 4, 5 et 6 quanta. Cette stratégie avait déjà été appliquée dans l'allocation de mémoire centrale pour le système CP (MI), avec évidemment d'autres critères de performances en vue.

V-2-1- Chaque sous-liste ne comporte qu'un seul élément

On s'aperçoit qu'on obtient en moyenne une réduction de 50 % sur le nombre des accès au disque, à l'occasion d'une demande. En effet, une demande, rencontrant l'élément vide, utilisera l'allocation normale pour être satisfaite, mais remplira par la même occasion cet élément, qui sera disponible lors de la prochaine demande de même taille. Il est possible de remplir dans le même appel les autres éléments vides de taille différente.

V-2-2 - La longueur des sous-listes dépend de la fréquence de la demande

Le tableau suivant montre comment à l'aide de petites quick-cell on arrive à supprimer 97 % des accès au disque, alors que la quick-cell pour la taille 1 ne peut supprimer que 79 % des accès.

Nous avons choisi les petites quick-cell de manière à ce que la configuration 7 occupe autant de place qu'une quick-cell de taille 78. Les fréquences des demandes par taille sont indiquées dans le chapitre sur l'analyse des mesures.

Notons néanmoins, qu'en cas de panne du système où les tables de quick-cell sont inaccessibles, les blocs libres qu'elles contiennent sont inutilisables. Il faudrait prévoir un "nettoyage" périodique des disques pour récupérer ces blocs en balayant tous les fichiers contenus sur ces disques.

Au cas où les tables devraient être réécrites à l'occasion d'allocations, notons que l'économie de 97 % permet une sauvegarde toutes les 105 s $(\frac{117}{12600})$ ce qui peut paraître suffisant.

CONFIGURATION	Nombre d'accès disque	Proportion d'accès / cas sans quick-cell
0- sans quick-cell	4275	1.0
1- Q.cell(T=36) pour allocation de 1 quantum	946	0.22
2- $\hat{m}$ config. avec la modification du paragraphe 1	749	0.173
3- $\hat{m}$ config. que 2 avec Q.cell pour 2 quanta (T=12) $\Sigma T = 48$	513	0.12
4- $\hat{m}$ config. que 3 avec Q.cell pour 4 quanta (T=6) $\Sigma T = 54$	369	0.086
5- $\hat{m}$ config. que 4 avec Q.cell pour 5 quanta (T=6) $\Sigma T = 60$	233	0.054
6- $\hat{m}$ config. que 5 avec Q.cell pour 6 quanta (T=12) $\Sigma T = 72$	143	0.033
7- $\hat{m}$ config. que 6 avec Q.cell pour 3 quanta (T=6) $\Sigma T = 78$	117	0.027

V-3- INFLUENCE DU NOMBRE DE PORTIONS ET DE LA VALEUR DU QUANTUM

Avec les nouveaux disques MD50 de 50 millions d'octets de capacité et les disques MD100 de 100 millions, la taille limitée de la TABLEDISQUE oblige soit à mettre plusieurs tables sur le disque et en mémoire centrale, soit à changer la valeur du quantum dans ces tables. Les hypothèses envisagées pour cette valeur sont des multiples de cette valeur actuelle, soit respectivement 8,16 et 32 k octets.

Les demandes sur la portion "quick-cell" ne sont pas influencées par ces changements car on garde impérativement la valeur d'un quantum de 8 k octets pour la portion quick-cell. Les changements interviennent pour des demandes non égales à un multiple de l'unité d'allocation.

Exemple : Doit-on mettre une portion sur le disque MD50, avec la valeur du quantum de 16 k octets ou 2 portions avec la valeur représentative normale du quantum ?

A) Cas d'une portion

Espace perdu sur le dique : si on demande un nombre impair de quanta, on accorde le nombre pair immédiatement supérieur. Soit f_1, f_2, ..., f_M les fréquences des demandes de taille 1, 2, ..., M. L'espace perdu sur le disque est de $\Sigma_{p=o}^{h} f_{2p+1}$ quanta par unité de temps ($h = \frac{M}{2}$ si M est pair, sinon $h = \frac{M+1}{2}$), l'espace demandé étant : $\Sigma_{p=1}^{M} p \times f_p$, le pourcentage d'espace accordé en trop est :

$$\boxed{\frac{\Sigma_{p=o}^{h} f_{2p+1}}{\Sigma_{p=1}^{M} p \times f_p}} \qquad h = \begin{cases} M/2, \text{ M pair} \\ (M+1)/2, \text{ M impair} \end{cases}$$

Si on fait le calcul avec les ordres GET QUANTA des mesures, nous avons 43 % de demandes impaires qui représentent 10 % de l'espace demandé.

B) Cas de deux portions

- Espace perdu sur le disque = taille de la TABLEDISQUE
- Espace perdu en mémoire = taille de la TABLEMEMOIRE
- Temps perdu : lors d'un ordre release qui libère des zones sur 2 portions (2 E/S au lieu d'une) ou lors d'un ordre get qui ne peut être satisfait sur la première portion, saturée. Il n'en sera pas tenu compte dans la suite.

Soit : C_M le coût de la place en mémoire de la TABLEMEMOIRE,
C_D le coût d'un 1 k octet sur disque = coût de la tabledisque.

La différence de coût entre A et B, qu'on notera $C_A - C_B$, est :

$$C_A - C_B = 8\, C_D \left[\Sigma_{p=o}^{h} f_{2p+1}\right] - \left(C_D + C_M\right)$$

Si cette différence est positive, on choisira le cas B : 2 portions.

Mais il faut remarquer que les quanta alloués en trop dans le cas A n'ont pas du tout la même valeur si on se trouve en période de saturation ou non.

Lorsque la demande est très proche de l'espace disponible, on arrive à saturation beaucoup plus rapidement que lorsque la valeur du quantum est de 8 k octets.

Nous aurions pu mener une étude plus précise de l'influence de la valeur du bit sur l'occupation de l'espace mémoire et la saturation, si nous avions disposé de mesures lors d'une saturation de l'espace sur disque.

A titre indicatif, précisons que le "buddy-system", allocation de bloc de taille égale à une puissance de 2, nécessite près de 40 % de mémoire supplémentaire, lorsque la demande n'est pas contrainte à être une puissance de 2. (En supposant que l'on n'accorde pas de demande en plusieurs morceaux). C'est une borne supérieure pour l'espace perdu si l'on prend une valeur du quantum de 16 ou 32 k octets.

S'il y a beaucoup de disques, un mécanisme tampon pour les tables TABLEMEMOIRE pourrait se justifier : on ne garderait en mémoire centrale que les dernières tables utilisées.

5-4 - CONCLUSION

Rappelons que la représentativité des données qui nous servent de base peut être mise en doute. Cependant, le fait que la dispersion des tailles demandées reste faible paraît être une caractéristique du système. Il est préférable de partager l'espace pris par une seule quick-cell, pour les allocations de taille 1, entre plusieurs tailles fréquemment utilisées. Nous avons réduit le nombre d'accès disque de 97 %. C'est à dire qu'il n'y a plus que 3 % d'ordres "GET QUANTA" permettant de tester la valeur des seuils. Les bons résultats obtenus confirment ce que l'on avait vu dans les paragraphes 1 et 2, à savoir que les seuils n'ont pas d'influence notable sur le comportement du système.

CONCLUSIONS GENERALES

Pour éliminer des accès aux disques lors d'allocations, nos expériences ont montré que :

1°) L'algorithme actuel supprime 78 % de tous les accès.

2°) La taille de la quick-cell est un paramètre plus important que les seuils et les niveaux.

3°) Il est préférable de partager l'espace pris par une seule quick-cell (pour les allocations d'un quantum) entre plusieurs tailles fréquemment utilisées. Ce mécanisme d'allocation rapide est très efficace et on gagnerait en le généralisant (jusqu'à 97 % des accès).

4°) Le rôle des seuils, donc des mouvements marginaux, est inutile, du moins dans le cas de demandes ayant la physionomie de celles fournies par les mesures. Leur suppression permettrait de simplifier les algorithmes.

Nous rappelons que la représentativité des données qui nous servent de base peut être mise en doute. Ainsi, la saturation de l'espace disque, ainsi que le morcellement du disque en petits blocs n'a pu être étudié, car 92 % du disque DIMAS est inoccupé.

Les recommandations, que nous pouvons faire à l'issue de cette étude, concernent la méthode de réalisation des algorithmes de gestion.

Il est préférable de réaliser des algorithmes très simples, puis d'apporter des améliorations, suivant leurs importances relatives. Ceci peut se faire à l'aide de compteurs sur les services demandés ou effectués, incorporés à ces algorithmes.

L'amélioration d'un de ces algorithmes pourra être définie par une étude de simulation s'appuyant sur les données recueillies. Elle pourra conduire à une gestion dynamique qui utilisera elle-même les mesures faites.

Ainsi l'algorithme de gestion de l'espace sur disque, le cas d'allocation en plusieurs morceaux ou sur plusieurs disques auraient pu être ignorées et les seuils d'alarme supprimés.

ANNEXES

ANNEXE 1 : DESCRIPTION DE LA TABLE D'ALLOCATION RAPIDE.

ANNEXE 2 : ALGORITHMES DES PROCEDURES D'ALLOCATION ET DE RESTITUTION.

ANNEXE 3 : MODELE DE SIMULATION, CAS DE L'ALGORITHME DE LA MESURE.

ANNEXE 4 : EXPERIENCES DE SIMULATION : ETUDE DES SEUILS ET NIVEAUX.

ANNEXE 5 : EXPERIENCES DE SIMULATION : ETUDE DE LA TAILLE.

ANNEXE 1

TABLE D'ALLOCATION RAPIDE

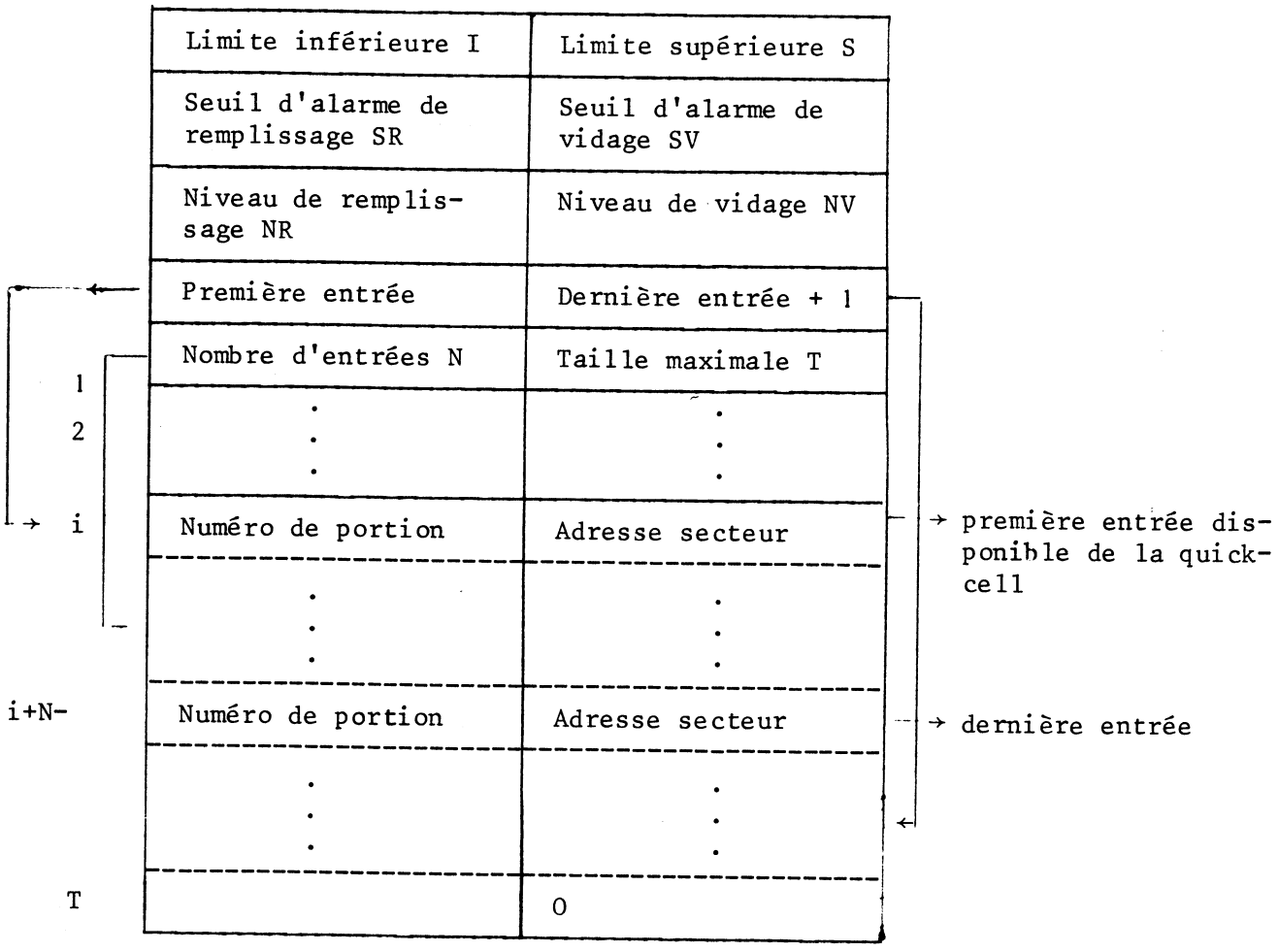
ANNEXE 1

TABLE D'ALLOCATION RAPIDE

Par analogie avec le système de distribution du commerce, la chaîne (grossiste → détaillant → client) est ici représentée par (portion ↔ quick-cell ↔ demande).

La portion délivre à la quick-cell un stock de quanta et la quick-cell les délivre un par un aux clients (les demandes). D'ailleurs, la quick-cell est gérée comme un stock avec une gestion à deux seuils. En réalité, le retour des quanta à la quick-cell symétrise cette gestion, puisque le stock peut déborder.

LA TABLE D'ALLOCATION RAPIDE "QUICK-CELL"



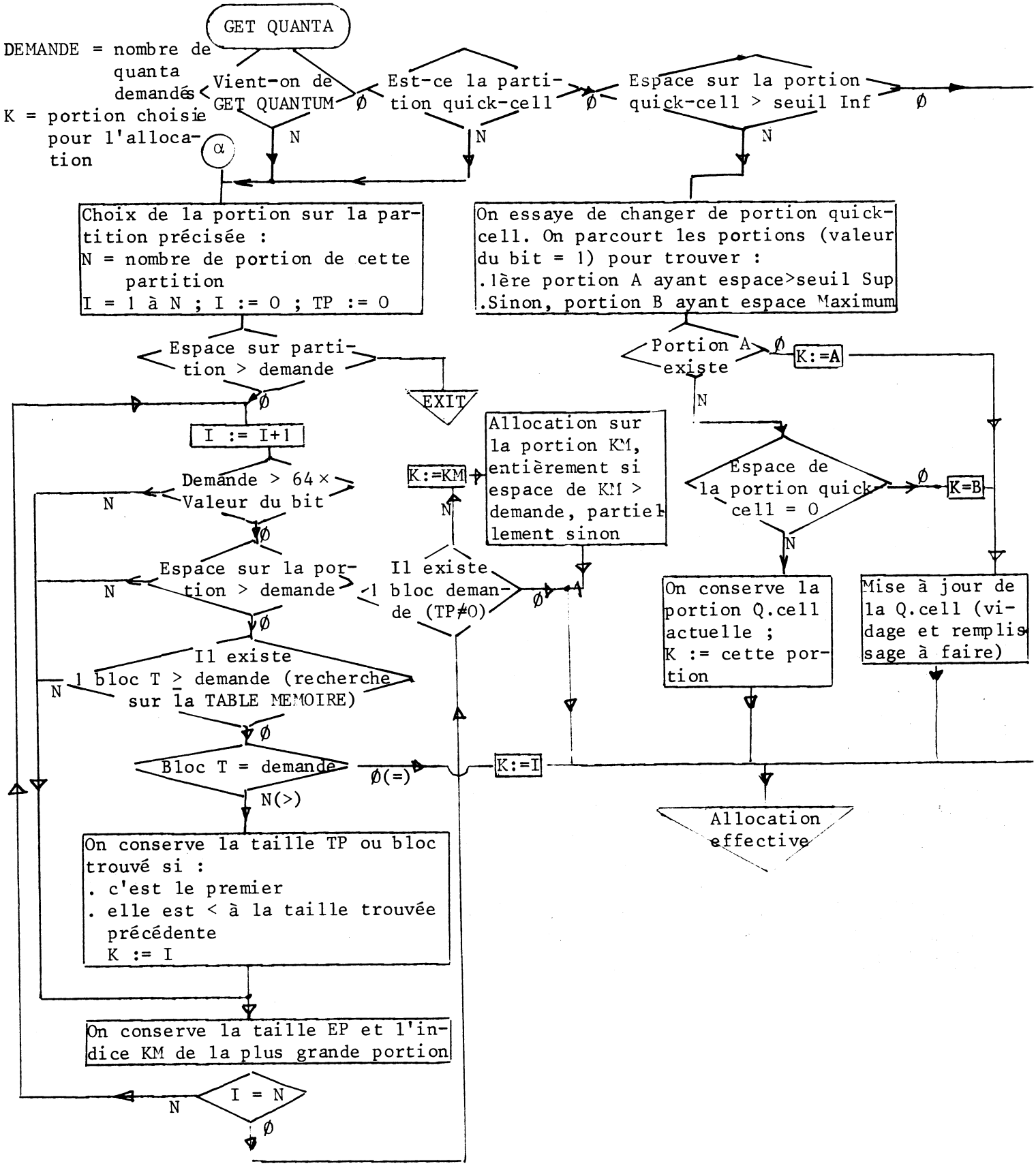
Occupation mémoire : 263 mots au maximum

ANNEXE 2

ALGORITHMES DES PROCEDURES D'ALLOCATION ET DE RESTITUTION

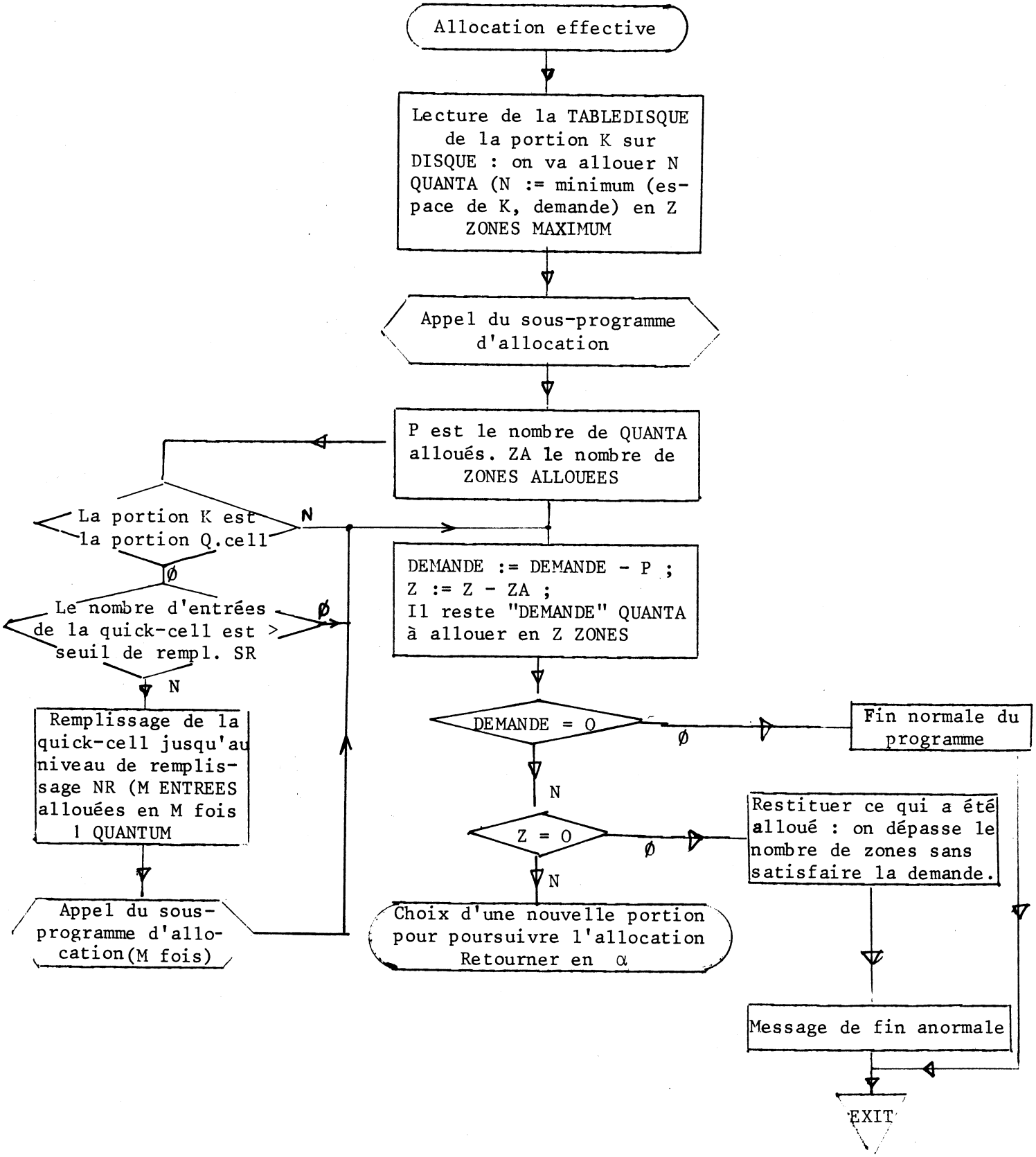
ANNEXE 2

ALGORITHME DE LA PROCEDURE "GET QUANTA"



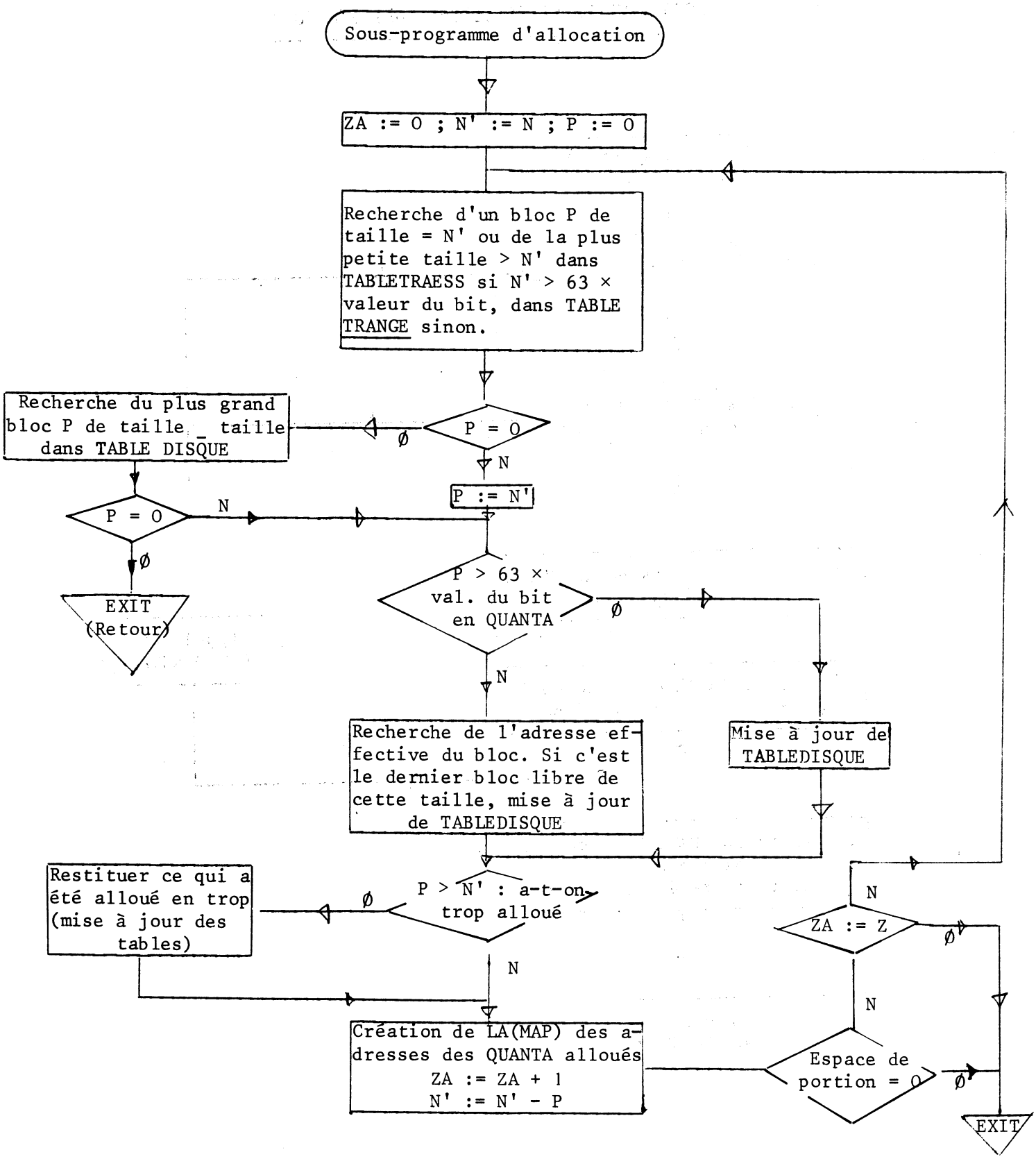
ANNEXE 2

ALGORITHME DE LA PROCEDURE "GET QUANTA"



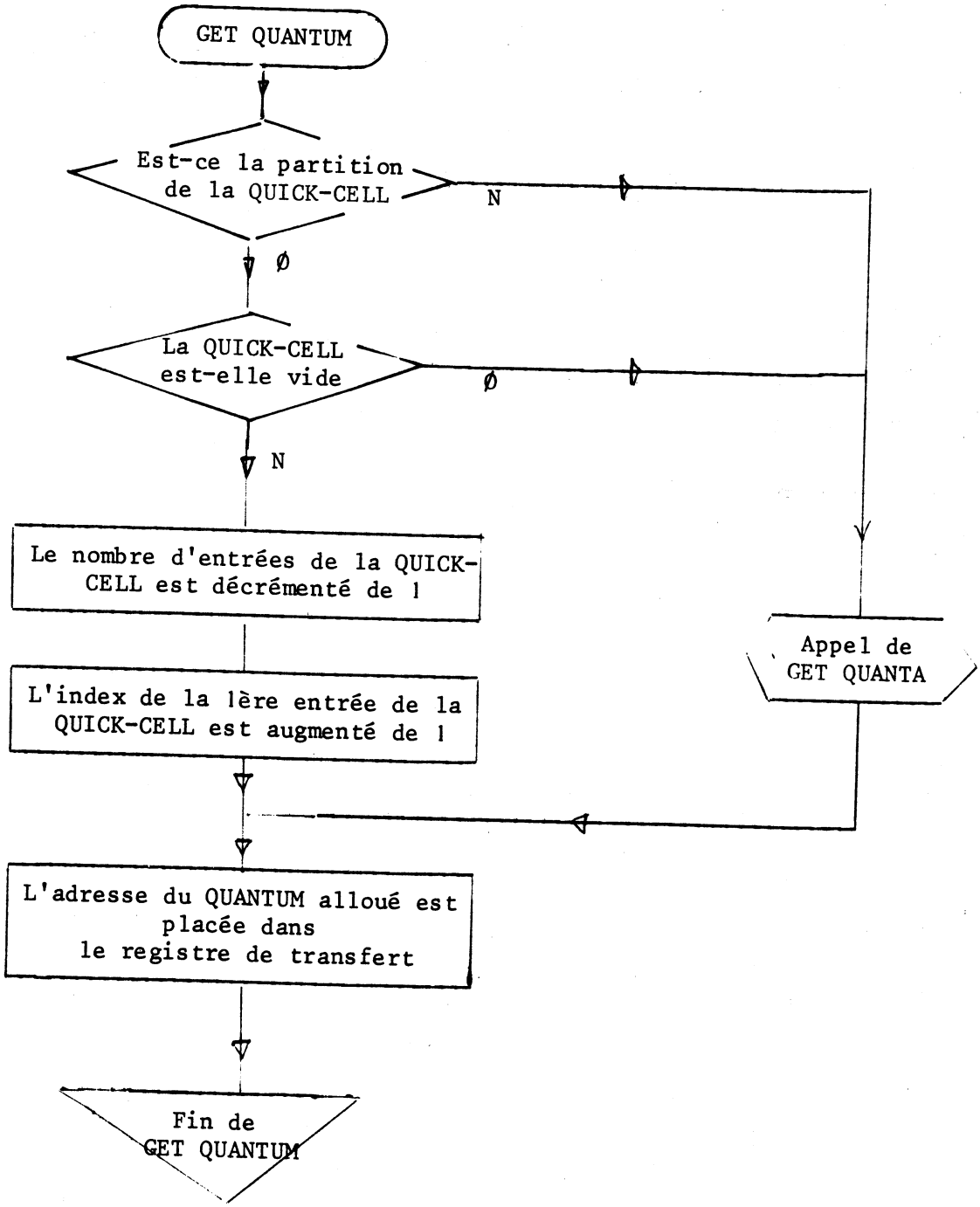
ANNEXE 2

ALGORITHME DE LA PROCEDURE "GET QUANTA"



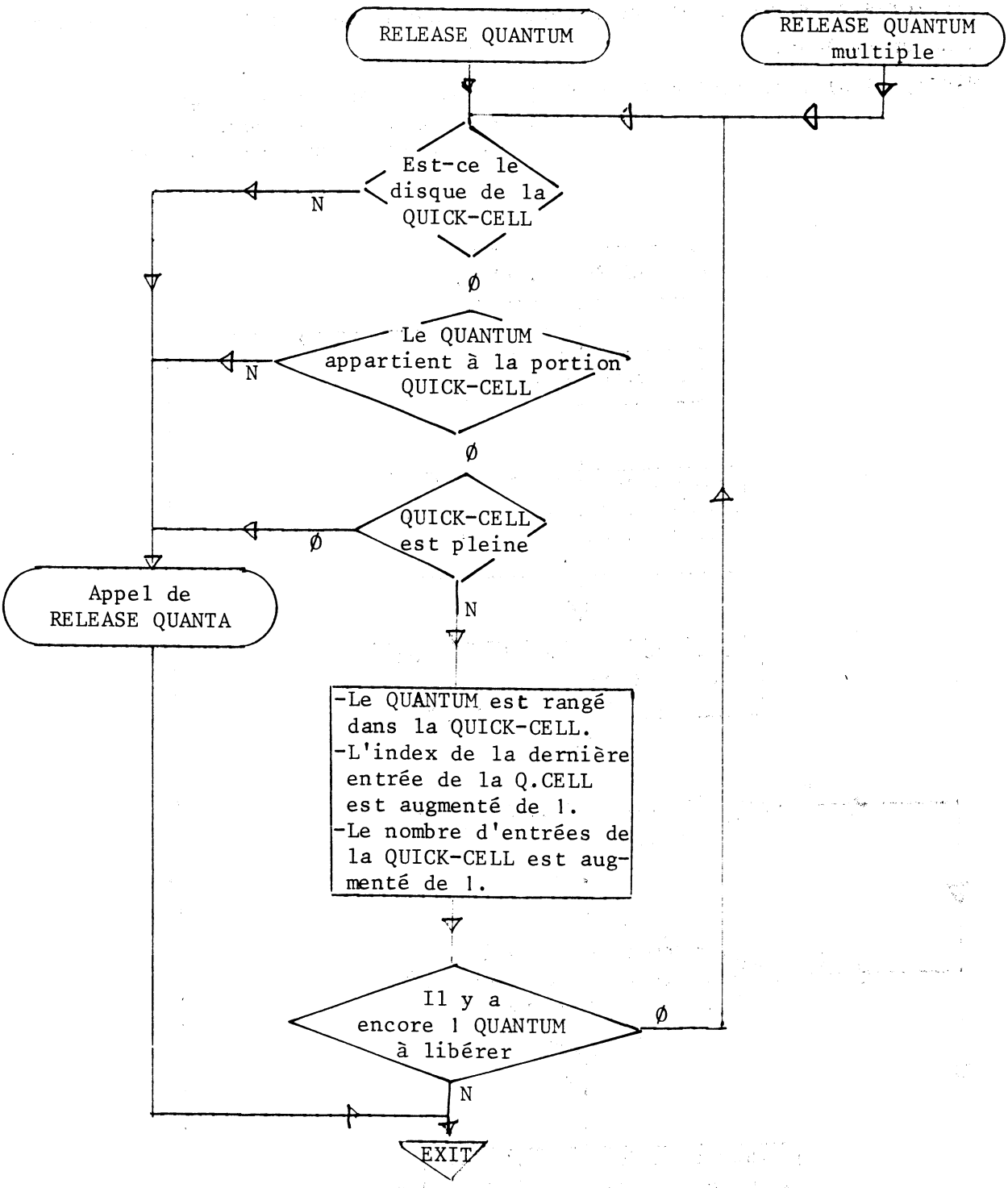
ANNEXE 2

ALGORITHME DE LA PROCEDURE "GET QUANTUM"



ANNEXE 2

ALGORITHME DE LA PROCEDURE "RELEASE QUANTUM"

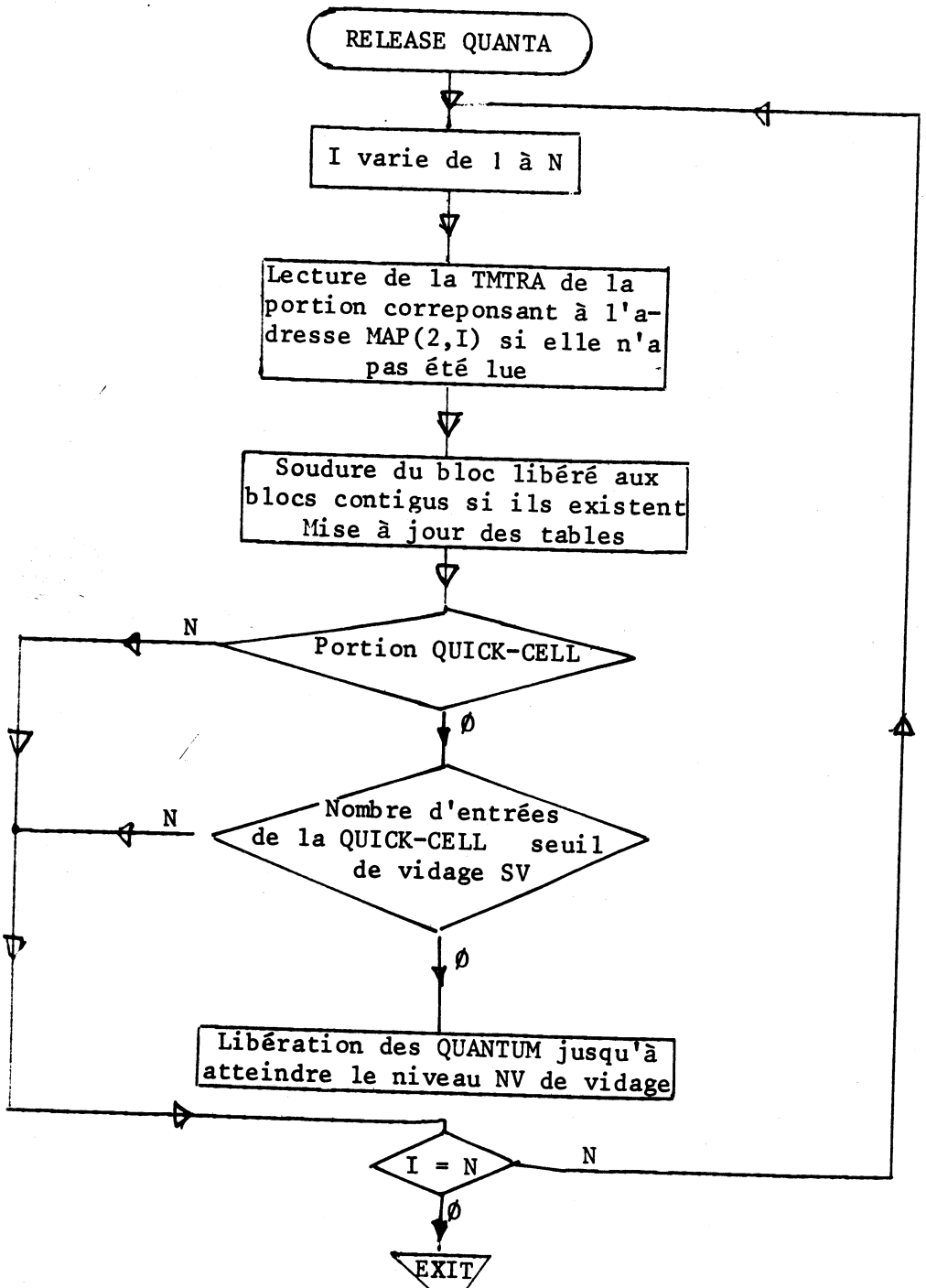


ANNEXE 2

ALGORITHME DE LA PROCEDURE "RELEASE QUANTA"

Les adresses secteurs et les numéros de disques des quantum à libérer sont dans un tableau.

MAP(1,3,1:N) : on libère N ZONES
- MAP(1,M) : Numéro de disque
- MAP(2,M) : Adresse secteur début
- MAP(3,M) : Adresse secteur fin



ANNEXE 3

MODELE DE SIMULATION, CAS DE L'ALGORITHME DE LA MESURE

ANNEXE 3

```
BEGIN
$ DECLARATIONS GENERALES. $
   INTEGER H,I,J,K,M,P,S,
           NBPAR,          $ NOMBRE DE PARTITIONS          $
           NBPOR,          $ NOMBRE DE PORTIONS            $
           IPP,            $ INDICE DE 1 ERE PORT. DE LA PART. $
           IDP,            $ INDICE DE LA DERNIERE         $
           DEMANDE; $ GARDE LA DEM. INITIALE (ALLOC. PART. ) $
$LA DEMANDE . $
   INTEGER ARRAY AD1,AD2(1:12); $ ADRESSES DES ZONES RENDUES (RELEASE) $
   INTEGER DATE,           $ DATE DE LA DEMANDE            $
           TYPORD,         $ TYPE DE LA DEMANDE            $
           IPOR,           $ INDEX DE LA PORTION(RELEASE)  $
           NPART,          $ NB DE ZONES RENDUES           $
           PART,           $ PARTITION(SORRES GET)         $
           DEM,            $ QUANTA DEMANDES               $
           ADR,            $ ADRESSE DU QUANTUM            $
           MORMAX,         $ MORCELLEMENT MAXIMUM          $
           MREL;           $ MORCELLEMENT                  $
$ LA QUICK-CELL $
   INTEGER INDCELL,        $ INDEX PORTION  QUICK-CELL     $
           KIKTAILLE,      $ TAILLE DE LA Q.C.             $
           KIKNOMBRE,      $ NOMBRE D ENTREES DE LA Q.C.   $
           KIKDISP,        $ PREMIERE ENTREE DISPONIBLE    $
           KIKDISPF,       $ DERNIERE ENTREE DISPONIBLE    $
           SREMP,          $ SEUIL DE REMPLISSAGE SR       $
           SREMPSUP,       $ SEUIL DE REMPLISSAGE SUP. <- $ NIVEAU NR
           SLIB,           $ SEUIL DE LIBERATION  SV       $
           SLIBINF,        $ SEUIL DE LIBERATION  INF. <- $ NIVEAU NV
           KIKINF,         $ ESPACE  INF. PORTION  Q.C.    $
           KIKSUP,         $ ESPACE  SUP. PORTION  Q.C.    $
           NRQC;
!*****************************************************************************!

   $ PREMIERES DONNEES. $
    NBPAR:=ININT;  NBPOR:=ININT;
    KIKTAILLE:=79;
    INDCELL:=2;
    KIKDISP:=0;
    KIKDISPF := 0;
    KIKNOMBRE := 0;
    SLIB :=60; SLIBINF:=20;
    SREMP:=30; SREMPSUP:=60;
    KIKINF:=99; KIKSUP:=198;

!*****************************************************************************!

$IMSET      BEGIN
      INTEGER ARRAY IPPOR,IDPOR(1:NBPAR); $ 1 ERE ET DERNIERE PORT. DE LA PAR.
      REF(PORTION) ARRAY POR(1:NBPOR); $ NOMS DES PORTIONS $
      INTEGER ARRAY TABUMAX(1:NBPOR);
      INTEGER ARRAY ADKIK(0:KIKTAILLE); $ LA QUICK CELL    $
      REF(READ) RESERVE ;
```

```
   LINK CLASS TROUP ;
      BEGIN
         INTEGER N,AD;REF(TROUS)X;X:-NEW TROUS(THIS TROUP);
      END;
      LINK CLASS TROUS(Y); REF(TROUP)Y;NULL;
   REF(TROUP) Y,Z;
   REF(TROUS) W ;
   REF(PORTION) PR;

*************************************************************************$

 $ LE SYSTEME DISQUE $
   CLASS PORTION  ;
   BEGIN
      REF(HEAD) DISQUE ;
      REF(HEAD) ARRAY L(1:64) ;

      INTEGER PLACE;       $ PLACE LIBRE DE LA PORTION    $
              NOMAX;       $  TAILLE DE LA PORTION        $

   INTEGER PROCEDURE TROMAX ;
   $ CETTE PROCEDURE DETERMINE LE RANG DE LA PLUS GRANDE LISTE L(I) NON VIDE$
      BEGIN
         INTEGER I;
         IF DISQUE.EMPTY THEN TROMAX:=0
         ELSE BEGIN I:=64;
              WHILE L(I).EMPTY DO I:=I-1;  TROMAX:=I;
              END;
      END; $ PROCEDURE TROMAX $

      PROCEDURE OCCUPATION(T,D) ;REF(TROUP) T;INTEGER D ;
      BEGIN
         REF(TROUS) W;   REF(HEAD) LL;
   OUTTEXT(' GET'); OUTINT(T.AD,5); OUTTEXT(' LONG'); OUTINT(D,4);
         OUTIMAGE;
         PLACE:=PLACE-D;
         INSPECT T DO BEGIN
         X.OUT;
         IF N=D THEN BEGIN  INTO(RESERVE);  EXIT OCCUPATION END;
         N:=N-D; AD:=AD+D ;
         LL:-IF N>63 THEN L(64) ELSE L(N);
         IF LL.EMPTY OR LL.LAST QUA TROUS.Y.AD<AD THEN X.INTO(LL)
         ELSE BEGIN W:-LL.FIRST; WHILE W.Y.AD<AD DO W:-W.SUC;
              X.PRECEDE(W);
              END;
         END;
      END $PROCEDURE OCCUPATION $;

      PROCEDURE EDIT ;
      BEGIN
      REF(TROUP) Y;
      IF NOT DISQUE.EMPTY THEN
         FOR Y:=DISQUE.FIRST,Y.SUC WHILE Y=/=NONE DO
         BEGIN OUTINT(Y.AD,4); OUTINT(Y.N,5);OUTIMAGE END;
      END ; $ EDIT $
```

```
PROCEDURE RESTITUTION(ADD,M); INTEGER ADD,M;
   BEGIN
      REF(TROUP)Y,TAP,TAV ; REF(TROUS) W;
      REF(HEAD) LL;
      PLACE:=PLACE+M;
      Y:- DISQUE.FIRST ;
      IF Y==NONE $ DISQUE EST PLEIN $ THEN
      BEGIN
         Y:- IF RESERVE .EMPTY THEN NEW TROUP ELSE RESERVE.FIRST ;
                  Y.N:=M ;Y.AD:=ADD ;
         Y.INTO(DISQUE);
         Y.X.INTO(L(IF M>63 THEN 64 ELSE M));
         EXIT RESTITUTION ;
      END ;
      RECHERCHE DU TROU QUI SUIT LA ADRESSE ADD.
      FOR Y:-Y,Y.SUC WHILE Y=/= NONE DO IF Y.AD>ADD THEN
      BEGIN TAP:-Y; TAV:-Y.PRED; GOTO SOUDAV END;
      TAV:-DISQUE.LAST;
      IF TAV.AD+TAV.N=ADD THEN
      BEGIN TAV.N:=TAV.N+M;TAV.X.OUT; Y:-TAV; END ELSE
      BEGIN   Y:- IF RESERVE.EMPTY THEN NEW TROUP ELSE RESERVE.FIRST;
      Y.N:=M ;Y.AD := ADD ;
      Y.FOLLOW(TAV)  END;
      GOTO RANGE ;
      SOUDAV: INSPECT TAV DO IF AD+N=ADD THEN
               BEGIN X.OUT; M:=M+N;  ADD:=AD; INTO(RESERVE) END;
      IF TAP.AD=ADD+M THEN BEGIN  TAP.AD:=ADD; TAP.N:= TAP.N+M;
      TAP.X.OUT;  Y:-TAP  END  ELSE
      BEGIN   Y:- IF RESERVE.EMPTY THEN NEW TROUP ELSE RESERVE.FIRST;
      Y.N:=M ;Y.AD := ADD ;
      Y.PRECEDE(TAP) END;
      RANGE:    $ RANGEMENT DE Y.X DANS L(Y.N) PAR BALAYAGE . $
      ADD:=Y.AD ;
      LL:-L(IF Y.N>63 THEN 64 ELSE Y.N);
      IF NOT LL.EMPTY THEN
      FOR W:- LL.FIRST,W.SUC WHILE W=/= NONE DO
            IF W.Y.AD>ADD THEN
            BEGIN
               Y.X.PRECEDE(W);
               GOTO FIN  END ;
      Y.X.INTO(LL);
   FIN : NULL ;
   END $PROCEDURE RESTITUTION $ ;
      DISQUE:- NEW HEAD ;
      FOR  H:=1 STEP 1 UNTIL 64 DO  L(H):- NEW HEAD ;
   END  $ FIN DE LA CLASSE PORTION $ ;

*****************************************************************************

   INITIALISATIONS
SWITCH DEMA:= GETUM,GETA,RELUM,RELA;
   RESERVE:- NEW HEAD;
* PORTIONS DE CHAQUE PARTITION $
  FOR I:= 1 STEP 1 UNTIL NBPAR DO BEGIN IPPOR(I):= ININT;
                                        IDPOR(I):= ININT;
                                END;
```

```
$ REPRESENTATIONS DES PORTIONS $
 FOR I := 1 STEP 1 UNTIL NBPOR DO
 BEGIN POR(I):= NEW PORTION;
     H:=ININT; $ NB DE '' TROUS'' $
       FOR J:=1 STEP 1 UNTIL H DO BEGIN Y:= NEW TROUP;
                                        Y.AD:=ININT; Y.N:=M:=ININT;
                                        Y.INTO(POR(I).DISQUE) ;
        Y.X.INTO(IF M>63 THEN POR(I).L(64) ELSE POR(I).L(M));
                                    END;
       POR(I).PLACE:=ININT; POR(I).NGMAX:=ININT;
       POR(I).EDIT;
 END;

$**********************************************************************$

$ -A- GENERATION D'UNE DEMANDE ET SA PROGRAMMATION DANS LE TEMPS       $
  BOUCLE:  DATE:=ININT;
OUTTEXT('TEMPS');
OUTINT(DATE,5);
  TYPORD := ININT-5;
  GOTO DEMA(TYPORD);

$**********************************************************************$

GETUM   : $ DONNEES DE LA DEMANDE D'UN ORDRE GET QUANTUM               $
          PART:=ININT-24;     ADR:=ININT;
          IF KIKNOMBRE =0 THEN BEGIN DEM := 1;
$ CHANGEMENT DE PORTION QUICK CELL $
          GOTO VOIRA END;
          KIKDISP := KIKDISP +1 ; KIKNOMBRE := KIKNOMBRE -1 ;
          IF KIKDISP>KIKTAILLE THEN KIKDISP:=0;
           GOTO FINDEM;

$**********************************************************************$

RELUM   : $ DONNEES DE LA DEMANDE D'UN ORDRE RELEASE QUANTUM           $
          IPOR:=ININT-24;  ADR:=ININT;
          POR(IPOR).RESTITUTION(ADR,1);
          TROUMAX(INDCELL):= POR(INDCELL).TROMAX;
          GOTO FINDEM;

RELA:$ DONNEES DE LA DEMANDE D'UN ORDRE RELEASE QUANTA                 $
          IPOR:=ININT-24;
          PR:=POR(IPOR);
          NPART:= ININT;
          FOR I := 1 STEP 1 UNTIL NPART DO
            BEGIN AD1 (I) := ININT ;
                                    AD2 (I) := ININT ; END ;
     FOR I:=1 STEP 1 UNTIL NPART DO
     BEGIN   PR.RESTITUTION(AD1(I),AD2(I)-AD1(I)+1) ;      END;
    TROUMAX(J):= POR(J).TROMAX;
          GOTO FINDEM;

$**********************************************************************$

REMPOC : J := 0 ; N := SREMPSUP - KIKNOMBRE;
```

```
    NREC:=NREC+1;
    PR:=POR(INDCELL);
    N:= IMIN(N,PR.PLACE);
    KIKNOMBRE := KIKNOMBRE +N ;
    I:=1;
    WHILE N>0 DO IF PR.L(I).EMPTY THEN I:=I+1 ELSE BEGIN
    IF I=64 THEN  BEGIN J:= PR.NOMAX+1;
    FOR Y:=PR.L(64).FIRST,Y.SUC WHILE Y<>NONE DO
    IF U.Y>N<J THEN BEGIN J:=U.Y.N;  Y:=W.Y END END
    ELSE Y:=PR.L(I).FIRST QUA TROUS.Y;
    U:=IMIN(N,Y.N);
    FOR U:=1 STEP 1 UNTIL J DO BEGIN  ADKIK(KIKDISPF) := Y.AD+M-1;
    KIKDISPF := KIKDISPF+1 ;
    IF KIKDISPF > KIKTAILLE THEN KIKDISPF:= 0 ;
    END;
    N:=N-J;
    PR.OCCUPATION(Y,J);
    END;
    TROUMAX(INDCELL):= POR(INDCELL).TROMAX;
  PR.EDIT;
  GOTO FINDEM;

*****************************************************************************

GETA  : { DONNEES DE LA DEMANDE D'UN ORDRE GET QUANTA
        PART:=ININT-24;
        DEM:=ININT;
        IPP:=IPPOR(PART);  IDP:=IDPOR(PART);
VOMGA : MORMAX := 12 ;
        NREL:= MORMAX;

 * SELECTION DE LA PORTION *

GA :
    IF DEM<64 THEN
    BEGIN
    K:=0;
       * PETITE DEMANDE . POSSIBILITE D'ALLOCATION UNIQUE .
      FOR I:=IPP STEP 1 UNTIL IDP DO  BEGIN
          PR:=POR(I);
          IF PR.PLACE>= DEM THEN BEGIN
             IF NOT  PR.L(DEM).EMPTY THEN
             BEGIN  K:=I;GOTO ALLOC ;END;
             IF TROUMAX(I)>DEM THEN K:=I ;
          END END;
       IF K<>0 THEN   GOTO ALLOC    ;
    END;
       { DEMANDE DE PLUS DE 63 O. ,OU ALLOCATION UNIQUE IMPOSSIBLE; ESSAI
      POUR VOIR SI PLACE SUFFISANTE SUR LES PORTIONS DE LA PARTITION *
       S:=0 ; J:=IPP ;
       FOR I:=IPP STEP 1 UNTIL IDP DO    BEGIN
          S:=S+POR(I).PLACE;
       IF POR(J).PLACE<POR(I).PLACE THEN J:=I;
    END;
    * ICI S EST LA PLACE TOTALE SUR LES PORTIONS DE LA PARTITION ,J INDIQUE
      CELUI QUI A LE PLUS DE PLACE.
```

```
      IF S< DEM  THEN  BEGIN OUTTEXT('DEMANDE REFUSEE!'); OUTIMAGE;
                                  GOTO FINDEM END;
         K:=J ;
         IF POR(J).PLACE < DEM THEN  GOTO ALDD;

$*************************************************************************

   ALLOC:
   % ALLOCATION EFFECTIVE A PARTIR DE TABLES SUR DISQUE,L'UNITE  SELECTIONNEE
    EST POR(K)    NOUS ECRIVONS ICI L'ALGORITHME UTILISE EFFECTIVEMENT;
   PR:-POR(K); DEMANDE:= DEM;
   % POSSIBILITE D'ALLOCATION UNIQUE.%
ALF :  IF DEM<64 THEN
   BEGIN
      FOR I:= DEM STEP 1 UNTIL 63 DO
      IF NOT PR.L(I).EMPTY THEN    BEGIN
         Z:- PR.L(I).FIRST QUA TROUS.Y ;
         PR.OCCUPATION(Z,DEM) ;
         GOTO FALLOC ;
      END;
      IF NOT PR.L(64).EMPTY THEN   BEGIN
         I:=POR(K).NQMAX+1;
         FOR W:-PR.L(64).FIRST,W.SUC WHILE W=/=NONE DO
         IF W.Y.N<I THEN
               BEGIN
                     Z:-W.Y ;I:=Z.N; END ;
         PR.OCCUPATION(Z,DEM) ;
         GOTO FALLOC ;
      END ;
      FOR I:=DEM-1 STEP -1 UNTIL 1 DO
      IF NOT PR.L(I).EMPTY THEN
         BEGIN Y:- PR.L(I).FIRST QUA TROUS.Y; GOTO ALA END;
   END;
      IF NOT PR.L(64).EMPTY THEN
      BEGIN
         I:=0 ; J:=POR(K).NQMAX+1;
         FOR W:-PR.L(64).FIRST,W.SUC WHILE W=/= NONE DO
         BEGIN
               P:=W.Y.N ;
               IF P>I THEN BEGIN Y:-W.Y;I:=P END;
               IF P >= DEM  AND P<J THEN
                                 BEGIN Z:-W.Y;J:=P      END;
         END;
         IF J<=PR.NQMAX THEN BEGIN PR.OCCUPATION(Z,DEM) ; GOTO FALLOC END ;
         GOTO ALA ;
      END;
      FOR I:=63 STEP -1 UNTIL 1 DO
      IF NOT PR.L(I).EMPTY THEN
      BEGIN Y:-PR.L(I).FIRST QUA TROUS.Y; GOTO ALA END;
   ALA :%ALLOCATION UNIQUE IMPOSSIBLE. Y POINTE SUR LE TROU LE PLUS GRAND
      % INFERIEUR A DEMANDE. ALLOCATION DE Y.                    %
   DEM:= DEM-Y.N;
   AD1(NORMAX):= Y.AD;
   AD2(NORMAX):= Y.N ;
   PR.OCCUPATION(Y,Y.N);
   NORMAX:=NORMAX-1 ;
```

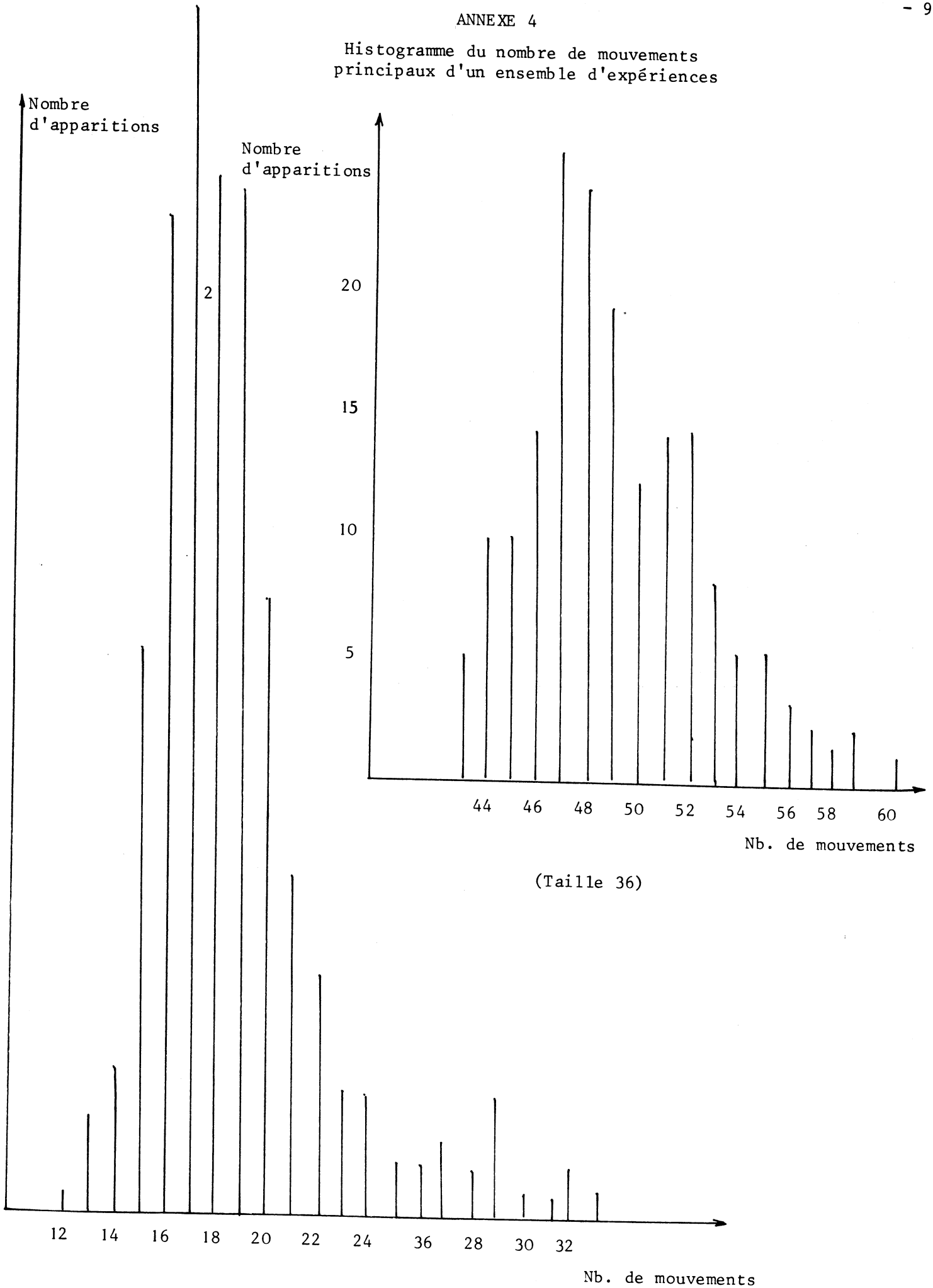
```
    IF MORMAX/=0 THEN GOTO ALF;
$ RESTITUTION DE L'ALLOCATION $
    FOR I:=1 STEP 1 UNTIL MREL DO PR.RESTITUTION(AD1(I),AD2(I));
        GOTO FALLOC;
  ALDD: OUTTEXT('PLUS.PORTIONS'); OUTIMAGE;
  FALLOC: $ FIN DE L' ALLOCATION EFFECTIVE$ NULL;
    TROUMAX(K):= POR(K).TROMAX;
    TQC : IF K= INDCELL AND  KIKNOMBRE < SREMP THEN
    GOTO REMPQC ;
FINDEM: NULL;
  IF DATE/= 12764 THEN GOTO BOUCLE;

$*************************************************************************

    END $SIM$ ;
END#
```

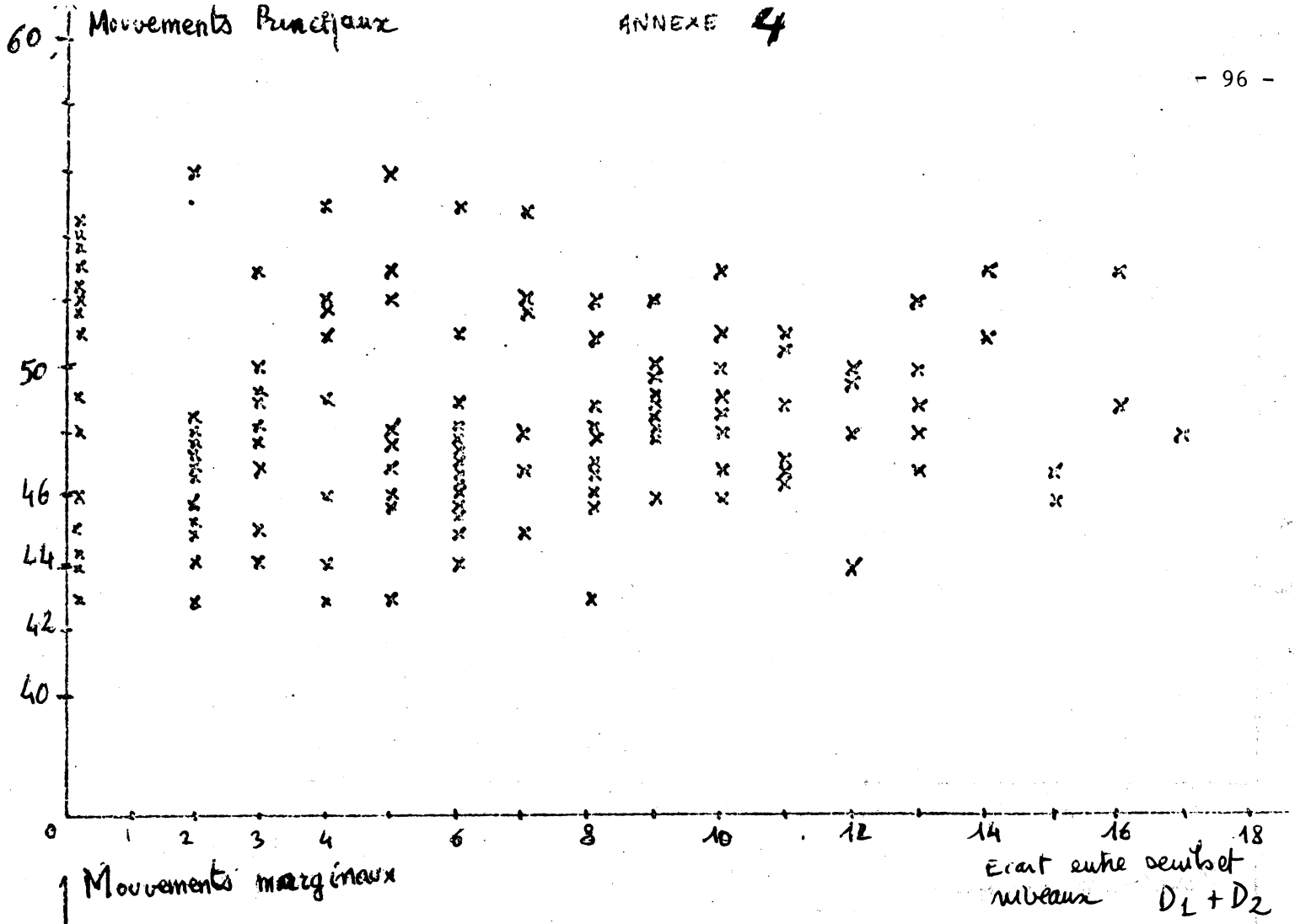
ANNEXE 4

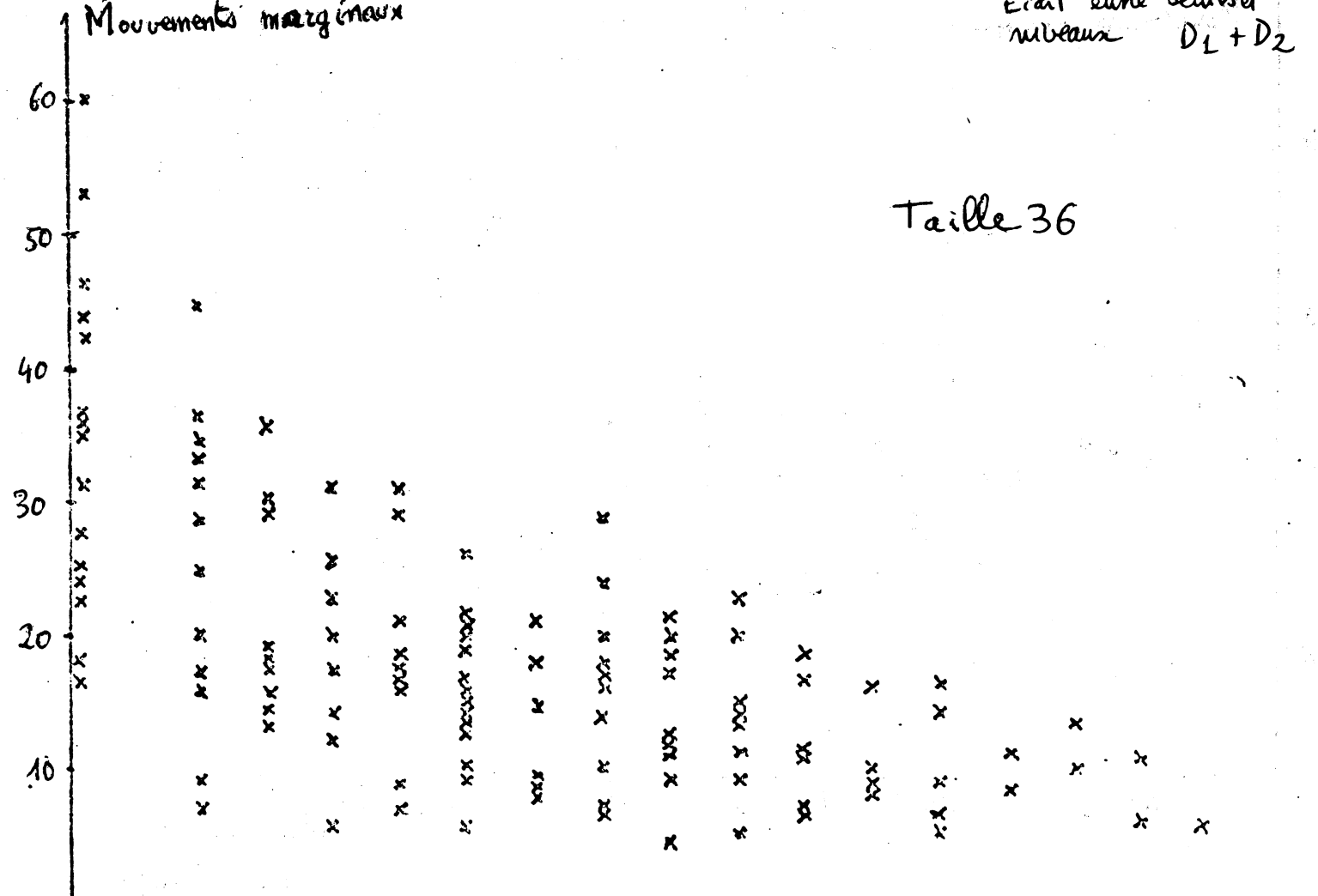
EXPERIENCES DE SIMULATION : ETUDE DES SEUILS ET NIVEAUX

ANNEXE 4

Histogramme du nombre de mouvements principaux d'un ensemble d'expériences

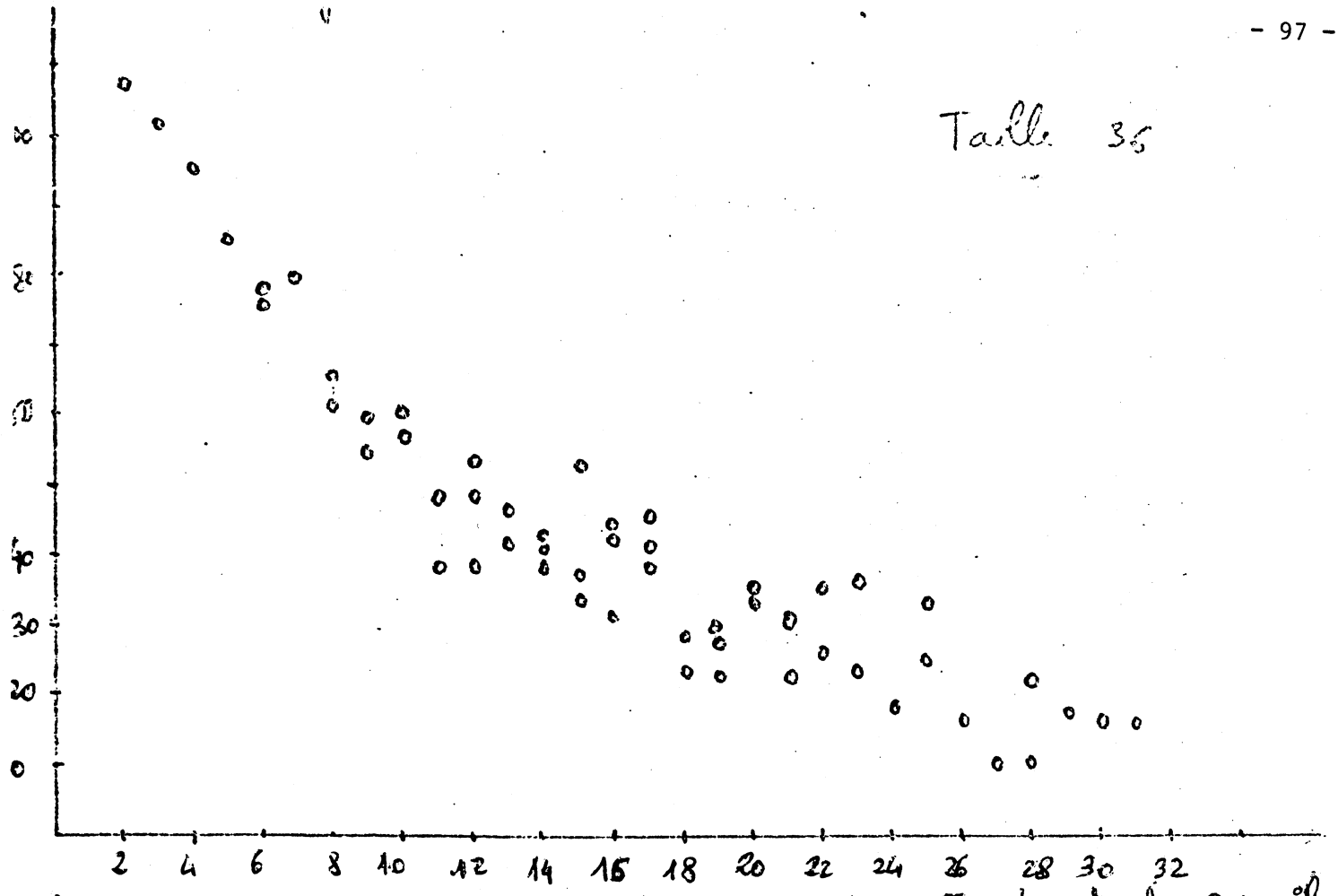



Mouvements Principaux
60
50
46
44
42
40
0
1
2
3
4
6
8
10
12
14
16
18
Ecart entre deux niveaux $D_1 + D_2$

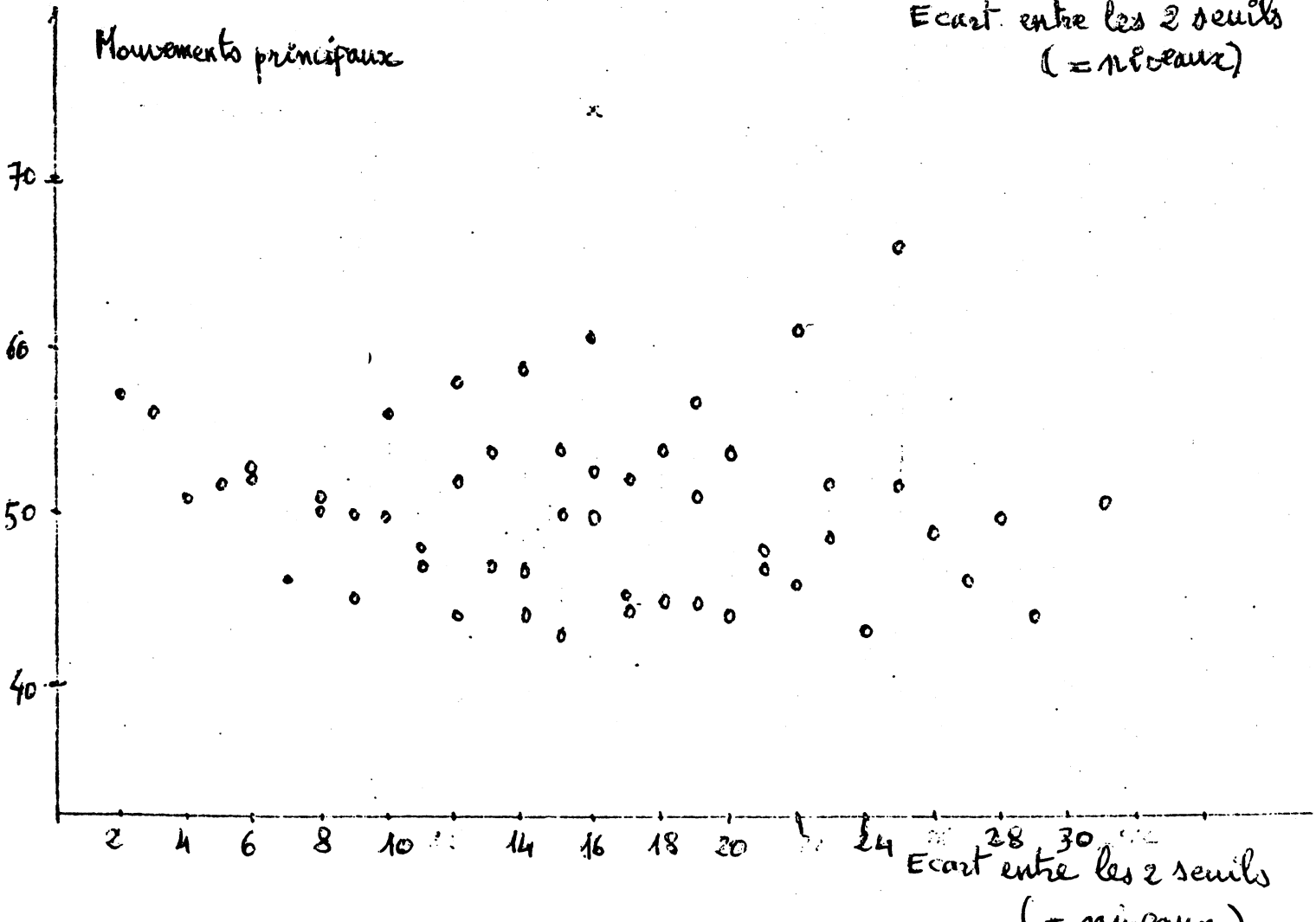

Mouvements marginaux
60
50
40
30
20
10

Taille 36

Taille 36



Mouvements principaux



ANNEXE 5

EXPERIENCES DE SIMULATION : ETUDE DE LA TAILLE

ANNEXE 5

EXPERIENCES DE SIMULATION QUICK-CELLS DE DIVERSES TAILLES

TAILLE		2	3	5	6	10	15	20		25	30	
SEUILS	SV	1	1	3	4	8	12	16	18	20	25	24
ET	NR	1	1	3	4	7	9	12	14	15	20	18
NIVEAUX	NV	1	1	1	1	3	6	8	6	10	10	12
	SR	1	1	1	1	**2**	3	4	2	8	5	6
MVTS	V	572	343	206	167	118	87	63	56	45	35	36
PRINCIPAUX	R	564	526	261	203	103	74	54	54	42	30	31
	TOTAL	1136	869	467	370	221	161	117	110	87	65	67
MVTS	R	87	79	37	23	30	24	20	6	12	10	12
MARGINAUX	V	56	71	33	21	7	4	6	2	5	2	4
	TOTAL	143	150	70	44	37	28	26	8	17	12	16

TAILLE		32	48		58	70	79	83		88		
SEUILS	SV	33	40	43	43	48	55	70	74	70	70	75	80
ET	NR	28	30	38	38	38	45	60	64	60	60	65	70
NIVEAUX	NV	10	15	10	15	15	15	15	15	15	15	20	15
	SR	5	10	5	10	10	10	10	10	10	10	15	10
MVTS	V	25	17	18	21	15	11	8	8	7	7	5	7
PRINCIPAUX	R	18	16	15	14	11	11	7	6	7	8	5	5
	TOTAL	43	33	33	35	25	22	15	14	14	15	10	12
MVTS	R	12	12	5	11	10	8	3	6	4	5	6	5
MARGINAUX	V	3	5	1	5	1	3	0	2	2	3	4	1
	TOTAL	15	17	6	16	11	11	3	8	6	8	10	6

ANNEXE 5

EXPERIENCES DE SIMULATION QUICK-CELLS DE DIVERSES TAILLES (suite)

TAILLE		98			100	120	160
SEUILS	SV	70	85	90	75	75	130
	NR	20	75	80	65	65	120
NIVEAUX	NV	20	20	15	20	20	40
	SR	15	15	10	15	15	30
MVTS	V	6	7	8	6	4	3
PRINCIPAUX	R	7	5	8	5	7	2
	TOTAL	13	12	16	11	11	5
MVTS	R	3	4	4	8	8	5
MARGINAUX	V	2	0	2	4	6	4
	TOTAL	5	4	6	12	14	9

ANNEXE 5

EXPERIENCES DE SIMULATION QUICK-CELLS SIMPLIFIEES

A) SANS SEUILS, NI NIVEAUX : (les GET QUANTA n'interviennent pas)

TAILLE	7	10	16	20	25	36	50	60	70	78	100
MVTS PRINCIPAUX	410	280	230	154	170	130	90	25	26	27	20
MVTS MARGINAUX	0	0	0	0	0	0	0	0	0	0	0

B) SANS SEUILS : (les GET QUANTA n'interviennent pas)

TAILLE	3	6	10	20	30	40	50	60	80	100	120	160
MVTS PRINCIPAUX	774	359	231	109	71	43	31	21	17	7	7	3
MVTS MARGINAUX	0	0	0	0	0	0	0	0	0	0	0	0
MVTS VIDAGE	1	1	2	3	5	7	9	10	13	17	20	27
MVTS REMPLISSAGE	2	5	3	16	25	33	41	50	67	83	100	133

B I B L I O G R A P H I E

BIBLIOGRAPHIE

A.1 - B. ARDEN, BOETTNER

Measvrements and performances of a multiprogramming system. Second Symposium on operating systems principles. Oct 1969. Princeton Univers.

B.1 - P. BERARD, C. BOKSENBAUM, J.P. MACHEFAUX

Analyse d'une gestion d'espace sur disque. Journées sur les modèles et mesures de performances, Juin 1973, Toulouse.

B.2 - P. BERARD, C. BOKSENBAUM, J.P. MACHEFAUX

Allocation d'espace disques, mécanisme de la QUICK-CELL. Notes CII, Février 1973.

B.3 - P. BERARD, J.P. MACHEFAUX

Analyse des mesures fournies par le Centre CII. Les Clayes, carnet de notes CII 1972.

B.4 - C. BOKSENBAUM

Problèmes de la simulation des systèmes. Séminaires de l'IMAG, Mai 1971.

C.1. - J. COUTTAZ

Simulation d'un système conversationnel. Thèse de 3ème cycle, Juin 1970, Grenoble.

C.2 - D.R. COX, P.A.W. LEWIS

The statistical analysis of series of events, Methnen and Co. 1968.

D.1 - R.C. DALEY, P.G. NEUMANN

A general purpose file system for secondary storage. Fall Joint Computer conf. 1965.

G.1 - P. LE GALL

Les systèmes avec ou sans attente et les processus stochastiques. Dunod, Paris 1962.

H.1 - *HASP Control Program Library 360 D Ø S1.014.*

I.1 - *IBM Program Logic, system 360 : Direct access device space management Form 428-6607-4.*

I.2 - *IBM Pogram Logic, CP-67 Manual Form GY. 20.0590.*

I.3 - *IBM Program Logic, CMS Manual Form GY.20.0591.*

I.4 - *APL 360 User's manual, AD. Falkoff. K.E. Iverson.*

K.1 - K. KNOWLTON

A programmer's description of L^6. Com. ACM, Août 1966.

K.2 - D.E. KNUTH

The art of Computer programming. Vol. 1, pp. 435-451, Fundamental algorithms 1969 Addison-Wesley.

L.1 - LE HEIGET

Généralisation de la notion d'espaces virtuels sous les systèmes CP67-CMS. Thèse CNAM, Grenoble 1972.

M.1 - B.H. MARGOLIN, R.P. PARMELEE, M. SCHATZOFF

Analysis of free storage algorithms. Statistical Computer Performance Evaluation edited by W. Freiberger. Academic Press, 1972.

N.1 - N.R. NIELSEN

The simulation of time-sharing systems. Com. ACM, n° 7, Juillet 1967.

N.2 - T. NAYLOR

Computer Simulation techniques. Wiley and Sons, New-York 1966.

P.1 - N.V. PRABHU

Quenes and Inventories. Wiley and Sons, New York 1965.

R.1 - D.T. ROSS

The A.E.D. free storage package. Com. ACM, n° 10, 1967.

S.1 - SIRIS 8

Spécifications de réalisation. Notes CII.

S.2 - SIRIS 7

Gestion de fichiers et processeurs associés, brochure CII.
Langage de Commande, brochure CII.
Symbol et Metasymbol : manuel d'utilisation, brochure CII.
Spécification de réalisation, notes CII.

S.3 - SIMULA 67

CII 10070 Simula language reference manual 1971. S.D. Ichbiah

S.4 - SIMULA 67

Simula par l'exemple. C. Scherer 5224 T/FR, Déc. 1972