Quelques problèmes de restructuration dans un environnement paginé

Bernard Lacolle

THESE

présentée à

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

POUR OBTENIR LE GRADE DE
Docteur de 3ème cycle
en MATHEMATIQUES APPLIQUEES
option : ANALYSE NUMERIQUE

Bernard LACOLLE

QUELQUES PROBLEMES DE RESTRUCTURATION DANS UN ENVIRONNEMENT PAGINE

Soutenue le 14 septembre 1976 devant la commission d'examen :

Président : Monsieur N. GASTINEL

Examinateurs : Monsieur P. JORRAND
Monsieur F. ROBERT
Monsieur M. SAKAROVITCH

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

Monsieur Gabriel CAU : Président
Monsieur Pierre JULLIEN : Vice-Président

MEMBRES DU CORPS ENSEIGNANT DE L'U.S.M.G.

PROFESSEURS TITULAIRES

MM.	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
	BEAUDOING André	Clinique de Pédiatrie et Puériculture
	BERNARD Alain	Mathématiques Pures
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques Pures
	BEZES Henri	Pathologie chirurgicale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Clinique gastro-entérologique
Mme	BONNIER Marie-Jeanne	Chimie générale
MM.	BOUCHERLE André	Chimie et toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques Appliquées
	BOUTET DE MONTVEL Louis	Mathématiques Pures
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie végétale
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Clinique Oto-rhino-laryngologique
	CHATEAU Robert	Clinique de neurologie
	CHIBON Pierre	Biologie animale
	COEUR André	Pharmacie chimique et chimie analytique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie pathologique
Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumophtisiologie

MM.	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée (IUT A)
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de dermatologie et syphiligraphie
	GAGNAIRE Didier	Chimie physique
	GALLISSOT François	Mathématiques Pures
	GALVANI Octave	Mathématiques Pures
	GASTINEL Noël	Analyse numérique
	GAVEND Michel	Pharmacologie
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques Pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	KAHANE André	Physique générale
	KLEIN Joseph	Mathématiques Pures
	KOSZUL Jean-Louis	Mathématiques Pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques Appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
Mme	LAJZEROWICZ Janine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre-Jean	Mathématiques Appliquées
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LOISEAUX Pierre	Sciences nucléaires
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LOUP Jean	Géographie
Mlle	LUTZ Elisabeth	Mathématiques Pures
MM.	MALGRANGE Bernard	Mathématiques Pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Clinique cardiologique
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MICOUD Max	Clinique maladies infectieuses
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	MULLER Jean-Michel	Thérapeutique (Néphrologie)
	NEEL Louis	Physique du Solide
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques Pures
	PEBAY-PEYROULA Jean-Claude	Physique
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REVOL Michel	Urologie
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-chirurgie
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie

MM. SIBILLE Robert	Construction mécanique (IUT A)
SOUTIF Michel	Physique générale
TANCHE Maurice	Physiologie
TRAYNARD Philippe	Chimie générale
VAILLANT François	Zoologie
VALENTIN Jacques	Physique nucléaire
VAUQUOIS Bernard	Calcul électronique
Mme VERAIN Alice	Pharmacie galénique
MM. VERAIN André	Physique
VEYRET Paul	Géographie
VIGNAIS Pierre	Biochimie médicale
YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM. CLARK Gilbert	Spectrométrie physique
CRABBE Pierre	CERMO
ENGLMAN Robert	Spectrométrie physique
HOLTZBERG Frédéric	Basses températures
DEMBICKI Eugéniuz	Mécanique
MATSUSHIMA Yozo	Mathématiques Pures

PROFESSEURS SANS CHAIRE

Mlle AGNIUS-DELORD Claudine	Physique pharmaceutique
ALARY Josette	Chimie analytique
MM. AMBROISE-THOMAS Pierre	Parasitologie
BELORIZKY Elie	Physique
BENZAKEN Claude	Mathématiques Appliquées
BIAREZ Jean-Pierre	Mécanique
BILLET Jean	Géographie
BOUCHET Yves	Anatomie
BRUGEL Lucien	Energétique (IUT A)
BUISSON René	Physique (IUT A)
BUTEL Jean	Orthopédie
COHEN ADDAD Pierre	Spectrométrie physique
COLOMB Maurice	Biochimie
CONTE René	Physique (IUT A)
DEPASSEL Roger	Mécanique des fluides
FONTAINE Jean-Marc	Mathématiques Pures
GAUTHIER Yves	Sciences Biologiques
GAUTRON René	Chimie
GIDON Paul	Géologie et Minéralogie
GLENAT René	Chimie organique
GROULADE Joseph	Biochimie médicale
HACQUES Gérard	Calcul numérique
HOLLARD Daniel	Hématologie
HUGONOT Robert	Hygiène et Médecine préventive
IDELMAN Simon	Physiologie animale
JOLY Jean-René	Mathématiques Pures
JULLIEN Pierre	Mathématiques Appliquées
Mme KAHANE Josette	Physique
MM. KRAKOWIAK Sacha	Mathématiques Appliquées
KUHN Gérard	Physique (IUT A)
LE ROY Philippe	Mécanique (IUT A)
LUU DUC Cuong	Chimie organique

MM.	MAYNARD Roger	Physique du solide
Mme	MINIER Colette	Physique (IUT A)
MM.	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et Minéralogie
	PFISTER Jean-Claude	Physique du solide
Mlle	PIERY Yvette	Physiologie animale
MM.	RAYNAUD Hervé	M.I.A.G.
	REBECQ Jacques	Biologie (CUS)
	REYMOND Jean-Charles	Chirurgie générale
	RICHARD Lucien	Biologie végétale
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROBERT André	Chimie papetière
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
MM.	STREGLITZ Paul	Anesthésiologie
	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques Appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	AMBLARD Pierre	Dermatologie
	ARMAND Gilbert	Géographie
	ARMAND Yves	Chimie (IUT A)
	BACHELOT Yvan	Endocrinologie
	BARGE Michel	Neuro-chirurgie
	BARJOLLE Michel	M.I.A.G.
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamie
MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BRODEAU François	Mathématiques (IUT B)
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie papetière
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
	CORDONNIER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	CYROT Michel	Physique du solide
	DELOBEL Claude	M.I.A.G.
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie végétale
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	FAURE Gilbert	Urologie
	GAUTIER Robert	Chirurgie générale
	GENSAC Pierre	Botanique
	GIDON Maurice	Géologie
	GROS Yves	Physique (IUT A)

MM. GUITTON Jacques	Chimie
HICTER Pierre	Chimie
IVANES Marcel	Electricité
JALBERT Pierre	Histologie
JUNIEN-LAVILLAVROY Claude	O.R.L.
KOLODIE Lucien	Hématologie
LE NOC Pierre	Bactériologie-virologie
LEROY Philippe	IUT A
MACHE Régis	Physiologie végétale
MAGNIN Robert	Hygiène et médecine préventive
MALLION Jean-Michel	Médecine du travail
MARECHAL Jean	Mécanique (IUT A)
MARTIN-BOUYER Michel	Chimie (CUS)
MICHOULIER Jean	Physique (IUT A)
NEGRE Robert	Mécanique (IUT A)
NEMOZ Alain	Thermodynamique
NOUGARET Marcel	Automatique (IUT A)
PARAMELLE Bernard	Pneumologie
PECCOUD François	Analyse (IUT B)
PEFFEN René	Métallurgie (IUT A)
PERRET Jean	Neurologie
PERRIER Guy	Géophysique - Glaciologie
PHELIP Xavier	Rhumatologie
RACHAIL Michel	Médecine interne
RACINET Claude	Gynécologie et obstétrique
RAMBAUD André	Hygiène et hydrologie
RAMBAUD Pierre	Pédiatrie
Mme RENAUDET Jacqueline	Bactériologie
MM. ROBERT Jean-Bernard	Chimie Physique
ROMIER Guy	Mathématiques (IUT B)
SHOM Jean-Claude	Chimie générale
STOEBNER Pierre	Anatomie pathologique
VROUSOS Constantin	Radiologie

MAITRE DE CONFERENCES ASSOCIES

M. COLE Antony	Sciences nucléaires

Fait à SAINT MARTIN D'HERES, AVRIL 1976.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : M. Philippe TRAYNARD

Vice-Président : M. Pierre-Jean LAURENT

PROFESSEURS TITULAIRES

MM. BENOIT Jean	Radioélectricité
BESSON Jean	Electrochimie
BLOCH Daniel	Physique du solide
BONNETAIN Lucien	Chimie Minérale
BONNIER Etienne	Electrochimie et Electrométallurgie
BOUDOURIS Georges	Radioélectricité
BRISSONNEAU Pierre	Physique du solide
BUYLE-BODIN Maurice	Electronique
COUMES André	Radioélectricité
DURAND Francis	Métallurgie
FELICI Noël	Electrostatique
FOULARD Claude	Automatique
LESPINARD Georges	Mécanique
MOREAU René	Mécanique
PARIAUD Jean-Charles	Chimie-Physique
PAUTHENET René	Physique du solide
PERRET René	Servomécanismes
POLOUJADOFF Michel	Electrotechnique
SILBER Robert	Mécanique des Fluides

PROFESSEUR ASSOCIE

M. ROUXEL Roland	Automatique

PROFESSEURS SANS CHAIRE

MM. BLIMAN Samuel	Electronique
BOUVARD Maurice	Génie Mécanique
COHEN Joseph	Electrotechnique
LACOUME Jean-Louis	Géophysique
LANCIA Roland	Electronique
ROBERT François	Analyse numérique
VEILLON Gérard	Informatique Fondamentale et Appliquée
ZADWORNY François	Electronique

MAITRES DE CONFERENCES

MM. ANCEAU François	Mathématiques Appliquées
CHARTIER Germain	Electronique
GUYOT Pierre	Chimie Minérale
IVANES Marcel	Electrotechnique
JOUBERT Jean-Claude	Physique du solide
MORET Roger	Electrotechnique Nucléaire
PIERRARD Jean-Marie	Mécanique
SABONNADIERE Jean-Claude	Informatique Fondamentale et Appliquée
Mme SAUCIER Gabrièle	Informatique Fondamentale et Appliquée

MAITRE DE CONFERENCES ASSOCIE

M. LANDAU Ioan	Automatique

CHERCHEURS DU C.N.R.S. (Directeur et Maître de Recherche)

MM. FRUCHART Robert	Directeur de Recherche
ANSARA Ibrahim	Maître de Recherche
CARRE René	Maître de Recherche
DRIOLE Jean	Maître de Recherche
MATHIEU Jean-Claude	Maître de Recherche
MUNIER Jacques	Maître de Recherche

A ma Fille,

ma Femme,

et mes Parents.

Je tiens, tout d'abord, à remercier Monsieur le Professeur N. GASTINEL qui m'a enseigné l'Analyse Numérique et m'a communiqué sa passion pour l'étude de problèmes très appliqués.

Les longues heures qu'il m'a consacrées, ses nombreux conseils et encouragements m'ont permis de mener à bien ce travail.

Que soient aussi remerciés Messieurs P. JORRAND, F. ROBERT, et M. SAKAROVITCH d'avoir accepté de faire partie du jury.

Mes remerciements vont aussi à toute l'équipe d'Analyse Numérique et particulièrement à Monsieur J. DELLA DORA qui m'a si souvent encouragé.

Je remercie aussi tout le personnel du Centre Interuniversitaire de Calcul de Grenoble et en particulier les opérateurs et le service de perforation qui ont contribué chacun dans leur domaine à la réalisation de ce travail.

Enfin, je remercie Madame Cl. MEYRIEUX qui malgré un planning chargé a assuré avec compétence et patience la frappe de ce texte, ainsi que le service de reproduction pour la qualité de son travail.

TABLE DES MATIERES

INTRODUCTION

CHAPITRE I GESTION DE MEMOIRE PAGINEE ET EXPRESSIONS MATRICIELLES DE QUELQUES PROBLEMES DE RESTRUCTURATION

CHAPITRE II FORMES MATRICIELLES EQUIVALENTES DES PROBLEMES DE RESTRUCTURATION : PARTITIONNEMENT DE MATRICES ET PROBLEMES DE DISTANCES EUCLIDIENNES MAXIMALES

CHAPITRE III BIPARTITIONNEMENT DE MATRICES

CHAPITRE VI

BIBLIOGRAPHIE

INTRODUCTION

Les notions de mémoires virtuelles et d'environnement paginé sont aujourd'hui très répandues [1]. Comme toute nouvelle technique son utilisation a mis en évidence de nouveaux problèmes. En particulier, le "bon déroulement" d'un programme devient tributaire de certaines propriétés liées à la structure de ce programme [2]. Comme conséquence il est vite apparu que l'ordre dans lequel étaient chargés en mémoire les différentes parties (modules, données, ...) d'un programme pouvait être fort important, l'idée de base étant d'éviter une trop grande dispersion des parties de la mémoire référencées lors de l'exécution du porgramme : de là l'idée de restructuration.

Ce problème n'a cependant pas donné lieu à beaucoup de résultats et l'essentiel de ceux-ci figurent dans [3][4][5]. La raison de cette carence est certainement l'énorme complexité du problème surtout dans les contextes qu'ont voulu aborder la plupart des auteurs : grands systèmes, multiprogrammation... .

Notre première préoccupation a été de décrire un petit système à mémoire virtuelle en le réduisant à quelques principes fondamentaux : en particulier nous réservons la mémoire virtuelle à un ensemble de données. Notre idée est de nous limiter à des données numériques utilisées par des programmes simples. Après avoir défini une gestion de mémoire nous adoptons comme coût d'exécution de notre programme le nombre de demandes de pages. Nous pouvons alors mettre en évidence certaines relations entre l'organisation de données et ce coût. Un contexte matriciel a été choisi pour cette étude ; si ce contexte peut être limitatif en taille il a cependant pour avantage de permettre de poser assez clairement certains problèmes. En particulier l'importance d'un problème appelé "partitionnement de matrice" apparaît nettement.

Ce dernier problème, sous la nomenclature de la théorie des graphes, est d'ailleurs fort connu, surtout par sa difficulté. Nous avons délaissé délibérément toutes les heuristiques relatives à la résolution de tels problèmes. La formulation matricielle adoptée nous a poussé à rechercher

quelles particularités matricielles pouvaient faciliter la résolution. Certains cas probabilistes, réputés faciles et donnant des matrices de rang 1 ($A(i,j) = p_i p_j$) ont attiré notre attention sur l'importance du rang des matrices utilisées. Nous dégageons ainsi au chapitre II une notion de problèmes de distance euclidienne maximale ainsi que quelques principes généraux utilisés par la suite : principe de projection maximum, solution à la précision relative ε , approximation de matrices définies-positives par des matrices de rang donné .

Ces interprétations se révèlent très fructueuses dans certains cas exposés au chapitre III et qui concernent le bi-partitionnement d'une matrice de rang très faible. Deux types d'algorithmes sont décrits : le premier, un peu du type "branch and bound", pour une matrice de rang 2 et le second, basé sur une énumération, pour une solution à la précision relative ε . Une combinaison de ce type d'algorithmes pourrait être envisagée. A la base de ces deux types d'algorithmes se trouvent des problèmes de tris dont la rapidité assure le succès des méthodes précédentes. Une extension à des matrices quelconques est envisagée, utilisant l'approximation par des matrices de rang donné.

Dans le chapitre IV une extension à un partitionnement plus général est développée ; de nombreuses difficultés apparaissent : en particulier, le problème de tri, cité plus haut, devient un problème plus complexe de programmation linéaire en nombres entiers. Le problème reste simple seulement pour une matrice de rang 1. Nous pouvons cependant élaborer des méthodes dont les résultats sont fort acceptables.

Notons que le point de vue adopté nous a toujours permis de borner les solutions des problèmes traités.

Nous présentons dans le chapitre V quelques situations non classiques où pourrait intervenir la notion de mémoire virtuelle qui n'est certainement pas réservée aux grands calculateurs. Le schéma développé au chapitre I aurait des applications assez directes dans le domaine des micro-calculateurs et des mini-calculateurs si la technologie voulait tirer

profit d'une gestion paginée des mémoires fines (registres, caches,...). Quelques aspects pratiques de la restructuration sont alors illustrés par des exemples empruntés au calcul numérique.

On peut s'étonner du nombre relativement faible des données utilisées par nos algorithmes (environ 100).

A ce sujet remarquons simplement que puisque les problèmes posés avec une centaine de données apparaissent très difficiles il n'y a aucune raison pour qu'ils deviennent faciles lorsqu'il s'agit de 10 000 données ! Ceci pour dire que certaines restructurations qui concernent effectivement autant de données ne peuvent être abordées dans toutes leur généralité. Certaines hypothèses simplificatrices doivent donc être faites à priori pour pouvoir traiter ces problème ; ces hypothèses sont d'ailleurs souvent justifiées pour des raisons de calculs d'adresses. Nous sommes alors en face de problèmes réduits à des dimensions convenables.

Ce n'est pas notre but d'aborder de telles situations.

Le chapitre VI présentera quelques résultats annexes sur la "complexité" de certains algorithmes élaborés dans les chapitres III et IV. En particulier nous montrons comment certains problèmes de partitionnement qui, relativement au nombre d'éléments n , rentrent dans une classe de problèmes NP-complets [6], peuvent,si on les particularise, se résoudre de manière approchée en $A\ n\ Log_2\ n$ opérations élémentaires.

A la fin de ce même chapitre nous montrons l'analogie de certains problèmes traités avec des problèmes de "Clustering" manipulant des nuages de points. Une adaptation des algorithmes décrits au chapitre III est alors envisagée.

Pour conclure, signalons que ce travail n'est pas un catalogue de "recettes" et de techniques de restructuration de programme. Notre but a été d'essayer de comprendre, sur des cas extrêmement simples et dépouillés d'artifices extérieurs, la nature des problèmes posés. Nous avons toujours chercher des voies originales dans un domaine de recherche dont les bases mêmes sont certainement encore mal établies.

NOTATIONS

Nous noterons un vecteur de $\mathbb{R}^n$

$$X = \begin{pmatrix} X(1) \\ X(2) \\ \vdots \\ X(n) \end{pmatrix}$$

X^T désignera le vecteur ligne :

$$(X(1),X(2),\ldots,X(n))$$

$\mathcal{M}_{n,n}(\mathbb{R})$ (respectivement $\mathcal{M}_{n,n}(\mathbb{C})$) désigne l'ensemble des matrices à n lignes et n colonnes à coefficients dans $\mathbb{R}$ (respectivement dans $\mathbb{C}$).

A(i,j) désigne l'élément de la ligne i colonne j.

A^T désigne la matrice transposée de A .

$E_{i,j}$ est la matrice de $\mathcal{M}_{n,n}(\mathbb{R})$ définie par :

$$E_{i,j}(k,\ell) = \begin{cases} 1 & \text{si } i=k \text{ , } j=\ell \\ 0 & \text{sinon} \end{cases}$$

φ_p désigne la norme sur $\mathbb{R}^n$:

$$\varphi_p(X) = \left(\sum_{i=1}^{n} |X(i)|^p \right)^{1/p} \qquad p \geq 1$$

$S_{\varphi_p\varphi_p}$ est la norme d'opérateurs sur $\mathcal{M}_{n,n}(\mathbb{R})$:

$$S_{\varphi_p\varphi_p}(A) = \max_{\substack{\varphi_p(X) = 1 \\ X \in \mathbb{R}^n}} \varphi_p(AX)$$

Nous noterons aussi :

$$\varphi_2(X) = (X^T X)^{1/2}$$

A étant une matrice définie positive de $\mathcal{M}_{n,n}(\mathbb{R})$

nous noterons $\| \ \|_A$ la norme sur $\mathbb{R}^n$:

$$\|X\|_A = (X^T A X)^{1/2} .$$

CHAPITRE I

GESTION DE MEMOIRE PAGINEE ET EXPRESSIONS MATRICIELLES DE QUELQUES PROBLEMES DE RESTRUCTURATION

I - GESTION DE MEMOIRE PAGINEE ET PROBLEMES DE RESTRUCTURATION

1.1. UN CALCULATEUR ELEMENTAIRE

Nous nous bornerons à quelques notions nécessaires pour introduire les problèmes traités par la suite.

De nombreuses variantes sont possibles autour du même thème mais les problèmes de base sont les mêmes.

1.1.1. Mémoire programme

Notre calculateur dispose d'une "mémoire programme" contenant le "code" (instructions) nécessaire à l'exécution du programme.

1.1.2. Mémoires de type "donnée"

Les mémoires de type "donnée" contiendront les données (variables, constantes, ...) utilisées par le programme. L'unité de taille mémoire sera appelée "mot".

Nous distinguerons dans ces mémoires de type "donnée" :

1/ La "mémoire opératoire" divisée en m blocs de d mots,

2/ La "mémoire périphérique" divisée en blocs de d mots mais de taille aussi grande que l'on veut.

1.1.3. Transferts d'information entre les deux types de mémoire "données"

Des transferts peuvent se faire entre mémoire "donnée" "opératoire" et mémoire "donnée" "périphérique" mais uniquement sur les blocs de d mots cités plus haut.

1.2. NOTATIONS ET DEFINITIONS

1.2.1. Ensemble des données

L'ensemble fini des données sera noté :

$$X = \{x_1, x_2, \ldots, x_n\}$$

où x_i sera une "donnée" pouvant être mémorisée sur un mot.

1.2.2. Programme - Programme à références simples

Nous supposerons que notre programme "utilise" ou "référence" successivement L ensembles des données aux "instants" $t=1,2,\ldots,L$.

Nous supposerons notre programme à "références simples" c'est à dire que l'ensemble de données référencé à l'instant t est du type :

$$\{\ell_t\} \qquad \ell_t \in X$$

Nous noterons :

$$\mathcal{L} = \{\ell_1\} , \{\ell_2\} ,\ldots, \{\ell_L\} \quad .$$

1.2.3. Organisation des données en "pages"

Notre machine n' acceptant que des transferts par blocs de d mots nos données seront réparties en r "pages" $y_1,y_2,\ldots,y_r$ contenant chacune au plus d données.

Ces "pages" pourront être mémorisées dans un bloc de d mots, soit en mémoire "opératoire", soit en mémoire "périphérique".
Nous noterons :

$$Y = \{y_1,y_2,\ldots,y_r\}$$

Nous aurons :

$$n \leq r \times d \quad .$$

Nous verrons plus loin (chapitre I, paragraphe 2.1.) comment représenter une "organisation de données" en pages. π désignera une telle organisation.

1.2.4. Suite des pages référencées

Les données étant réparties en page suivant π le programme à références simples "utilise" une suite d'ensembles de pages du type :

$$\{p_1\} , \{p_2\} ,\ldots, \{p_t\} ,\ldots, \{p_L\}$$

$$p_t \in Y \quad , \quad t=1,2,\ldots,L$$

p_t désigne la page contenant ℓ_t .

1.2.5. Contrainte pour l'exécution du programme

La seule contrainte que nous imposerons pour l'exécution du programme est que la page utilisée à l'instant t doit se trouver en "mémoire opératoire".

Nous allons voir comment par un transfert de pages précédent l'utilisation de p_t nous réaliserons cette condition.

Précisons tout de suite que nous ferons abstraction des moyens pratiques réalisant de tels transferts et de tout problème de calculs d'adresses résultant d'une organisation de donnée π .

1.3. GESTION DE MEMOIRE - GESTION DE MEMOIRE VALIDE [5] [8]

1.3.1. Contexte

Nous possédons un ensemble de "pages"

$$Y = \{y_1, y_2, \ldots, y_r\}$$

Notre mémoire opératoire peut contenir m pages.
Le programme nous fournit une suite

$$\{p_1\}\ ,\ \{p_2\}\ ,\ \ldots,\ \{p_L\} \qquad p_t \in Y \qquad t=1,2,\ldots,L\ .$$

1.3.2. Gestion de mémoire

Nous appellerons "gestion de mémoire" un couple (R,S) de suites du type :

$$R = (r_1, r_2, \ldots, r_t, \ldots, r_L) \qquad r_t \in \mathcal{P}(Y)$$

$$S = (s_1, s_2, \ldots, s_t, \ldots, s_L) \qquad S_t \in \mathcal{P}(Y)$$

R représente la suite des pages "rentrer" en mémoire opératoire.
S représente la suite des pages à "sortir" de la mémoire opératoire.

Plus précisément, l'utilisation de la page p_t sera précédée des transferts définis par r_t et s_t .

1.3.3. Suite des contenus en pages de la mémoire opératoire

Soit (R,S) une gestion de mémoire.

Posons $M_o = \emptyset$

La relation

$$M_t = (M_{t-1} \setminus s_t) \cup r_t \qquad t=1,2,\ldots,L$$

définit une suite finie $\{M_t\}_{t=1,2,\ldots,L}$

de sous-ensembles de Y représentant les contenus successifs en pages de la mémoire opératoire, résultant de la gestion de mémoire (R,S).

1.3.4. Gestion de mémoire valide

(R,S) sera dit valide si :

α) $r_t \cap M_{t-1} = \emptyset$ $\qquad t=1,2,\ldots,L$

β) $s_t \subset M_{t-1}$ $\qquad t=1,2,\ldots,L$

γ) $\{p_t\} \subset M_t$ $\qquad t=1,2,\ldots,L$

δ) $\text{card}\,(M_t) \leq m$ $\qquad t=1,2,\ldots,L$

α) signifie que les pages amenées ne figurent pas déjà en mémoire opératoire,

β) signifie que les pages à sortir sont bien en mémoire opératoire,

γ) signifie que les pages référencées par le programme sont en mémoire opératoire,

δ) signifie que le nombre de pages en mémoire opératoire ne dépasse pas m .

1.4. COÛT D'EXECUTION EN DEMANDES DE PAGES

X et la suite $\{\ell_1\},\{\ell_2\},\ldots,\{\ell_L\}$ sont fixés.

Considérons le déroulement de notre programme sur une machine effectuant des transferts de blocs de taille d et ayant une capacité mémoire opératoire de m blocs (pages).

Les données étant organisées suivant π et la mémoire gérée par (R,S) valide on pose :

$$DP(m,d,\pi,R,S) = \sum_{t=1}^{L} \text{card}\,(r_t)$$

$$(\text{card}\,(r_t) = 0 \quad \text{si} \quad r_t = \emptyset) \ .$$

1.5. GESTION DE MEMOIRE "A LA DEMANDE" ET ALGORITHME DE REMPLACEMENT

La gestion de mémoire la plus réaliste est celle qui amène à l'instant t la page unique p_t si celle-ci ne figure pas en mémoire.

1.5.1. Définition : Gestion de mémoire "à la demande"

Une gestion de mémoire (R,S) est dite "à la demande" si :

$\alpha)$ (R,S) est valide

$\beta)$ $\text{card}(r_t) \leq 1 \qquad \text{card}(s_t) \leq 1 \qquad t=1,2,\ldots,L$

$\gamma)$ $p_t \in M_{t-1} \Rightarrow r_t = s_t = \emptyset \qquad t=1,2,\ldots,L$

$\delta)$ $p_t \notin M_{t-1}$ et $\text{card}(M_{t-1}) < m \Rightarrow s_t = \emptyset \qquad t=1,2,\ldots,L$

Remarque

(R,S) étant valide, on a forcément $r_t = p_t$ si $p_t \notin M_{t-1}$.
Il faut alors remarquer que la gestion de mémoire est parfaitement déterminée si on connait la page à sortir de la mémoire lorsque

$$p_t \notin M_{t-1} \qquad \text{card}(M_{t-1}) = m$$

1.5.2. Algorithme de remplacement [1] [5]

La "stratégie permettant" de choisir la page de M_t à sortir sera appelée "Algorithme de remplacement".
Plutôt que de définir formellement une stratégie donnons deux exemples que nous utiliserons par la suite.

Algorithme de remplacement LRU (Least Recently Use)

Supposons que nous ayons à choisir une page dans M_{t-1}. Pour chaque page $a_i \in M_{t-1}$ définissons :

$$\theta_i = \max_{\substack{\theta<t \\ p_\theta=a_i}} \theta$$

On choisira pour s_t la page a_{i_o} telle que :

$$\theta_{i_o} = \min_{i\in\{i:a_i\in M_{t-1}\}} \theta_i$$

Cela signifie que la page choisie est celle qui est la moins récemment référencée.

Nous noterons (R^{LRU}, S^{LRU}), toute gestion de mémoire à la demande obtenue avec l'algorithme LRU.

Algorithme de remplacement "optimum" Belady

Supposons que nous ayons à choisir une page dans M_{t-1}. Pour chaque page $a_i \in M_{t-1}$ définissons :

$$\theta_i = \underset{\substack{\theta>t \\ p_\theta=a_i}}{\text{Min}}\ \theta$$

éventuellement $\theta_i = +\infty$ si la page n'est pas référencée par la suite.

On choisira pour s_t la page a_{i_o} telle que :

$$\theta_{i_o} = \underset{i\in\{i:a_i\in M_{t-1}\}}{\text{Max}}\ \theta_i$$

Cela signifie que la page référencée le plus longtemps après sera choisie.

Optimalité [1] [5]

Belady a montré que cet algorithme de remplacement était celui qui minimisait le nombre de demande de pages, parmi toutes les gestions de mémoire "à la demande", nous l'appellerons algorithme "optimum" ; nous noterons (R^{opt}, S^{opt}) toute gestion de mémoire à la demande obtenue avec l'algorithme optimum.

1.6. GESTION DE MEMOIRE VALIDE ET GESTION DE MEMOIRE A LA DEMANDE

1.6.1. PROPRIETE

Etant donnée une gestion de mémoire valide (R,S) il existe une gestion de mémoire à la demande (R^D, S^D) telle que :

$$DP(m,d,\pi,R^D,S^D) \leq DP(m,d,\pi,R,S).$$

La démonstration (peu difficile mais longue) figure dans [5].

1.6.2. Conséquence

La propriété 1.6.1. justifie le peu d'intérêt des gestions de mémoire générales. Cette notion nous sera utile comme outil de démonstration mais nous nous intéresserons uniquement à des gestions de mémoire "à la demande".

1.7. PROBLEMES DE RESTRUCTURATION [7] [8]

1.7.1. RESTRUCTURATION D'UN ENSEMBLE DE DONNEES

Nous avons vu que l'organisation de données π est paramètre du coût en demandes de pages. Naturellement, il se pose le problème de savoir si en modifiant π c'est-à-dire en "restructurant" notre ensemble de données on peut diminuer le nombre de demande de pages.

Suivant les cas l'algorithme de remplacement pourra être fixé ou non.

1.7.2. UNE FORMULATION MATRICIELLE

Le premier problème étudié sera celui de "la minimisation du nombre de changements de pages" par restructuration.

Les méthodes élaborées pour la résolution de ce problème nous ont permis par la suite de fournir une aide précieuse pour l'approche de quelques problèmes beaucoup plus difficiles.

1.7.3. PROBLEME 1 MINIMISATION DU NOMBRE DE CHANGEMENTS DE PAGES

Nous supposons que m (capacité en pages de la mémoire opératoire) est égale à 1.

Une seule gestion de mémoire à la demande est possible car M_t est parfaitement défini : $M_t = \{p_t\}$. Le nombre de demandes de pages sera en fait le nombre de fois que p_t sera différent de p_{t+1} (plus la demande de p_1).

π désignant une organisation de données en page de d éléments au plus on notera $CP(d,\pi)$ ce nombre de changements de pages.

$$CP(d,\pi) = \text{nombre de fois que } p_t \neq p_{t+1} \qquad t=1,2,\ldots,L-1.$$

Le premier problème posé sera :

Trouver une organisation de donnée π^* telle que :

$$CP(d,\pi^*) = \min_{\pi} CP(d,\pi) .$$

1.7.4. PROBLEME 2

Soit $m > 1$, π une organisation de données et "alg" un algorithme de remplacement.

(R^{alg}, S^{alg}) désignera toute gestion de mémoire à la demande obtenue avec l'algorithme "alg".

Un deuxième type de problème sera :

Trouver une organisation de donnée π^* telle que :

$$DP(m,d,\pi^*,R^{alg},S^{alg}) = \min_{\pi} DP(m,d,\pi,R^{alg},S^{alg})$$

Nous étudierons plus précisément :

Trouver π^* telle que :

$$DP(m,d,\pi^*,R^{LRU},S^{LRU}) = \min_{\pi} DP(m,d,\pi,R^{LRU},S^{LRU})$$

1.7.5. PROBLEME 3

m étant toujours supérieur à 1 nous nous proposons de minimiser le nombre de demande de pages parmi toutes les organisations de données et les gestions de mémoire à la demande.

Etant donnée l'optimalité de la gestion (R^{opt},S^{opt}) le problème est formellement équivalent à :

Trouver une organisation de donnée π^* telle que :

$$DP(m,d,\pi^*,R^{opt},S^{opt}) = \min_{\pi} DP(m,d,\pi,R^{opt},S^{opt}) .$$

II - RESTRUCTURATION ET CHANGEMENT DE PAGES. MATRICE D'ADRESSES JOINTIVES

2.1. ORGANISATION DE DONNEES

2.1.1. Traduction vectorielle d'une organisation de données

Soit $X = \{x_1, x_2, \ldots, x_n\}$ l'ensemble des données à répartir en r pages $Y_1, Y_2, \ldots, Y_r$ pouvant chacune contenir au plus d données.

A chaque page Y_k on associe un vecteur V_k de $\mathbb{R}^n$ vérifiant :

$$V_k(i) = \begin{cases} +1 & \text{si} \quad x_i \in Y_k \\ 0 & \text{sinon} \end{cases} \qquad i=1,2,\ldots,n \ .$$

Pour représenter une organisation de données on choisira donc un système de r vecteurs de $\mathbb{R}^n$ $(V_1, V_2, \ldots, V_r)$ vérifiant les contraintes (C) :

$$(C) \begin{cases} (C1) \quad V_k(i) = \begin{cases} 0 \\ 1 \end{cases} & \begin{matrix} k=1,2,\ldots,r \\ i=1,2,\ldots,n \end{matrix} \\ (C2) \quad \sum_{i=1}^{n} V_k(i) \leq d & k=1,2,\ldots,r \\ (C3) \quad \sum_{k=1}^{r} V_k(i) = +1 & i=1,2,\ldots,n \end{cases}$$

La contrainte (C2) signifie qu'une page ne peut contenir plus de d données.
La contrainte (C3) signifie que chaque donnée appartient à une page et une seule.

2.1.2. Indicateur d'appartenance

Nous utiliserons par la suite une fonction f :

$f : \quad X \times X \to \{0,1\}$

qui étant donnée une organisation de données particulière est définie par :

$$f(x_i,x_j) = \begin{cases} 0 & \text{si} \quad x_i \text{ et } x_j \in \text{même page} \\ 1 & \text{sinon.} \end{cases}$$

Soit E_{ij} la matrice de base de $\mathcal{M}_{n,n}(\mathbb{R})$. Pour l'organisation de donnée représentée par les vecteurs $(V_1,V_2,\ldots,V_r)$ satisfaisant les contraintes (C) on a :

$$f(x_i,x_j) = \sum_{\substack{k_1=1,\ldots,r \\ k_2=1,\ldots,r \\ k_1 \neq k_2}} V_{k_1}^T E_{ij} V_{k_2}$$

Démonstration

On a $V_{k_1}^T E_{ij} V_{k_2} = (V_{k_1}^T e_i)(e_j^T V_{k_2})$.

Le seul terme $V_{k_1}^T e_i$ non nul est celui dont l'indice k_1 est tel que $x_i \in Y_{k_1}$ et de même pour $V_{k_2}^T e_j = e_j^T V_{k_2}$.

Si x_i et $x_j \in$ même page les termes $V_{k_1}^T b_{ij} V_{k_2}$, $k_1 \neq k_2$ sont donc tous nuls.

Si x_i et $x_j \notin$ même page un seul des termes $V_{k_1}^T E_{ij} V_{k_2}$, $k_1 \neq k_2$ est non nul et égal à 1.

2.2. NOMBRE DE CHANGEMENTS DE PAGES : MATRICE D'ADRESSES JOINTIVES

Soit $\mathcal{L} = \{\ell_1\},\{\ell_2\},\ldots,\{\ell_t\},\{\ell_{t+1}\},\ldots,\{\ell_L\}$ la suite des ensembles de données à un élément référencés par le programme.

Supposons l'organisation de données représentée par $(V_1,V_2,\ldots,V_r)$.

2.2.1. Changement de page

Considérons les références $\ell_t = x_{i_t}$ et $\ell_{t-1} = x_{i_{t-1}}$.
Il y aura changement de page si ℓ_t et ℓ_{t-1} n'appartiennent pas à la même page.
Le nombre de changement de page lors de la référence à ℓ_t sera :

$$f(\ell_t,\ell_{t-1}) = \sum_{k_1 \neq k_2} V_{k_1}^T E_{i_t,i_{t-1}} V_{k_2}$$

2.2.2. Nombre total de changements de pages ;
Matrice d'adresses jointives

Le nombre total de changements de pages durant l'exécution du programme sera :

$$\sum_{t=2}^{L} f(\ell_t,\ell_{t-1}) = \sum_{k_1 \neq k_2} V_{k_1}^T \left(\sum_{t=2}^{L} E_{i_t,i_{t-1}} \right) V_{k_2}$$

Nous pouvons aussi transposer l'expression matricielle suivante ce qui donne :

$$\sum_{k_1 \neq k_2} V_{k_2}^T \left(\sum_{t=2}^{L} E_{i_{t-1},i_t} \right) V_{k_1}$$

et après permutation des indices :

$$\sum_{k_1 \neq k_2} V_{k_1}^T \left(\sum_{t=2}^{L} E_{i_{t-1},i_t} \right) V_{k_2}$$

En faisant la demi somme de ces deux expressions nous obtenons la forme symétrique :

$$\sum_{k_1 \neq k_2} V_{k_1}^T \left[\frac{1}{2} \left(\sum_{t=2}^{L} E_{i_t,i_{t-1}} + E_{i_{t-1},i_t} \right) \right] V_{k_2} = \frac{1}{2} \sum_{k_1 \neq k_2} V_{k_1}^T A V_{k_2} .$$

$$A = \sum_{t=2}^{L} \left(E_{i_t,i_{t-1}} + E_{i_{t-1},i_t} \right) .$$

Nous pouvons écrire $A = \sum_{i,j} A(i,j)E_{ij}$

En identifiant les deux expressions de A il vient :

$$A(i,j) = \begin{cases} \text{nombre de fois que } (\ell_t,\ell_{t-1}) = (x_i,x_j) \ ; & t=2,\ldots,L \\ + \\ \text{nombre de fois que } (\ell_t,\ell_{t-1}) = (x_j,x_i) \ ; & t=2,\ldots,L \end{cases}$$

$i=1,2,\ldots,n$
$j=1,2,\ldots,n$

Cette matrice A symétrique de $\mathcal{M}_{n,n}(\mathbb{R})$ est appelée matrice d'adresses jointives du programme.
Sous des formes assez voisines, c'est un outil déjà utilisé dans [2] [4] [7] [8].

Le nombre de changements de pages sera donc fonction de l'organisation de données π et de la matrice d'adresses jointives A , on notera maintenant :

$CP(d,\pi,A)$

Si l'organisation de données est représentée par un système de vecteurs $V_1,V_2,\ldots,V_r$ on aura :

$$CP(d,\pi,A) = \frac{1}{2} \sum_{k_1 \neq k_2} V_{k_1}^T \, A \, V_{k_2}$$

2.3. NOMBRE DE DEMANDES DE PAGES :

UNE SEULE PAGE EN MEMOIRE OPERATOIRE

Nous avons vu que dans le cas où la mémoire opératoire possède l'emplacement pour une seule page le nombre de demandes de pages est égal au nombre de changements de pages plus le chargement de la page référencée à l'instant $t = 1$.

2.4. RESTRUCTURATION : PROBLEME 1

Le problème 1 de restructuration (paragraphe 1.7.3.) prend donc la forme :

Trouver l'organisation de donnée π^* telle que :

$$CP(d,\pi^*,A) = \underset{\pi}{\text{Min}}\ CP(d,\pi^*,A) \ .$$

La représentation d'une organisation de donnée, développée plus haut nous servira, dans leschapitres suivants à étudier le problème de minimisation et dans l'immédiat à étudier dans un formalisme identique au problème 1 des problèmes proches des problèmes 2 et 3.

III - GESTION DE MEMOIRE ET RESTRUCTURATION
MATRICE D'ADRESSES JOINTIVES GENERALISEES

3.1. BORNES INFERIEURES POUR LE NOMBRE DE DEMANDE DE PAGES

3.3.1. Gestion de mémoire "donnée par donnée"

Si nous considérons des pages de taille d = 1 au nombre de n l'organisation de données est unique. Nous pouvons donc envisager le déroulement du programme avec une gestion de mémoire à la demande et l'algorithme de remplacement optimum. Ceci nous donnera un nombre de demandes de pages uniquement fonction de la suite des références aux données $\mathcal{L} = \{\ell_1\},\{\ell_2\},\ldots,\{\ell_L\}$ et de la capacité en pages de la mémoire opératoire m.

On notera : $DP(m,1,\pi,R^{opt},S^{opt}) = F(m)$.

3.3.2. Théorème (Johnson [8])

Pour une capacité de mémoire opératoire de m pages de d données, pour une organisation de donnée π

$$DP(m,d,\pi,R,S) \geq \frac{1}{d} F(m \times d)$$

$\forall$ m,d,π,(R,S) valide.

3.2. ALGORITHME DE REMPLACEMENT LRU ET RESTRUCTURATION

L'algorithme de remplacement LRU a été défini au paragraphe 1.5.2.

Nous commencerons par étudier le cas où deux pages peuvent se trouver en mémoire opératoire.

3.2.1. Cas de deux pages en mémoire

Rappelons la fonction "indicateur d'appartenance"

$$f : X \times X \to [0,1]$$

$$f(x_i,x_j) = \begin{cases} 0 & \text{si } x_i \text{ et } x_j \in \text{même page} \\ 1 & \text{sinon.} \end{cases}$$

Notre programme fournit la suite :

$$\{\ell_1\},\{\ell_2\},\ldots,\{\ell_t\},\{\ell_{t+1}\},\ldots,\{\ell_L\}.$$

Essayons d'exprimer le nombre de demandes de pages lors du passage de $\{\ell_{t-1}\}$ à $\{\ell_t\}$.

- Si ℓ_{t-1} et ℓ_{t-2} n'appartiennent pas à la même page elles appartiennent aux deux pages le plus récemment référencées et ce sont donc les deux pages en mémoire après la référence à ℓ_{t-1} . Le nombre de demandes de pages est donc égal à :

$$f(\ell_t,\ell_{t-1})f(\ell_t,\ell_{t-2}) .$$

Malheureusement, il est déjà difficile de travailler avec ce produit et nous nous contenterons de la majoration :

$$f(\ell_t,\ell_{t-1})f(\ell_t,\ell_{t-2}) \leq \frac{1}{2}[f(\ell_t,\ell_{t-1})+f(\ell_t,\ell_{t-2})].$$

- Si ℓ_{t-1} et ℓ_{t-2} appartiennent à la même page il faudrait faire intervenir ℓ_{t-3} . Nous pouvons remarquer que la quantité

$$\frac{1}{2}\,[f(\ell_t,\ell_{t-1}) + f(\ell_t,\ell_{t-2})]$$

majore dans tous les cas le nombre de demandes de pages.

En effet s'il n'y a pas demande de page la majoration est évidente et s'il y a demande de page :

ℓ_t , ℓ_{t-1} $\notin$ même page : $f(\ell_t,\ell_{t-1}) = 1$

ℓ_t, ℓ_{t-2} $\notin$ même page : $f(\ell_t,\ell_{t-2}) = 1$

et la majoration reste vraie.

Nous prendrons comme majoration du coût de passage de ℓ_{t-1} à ℓ_t la quantité

$$\frac{1}{2}[f(\ell_t,\ell_{t-1})+f(\ell_t,\ell_{t-2})]$$

Pour t=2 on prendra :

$$\frac{1}{2}[f(\ell_t,\ell_{t-1})+1]$$

et pour t=1 on prendra : 1 .

Une majoration du coût en demandes de pages total sera donc :

$$1 + \frac{1}{2}[f(\ell_2,\ell_1)+1] + \sum_{t=3}^{L} \frac{1}{2}[f(\ell_t,\ell_{t-1})+f(\ell_t,\ell_{t-2})]$$

$$\frac{3}{2} + \frac{1}{2} \sum_{t=2}^{L} f(\ell_t,\ell_{t-1}) + \frac{1}{2} \sum_{t=3}^{L} f(\ell_t,\ell_{t-2}) \ .$$

Nous retrouvons le formalisme des matrices d'adresses jointives du paragraphe 2.

Notons A la matrice d'adresses jointives du programme.

Notons $A^{(2)}$ la matrice de $\mathcal{M}_{n,n}(\mathbb{R})$ définie par :

$$A^{(2)}(i,j) = \begin{cases} \text{nombre de fois que } (\ell_{t-2},\ell_t) = (i,j) \; ; t=3,\ldots,L \; . \\ + \\ \text{nombre de fois que } (\ell_{t-2},\ell_t) = (j,i) \; ; t=3,\ldots,L \; . \end{cases}$$

Notre majoration s'exprime alors par :

$$\frac{3}{2} + \frac{1}{2} \sum_{k_1 \neq k_2} V_{k_1}^T \frac{A}{2} V_{k_2} + \frac{1}{2} \sum_{k_1 \neq k_2} V_{k_1}^T \frac{A^{(2)}}{2} V_{k_2}$$

i.e. $$\frac{3}{2} + \frac{1}{2} \sum_{k_1 \neq k_2} V_{k_1}^T \left(\frac{A+A^{(2)}}{2}\right) V_{k_2} \; .$$

La matrice $\frac{A+A^{(2)}}{2}$ symétrique à coefficients positifs ou nuls sera appelée "matrice d'adresses jointives généralisée" ou matrice d'adresses bi-jointives.

Nous avons en fait :

$$DP(2,d,\pi,A^{LRU},S^{LRU}) \leq CP(d,\pi, \frac{A+A^{(2)}}{2})$$

Nous allons généraliser ce résultat dans le paragraphe 3.2.2.

3.2.2. Cas de m pages en mémoire

Le problème est de plus en plus difficile et nous proposons quelques outils de travail.

S'il y a faute de page lors de la référence ℓ_t aucune des pages contenant $\ell_{t-1},\ell_{t-2},\ldots,\ell_{t-m}$ ne contient ℓ_t car ces pages sont en mémoire opératoire. On a donc :

$$f(\ell_t,\ell_{t-1}) = 1$$
$$f(\ell_t,\ell_{t-2}) = 1$$
$$\vdots$$
$$f(\ell_t,\ell_{t-m}) = 1$$

Si $(\mu_1,\mu_2,\ldots,\mu_m)$ est un m-uplet de réels positifs tel que :

$$\mu_1+\mu_2+\ldots+\mu_m = 1$$

On a :

$$\sum_{q=1}^{m} \mu_q \, f(\ell_t,\ell_{t-q}) = +1$$

Le nombre de demandes de pages est donc majoré par :

$$\sum_{q=1}^{m} \mu_q \, f(\ell_t,\ell_{t-q})$$

Si $t \leq m$ on majorera par :

$$\sum_{q=1}^{t-1} \mu_q \, f(\ell_t,\ell_{t-q}) + \sum_{q=t}^{m} \mu_q$$

Le nombre total de demandes de pages sera donc majoré par :

$$\sum_{t=1}^{m} \left(\sum_{q=1}^{t-1} \mu_q \, f(\ell_t,\ell_{t-q}) + \sum_{q=t}^{m} \mu_q \right) + \sum_{t=m+1}^{L} \sum_{q=1}^{m} \mu_q \, f(\ell_t,\ell_{t-q})$$

$$\sum_{t=1}^{m} \sum_{q=t}^{m} \mu_q + \sum_{q=1}^{m} \mu_q \sum_{t=q+1}^{L} f(\ell_t,\ell_{t-q}) \, .$$

Notons $A^{(q)}$ la matrice symétrique positive de $\mathcal{M}_{n,n}(\mathbb{R})$ définie par :

$$A^{(q)}(i,j) = \begin{cases} \text{nombre de fois que } (\ell_t,\ell_{t-q}) = (i,j) \; ; \; t=q+1,\ldots,L \\ + \\ \text{nombre de fois que } (\ell_t,\ell_{t-q}) = (j,i) \; ; \; t=q+1,\ldots,L \end{cases}$$

D'après un calcul déjà effectué à propos de la matrice d'adresses jointives on a :

$$\sum_{t=q+1}^{L} f(\ell_t,\ell_{t-q}) = \frac{1}{2} \sum_{k_1 \neq k_2} V_{k_1}^{T} \, A^{(q)} \, V_{k_2}$$

Donc :

$$\sum_{q=1}^{m} \mu_q \sum_{t=q+1}^{L} f(\ell_t, \ell_{t-q}) = \frac{1}{2} \sum_{q=1}^{m} \mu_q \sum_{k_1 \neq k_2} V_{k_1}^T A^{(q)} V_{k_2}$$

$$= \frac{1}{2} \sum_{k_1 \neq k_2} V_{k_1}^T \left(\sum_{q=1}^{m} \mu_q A^{(q)} \right) V_{k_2}$$

$$= CP(d, \pi, A)$$

avec

$$A = \sum_{q=1}^{m} \mu_q A^{(q)}$$

A sera appelée "matrices d'adresses jointives généralisée" ou matrice d'adresses **m**-jointives

$$DP(m,d,\pi,A^{LRU},S^{LRU}) \leq CP(d,\pi,A) + \sum_{t=1}^{m} \sum_{q=t}^{m} \mu_q$$

Le cas du paragraphe 3.2.1. correspond à :

$$m = 2 \qquad \mu_1 = \mu_2 = \frac{1}{2} .$$

3.2.2. Restructuration : problème 2

L'expression $DP(m,d,\pi,R^{LRU},S^{LRU})$ étant difficilement manipulable nous lui substituerons la majoration ci-dessus ce qui nous conduira a un problème de restructuration du type :

$$\min_{\pi} CP(d,\pi,A)$$

identique au problème 1.

3.3. ALGORITHME DE REMPLACEMENT OPTIMUM ET RESTRUCTURATION

3.3.1. Notations

Soit $\mathcal{L} = \{\ell_1\},\{\ell_2\},\ldots,\{\ell_L\}$ la suite des ensembles de données référencées par le programme.

L'organisation de données π permet d'en déduire une suite

$$\{p_1\},\{p_2\},\ldots,\{p_L\}$$

de références aux pages.

La mémoire opératoire peut contenir m pages et possède donc m blocs pour contenir ces pages que l'on numérotera de 1 à m.

Nous définirons pour $t=0,1,2,\ldots,L$

$$E_t = \{e_t^1\}\{e_t^2\} \ldots \{e_t^m\}$$

où e_t^i est la page figurant dans le bloc i de la mémoire opératoire après la référence à la page p_t.

$$E_o = \emptyset , \emptyset , \ldots, \emptyset$$

On notera $$M_t = \bigcup_{i=1}^{m} \{e_t^i\}$$

3.3.2. Eclatement de la suite $\mathcal{L}$

A chaque référence ℓ_t nous affectons un bloc de la mémoire opératoire i_t dans lequel sera chargée une copie de la page p_t.

Pour i fixé, $1 \leq i \leq m$, nous pouvons donc extraire de $\mathcal{L}$ la sous-suite $\mathcal{L}_i$ formée des éléments de $\mathcal{L}$ auquels ont été affecté le bloc i.

Pour chaque sous-suite $\mathcal{L}_i$ nous pouvons construire une matrice d'adresses-jointives A_i.

3.3.3. Propriété 1

Soit $1 \leq i \leq m$

Le nombre de fois que $\{e_t^i\} \neq \{e_{t-1}^i\}$, $t=1,\ldots,L$, est égal à

$$1 + CP(d,\pi,A_i).$$

Démonstration

e_t^i ne peut être modifié que si la page ℓ_t est affectée au bloc i

$e_{t-1}^i \neq e_t^i$ si on est à la première référence ℓ_t telle que $i_t = i$ ou si la référence précédente a cet emplacement était différente, ce qui correspond bien au nombre de changements de page $CP(d,\pi,A_i)$ +le chargement initial.

3.3.4. Propriété 2

Il existe une gestion de mémoire à la demande (R^*,S^*) telle que $M_t^* \supseteq M_t$; t=1,2,...,L.

Démonstration

Nous allons construire une suite (R^i,S^i) i=1,2,...,L , de gestion de mémoire à la demande pour la suite des références en pages

$$\{p_1\}\ \{p_2\}\ \ldots\ \{p_i\}$$

et vérifiant

$$M_t^i \supseteq M_t \qquad t=1,2,\ldots,i$$

- On pose $(R^1,S^1) = (\{p_1\},\emptyset)$
- Supposons construit (R^i,S^i) et construisons (R^{i+1},S^{i+1}).

  On pose

$$r_t^{i+1} = r_t^i \qquad t=1,2,\ldots,i$$
$$s_t^{i+1} = s_t^i \qquad t=1,2,\ldots,i$$

  d'où

$$M_t^{i+1} = M_t^i \supseteq M_t \qquad t=1,2,\ldots,i$$

La page demandée en t=i+1 est p_{i+1}

PREMIER CAS

$$p_{i+1} \in M_i^{i+1} = M_i^i$$

alors

$$r_{i+1}^{i+1} = \emptyset \qquad ; \qquad s_{i+1}^{i+1} = \emptyset$$

dans ce cas on a :

$$M_{i+1}^{i+1} = M_i^{i+1} = M_i^i \cup \{p_{i+1}\} \supseteq M_i \cup \{p_{i+1}\} \quad \supseteq \quad M_{i+1} .$$

DEUXIEME CAS

$$p_{i+1} \notin M_i^{i+1} = M_i^i$$

$$\text{card } (M_i^{i+1}) < m$$

alors

$$r_{i+1}^{i+1} = \{p_{i+1}\} \qquad ; \qquad s_{i+1}^{i+1} = \emptyset$$

dans ce cas :

$$M_{i+1}^{i+1} = M_i^{i+1} \cup \{p_{i+1}\} = M_i^i \cup \{p_{i+1}\}$$

$$M_{i+1}^{i+1} \supseteq M_i \cup \{p_{i+1}\} \supseteq M_{i+1}$$

TROISIEME CAS

$$p_{i+1} \notin M_i^{i+1} = M_i^i$$

$$\text{card } (M_i^{i+1}) = m .$$

Remarquons que $p_{i+1} \notin M_i$ car $M_i^i \supseteq M_i$

- Si card $(M_i) = m$ pour amener en mémoire p_{i+1} il fallait donc sortir une page s de M_i.
  Cette page s figure aussi dans $M_i^{i+1} \supseteq M_i$.

On pose alors :

$$r_{i+1}^{i+1} = \{p_{i+1}\} \qquad ; \qquad s_{i+1}^{i+1} = \{s\}$$

dans ce cas :

$$M_{i+1}^{i+1} = (M_i^{i+1} \setminus \{s\}) \cup \{p_{i+1}\} = (M_i^i \setminus \{s\}) \cup \{p_{i+1}\}$$

$$M_{i+1}^{i+1} \supseteq (M_i \setminus \{s\}) \cup \{p_{i+1}\} = M_{i+1}$$

- Si card $(M_i) < m$ il existe donc $s \in M_i^{i+1} \setminus M_i$

On pose alors :

$$r_{i+1}^{i+1} = \{p_{i+1}\} \qquad ; \qquad s_{i+1}^{i+1} = \{s\}$$

dans ce cas :

$$M_{i+1}^{i+1} = (M_i^{i+1} \setminus \{s\}) \cup \{p_{i+1}\} \supseteq M_i \cup \{p_{i+1}\} = M_{i+1}$$

il est évident que si (R^i, S^i) est une gestion de mémoire à la demande par construction (R^{i+1}, S^{i+1}) est une gestion de mémoire à la demande.

On posera : $(R^L, S^L) = (R^*, S^*)$.

3.3.5. Propriété 3

$$DP(m,d,\pi,R^*,S^*) \leq m + CP(d,\pi,A) \quad \text{avec} \quad A = \sum_{i=1}^{m} A_i$$

Démonstration

S'il y a demande de page à l'instant t c'est que $p_t \notin M_{t-1}^*$. Mais alors $p_t \notin M_{t-1}$ et il y a donc forcément changement de la configuration E_t.

Ceci signifie qu'il existe i unique $1 \leq i \leq m$ tel que :

$$\{e_t^i\} \neq \{e_{t-1}^i\} .$$

Pour avoir une majoration du nombre de demandes de pages on prendra donc :

$$\sum_{i=1}^{m} (1+CP(d,\pi,A_i)) = m + CP(d,\pi,A) \quad ; \quad A = \sum_{r=1}^{m} A_i .$$

3.3.6. Propriété 4

Si on considère l'algorithme de remplacement optimum

$$DP(m,d,\pi,R^{opt},S^{opt}) \leq m + CP(d,\pi,A)$$

3.3.7. Restructuration : problème 3

De même que pour l'algorithme LRU nous substituerons à $DP(m,d,\pi,R^{opt},S^{opt})$ la majoration ci-dessus ce qui nous conduira à un problème du type :

$$\min_{\pi} CP(d,\pi,A)$$

identique au problème 1.
L'éclatement de $\mathcal{L}$ à choisir peut être guidé par plusieurs considérations. Nous en présentons une intéressante.

3.3.8. Amélioration d'une gestion de mémoire à la demande par restructuration

Notre programme s'exécute avec une certaine organisation de donnée π et une gestion de mémoire à la demande (R,S) .
Nous pouvons définir l'état E_t des emplacements mémoires

$$E_t = \{e_t^1\}\{e_t^2\} \ldots \{e_t^m\}$$

L'éclatement de la suite se fera de la manière suivante :
à l'instant t on demande p_t. Si p_t est en mémoire à l'emplacement i on pose $i = i_t$. Si p_t n'est pas en mémoire elle sera chargée à l'emplacement i et on posera $i = i_t$.

Nous pourrons donc construire m matrices d'adresses jointives A_i et

$$A = \sum_{i=1}^{m} A_i \ .$$

Par construction :

$$DP(m,d,\pi,R,S) = m + CP(d,\pi,A)$$

Si on trouve π^* tel que $CP(d,\pi^*,A) \leq CP(d,\pi,A)$ on aura donc :

$$DP(m,d,\pi^*,R^*,S^*) \leq m + CP(d,\pi^*,A^*) < DP(m,d,\pi,R,S).$$

3.3.9. Existence d'un éclatement optimal

Il existe forcément π^* tel que :

$$DP(m,d,\pi^*,R^{opt},S^{opt}) \leq DP(m,d,\pi,R^{opt},S^{opt}) \qquad \forall\ \pi$$

Pour la matrice A^* construite à partir de π^* (R^{opt},S^{opt}) on aura donc :

$$\min_{\pi} CP(d,\pi,A^*) = CP(d,\pi^*,A^*)$$

et le problème de trouver π^* pourra se présenter sous la forme d'un problème exposé au paragraphe 2.
Malheureusement A^* n'est pas accessible.

CHAPITRE II

FORMES MATRICIELLES EQUIVALENTES DES PROBLEMES DE RESTRUCTURATION : PARTITIONNEMENT DE MATRICES ET PROBLEMES DE DISTANCES EUCLIDIENNES MAXIMALES

0 - CONVENTIONS DE NOTATIONS

Problèmes d'optimisation

Pour un problème du type :

Trouver $x^* \in \mathcal{D}$ tel que

$$f(x^*) = \operatorname{Min}_{x \in \mathcal{D}} f(x)$$

Nous noterons seulement pour abréger :

$$\operatorname{Min}_{x \in \mathcal{D}} f(x)$$

la recherche de x^* étant sous entendue ; de même pour un problème de max.

Problèmes équivalents

Soient par exemple deux problèmes :

(1) $\operatorname{Min}_{x \in \mathcal{D}} f(x)$ (2) $\operatorname{Max}_{y \in \mathcal{K}} g(y)$

Nous dirons que ces deux problèmes sont équivalents si nous connaissons deux applications :

$$s_1 : \mathcal{D} \to \mathcal{K}$$

$$s_2 : \mathcal{K} \to \mathcal{D}$$

telles que :

x^* solution de (1) => $s_1(x^*)$ solution de (2)

y^* solution de (2) => $s_2(x^*)$ solution de (1) .

I - RAPPELS DE NOTIONS INTRODUITES AU CHAPITRE I

1.1. LA MATRICE D'ADRESSES JOINTIVES (chapitre I, paragraphe 2.2. ; chapitre I, paragraphes 3.2., 3.3.)

$A \in \mathcal{M}_{n,n}(\mathbb{R})$ A symétrique, positive.

1.2. LES VECTEURS D'ORGANISATION DE DONNEES (chapitre I, paragraphe 2.1.)

$V = (V_1, V_2, \ldots, V_r)$ vérifiant les contraintes (C)

$$(C)\begin{cases} (C1) \quad V_k(i) = \begin{cases} 0 \\ 1 \end{cases} & k=1,2,\ldots,r \quad i=1,2,\ldots,n \\ (C2) \quad \sum_{i=1}^{n} V_k(i) \leq d & k=1,2,\ldots,r \\ (C3) \quad \sum_{k=1}^{r} V_k(i) = 1 & i=1,2,\ldots,r \end{cases}$$

$\mathcal{V}$ sera l'ensemble des systèmes de vecteurs V vérifiant (C).

1.3. LE PROBLEME D'OPTIMISATION (chapitre I, paragraphe 2.4.)

$$\underset{(V_1,V_2,\ldots,V_r)\in\mathcal{V}}{\text{Min}} \ \frac{1}{2} \sum_{k_1 \neq k_2} V_{k_1}^T A V_{k_2} \qquad \text{(Pb.I)}$$

II - QUELQUES PROBLEMES EQUIVALENTS

2.1. TRANSFORMATION EN UN PROBLEME DE MAXIMUM

Notons U le vecteur de $\mathbb{R}^n$ défini par :

$$U(i) = 1 \qquad i=1,2,\ldots,n$$

La contrainte C3 s'exprime par :

$$U = \sum_{k=1}^{r} V_k$$

Nous avons donc :

$$U^T A U = \Big(\sum_{k_1=1}^{r} V_{k_1}\Big)^T A \Big(\sum_{k_2=1}^{r} V_{k_2}\Big)$$

$$= \sum_{k_1 \neq k_2} V_{k_1}^T A V_{k_2} + \sum_{k=1}^{r} V_k^T A V_k$$

d'où :

$$\sum_{k_1 \neq k_2} V_{k_1}^T A V_{k_2} = U^T A U - \sum_{k=1}^{r} V_k^T A V_k$$

Le problème I est donc équivalent au problème suivant :

$$\max_{(V_1,V_2,\ldots,V_r)\in\mathcal{V}} \sum_{k=1}^{r} V_k^T A V_k \qquad \text{(Pb.II)}$$

2.2. MODIFICATION DE LA DIAGONALE DE LA MATRICE D'ADRESSES JOINTIVES

2.2.1. Addition d'une matrice diagonale

Soit D une matrice diagonale de $\mathcal{M}_{n,n}(\mathbb{R})$.

$$V_k^T D V_k = \sum_{i=1}^{n} V_k^2(i) D(i,i)$$

$$\sum_{k=1}^{r} V_k^T D V_k = \sum_{i=1}^{n} \left(\sum_{k=1}^{r} V_k^2(i) \right) D(i,i)$$

Or $V_k^2(i) = V_k(i)$ (0 ou 1)

$$\sum_{k=1}^{r} V_k^T D V_k = \sum_{i=1}^{n} \left(\sum_{k=1}^{r} V_k(i) \right) D(i,i) = \sum_{i=1}^{n} D(i,i)$$

Nous avons donc :

$$\sum_{k=1}^{r} V_k^T (A+D) V_k = \sum_{k=1}^{r} V_k^T A V_k + \sum_{i=1}^{n} D(i,i)$$

Le terme ajouté ne dépend donc pas de $V = (V_1, V_2, \ldots, V_r)$ et le problème II est donc équivalent à :

$$\text{Max} \sum_{k=1}^{r} V_k^T (A+D) V_k$$
$$(V_1, V_2, \ldots, V_r) \in \mathcal{V} \qquad \text{(Pb.III)}$$

2.2.2. Propriété

Soit $A \in \mathcal{M}_{n,n}(\mathbb{R})$.

Il existe D matrice diagonale de $\mathcal{M}_{n,n}(\mathbb{R})$ telle que A + D soit définie positive.

Démonstration

Soit $X \in \mathbb{R}^n$ $\quad X \neq 0$

I la matrice identité de $\mathcal{M}_{n,n}(\mathbb{R})$

On pose $D = \mu I$

$$X^T(A+\mu I)X = X^TAX + \mu X^TX$$

On a : $X^TX = \varphi_2^2(X)$

$$|X^TAX| \leq \varphi_2(X).\varphi_2(AX) \leq \varphi_2^2(X) S_{\varphi_2\varphi_2}(A)$$

donc :

$$X^T(A+\mu I)X \geq \varphi_2^2(X)\ [\mu - S_{\varphi_2\varphi_2}(A)]$$

Si on prend $\mu > S_{\varphi_2\varphi_2}(A)$

on a $X^T(A+\mu I)X > 0$ $\quad$ Si $\varphi_2(X) \neq 0$

2.3. PROBLEMES A "TAILLES EGALES"

La contrainte (C2) limite le nombre des coefficients égaux à + 1. Nous allons transformer ce problème de manière à manipuler des vecteurs possédant exactement d composantes égales à + 1.

Posons $m = r \times d$ et construisons $A' \in \mathcal{M}_{m,m}(\mathbb{R})$ de la manière suivante :

$$A' = \left(\begin{array}{c|c} A & 0 \\ \hline 0 & 0 \end{array}\right) \begin{matrix} \\ \updownarrow\ m-n \\ \\ \end{matrix}$$

$$\qquad\qquad\quad \underset{m-n}{\longleftrightarrow}$$

A' sera symétrique, positive et sera appelée également matrice d'adresses jointives : elle correspond à l'adjonction de données fictives non référencées.

Soit V' un système de r vecteurs de $\mathbb{R}^m$

$(V'_1, V'_2, \ldots, V'_r)$

vérifiant les contraintes (C') :

$$(C') \begin{cases} (C'1) & V'_k(i) = \begin{cases} 0 \\ 1 \end{cases} \\ (C'2) & \sum_{i=1}^{m} V'_k(i) = d \qquad k=1,2,\ldots,r \\ (C'3) & \sum_{k=1}^{r} V'_k(i) = 1 \qquad i=1,2,\ldots,m \end{cases}$$

L'ensemble des systèmes V' sera noté $\mathcal{V}'$.

Il est évident qu'à chaque $V = (V_1, V_2, \ldots, V_r)$ vérifiant (C) on peut associer un (ou plusieurs) $V' = (V'_1, V'_2, \ldots, V'_r)$ vérifiant (C') et tel que :

$$\sum_{k=1}^{r} V_k^T A V_k = \sum_{k=1}^{r} V_k'^T A' V'_k$$

et réciproquement.

Ceci nous permettra de substituer au problème II un problème du type :

$$\max_{(V'_1, V'_2, \ldots, V'_r) \in \mathcal{V}'} \sum_{k=1}^{r} V_k'^T A' V'_k \qquad \text{Pb.IV}$$

dit "à tailles égales" la contrainte $C'2$ se traduisant en égalités et non plus en inégalités.

III - QUELQUES PROBLEMES RELATES AU PROBLEME A"TAILLES EGALES"

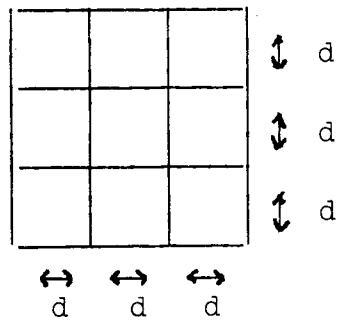
3.1. MATRICES BLOC-DIAGONALES ET PERMUTATIONS

PARTITIONNEMENT DE MATRICES

Soit $n = r \times d$

$A \in \mathcal{M}_{n,n}(\mathbb{R})$ symétrique positive

Soit $\mathcal{H}$ l'ensemble des indices des éléments d'une matrice $\mathcal{M}_{n,n}(\mathbb{R})$ qui constituent les r sous matrices de $\mathcal{M}_{d,d}(\mathbb{R})$ bloc-diagonales.



Exemple $n = 3d$

Notons :

$$\Psi(A) = \sum_{(i,j)\in\mathcal{H}} A(i,j)$$

Soit $\mathcal{S}$ l'ensemble des matrices de permutations de $\mathcal{M}_{n,n}(\mathbb{R})$

Nos problèmes précédents sont équivalents :

$$\max_{P\in\mathcal{S}} \Psi(P^T A P)$$

Nous regrouperons tous ces problèmes sous le nom de "partitionnement de matrices".

3.2. PARTITIONNEMENT DE GRAPHES

Soit $n = r \times d$

Soit $A \in \mathcal{M}_{n,n}(\mathbb{R})$ symétrique positive.

Soit G un graphe comportant n noeuds, les noeuds i et j étant joints par un arc non orienté de poids $A(i,j)$.

Le problème est de partitionner l'ensemble des noeuds de G en r ensembles de d noeuds de manière à minimiser la somme des poids des arcs joignant des noeuds appartenant à des parties différentes.

Il existe de nombreuses variantes autour de ce problème en théorie des graphes [12].

3.3. METHODES DE RESOLUTION

3.3.1. Quelques mots sur la complexité du problème

C'est un problème d'optimisation sur un ensemble fini mais même pour des matrices d'adresses jointives de tailles modestes l'énumération est illusoire (n = 40 , d = 20 , 155 117 520 configurations possibles).

Sous la forme de partitionnement de graphes ce problème est bien connu et non résolu.

Dans la classification de Karp [9] c'est un problème NP - complet. De nombreuses études sur la complexité de ce type de problèmes ont été faites [6][8][9].

3.3.2. Les heuristiques

De très nombreux auteurs ont proposé des méthodes "intuitives" pour résoudre ce problème ; certaines d'entre elles donnent d'excellents résultats mais il est souvent difficile de situer la solution obtenue par rapport à la solution réelle. On peut avoir un aperçu de ces méthodes dans [1][10].

Certaines de ces méthodes heuristiques s'interprêtent géométriquement (gradients,...) mais ces interprétations ne sont pas très fructueuses

Un certain nombre de méthodes d'amélioration locale sont utilisées pour affiner une solution proposée : parmi elles la méthode d'échange.

3.3.3. La méthode d'échange

Nous citons ce type de méthode car elle sera utilisée plus tard pour "affiner" des solutions proposées par d'autres algorithmes.

Il est inutile de l'expliciter formellement :
la méthode d'échange consiste à chercher si par "échange" d'un élément affecté au groupe k_1 avec un élément affecté au groupe k_2 on peut améliorer la valeur du critère [3][11].

3.3.4. Notre position vis-à-vis du problème

L'idée d'utiliser les vecteurs propres de la matrice d'adresses jointives apparaît dans [7].

Nous développerons cette idée en essayant de mettre en évidence "les cas simples" ou apparaîtront l'importance du rang de la matrice.

Nous nous efforcerons par la suite de voir ce que la connaissance de certains vecteurs propres peut apporter pour la résolution du problème.

IV - PROBLEMES DE DISTANCE EUCLIDIENNE MAXIMALE

4.1. PROBLEME

Soit $n = r \times d$ $\quad n,r,d \in \mathbb{N}$

Soit $V = (V_1, V_2, \ldots, V_r)$ $\quad r$ vecteurs de $\mathbb{R}^n$

vérifiant les contraintes (C) :

$$(C)\begin{cases} (C1) \quad V_k(i) = \begin{cases} 0 & i=1,2,\ldots,n \\ 1 & i=1,2,\ldots,r \end{cases} \\ (C2) \quad \sum_{i=1}^{n} V_k(i) = d & k=1,2,\ldots,r \\ (C3) \quad \sum_{k=1}^{r} V_k(i) = 1 & i=1,2,\ldots,n \end{cases}$$

$\mathcal{V}$ désigne l'ensemble des V vérifiant (C).

Soit $A \in \mathcal{M}_{n,n}(\mathbb{R})$ symétrique semi-définie positive.

Le problème posé est :

$$\underset{(V_1,V_2,\ldots,V_k)\in\mathcal{V}}{\text{Max}} \sum_{k=1}^{r} V_k^T A V_k$$

4.2. Matrice symétrique définie positive et distance euclidienne maximale dans $\mathbb{R}^{nr}$

Soit $X \in \mathbb{R}^{nr}$

Soit A_r la matrice de $\mathcal{M}_{nr,nr}(\mathbb{R})$ définie par :

$$A_r = \begin{pmatrix} A & & \\ & A & 0 \\ & \ddots & \\ 0 & & A \end{pmatrix}$$

(A_r est composée de r blocs diagonaux constituée de matrices A et de zéro ailleurs).

A_r est définie positive et notons $\| \ \|_{A_r}$ la norme sur $\mathbb{R}^{nr}$ définie par :

$$\|X\|^2_{A_r} = X^T A_r X$$

Soit $\mathcal{W}$ l'ensemble de $\mathbb{R}^{nr}$ définie par :

$$\mathcal{W} = \{ W = \begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_r \end{pmatrix} : (V_1,V_2,\ldots,V_r) \in \mathcal{V} \}$$

Nous avons :

$$\sum_{k=1}^{r} V_k^T A V_k = W^T A_r W = \|W\|^2_{A_r}$$

Nous pouvons donc poser le problème en ces termes :

$$\max_{W \in \mathcal{W}} \|W\|_{A_r}$$

Ce qui revient à chercher le point de $\mathcal{W}$ plus loin de l'origine pour $\|\ \|_{A_r}$.

4.2.2. Résolution

Un certain nombre d'algorithmes peuvent s'interpréter comme des tentatives de résolution du problème :

$$\max_{X \in Co(\mathcal{W})} \|X\|$$

par des méthodes de gradients projetés par exemple.

La première chose à noter est qu'il n'est pas forcément facile d'exprimer $Co(\mathcal{W})$ par un ensemble de contraintes simple (il ne suffit pas en tout cas de mettre des inégalités dans les contraintes (C)!).

La seconde chose, et la plus grave, est que toutes ces méthodes convergent vers des maxima locaux : il est de plus difficile de situer la solution trouvée par rapport au maximum global.

4.3. MATRICES DONNEES PAR $A = \sum_{i=1}^{p} B_i B_i^T$ $\quad p \leq n \quad B_i \in \mathbb{R}^n$

PROBLEME DE DISTANCE EUCLIDIENNE MAXIMALE DANS $\mathbb{R}^{pr}$

Soit A donnée par $A = \sum_{k=1}^{p} B_i B_i^T$

Calculons l'expression :

$$\sum_{k=1}^{r} V_k^T A V_k$$

$$\sum_{k=1}^{r} V_k^T \Big(\sum_{i=1}^{P} B_i B_i^T \Big) V_k = \sum_{k=1}^{r} \sum_{i=1}^{P} (B_i^T V_k)^2$$

Désignons par Y_V le point de $\mathbb{R}^{pr}$ de coordonnées :

$$B_1^T V_1, B_2^T V_1, \ldots B_p^T V_1, B_1^T V_2, B_2^T V_2, \ldots, B_r^T V_2, B_1^T V_r, \ldots, B_p^T V_r$$

Nous avons donc :

$$\sum_{k=1}^{r} V_k^T A V_k = Y_v^T Y_v = \|Y_v\|_2^2$$

Le problème posé en 4.1. est donc formellement équivalent à :

$$\operatorname{Max}_{v \in \mathcal{V}} \|Y_v\|_2$$

4.4. "MEILLEURE APPROXIMATION DE RANG p" D'UNE MATRICE SEMI-DEFINIE POSITIVE

4.4.1. Valeurs propres et vecteurs propres de matrices symétriques semi-définie positives [4]

A symétrique semi-définie positive possède n valeurs propres positives :

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$$

et une base orthonormée (pour φ_2) de vecteurs propres associés :

$$W_1, W_2, \ldots, W_n$$

A peut s'écrire sous la forme :

$$A = \sum_{i=1}^{n} \lambda_i W_i W_i^T = \sum_{i=1}^{n} C_i C_i^T$$

avec $C_i = \sqrt{\lambda_i}\, W_i \qquad i=1,2,\ldots,n.$

4.4.2. "Meilleure approximation" par des matrices données par

$$\sum_{i=1}^{P} B_i B_i^T \qquad \underline{p \text{ fixé}}$$

Les matrices $\sum_{i=1}^{P} B_i B_i^T$ sont des matrices symétriques semi-définie positives de rang inférieur ou égal à p .

Appelons S_p l'ensemble de ces matrices.

Théorème [5]

$$\min_{X \in S_p} S_{\varphi_2 \varphi_2}(A-X) = \lambda_{p+1}$$

et le minimum est atteint par

$$X^* = \sum_{i=1}^{P} C_i C_i^T$$

X^* étant de rang p nous l'appellerons "meilleure approximation de rang p" de A.

4.4.3. Calcul d'une "meilleure approximation de rang p".

Nos méthodes de partitionnement seront élaborées pour des matrices du type $\sum_{i=1}^{P} B_i B_i^T$ p étant faible, nous remplacerons une matrice A symétrique semi-définie positive par une meilleure approximation de rang p .

Le problème qui se pose est donc le calcul des p plus grandes valeurs propres d'une matrice A symétrique semi-définie positive et d'un système orthonormé de vecteurs propres associés ; ce calcul devra se faire pour des matrices de taille assez importante (n = 100 comme ordre de grandeur).

Nous utiliserons une méthode de calcul simultané de ces p valeurs propres et vecteurs propres due à Bauer [2] et dont des versions très efficaces pour le cas de matrices définies positives figurent dans [13][14].

4.4.4. Remarque : matrices des problèmes "à tailles égales"

Remarquons que si A est symétrique semi-définie positive toute matrice construite à partir de A pour un problème "à tailles égales" (paragraphe 2.3.) est aussi symétrique semi-définie positive et qu'une meilleure approximation de rang p $(p < n)$ se déduit immédiatement d'une meilleure approximation de rang p de A uniquement en rajoutant des composantes nulles aux vecteurs C_i .

V. PRINCIPE DE PROJECTION MAXIMUM POUR DES PROBLEMES DE DISTANCES EUCLIDIENNES MAXIMALES

Tous les problèmes exposés au paragraphe 4 peuvent se mettre sous la forme suivante :

Soit $\mathcal{D}$ un ensemble fini de $\mathbb{R}^m$

Soit M une matrice symétrique définie positive de $\mathcal{M}_{m,m}(\mathbb{R})$

Soit $\| \ \|_M$ la norme euclidienne sur $\mathbb{R}^m$ définie par :

$X \in \mathbb{R}^m$ $\qquad\qquad$ $\|X\|_M^2 = X^T M X$

Trouver $X^* \in \mathbb{R}^m$ vérifiant :

$$\|X^*\|_M = \max_{X \in \mathcal{D}} \|X\|_M$$

5.1. NOTATIONS

Soit $\mathcal{U}(m) = \{u \in \mathbb{R}^m \quad u^T u = +1\}$

$\mathcal{U}(m)$ désigne l'ensemble des vecteurs unitaires de $\mathbb{R}^m$.

On notera $\mathcal{U}^S(m)$ un sous-ensemble de $\mathbb{R}^m$ vérifiant :

$$\mathcal{U}^S(m) \subseteq \mathcal{U}(m) \qquad \mathcal{U}^S(m) \cup (-\mathcal{U}^S(m)) = \mathcal{U}(m) .$$

On notera $\mathcal{U}^+(m)$ le sous-ensemble de $\mathbb{R}^m$ défini par :

$$\mathcal{U}^+(m) = \{u \in \mathcal{U}(m) : \quad u(i) \geq 0 \quad i=1,2,\ldots,m\}$$

5.2. PRINCIPE DE PROJECTION MAXIMUM : CAS GENERAL

Il existe $u^* \in \mathcal{U}(m)$ et $\mathcal{J}$ voisinage de u^* dans $\mathcal{U}(m)$ tels que pour tout $u \in \mathcal{J}$ les problèmes :

$$\max_{X \in \mathcal{D}} X^T M u$$

aient une solution unique X^* commune vérifiant

$$X^{*T} M X^* = \max_{X \in \mathcal{D}} X^T M X$$

Démonstration

Soit X^* solution du problème :

$$\max_{X \in \mathcal{D}} X^T M X$$

Posons

$$u^* = X^* / \varphi_2(X^*)$$

Nous avons

$$X^T M X \leq X^{*T} A X^* \qquad \forall\, X \in \mathcal{D}$$

$$(X^*+X-X^*)^T M (X^*+X-X^*) \leq X^{*T} M X^{*T} \qquad \forall\, X \in \mathcal{D}$$

donc

$$(X-X^*)^T M (X-X^*) + 2X^{*T} M (X-X^*) \leq 0 \qquad \forall\, X \in \mathcal{D}$$

De plus $\mathcal{D}$ étant fini on peut trouver $\varepsilon > 0$ tel que :

$$(X-X^*)^T M (X-X^*) \geq \varepsilon > 0 \qquad \forall\, X \in \mathcal{D},\ X \neq X^*$$

Nous avons donc :

$$2\, X^{*T} M(X - X^*) \leq -\varepsilon \qquad \forall\, X \in \mathcal{D},\ X \neq X^*$$

$$X^{*T} MX = X^T MX^* \leq -\frac{\varepsilon}{2} + X^{*T} MX^*$$

et en divisant par $\varphi_2(X^*)$:

$$X^T (Mu^*) \leq -\eta + X^{*T} (Mu^*) \qquad \forall\, X \in \mathcal{D},\ X \neq X^*$$

$\mathcal{J}$ étant fini il existe un voisinage $\mathcal{J}$ de u^* dans $\mathcal{U}(m)$ tel que :

$$X^T (Mu^*) - \frac{\eta}{3} \leq X^T Mu \leq X^T (Mu^*) + \frac{\eta}{3} \qquad \forall\, X \in \mathcal{D},\ \forall\, u \in \mathcal{J}$$

donc :

$$X^T Mu \leq \frac{\eta}{3} + X^T Mu^* \leq \frac{\eta}{3} - \eta + X^{*T} Mu^* \leq \frac{\eta}{3} - \eta + \frac{\eta}{3} + X^{*T} Mu$$

Il en résulte que :

$$\left.\begin{array}{l} \forall\, X \in \mathcal{D},\ X \neq X^* \\ \forall\, u \in \mathcal{J} \end{array}\right\} \Rightarrow X^T Mu \leq -\frac{\eta}{3} + X^{*T} Mu$$

X^* est donc l'unique solution du problème

$$\max_{X \in \mathcal{D}} X^T Mu \qquad \forall\, u \in \mathcal{J}$$

5.3. PRINCIPE DE PROJECTION MAXIMUM : CAS SYMETRIQUE

Supposons que $\mathcal{D}$ soit symétrique par rapport à l'origine i.e.

$$X \in \mathcal{D} \Rightarrow -X \in \mathcal{D}$$

Nous pouvons énoncer le principe suivant :

Il existe $u^* \in \mathcal{U}^S(m)$ et $\mathcal{J}$ voisinage de u^* dans $\mathcal{U}^S(m)$ tel que pour tout $u \in \mathcal{J}$ les problèmes :

$$\underset{X \in \mathcal{D}}{\text{Max}} \; X^T M u$$

aient une solution unique X^* commune vérifiant :

$$X^{*T} M X^* = \underset{X \in \mathcal{D}}{\text{Max}} \; X^T M X$$

5.4. UTILISATION DU PRINCIPE PRECEDENT

Nos méthodes présentée plus loin reposeront toutes sur une "exploration" systématique mais guidée de $\mathcal{U}(m)$ et des problèmes $\underset{M \in \mathcal{D}}{\text{Max}} \; X^T M u$ qui dans certains cas seront extrèmement simples.

VI. METHODES A PRECISION RELATIVE ε POUR DES PROBLEMES DE DISTANCES EUCLIDIENNES MAXIMALES

6.1. DEFINITION

Nous choisirons la distance euclidienne usuelle.
Considérons le problème :

Trouver $X^* \in \mathcal{D}$ tel que $X^{*T} X^* = \underset{X \in \mathcal{D}}{\text{Max}} \; X^T X$

où $\mathcal{D}$ est un sous-ensemble de $\mathbb{R}^m$.

Nous supposons toujours savoir résoudre les problèmes :

Trouver $X_u \in \mathcal{D}$ tel que $X_u^T u = \underset{X \in \mathcal{D}}{\text{Max}} \; X^T u$

X_ε sera appelée solution à la précision relative si :

$$\frac{X^{*T} X^* - X_\varepsilon^T X_\varepsilon}{X_\varepsilon^T X_\varepsilon} \leq \varepsilon \quad \text{et} \quad X_\varepsilon \in \mathcal{D}$$

6.2. ENSEMBLES $\mathcal{U}(m)$, $\mathcal{U}^S_\varepsilon(m)$, $\mathcal{U}^+_\varepsilon(m)$

Par analogie avec les notations du paragraphe 5.1., nous adopterons les notations suivantes :

$\mathcal{U}_\varepsilon(m)$ désigne un sous-ensemble de $\mathcal{U}(m)$ vérifiant la propriété suivante :

$$\forall\, u \in \mathcal{U}(m) \quad \exists\, u_\varepsilon \in \mathcal{U}_\varepsilon(m) \;:\; u^T u_\varepsilon \geq \frac{1}{\sqrt{1+\varepsilon}}$$

$\mathcal{U}^S_\varepsilon(m)$ désigne un sous-ensemble de $\mathcal{U}(m)$ tel que $\mathcal{U}^S_\varepsilon(m) \cup (-\mathcal{U}^S_\varepsilon(m))$ soit un ensemble $\mathcal{U}_\varepsilon(m)$.

$\mathcal{U}^+_\varepsilon(m)$ désigne un sous-ensemble de $\mathcal{U}^+(m)$ vérifiant la propriété suivante :

$$\forall\, u \in \mathcal{U}^+(m) \quad \exists\, u^+_\varepsilon \in \mathcal{U}^+_\varepsilon(m) \;:\; u^T u^+_\varepsilon \geq \frac{1}{\sqrt{1+\varepsilon}}$$

6.3. THEOREME : CAS NON SYMETRIQUE

Soit un ensemble $\mathcal{U}_\varepsilon(m)$

Soient $X^*_\varepsilon \in \mathcal{D}$ et $u^*_\varepsilon \in \mathcal{U}_\varepsilon(m)$ vérifiant :

$$X^{*T}_\varepsilon\, u^*_\varepsilon = \max_{u \in \mathcal{U}_\varepsilon(m)} \; \max_{X \in \mathcal{D}} X^T u$$

Alors X^*_ε est une solution à la précision relative ε .

Démonstration

Soit X^* une solution du problème et soit le vecteur unitaire

$$u = \frac{X^*}{(X^{*T}X^*)^{1/2}}$$

Il existe $u_\varepsilon \in \mathcal{U}_\varepsilon(m)$ vérifiant :

$$u^T u_\varepsilon \geq \frac{1}{\sqrt{1+\varepsilon}}$$

Nous avons donc :

$$\frac{X_*^T \; u_\varepsilon}{(X^{*T}X^*)^{1/2}} \geq \frac{1}{\sqrt{1+\varepsilon}}$$

Mais $\quad X_*^T \; u_\varepsilon \leq \underset{X \in \mathcal{D}}{\text{Max}} \; X^T \; u_\varepsilon \leq X_\varepsilon^{*T} \; u_\varepsilon^*$

d'où $\quad \frac{1}{\sqrt{1+\varepsilon}} \leq \frac{X_\varepsilon^{*T} \; u_\varepsilon^*}{(X^{*T}X^*)^{1/2}}$

et $\quad \frac{1}{1+\varepsilon} \leq \frac{(X_\varepsilon^{*T} \; u_\varepsilon^*)^2}{(X^{*T}X^*)}$

de plus $\quad (X_\varepsilon^{*T} \; u_\varepsilon^*)^2 \leq X_\varepsilon^{*T} \; X_\varepsilon^*$

ce qui donne $\quad X^{*T}X^* \leq (1+\varepsilon) X_\varepsilon^{*T} \; X_\varepsilon^*$

$$\frac{X^{*T}X^* - X_\varepsilon^{*T}X_\varepsilon^*}{X_\varepsilon^{*T} \; X_\varepsilon^*} \leq \varepsilon$$

6.4. THEOREME : CAS SYMETRIQUE

Soit un ensemble $\mathcal{U}_\varepsilon^S(m)$ et $\mathcal{D}$ symétrique par rapport à l'origine.

Soient $X_\varepsilon^* \in \mathcal{D}$ et $u_\varepsilon^{S*} \in \mathcal{U}_\varepsilon(p)$ vérifiant :

$$X_\varepsilon^{*T} \quad u_\varepsilon^{S*} = \max_{u \in \mathcal{U}_\varepsilon^S(m)} \max_{X \in \mathcal{D}} X^T u$$

Alors X_ε^* est une solution à la précision relative .

Démonstration

Soit X^* une solution du problème et soit

$$u = \frac{X^*}{(X^{*T} X^*)^{1/2}}$$

Il existe u_ε ou $- u_\varepsilon \in \mathcal{U}_\varepsilon(m)$ tel que

$$u^T \quad u_\varepsilon \geq \frac{1}{\sqrt{1+\varepsilon}}$$

ou

$$u^T(- \quad u_\varepsilon) \geq \frac{1}{\sqrt{1+\varepsilon}}$$

ce qui peut s'écrire :

$$\frac{X^{*T} \quad u_\varepsilon}{(X^{*T} X^*)^{1/2}} \geq \frac{1}{\sqrt{1+\varepsilon}}$$

ou

$$\frac{(-X^{*T} \quad u_\varepsilon)}{(X^{*T} X^*)^{1/2}} = \frac{1}{\sqrt{1+\varepsilon}}$$

Mais $- X^* \in \mathcal{D}$.

Dans les deux cas, on a donc :

$$X^{*T} \quad u_\varepsilon \leq \max_{X \in \mathcal{D}} X^T \quad u_\varepsilon \qquad (-X^*)^T \quad u_\varepsilon \leq \max_{X \in \mathcal{D}} X^T \quad u_\varepsilon$$

la suite de la démonstration est alors identique au théorème 4.3.

6.5. DIMINUTION A POSTERIORI DE L'ERREUR RELATIVE

Si on reprend les démosntrations des théorèmes 4.3. et 4.4. on remarque que :

$$X^{*T}X^{*} \leq (1+\varepsilon)(X_{\varepsilon}^{*T} \quad u_{\varepsilon}^{*})^2$$

On peut poser :

$$(X_{\varepsilon}^{*T} \quad u_{\varepsilon}^{*})^2 = \frac{1}{(1+\eta)} X_{\varepsilon}^{*T} X_{\varepsilon}^{*} \qquad \eta > 0$$

L'égalité devient finalement :

$$X^{*T}X^{*} \leq \frac{1+\varepsilon}{1+\eta} \; X_{\varepsilon}^{*T}X_{\varepsilon}^{*}$$

d'où

$$\frac{X^{*T}X^{*} - X_{\varepsilon}^{*T}X_{\varepsilon}^{*}}{X_{\varepsilon}^{*T} \; X_{\varepsilon}^{*T}} \leq \frac{\varepsilon-\eta}{1+\eta} < \varepsilon$$

Notons que l'on a forcément $\eta \leq \varepsilon$ sinon l'inégalité

$X^{*T} \; X^{*} \leq \frac{1+\varepsilon}{1+\eta} \; X_{\varepsilon}^{*T}X_{\varepsilon}^{*}$ donnerait

$$X^{*T}X^{*} < X_{\varepsilon}^{*T}X_{\varepsilon}^{*}$$

6.6. ALGORITHMES D'ENUMERATION

Les algorithmes d'énumération consisteront à résoudre tous les problèmes $\underset{X \in \mathcal{D}}{\text{Max}} \; X^{T}u_{\varepsilon}$ et a retenir X_{ε}^{*} (et u_{ε}^{*}) tels que :

$$X_{\varepsilon}^{*T} \quad u_{\varepsilon}^{*} = \underset{u \in \mathcal{U}_{\varepsilon}(m)}{\text{Max}} \; \underset{X \in \mathcal{D}}{\text{Max}} \; X^{T}u$$

Pour qu'une telle énumération soit réaliste il faut :

1/ que les problèmes projetés soient "faciles" à résoudre : nous verrons qu'il existe des cas où cette condition est réalisée;

2/ que les ensembles $\mathcal{U}_\varepsilon(m)$ ne possède qu'un nombre relativement restreint d'éléments.

C'est cette dernière question que nous aborderons en 6.7.

6.7. CONSTRUCTION D'ENSEMBLES $\mathcal{U}_\varepsilon(m)$ ET $\mathcal{U}_\varepsilon^S(m)$

6.7.1. m = 1 ; m = 2

Dans le cas de m = 1 l'ensemble $\mathcal{U}(1)$ se réduit aux deux vecteurs (1) (-1) . Nous prendrons donc :

$$\mathcal{U}_\varepsilon(1) = \{(1),(-1)\} \qquad \mathcal{U}_\varepsilon^S(1) = \{(1)\}.$$

Cas non symétrique $\mathcal{U}_\varepsilon(2)$

Divisons l'intervalle $[0,2\pi]$ en n $(n \geq 2)$ intervalles égaux et posons :

$$\theta_i = \frac{2\pi}{n} \times i \qquad i=0,1,\ldots,n-1$$

$$u_i = \begin{pmatrix} \cos\ \theta_i \\ \sin\ \theta_i \end{pmatrix} \qquad i=0,1,\ldots,n-1$$

Tout vecteur de $\mathcal{U}(2)$ peut s'écrire

$$u = \begin{pmatrix} \cos\ \theta \\ \sin\ \theta \end{pmatrix} \qquad \theta \in [-\frac{\pi}{n}\ ,\ 2\pi - \frac{\pi}{n}[$$

Nous pouvons alors trouver θ_i tel que :

$$|\theta - \theta_i| \leq \frac{\pi}{n}$$

Nous avons $u^T u_i = \cos\theta\ .\ \cos\theta_i + \sin\theta\ \sin\theta_i = \cos(\theta - \theta_i)$

$$u^T u_i \geq \cos\frac{\pi}{n}\ .$$

Si nous choisissons n tel que $\cos \frac{\pi}{n} \geq \frac{1}{\sqrt{1+\varepsilon}}$

nous pourrons prendre :

$$\mathcal{U}_\varepsilon(2) = \{u_i \qquad i=0,1,\ldots,n-1\}$$

Pour cela il nous faut prendre :

$$\frac{\pi}{n} \leq \text{arctg } \sqrt{\varepsilon}$$

$$n \geq \pi / \text{ Arctg } \sqrt{\varepsilon}$$

Choix réalisé si $n = \left\lceil \frac{\pi}{\text{Arctg } \sqrt{\varepsilon}} \right\rceil$

où $\lceil x \rceil$ désigne l'entier tel que : $\lceil x \rceil - 1 < x \leq \lceil x \rceil$

Cas symétrique $\mathcal{U}^S_\varepsilon(2)$

Divisons l'intervalle $[0,\pi]$ en n intervalles égaux et posons :

$$\theta_i = \frac{\pi}{n} i \qquad i=0,1,\ldots,n-1$$

$$u_i = \begin{pmatrix} \cos \theta_i \\ \sin \theta_i \end{pmatrix} \qquad i=0,1,\ldots,n-1$$

u étant un vecteur de $\mathcal{U}(2)$ u ou $-u$ peut se mettre sous la forme :

$$\begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} \qquad \theta \in [-\tfrac{\pi}{2n}, \pi - \tfrac{\pi}{2n}[$$

Nous pouvons alors trouver θ_i tel que :

$$|\theta - \theta_i| \leq \frac{\pi}{2n}$$

Un raisonnement identique a celui fait ci-dessus montre que l'on peut choisir :

$$\mathcal{U}^S_\varepsilon(2) = \{u_i \qquad i=0,1,\ldots,n-1\}$$

a condition que : $\cos \frac{\pi}{2n} \geq \frac{1}{\sqrt{1+\varepsilon}}$

condition réalisée si : $n = \left\lceil \frac{\pi}{2 \text{ Arctg } \sqrt{\varepsilon}} \right\rceil$

Premier quadrant de $\mathbb{R}^2$: $\mathcal{U}_\varepsilon^+(2)$

Nous aurons besoin pour la suite de ce type d'ensemble.

Nous divisons l'intervalle $[0, \frac{\pi}{2}]$ en n intervalles égaux et nous posons :

$$\theta_i = \frac{\pi}{2n} i \qquad i=0,1,\ldots,n$$

$$u_i = \begin{pmatrix} \cos \theta_i \\ \sin \theta_i \end{pmatrix} \qquad i=0,1,\ldots,n$$

Tout vecteur de $^+(2)$ peut s'écrire :

$$u = \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} \qquad 0 \leq \theta \leq \frac{\pi}{2}$$

et nous pouvons alors trouver θ_i tel que :

$$|\theta - \theta_i| < \frac{\pi}{4n}$$

Nous choisirons donc :

$$\mathcal{U}_\varepsilon^+(2) = \{u_i \qquad i=0,1,\ldots,n\}$$

avec $\cos \frac{\pi}{4n} \geq \frac{1}{\sqrt{1+\varepsilon}}$

condition réalisée si :

$$n = \left\lceil \frac{\pi}{4 \text{ Arctg } \sqrt{\varepsilon}} \right\rceil \quad .$$

6.7.2. Construction récurrente d'ensemble $\mathcal{U}_\varepsilon(m)$

La construction d'ensembles $\mathcal{U}_\varepsilon(m)$ est capitale pour la résolution des problèmes à la précision relative ε . Il serait souhaitable de construire l'ensemble $\mathcal{U}_\varepsilon(m)$ possédant le moins d'éléments mais ce problème semble difficile.

Nous allons indiquer une méthode qui permettra de construire les ensembles $\mathcal{U}_\varepsilon(m)$ de manière récurrente.

$\underline{m = m_1 + m_2}$

Nous pouvons écrire tout vecteur de $\mathcal{U}(m)$ sous la forme :

$$u = \begin{pmatrix} \alpha^1 & u^1 \\ \alpha^2 & u^2 \end{pmatrix}$$

$$u^1 \in \mathcal{U}(m_1)$$
$$u^2 \in \mathcal{U}(m_2)$$
$$\begin{pmatrix} \alpha^1 \\ \alpha^2 \end{pmatrix} \in \mathcal{U}^+(2)$$

Soient $\mathcal{U}_a(m_1)$, $\mathcal{U}_b(m_2)$ et $\mathcal{U}_c^+(2)$. Nous pouvons trouver trois vecteurs appartenant respectivement aux ensembles précédents et tel que :

$$u^{1T} u_a^1 \geq \frac{1}{\sqrt{1+a}}$$

$$u^{2T} u_b^2 \geq \frac{1}{\sqrt{1+b}}$$

$$\alpha^1 \alpha_c^1 + \alpha^2 \alpha_c^2 \geq \frac{1}{\sqrt{1+c}}$$

Nous avons alors :

$$u^T \begin{pmatrix} \alpha_c^1 u_a^1 \\ \alpha_c^2 u_b^2 \end{pmatrix} = \alpha^1 \alpha_c^1 \, u^{1T} u_a^1 + \alpha^1 \alpha_c^2 \, u^{2T} u_b^2$$

Pour simplifier nous prendrons : $a = b = c = \eta$.

Le produit scalaire précédent sera minoré par :

$$\alpha^1 \alpha_c^1 \frac{1}{\sqrt{1+\eta}} + \alpha^2 \alpha_c^2 \frac{1}{\sqrt{1+\eta}} \geq \frac{1}{1+\eta}$$

Si nous voulons que l'ensemble des vecteurs

$$\begin{pmatrix} \alpha_\eta^1 & u_\eta^1 \\ \alpha_\eta^2 & u_\eta^2 \end{pmatrix} \qquad \begin{array}{l} u_\eta^1 \in \mathcal{U}_\eta(m_1) \\ u_\eta^2 \in \mathcal{U}_\eta(m_2) \\ \begin{pmatrix} \alpha_\eta^1 \\ \alpha_\eta^2 \end{pmatrix} \in \mathcal{U}_\eta^T(2) \end{array}$$

soit un ensemble $\mathcal{U}_\varepsilon(m)$ il suffira de choisir η tel que :

$$\frac{1}{1+\eta} \geq \frac{1}{\sqrt{1+\varepsilon}}$$

$$\eta \leq \sqrt{1+\varepsilon} - 1 \ .$$

Construction du $\mathcal{U}_\varepsilon(m)$

En utilisant la construction précédente avec la convention suivante :

$m = 2q$	$m_1 = m_2 = q$
$m = 2q+1$	$m_1 = q \qquad m_2 = q+1$

Nous remarquerons qu'il est possible de construire $\mathcal{U}_\varepsilon(m)$ par récurrence en utilisant des ensembles $\mathcal{U}_{\varepsilon'}(2)$, $\mathcal{U}_{\varepsilon'}(1)$ et $\mathcal{U}_{\varepsilon'}^+(2)$ construits en 6.7.1.

Nombre d'éléments de $\mathcal{U}_\varepsilon(m)$

Nous noterons $N(m,\varepsilon)$ le nombre d'éléments de $\mathcal{U}_\varepsilon(m)$ construit de la manière précédente.

Nous noterons de même $N^+(2,\varepsilon)$ le nombre d'éléments de $\mathcal{U}_\varepsilon^+(2)$.

Nous posons $\eta = \sqrt{1+\varepsilon} - 1$

Si aucune des composantes de $\begin{pmatrix}\alpha^1_\eta\\ \alpha^2_\eta\end{pmatrix}$ n'est nulle nous aurons pour chaque valeurs de $\begin{pmatrix}\alpha^1_\eta\\ \alpha^2_\eta\end{pmatrix}$:

$N(m_1,\eta) \times N(m_2,\eta)$ vecteurs différents.

Les deux vecteurs $\begin{pmatrix}0\\1\end{pmatrix}$ et $\begin{pmatrix}1\\0\end{pmatrix}$ de $\mathcal{U}^+_\eta(2)$ donnent respectivement $N(m_2,\eta)$ et $N(m_1,\eta)$ vecteurs différents.

Nous avons donc la relation :

$$N(m_1+m_2,\varepsilon) = [N^+(2,\eta)-2] \times N(m_1,\eta) \times N(m_2,\eta) + N(m_1,\eta) + N(m_2,\eta)$$

$$\eta = \sqrt{1+\varepsilon} - 1$$

6.7.3. Construction récurrente d'ensembles $\mathcal{U}^S_\varepsilon(m)$

La méthode est exactement la même et basé sur le fait que l'on peut construire un ensemble $\mathcal{U}^S_\varepsilon(m)$ avec $m_1 + m_2 = m$ en considérant les vecteurs :

$$\begin{pmatrix}\alpha^1_\eta & u^1_\eta\\ \alpha^2_\eta & u^2_\eta\end{pmatrix}$$

$$u^1_\eta \in \mathcal{U}^S_\eta(m_1)$$
$$u^2_\eta \in \mathcal{U}_\eta(m_2)$$
$$\begin{pmatrix}\alpha^1_\eta\\ \alpha^2_\eta\end{pmatrix} \in \mathcal{U}^T_\eta(2)$$

avec $\eta = \sqrt{1+\varepsilon} - 1$.

Si $N^S(m,\varepsilon)$ désigne le nombre d'éléments de $\mathcal{U}^S_\varepsilon(m)$ nous aurons :

$$N^S(m_1+m_2,\varepsilon) = [N^+(2,\eta)-2] \times N^S(m_1,\eta)N(m_2,\eta) + N^S(m_1,\eta) + N(m_2,\eta)$$

6.7.4. Quelque valeurs de $N(m,\varepsilon)$ et $N^S(m,\varepsilon)$

Nous donnons quelques valeurs de $N(m,\varepsilon)$ et $N^S(m,\varepsilon)$. Nous noterons * lorsque ce nombre dépasse 10000.

$N(m,\varepsilon)$

m \ ε	0.1	0.5	1
2	11	6	5
3	107	44	20
4	705	80	48
5	*	476	188
6	*	2808	728
7	*	*	2186
8	*	*	6560
9	*	*	*

$N^S(m,\varepsilon)$

ε	0.1	0.5	1
2	6	3	2
3	58	26	11
4	383	44	27
5	5673	251	104
6	*	1483	404
7	*	6147	1214
8	*	*	3644
9	*	*	*

6.7.5. Précision relative effective donnée par les constructions précédentes

Prenons l'exemple de construction de $\mathcal{U}_\varepsilon^S(2)$.
La précision relative ε serait obtenue exactement si le découpage en n parties de $[0,\pi]$ était tel que :

$$n = \frac{\pi}{2 \operatorname{Arctg} \sqrt{\varepsilon}} .$$

Le fait de choisir $n = \left\lceil \frac{\pi}{2 \operatorname{Arctg} \sqrt{\varepsilon}} \right\rceil$ nous fournit donc une précision supérieure à celle demandée.

Exemple

Nous demandons = 0.1

$$n = \left| \frac{\pi}{2 \operatorname{Arctg} \sqrt{0.1}} \right| = 6$$

L'ensemble $\mathcal{U}^{S}_{\varepsilon'}(2)$ réellement construit en divisant $[0,\pi]$ en six intervalles correspond à :

$\varepsilon' = 0.07$

Malheureusement, cela ne signifie pas que l'on peut diminuer le nombre des éléments de $\mathcal{U}^{S}_{\varepsilon}(2)$ pour la précision demandée.

CHAPITRE III

BIPARTITIONNEMENT DE MATRICES

I. RAPPEL ET ELIMINATION DE CONTRAINTES

1.1. RAPPEL. PRESENTATION DU CHAPITRE

Compte tenu des remarques faites au chapitre II nous pouvons sans perdre de généralités étudier un problème de type IV (chapitre II, paragraphe 2.3.) avec une matrice définie positive.

Dans ce chapitre nous étudierons le cas du bipartionnement ($r=2$) mais en contrepartie nous ne supposerons plus que le partitionnement se fait en deux parties égales mais seulement en deux parties de tailles fixes.

Nous supposerons de plus que A est seulement semi définie positive.

Voici donc la forme de notre problème :

Soit $n = d_1+d_2$ $\quad n,d_1,d_2 \in \mathbb{N}^*$

Soit $A \in \mathcal{M}_{n,n}(\mathbb{R})$ symétrique semi-définie positive.

Soit $\mathcal{V}$ l'ensemble des couples $V = (V_1,V_2)$ V_1 et V_2 étant deux vecteurs de $\mathbb{R}^n$ vérifiant les contraintes :

$$(C)\begin{cases} \text{C1} & V_k(i) = \begin{cases} 0 \\ 1 \end{cases} & k=1,2 \\ & & i=1,2,\ldots,n \\ \text{C2} & \sum_{i=1}^{n} V_1(i) = d_1 & \sum_{i=1}^{n} V_2(i) = d_2 \\ \text{C3} & V_1(i) + V_2(i) = 1 & i=1,2,\ldots,n \end{cases}$$

Le problème posé sera de trouver $V^* = (V_1^*,V_2^*) \in \mathcal{V}$ tel que :

$$V_1^{*T}AV_1^* + V_2^{*T}AV_2^* = \max_{(V_1,V_2)\in \mathcal{V}} V_1^TAV_1 + V_2^TAV_2$$

1.2. ELIMINATION DE CONTRAINTES PAR PARAMETRISATION [3]

Soit U le vecteur de $\mathbb{R}^n$ défini par :

$$U(i) = +1 \qquad i=1,2,\ldots,n$$

La contrainte C3 se traduit par :

$$V_1 + V_2 = U$$

Nous poserons donc :

$$V_1 = \frac{1}{2}(U + K)$$

$$V_2 = \frac{1}{2}(U - K)$$

avec $K = V_1 - V_2$.

Les contraintes (C1) et (C2) sont globalement équivalentes à :

$$(C'1) \qquad K(i) = \pm 1 \qquad i=1,2,\ldots,n$$

$$(C'2) \qquad \sum_{i=1}^{n} K(i) = d_1 - d_2$$

Nous appellerons $\mathcal{K}$ l'ensemble des vecteurs K de $\mathbb{R}^n$ vérifiant les contraintes (C'1) et (C'2).
Evaluons la quantité :

$$V_1^T A V_1 + V_2^T A V_2$$

$$\frac{1}{4}(U+K)^T A(U+K) + \frac{1}{4}(U-K)^T A(U-K) = \frac{1}{2}\{U^T AU + K^T AK\}$$

Notre problème se traduira donc de la manière suivante :

Trouver $K^* \in \mathcal{K}$ tel que $K^{*T} AK^* = \underset{K \in \mathcal{K}}{\text{Max}}\ K^T AK$.

1.3. REMARQUE

Si $d_1 = d_2$

$\mathcal{K}$ est symétrique c'est-à-dire :

$$K \in \mathcal{K} \quad \Rightarrow \quad -K \in \mathcal{K}$$

1.4. NOTATIONS

Soit $K \in \mathcal{K}$. Nous noterons :

$$K_+ = \{i \mid 1 \leq i \leq n \qquad K(i) = +1\}$$

$$K_- = \{i \mid 1 \leq i \leq n \qquad K(i) = -1\}$$

D'après la définition de K : card $(K_+) = d_1$ card $(K_-) = d_2$.

II. MATRICES DONNEESPAR $A = \sum_{i=1}^{P} B_i B_i^T$ ET PRINCIPE DE MAXIMUM

2.1. PROBLEME DE MAXIMUM DANS $\mathbb{R}^P$

Soit $A \in \mathcal{M}_{n,n}(\mathbb{R})$ donnée par :

$$A = \sum_{i=1}^{P} B_i B_i^T$$

Nous avons :

$$K^T A K = \sum_{i=1}^{P} (K^T B_i)^2$$

Notons :

$$\mathcal{D} = \{ X \in \mathbb{R}^P \mid \begin{pmatrix} X(1) \\ X(2) \\ \vdots \\ X(p) \end{pmatrix} = \begin{pmatrix} K^T B_1 \\ K^T B_2 \\ \vdots \\ K^T B_p \end{pmatrix} \quad K \in \mathcal{K} \}$$

Soit φ_2 la norme euclidienne usuelle sur $\mathbb{R}^P$.

Si à chaque $X \in \mathcal{D}$ on sait associer $K \in \mathcal{K}$ tel que :

$$X(i) = K^T B_i \qquad i=1,2,\ldots,p$$

le problème 1.2. est équivalent à trouver $X^* \in \mathcal{D}$ vérifiant :

$$\varphi_2(X^*) = \underset{X \in \mathcal{D}}{\text{Max}}\ \varphi_2(X)$$

Il est évident que si $d_1 = d_2$, $\mathcal{K}$ et $\mathcal{D}$ sont alors symétriques par rapport à l'origine.

2.2. PRINCIPE DE PROJECTION MAXIMUM : CAS GENERAL

Reprenons le principe de projection maximum cité dans le chapitre II, paragraphe 5.

Soit $\mathcal{U}(p) = \{u \in \mathbb{R}^P : u^T u = +1\}$

2.2.1. Principe

Il existe $u^* \in \mathcal{U}(p)$ et un voisinage $\mathcal{J}$ de u^* dans $\mathcal{U}(p)$ tel que pour tout $u \in \mathcal{J}$ les problèmes :

$$\underset{X \in \mathcal{D}}{\text{Max}}\ X^T u$$

aient une solution unique X^* commune vérifiant

$$X^{*T} X^* = \underset{X \in \mathcal{D}}{\text{Max}}\ X^T X$$

2.2.2. Résolution du problème de projection maximum [4]

Nous allons voir que la résolution d'un problème

$$\underset{X \in \mathcal{D}}{\text{Max}}\ X^T u$$

est particulièrement simple.

Nous avons : $X^T u = \sum_{i=1}^{P} X(i)u(i)$

$$X^T u = \sum_{i=1}^{P} (K^T B_i)u(i)$$

$$X^T u = K^T \sum_{i=1}^{P} B_i u(i)$$

$$X^T u = K^T B \qquad B \in \mathbb{R}^n$$

Théorème

Soit $B \in \mathbb{R}^n$

$$K^{*T}B = \max_{K \in \mathcal{K}} K^T B \iff \forall\, i \in K_+^*\ ,\ \forall\, j \in K_-^* : B(i) \geq B(j)$$

Démonstration

Soit $U \in \mathbb{R}^n$ tel que $U(i) = +1 \quad i=1,2,\ldots,n.$

Remarquons d'abord que :

$$K^{*T}B = \max_{K \in \mathcal{K}} K^T B \iff \left(\frac{K^*+U}{2}\right)^T B = \max_{K \in \mathcal{K}} \left(\frac{K+U}{2}\right)^T B$$

et nous avons en posant $U_K = \frac{K+U}{2}$

$$U_K(i) = 1 \iff K(i) = 1$$

$$U_K(i) = 0 \iff K(i) = -1$$

* Supposons que $U_{K^*}^T B = \max_{K \in \mathcal{K}} U_K^T B$

Soient $i_o \in K_+^*$ et $j_o \in K_-^*$.

Considérons K' tel que :

$$K'(i) = K^*(i) \qquad i \neq i_o, j_o$$

$$K'(i_o) = K^*(j_o) = -1$$

$$K'(j_o) = K^*(i_o) = +1$$

Nous avons :

$$U^T_{K^*} B \geq U^T_{K'} B \ ;$$

ce qui se traduit par :

$$\sum_{i \in K^*_+} B(i) \geq \sum_{i \in K^*_+} B(i) - B(i_o) + B(j_o)$$

d'où $B(i_o) \geq B(j_o)$.

* Réciproquement supposons que :

$$\forall\, i \in K^*_+ \ , \ \forall\, j \in K^*_- \ ; \qquad B(i) \geq B(j)$$

$$U^T_{K^*} B - U^T_K B = \sum_{i \in K^*_+} B(i) - \sum_{i \in K_+} B(i)$$

Nous pouvons écrire :

$$K^*_+ = K^*_+ \cap (K_+ \cup K_-) = (K^*_+ \cap K_+) \cup (K^*_+ \cap K_-)$$

$$K_+ = K_+ \cap (K^*_+ \cup K^*_-) = (K_+ \cap K^*_+) \cup (K_+ \cap K^*_-)$$

et finalement :

$$U^T_{K^*} B - U^T_K B \geq \sum_{i \in K^*_+ \cap K_-} B(i) - \sum_{i \in K^*_- \cap K_+} B(i)$$

$K_+^* \cap K_-$ contient le même nombre d'indices (éventuellement 0) que $K_-^* \cap K_+$.
Chaque indice i appartenant à $K_+^* \cap K_-$ appartient donc à K_+^* et de même chaque indice $j \in K_-^* \cap K_+$ appartient à K_-^+ .
Nous avons donc :

$$U^T_{K^*} B - U^T_K B \geq 0 \quad .$$

La résolution d'un problème de projection maximum se réduit donc a un tri des n nombres B(i).

Ce tri peut même être qualifié de partiel en ce sens que seuls les ensembles K_+ et K_- sont demandés.

C'est en fait l'efficacité de certains algorithmes de tri qui permettra d'utiliser le principe de projection maximum [1] [2].

2.3. PRINCIPE DE PROJECTION MAXIMUM : CAS SYMETRIQUE

En reprenant les notations de II.5. nous avons :

$$\mathcal{U}^S(p) \subseteq \mathcal{U}(p) \qquad \mathcal{U}^S(p) \cup (-\mathcal{U}^S(p)) = \mathcal{U}(p)$$

Nous pouvons énoncer le même principe en remplaçant $\mathcal{U}(p)$ par $\mathcal{U}^S(p)$ si $\mathcal{D}$ est symétrique par rapport à l'origine.

Si $\mathcal{D}$ est symétrique par rapport à l'origine, il existe $u^* \in \mathcal{U}^S(p)$ et un voisinage $\mathcal{J}$ de u^* dans $\mathcal{U}^S(p)$ tel que pour tout $u \in \mathcal{J}$ les problèmes

$$\operatorname{Max}_{X \in \mathcal{D}} X^T u$$

aient une solution unique X^* commune vérifiant :

$$X^{*T} X^* = \operatorname{Max}_{X \in \mathcal{D}} X^T X \ .$$

2.4. RELATION : CAS SYMETRIQUE ET CAS GENERAL

Conformément à 2.2.2. le problème :

$$\max_{X \in \mathcal{D}} X^T u$$

se résoud par un tri des composantes d'un vecteur B.
Il est important de noter que le tri qui permet de résoudre $\max_{X \in \mathcal{D}} X^T(-u)$

est celui des composantes du vecteur $-B$.

Si nous faisons donc un tri complet des composantes de B nous avons la solution des deux problèmes

$$\max_{X \in \mathcal{D}} X^T u \quad \text{et} \quad \max_{X \in \mathcal{D}} X^T(-u)$$

Pour cette raison, il sera souvent utile de considérer $\mathcal{U}_{(p)}$ sous la forme :

$$\mathcal{U}_{(p)} = \mathcal{U}^S_{(p)} \cup (-\mathcal{U}^S_{(p)})$$

2.5. METHODES EXACTES ET APPROCHEES

Nous allons exposer des algorithmes permettant de trouver une solution exacte pour p très faible. Le cas $p = 1$ étant trivial nous nous attarderons au cas $p = 2$, compte-tenu de certaines applications intéressantes de ce cas particulier que nous exposerons au chapitre VI.

Les principes élaborés permettraient de développer des algorithmes pour des valeurs de p légèrement plus grandes (3, 4 ?).

Cependant, ces problèmes intervenant comme des approximations de problèmes plus complexes nous étudierons pour des valeurs de p de l'ordre de 3, 4, 5, des méthodes à précision relative donnée très efficaces et dont les avantages seront la simplicité et la rapidité ce qui est primordial étant donnée la taille importante des problèmes traités.

III. METHODES EXACTES p = 1,2

3.1. p = 1

La matrice A est égale à $B_1 B_1^T$ $B_1 \in \mathbb{R}$.
Les seuls vecteurs unitaires de $\mathbb{R}$ étant (1) et son opposé (-1) il suffira de trouver K_1 et K_2 tels que :

$$K_1^T B_1 = \underset{K \in \mathcal{H}}{\text{Max}} \; K^T B_1$$

$$K_2^T (-B_1) = \underset{K \in \mathcal{H}}{\text{Max}} \; K^T (-B_1)$$

K^* sera alors celui des deux vecteurs trouvés qui maximise $(K^T B_1)^2$.

A noter que les deux opérations de base permettant de trouver K_1 et K_2 sont un même tri des composantes de B_1.

Nous verrons (chapitre IV, paragraphe 3.1.2.) quelle interprétation donner à de telles matrices.

3.2. p = 2

$$A = B_1 B_1^T + B_2 B_2^T \qquad [4]$$

Les notations sont celles introduites au paragraphe 2.2.1.

3.2.1. Description de $\mathcal{U}(2)$, $\mathcal{U}^S(2)$ et principe de projection maximum

Nous prendrons

$$\mathcal{U}(2) = \{ V_\theta = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \qquad \theta \in [0, 2\pi[\; \}$$

$$\mathcal{U}^S(2) = \{ V_\theta = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \qquad \theta \in [0, \pi[\; \}$$

Soit $\mathcal{D}$ ensemble fini de $\mathbb{R}^2$. Le principe de projection maximum peut prendre la forme suivante.

Il existe un intervalle ouvert $\mathcal{J} =]\theta_o, \theta_1[\subset [0,2\pi[$ tel que pour tout $\theta \in \mathcal{J}$ les problèmes :

$$\underset{X \in \mathcal{D}}{\text{Max}}\ X^T V_\theta$$

aient une solution unique X^* commune vérifiant

$$X^{*T} X^* = \underset{X \in \mathcal{D}}{\text{Max}}\ X^T X$$

Pour le cas symétrique il suffit de remplacer $[0,2\pi[$ par $[0,\pi[$.

C'est dans une exploration systématique des intervalles $[0,2\pi[$ ou $[0,\pi[$ que seront construits les algorithmes suivants.

3.2.2. Bornes

Notations

- V_{θ_1}, V_{θ_2} désigneront par la suite deux vecteurs unitaires de $\mathbb{R}^2$ avec :

$$\theta_1, \theta_2 \in [0,2\pi] \qquad 0 \leq \theta_2 - \theta_1 \leq \frac{\pi}{4}$$

par conséquent, $V_{\theta_1}^T V_{\theta_2} = V_{\theta_2}^T V_{\theta_1} = \cos(\theta_2 - \theta_1) > 0$.

- $y_1 = \underset{X \in \mathcal{D}}{\text{Max}}\ X^T V_{\theta_1}$ $\qquad y_2 = \underset{X \in \mathcal{D}}{\text{Max}}\ X^T V_{\theta_2}$

- $C_{\theta_1 \theta_2} = \{X \in \mathbb{R}^2 : X = \lambda V_{\theta_1} + \mu V_{\theta_2} \quad \lambda, \mu \geq 0\}$

Lemme 1

$$y_1 \text{ ou } y_2 < 0 \quad \Rightarrow \quad C_{\theta_1\theta_2} \cap \mathcal{D} = \emptyset$$

Par convention nous poserons :

$$\underset{X \in C_{\theta_1\theta_2} \cap \mathcal{D}}{\text{Max}}\ X^T X = 0 .$$

Démonstration

Si $X \in C_{\theta_1\theta_2} \cap \mathcal{D}$ il résulte des définitions que

$$X^T V_{\theta_1} \geq 0 \qquad \text{et} \qquad X^T V_{\theta_2} \geq 0$$

ce qui est incompatible avec y_1 ou $y_2 < 0$.

Lemme 2

Si $X \in C_{\theta_2\theta_1} \cap \mathcal{D}$ et $y_1, y_2 \geq 0$

Alors :

$$(X^TX)^{1/2} \leq \text{Min} \left\{ \frac{(y_1^2+y_2^2-2y_1y_2\cos(\theta_2-\theta_1))^{1/2}}{\sin(\theta_2-\theta_1)}, \frac{y_1}{\cos(\theta_2-\theta_1)}, \frac{y_2}{\cos(\theta_2-\theta_1)} \right\}$$

Démonstration

Soit $X \in C_{\theta_1\theta_2}$:

$$X = \lambda V_{\theta_1} + \mu V_{\theta_2} \qquad \lambda , \mu \geq 0$$

Posons :

$$x_1 = X^T V_{\theta_1} = \lambda + \mu \cos(\theta_2-\theta_1)$$

$$x_2 = X^T V_{\theta_2} = \lambda \cos(\theta_2-\theta_1) + \mu .$$

(Notons que x_1 et x_2 sont positifs si λ et μ sont positifs).
Les expressions de λ et μ en fonction de x_1 et x_2 sont :

$$\lambda = \frac{x_1-x_2\cos(\theta_2-\theta_1)}{\sin^2(\theta_2-\theta_1)} \qquad \mu = \frac{x_2-x_1\cos(\theta_2-\theta_1)}{\sin^2(\theta_2-\theta_1)}$$

$\lambda \geq 0 \iff x_1-x_2\cos(\theta_2-\theta_1) \geq 0$

$\mu \geq 0 \iff x_2-x_1\cos(\theta_2-\theta_1) \geq 0$.

Nous avons :

$$X^T X = \lambda^2 + \mu^2 + 2\lambda\mu \cos(\theta_2-\theta_1)$$

ce qui donne en remplaçant λ et μ par leurs expressions en fonction de x_1 et x_2 :

$$X^T X = \frac{1}{\sin^2(\theta_2-\theta_1)} [x_1^2+x_2^2-2x_1x_2 \cos(\theta_2-\theta_1)]$$

Nous avons donc pour $X \in C_{\theta_1\theta_2}$:

$$X^T X = \frac{1}{\sin^2(\theta_2-\theta_1)} [x_1^2+x_2^2-2x_1x_2 \cos(\theta_2-\theta_1)]$$

$$x_1 \geq x_2 \cos(\theta_2-\theta_1) \quad ; \quad x_2 \geq x_1 \cos(\theta_2-\theta_1).$$

De plus, pour $X \in \mathcal{D}$ nous avons :

$$X^T V_{\theta_1} = x_1 \leq y_1 \qquad \text{et} \qquad X^T V_{\theta_2} = x_2 \leq y_2$$

Notons γ l'ensemble suivant :

$$\gamma = \left\{ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2 \ : \ \begin{array}{l} x_1 \geq x_2 \cos(\theta_2-\theta_1) \ , \ x_1 \leq y_1 \\ x_2 \geq x_1 \cos(\theta_2-\theta_1) \ , \ x_2 \leq y_2 \end{array} \right\}$$

Nous avons donc :

$$C_{\theta_1\theta_2} \cap \mathcal{D} \subset \gamma$$

Nous allons majorer $X^T X$ sur γ :

$$\operatorname*{Max}_{X \in C_{\theta_1\theta_2} \cap \mathcal{D}} X^T X \leq \operatorname*{Max}_{\binom{x_1}{x_2} \in \gamma} \frac{x_1^2+x_2^2-2x_1 x_2 \cos(\theta_1-\theta_2)}{\sin^2(\theta_2-\theta_1)}$$

(*) Posons :

$$y_1 = x_1 + f_1$$

$$y_2 = x_2 + f_2$$

$$y_1^2+y_2^2-2y_1y_2 \cos(\theta_2-\theta_1) = x_1^2+x_2^2-2x_1x_2 \cos(\theta_2-\theta_1)$$
$$+ 2f_1(x_1-x_2 \cos(\theta_2-\theta_1))$$
$$+ 2f_2(x_2-x_1 \cos(\theta_2-\theta_1))$$
$$+ f_1^2+f_2^2-2f_1f_2 \cos(\theta_2-\theta_1)$$

$$f_1^2+f_2^2-2f_1f_2 \cos(\theta_2-\theta_1) \geq 0 \qquad \forall f_1, f_2$$

De plus, si $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \gamma$:

$$f_1 \geq 0 \ , \ f_2 \geq 0 \qquad x_1-x_2 \cos(\theta_2-\theta_1) \geq 0$$

$$x_2-x_1 \cos(\theta_2-\theta_1) \geq 0$$

Donc :

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \gamma \Rightarrow x_1^2+x_2^2-2x_1x_2 \cos(\theta_2-\theta_1) \leq y_1^2+y_2^2-2y_1y_2 \cos(\theta_2-\theta_1)$$

et en fin de compte :

$$\max_{X \in C_{\theta_1\theta_2} \cap \mathcal{D}} X^TX \leq \frac{y_1^2+y_2^2-2y_1y_2 \cos(\theta_2-\theta_1)}{\sin^2(\theta_2-\theta_1)}$$

(**) Considérons maintenant la fonction de x_1 suivante :

$$f_{x_2}(x_1) = x_1^2+x_2^2 - 2x_1x_2 \cos(\theta_2-\theta_1)$$

$$\text{sur } \gamma_1 = \{x_1 \mid \begin{matrix} x_1 \geq x_2 \cos(\theta_2-\theta_1) \\ x_2 \geq x_1 \cos(\theta_2-\theta_1) \end{matrix}, \quad x_1 \leq y_1\}$$

x_2 vérifiant $x_2 \leq y_2$

$$f'_{x_2}(x_1) = 2x_1(x_1 - x_2 \cos(\theta_2-\theta_1)).$$

Sur γ_1, f_{x_2} est donc croissante et comme $x_1 \cos(\theta_2-\theta_1) \leq x_2$, on a :

$$f_{x_2}(x_1) \leq \frac{x_2^2}{\cos^2(\theta_2-\theta_1)} + x_2^2 - 2x_2^2 \qquad x_1 \in \gamma_1$$

$$f_{x_2}(x_1) \leq x_2^2 \frac{\sin^2(\theta_2-\theta_1)}{\cos^2(\theta_2-\theta_1)} \qquad x_1 \in \gamma_1$$

Mais : $$f_{x_2}(x_1) \leq y_2^2 \frac{\sin^2(\theta_2-\theta_1)}{\cos^2(\theta_2-\theta_1)}$$

indépendamment de $x_2 \leq y_2$.

On a donc :

$$\frac{x_1^2+x_2^2-2x_1x_2\cos^2(\theta_2-\theta_1)}{\sin^2(\theta_2-\theta_1)} \leq \frac{y_2^2}{\cos^2(\theta_2-\theta_1)} \qquad \binom{x_1}{x_2} \in \gamma$$

et finalement $$\operatorname{Max}_{X \in C_{\theta_2\theta_1} \cap \mathcal{D}} X^T X \leq \frac{y_1^2}{\cos^2(\theta_2-\theta_1)}$$

(***) On démontre de même $\underset{X \in C_{\theta_2\theta_1} \cap \mathcal{D}}{\text{Max}} X^T X \leq \dfrac{y_1^2}{\cos^2(\theta_2-\theta_1)}$.

La réunion des trois résultats de (*) , (**) , et (***) donne le lemme annoncé.

Interprétation géométrique et remarque

Posons : $\alpha = \dfrac{(y_1^2+y_2^2-2y_1y_2 \cos(\theta_2-\theta_1))^{1/2}}{\sin(\theta_2-\theta_1)}$

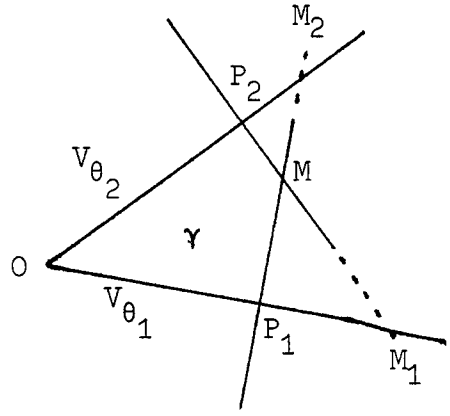
$$\beta_1 = \frac{y_1}{\cos(\theta_2-\theta_1)}$$

$$\beta_2 = \frac{y_2}{\cos(\theta_2-\theta_1)}$$

γ est l'ensemble du lemme précédent et sera hachuré sur les figures ci-dessous :

FIGURE 1

α est la meilleure borne sur γ .



On a en fait :

$\alpha = OM$

$\beta_1 = OM_2$

$\beta_2 1 = OM_1$

FIGURE 2

$\beta_1 = OM_2$ est la meilleure borne sur γ .

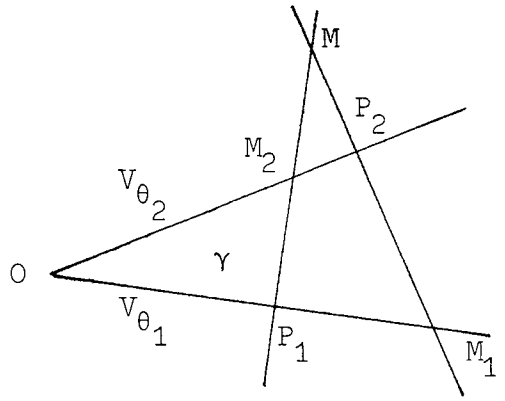
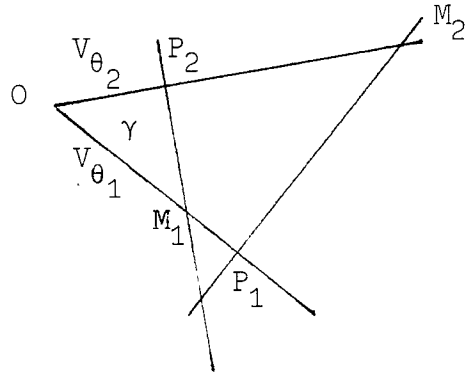


FIGURE 3

$\beta_2 = OM_1$ est la meilleure borne sur γ .



Remarque

La formule $$\frac{(y_1^2+y_2^2-2y_1y_2 \cos(\theta_2-\theta_1))^{1/2}}{\sin(\theta_2-\theta_1)}$$

peut poser des problèmes de calcul lorsque $\theta_2 \to \theta_1$.

Ceci nous a conduit à utiliser d'autres bornes présentées dans les lemmes suivants.

LEMME 3

$$X \in C_{\theta_2\theta_1} \cap \mathcal{D} \Rightarrow X^T X \leq \frac{(y_1+y_2)\ \mathrm{Max}\ (y_1,y_2)}{1+\cos(\theta_2-\theta_1)}$$

Démonstration

$$X \in C_{\theta_1\theta_2} \quad X = \lambda V_{\theta_1} + \mu V_{\theta_2} \qquad \lambda, \mu \geq 0$$

$$X^T X = \lambda X^T V_{\theta_1} + \mu X^T V_{\theta_2} \leq (\lambda+\mu) \text{ Max } \{y_1, y_2\} \text{ si } X \in \quad .$$

De plus :

$$X^T V_{\theta_1} = \lambda + \mu \cos (\theta_2-\theta_1)$$

$$X^T V_{\theta_2} = \lambda \cos (\theta_2-\theta_1) +$$

d'où $$(\lambda+\mu) = \frac{X^T V_{\theta_1} + X^T V_{\theta_2}}{1 + \cos (\theta_2-\theta_1)} \leq \frac{y_1+y_2}{1 + \cos (\theta_2-\theta_1)}$$

d'où $$X^T X \leq \frac{(y_1+y_2) \text{ Max } (y_1,y_2)}{1 + \cos (\theta_2-\theta_1)} \qquad X \in C_{\theta_1\theta_2} \cap \mathcal{D} \ .$$

LEMME 4

$$\text{Si } X \in C_{\theta_1\theta_2} \cap \mathcal{D} \quad \text{et} \quad y_1, y_2 \geq 0$$

$$(X^T X)^{1/2} \leq \text{Min} \left\{ \frac{(y_1+y_2) \text{ Max } (y_1,y_2)}{1 + \cos (\theta_2-\theta_1)}, \frac{y_1}{\cos (\theta_2-\theta_1)}, \frac{y_2}{\cos (\theta_2-\theta_1)} \right\}$$

Démonstration

Il suffit de remplacer la formule

$$(X^T X)^{1/2} \leq \frac{(y_1^2+y_2^2-2y_1y_2 \cos (\theta_2-\theta_1))^{1/2}}{\sin (\theta_2-\theta_1)}$$

par l'inégalité de lemme 3.

3.2.3. Algorithme de base : Cas général

Soit une suite $\{S_i\}$ de subdivisions de $[0,2\pi]$

$$S_i = \{\theta_o^i, \theta_1^i, \ldots, \theta_{n_i}^i\}$$

vérifiant les propriétés suivantes :

1/ $\theta_o^i = 0 \qquad \theta_{n_i}^i = 2\pi$

2/ $\theta_o^i < \theta_1^i < \ldots < \theta_{n_i}^i$

3/ $\theta_{j+1}^i - \theta_j^i \leq \pi/4$

4/ $S_{i+1} \supseteq S_i$

5/ $\underset{j}{\text{Max}} \, |\theta_{j+1}^i - \theta_j^i| \to 0 \quad (i \to +\infty)$

Description de l'étape i

Après avoir trouvé une solution a chaque problème

$$\underset{X \in \mathcal{D}}{\text{Max}} \; X^T V_{\theta_j^i} \qquad j=0,\ldots,n_i$$

nous noterons X_i^* la solution qui est de norme euclidienne $(X^TX)^{1/2}$ maximale.

μ_j^i sera un majorant de X^TX pour $X \in C_{\theta_j^i \theta_{j+1}^i} \cap \mathcal{D}$ calculé à l'aide des lemmes 1 et 2 (ou 1 et 4).

Nous noterons $M^i = \underset{j}{\text{Max}} \, \mu_j^i$

X^* désigne une solution du problème $\underset{X \in \mathcal{D}}{\text{Max}} \; X^TX$

THEOREME 1

Pour tout $\varepsilon > 0$ on peut déterminer i^* tel que :

$$\forall\ i \geq i^* \qquad X_i^{*T}\, X_i^* \geq X^{*T}\, X^* - \varepsilon$$

Démonstration

Dans le cas où $\underset{X \in \mathcal{D}}{\text{Max}}\ X^T V_{\theta_j^i}$ et $\underset{X \in \mathcal{D}}{\text{Max}}\ X^T V_{\theta_{j+1}^i}$ sont tous les deux positifs, nous avons (lemmes 2 ou 4) :

$$\mu_j^i \leq \frac{(\underset{X}{\text{Max}}\ X^T V_{\theta_j^i})^2}{\cos^2(\theta_{j+1}^i - \theta_j^i)} \leq \frac{X_i^{*T}\, X_i^*}{\cos^2(\theta_{j+1}^i - \theta_j^i)}$$

Si une des deux quantités précédente est négative l'inégalité reste vraie car nous avons posé par convention :

$$\mu_j^i = 0$$

Nous pouvons donc écrire :

$$M^i \leq \frac{X_i^{*T}\, X_i^*}{\underset{j}{\text{Min}}\ \cos^2(\theta_{j+1}^i - \theta_j^i)} = \frac{X_i^{*T}\, X_i^*}{\cos^2\ \underset{j}{\text{Max}}\ (\theta_{j+1}^i - \theta_j^i)}$$

Puisque $\theta_{j+1}^i - \theta_j^i \leq \pi/4$ nous avons :

$$M^i \leq \frac{X^{*T}\, X^*}{(1/\sqrt{2})^2} = 2\, X^{*T}\, X^*$$

Nous pouvons choisir i^* tel que :

$$i \geq i^* \qquad 2X^{*T}X^*\ \text{Sin}^2\ \underset{j}{\text{Max}}\ (\theta_{j+1}^i - \theta_j^i) \leq \varepsilon\ .$$

Pour $i \geq i^*$ nous avons donc :

$$M^i \left(1 - \text{Sin}^2 \max_j (\theta^i_{j+1} - \theta^i_j)\right) \leq X_i^{*T} X_i^*$$

$$M^i - \varepsilon \leq X_i^{*T} X_i^*$$

et donc $X^{*T} X^* - \varepsilon \leq X_i^{*T} X_i^*$.

Remarquons de plus que la suite $X_i^{*T} X_i^*$ est croissante ($S_{i+1} \supseteq S_i$) et donc que lorsque l'algorithme fournit au plus i^* une solution provisoire X_i^* telle que :

$$X^{*T} X^* - \varepsilon \leq X_i^{*T} X_i^*$$

l'inégalité reste vraie pour $i \geq i^*$.

THEOREME 2

Il existe i^* tel que :

$$\forall\, i \geq i^* \qquad X_i^{*T} X_i^* = X^{*T} X^*$$

Démonstration

$\mathcal{X}$ étant fini il existe $\varepsilon^* > 0$ tel que

$$X^{*T} X^* - \max_{\substack{X \in \mathcal{X} \\ X^T X \neq X^{*T} X^*}} X^T X = \varepsilon^*$$

il suffit donc d'appliquer le théorème précédent en prenant $\varepsilon < \varepsilon^*$.

3.2.4. Elimination de directions de projections : algorithme modifié

Soit $\varepsilon > 0$

Supposons qu'au début de l'étape i nous possédions :

. une solution provisoire X^*_{i-1}

. une partition de $\{0,1,2,\ldots,n_i-1\} = J_1^i \cup J_2^i$
vérifiant la propriété suivante :

$$M_2^i = \max_{X \in \left(\bigcup_{j \in J_2^i} C_{\theta_j^i \theta_{j+1}^i} \right) \cap \mathcal{D}} X^T X \leq X_{i-1}^{*T} X_{i-1}^{*T} + \varepsilon .$$

Description de l'étape i

. Si $J_1^i = \emptyset$, nous avons :

$$X^{*T} X^* \leq M_2^i \leq X_{i-1}^{*T} X_{i-1}^* + \varepsilon$$

On pose alors $X_i^* = X_{i-1}^*$ et $J_1^{i+1} = \emptyset$ et l'algorithme stationne.

. Si $J_1^i \neq \emptyset$, après avoir trouvée une solution à chaque problème

$$\max_{X \in \mathcal{D}} X^T V_{\theta_j^i} \qquad j \in J_1^i$$

$$\max_{X \in \mathcal{D}} X^T V_{\theta_{j+1}^i}$$

Nous noterons X_i la solution de norme euclidienne maximale.
Nous choisissons X_i^* celui des vecteurs X_i, X_{i-1}^* de norme euclidienne maximale.
Nous construisons J_1^{i+1} J_2^{i+1} comme cela sera indiquée par la suite.

THEOREME 3

Pour tout $\varepsilon > 0$ on peut déterminer i^* tel que :

$$\forall\ i \geq i^* \qquad X_i^{*T}\, X_i^* \geq X^{*T}\, X^* - \varepsilon$$

Démonstration

Elle est pratiquement identique à celle du théorème 1.
Nous pourrons choisir i_m tel que :

$$i \geq i_m \qquad 2X^{*T}\, X^* \sin^2 \operatorname{Max}_j \left(\theta_{j+1}^i - \theta_j^i\right) \leq \varepsilon \quad .$$

PREMIER CAS

Il est possible (et souhaitable) que l'on ait trouvée $i^* \leq i_m$ tel que $J_1^{i^*} = \emptyset$. L'algorithme stationne à partir de i^* avec la solution $X_{i^*}^*$.

DEUXIEME CAS

$$J_1^i \neq \emptyset \qquad\qquad i \leq i_m$$

Posons $M_1^{i_m} = \operatorname{Max}_{j \in J_1^{i_m}} \mu_j^{i_m}$

où $\mu_j^{i_m}$ désigne un majorant de $X^T X$ pour $X \in C_{\theta_j^{i_m}\ \theta_{j+1}^{i_m}} \cap \mathcal{D}$ calculé à l'aide des lemmes 1 et 2 (1 et 4).

Nous pouvons montrer comme dans la démonstration du théorème 1 :

$$M_1^{i_m} - \varepsilon \leq X_{i_m}^T\, X_{i_m}$$

$X^*_{i_m}$ vérifie donc :

$$M_1^{i_m} - \varepsilon \leq X^{*T}_{i_m} X^*_{i_m}$$

$$M_2^{i_m} - \varepsilon \leq X^{*T}_{i_m} X^*_{i_m}$$

donc : $X^{*T} X^* - \varepsilon \leq X^{*T}_{i_m} X^*_{i_m}$

La croissance de $X^{*T}_i X^*_i$ permet donc de conclure et de poser $i^* = i_m$.

Construction de J_1^{i+1} J_2^{i+1}

A la fin de l'étape i on calcule μ_j^i $j \in J_1^i$ et on note :

$$I_2^i = \{j \in J_1^i : \mu_j^i \leq X_i^{*T} X_i^* + \varepsilon \}$$

Nous prendrons alors :

$$J_2^{i+1} = \{j \in \{0,1,\ldots,n_{i+1}-1\} \mid [\theta_j^{i+1},\theta_i^{i+1}] \subseteq \bigcup_{j \in J_2^i \cup I_2^i} [\theta_j^i,\theta_{j+1}^i]\}$$

ce qui correspond à l'idée intuitive d'éliminer les secteurs où $\max_{X \in \mathcal{D}} X^T X$ est borné à la fin du pas i par $X_i^{*T} X_i^* + \varepsilon$.

Initialisation J_1^o J_2^o

Nous prendrons $J_2^o = \emptyset$ et X_{-1} n'a même pas besoin d'être défini.

3.2.5. Construction des ensembles S_i

Nous prendrons

$$S_o = \{0,\pi/4,\pi/2, 3\pi/4,\pi, 5\pi/4,3\pi/2,7\pi/4,2\pi\}$$

et on construira S_{i+1} en découpant chaque intervalle $[\theta_j^i,\theta_{j+1}^i]$ en p_{i+1} parties égales.

Nous essayerons les deux découpages suivants :

$$p_1 = 4 \quad , \quad p_2 = 4 \quad , \quad p_i = 2 \qquad i > 2$$

$$p_1 = 4 \quad , \quad p_2 = 2 \qquad i > 2$$

3.2.6. Cas symétrique

L'algorithme 3.2.4. se traduit dans le cas symétrique en remplaçant $[0,2\pi]$ par $[0,\pi]$.

Nous poserons $S_o = \{0,\pi/4,\pi/2,3\pi/4,\pi\}$

et nous essayerons les découpages :

$$p_1 = 4 \quad , \quad p_2 = 4 \quad , \quad p_i = 2 \qquad i \geq 2$$

$$p_1 = 4 \quad , \quad p_2 = 2 \qquad i \geq 2$$

Que donne l'algorithme ?

L'algorithme précédent donne X_i^* tel que :

$$X_i^{*T} X_i^* \geq X^{*T} X^* - \varepsilon \qquad i \geq i^*$$

Pour que la solution soit optimal il faut choisir correctement (suffisamment petit), ce qui peut présenter des difficultés.

3.3. RESULTATS NUMERIQUES DES ALGORITHMES 3.2.4., 3.2.6.

Nous prendrons des vecteurs B_1 et B_2 à composantes entières. La variation minimum de $(K^T B_1)^2 + (K^T B_2)^2$ est donc de 4 et en prenant $\varepsilon < 4$ nous sommes sûrs d'atteindre le maximum.

La méthode de tri utilisée est la méthode du tas ou Heapsort [1] [2] .

Nous donnons les temps d'exécution des algorithmes précédents en utilisant pour les mêmes exemples deux découpages différents (paragraphes 3.2.5. et 3.2.6.).

Cas de deux blocs égaux $d_1 = d_2$

Série 1

dimension		20	40	60	80	100
Temps d'exécution (1/100 s.)	Découpage 1	18	50	73	168	168
	Découpage 2	20	50	73	166	170

Série 2

dimension		20	40	60	80	100
Temps d'exécution (1/100 s.)	Découpage 1	23	21	38	71	130
	Découpage 2	25	21	40	73	135

Cas de deux blocs inégaux $d_1 \neq d_2$

Série 3

Dimensions	d_1	15	30	40	30	40
	d_2	5	10	20	50	60
Temps d'exécution (1/100 s.)	Découpage 1	31	60	145	101	175
	Découpage 2	36	66	160	110	191

IV. METHODE A LA PRECISION RELATIVE ε $\quad p << n$

4.1. METHODE

La méthode est celle exposée au chapitre II.6. appliquée au problème de distance euclidienne maximale du paragraphe 2.1. :

$$\underset{K\in\mathcal{K}}{\text{Max}} \sum_{i=1}^{P} (K^T B_i)^2$$

Elle consiste en une énumération des solutions des problèmes projetés :

$$\underset{K\in\mathcal{K}}{\text{Max}} \sum_{i=1}^{P} (K^T B_i) u(i)$$

le vecteur u décrivant un ensemble $\mathcal{U}_\varepsilon(p)$, ou $\mathcal{U}^S_\varepsilon(p)$ si K est symétrique.
Le vecteur K^ε solution du problème projeté de valeur maximum sera solution à ε-près du problème.

Conformément au paragraphe 2.2.2. la solution du problème projeté se déduit d'un tri du vecteur

$$\sum_{i=1}^{P} B_i\, u(i)$$

Ce tri étant identique pour u et $-u$ nous construirons des ensembles $\mathcal{U}_\varepsilon(p)$ de la manière suivante (paragraphe 2.4.) :

$$\mathcal{U}_\varepsilon(p) = \mathcal{U}^S_\varepsilon(p) \cup \{-\mathcal{U}^S_\varepsilon(p)\} .$$

4.2. PRECISION RELATIVE EFFECTIVE

Nous avons vu (chapitre II, paragraphe 6.7.5.) que la précision relative après construction d'un ensemble $\mathcal{U}_\varepsilon(p)$, $\mathcal{U}^S_\varepsilon(p)$ ou $\mathcal{U}^+_\varepsilon(p)$ était en fait meilleure que celle demandée.
Nous donnerons pour les exemples qui suivent la précision effective.

4.3. DIMINUTION A POSTERIORI DE L'ERREUR RELATIVE

Nous calculerons comme il a été indiqué au chapitre II, paragraphe 6.5. le facteur η de correction de l'erreur.

Rappelons que la précision relative à postériori est alors :

$$\frac{\varepsilon-\eta}{1+\eta}$$

si ε désigne la précision demandée (ou effective).

4.4. RESULTATS NUMERIQUES

La méthode de tri est toujours "Heapsort". Nous reprenons, pour comparer, quelques exemples du paragraphe 3.3. .

Nous donnons la précision relative réelle lorsque la solution nous est connue.

Exemples $p = 2$, $d_1 = d_2$ Exemples du paragraphe 3.3., Série 1

Précision demandée $\varepsilon = 0.1$ (Précision effective $\varepsilon = 0.07$)

Dimension	20	40	60	80	100
Temps d'exécution (1/100 sec.)	5	11	19	27	36
η	0.049	0.003	0.009	0.	0.015
Précision réelle	0.	0.006	0.	0.01	0.

Précision demandée $\varepsilon = 0.5$ (Précision effective 0.33)

Dimension	20	40	60	80	100
Temps d'exécution (1/100 sec.)	3	6	11	16	21
η	0.098	0.010	0.076	0.005	0.002
Précision réelle	0.	0.08	0.07	0.07	0.03

Exemples $p = 2$, $d_1 \neq d_2$ Exemples du paragraphe 3.3., Série 3

Précision demandée $\varepsilon = 0.1$ (Précision effective 0.07)

Dimension	d_1	15	30	40	30	40
	d_2	5	10	20	50	60
Temps d'exécution (1/100 sec.)		5	12	20	25	35
η		0.016	0.003	0.014	0.02	0.017
Précision réelle		0.	0.	0.008	0.	0.

Précision demandée $\varepsilon = 0.5$ (Précision effective $\varepsilon = 0.33$)

Dimension	d_1	15	30	40	30	40
	d_2	5	10	20	50	60
Temps d'exécution (1/100 sec.)		3	8	11	14	22
η		0.173	0.003	0.010	0.02	0.017
Précision réelle		0.	0.	0.07	0.	0.

Exemples $p = 4$, $d_1 = d_2$

Précision demandée $\varepsilon = 0.1$ (Précision effective $\varepsilon = 0.086$)

Dimension	16	48	100
Temps d'exécution (1/100 sec.)	211	763	1857
η	0.038	0.024	0.05

Précision demandée $\varepsilon = 0.5$ (Précision effective $\varepsilon = 0.37$)

Dimension	16	48	100
Temps d'exécution (1/100 sec.)	24	91	217
η	0.092	0.029	0.034

Précision demandée ε = 1 (Précision effective 0.56)

Dimension	16	48	100
Temps d'exécution (1/100 sec.)	15	58	134
η	0.154	0.015	0.067

Exemples $p = 4$, $d_1 \neq d_2$

Précision demandée ε = 0.1 (Précision effective ε = 0.086)

Dimension	d_1	12	32	60
	d_2	4	16	40
Temps d'exécution (1/100 sec.)		219	844	1927
η		0.019	0.032	0.012

Précision demandée ε = 0.5 (Précision effective ε = 0.37)

Dimension	d_1	12	32	60
	d_2	4	16	40
Temps d'exécution (1/100 sec.)		25	100	235
η		0.089	0.032	0.034

Précision demandée $\varepsilon = 1$ (Précision effective $\varepsilon = 0.56$)

Dimension	d_1	12	32	60
	d_2	4	16	40
Temps d'exécution (1/100 sec.)		15	64	155
η		0.005	0.032	0.067

V. BIPARTITIONNEMENT DE MATRICES SYMETRIQUES SEMI-DEFINIES POSITIVES UTILISANT UNE MEILLEURE APPROXIMATION DE RANG p

5.1. METHODE EMPLOYEE

Conformément au problème posé en 1.2. nous voulons trouver $K^* \in \mathcal{K}$ tel que :

$$K^{*T} A K^* = \operatorname{Max}_{K \in \mathcal{K}} K^T A K$$

Soit $\sum_{i=1}^{P} B_i B_i^T = \sum_{i=1}^{P} \lambda_i V_i V_i^T$

une meilleure approximation de rang p de la matrice A (chapitre II, paragraphe 4.4.).

En utilisant les méthodes exposées en 3 et 4 nous pouvons trouver une solution approchée ou exacte K_o du problème :

$$\operatorname{Max}_{K \in \mathcal{K}} K^T \sum_{i=1}^{P} B_i B_i^T K$$

K_o sera pris comme solution approchée du problème $\operatorname{Max}_{K \in \mathcal{K}} K^T A K$.

5.2. AMELIORATION LOCALE DE LA SOLUTION

Connaissant K_o solution approchée du problème

$$\underset{K \in \mathcal{H}}{\text{Max}} \quad K^T A K$$

Nous pourrons à partir de K_o appliquer des méthodes locales de maximisation du critère :

$$K^T A K$$

pour obtenir une solution approchée meilleure que K_o : par exemple une méthode d'échange (chapitre II, paragraphe 3.3.).

5.3. MAJORANTS

Notons d'abord l'existence de majorants liées aux valeurs propres de la matrice A concernant des problèmes de partitionnement plus généraux. Ces majorants seront étudiées plus précisément lors de l'étude du partitionnement en plusieurs blocs.

Soit A symétrique définie positive s'écrivant :

$$A = \sum_{i=1}^{P} \lambda_i V_i V_i^T + \sum_{i=p+1}^{n} \lambda_i V_i V_i^T$$

$\lambda_1, \lambda_2, \ldots, \lambda_n$ étant les valeurs propres en ordre décroissant et
$V_1, V_2, \ldots, V_n$ un système orthonormé de vecteurs propres.

Supposons que l'étude du problème :

$$\underset{K \in \mathcal{H}}{\text{Max}} \; K^T \left(\sum_{i=1}^{P} \lambda_i V_i V_i^T \right) K$$

nous fournise un majorant M tel que :

$$K^T \left(\sum \lambda_i V_i V_i^T \right) K \leq M \leq n \lambda_1 \qquad \forall K \in \mathcal{H}$$

$V_1, V_2, \ldots, V_n$ formant une base orthonormée nous avons :

$$\sum_{i=1}^{n} (K^T V_i)^2 = \|K\|_2^2 = n$$

Posons

$$\sum_{i=1}^{P} (K^T V_i)^2 = \alpha n \qquad 0 \leq \alpha \leq 1$$

Nous avons :

$$K^T A K = K^T \Big(\sum_{i=1}^{P} \lambda_i V_i V_i^T \Big) K + K^T \Big(\sum_{i=p+1}^{n} \lambda_i V_i V_i^T \Big) K$$

$$K^T A K \leq M + \lambda_{p+1} \sum_{i=p+1}^{n} (K^T V_i)^2$$

$$K^T A K \leq M + \lambda_{p+1} \; (1-\alpha) n$$

D'autre part :

$$K^T A K \leq \lambda_1 \sum_{i=1}^{P} (K^T V_i) + \lambda_{p+1} \sum_{i=p+1}^{n} (K^T V_i)^2$$

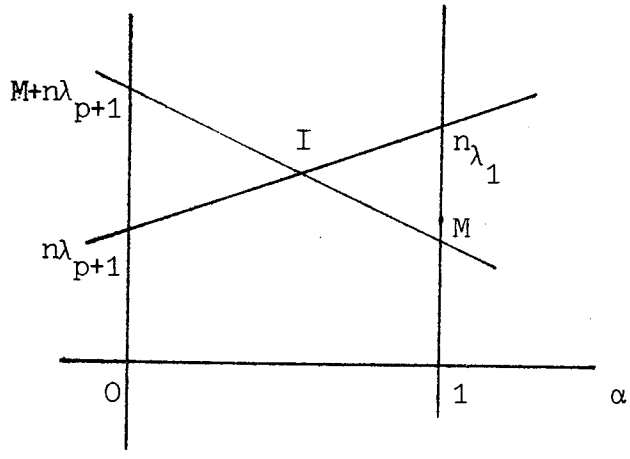
$$K^T A K \leq \lambda_1 \; \alpha n + \lambda_{p+1} \; (1-\alpha) \; n$$

$$K^T A K \leq \alpha n (\lambda_1 - \lambda_{p+1}) + n \lambda_{p+1}$$

Nous avons donc deux majorants de $K^T A K$ qui sont fonctions de α .

Représentons sur un graphique ces deux majorants :
la pente de la première droite est négative, celle de la seconde positive.
De plus nous avons :

$$M \leq n \lambda_1$$



Un autre majorant nous est donné par l'ordonnée du point I intersection des deux droites.

$$M + (1-\alpha)\ \lambda_{p+1}\ n = n\ \alpha\ (\lambda_1 - \lambda_{p+1}) + n\ \lambda_{p+1}$$

$$M + n\ \lambda_{p+1} - n\ \lambda_{p+1} = n\ \alpha\ (\lambda_1 - \lambda_{p+1} + \lambda_{p+1})$$

$$\alpha = \frac{M}{n\ \lambda_1}$$

L'ordonnée du point I est donc :

$$M + n\ (1 - \frac{M}{n\ \lambda_1})\ \lambda_{p+1}$$

$$M(1 - \frac{\lambda_{p+1}}{\lambda_1}) + n\ \lambda_{p+1}$$

5.4. RESULTATS NUMERIQUES

Nous allons donner quelques résultats numériques concernant un bipartitionnement en deux parties égales $(d_1 = d_2)$.

Nous considérerons quatre valeurs et vecteurs propres et nous utiliserons une méthode à précision relative $\varepsilon = 0.5$.

Le majorant sera celui obtenu au paragraphe 5.3. (nous remplacerons même λ_{p+1} par λ_p).

Nous donnons d'abord la valeur du critère $K^T(\ )K$ puis celle de $V_1^T(\ )V_1 + V_2^T(\ )V_2$.

Dimension	Critère K^T ( ) K			Critère V_1^T ( ) V_1 + V_2^T ( ) V_2		
	Valeur pour $\sum_{i=1}^{P} \lambda_i W_i W_i^T$	Majorant pour A	Valeur pour A	Valeur pour $\sum_{i=1}^{P} \lambda_i W_i W_i^T$	Majorant pour A	Valeur pour A
16	418.46	584.88	457	507.43	592.94	529
48	501.84	754.01	586.00	575.83	736.005	652.00
100	1563.65	3663.71	2971	2069	3375.35	3029.00

Commentaires

La méthode de bipartitionnement employée donne en général de bons résultats. Cela provient certainement du fait que les quatre vecteurs propres employés constituent une "information suffisante" sur la matrice comme en témoigne la concentration du critère sur la meilleure approximation de rang 4 (jusqu'à 95 %).

CHAPITRE IV

PARTITIONNEMENTS "A TAILLES EGALES" DE MATRICES

I. RAPPELS DES PROBLEMES

Soit $A \in \mathcal{M}_{n,n}(\mathbb{R})$ symétrique semi-définie-positive.

Soient $r, d \in \mathbb{N}$: $r \times d \geq n$

V désigne un ensemble de r vecteurs de $\mathbb{R}^n$

$V = (V_1, V_2, \ldots, V_r)$ vérifiant les contraintes :

$$(C) \begin{cases} (C1) \quad V_k(i) = \begin{cases} 0 \\ 1 \end{cases} & \begin{matrix} k=1,2,\ldots,r \\ i=1,2,\ldots,n \end{matrix} \\ (C2) \quad \sum_{i=1}^{n} V_k(i) \leq d & k=1,2,\ldots,r \\ (C3) \quad \sum_{k=1}^{r} V_k(i) = 1 & i=1,2,\ldots,r \end{cases}$$

$\mathcal{V}$ sera l'ensemble des systèmes de vecteurs V vérifiant (C).

Le problème posé est :

Trouver $V^* \in \mathcal{V}$; $V^* = (V_1^*, V_2^*, \ldots, V_r^*)$ tel que :

$$\sum_{k=1}^{r} V_k^{*T} A V_k^* = \underset{(V_1,V_2,\ldots,V_r)\in \mathcal{V}}{\text{Max}} \sum_{k=1}^{r} V_k^T A V_k$$

Rappelons aussi le problème "à tailles égales" caractérisé par :
$n = r \times d$ où la contrainte (C2) devient :

$$\sum_{i=1}^{n} V_k(i) = d \qquad k=1,2,\ldots,r$$

II. MAJORANTS

2.1. THEOREME DE HOFFMANN ET WIELANDT

Soient A et B deux matrices normales de $\mathcal{M}_{n,n}(\mathbb{C})$.

Soient : $\lambda_1,\lambda_2,\ldots,\lambda_n$ le spectre de A

$\lambda_1,\lambda_2,\ldots,\lambda_n$ le spectre de B

Soit $|||\ |||$ la norme sur $\mathcal{M}_{n,n}(\mathbb{C})$ définie par

$$|||A|||^2 = \sum_{i,j} |a_{ij}|^2$$

Il existe σ permutation de $\{1,2,\ldots,n\}$ tel que :

$$\sum_{i=1}^{n} |\lambda_i - \mu_{\sigma(i)}|^2 \leq |||A-B|||^2$$

Nous admettrons ce théorème dont la démonstration figure dans [2] .

2.2. THEOREME

Soient A et B deux matrices symétriques de $\mathcal{M}_{n,n}(\mathbf{R})$ semi-définies-positives.

Soient : $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$

$\mu_1 \geq \mu_2 \geq \ldots \geq \mu_n \geq 0$

les spectres réels et positifs de A et B .

On a l'inégalité suivante :

$$\sum_{i,j=1}^{n} A(i,j)B(i,j) \leq \sum_{i=1}^{n} \lambda_i \mu_i \ .$$

Démonstration

A et B étant symétriques elle sont normales. Nous pourrons donc appliquer le théorème 2.1. et il existe σ permutation de $\{1,2,\ldots,n\}$ tel que :

$$\sum_{i=1}^{n} (\lambda_i - \mu_{\sigma(i)})^2 \leq \sum_{i,j=1}^{n} (A(i,j)-B(i,j))^2$$

c'est-à-dire :

$$\sum_{i=1}^{n} \lambda_i^2 + \sum_{i=1}^{n} \mu_{\sigma^2(i)} - 2 \sum_{i=1}^{n} \lambda_i \mu_{\sigma(i)}$$

$$\leq \sum_{i,j=1}^{n} (A(i,j))^2 + \sum_{i,j=1}^{n} (B(i,j))^2 - 2 \sum_{i,j=1}^{n} A(i,j)B(i,j).$$

Remarquons que :

$$\sum_{j=1}^{n} (A(i,j))^2 = \sum_{j=1}^{n} A(i,j)A(j,i) = A^2(i,i)$$

Donc :

$$\sum_{i,j=1}^{n} (A(i,j))^2 = \sum_{i=1}^{n} A^2(i,i)$$

où $A^2 = A \times A$

ce qui peut s'écrire : $\sum_{i,j=1}^{n} (A(i,j))^2 = \text{trace } (A^2)$.

Or les valeurs propres de A^2 sont :

$$\lambda_1^2 \geq \lambda_2^2 \geq \ldots \geq \lambda_n^2$$

d'où :

$$\sum_{i,j=1}^{n} (A(i,j))^2 = \text{trace } A^2 = \sum_{i=1}^{n} \lambda_i^2 .$$

De même, on a :

$$\sum_{i,j=1}^{n} (B(i,j))^2 = \text{trace } B^2 = \sum_{i=1}^{n} \mu_i^2 = \sum_{i=1}^{n} \mu_{\sigma(i)}^2$$

L'inégalité fournie par le théorème de Hoffmann-Wielandt devient donc :

$$-2 \sum_{i=1}^{n} \lambda_i \mu_{\sigma(i)} \leq -2 \sum_{i,j=1}^{n} A(i,j)B(i,j)$$

ou :

$$\sum_{i,j=1}^{n} A(i,j)B(i,j) \geq \sum_{i=1}^{n} \lambda_i \mu_{\sigma(i)}$$

Or quelle que soit σ on a :

$$\sum_{i=1}^{n} \lambda_i \mu_{\sigma(i)} \geq \sum_{i=1}^{n} \lambda_i \mu_i$$

puisque toutes les valeurs propres sont positives et dans l'ordre décroissant.

2.3. VALEURS PROPRES ET MAJORANTS POUR LES PROBLEMES DE PARTITIONNEMENT [1]

Nous prenons le problème posé en I et nous supposerons que les valeurs propres de A sont :

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$$

Soit $q \in \mathbb{N}$ vérifiant : $n-d \leq qd < n$. On a $q < r$.

2.3.1. Propriété

$$\max_{(V_1,V_2,\ldots,V_k)\in \mathcal{V}} \sum_{k=1}^{r} V_k^T A V_k \leq d \sum_{k=1}^{q} \lambda_k + (n-qd)\lambda_{q+1}$$

Démonstration

On a : $V_k^T A V_k = \sum_{i=1}^{n} V_k(i) \left(\sum_{j=1}^{n} A(i,j) V_k(j) \right)$

$$V_k^T A V_k = \sum_{i=1}^{n} \sum_{j=1}^{n} A(i,j) V_k(i) V_k(j)$$

$$V_k^T A V_k = \sum_{i,j=1}^{n} A(i,j) (V_k V_k^T)(i,j)$$

D'où : $\sum_{k=1}^{r} V_k^T A V_k = \sum_{i,j} A(i,j) B(i,j)$

avec : $\sum_{k=1}^{r} V_k V_k^T = B$.

B est de rang r .

De plus, V_k (k=1,2,...,r) est un vecteur propre de B et on a :

$$B V_k = (V_k^T V_k) V_k$$

la valeur propre associée à V_k est donc :

$$V_k^T V_k = \sum_{i=1}^{n} V_k(i)$$

B possède donc r valeurs propres

$$\mu_1 \geq \mu_2 \geq \dots \mu_r \geq 0$$

vérifiant

$$\sum_{k=1}^{r} \mu_k = n \qquad \mu_k \leq d \quad ; \quad k=1,2,\dots,r$$

Le théorème 2.2. nous donne donc :

$$\sum_{k=1}^{r} V_k^T A V_k \leq \sum_{k=1}^{r} \lambda_k \mu_k$$

Soit $q \in \mathbb{N}$ défini par :

$$n-d \leq qd < n$$

considérons la quantité γ :

$$\gamma = d \sum_{k=1}^{q} \lambda_k + (n-qd)\,\lambda_{q+1}$$

$$\gamma - \sum_{k=1}^{r} \lambda_k \mu_k = \sum_{k=1}^{q} \lambda_k (d-\mu_k) + (n-qd)\lambda_{q+1} - \sum_{k=q+1}^{r} \lambda_k \mu_k$$

$$\gamma - \sum_{k=1}^{r} \lambda_k \mu_k \geq \lambda_{q+1} \sum_{k=1}^{q} (d-\mu_k) + (n-qd)\lambda_{q+1} - \Big(\sum_{k=q+1}^{r} \mu_k\Big)\lambda_{q+1}$$

$$\geq \lambda_{q+1}\Big[qd+n-qd - \sum_{k=1}^{r} \mu_k\Big]$$

d'où $$\gamma - \sum_{k=1}^{r} \lambda_k \mu_k \geq 0$$

Or γ ne dépend pas de $V = (V_1, V_2, \ldots, V_r)$ donc

$$\operatorname*{Max}_{(V_1,V_2,\ldots,V_r) \in \mathcal{V}} \sum_{k=1}^{r} V_k^T A V_k \leq \gamma$$

2.3.2. Propriété

$$\operatorname*{Max}_{(V_1,V_2,\ldots,V_r) \in \mathcal{V}} \sum_{k=1}^{r} V_k^T A V_k \leq d \sum_{i=1}^{m} \lambda_i + \lambda_m (n-md)$$

Si $m \leq q$

Démonstration

Il suffit de majorer λ_i par λ_m pour $i > m$

Cette inégalité est utile lorsque l'on connait seulement les m valeurs propres de plus grand module.

2.4. PROBLEMES APPROCHES LIES AUX VALEURS ET VECTEURS PROPRES ET MAJORANTS POUR LE PROBLEME DU PARTITIONNEMENT

2.4.1. PROBLEMES APPROCHES LIES AUX VECTEURS ET VALEURS PROPRES

A étant une matrice symétrique semi-définie-positive possédant les valeurs propres :

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$$

nous avons vu (chapitre III, paragraphe 5.1.2.) que :

$$A = \sum_{i=1}^{n} \lambda_i W_i W_i^T$$

où les W_i sont des vecteurs propres orthonormés associés aux λ_i .

Nous poserons

$$A_p = \sum_{i=1}^{P} \lambda_i W_i W_i^T$$

A_p étant la meilleure approximation de rang p .

Conformément à ce qui a été développé au chapitre III, paragraphe 5.2., nous utiliserons le problème approché :

$$\max_{(V_1,V_2,\ldots,V_r) \in \mathcal{V}} \sum_{k=1}^{r} V_k^T A_p V_k \quad .$$

2.4.2. Majorant utilisant le problème approché

Si $$\max_{(V_1,V_2,\ldots,V_r)\in \mathcal{V}} \sum_{k=1}^{r} V_k^T A_p V_k \leq M \leq n\lambda_1$$

alors $$\max_{(V_1,V_2,\ldots,V_r)\in \mathcal{V}} \sum_{k=1}^{r} V_k^T A V_k \leq M\left(1 - \frac{\lambda_{p+1}}{\lambda_1}\right) + n\lambda_{p+1}$$

Démonstration

Nous avons :

$$V_k^T \sum_{i=1}^{n} W_i W_i^T V_k = \sum_{i=1}^{n} (W_i^T V_k)^2 = \|V_k\|_2^2$$

donc :

$$\sum_{k=1}^{r} V_k^T \left(\sum_{i=1}^{n} W_i W_i^T\right) V_k = \sum_{k=1}^{r} \|V_k\|_2^2 = n$$

$$V_k^T \sum_{i=1}^{P} W_i W_i^T V_k = \sum_{i=1}^{P} (W_i^T V_k)^2 < \|V_k\|_2^2$$

Nous poserons :

$$\sum_{k=1}^{r} V_k^T \sum_{i=1}^{P} (W_i W_i^T) V_k = \delta n \qquad 0 \leq \delta \leq 1$$

Considérons maintenant :

$$\sum_{k=1}^{r} V_k^T A V_k = \sum_{k=1}^{r} V_k^T A_p V_k + \sum_{k=1}^{r} V_k^T \sum_{i=p+1}^{n} \lambda_i W_i W_i^T V_k$$

Nous pourrons écrire :

$$\sum_{k=1}^{r} V_k^T A V_k \leq M + \sum_{k=1}^{r} \sum_{i=p+1}^{n} \lambda_i (V_k^T W_i)^2$$

$$\leq M + \lambda_{p+1} \sum_{k=1}^{r} \sum_{i=p+1}^{n} (V_k^T W_i)^2$$

$$\leq M + \lambda_{p+1} (1 - \delta) n .$$

Nous pourons aussi écrire :

$$\sum_{k=1}^{r} V_k^T A V_k \leq \lambda_1 \delta n + \lambda_{p+1}(1 - \delta) n$$

Des deux inégalités précédentes dépendants de δ on déduit suivant la même démonstration que chapitre III, paragraphe 5.3., la majoration suivante :

$$\sum_{k=1}^{r} V_k^T A V_k \leq M \left(1 - \frac{\lambda_{p+1}}{\lambda_1}\right) + n \lambda_{p+1} \quad .$$

III. PROBLEMES "A TAILLES EGALES"

ET MATRICES DONNEES PAR $A = \sum_{i=1}^{P} B_i B_i^T$

3.1. MATRICES DU TYPE $A = B_1 B_1^T$

3.1.1. Propriété

A est une matrice de $\mathcal{M}_{n,n}(\mathbb{R})$ avec $n = r \times d$.
Soit σ une permutation ordonnant les éléments de B_1 dans un ordre décroissant :

$$B_1(\sigma(i)) \geq B_1(\sigma(j)) \qquad n \geq i > j \geq 1$$

Le système de vecteurs V_k^* défini par :

$$V_k^*(i) = \begin{cases} 1 \text{ si } i = \sigma(j) \text{ , } j \in \{(k-1)d+1, \ldots, kd\} \\ 0 \text{ sinon} \end{cases} \qquad k=1,2,\ldots,r$$

vérifie

$$\sum_{k=1}^{r} V_k^{*T} A V_k^* = \underset{(V_1,V_2,\ldots,V_k)\in \mathcal{V}}{\text{Max}} \sum_{k=1}^{r} V_k^T A V_k$$

Démonstration

Soit V^o un système de vecteurs $V_1^o, V_2^o, \ldots, V_r^o$ vérifiant :

$$\sum_{k=1}^{r} V_k^{oT} A V_k^o = \underset{(V_1,V_2,\ldots,V_r)\in \mathcal{V}}{\text{Max}} \sum_{k=1}^{r} V_k^T A V_k \quad .$$

Nous pouvons supposer sans perdre de généralité que si i_o est l'indice de la plus grande composante de B_1 alors :

$$V_1^o(i_o) = +1 \ .$$

(Il suffit de changer la numérotation des vecteurs V_k^o).

Considérons la quantité :

$$V_1^{oT} A V_1^o + V_k^{oT} A V_k^o \qquad k > 1$$

$$(V_1^{oT} B_1)^2 + (V_k^{oT} B_1)^2$$

Considérons les couples de vecteurs (V_1, V_k) vérifiant :

$$C' \begin{cases} V_1(i) = \begin{cases} 0 \\ 1 \end{cases} & i=1,2,\ldots,n \\ V_k(i) = \begin{cases} 0 \\ 1 \end{cases} & i=1,2,\ldots,n \\ \sum_{i=1}^{n} V_1(i) = \sum_{i=1}^{n} V_k(i) = d \\ V_1(i) + V_k(i) = V_1^o(i) + V_k^o(i) & i=1,2,\ldots,n \end{cases}$$

Il est évident que si le couple (V_1, V_k) vérifie ces contraintes le système $(V_1, V_2^o, \ldots, V_{k-1}^o, V_k, V_{k+1}^o, \ldots, V_r^o)$ vérifie toujours les contraintes associées au problème initial.

Nous devons donc avoir :

$$(V_1^{oT} B_1)^2 + (V_k^{oT} B_1)^2 = \text{Max}\{(V_i^T B_1)^2 + (V_k^T B_1)^2\}$$

V_1, V_k vérifiant C' .

Or ce problème est en fait un problème de bipartitionnement d'une matrice de $\mathcal{M}_{2d,2d}(\mathbb{R})$ égale à :

$$a = bb^T \quad .$$

b étant le vecteur déduit de B_1 en ne conservant que les indices i tels que :

$$V_1^o(i) = 1 \quad \text{ou} \quad V_k^o(i) = 1 \ .$$

Il s'agit donc d'un bipartitionnement de matrices du type $a = bb^T$.

Nous avons vu que la solution se ramenait à un tri du vecteur b . De plus comme nous avons fait l'hypothèse que le plus grand élément de B (et donc de b) était sélectionné par V_1^o on a :

$$\forall\, i \mid V_1^o(i) = +1$$
$$\forall\, j \mid V_k^o(j) = +1 \qquad B_1(i) \geq B_1(j)$$

Ceci étant vrai pour $k = 2,3,\ldots,r$ on a :

$$\forall\, i \mid V_1^o(i) = +1$$
$$\forall\, j \mid V_1^o(j) = 0 \qquad B_1(i) \geq B_1(j)$$

Ayant fixé un vecteur V_1^o remplissant les conditions précédentes il suffit d'appliquer le même résultat à un problème de taille n - d pour affirmer :

Il existe un système de vecteurs $(V_1^o, V_2^o, \ldots, V_r^o)$ vérifiant :

$$\forall i \mid V_{k_1}(i) = 1 \qquad \forall j \mid V_{k_2}(j) = 1 \qquad B_1(i) \geq B_1(j) \qquad \forall k_1, k_2 \quad 1 \leq k_2 < k_1 \leq r$$

et $$\sum_{k=1}^{r} (V_k^{oT} B_1)^2 = \max_{(V_1, V_2, \ldots, V_r) \in \mathcal{V}} \sum_{k=1}^{r} (V_k^T B_1)^2$$

Il suffit alors de constater que tout système de vecteurs de $\mathcal{V}$ vérifiant les premières contraintes donne une même valeur du critère pour achever la démonstration.

3.1.2. Matrices $B_1 B_1^T$ et cas probabiliste

Dans un cadre probabiliste on peut être amené à construire des matrices d'adresses jointives telles que :

$$A(i,j) = \alpha p_i p_j \qquad \alpha > 0$$

avec p_i = probabilité de référencer la donnée i .

Ce cas donne des matrices du type traité ci-dessus.

3.2. MATRICES DONNES PAR $A = \sum_{i=1}^{P} B_i B_i^T$

3.2.1. Remarque fondamentale

Contrairement au problème de bipartitionnement nous n'avons trouvé aucune méthode simple pour exhiber un problème de type tri ou même un problème d'équipartitionnement relatif à une matrice $C\ C^T$ de rang 1 ayant même solution que le problème initial.

Plus précisément nous verrons que l'application du principe de projection maximum nous conduit à des problèmes de programmation en nombres entiers généralisant d'une certaine manière le tri mais dont la résolution est loin d'être aussi simple.

3.2.2. Problème de distance euclidienne maximale dans $\mathbb{R}^{pr}$

Ses difficultés

Conformément à la présentation faite au chapitre II, paragraphe 4.3.2., nous avons :

$$\sum_{K=1}^{r} V_K^T A V_K = \sum_{k=1}^{r} \sum_{i=1}^{P} (B_i^T V_K)^2$$

et nous sommes ramenés à un problème de distance euclidienne maximale dans $\mathbb{R}^{pr}$.

Si nous voulons utiliser le principe de projection maximum nous serons amener à résoudre des problèmes du type suivant :

Soit un vecteur unitaire de $\mathbb{R}^{pr}$ donnée par rp composante $a_{i,k}$

$i=1,\ldots,p$

$k=1,\ldots,r$

$$\sum_{k=1}^{r} \sum_{i=1}^{P} a_{ik}^2 = 1 .$$

Le problème projeté sera du type suivant :

$$\max_{(V_1,V_2,\ldots,V_r)\in \mathcal{V}} \sum_{k=1}^{r} \sum_{i=1}^{P} a_{i,k} B_i^T V_k$$

$$\max_{(V_1,V_2,\ldots,V_r)\in \mathcal{V}} \sum_{k=1}^{r} V_k^T \left(\sum_{i=1}^{P} a_{i,k} B_i \right)$$

Deux constatations nous feront délaisser de telles méthodes :

1/ Le problème projeté est un problème (complexe) de programmation linéaire en nombres entiers (0,1).
Aucune paramétrisation ne peut le ramener à un problème de tri comme pour le bi-partitionnement.

2/ Le principe de projection maximum est appliqué à un espace $\mathbb{R}^{pr}$ de dimension élevé :

$$p = 2 \quad ; \quad r = 4 \quad ; \quad pr = 8 \ .$$

Bien qu'il soit possible de réduire la dimension à $p(r-1)$ il n'est pas possible de résoudre 1000 problèmes de programmation linéaire en nombres entiers du type précédent en un temps raisonnable.

3.2.3. Utilisation des seules matrices $B_i\ B_i^T$

Ayant résolu en 3.1. le problème d'équipartitionnement pour de telles matrices nous pouvons trouver p systèmes de vecteurs $(V_1^i,\ldots,V_r^i)$ vérifiant :

$$\sum_{k=1}^{r} (V_k^{iT} B_i)^2 = \max_{(V_1,V_2,\ldots,V_r)\in \mathcal{V}} \sum_{k=1}^{r} (V_k^T B_i)^2$$

Soit $(V_1^{i_o},V_2^{i_o},\ldots,V_r^{i_o})$ celui de ces systèmes de vecteurs vérifiant :

$$\sum_{k=1}^{r} (V_k^{i_oT} B_{i_o})^2 = \max_i \Sigma (V_k^{iT} B_i)^2$$

Propriété

$(V_1^{i_o},V_2^{i_o},\ldots,V_r^{i_o})$ est solution à la précision relative $p-1$ du problème d'équipartitionnement pour la matrice : $A = \sum_{i=1}^{P} B_i\ B_i^T$.

Démonstration

Elle est triviale ; soit $(V_1^*,V_2^*,\ldots,V_r^*)$ une solution du problème avec la matrice A .

$$\sum_{k=1}^{r} V_k^{*T} A V_k^* = \sum_{i=1}^{P} \sum_{k=1}^{r} (V_k^{*T} B_i)^2 \leq \sum_{i=1}^{P} \sum_{k=1}^{r} (V_k^{iT} B_i)^2$$

$$\sum_{k=1}^{r} V_k^{*T} A V_k^* \leq p \sum_{k=1}^{r} (V_k^{i_oT} B_{i_o})^2$$

$$\sum_{k=1}^{r} V_k^{*T} A V_k^{*} - \sum_{k=1}^{r} (V_k^{o} B_{i_o})^2 \leq (p-1) \sum_{k=1}^{r} (V_k^{o} B_{i_o})^2$$

Donc puisque

$$\sum_{k=1}^{r} (V_k^{o} B_{i_o})^2 \leq \sum_{k=1}^{r} V_k^{o} A V_k^{o}$$

$$\sum_{k=1}^{r} V_k^{*T} A V_k^{*} - \sum_{k=1}^{r} V_k^{o} A V_k^{o} \leq (p-1) \sum_{k=1}^{r} V_k^{o} A V_k^{o}$$

Nous pouvons d'ailleurs avoir une meilleure borne en prenant :

$$\sum_{i} \sum_{k} (V_k^{iT} B_i)^2$$

Diminution a posteriori de l'erreur relative

Nous pourons poser :

$$\sum_{k=1}^{r} (V_k^{o} B_{i_o})^2 = \frac{1}{1+\eta} \sum_{k=1}^{r} V_k^{o} A V_k^{o}$$

L'erreur relative a posteriori sera alors :

$$\frac{p-1-\eta}{1+\eta}$$

Exemple $p = 4$, $\eta = 1$

A priori l'erreur est : $p-1 = 3$

A posteriori l'erreur devient : $\frac{p-2}{2} = 1$

Avantages de telles méthodes

La rapidité due à la facilité de résolution des problèmes

$$\text{Max} \sum_{k=1}^{r} (V_k^{T} B_i)^2$$

La diminution a posteriori de l'erreur relative assez importante en général ramène la solution à des précisions fort acceptables.

L'utilisation d'améliorations locales (chapitre II, paragraphe 3.3.3., ou méthodes locales issues du bipartitionnement) permettent d'améliorer considérablement les solutions précédentes.

3.2.4. Méthodes "locales" issues du bipartitionnement

Possédant un système de vecteurs $(V_1,V_2,\ldots,V_r) \in \mathcal{V}$ nous pourons essayer de maximiser "localement" le critère

$$\sum_{k=1}^{r} V_k^T A V_k$$

de la manière suivante :

On sélectionne deux indices i et $j \in \{1,2,\ldots,r\}$ $i \neq j$ et on considère les éléments de $\mathcal{V}$ de la forme :

$$(V_1,V_2,\ldots,V_i',\ldots,V_j',\ldots,V_r)$$

(qui ne diffèrent que par V_i et V_j).

Le problème $$\max_{(V_1,V_2,\ldots,V_i',\ldots,V_j',\ldots,V_r)\in\mathcal{V}} \sum_{k\neq i,j} (V_k^T A V_k + V_i'^T A V_i + V_j'^T A V_j)$$

est un problème de bipartitionnement que l'on peut tenter de résoudre avec les méthodes du chapitre III.

3.3. MATRICES SYMETRIQUES SEMI-DEFINIES-POSITIVES ET RESULTATS NUMERIQUES

3.3.1. Méthode employée

Elle est identique a celle employée au chapitre III, paragraphe 5, et consiste a remplacer une matrice A semi-définie-positive par une meilleure approximation de rang p . (chapitre II, paragraphe 4.4.).

La résolution approchée du problème simplifié qui s'en suit nous fournit alors une approximation de la solution du problème initial relatif à la matrice A que l'on peut situer en utilisant les différentes bonnes données au paragraphe 2.

Nous pouvons enfin utiliser une méthode locale comme la méthode d'échange (chapitre II, paragraphe 3.3.3.)pour améliorer notre solution approchée.

3.3.2. Résultats numériques

Les quantités calculées seront $\sum_{k_1 \neq k_2} V_{k_1}^T ( ) V_{k_2}$.

Nous utiliserons quatre valeurs propres et vecteurs propres d'une matrice définie positive : A .

La solution trouvée sera construite de la manière suivante :

- recherche de la meilleure solution pour la matrice $\sum_{i=1}^{4} \lambda_i W_i W_i^T$ en utilisant les matrices de rang 1 , $B_i B_i^T$, $B_i = \sqrt{\lambda_i} W_i$ $(i=1,2,3,4)$ conformément au paragraphe 3.2.3.

- Amélioration locale du critère pour la matrice $\sum_{i=1}^{4} \lambda_i W_i W_i^T$ par une méthode "locale" issue du bipartitionnement (paragraphe 3.2.4).

Exemple 1 : n = 16 , d = 4 , r = 4

Majorant déduit du sous-problème	Majorant déduit des valeurs propres	Solution trouvée : valeur pour $\sum_{1}^{4} \lambda_i W_i W_i^T$	Solution trouvée : valeur pour A
605.92	498.00	372.36	461

Exemple 2 : n = 48 , d = 16 , r = 3

795.82	725.38	531.27	628

Exemple 3 : n = 48 , d = 8 , r = 6

795.82	629.43	376.80	556

Exemple 4 : n = 100 , d = 20 , r = 5

3738.33	3483.84	1516.26	2917

Exemple 5 : $n = 100$, $d = 10$, $r = 10$

3463.69	3298.14	1122.29	2757

Commentaires

Le premier commentaire est que la qualité de la méthode semble se dégrader quand augmente le nombre de blocs. Cela est certainement dû au fait que le nombre de vecteurs propres considérés (quatre dans les exemples ci-dessus) n'est pas assez important.

Ce fait se traduit aussi par un majorant déduit du sous-problème généralement moins bon que celui déduit du théorème d'Hoffmann et Wielandt.

Nous pouvons remarquer aussi que la valeur du critère est beaucoup moins concentrée sur les vecteurs propres considérés que lors du bipartitionnement.

En conclusion, notons que sur chaque exemple des considérations particulières permettent d'améliorer les solutions fournies par l'algorithme précédent.

CHAPITRE V

QUELQUES APPLICATIONS DES TECHNIQUES DE RESTRUCTURATION

I. NATURE DES APPLICATIONS POSSIBLES

1.1. RESTRUCTURATION ET "GRANDS" SYSTEMES

L'idée de restructuration pour une mémoire virtuelle paginée est probablement issue de constatations faites sur le mauvais comportement de gros programmes se déroulant dans un système à mémoire virtuelle. Ce gros programme , un compilateur par exemple, est en général découpé en un grand nombre de modules dont la répartition en mémoire est évidemment importante ; en effet certains de ces modules s'appellent très souvent entre eux et il est important que de tels modules soient dans une même page.

C'est dans cette optique qu'ont été faits les travaux de Hatfield, Gerald et Johnson dont nous avons parlé au chapitre I.

Remarquons cependant que, mis à part les études précédentes, relativement peu de travaux ont été faits sur le sujet précédent. Cela est certainement dû à l'énorme difficulté du problème. En effet, pour ce type d'applications, il faudrait, en plus de ce qui a été,ait au chapitre I, noter les difficultés supplémentaires suivantes :

- c'est le programme lui-même qui est restructuré,
- le déroulement du programme n'est connu que par des mesures statistiques,
- l'algorithme de remplacement est en général mal connu,
- la capacité en pages de la mémoire opératoire n'est pas constante en général si (et c'est le cas des exemples cités) on fonctionne en multi-programmation,
- la taille des "données" (les modules) n'est pas constante,

et on pourrait en citer d'autres.

Toutes ces constatations font qu'il est même très difficile de poser correctement le problème.

Notons cependant la qualité des résultats obtenus dans de si difficiles circonstances.

1.2. ADAPTATION D'ALGORITHMES A UNE MEMOIRE VIRTUELLE

Pour la résolution d'un problème précis, l'apparition des systèmes à pagination a suscité la recherche de nouvelles méthodes et de nouveaux algorithmes mieux adaptés.

L'optique adoptée est alors très différente de celle de la restructuration. En effet une modification de la structure de l'ensemble des données est rarement envisagée dans sa généralité mais des algorithmes totalement nouveaux sont élaborés.

Par exemple dès 1967 R.C. Singleton [9] étudiait un algorithme de "Transformation de Fourier rapide" en environnement paginé.

L'étude d'algorithmes de calculs matriciels a été faite dans cette optique ([6][7][8]) .

Bien que l'idée de changer l'organisation des données soit sous-jacente dans les travaux précédents, cela n'a pas été fait systématiquement en raison du cadre de travail que s'était fixé les auteurs précédents. En effet, si l'on étudie des opérations matricielles sur des matrices de $\mathcal{M}_{100,100}(\mathbb{R})$ le nombre de "données" considérées est de l'ordre de 10 000 or rien ne permet d'aborder de tels problèmes actuellement dans toute leur généralité.

1.3. MICRO-CALCULATEURS ET MINI-CALCULATEURS

Les contraintes de simplicité imposées par nos techniques de restructuration nous limitent donc à des problèmes beaucoup plus modestes si aucun regroupement préalable n'a lieu .

En particulier, la construction de micro-calculateurs très spécialisées, utilisant forcément des mémoires de hiérarchies différentes, pourrait certainement tirer profit de telles optimisations de programmes.

Dans un cadre voisin, celui de la micro-programmation, peut-être serait-il intéressant de procéder à des transferts groupés de données inspirés des techniques de mémoires virtuelles. Les procédés de restructuration pourraient alors être utilisés de manière à réduire le volume d'un

micro-programme par exemple.

Nous sommes d'ailleurs quelquefois très proche de préoccupations d'optimisation d'allocation de registres [1][4][8].

Les mini-calculateurs, dont certains répondent d'ailleurs assez fidèlement à la description donnée au chapitre I sont directement concernés par de telles méthodes.

En effet, si certains de ces calculateurs possèdent un répertoire d'instructions assez important le nombre de mémoires adressables est souvent fort limité.

Nous pourrions donc envisager un mode de transfert de données sur une mémoire extérieure : carte magnétique ou feuille de papier !

Il est alors important de réduire au minimum ce nombre de transferts et une restructuration de l'ensemble des données pourra s'avérer utile.

C'est un peu dans cette optique que seront traités les quelques exemples du paragraphe 3.

Notons que nous n'aborderons pas en général des problèmes techniques d'adressage et d'écriture du programme dans un langage bien définie : nous ferons remarquer au passage, seulement, que les solutions trouvées peuvent déboucher sur des programmations faciles.

1.4. OPTIMISATION DE PROGRAMME ET COMPILATION

Dans le cadre général des optimisations de programmes à la charge du compilateur il est envisageable de procéder à des restructurations d'ensembles de données ou même de parties du code généré afin d'adapter ce code à une utilisation en environnement paginé.

II. CONSTRUCTION DE MATRICES D'ADRESSES JOINTIVES OU m-JOINTIVES

2.1. MATRICES D'ADRESSES JOINTIVES OU m-JOINTIVES

La matrice d'adresses jointives a été définie au chapitre I, paragraphe 2.2.2.

Les matrices d'adresses m-jointives ont été définies au chapitre I, paragraphe 3.2.2.. Nous nous limiterons au cas où les m coefficients μ_q sont tous égaux à 1/m. Pour des raisons de simplicité nous construirons la matrice à coefficients entiers égales à m fois la matrice précédente.

La définition même de ces matrices nous fournit un procéder de construction :
à chaque appel d'une donnée ℓ_t nous ajoutons +1 à tous les éléments de la matrice d'indices (i,j) tels que :

$$(x_i, x_j) = (\ell_t, \ell_{t-q}) \qquad q=1,\ldots,m$$

ou

$$(x_j, x_i) = (\ell_t, \ell_{t-q}) \qquad q=1,\ldots,m$$

la matrice étant bien sûr initialisée à zéro.

2.2. UN PROCEDE DE CONSTRUCTION DE MATRICES D'ADRESSES m-JOINTIVES

Pour mettre en oeuvre sur quelques exemples les techniques de restructuration nous avons été amené a construire de manière systématique des matrices d'adresses m-jointives.

Nous avons choisi l'hypothèse d'un programme initial donné sous forme d'un code FORTRAN [5] écrit avec certaines conventions.

La construction des matrices d'adresses m-jointives relatives au déroulement de ce programme se fera en trois étapes :

Etape n° 1 Définition du travail a effectuer

Nous définirons d'abord quel type de matrices d'adresses m-jointives nous voulons calculer,c'est-à-dire quelle valeur de m avons-nous choisie.

Nous procéderons à l'insertion de cartes "commandes" dans le programme initial de façon à définir exactement le travail a effectuer.

L'étape n° 1 est donc à la charge de l'utilisateur.

Etape n° 2

Le programme initial avec les cartes "commandes" est transformé de manière automatique en un programme définitif toujours sous forme d'un code FORTRAN.

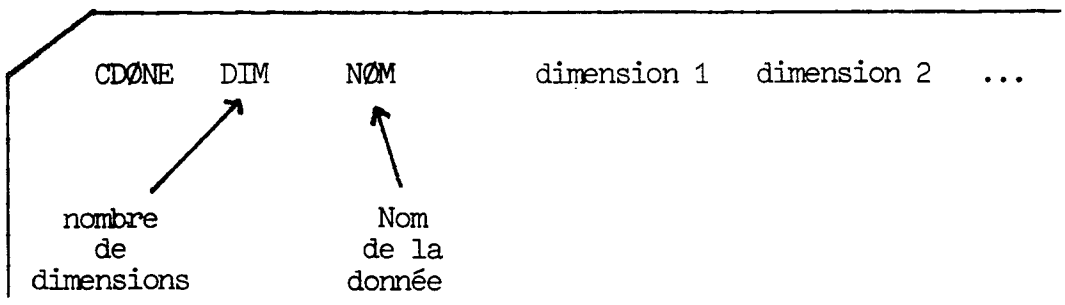
Etape n° 3

Le programme transformé est exécuté et fournit, en plus des résultats obtenus par le programme initial, la matrice d'adresses m-jointives.

2.2.1. Commandes

Les cartes "commandes" sont de deux types et sont sous formes de cartes commentaires FORTRAN particulières.

Commande de prise en compte d'une donnée



Le nombre de dimension est un entier :

0 pour une variable non indicée

1 pour un vecteur

2 pour une matrice etc...

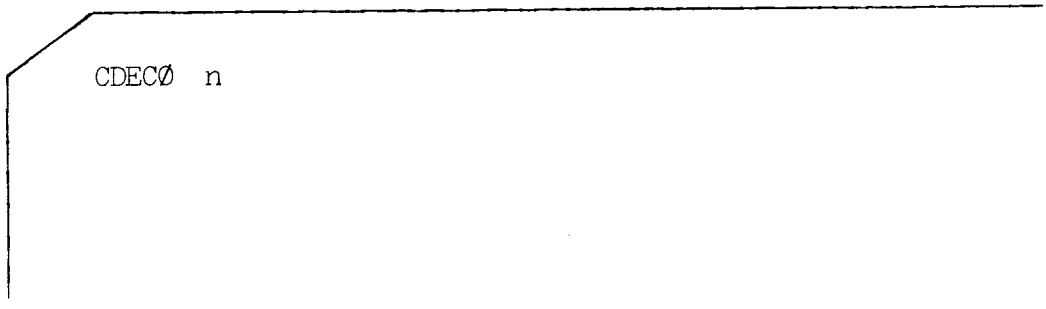
Le nom correspond au nom de la variable utilisée dans le programme et que l'on désire voir figurer dans l'ensemble des données.

Les dimensions sont des entiers ou des variables entières du programme définies au moment de l'exécution.

La carte commande précédente provoquera lors de la transformation du programme la génération d'instructions spéciales ; leur but est de répertorier les données que l'on veut prendre en compte et de leur affecter une numérotation. La numérotation se fait dans l'ordre d'apparition des données dans les cartes commandes. Pour un tableau les éléments sont rangés suivant la convention FORTRAN par indice croissant, le premier indice variant le plus rapidement, et ainsi de suite.

EXEMPLE : l'élément A(i,j) d'une matrice de $\mathcal{M}_{n,n}(\mathbb{R})$ est numéroté $(j-1) \times n + i$ si le premier élément est numéroté 1.

Commande de traitement d'instructions



n indique que le traitement portera sur les n instructions suivant la carte commande.

Le traitement d'une instruction signifie, lors de la transformation du programme, la génération d'un code permettant la mise à jour éventuelle de la matrice d'adresses m-jointives lors de l'exécution si une donnée prise en compte figure dans l'instruction traitée.

Une instruction comportant une donnée prise en compte devra être de la forme :

X = Y

ou

X = Y t Z $\qquad t \in \{*,/,-,+\}$

ces restrictions assez draconiennes ont pour but de simplifier à l'extrême le programme effectuant la transformation qui ressemblerait sinon, de fort près au compilateur FORTRAN.

Si X,Y,Z sont des données prises en compte, leur ordre d'appel lors del'exécution sera :

- pour X = Y ordre Y,X
- pour X = Y t Z ordre Y,Z,X

L'exécution de ce type d'instruction se réalise en général avec un accumulateur ACC suivant le schéma :

ACC := Y
ACC := ACC t Z
X := ACC

Restriction sur l'écriture du programme initial

Nous ne rentrerons pas dans les détails techniques et nous dirons simplement que ces restrictions concernent l'emploi de certains mots réservés et comme nous l'avons vu au paragraphe 1.3. la forme des instructions traitées.

Nous allons donner un exemple simple et brièvement commenté de programme transformé.

Programme initial avec cartes commandes

```
C PROGRAMME DE MULTIPLICATION DE MATRICES
      REAL A(4,4),B(4,4),C(4,4)
      READ(5,50) N
   50 FORMAT(I3)
C INITIALISATION DES MATRICES
      DO 1 I=1,N
      DO 1 J=1,N
      A(I,J)=1.
      B(I,J)=1.
    1 CONTINUE
      CALL PROD(A,B,C,N)
C SORTIE DE LA MATRICE C
      DO 3 I=1,N
      WRITE(6,51) (C(I,J),J=1,N)
    3 CONTINUE
   51 FORMAT(1X,6E15.7)
      STOP
      END
      SUBROUTINE PROD(A,B,C,N)
      DIMENSION A(N,N)
      DIMENSION B(N,N)
      DIMENSION C(N,N)
```

Cartes commandes

```
CDONE 2 A N N
CDONE 2 B N N
CDONE 2 C N N
CDECO 09
```

```
      DO 1 I=1,N
      DO 1 J=1,N
      ACC1=0.
      DO 2 K=1,N
      ACC2=A(I,K)*B(K,J)
      ACC1=ACC1+ACC2
    2 CONTINUE
      C(I,J)=ACC1
    1 CONTINUE
      RETURN
      END
```

Programme transformé

- déclaration et initialisation des variables supplémentaires créées pour la transformation du programme.

```
      INTEGER*2 TRANSI( 0000000050, 0000000050)
      COMMON TRANSI
      INTEGER XXX000,XXX001,XXX002,XXX003,XXX004,XXX005
      COMMON XXX000,XXX001,XXX002,XXX003,XXX004,XXX005
      INTEGER XXXX00,XXXX01
      COMMON XXXX00,XXXX01
      INTEGER XXXX02,XXXX03,XXXX04,XXXX05
      INTEGER XXXX06,XXXX07,XXXX08,XXXX09
      COMMON XXXX02,XXXX03,XXXX04,XXXX05
      COMMON XXXX06,XXXX07,XXXX08,XXXX09
      INTEGER XXXXX0
      COMMON XXXXX0
      INTEGER XX0000
      COMMON XX0000
      EXTERNAL XXXXXX
      EXTERNAL X00000
      XXXXX0= 0000000050
      XXXX00=XXXXX0
      XXXX01=XXXXX0
      XXX000=0
      DO 1 I=1,XXXXX0
      DO 1 J=1,XXXXX0
      TRANSI(I,J)=0
    1 CONTINUE
      CALL XXXXXX
      CALL X00000
      STOP
      END
```

```
      SUBROUTINE XXXXXX
      INTEGER*2 TRANSI( 0000000050, 0000000050)
      COMMON TRANSI
      INTEGER XXX000,XXX001,XXX002,XXX003,XXX004,XXX005
      COMMON XXX000,XXX001,XXX002,XXX003,XXX004,XXX005
      INTEGER XXXX00,XXXX01
      COMMON XXXX00,XXXX01
      INTEGER XXXX02,XXXX03,XXXX04,XXXX05
      INTEGER XXXX06,XXXX07,XXXX08,XXXX09
      COMMON XXXX02,XXXX03,XXXX04,XXXX05
      COMMON XXXX06,XXXX07,XXXX08,XXXX09
      INTEGER XXXXX0
      COMMON XXXXX0
      INTEGER XX0000
      COMMON XX0000
C PROGRAMME DE MULTIPLICATION DE MATRICES
      REAL A(4,4),B(4,4),C(4,4)
      READ(5,50) N
   50 FORMAT(I3)
C INITIALISATION DES MATRICES
      DO 1 I=1,N
      DO 1 J=1,N
      A(I,J)=1.
      B(I,J)=1.
    1 CONTINUE
      CALL PROD(A,B,C,N)
C SORTIE DE LA MATRICE C
      DO 3 I=1,N
      WRITE(6,51) (C(I,J),J=1,N)
    3 CONTINUE
   51 FORMAT(1X,6E15.7)
      RETURN
      END
      SUBROUTINE PROD(A,B,C,N)
      INTEGER*2 TRANSI( 0000000050, 0000000050)
      COMMON TRANSI
      INTEGER XXX000,XXX001,XXX002,XXX003,XXX004,XXX005
      COMMON XXX000,XXX001,XXX002,XXX003,XXX004,XXX005
      INTEGER XXXX00,XXXX01
      COMMON XXXX00,XXXX01
      INTEGER XXXX02,XXXX03,XXXX04,XXXX05
      INTEGER XXXX06,XXXX07,XXXX08,XXXX09
      COMMON XXXX02,XXXX03,XXXX04,XXXX05
      COMMON XXXX06,XXXX07,XXXX08,XXXX09
      INTEGER XXXXX0
      COMMON XXXXX0
      INTEGER XX0000
      COMMON XX0000
      DIMENSION A(N,N)
      DIMENSION B(N,N)
      DIMENSION C(N,N)
CDONE 2 A N N
      XXX001=XXX000+N*N
C RECUPERATION DU NOMBRE TOTAL DE DONNEES TRAITEES
      XX0000=XXX001
CDONE 2 B N N
      XXX002=XXX001+N*N
C RECUPERATION DU NOMBRE TOTAL DE DONNEES TRAITEES
      XX0000=XXX002
CDONE 2 C N N
      XXX003=XXX002+N*N
C RECUPERATION DU NOMBRE TOTAL DE DONNEES TRAITEES
      XX0000=XXX003
```

Début du programme à transformer

Prise en compte des données

```
CDECO 09
      DO 1 I=1,N
      DO 1 J=1,N
      ACC1=0.
      DO 2 K=1,N
      ACC2=A(I,K)*B(K,J)
      XXXX02=XXX000+I+(K-1)*N
      TRANSI(XXXX02,XXXX00)=TRANSI(XXXX02,XXXX00)+1
      TRANSI(XXXX00,XXXX02)=TRANSI(XXXX02,XXXX00)
      TRANSI(XXXX02,XXXX01)=TRANSI(XXXX02,XXXX01)+1
      TRANSI(XXXX01,XXXX02)=TRANSI(XXXX02,XXXX01)
      XXXX00=XXXX01
      XXXX01=XXXX02
      XXXX02=XXX001+K+(J-1)*N
      TRANSI(XXXX02,XXXX00)=TRANSI(XXXX02,XXXX00)+1
      TRANSI(XXXX00,XXXX02)=TRANSI(XXXX02,XXXX00)
      TRANSI(XXXX02,XXXX01)=TRANSI(XXXX02,XXXX01)+1
      TRANSI(XXXX01,XXXX02)=TRANSI(XXXX02,XXXX01)
      XXXX00=XXXX01
      XXXX01=XXXX02
      ACC1=ACC1+ACC2
    2 CONTINUE
      C(I,J)=ACC1
      XXXX02=XXX002+I+(J-1)*N
      TRANSI(XXXX02,XXXX00)=TRANSI(XXXX02,XXXX00)+1
      TRANSI(XXXX00,XXXX02)=TRANSI(XXXX02,XXXX00)
      TRANSI(XXXX02,XXXX01)=TRANSI(XXXX02,XXXX01)+1
      TRANSI(XXXX01,XXXX02)=TRANSI(XXXX02,XXXX01)
      XXXX00=XXXX01
      XXXX01=XXXX02
    1 CONTINUE
      RETURN
      END
```

Instructions nécessaire à la construction de la matrice d'adresses jointives

Sortie de la matrice d'adresses jointives

```
      SUBROUTINE X00000
      INTEGER*2 TRANSI( 0000000050, 0000000050)
      COMMON TRANSI
      INTEGER XXX000,XXX001,XXX002,XXX003,XXX004,XXX005
      COMMON XXX000,XXX001,XXX002,XXX003,XXX004,XXX005
      INTEGER XXXX00,XXXX01
      COMMON XXXX00,XXXX01
      INTEGER XXXX02,XXXX03,XXXX04,XXXX05
      INTEGER XXXX06,XXXX07,XXXX08,XXXX09
      COMMON XXXX02,XXXX03,XXXX04,XXXX05
      COMMON XXXX06,XXXX07,XXXX08,XXXX09
      INTEGER XXXXX0
      COMMON XXXXX0
      INTEGER XX0000
      COMMON XX0000
      DO 1 I=1,XX0000
      WRITE(4,2) (TRANSI(I,J),J=1,XX0000)
      WRITE(8,3) (TRANSI(I,J),J=1,XX0000)
    1 CONTINUE
    2 FORMAT(1X,6512)
    3 FORMAT(4012)
      RETURN
      END
```

III. ASPECT PRATIQUE DES METHODES DE RESTRUCTURATION

3.1. SES DIFFICULTES

La première et sans doute la plus grande difficulté réside dans l'étude des gestions de mémoires avec plusieurs pages en mémoire opératoire. Nous avions vu (chapitre I, paragraphe 3.2.2.) que les matrices d'adresses jointives généralisées associées à l'algorithme de remplacement LRU ne permettaient seulement que d'avoir des majorations du nombre de demandes de pages. Les matrices d'adresses jointives généralisées sont elles relativement a construire dans le cas général.

En particulier, il faut bien avouer que les restructurations élaborées à partir de certaines matrices d'adresses jointives généralisées n'ont donné aucun résultat satisfaisant (cf. page 129).

Pour ces raisons il est plus juste de considérer ses méthodes comme une "aide à la restructuration" utilisée de manière intéractive.

3.2. REMARQUES GENERALES SUR LES EXEMPLES TRAITES

Nos exemples porteront sur des algorithmes très simples de calcul matriciel [2] : la donnée élémentaire sera un nombre réel. Nous envisagerons pour une capacité de mémoire opératoire fixe quelques gestions de mémoire de type paginé.

Les méthodes de partitionnement de matrices sont celles étudiées aux chapitres II et III éventuellement suivies d'amélioration locale (algorithme d'échange). Notons que pour les matrices d'adresses jointives généralisées en particulier une optimisation très grossière basée sur un ou deux vecteurs propres est souvent préférable; : cela se comprend assez bien car une "idée globale" du problème même assez floue est certainement plus réaliste qu'une optimisation fine d'une quantité qui en fin de compte n'est qu'un majorant.

Les minorants proposés pour le nombre de demandes de pages sont calculés à l'aide de la matrice d'adresses jointives (dans le cas où la mémoire opératoire ne contient qu'une seule page) où à l'aide de la gestion de mémoire "donnée par donnée" (chapitre I, paragraphe 3.3.2.). Ces minorants sont souvent médiocres et ne tiennent certainement pas assez compte des particularités des algorithmes.

L'ordre dans lequel sont initialement rangées les données sera précisé à chaque exemple ; à l'intérieur d'un tableau l'ordre sera celui indiqué au paragraphe 2.3.1.
La répartition initiale en pages est alors obtenu en remplissant successivement les pages avec les données dansl'ordre précédent.

Nous entendrons par taille d'une page le nombre maximal de données qu'elle peut contenir.

IV. EXEMPLES

4.1. EXEMPLES : RESOLUTION D'UN SYSTEME LINEAIRE A DEUX INCONNUES AVEC QUATRE MEMOIRES OPERATOIRES

Soit le système linéaire :

$$AX + BY = E$$

$$CX + DY = F$$

Nous supposerons que toutes les opérations sont faites par l'intermédiaire d'un accumulateur ACC et nous désirons avoir le résultat dans X et Y .

Nous avons donc huit données : initiallement dans l'ordre :

A , B , C , D , E , F , X , Y .

4.1.1. Méthode de Gauss

L'algorithme pourra s'écrire symboliquement en un code de type FORTRAN :

1	ACC = C
2	ACC = ACC/A
3	ACC = ACC*B
4	ACC = -ACC
5	ACC = ACC+D
6	D = ACC
7	ACC = C
8	ACC = ACC/A
9	ACC = ACC*E
10	ACC = -ACC
11	ACC = ACC+F
12	F = ACC
13	ACC = ACC/D
14	Y = ACC
15	ACC = ACC*B
16	ACC = -ACC
17	ACC = ACC+E
18	ACC = ACC/A
19	X = ACC

Quatre pages de taille 1 : Gestion de mémoire "donnée par donnée"

L'ordre des données n'intervient pas. Avec l'algorithme de remplacement optimum nous avons :

9 demandes de pages.

Deux pages de taille 2

Le minorant du nombre de demandes de pages fournit par l'étude de la gestion de mémoire "donnée par donnée" est égal à 5 .

Nous pouvons utilisée la propriété exposée au chapitre I, paragraphe 3.3.8. en partant de l'organisation de données initiale et de l'algorithme de remplacement optimum.

La restructuration nous fournit la même organisation de données :

Page 1 = {A,B} Page 2 = {C,D}

Page 3 = {E,F} Page 4 = {X,Y}

Avec l'algorithme de remplacement optimum nous avons :

6 demandes de pages.

Une page de dimension 4

La gestion de mémoire donnée par donnée nous fournit un minorant égal à 3 pour le nombre de demandes de pages.

La matrice d'adresses jointives nous fournit la même minoration.

La répartition initiale provoque 8 demandes de pages.

Par restructuration nous obtenons l'organisation de données suivantes :

Page 1 = {A,C,E,X}

Page 2 = {B,C,F,Y}

Le nombre de demandes de pages est alors de 5

Des considérations particulières a cette petite matrice nous permettent d'affirmer que la solution trouvée est en fait optimale (pour le cas d'une page de dimension 4).

4.1.2. Méthode des déterminants de Crammer

Pour une valeur aussi faible de l'ordre du système, peut-être serait-il plus simple d'utiliser les formules de Crammer.
Il nous faut pour cela introduire une nouvelle variable DELTA .
Les formules de Crammer sont les suivantes :

DELTA = AD-BC

X = (ED-FB)/DELTA Y = (FA-EC)/DELTA

Les données sont initialement dans l'ordre :

A , B , C , D , E , F , X , Y , DELTA

Le code FORTRAN utilisé pour cet algorithme est le suivant :

```
 1 | ACC = C                    13 | ACC = ACC-X
 2 | ACC = ACC*B                14 | ACC = ACC/DELTA
 3 | DELTA = ACC                15 | X = ACC
 4 | ACC = A                    16 | ACC = C
 5 | ACC = ACC*D                17 | ACC = ACC*E
 6 | ACC = ACC-DELTA            18 | Y = ACC
 7 | DELTA = ACC                19 | ACC = A
 8 | ACC = F                    20 | ACC = ACC*F
 9 | ACC = ACC*B                21 | ACC = ACC-Y
10 | X = ACC                    22 | ACC = ACC/DELTA
11 | ACC = E                    23 | Y = ACC
12 | ACC = ACC*D
```

Quatre pages de taille 1 : Gestion de mémoire "donnée par donnée"

Avec l'algorithme de remplacement optimum nous avons :

12 demandes de pages.

Deux pages de taille 2

Le minorant fourni par la gestion de mémoire "donnée par donnée" est 6

Nous appliquons la même méthode que pour la méthode de Gauss avec l'algorithme de remplacement optimal et l'organisation de données initiales.

La restructuration nous fournit l'organisation suivante :

Page 1 = {A,B} Page 2 = {C,DELTA} Page 3 = {E,F}

Page 4 = {X,Y} Page 5 = {D} .

Le nombre de demandes de pages avec l'algorithme de remplacement optimum est : 12 .

Une page de taille 4

Le minorant fourni par la gestion de mémoire donnée par donnée est : 3 .

Le minorant fourni par la matrice d'adresses jointives est : 8 .

La restructuration nous donne l'organisation de données suivantes :

Page 1 : A,F,Y,DELTA

Page 2 : B,C,E,X

Page 3 : DELTA

Le nombre de demandes de pages est alors de : 10 .

Conclusion

Parmi les éventualités envisagées la meilleure (pour le nombre de demandes de pages) est celle de la méthode de Gauss utilisant un système de pagination à une seule page en mémoire avec la répartition proposée.

4.2. Exemples : Calcul de valeurs propres d'une matrice de $\mathcal{M}_{4,4}(\mathbb{R})$ avec huit mémoires opératoires

Nous emploierons l'algorithme LR, sans nous préoccuper de problèmes de convergence.

Nous étudierons la partie de l'algorithme correspondant à une itération de l'algorithme LR : décomposition LR et produit R*L le résultat se trouvant dans la matrice initiale A .

Le code FORTRAN ayant servi a étudier cet algorithme se trouve en annexe 1 de ce chapitre.

Les données prise en compte sont celle s de la matrice A et sont donc au nombre de 16 .

Huit pages de taille 1 : Gestion de mémoire "donnée par donnée"

Avec l'algorithme de remplacement optimum nous avons 31 demandes de pages.

Deux pages de taille 4

La gestion de mémoire "donnée par donnée" nous fournit un minorant de 8 demandes de pages.

La mémoire opératoire pouvant contenir deux pages nous avons construit la matrice d'adresses 2-jointives.

La restructuration nous fournit l'organisation de données suivantes

A =

4	2	1	3
4	2	1	3
4	2	1	2
4	3	1	3

A la place de chaque élément de la matrice A est inscrit le numéro de la page à laquelle il appartient.

Le nombre de demandes de pages avec l'algorithme LRU est de 44 et il devient 32 avec l'algorithme optimum.

Notons cependant que cette nouvelle organisation de données n'est pas meilleure que l'organisation initiale dont elle est d'ailleurs fort proche : en effet, la répartition initiale provoquait 42 demandes de pages avec l'algorithme LRU et 31 avec l'algorithme optimum.

C'est un premier exemple où il convient donc d'utiliser avec prudence les techniques de restructuration.

Une page de taille 8

Le meilleur minorant nous est fourni par la matrice d'adresses jointive et il est de 28 demandes de pages.

La restructuration nous donne l'organisation de données suivante :

1	2	1	2
1	2	1	2
1	2	1	2
1	2	1	2

Nous avons alors 37 demandes de pages alors que la répartition initiale provoque 52 demandes de pages.

Commentaire

Parmi les cas étudiés la meilleure solution semble être un découpage en quatre pages suivant la répartition initiale ou bien une gestion "donnée par donnée" puisque toutes deux aboutissent à 31 demandes de pages.

4.3. EXEMPLE — MULTIPLICATION DE DEUX MATRICES DE $\mathcal{M}_{4,4}(\mathbb{R})$ AVEC 16 MEMOIRES OPERATOIRES

A, B et C étant trois matrices de $\mathcal{M}_{4,4}(\mathbb{R})$ nous voulons réaliser le produit A*B et placer le résultat dans C .

Le code FORTRAN correspondant est en annexe 2 ; a noter l'emploi de deux accumulateurs.

Les seuls données prises en comptes sont les éléments des matrices A, B, C prises dans cet ordre.

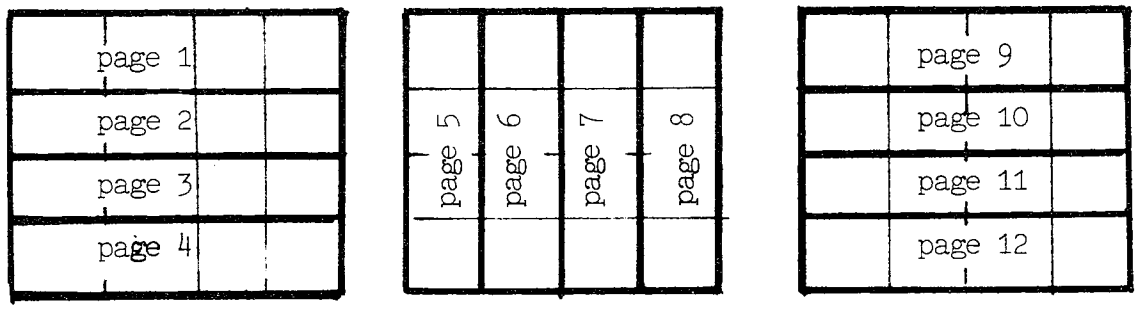
Seize pages de taille 1 : Gestion "donnée par donnée"

Avec l'algorithme de remplacement optimum nous avons 59 demandes de pages.

Quatre page de taille 4

La gestion de mémoire donnée par donnée nous fournit un minorant de 15 demandes de pages.

La restructuration à partir de la matrice d'adresses 4-jointives nous donnent la répartition de données suivante : chaque ligne de A et C constitue une page ainsi que chaque colonne de B comme cela est schématisé ci-dessous.



A B C

Le nombre de demandes de pages avec l'algorithme LRU est alors égal à 24 et l'utilisation de l'algorithme de remplacement optimum le réduit à 19 .

Avec la répartition initiale nous avions 96 demandes de pages avec l'algorithme LRU et 51 avec l'algorithme optimum.

Deux pages de taille 8

La gestion de mémoire donnée par donnée nous fournit un minorant de 8 demandes de pages.

Nous procédons comme précédemment avec une matrice d'adresse 2-jointives : malheureusement les résultats obtenus par restructuration de cette matrice sont peu significatifs.

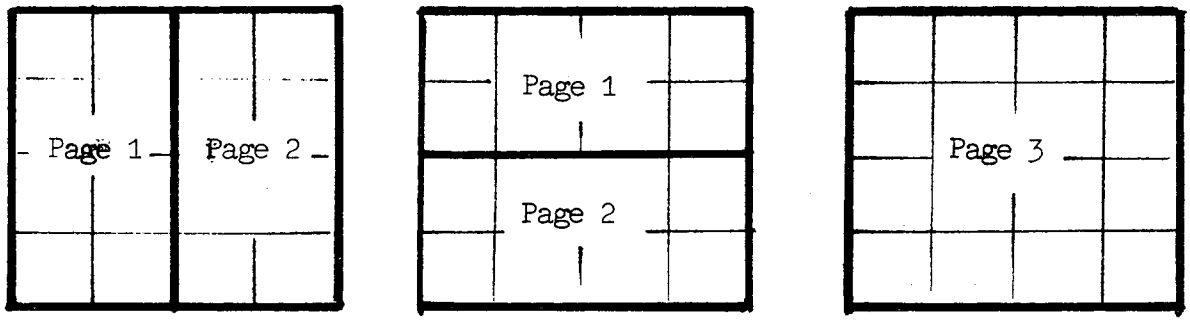
En particulier, la restructuration obtenue ne permet pas d'améliorer la répartition initiale qui provoque 64 demandes depages avec l'algorithme de remplacement LRU et 56 avec l'algorithme de remplacement optimum.

Il faut cependant noter une restructuration efficace obtenue en regroupant deux pages consécutives de taille 4 conformément au schéma précédemment obtenu avec la matrice d'adresses 4-jointives : nous avons alors 48 demandes de pages avec l'algorithme LRU et 34 avec l'algorithme optimum.

Un page de taille 16

La gestion de mémoire donnée par donnée nous fournit un minorant de 4 demandes de pages.

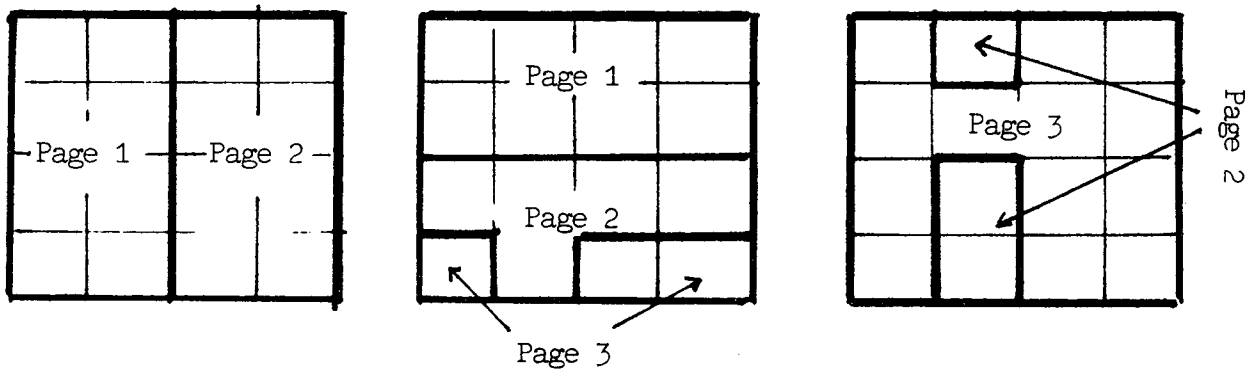
Par une première restructuration nous obtenons l'organisation de données suivantes :



A B C

Le nombre de demandes de pages est alors de 48 .
La répartition initiale desdonnées en provoquait 144 .

Notons qu'une restructuration plus fine nous permet d'arriver à une organisation de données provoquant seulement 45 demandes de pages et qui est la suivante :



Commentaire

Nous pouvons faire quelques commentaires de portée tout à fait générale pour les problèmes de restructuration.

Tout d'abord le changement radical de la répartition lorsqu'on envisage une page de taille 16 ou quatre pages de taille 4 : la difficulté du problème avec deux pages de taille 8 s'explique peut-être par un compromis entre deux situations très différentes.

Le compromis entre une certaine "régularité" de la restructuration et la qualité de la restructuration apparaît clairement dans le dernier cas traité.

4.4. EXEMPLE — DECOMPOSITION DE CHOLESKY D'UNE MATRICE SYMETRIQUE DEFINIE POSITIVE DE $\mathcal{M}_{10,10}(IR)$ AVEC 20 MEMOIRES OPERATOIRES

La matrice A est décomposée sous la forme

$A = RR^T$

la matrice R étant placée dans la partie triangulaire inférieure de A.

Le code FORTRAN utilisé figure en annexe 3 : a noter l'utilisation de deux accumulateurs.

Les seules données prises en compte sont les éléments de la matrice A.

Vingt pages de taille 1 : Gestion de mémoire donnée par donnée

Le nombre de demandes de pages est égal à 87 .

Deux pages de taille 10

Le minorant fournit par la gestion de mémoire donnée par donnée est 9 , minorant sans intérêt et que l'on peut de toute manière remplacer par 10 car toutes les pages sont demandées.

L'organisation de donnée résultat de la restructuration est la suivante *

6	5	5	8	9	9	9	10	7	7
1	6	5	8	8	9	10	10	7	7
1	1	6	9	9	9	10	10	7	8
3	3	3	5	9	10	10	10	7	8
4	3	3	4	5	10	9	10	7	8
3	3	3	4	2	5	9	5	7	8
1	1	3	2	2	4	5	6	7	8
1	1	4	2	2	4	4	6	7	8
1	1	4	2	2	4	5	6	6	8
1	3	2	2	2	4	5	6	6	6

Le nombre de demande de pages est alors de 123 avec l'algorithme de remplacement LRU et de 106 avec l'algorithme optimum.

L'amélioration est certaine par rapport à la répartition initiale qui provoque 192 demandes de pages avec l'algorithme LRU et 146 avec l'algorithme de remplacement optimum.

Un page de taille 20

Le minorant fournit par la gestion de mémoire donnée par donnée est 3 demandes de pages, minorant sans intérêt pouvant être remplacé par 5 (il y a en effet cinq pages).

* A chaque élément de matrice nous avons affecté le numéro de la page à laquelle elle appartient.

La restructuration nous donne la répartition indiquée sur le schéma et nous avons alors 81 demandes de pages.

3	3	3	3	4	5	4	4	5	5
1	3	3	3	5	4	4	4	5	5
1	1	1	3	5	4	4	4	5	5
1	1	1	3	3	4	4	4	5	5
1	1	1	2	2	4	4	4	5	5
1	1	2	2	2	4	4	4	5	5
1	1	2	2	2	2	4	4	5	5
1	1	2	2	2	2	3	3	4	4
1	1	2	2	2	2	3	3	3	5
1	1	2	2	2	3	3	3	3	3

Notons que la répartition initiale provoquait 106 demandes de pages.

Remarquons que la répartition proposée n'est pas très simple : il semble cependant que les éléments de la matrice aient été rangés approximativement par colonnes d'abord dans la partie triangulaire inférieure stricte et ensuite dans la partie triangulaire supérieure. Cette remarque pourrait donc nous suggérer l'organisation suivante beaucoup plus "régulière" et aisément applicable à toute matrice

3	3	3	3	3	4	4	4	5	5
1	3	3	3	3	4	4	4	5	5
1	1	3	3	3	4	4	4	5	5
1	1	1	3	3	4	4	4	5	5
1	1	1	2	3	4	4	4	5	5
1	1	1	2	2	4	4	4	5	5
1	1	2	2	2	2	4	4	5	5
1	1	2	2	2	2	2	5	5	5
1	1	2	2	2	2	3	3	5	5
1	1	2	2	2	2	3	3	3	5

Le nombre de demandes de pages est alors 82 peu différent du précédent.

Commentaire :

La dernière solution envisagée, même après une certaine régularisation, reste la meilleure du point de vue du nombre de demandes de pages.

V. CONCLUSION

La pratique de la restructuration apparait donc difficile et les méthodes élaborées doivent être considérées plutôt comme une aide à la résolution de tels problèmes : en particulier l'utilisation des matrices d'adresses jointives généralisées doit être prudente.

Remarquons aussi que toutes les possibilités, d'une mémoire opératoire réduite à une seule page jusqu'à la gestion de mémoire "donnée par donnée" ont pu donner sur des exemples précis des résultats satisfaisants.

Notons qu'il est probablement illusoire d'espérer des méthodes plus générales pour résoudre de tels problèmes. Ce n'est certainement qu'en tirant parti des particularités de chaque problème que l'on pourrait espérer de meilleurs résultats.

ANNEXE 1

```
      REAL A(4,4)
      EXTERNAL PASLR
      READ(5,50) N
   50 FORMAT(I3)
      DO 1 I=1,N
      READ(5,51) (A(I,J),J=1,N)
    1 CONTINUE
      CALL PASLR(A,N)
      DO 2 I=1,N
      WRITE(6,52) (A(I,J),J=1,N)
    2 CONTINUE
   52 FORMAT(1X,6E15.7)
   51 FORMAT(10F5.1)
      STOP
      END
      SUBROUTINE PASLR(A,N)
      DIMENSION A(N,N)
CDONE 2 A N N
CDECO 58
      DO 2 I=2,N
      ACC=A(I,1)
      ACC=ACC/A(1,1)
      A(I,1)=ACC
    2 CONTINUE
      DO 3 K=2,N
      KM1=K-1
      DO 4 J=K,N
      DO 5 I=1,KM1
      ACC=A(K,I)
      ACC=ACC*A(I,J)
      ACC=ACC-A(K,J)
      ACC=-ACC
      A(K,J)=ACC
    5 CONTINUE
    4 CONTINUE
      IF (K.EQ.N) GOTO 9
      KP1=K+1
      DO 6 I=KP1,N
      DO 7 J=1,KM1
      ACC=A(I,J)
      ACC=ACC*A(J,K)
      ACC=ACC-A(I,K)
      ACC=-ACC
      A(I,K)=ACC
    7 CONTINUE
      ACC=A(I,K)
      ACC=ACC/A(K,K)
      A(I,K)=ACC
    6 CONTINUE
    9 CONTINUE
    3 CONTINUE
```

```
      DO 11 I=1,N
      IF (I.EQ.1) GOTO 14
      IM1=I-1
      DO 12 J=1,IM1
      DO 13 K=I,N
      ACC=A(I,K)
      ACC=ACC*A(K,J)
      IF (I.EQ.K) GOTO 18
      ACC=ACC+A(I,J)
   18 CONTINUE
      A(I,J)=ACC
   13 CONTINUE
   12 CONTINUE
   14 CONTINUE
      DO 15 J=I,N
      IF (J.EQ.N) GOTO 16
      JP1=J+1
      DO 17 K=JP1,N
      ACC=A(I,K)
      ACC=ACC*A(K,J)
      ACC=ACC+A(I,J)
      A(I,J)=ACC
   17 CONTINUE
   16 CONTINUE
   15 CONTINUE
   11 CONTINUE
      RETURN
      END
```

ANNEXE 2

```
      SUBROUTINE PROD(A,B,C,N)
      DIMENSION A(N,N)
      DIMENSION B(N,N)
      DIMENSION C(N,N)
CDONE 2 A N N
CDONE 2 B N N
CDONE 2 C N N
CDECO 09
      DO 1 I=1,N
      DO 1 J=1,N
      ACC1=0.
      DO 2 K=1,N
      ACC2=A(I,K)*B(K,J)
      ACC1=ACC1+ACC2
    2 CONTINUE
      C(I,J)=ACC1
    1 CONTINUE
      RETURN
      END
```

ANNEXE 3

```
      SUBROUTINE CHOLES(A,N)
      DIMENSION A(N,N)
CDONE 2 A N N
CDECO 03
      ACC1=A(1,1)
      ACC1=SQRT(ACC1)
      A(1,1)=ACC1
      DO 1 I=2,N
CDECO 01
      A(I,1)=A(I,1)/ACC1
    1 CONTINUE
      DO 2 J=2,N
      ACC1=0.
      JM1=J-1
      DO 3 I=1,JM1
CDECO 02
      ACC2=A(J,I)*A(J,I)
      ACC1=ACC1+ACC2
    3 CONTINUE
CDECO 03
      ACC1=A(J,J)-ACC1
      ACC1=SQRT(ACC1)
      A(J,J)=ACC1
      IF (J.EQ.N) GOTO 6
      JP1=J+1
CDECO 01
      DO 4 I=JP1,N
      ACC2=0.
      DO 5 K=1,JM1
CDECO 02
      ACC3=A(J,K)*A(I,K)
      ACC2=ACC2+ACC3
    5 CONTINUE
CDECO 02
      ACC2=A(I,J)-ACC2
      A(I,J)=ACC2/ACC1
    4 CONTINUE
    6 CONTINUE
    2 CONTINUE
      RETURN
      END
```

CHAPITRE VI

COMPLEXITE DES ALGORITHMES ETUDIES .

PROBLEMES DE "CLUSTERING".

I. "COMPLEXITE" DES ALGORITHMES A LA PRECISION RELATIVE ε

1.1. BIPARTITIONNEMENT

1.1.1. Décompte d'opérations élémentaires

Soit $A \in \mathcal{M}_{n,n}(\mathbb{R})$ donnée par :

$$A = \sum_{i=1}^{P} B_i B_i^T \qquad B_i \in \mathbb{R}^n \quad ; \quad i=1,2,\ldots,p .$$

Nous avons mis (chapitre III) le problème de bipartitionnement de cette matrice sous la forme :

$$\underset{K \in \mathcal{K}}{\text{Max}} \sum_{i=1}^{p} (K^T B_i)^2$$

Les méthodes à précision relative ε consistaient en l'énumération des solutions des problèmes projetés :

$$\underset{K \in \mathcal{K}}{\text{Max}}\ K^T \sum_{i=1}^{P} B_i\, u_\varepsilon(i)$$

u_ε décrivant un ensemble de $\mathbb{R}^P$ ($\mathcal{U}_\varepsilon^S(p)$ ou $\mathcal{U}_\varepsilon(p)$) dont le nombre d'éléments ne dépend que de ε et p ($N^S(p,\varepsilon)$, $N(p,\varepsilon)$) .

Schématiquement l'algorithme se déroulait de la manière suivante :

Tant qu'il reste des vecteurs u_ε :

- trier les composantes du vecteur $\sum_{i=1}^{P} B_i\, u_\varepsilon(i)$
- évaluer $\underset{K \in \mathcal{K}}{\text{Max}}\ K^T \sum_{i=1}^{P} B_i\, u_\varepsilon(i)$
- mettre en mémoire le vecteur K_ε^* solution du problème précédent si la valeur correspondante est supérieure au maximum trouvé pour les u déjà énumérés.

Essayons d'évaluer sommairement le nombre d'opérations élémentaires du type "comparaison" et "opérations arithmétiques usuelles" nécessaire pour le déroulement de cet algorithme.

L'évaluation du vecteur $\sum_{i=1}^{P} B_i \, u_\varepsilon(i)$ demande de l'ordre de 2 p n additions et multiplications.

Le tri du vecteur $\sum_{i=1}^{P} B_i \, u_\varepsilon(i)$ demande de l'ordre de K n Log_2 n "comparaisons" par la méthode utilisée (Heapsort).

L'évaluation du critère demande de l'ordre de n additions-soustractions.

Il reste à faire un test pour réaliser la dernière étape. Compte tenu que nous devons réaliser cet ensemble d'opérations pour chaque u_ε le bilan total est donc de :

$$M(p,\varepsilon)\ (A\ n\ \text{Log}_2\ n + B(p)n + C)$$

où $M(p,\varepsilon)$ est le nombre de vecteurs u_ε .

Si nous faisons entrer en compte le calcul du vecteur u_ε le bilan est plus difficile a établir mais le nombre "d'opérations élémentaires" intervenant dans le calcul des u_ε ne dépend que de p et ε .

De toute manière le nombre "d'opérations élémentaires" est donc de la forme :

$$A(p,\varepsilon)\ n\ \text{Log}_2\ n + B(p,\varepsilon)n + C(p,\varepsilon)$$

Nous avons donc la propriété :

1.1.2. Propriété

p et ε étant fixés le nombre "d'opérations élémentaires" (comparaisons, opérations arithmétiques usuelles) nécessaires au déroulement de l'algorithme précédent est asymptotiquement en $A\ n\ Log_2\ n$ lorsque $n \to +\infty$.

1.1.3. Transferts et occupation mémoire

Il est facile de voir que la propriété reste vraie si nous prenons en considération les transferts de mémoire.

Au point de vue occupation mémoire l'algorithme précédent peut se dérouler, du point de vue des variables utilisées avec environ (p+1)n emplacements pour des variables réelles.

1.1.4. Remarque

Le résultat précédent est assez surprenant car le problème de bipartitionnement d'une matrice quelconque est NP-complet [1].

Cependant car si le comportement asymptotique en $n\ Log_2\ n$ peut paraître favorable les constantes $A(p,\varepsilon)$, $B(p,\varepsilon)$ et $C(p,\varepsilon)$ ne sont évidemment pas polynomiales en p et sont très importantes même pour des valeurs de p assez modestes à priori ($p \gtrsim 10$).

1.2. PROBLEMES DE PARTITIONNEMENTS A "TAILLES EGALES"

Soit $A \in \mathcal{M}_{n,n}(\mathbb{R})$ donnée par :

$$A = \sum_{i=1}^{p} B_i B_i^T \qquad B_i \in \mathbb{R}^n$$

Nous avons vu que l'application des méthodes à la précision relative ε fournissait pour les matrices ci-dessus des problèmes projetés qui étaient en fait des problèmes de programmation linéaire en nombres entiers [4].

Ces problèmes sont donc "à priori" NP-complet et l'algorithme ne peut donc pas être polynomial en n.

Notons cependant qu'il est possible que les problèmes projetés possèdent certaines particularités qui rendraient possible l'élaboration d'un algorithme polynomial en n.

De manières plus réaliste peut-être pourrait-on élaborer pour les problèmes projetés précédents des algorithmes polynomiaux fournissant une solution a une précision donnée comme cela a été fait pour le célèbre problème du knapsack [3] . Cela nous permettrait d'élaborer des algorithmes polynomiaux de partitionnement à "tailles égales".

II. PROBLEMES DE "CLUSTERING"

2.1. RESTRUCTURATION ET CLUSTERING

Dans pratiquement tous les travaux de restructuration l'analogie avec les problèmes de "clustering" est faite. Malheureusement le terme est bien vague et regroupe trop de problèmes différents. La notion de méthode de "clustering" n'a pas tellement de sens dans un cadre général et souvent la plus grande anarchie règne dans l'utilisation de ces méthodes.

Nous avons étudié quelques problèmes particuliers liés à la restructuration en nous efforçant de les poser clairement. Nous pouvons alors constater que les problèmes posés sont effectivement très voisins de quelques problèmes de "clustering" eux aussi parfaitement posés.

Les algorithmes étudiés fournissant des méthodes efficaces pour résoudre ces problèmes nous en donnons quelques exemples.

2.2. APPLICATION DU BIPARTITIONNEMENT DE MATRICES DU TYPE $B_1B_1^T + B_2B_2^T$; $B_1, B_2 \in \mathbb{R}^n$

2.2.1. Problèmes combinatoires sur un ensemble de points de $\mathbb{R}^2$

Soit un ensemble de n points de $\mathbb{R}^2$

$\{X_1, X_2, \ldots, X_n\}$

Le point X_i aura pour coordonnées $\begin{pmatrix} B_1(i) \\ B_2(i) \end{pmatrix}$

B_1 et B_2 étant deux vecteurs de $\mathbb{R}^n$.

En vue de simplifier certaines expressions nous associerons à une partition de l'ensemble des n points un couple de vecteurs de $\mathbb{R}^n$: (V_1, V_2) vérifiant les contraintes :

$$(C) \quad \begin{cases} V_k(i) = \begin{cases} 0 \\ 1 \end{cases} & k=1,2 \quad i=1,2,\ldots,n \\ V_1(i) + V_2(i) = 1 & i=1,2,\ldots,n \end{cases}$$

avec l'interprétation $V_k(i) = 1 \iff X_i \in$ groupe k

Nous allons voir que certains problèmes combinatoires sur la bipartition de cet ensemble de point sont extrêmement proches de ceux posés par le bipartitionnement d'une matrice $B_1B_1^T + B_2B_2^T$ et peuvent donc recevoir une résolution efficace par les algorithmes longuement développés au chapitre III.

Ces algorithmes sont à rapprocher avec certaines méthodes de "branch and bound" adaptées à ces problèmes [2].

2.2.2. Séparation optimale des centres de gravités

Problème : Nous voulons trouver le bipartitionnement qui maximise la distance des deux centres de gravités des deux groupes de points.

Posons : $\sum_{i=1}^{n} V_1(i) = d_1$ $\qquad$ $\sum_{i=1}^{n} V_2(i) = d_2$

Nous avons :

$$d_1 + d_2 = n$$

G_1 désignera le centre de gravité du premier groupe de points (G_2 le second).

Les coordonnées de G_1 seront donc :

$$\frac{V_1^T B_1}{d_1} \quad \text{et} \quad \frac{V_1^T B_2}{d_1}$$

et nous avons des formules identiques pour G_2 .

Le carré de la distance euclidienne de G_1 à G_2 sera donc :

$$\left(\frac{V_1^T B_1}{d_1} - \frac{V_2^T B_1}{d_2}\right)^2 + \left(\frac{V_1^T B_2}{d_1} - \frac{V_2^T B_2}{d_2}\right)^2$$

Le problème posé est donc un problème de distance euclidienne maximale

$$\max_{X \in \mathcal{D}} X^T X$$

$$\mathcal{D} = \left\{ X \in \mathbb{R}^2 : \begin{array}{ll} X(1) = \dfrac{V_1^T B_1}{d_1} - \dfrac{V_2^T B_1}{d_2} & \\ & (V_1, V_2) \text{ vérifiant les} \\ X(2) = \dfrac{V_1^T B_2}{d_1} - \dfrac{V_2^T B_2}{d_2} & \text{contraintes (C)} \end{array} \right\}$$

La résolution de ce problème par les méthodes exposées au chapitre III sera donc tributaire des problèmes projetés :

$$\max_{(V_1,V_2)} \left(\frac{V_1^T B_1}{d_1} - \frac{V_2^T B_1}{d_2}\right)\cos\theta + \left(\frac{V_1^T B_2}{d_1} - \frac{V_2^T B_2}{d_2}\right)\sin\theta$$

Posons : $V_1 + V_2 = V$ $\qquad V(i) = 1$ $\qquad i=1,2,\ldots,n$

$V_1 - V_2 = K$.

L'expression à maximiser est donc :

$$\frac{(K+V)^T B_1 \cos\theta}{2 d_1} - \frac{(V-K)^T B_1 \cos\theta}{2 d_2} + \frac{(V+K)^T B_2 \sin\theta}{2 d_1} - \frac{(V-K)^T B_2 \sin\theta}{2 d_2}$$

d'où :

$$K^T \left[\left(\frac{\cos\theta}{2 d_1} + \frac{\cos\theta}{2 d_2}\right)B_1 + \left(\frac{\sin\theta}{2 d_1} + \frac{\sin\theta}{2 d_2}\right)B_2\right]$$

$$+ V^T B_1 \left(\frac{\cos\theta}{2 d_1} - \frac{\cos\theta}{2 d_2}\right) + V^T B_2 \left(\frac{\sin\theta}{2 d_1} - \frac{\sin\theta}{2 d_2}\right)$$

d_1 et d_2 étant fixés le problème sera donc équivalent à la maximisation de :

$$K^T \left[\frac{\cos\theta}{2 d_1 d_2} \cdot n B_1 + \frac{\sin\theta}{2 d_1 d_2} \cdot n B_2\right]$$

et se résoudra à l'aide d'un tri des composantes du vecteur :

$$B_1 \cos\theta + B_2 \sin\theta .$$

La constante multiplicative $\frac{n}{2 d_1 d_2}$ ne changeant rien.

Ce tri étant donc le même pour toutes les valeurs de d_1 et d_2 il sera facile d'énumérer les solutions correspondant à chaque valeur du couple (d_1,d_2) et de choisir ensuite celle qui maximise le critère du problème projeté.

Nous pourrons donc appliquer avec quelques petites modifications évidentes les algorithmes du chapitre III.

2.2.3. Minimisation d'un critère "inter-classes"

Problème : Nous désirons maintenant minimiser la fonction :

$$\sum_{X_i \in \text{Groupe } 1} \varphi_2^2(X_i - G_1) + \sum_{X_i \in \text{Groupe } 2} \varphi_2^2(X_i - G_2)$$

où φ_2 désigne la norme euclidienne usuelle sur $\mathbb{R}^2$.

Nous allons voir que ce problème est très voisin de ceux déjà étudiés.

Nous introduirons deux matrices diagonales D_1 et D_2 telles que :

$$D_1(i,i) = V_1(i)$$

$$D_2(i,i) = V_2(i) \qquad i=1,2,\ldots,n$$

En ce qui concerne les points du premier groupe nous avons :

$$\varphi_2^2(X_i - G_1) = (X_i(1) - \frac{V_1^T B_1}{d_1})^2 + (X_i(2) - \frac{V_1^T B_2}{d_1})^2$$

Introduisons le vecteur :

$$D_1 B_1 - \frac{V_1^T B_1}{d_1} V_1$$

Sa i^e composante est nulle si X_i n'appartient pas au premier groupe de points et sinon égale à :

$$B_1(i) - \frac{V_1^T B_1}{d_1} = X_i(1) - \frac{V_1^T B_1}{d_1}$$

La partie du critère correspondant aux points du premier groupe est donc :

$$(D_1B_1 - \frac{V_1^T B_1}{d_1} V_1)^T(D_1B_1 - \frac{V_1^T B_1}{d_1} V_1) + (D_1B_2 - \frac{V_1^T B_2}{d_1} V_1)^T(D_1B_2 - \frac{V_1^T B_2}{d_1} V_1) .$$

Développons le premier facteur :

$$B_1^T D_1^T D_1 B_1 - B_1^T D_1^T V_1 \cdot \frac{V_1^T B_1}{d_1} - V_1^T D_1 B_1 \frac{V_1^T B_1}{d_1} + (\frac{V_1^T B_1}{d_1})^2 V_1^T V_1$$

donc :

$$B_1^T D_1 B_1 - 2 \frac{(V_1^T B_1)^2}{d_1} + \frac{(V_1^T B_1)^2}{d_1} = B_1^T D_1 B_1 - \frac{(V_1^T B_1)^2}{d_1}$$

En écrivant le second facteur et les termes analogues pour le second groupe le critère s'exprime alors par :

$$B_1^T D_1 B_1 - \frac{(V_1^T B_1)^2}{d_1} + B_2^T D_1 B_2 - \frac{(V_1^T B_2)^2}{d_1} + B_1^T D_2 B_1 - \frac{(V_2^T B_1)^2}{d_2}$$

$$+ B_2^T D_2 B_2 - \frac{(V_2^T B_2)^2}{d_2}$$

La matrice $D_1 + D_2$ étant la matrice identité nous avons donc :

$$B_1^T B_1 + B_2^T B_2 - [\frac{(V_1^T B_1)^2 + (V_1^T B_2)^2}{d_1} + \frac{(V_2^T B_1)^2 + (V_2^T B_2)^2}{d_2}]$$

Le problème de minimisation du critère précédent est donc équivalent au problème suivant :

$$\underset{(V_1,V_2)}{\text{Max}} \frac{(V_1^T B_1)^2 + (V_1^T B_2)^2}{d_1} + \frac{(V_2^T B_1)^2 + (V_2^T B_2)^2}{d_2}$$

Si l'on pose $V_1 + V_2 = V$

$V_1 - V_2 = K$

après calcul le critère devient :

$$\frac{1}{n}[(V^T B_1)^2 + (V^T B_2)^2] + \frac{n}{4}[K^T B_1 \times \sqrt{\frac{1}{d_1 d_2}} + V^T B_1 \frac{d_2 - d_1}{n\sqrt{d_1 d_2}}]^2$$

$$+ \frac{n}{4}[K^T B_2 \times \sqrt{\frac{1}{d_1 d_2}} + V^T B_2 \frac{d_2 - d_1}{n\sqrt{d_1 d_2}}]^2$$

Finalement, nous sommes ramené au problème de distance euclidienne maximale ou l'ensemble $\mathcal{D}$ est :

$$\mathcal{D} = \left\{ X \in \mathbb{R}^2 \ : \ X(i) = K^T B_i \sqrt{\frac{1}{d_1 d_2}} + V^T B_i \frac{d_2 - d_1}{n \sqrt{d_1 d_2}} \quad \begin{matrix} K = V_1 - V_2 \\ (V_1, V_2) \text{ vérifiant } C \\ i=1,2 \end{matrix} \right\}$$

La remarque importante est que les problèmes projetés se résolvent de la même manière que ceux exposés au paragraphe précédent car le tri a effectué est celui du vecteur :

$$B_1 \cos \theta + B_2 \sin \theta$$

indépendant de d_1 et d_2 ce qui permet de résoudre aisément ce problème.

2.3. AUTRES POSSIBILITES

Il est évident que les problèmes précédent peuvent se traiter dans $\mathbb{R}^p$ avec des matrices

$$\sum_{i=1}^{P} B_i B_i^T \qquad B_i \in \mathbb{R}^n \quad i=1,2,\ldots,p$$

De même des interprétations voisines peuvent être données aux problèmes de partitionnements en plus de deux parties.

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

INTRODUCTION

[1] DENNING, P.J. : "Virtual memory"
Computing Surveys, Vol.2, n° 3, (sept. 1970) 153-190.

[2] DENNING, P.J. : "The working set model for program behavior"
Com. ACM, Vol.11, n° 5 (may 1968) 323-333.

[3] COMMEAU, L.W. : "A study of the effect of user program optimization in a paging system"
ACM Symp. on operating system principles, Gatlinburg,Tenn.(1967).

[4] HATFIELD, D.J. and GERALD, J.
"Program restructuring for virtual memory"
IBM Systems Journal, Vol. 10, n° 3 (1971) 168-192.

[5] JOHNSON, J.W. : "Program restructuring for virtual memory systems".
MIT Project MAC, (march 1975), Cambridge Massachusetts 02139.

[6] KARP : "On the computational complexity of combinatorial problems" Networks, 5, (1975).

CHAPITRE I

[1] BELADY, "A study of remplacement algorithms for a virtual-storage computer."
IBM Systems Journal, Vol. 5, n° 2 (1966).

[2] BOGOTT, R.B. and FRANKLIN, M.A.
"Evaluation of markov program models in a virtual memory systems."
Software Practice and Experience, Vol. 5, (1975) 337-346.

[3] DENNING, P.J. : "Virtual memory"
Computing Surveys, Vol. 2, n° 3, (sept. 1970) 153-189.

[4] FRANKLIN, M.A. and GUPTA, R.K.
"Computation of page fault probability from program transition diagram".
C. ACM (april 1974), Vol. 17, number 4.

[5] GECSEI, J. SLUTZ, D.R. and TRAIGER, J.L.
"Evaluation techniques for storage hierarchies "
IBM Systems Journal, n° 2, (1970).

[6] HATFIELD, D.J. : "Experiments on page size, program access patterns and virtual memory performance"
IBM J. Res. Develop. (january 1972)

[7] HATFIELD, D.J. and GERALD, J.
"Program restructuring for virtual memory"
IBM Systems Journal, n° 3, (1971).

[8] JOHNSON, J.W. : "Program restructuring for virtual memory systems"
MIT Project MAC, (march 1975),
Cambridge Massachusetts 02139.

CHAPITRE II

[1] ASPINALL : "Data-base reorganisation algorithms"
IBM, UKSC-0029 (february 1972).

[2] BAUER, F.L. : "Das Verfahren der Treppeniteration und werwandte Verfahren zur Lösung algebraischer Eeigenwert -probleme"
Z. Angew. Math.Phys., 8, (1957) 214-235.

[3] BRIANE, M. : "L'algorithme d'échange en classification automatique". Classification automatique et perception par ordinateur. Séminaire IRIA, (1973).

[4] GANTMACHER : "Théorie des matrices, tome 1 : Théorie générale".
Dunod, Paris (1966).

[5] FIEDLER, M. et PTAK, V. :
"Sur la meilleure approximation de transformations linéaires par des transformations linéaires de rang prescrit"
CRAS, T 254, n° 22, (1962).

[6] GAREY, JOHNSON, STOCKMEYER :
"Some simplified NP complete problems"
6e Symposium ACM on theory of computing (1974).

[7] HATFIELD, D.J. and GERALD, J.
"Program restructuring for virtual memory".
IBM System Journal, n° 3, (1971).

[8] HYAFIL, RIVEST : "Graph partitionning and constructing optimal decisions trees are polynomial complete problems".
IRIA, Rapport de recherche n° 33, (1973).

[9] KARP : "On the computational complexity of combinatorial problems." Networks, 5, (1975).

[10] KERNIGHAN, B.W. and LIN, S. :
"An efficient procedure for partitioning graphs".
Bell. System Technical Journal 49.2, (february 1970).

[11] LACOLLE, B. : "Aspects pratiques et théoriques du problème du partitionnement".
Séminaire d'Analyse Numérique de Grenoble, n° 211.

[12] NARSHING Deo : "Graph theory with applications to engineering and computer science ".
Prentice Hall, series in Automatic computation

[13] RUTISHAUSER, H. : "Computational aspectsof F.L. Bauer's simultaneous iteration method.
Numer. Math. 13, 4-13, (1969).

[14] WILKINSON-REINSCH:"Handbook for automatic computation"
Volume II, linear algebra, Springer-Verlag.

CHAPITRE III

[1] COURTIN-VOIRON : "Introduction à l'algorithmique et aux structures de données. Ch. G : Les tris. I.U.T. B de Grenoble, département Informatique (1974-75).

[2] KNUTH : "The art of computer programming" Vol. 3 : Sorting and searching, Addison-Wesley.

[3] LACOLLE, B. : "Aspects théoriques et pratiques du problème du partitionnement". Séminaire d'Analyse Numérique de Grenoble, n° 211.

[4] LACOLLE, B. : "Bi-partitionnement d'une matrice somme de deux antiscalaires symétriques". Séminaire d'Analyse Numérique de Grenoble n° 232.

CHAPITRE IV

[1] DONATH, W.E. and HOFFMANN, A.J. : "Lower bounds for the partitionning of graph". IBM J. Res. Develop, (sept. 1973).

[2] HOFFMANN, A.J. and WIELANDT, H.W. : "The variation of the spectrum of a normal matrice" Duke Math, J. 20, 37 (1973).

[3] SALKIN : "Integer programming" Addison Wesley. Londres (1975).

CHAPITRE VI

[1] KARP : "On the computational complexity of combinatorial problems." Networks, 5 (1975).

[2] KOONTZ, NARENDRA, FUKUNGA :
"A branch and bound clustering algorithm".
I.E.E.E. Transaction on computers, Vol. C.24, n° 9, (sept. 1975).

[3] SAHNI, S. : "Approximate algorithms for the 0/1 knapsack problem".
J. ACM, Vol. 22, n° 1, (january 1975) 115-124.

[4] SALKIN : "Integer programming".
Addison Wesley, Londres (1975).

CHAPITRE V

[1] DAY, W.H.E. : "Compiler assignement of data items to registers". IBM Syst. Journal, n° 4, (1970).

[2] GASTINEL, N. : "Analyse Numérique linéaire". Hermann

[3] GUERTIN : "Programming in a paging environnement." Datamation (february 1972).

[4] HORWITZ, L.P. ; KARP, R.M. ; MILLER, R.E. and WINOGRAD, S. : "Index register allocation" J. ACM, Vol. 13, n° 1, (january 1966) 43-61.

[5] IBM : "Fortran IV language" "IBM System / 360" "IBM System / 370" G C 28 - 6515 - 8

[6] MacKELLAR, COFFMANN, G. Jr. : "Organizing matrices and matrix operations for paged memory system". C ACM, Vol. 12, n° 3 (march 1969).

[7] LAFON, J.C. : "Influence de la pagination sur la rapidité d'exécution des algorithmes". Séminaire d'Analyse Numérique de Grenoble, n° 190.

[8] DU MANOIR, H. et MARTIN, P. : "Réduction du volume des microprogrammes par une méthode d'optimisation de l'allocation des registres". R.A.I.R.O., B-1, (février 1974) 19-40.

[9] SINGLETON, R.C. : "On computing the fast fourier transform". C. ACM, Vol. 10, number 10, (oct. 1967).

Dernière page d'une thèse

VU

Grenoble, le

Le Président de la thèse

Vu, et permis d'imprimer,

Grenoble, le

Le Président de l'Université
Scientifique et Médicale