Université de Neuchâtel

Faculté de Droit et des Sciences économiques

Analyse critique des méthodes classiques et nouvelle approche par la programmation mathématique en classification automatique

Thèse

présentée à la Faculté de Droit et des Sciences économiques pour obtenir le grade de docteur ès sciences économiques

par

Thierry Gafner

Imprimerie de l'Evole, Neuchâtel

Monsieur Thierry GAFNER est autorisé à imprimer sa thèse de doctorat ès sciences économiques intitulée "Analyse critique des méthodes classiques et nouvelle approche par la programmation mathématique en classification automatique".

Il assume seul la responsabilité des opinions énoncées.

Neuchâtel, le 18 décembre 1991

Le Doyen
de la Faculté de droit
et des sciences économiques

Henri-Robert Schüpbach

Table des matières

Remerciements

Cet ouvrage représente l'aboutissement d'une étude réalisée aux Groupes d'Informatique et de Statistique de l'Université de Neuchâtel (Suisse). J'adresse mes sincères remerciements au Professeur Yadolah Dodge, premier rapporteur, pour ses conseils en matière de statistique, et au Professeur Paul Schönsleben, co-rapporteur, pour son soutien apporté dans la résolution des problèmes informatiques.

Mes remerciements vont aussi aux chercheurs et Professeurs associés au Postgrade en Statistique, qui ont eu l'amabilité d'examiner ce travail et de me faire part de leurs conseils et remarques; plus particulièrement le Professeur Rousseeuw de Bruxelles pour la mise à disposition de son programme PAM, pour le temps consacré à la relecture du manuscript et pour les remarques qui ont permis son amélioration, ainsi que le Professeur Diday et M. Lechevallier de l'INRIA.

A Carole, pour son soutien moral et la motivation qu'elle a su m'apporter dans les périodes de doute.

Résumé

Au cours de cette étude, nous avons cherché à montrer par la simulation que seules les méthodes de la programmation mathématique permettent d'obtenir une solution optimale en classification automatique et que les méthodes classiques ne fournissent qu'une approximation. Forts de ce résultat, nous avons tenté de réduire la complexité d'une des premières méthodes afin d'obtenir des temps de calcul raisonnables, sans pour autant perdre en efficacité. Pour obtenir cette réduction, nous avons combiné les points forts de trois méthodes, de telle sorte que les calculs ne se fassent qu'en des points représentatifs de la fonction-objectif.

Abstract

With this study we showed in a first phase using intensive simulations, that only the mathematical programing methods could find an optimal solution to a classification problem; and that the classical methods only could find an approximation. In a second phase, we reduced the complexity of the dynamic programing algorithm by the combination of the forces taken in three methods.

Chapitre 1

Introduction

1.1 Définition du problème, son origine, son évolution, buts et articulation du travail

Nous allons examiner des méthodes résolvant le problème suivant :

Soit un ensemble N={1,2,...,n} d'entités ou objets dont nous connaissons jusqu'à k caractéristiques. L'objectif est de regrouper les entités étudiées dans *m* groupes homogènes. Afin de permettre ce regroupement, nous considérons que les entités se distinguent par une *distance*.

Ce problème fut abordé très tôt par de nombreux scientifiques de toutes branches, comme le montre la perspective historique ci-dessous. Le développement des méthodes de résolution fut plus pragmatique que mathématique jusqu'au début des années soixante.

Aristote et les philosophes antiques furent parmi ces précurseurs qui cherchèrent à classer formellement plantes et animaux. Leurs idées étaient fort simples. Il existait, par exemple, quatre états de la matière : l'eau, l'air, la terre, le feu. Par un choix approprié de caractéristiques, ils développèrent des typologies où l'on retrouvait des *dominances semblables* des quatre états de base. Galen (129-199) définit l'une des plus connues qui permettait grâce à neuf types de tempéraments de trouver la vulnérabilité d'un individu à diverses maladies ou d'expliquer les différences de comportement en société.

Le système moderne, dans les sciences naturelles, est dû aux travaux de Linnée (1753). Après la zoologie et la botanique, la classification fut appliquée à bien d'autres domaines : psychologie, médecine, psychiatrie, archéologie, anthropologie, marketing, sciences économiques, sociologie, éducation, criminologie,... Longtemps ces techniques demeurèrent limitées à la définition de typologies fixées par des critères observables visuellement.

L'application des mathématiques connut un développement important au cours du 19e siècle. Les méthodes de regroupement en profitèrent également. Le développement des méthodes de classification resta longtemps l'oeuvre des scientifiques qui en avaient besoin. Par exemple, le professeur Robert Tryon, professeur de psychologie à l'Université de Berkeley, consacra l'essentiel de sa carrière à développer des techniques applicables à la psychologie individuelle et collective, et les utilisa dans ses études.

Les méthodes proposées par ces scientifiques sont sujettes à la même critique: aucune ne garantit une classification mathématiquement correcte. Elles sont trop rudimentaires, à l'image des moyens de calcul disponibles. Le développement d'une méthode de classification exige des connaissances

poussées dans divers domaines. Brian Everitt (1974) écrit en conclusion à un ouvrage consacré à ces méthodes : "The number of clustering techniques available is large, as the number of problems in applying them. The greater part of the mushrooming literature on classification is concerned with new techniques which in general suffer from many of the problems of those methods already in use. Perhaps the problem is that mentionned by Johnson (Rainbows' End : the quest for an optimal taxonomy, Proc. Linn. Soc. N.S.W. 93, 8-45, 1968), namely that anyone who is prepared to learn quite a deal of matrix algebra, some classical mathematical statistics, some advanced geometry, a little set theory, perhaps a little information theory and graph theory and some computer techniques and who has access to a good computer and enjoys mathematics (as he must if he gets this far) will probably find the development of new taximetric methods much more rewarding, more up-to-date, more 'general', and hence more prestigious than merely classifying plants and animals.".

Dès les années 60, des statisticiens confirmés commencèrent à s'intéresser à cette problématique. De nombreux ouvrages parurent, cherchant à formaliser les développements antérieurs, à établir de nouvelles méthodes plus fiables, en apparence du moins, et à tirer toujours plus de profits des ordinateurs. Mais elles souffrent toujours du même défaut que les précédentes. Aucune de ces méthodes ne converge.

Pour corriger ce défaut, quelques mathématiciens ou statisticiens ont étudié les possibilités d'appliquer diverses formulations issues de la programmation mathématique à la résolution d'un problème de classification automatique. Ces quelques études n'ont rencontré que peu d'échos en raison des problèmes de complexité des algorithmes et des coûts de leur mise en oeuvre.

Pourquoi un nouvesu travail dans ce domaine qui semble si bien couvert théoriquement ? Et pourquoi chercher à remettre au goût du jour des techniques jugées trop coûteuses ?

- Premièrement, nous avons pu nous rendre compte lors d'un congrès en 1987 à Aix-La-Chappelle, qu'aucune solution n'a été trouvée pour supprimer les défauts des méthodes classiques. **Dans ce travail nous appelons *méthodes classiques* toutes celles qui ne font pas appel à un *algorithme de programmation mathématique*.** Ces dernières semblent être ignorées. Aucun exposé ne leur était consacré dans cette conférence et nous n'avons trouvé aucun logiciel les appliquant.

- Deuxièmement, nos recherches bibliographiques nous ont prouvé que personne n'avait réalisé une étude comparative approfondie des diverses méthodes disponibles. Seuls, à notre connaissance, Everitt, Cooper et Milligan, ont montré les problèmes posés principalement par les méthodes hiérarchiques.

- Troisièmement, nous sommes convaincus que seules les méthodes issues de la recherche opérationnelle, permettent d'obtenir une solution correcte *mathématiquement*. Nous pensons qu'il est nécessaire d'examiner ces techniques sous un autre oeil que simplement celui de la performance.

Afin d'atteindre ces objectifs, nous effectuons une étude comparative en plusieurs volets :

- Premièrement, nous exposons les éléments conceptuels des algorithmes. Cette phase montrera quels sont les critères de classification utilisés et l'existence ou non d'une fonction-objectif globale à minimiser (ou maximiser). Chaque exposé est illustré par un exemple.

- Deuxièmement, nous cherchons à montrer la complexité des méthodes et les problèmes qu'elles posent mathématiquement.

- Troisièmement, nous présentons les problèmes rencontrés lors d'une implantation informatique. Nous chercherons également à juger le confort d'utilisation selon des critères d'ergonomie.

- Qustriémement, nous exposons deux méthodes n'utilisant pas la notion de distance et en comparons les concepts svec ceux de la classification automatique.

- Cinquièmement, nous présentons les résultata obtenus sur des jeux de données conçua de telle sorte que la classification optimale est connue.

- Finalement noua examinons les possibilités de développement à partir des meilleures méthodes et tirons les enseignements de cette étude.

1.2 Domaines d'application

L'étendue des domaines d'spplication, et des dénominstions, se compare aisément à la variété des solutions proposées. Il existe de très nombreuses méthodes de classification, qui s'utilisent en zoologie, en botanique, en médecine, en économie, en sociologie, en psychologie...

Tryon et Bailey (1965) donnent de nombreux exemples pratiques, issus de leurs propres recherches. Dans Hartigan (1975), Everitt (1976), Späth (1983), nous trouvons de nombreuses références à des études où la classification sutomatique s été utilisée.

La médecine, ls biologie et les sciences sociales semblent être les domaines de prédilection pour ce genre d'étude. A Aix-La-Chappelle de nombreuses sessions étaient consacrées aux applicstions dans ces branches.

Pourquoi cette technique statistique est-elle si répandue ? Pour tenter de répondre à cette question, nous allons examiner les raisons de regrouper des données :

- Le scientifique qui procède à une analyse statistique peut se trouver confronté à des problèmes techniques : capacité insuffisante de l'ordinateur, limites des logiciels, temps nécessaire pour opérer les calculs, saisir les données et exploiter les résultats... La classificstion peut servir à réduire le nombre des données à analyser, en n'étudiant que des cas *types*.

- Un sutre aspect consiste à classer les traitements pour voir s'ils suivent une loi statistique ou s'adaptent à un modèle déjà connu.

- Dans le domaine des sciences sociales, on s'intéresse à l'établissement de catégories des individus observés. Par exemple, dans un sondage d'opinion, on essaie de savoir si les partisans d'un leader politique ont des caractéristiques sociales communes.

- Enfin, à partir des données réunies et classifiées lors d'une première étude, on peut chercher à prévoir le comportement d'un individu en

fonction des ses caractéristiques; c'est-à-dire trouver à quel groupe de la précédente étude, il appartient.

Pour résumer, nous pouvons dire à l'exemple de Ball [1971], cité par Everitt [1974], que la classification doit permettre d'atteindre un des objectifs suivants :

- Trouver une vraie typologie.
- Ajustement à un modèle.
- Prédiction basée sur les groupes.
- Tests d'hypothèses.
- Exploration des données.
- Génération d'hypothèses.
- Réduction des données.

Chapitre 2

Les méthodes classiques de classification

Les techniques de classification automatique utilisent pour la plupart une *distance* pour mesurer les dissimilarités entre les objets à classer. Certaines peuvent également utiliser une mesure de similarité. Ces deux notions font l'objet de la première section de ce chapitre. Ensuite, nous exposons les fondements mathématiques des méthodes classiques et les différentes typologies qui permettent de les distinguer. Notre exposé sera illustré par un exemple.

Nous empruntons notre exemple à la gestion du personnel. Il s'agit d'un extrait supposé des feuilles de qualification des employés d'une entreprise. Ces qualifications sont données sous forme de notes de 1 (excellent) à 6 (très insuffisant).

Nom / qualification	Ponctualité	Précision	Contact	Qualité	Responsabilité
Dupont	2.5	3	4	5	2
Müller	6	5	4	5	5
Meyer	2	3	3	4	4
Durand	5	6	5	5	5
Scott	2	3	3	2	1
Henry	5	5	5	4	6

Tableau 2.1 : Notes de qualification

2.1 La notion de distance dans les méthodes classiques et les problèmes posés par les données

Nous abordons les problèmes liés aux données : la mesure de leur différences ou de leur ressemblance, leur unité de mesure, les divers types de données qui peuvent se présenter dans une analyse statistique. Ces problèmes sont exposés, afin de montrer que les défauts des méthodes de classification sont dûs à leurs concepts, et non pas à des influences extérieures.

2.1.1 Mesures de la dissimilarité entre entités

Les entités peuvent être distinguées les unes des autres par le calcul d'une distance mathématique. Avant de calculer une distance, il peut être nécessaire de procéder préalablement à une transformation des données, afin de corriger par exemple un effet de taille dû à des unités de mesures différentes, ou de codifier un ensemble de données de natures diverses. Ces divers problèmes et leurs solutions sont expliqués en détail dans les sections suivantes. De plus, une standardisation des données permet de supprimer les effets négatifs de certaines distances.

Une distance mathématique, entre une paire d'objets doit avoir les propriétés suivantes :

1) Elle doit être positive :

$$d(X_i, X_j) \geq 0$$

2) Elle doit être commutative :

$$d(X_i, X_j) = d(X_j, X_i)$$

3) Elle est nulle en un point i donné :

$$d(X_i, X_j) = 0, \text{ si } i = j$$

4) L'inégalité triangulaire doit être vérifiée pour obtenir une distance "métrique" :

$$d(X_i, X_j) \leq d(X_i, X_k) + d(X_k, X_j)$$

De nombreuses métriques répondent à ces critères. La plus naturelle est la distance euclidienne. Elle correspond à une notion connue : celle de distance entre les extrémités d'un segment. Elle est donnée par la formule ci-dessous :

$$d_2(X_i, X_j) = \left(\sum_{k=1}^{n} (x_i^k - x_j^k)^2\right)^{\frac{1}{2}} \qquad (2.1)$$

Une distance peut être pondérée. Dans ce cas, la distance euclidienne s'exprime par :

$$d_2(X_i, X_j) = \left(\sum_{k=1}^{n} w_k (x_i^k - x_j^k)^2\right)^{\frac{1}{2}} \qquad (2.2)$$

Nous ignorons par la suite cette notion. Cet aspect n'est pas à négliger en pratique. L'une ou l'autre des observations recueillies a une importance plus grande que d'autres. Ou inversement, une observation ne doit pas pouvoir prendre une prépondérance particulière dans les résultats. La pondération [positive ou négative] permet de tenir compte de telles influences dans le calcul d'une distance.

Usuellement, la matrice des distances est représentée sous forme de triangle, car elle est par définition symétrique. Pour notre exemple, nous obtenons la matrice suivante avec la distance euclidienne :

0.0	5.02	2.50	5.02	3.35	5.31
	0.0	4.79	1.73	6.78	2.00
		0.0	4.89	3.60	4.58
			0.0	6.85	1.73
				0.0	6.78
					0.0

La distance entre Dupont et Müller vaut 5.02. Elle est obtenue par le calcul suivant :

$$d_2(\text{Dupont,Müller}) = [(2.5-6)^2 + (3-5)^2 + (4-4)^2 + (5-5)^2 + (2-5)^2\ ]^{1/2} = 5.02$$

Cette distance n'est pas la panacée universelle. Si elle est mal utilisée, les conséquences sur les résultats sont catastrophiques; car elle tient compte de la variance de chacune des observations. Elle est sensible à l'effet de taille; c'est-à-dire que si toutes les observations d'un traitement sont multipliées par un facteur donné, toutes les distances de ce traitement aux autres sont également multipliées par ce facteur. Enfin, elle dépend de l'unité de mesure.

Une autre distance couramment utilisée est la distance du chi-carré.

$$d\ (X_i,X_j) = \sum_{k=1}^{n} \left| \frac{x_i^k}{x_i} - \frac{x_j^k}{x_j} \right|^2 \frac{x}{x^k} \qquad (2.3)$$

où $x_i = \sum_k x_i^k$

$x^k = \sum_i x_i^k$

$x = \sum_i \sum_k x_i^k$

Elle ne peut être utilisée que si les observations sont mesurées avec une même échelle et une même unité de mesure. Par contre, elle fait disparaître l'effet de taille de la distance euclidienne.

Afin d'éviter l'effet variance de la distance euclidienne, nous disposons d'autres armes : les distances de Minkovski d'ordre 1 (norme L_1). Elles privilégient les petites distances. La plus connue est la distance de Manhattan ou "city block metric" donnée par :

$$d(X_i,X_j) = \sum_{k=1}^{n} |x_i^k - x_j^k| \qquad (2.4)$$

La distance de Manhattan entre Dupont et Müller est obtenue par :

$$d_1(\text{Dupont,Müller}) = |2.5-6| + |3-5| + |4-4| + |5-5| + |2-5| = 8.50$$

Pour notre exemple, nous obtenons :

0.0	8.50	4.50	9.50	5.50	10.50
	0.0	9.00	3.00	14.00	4.00
		0.0	10.00	5.00	9.00
			0.0	15.00	3.00
				0.0	14.00
					0.0

Une variante est connue sous le nom de distance de Camberra :

$$d(X_i,X_j) = \sum_{k=1}^{n} |x_i^k - x_j^k| / (|x_i^k| + |x_j^k|) \qquad (2.5)$$

S'il s'avère nécessaire d'utiliser une méthode robuste pour calculer la matrice des dissimilarités, la distance basée sur L^∞ est la plus appropriée car elle privilégie la plus forte distance parmi toutes :

$$d(X_i,X_j) = \max_{k=1,n} |x_i^k - x_j^k| \qquad (2.6)$$

La distance entre Dupont et Müller est donnée par la distance maximale entre leurs paires d'observations, max[(2.5-6);(3-5);(4-4);(5-5);(2-5)], soit 3.5

Matrice des distances obtenues avec L infini :

0.0	3.50	2.00	3.00	3.00	4.00
	0.0	4.00	1.00	4.00	1.00
		0.0	3.00	3.00	3.00
			0.0	4.00	1.00
				0.0	5.00
					0.0

Une autre mesure de la dissimilarité peut être donnée par la distance de Mahanalobis :

$$d(X_i,X_j) = (X_i-X_j)' \sum_z^{-1} (X_i-X_j) \qquad (2.7)$$

avec $\sum_z$ la matrice variante-covariante des observations étudiées.

Matrice des distances obtenues avec Mahanalobis :

0.0	3.16	3.16	3.16	3.16	3.16
	0.0	3.16	3.16	3.16	3.16
		0.0	3.16	3.16	3.16
			0.0	3.16	3.16
				0.0	3.16
					0.0

Cette matrice montre que cette distance ne peut pas s'utiliser sans précautions. Si le nombre d'objets est égal au nombre de variables plus un, comme c'est le cas ici, toutes les distances seront égales et il n'y a paa de aens à

effectuer une classification.

2.1.2 Mesure de la ressemblance

Au lieu de calculer les différences entre les entités, nous pouvons obtenir une image de leur ressemblance. Elle est déterminée par un coefficient de similarité. Un tel coefficient mesure la relation entre deux entités, étant donné un ensemble de p observations communes. La présence ou l'absence d'une association aur ces observations peut être représentée par des variables binaires, 0 et 1. Il existe divers coefficients de similarité (diverses formules sont données dans Everitt (1974)). L'un des plus connus est le coefficient de Gower (1971). Il est possible de l'utiliser pour des données mixtes, soit un mélange de données quantitatives, qualitatives et binaires. Il est défini de la façon suivante :

$$S_{ij} = \sum_{k=1}^{p} s_{ijk} \Big/ \sum_{k=1}^{p} w_{ijk} \qquad (2.8)$$

où w_{ijk} : le poids prenant la valeur 0 ou 1 selon que la comparaison est valable pour la variable k. Si $w_{ijk} = 0$, alors $s_{ijk} = 0$, et si $w_{ijk} = 0$ pour tous i, j et k, alors S_{ij} est indéfini.

s_{ijk} : le score de chaque entité pour la variable k.

Nous pourrions discuter longtemps de ce problème du choix de la mesure entre les entités : ressemblance ou différence. Il convient de se rendre compte qu'il existe certaines difficultés à l'opérer. Ces problèmes sont toutefois indépendants du choix de la méthode de classification. Les quelques critiques émises sur les critères présentés dans cette section sont valables pour *toute méthode de classification.*

2.1.3 La problématique de la nature des données

Un autre problème commun à toutes les méthodes de classification est la nature des données. Nous pouvons distinguer les données quantitatives (numériques pures), binaire (0 et 1, vrai ou faux) et qualitatives (données ordinales et nominales). La difficulté est corsée par le fait que bien des études statistiques présentent des données mixtes. Les défauts des méthodes, que nous cherchons à expliciter, ne sont pas liés à cette problématique.

Il existe diverses techniques de recodification des données. Certaines produisent un appauvrissement de l'information, d'autres un enrichissement. L'idée générale de ces méthodes est de transformer une variable de nature donnée en une autre de nature différente, qui se prête mieux à l'analyse statistique. En sciences sociales, on considère des données entières qui ont une *valeur morale.* Il s'agit en réalité de données qualitatives recodées en données quantitatives au travers d'une échelle.

2.1.4 Le problème de l'unité de mesure

Ce problème est particulier aux données quantitatives, qui peuvent être mesurées dans divers aystèmes. Souvent l'échelle n'est par elle-même pas comparable. Est-il possible de comparer des données mesurées en mm avec d'autres mesurées en km sans prendre de précaution ?

Il faut transformer la matrice des données avant tout calcul. Cette opération a'appelle *normalisation* ou *standardisation* des données. Deux techniques sont possibles : centrer les données, c'est-à-dire ramener la moyenne de toutes les données à 0; centrer et réduire, soit ramener la moyenne des données à 0 et leur variance à 1. Cette problématique n'influence pas les méthodes que nous examinons.

2.2 Typologie des méthodes de classification automatique classique et leur fondements mathématiques

Diverses typologies des méthodes de classificstion automatique ont été définies. Nous nous réfèrons à celle utilisée par Everitt (1974).

2.2.1 Les méthodes de classification hiérarchique

Il existe deux familles de méthodes hiérarchiques. Elles se singularisent par leur point de départ : soit au début les n entités forment n groupes et l'on va chercher à les fusionner; soit un groupe formé des n entités est donné et il faut chercher à le diviser. Les premières de ces méthodes sont dites ascendantes, les secondes descendantes.

Elles sont présentées au moyen d'un graphique : le dendogramme, montrant dans quelle séquence et à quelle étape, les diverses classifications intermédiaires ont eu lieu.

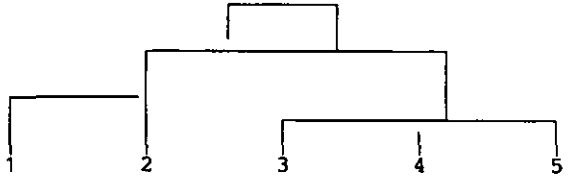


Figure 2.1 : Exemple de dendogramme

2.2.1.1 Les méthodes hiérarchiques ascendantes

Soit au départ n groupes à une entité, et une mesure de leur distance, respectivement de leur similarité, noua regroupons les deux entités les plus proches conformément à un critère pour former le premier sous-groupe.

A partir des n-1 sous-groupes ainsi formés, noua recalculons la métrique et fusionnons à nouveau deux entités ou sous-groupes. Le processus s'arrête s'il ne reste qu'un groupe, ou si un critère d'arrêt optionnel a été atteint.

Ces méthodes sont très répandues. Nous pouvons nommer les plus connues : méthode de Ward, méthode du chaînage simple, méthode du chaînsge complet, méthode de la distance moyenne dans les groupes, méthode de la distance entre les groupes, ...

2.2.1.2 Les méthodes hiérarchiques descendantes

Soit au départ un groupe formé des n entités, il s'agit de le partager en deux. Le processus se répète jusqu'à obtenir n groupes formés chacun de une entité, ou si le critère d'arrêt de la méthode est sstisfait.

En raison de particularités, qui les distinguent trop des autres méthodes sélectionnées, nous n'examinons aucune technique de cette sous-classe.

2.2.2 Les méthodes de partitionnement-optimisation

Au contraire des méthodes hiérarchiques, celles de partitionnement permettent la réallocation d'une entité déjà attribuée à un autre groupe. Elles visent à obtenir une classification optimisant un critère défini. Leur démarche est composée en général par trois étapes :

- Initialiser les groupes
- Allouer les entités aux groupes initialisés

- Si nécessaire réallouer une partie ou toutes les entités

Les différences entre les diverses méthodes de partitionnement-optimisation résident dans les étapes 1 et 3 et dans le choix du critère à optimiser. Le nombre de groupes à obtenir doit être fixé au départ de la procédure de calcul par l'utilisateur.

2.2.3 Les méthodes par recherche de densité de points

Ce concept introduit dans les années soixante, se base sur la représentation des entités sous forme de points dans un espace métrique (un espace euclidien par exemple). Si l'on considère cet espace, il doit y avoir des zones dans lesquelles la densité des points est très élevée et d'autres où elle est plutôt faible. Il est possible de retrouver les groupes auxquels appartiennent les diverses entités à partir de l'analyse de cette densité.

2.2.4 Les méthodes de "clumping"

La plupart des méthodes forment des groupes disjoints. Certaines recherches nécessitent parfois d'obtenir une classification où les groupes se recoupent. C'est la caractéristique des méthodes dites de *clumping*. Elles débutent par le calcul d'une matrice des similarités. Ensuite, elles recherchent une partition des entités en deux minimisant une fonction de cohésion entre les deux groupes. Nous ne traitons pas de telles méthodes.

2.2.5 Autres méthodes

Il existe encore d'autres méthodes de classification automatique difficiles à classer parmi les trois grands groupes présentés. Certaines sont plutôt des adaptations utilisées pour une analyse particulière d'une des précédentes.

2.2.6 Avenir des méthodes classiques

Malgré les nombreuses critiques les méthodes classiques sont implantées dans de nombreux logiciels statistiques. Elles font toujours l'objet de perfectionnement, ainsi que nous avons pu nous en rendre compte lors de la conférence de Aix-La-Chapelle en juin-juillet 1987.

Pourquoi un tel intérêt pour ces méthodes déjà anciennes ? Simplement parce qu'elles sont peu complexes, rapides et les résultats obtenus peuvent sembler satisfaisants.

Les recherches actuelles portent sur de nombreuses directions. L'une vise à inclure le concept de minimisation d'une fonction-objectif globale. L'autre recherche une amélioration de la formule de récurrence de Lance et Williams,

aussi dans le but d'obtenir une optimisation. Cette deuxième voie cherche de nouveaux indices de proximité plus performants que ceux actuellement connus. Parmi les autres domaines de recherche, nous trouvons la théorie des graphes, la classification de groupes mélangés (fuzzy clustering), la détermination du nombre de groupes, les méthodes de validation des grands jeux de données, etc.

Signalons encore une recherche originale, celle du professeur Späth. Il cherche à obtenir une anticlassification avec les groupes les plus hétérogènes possibles, pour pallier aux défauts des méthodes classiques (pas d'optimisation) et des méthodes mathématiques (trop complexes).

2.3 Exposé des méthodes retenues

Nous exposons dans le détail les méthodes classiques retenues parmi les deux familles les plus répandues :

2.3.1 La méthode du chaînage simple

Cette méthode hiérarchique peut être utilisée soit avec des similarités soit avec une distance. La matrice des distances est mise à jour après chaque fusion, de telle sorte que la distance entre deux groupes est égale à la distance entre leurs deux plus proches éléments. Il s'agit de la méthode la plus simple.

L'algorithme est le suivant :

Tant que le critère de fin n'est pas atteint :

1. Trouver i <> j tel que

$$d(J_i, J_j) = \min \{ d(X_p, X_q),\ X_p \in J_i,\ X_q \in J_j \quad (2.9)$$

J identifie une classe ou groupe. Cette classe peut être formée d'un seul élément.

2. Mémoriser l'indice de l'itération.

3. Agréger J_i et J_j, en formant un nouveau groupe

$$J_k = J_i \cup J_j$$

4. Définir la nouvelle partition

5. Calculer les nouvelles distances par la formule propre à la méthode :

$$d(J_k, J_i) = \min(J_k, J_i) \quad k <> i \quad (2.10)$$

6. Ajuster le nombre de groupes

Fin du tant que

Le critère de fin est atteint quand il ne reste qu'un groupe formé de toutes les entités, ou quand un nombre de groupes fixés par l'utilisateur sont déterminés.

Pour notre exemple, de la page 7, nous obtenons les résultats partiels suivants. Nous avons calculé la matrice des distances par la méthode de Manhattan.

Nous avons attribué le numéro d'entité suivant aux six employés supposés.

Dupont : 1

Müller : 2

Meyer : 3

Durand : 4

Scott : 5

Henry : 6

Au départ, nous considérons six groupes, formés chacun de une entité :

{1},{2},{3},{4},{5},{6}

Les distances entre ces six groupes sont :

	{1}	{2}	{3}	{4}	{5}	{6}
{1}	0.0	8.5	4.5	9.5	5.5	10.5
{2}		0.0	9.0	3.0	14.0	4.0
{3}			0.0	10.0	5.0	9.0
{4}				0.0	15.0	3.0
{5}					0.0	14.0
{6}						0.0

La classification débute en déterminant les deux entités les plus proches. L'examen de toutes les possibilités nous montre que {2} et {4},avec une distance de 3.0 remplissent cette condition. La paire {4} et {6} est également une alternative possible. Cependant, la règle est de sélectionner la première trouvée dans la matrice. Cette distance est mémorisée afin de fournir l'indice d'agrégation pour le dendogramme. Les entités sélectionnées sont fusionnées pour former le premier groupe, nous avons :

{1},{2,4},{3},{5},{6}

Nous recalculons les distances par la formule de Lance et William. il suffit de considérer les anciennes distances par paire. La distance d({1},{2,4}) sera le minimum sur d{1,2} et d{1,4} c'est-à-dire d{1,2}=8.5. En procédant de même pour tous les groupes, nous obtenons la nouvelle matrice ci-dessous :

{1}	{2,4}	{3}	{5}	{6}
0.0	8.5	4.5	5.5	10.5
	0.0	9.0	14.0	3.0
		0.0	5.0	9.0
			0.0	14.0
				0.0

Nous devons déterminer la prochaine paire d'objets à fusionner. La distance

la plus petite est celle entre {2,4} et {6}, avec une valeur de 3. Nous formons une nouvelle classe, et recalculons la matrice des distances. Nous considérons les distances de {2,4}, et {6} avec respectivement {1}, {3} et {5}. Nous obtenons ainsi :

```
{1}  {2,4,6}  {3}   {5}
0.0    8.5    4.5   5.5
       0.0    9.0  14.0
              0.0   5.0
                    0.0
```

Nous déterminons que {1} et {3} peuvent être fusionnés puisque leur distance est la plus petite, 4.5. Puis nous calculons la nouvelle matrice de distance :

```
{1,3}  {2,4,6}   {5}
 0.0     8.5     5.5
         0.0    14.0
                 0.0
```

L'examen des distances nous montre que {5} peut être fusionné avec {1,3} à une distance de 5.5.

```
{1,3,5}  {2,4,6}
  0.0      8.5
           0.0
```

Il reste deux groupes, que nous fusionnons à une distance, pour le dendogramme de 8.5. Les calculs sont représentés par le dendogramme suivant :

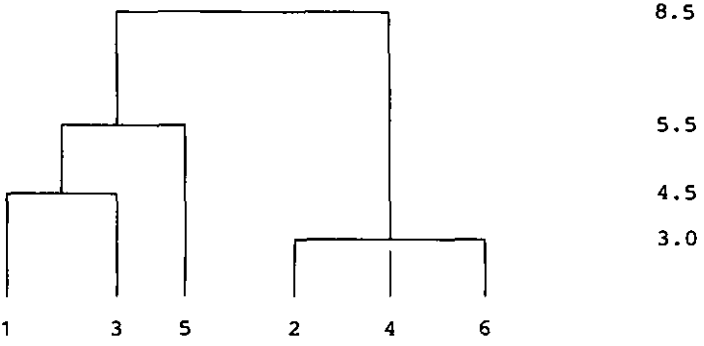


Figure 2.2 : dendogramme obtenu par chaînage simple

2.3.2 La méthode du chaînage complet

Pour cette seconde méthode hiérarchique, la distance entre deux groupes est la distance entre leurs deux objets les plus éloignés. Elle tend à former des groupes artificiels, mais très compacts.

Son algorithme est le même que pour la précédente, seul le critère de l'étape (5) est différent :

1. Trouver i <> j tel que

$$d(J_i,J_j) = \min \{d(X_p,X_q), \mid X_p \varepsilon J_i, X_q \varepsilon C_j\} \quad (2.9)$$

2. Mémoriser la distance d'agrégation.

3. Agréger les deux objets sélectionnés en 1.

4. Définir la nouvelle partition

5. Calculer la nouvelle matrice des distances, par la formule :

$$d(J_k,J_i) = \max (J_k,J_i), k <> i \quad (2.11)$$

6. Ajuster le nombre de groupe.

Illustrons une classification, pour notre exemple. Au départ, nous avons six groupes, formés chacun de une entité comme précédemment.

	{1}	{2}	{3}	{4}	{5}	{6}
{1}	0.0	8.5	4.5	9.5	5.5	10.5
{2}		0.0	9.0	3.0	14.0	4.0
{3}			0.0	10.0	5.0	9.0
{4}				0.0	15.0	3.0
{5}					0.0	14.0
{6}						0.0

Nous déterminons que {2} et {4}, à une distance de 3.0 forment la première classe. Nous obtenons cinq groupes :

{1},{2,4},{3},{5},{6}

Nous devons recalculer les dislances du nouveau groupe avec les anciens. La distance d{1,{2,4}} sera le maximum sur d{1,2} et d{1,4} c'est-à-dire d{1,2}=9.5.

{1}	{2,4}	{3}	{5}	{6}
0.0	9.5	4.5	5.5	10.5
	0.0	10.0	15.0	4.0
		0.0	5.0	9.0
			0.0	14.0
				0.0

Nous déterminons la prochaine paire d'objets à fusionner, parmi les cinq que nous avons obtenu après agrégation de {2,4}. La distance la plus petite est celle entre {2,4} et {6}, avec une valeur de 4. Nous obtenons la classification partielle et la nouvelle matrice des distances :

{1}	{2,4,6}	{3}	{5}
0.0	10.5	4.5	5.5
	0.0	10.0	15.0
		0.0	5.0
			0.0

les entités {1} et {3} sont fusionnées puisque leur distance est la plus petite, 4.5. Puis nous calculons la nouvelle matrice de distances :

```
{1,3}  {2,4,6}   {5}
 0.0    10.5     5.5
         0.0    15.0
                 0.0
```

L'examen de la matrice des distances nous montre que {5} peut être fusionné avec {1,3} à une distance de 5.5.

```
{1,3,5}  {2,4,6}
 0.0      10.5
           0.0
```

Les calculs peuvent être représentés par le dendogramme suivant :

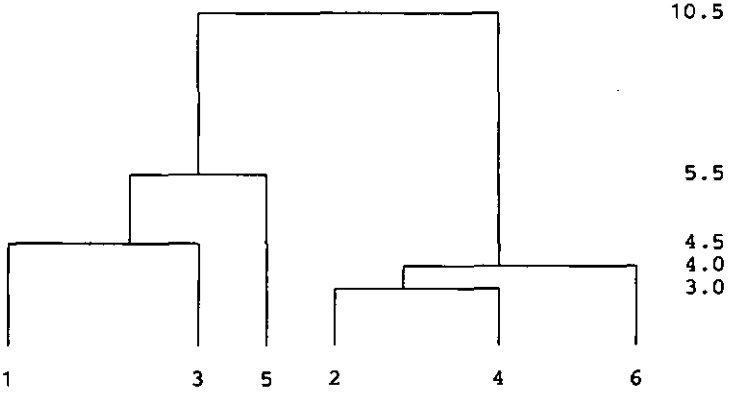


Figure 2.4 : dendogramme obtenu par chaînage complet

2.3.3 La méthode de la moyenne des distances dans les groupes

La démarche de l'algorithme est en tout point semblable à celle des deux précédents, seul le critère de l'étape 5 change :

$$d(J_k,J_i) = \frac{n_k n_i}{n_i+n_k} \sum d(X,Y) \qquad (2.12)$$

où $X \in J_k$, $Y \in J_l$, n_l et n_k le nombre d'éléments de chaque classe.

Au départ, nous considérons de nouveau que nous avons six groupes à une entité.

{1},{2},{3},{4},{5},{6}

Les distances initiales entre ces six groupes sont :

	{1}	{2}	{3}	{4}	{5}	{6}
{1}	0.0	8.5	4.5	9.5	5.5	10.5
{2}		0.0	9.0	3.0	14.0	4.0
{3}			0.0	10.0	5.0	9.0
{4}				0.0	15.0	3.0
{5}					0.0	14.0
{6}						0.0

L'examen des distances nous montre que à nouveau {2} et {4} forment la première paire à fusionner.

Nous obtenons à nouveau les cinq groupes de l'étape 1 :

{1},{2,4},{3},{5},{6}

Les distances sont calculées avec le critère (2.12). Nous obtenons les nouvelles distances :

{1}	{2,4}	{3}	{5}	{6}
0.0	12.0	4.5	5.5	10.5
	0.0	12.6	19.3	4.6
		0.0	5.0	9.0
			0.0	14.0
				0.0

Nous continuons les calculs à partir de cette matrice. Cette fois, nous formons la paire {1,3} à une distance de 4.5 et recalculons les distances :

{1,3}	{2,4}	{5}	{6}
0.0	41.0	7.0	13.0
	0.0	19.3	4.6
		0.0	9.0
			0.0

Nous déterminons que {6} et {2,4} sont fusionnés puisque leur distance est la plus petite, 4.6. Puis nous calculons la nouvelle matrice de distance :

{1,3}	{2,4,6}	{5}
0.0	72.6	7.0
	0.0	32.2
		0.0

Nous regroupons à cette étape {5} avec {1,3} à une distance de 7.0 et obtenons le dendogramme ci-dessous :

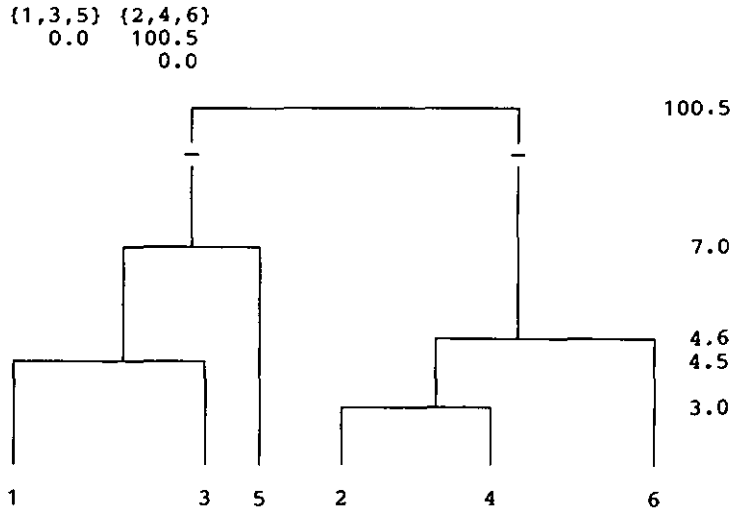


Figure 2.5 : dendogramme obtenu par la méthode de la moyenne des distances dans les groupes.

2.3.4 La méthode de la moyenne des distances entre les groupes (ou méthodes des centroïdes)

Pour cette méthode, le critère de l'étape 5 est :

$$d(J_i,J_K) = (\bar{X}_i-\bar{X}_K)'(\bar{X}_i-\bar{X}_k) \qquad (2.13)$$

$$\text{Avec } \bar{X}_i = 1/n_i \sum_{p \in J_i} X_p \text{ et } \bar{X}_k = 1/n_k \sum_{p \in J_k} X_q$$

La situation de départ est les six groupes à une entité :

{1},{2},{3},{4},{5},{6}

Les distances entre ces six groupes sont :

	{1}	{2}	{3}	{4}	{5}	{6}
{1}	0.0	8.5	4.5	9.5	5.5	10.5
{2}		0.0	9.0	3.0	14.0	4.0
{3}			0.0	10.0	5.0	9.0
{4}				0.0	15.0	3.0
{5}					0.0	14.0
{6}						0.0

Nous fusionnons à nouveau {2} et {4} et recalculons la matrice des distances, à l'aide du nouveau critère 2.13.

{1},{2,4},{3},{5},{6}

Et la matrice des distances devient :

{1}	{2,4}	{3}	{5}	{6}
0.0	9.0	4.5	5.5	10.5
	0.0	9.5	14.5	3.5
		0.0	5.0	9.0
			0.0	14.0
				0.0

Nous obtenons suite à la fusion de {2,4} et {6} à une distance de 3.5 les groupes et la matrice des distances suivants :

{1}	{2,4,6}	{3}	{5}
0.0	9.5	4.5	5.5
	0.0	9.3	14.3
		0.0	5.0
			0.0

Puis nous fusionnons {1} et {3} à une distance de 4.5, et nous obtenons les groupes et la matrice ci-dessous :

{1,3}	{2,4,6}	{5}
0.0	9.5	5.2
	0.0	14.3
		0.0

Finalement, {5} peut être fusionné avec {1,3} à une distance de 5.2.

{1,3,5}	{2,4,6}
0.0	12.3
	0.0

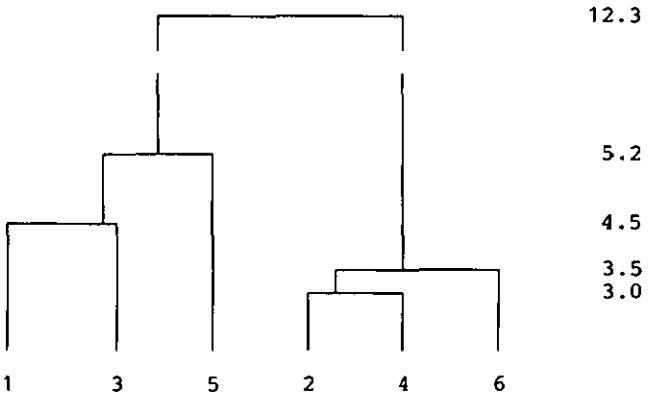


Figure 2.6 : dendogramme obtenu par la méthode des centroïdes

2.3.5 La méthode de Ward

Les groupes sont formés de façon à minimiser la somme des carrés dans les groupes. La distance entre deux groupes est l'accroissement de cette somme des carrés si les deux groupes sont fusionnés.Cette méthode utilise le critère ci-dessous pour le recalcul de la matrice des distances.

$$d(J_i, J_k) = \frac{n_i n_k}{n_i + n_k} (\bar{X}_i - \bar{X}_k)' (\bar{X}_i - \bar{X}_k) \qquad (2.14)$$

Avec $\bar{X}_i$ et $\bar{X}_k$ défini comme ci-dessus

La distance entre les entités doit se calculer à l'aide de la norme L_2. Il est possible d'utiliser efficacement la notion de poids dans cette méthode. Au départ, nous considérons toujours que nous avons six groupes,à une entité.

{1},{2},{3},{4},{5},{6}

Les distances entre ces six groupes sont :

	{1}	{2}	{3}	{4}	{5}	{6}
{1}	0.0	8.5	4.5	9.5	5.5	10.5
{2}		0.0	9.0	3.0	14.0	4.0
{3}			0.0	10.0	5.0	9.0
{4}				0.0	15.0	3.0
{5}					0.0	14.0
{6}						0.0

Pour débuter nous fusionnons à nouveau {2} et {4} à une distance de 3.0. Nous obtenons cinq groupes et calculons la nouvelle matrice des distances :

{1}	{2,4}	{3}	{5}	{6}
0.0	10.5	4.5	5.5	10.5
	0.0	10.6	17.3	3.6
		0.0	5.0	9.0
			0.0	14.0
				0.0

Nous obtenons en fusionnant {2,4} et {6} :

{1}	{2,4,6}	{3}	{5}
0.0	12.0	4.5	5.5
	0.0	12.5	19.3
		0.0	5.0
			0.0

Nous déterminons que {1} et {3} peuvent être fusionnés puisque leur distance est la plus petite, 4.5. Puis nous calculons la nouvelle matrice de distance :

{1,3}	{2,4,6}	{5}
0.0	18.8	6.5
	0	19.3
		0.0

Enfin, {5} est fusionné avec {1,3} à une distance de 6.5.

{1,3,5}	{2,4,6}
0.0	22.5
	0.0

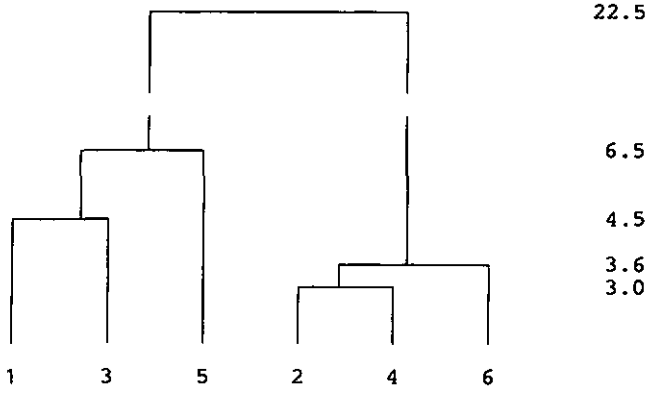


Figure 2.7 : dendogramme obtenu par la méthode de Ward

2.3.6 La méthode du K-mean

C'est une des méthodes de classificstion par partitionnement, qui opèrent en trois étapes : initialisation des groupes, première allocation des entités aux groupes initialisés, si nécessaire réallocation, avec comme objectif la

minimisation d'un critère donné. Le nombre de groupes à obtenir est à fixer comme donnée de départ.

La méthode du K-MEAN n'utilise pas la première étape décrite ci-dessus. Elle débute par la première allocation des entités, puis effectue la réallocation. Il est nécessaire de fournir comme donnée en entrée, une estimation des m groupes à obtenir, contenant chacun un rmitement. L'algorithme du K-MEAN doit être combiné avec une méthode permettant de trouver cette estimation. Ensuite des conditions de transfert déterminent comment les entités sont à reclasser pour améliorer le regroupement. La première de ces conditions examine si un transferi est possible. La seconde effectue le transfert nécessaire. L'algorithme s'arrête si aucun transfert n'a eu lieu, ou si un nombre maximum d'itérations a été effectué.

L'algorithme du K-mean est basé sur les notions suivantes :

- L'échelle des observations est telle que l'utilisation de la distance euclidienne est appropriée pour calculer les dissimilarités
- La partition P(N,M) est composée des m groupes $(J_1, J_2, \ldots, J_m)$
- Chacune des n entités n'appartient qu'à un seul groupe
- B(J,L) dénote le leader théorique L dans le groupe J, c'est-à-dire l'entité moyenne du groupe. Le nombre d'entités classées dans J est donné par n_J et A(I,L) l'observation I de la variable L, $D(I,L_J)$ est la distance euclidienne entre l'entité I et le leader du groupe auquel elle appartient.

La distance entre l'entité I et le groupe L est donnée par :

$$D(I,L_J) = \sum_{J=1}^{k} [(A(I,L)-B(J,L)^2]^{\frac{1}{2}} \qquad (2.15)$$

L'erreur dans la partition est :

$$e[P(N,M)] = \sum_{i=1}^{n} D(I,L_i)^2 \qquad (2.16)$$

Etape 1 : Soit une partition initiale donnée, déterminer les lesders et l'erreur initiale (2.16) dans la partition.

Etape 2 : Tant qu'une entité est transférée faire :

Pour i=1,...,n faire

Déterminer l'accroissement de l'erreur en transférant l'entité I du groupe auquel elle appartient (J_j) à chacun des autres groupes par la formule :

$$T = \frac{n_{Jj} D(I,L_J)^2}{n_{Jj} + 1} - \frac{n_{Ji} D(I,L_{Ji})^2}{n_{Ji} - 1} \qquad (2.17)$$

Opérer le transfert, si l'accroissement pour un groupe j est négatif, déterminer les nouveaux lesders et la valeur de l'erreur pour la partition

fin du pour

fin du tant que

Nous illustrons le développement de cette méthode par notre exemple de référence.

Nous supposons que la partition initiale est donnée par les deux groupes suivants :

{1,2,3} et {4,5,6}

Pour chaque groupe, nous calculons la valeur moyenne de chacune des caractéristiques.

{1,2,3} 3.5 3.66 3.66 4.66 3.66

{4,5,6} 4 4.66 4.33 3.66 4

L'erreur totale dans cette partition initiale est donnée par :

$$e[p(6,2)] = (2.5-3.5)^2 + (3-3.66)^2 + \ldots + (4-3.66)^2 + (6-4)^2 = 68.47$$

Soit la somme des écarts de chacune des caractéristiques de l'échantillon à la valeur moyenne du groupe dans lequel l'entité est classée. Par le calcul de la distance euclidienne de l'entité 1 à la moyenne de chacun des groupes, nous obtenons les deux distances suivantes :

$$d(1,L_{J1}) = [(2.5-3.5)^2+(3-3.66)^2+(4-3.66)^2+(5-4.66)^2 + (2-3.66)^2] = 2.10$$
$$d(1,L_{J2})) = 3.30$$

Nous calculons ensuite le coût de transfert de {1} du groupe J_1 au groupe J_2.

$$T(1,2) = \frac{3*10.91}{3+1} - \frac{3*4.42}{3-1} = 1.55$$

T > 0, nous ne pouvons pas transférer {1} dans J_2.

Nous effectuons les mêmes calculs pour {2}, nous obtenons que :

$$d(2,L_{J1}) = 3.17$$
$$d(2,L_{J2}) = 5.065$$
$$T(1,2) = 3*25.66/4 - 3*10.47/2 = 4.13$$

Nous ne pouvons pas transférer {2} dans le groupe J_2. Nous renonçons à présenter les calculs pour les entités {3},{4} et {6}.

Pour l'entité {5}, nous obtenons :

$$d(5,L_{J1}) = 4.15$$
$$d(5,L_{J2}) = 4.50$$
$$T(2,1) = 3*17.27/4 - 3*20.26/2 = -17.46$$

{5} doit être transféré de J_2 vers J_1

La nouvelle erreur dans la partition est :

$$e(P) = 68.47 - 17.46 = 51.01$$

Nous calculons les nouvelles moyennes des carsctéristiques suite à ce transfert :

{1,2,3,5}	3.12	3.45	3.45	3.99	2.99
{4,6}	5	5.5	5	4.5	5.5

Nous trouvons en procédant selon le même schéma que {2} doit être transféré de J_1 vers J_2. Nous obtenons ainsi les deux groupes finaux :

{1,3,5} et {2,4,6}

2.3.7 La méthode du partitionnement autour des médioïdes

Cette méthode est basée sur la recherche d'objets représentatifs des m groupes à obtenir. Ensuite, les entités restantes sont assignées su groupe dont le leader est le plus proche.

Les objets représentatifs sont déterminés de façon à être les médioïdes des groupes. Ils représentent la structure des données à snalyser. Un médioïde est l'objet pour lequel la dissimilarité moyenne par rspport aux autres entités du même groupe est la plus petite. Cette méthode est due aux travaux du professeur Rousseeuw.

Elle permet d'utiliser, soit la distance euclidienne, soit la distance de Manhattan, pour calculer les dissimilarités.

L'objectif recherché est de minimiser la somme des dissimilarités dans les groupes.

L'algorithme sera illustré directement par notre exemple. Il se compose de deux phases distinctes, l'une est sppelée "BUILD", la seconde "SWAP". Ces deux étapes sont itératives. La phase "BUILD"' permet de trouver les objets représentatifs ou médioïdes initiaux, puis de former la classification initiale. Par "SWAP", la solution est réexaminée, et si nécessaire de nouveaux médioïdes sont déterminés, une nouvelle classificstion est dans ce cas formée. SI aucun nouvel objet représentatif n'est déterminé, l'algorithme s'arrête.

Nous utilisons à nouveau la matrice des distances calculées avec L_1.

L'étape "BUILD" est caractérisée par deux phases : le premier objet représentatif est fixé, comme étant celui dont la distance à tous les autres est la plus petite.

$$\text{Trouver } i \mid \sum d_{ij} = \min_i \sum d_{ij} \quad i,j = 1,\ldots,n \qquad (2.18)$$

Pour notre exemple, nous avons pour chaque entité, les sommes des distances suivantes :

$d(1) = 38.5,\ d(2) = 38.5,\ d(3) = 37.5$
$d(4) = 40.5,\ d(5) = 53.5,\ d(6) = 40.5$

Le premier médioïde est (3).

Ensuite pour chacun des objets non sélectionnés, la deuxième phase itérative de "BUILD" calcule la somme des différences entre les distances à tous les médioïdes actuels et les objets à classer, l'entité pour laquelle le calcul est effectué faisant office de leader potentiel. L'objet pour lequel cette somme est maximale devient le prochain médioïde.

Nous ne montrons dans le détail que les calculs nécessaires pour l'entité (2). Pour les autres, seul le résultat final est donné.

Nous calculons tout d'abord la différence liée à l'entité (1), soit C_{21}.

$C_{21} = \max\ (\ [d_{13}-d_{12}],\ 0) = \max\ ([4.5-8.5],0) = 0$
$C_{24} = 8.5$
$C_{25} = 0$
$C_{26} = 8.5$

$C_2 = \sum_i C_{2i} = 17 \qquad i = 1,\ldots,n\text{-médioïdes}$

Pour les autres entités, nous obtenons les C_j suivants :

$C_1 = 1,\ C_4 = 6.5,\ C_6 = 12,\ C_5 = 3$

Ainsi (2) est le deuxième et dernier médioïde cherché, puisque nous voulons obtenir 2 groupes.

La phase de "SWAP" permet, par une série de permutations, de déterminer le meilleur ensemble de médioïdes. Pour opérer ces permutations, il convient de considérer les objets par paire, un des médioïdes et une entité non-sélectionnée. Si la permutation permet de réduire la somme des dissimilarités, alors elle devient effective.

Ainsi dans notre cas, avec (2) comme médioïde la somme des dissimilarités dans le deuxième groupe est 3 + 4 = 7. Si (4) est retenu comme objet représentatif, cette somme devient 3 + 3 = 6. Il est avantageux de procéder à cette substitution. A chaque substitution effectuée, la classification est à nouveau déterminée, chaque objet étant affecté au groupe du médioïde dont il est le plus proche comme pour "BUILD".

La méthode PAM se propose de minimiser la somme des dissimilarités aux objets médians, comme certaines méthodes basées sur la programmation entière. Cette technique utilise cependant une heuristique, qui ne minimise le critère que localement.

2.3.8 La méthode de partitionnement par les médianes

Cette méthode cherche également à minimiser par une heuristique la somme des distances aux objets médians, comme celle de 2.3.7.. Elle utilise toutefois une heuristique différente, assez proche de celle de K-mean. Tout d'abord il convient de déterminer par une technique quelconque une classification initiale.

Ensuite, pour ce regroupement, nous déterminons les objets médians, soit ceux pour lesquels la somme des distances aux autres contenus dans le même groupe est minimale. Puis, nous cherchons à réallouer toute entité à la médiane la plus proche. Les permutations concernent les entités autres que les leaders. Après chaque permutation, de nouvelles médianes sont calculées.

Pour notre exemple, nous obtenons comme pour K-mean, la classification initiale suivante :

{1,2,3} et {4,5,6}

Pour ces deux groupes les médianes initiales sont calculées :

$$d_1 = 13,\ d_2 = 17.5,\ d_3 = 13.5$$

La médiane du groupe 1 est {1}.

$$d_4 = 18.0,\ d_5 = 29.0,\ d_6 = 17.0$$

{6} est la médiane du deuxième groupe.

Chaque entité non-médiane est affectée au groupe dont la médiane est la plus proche. Ainsi, nous déterminons que {2} est plus proche de {6} que de {1}. Nous procédons à l'échange et obtenons les groupes : {1,3} et {2,4,5,6}.

Nous déterminons les médianes pour cette classification : {1} et {2}. Nous examinons à nouveau dans quel groupe doivent être classées les autres entités, {5} étant plus proche de {1} que de {2}, nous procédons à l'échange. Nous avons les groupes {1,3,5} et {2,4,6}. Les médianes sont cette fois {5} et {4}. Plus sucun transfert n'est nécessaire. Les calculs s'arrêtent. Cette heuristique est moins efficace que celle de 2.3.7, comme nous le verrons plus loin, car elle dépend beaucoup de la classification initiale, qu'elle ne peut guère modifier.

Chapitre 3

Problèmes posés par les méthodes classiques

C3.1 Complexité des algorithmes

Pour déterminer et comparer la complexité des divers algorithmes examinés, nous n'avons jamais tenu compte des opérations de lecture des données, de calcul des matrices de dissimilarités et d'impression des résultats, puisque toutes ces opérations sont communes à toutes les méthodes. Les composants de la complexité sont décrits dans le chapitre 6. Le calcul de la distance a une complexité de $O(n^2p)$, ce qui n'est pas négligeable. Nous avons voulu ne tenir compte que de la part de la complexité due aux étapes propres des méthodes. Nous parlerons de complexité moyenne lorsque nous donnerons une valeur probable. Nous avons constaté que certaines méthodes volent en pratique leur complexité varier entre un minimum et un maximum que nous avons pu déterminer.

Les techniques de classification hiérarchique ne diffèrent que par le critère utilisé lors du calcul de la nouvelle matrice des distances. Cette seule différence ne modifie pas leur complexité. Aussi pour l'illustrer, nous examinons dans le détail la technique du chaînage simple. Pour les autres méthodes, nous ne montrons que les différences dues à la variante de la formule de Lance et Williams.

3.1.1 La complexité de la méthode du chaînage simple

Pour déterminer la complexité de cette méthode, nous relevons toutes les opérations nécessaires pour classer les six employés de notre exemple. Nous cherchons également à établir la nature de l'opération : s'agit-t-il d'une affectation de résultat, d'un calcul simple, complexe, ou d'une comparaison ? Une telle énumération est nécessaire pour évaluer la performance informatique de l'algorithme. Nous donnons dans le chapitre six une table de pondération de chaque opération en son temps relatif d'exécution par un ordinateur.

A partir de la classification initiale formée par les six entités, nous devons pour déterminer la première paire à fusionner :

- effectuer 6*(6-1) comparaisons de distances, à partir d'un tableau;
- enregistrer la distance à laquelle la fusion a lieu, soit une opération d'affectation simple;
- former et mémoriser la classe résultant de la fusion : deux affectations dans un tableau;

- définir la nouvelle partition, soit mémoriser les objets non touchés par la fusion dans un tablesu : 5 affectations.

- Calculer la nouvelle matrice des distances, il faut effectuer dans notre cas 2*4 accès à la matrice des distances et comparer par paire, puis mémoriser les nouvelles distances, soit 4 affectations dans un tsblesu. Pour que cette matrice des distances aoit complète, noua devons encore récupérer les (6-2)(6-2)/2 autres dissimilarités.

- Ajuster le nombre de groupes, aoit une opération de calcul aimple, soustraction ou addition.

Pour la première paire fusionnée, nous effectuons 6(6-1) + 2 (6-2) +(6-2)^2/2 opérations.

Pour fusionner la seconde paire, les opérations sont :

- (6-1)*(6-2) comparaisons pour trouver la paire;

- 2*(6-3) comparaisons pour déterminer les nouvelles distances pour la paire fusionnée;

- (6-3)^2/2 affectations des anciennes distances, non touchées par la fusion;

Nous effectuons pour trouver cette deuxième paire au total 3/2*6^2 -2*6 -3/2 opérations. A chaque itération, nous avons au minimum une complexité de 3/2 * 6^2- 6 k, où k représente le numéro de l'itération.

Pour classer les aix employés nous avons besoin de aix itérations. Dans le cas général, nous en aurions n. La complexité moyenne de la méthode du chaînage simple est de n(3/2n^2-n), soit de l'ordre de n^3.

3.1.2 La complexité des autres méthodes hiérarchiques

La complexité des autres méthodes hiérarchiques est identique. Elles utilisent une formule de calcul pour ces nouvelles distances, au lieu d'une série de comparaison. Ces formules ne sont formées que d'opérations simples : addition, multiplication. Elles ont besoin d'autant d'accès aux tablesux de données, mais en changent la nature. Noua aurons toujours autant d'opérations.

Nous pouvons conclure que les méthodes de classification hiérarchique ont une complexité moyenne de l'ordre de n^3.

3.1.3 La complexité des méthodes de partitionnement

Selon Hartigan, elle est de l'ordre de n^2 pour K-mean. Pour Rousseeuw, la méthode PAM demande au moins n(n-1)/2 accès à la matrice des distances. Sa complexité serait d'au moins n^2/2.

Nous examinons tout d'abord la méthode K-mean. Pour obtenir la classification initiale, il nous faut n opérations d'affectation dans des tableaux, plus un calcul pour déterminer le nombre d'objets pour chacun des deux

groupes. Cette phase nécessite au moins n+1 opérations.

Le calcul des moyennes demande k*n accès à la matrice des données originales, k est ici le nombre de caractéristiques par entité observée, et k*n additions. Pour calculer la somme des erreurs, nous avons besoins de k*n accès à la matrice des données et de 2k*n calculs, soustraction et calcul du carré, puis nous avons k*n additions et un calcul de racine.

Pour chaque entité, nous avons 2mk calculs et accès à des données pour déterminer sa distance aux moyennes de chacun des m groupes. Puis le calcul du coût de transfert nécessite 2m opérations simples. Si nécessaire, nous devons effectuer le transfert d'une entité vera un groupe, soit une opération.

Si un transfert a eu lieu, nous devons recommencer tous ces calculs, et ainsi de suite jusqu'à ce que la procédure s'arrête.

La complexité minimale de K-mean est de n+1+5kn+2mk+2m, soit n(5k+1)+2m(k+1)+1. Dans le pire des cas, nous pouvons avoir à transférer les n entités. La complexité maximale est n(n(5k+1)+2m(k+1)+1), soit supérieure à n^2. Raisonnablement, nous pouvons penser qu'elle est de l'ordre de n^2 en moyenne, comme l'a énoncé Hartigan.

Pour la méthode PAM, par la procédure de "BUILD", nous avons n(n-1) additions pour trouver la somme des distances de toutes les entités. Ensuite, il nous faut n(n-1) comparaisons pour déterminer le premier médioïde. Ainsi la complexité est de 2n(n-1) pour cette première partie.

La seconde phase de "BUILD", nécessite m-1 itérations, à $(n-k)^2$ (k =nombre de médioïdes courants) opérations pour déterminer le reste des objets représentatifs. Cette deuxième phase a une complexité de $(m-1)(n-k)^2$.

Au minimum, nous devons effectuer une itération dans "SWAP". Ce passage coûte au moins m(n-m) opérations. Nous pouvons penser raisonnablement que le nombre d'itérations est en tout cas inférieur au nombre d'entités. Une valeur moyenne de n-m est sans doute correcte. La complexité de PAM est au moins de l'ordre de $(m+2)n^2+m^3$.

La méthode de partitionnement par les médianes demande n+1 opérations pour former la classification initiale. Ensuite, il faut n(n-1) calculs pour déterminer les distances pour chacune des entités, puis n(n-1) comparaisons pour déterminer les médianes. Pour les n-m objets non-médians, il faut effectuer m calculs pour prouver la médiane la plus proche. A ce stade-là, nous avons une complexité minimale de la méthode de 2n(n-1) + m(n-m), soit $2n^2 - m^2 - 2n + nm$.

Au pire, il faudra réallouer les n entités soit, effectuer n itérations qui coûtent chacune m(n-m) calculs, la complexité maximale est :

- $2n(n-1)+nm(n-m)$, soit $(m+2)n^2-m^2-2n$. Nous pouvons nous attendre à ce que la complexité moyenne soit de l'ordre de $mn^2/2$.

3.2 Propagation d'erreurs par les distances (Tied distances)

Une erreur type des algorithmes hiérarchiques est de mal sélectionner les paires d'objets. Nous verrons un tel cas, au chapitre sept. Dans notre exemple, nous avons toujours choisi de fusionner en premier la paire (2,4), avec une distance de 3. Mais la paire (4,fi) a également la même distance et est tout autant correcte par rapport au critère de sélection. En cas d'égalité de distance entre deux paires, en général, nous choisissons arbitrairement la première trouvée. Ce choix peut s'avérer être faux.

Cette erreur initiale sera propagée à toutes les étspes au travers de la formule de récurrence (Lance et Williams). Concrètement cela signifie que tous les calculs des distances sont "contaminés" par cette erreur. Comme les distances sont utilisées, à chaque sélection de paire d'objets à fusionner, toutes les agrégations aont touchées. La classificstion peut être complétement incorrecte dès le début.

Les méthodes de partitionnement souffrent plus ou moins d'un problème semblable. Elles ne considérent que des critéres d'échange locaux. Si l'étst actuel de la solution est très éloigné de l'optimum, il est à craindre que les heuristiques utilisées ne convergent pas, bien que ces techniques visent à minimiser une fonction- objectif globale.

Nous avons relevé le falt que ces méthodes voient leur critère d'arrêt, pas de transfert effectué par la dernière itération, renforcé par une condition draconienne : un nombre d'itérstions maximum. Ainsl, une entité peut très bien ne pas changer de groupe, simplement parce que ce critère a été atteint. Sans cette restriction, le programme pourrait boucler sur un échange identique.

Leur plus grande source d'erreur vient de la détermination des objets "centraux". Il n'est pas garanti, à cause des critéres d'arrêt, que les échanges suffisent à corriger une erreur à cette étspe. Seule la méthode du Professeur Rousseuw nous semble donner de bons résultats quant à cette déterminstion.

3.3 Pourquoi un optimum n'est pas atteint par ces méthodes ?

Noua allons mettre en évidence les raisons pour lesquelles les méthodes classiques ne peuvent pas atleindre un optimum global. Hartigan (1975, p. 11) précise : "All clustering algorithms are procedures for searching through the set of all possible clusterings to find one that fits the dats well. Frequently, there is a numerical measure of fit which the algorithm attempts to optimize, but many useful algorithms do not explicitly optimize a criterion...".

Nous pouvons aouligner les deux éléments essentiels de cette citation : l'optimisation est vue sous l'angle d'un critére numérique. Et de nombreux algorithmes ne cherchent même pas à optimiser un tel critère.

Nous préciserons que l'optimisation d'un critère numérique revient à rechercher un maximum ou un minimum à une fonction dite objectif. Nous aurons l'optimum global, si le point de la fonction trouvé est le minimum des

minimums existants, et inversement en cas de maximisation. Nous admettons le postulat qu'il existe une classification qui représente un de ces points minimums.

Nous pouvons relever plusieurs raisons pour lesquelles les méthodes de classification, tant classiques que basées sur les techniques de programmation mathématique, n'atteignent pas un optimum :

- Aucun critère n'est optimisé;
- Un critère local est utilisé;
- Un critère global est utilisé, mais l'heuristique de calcul le réduit en critère local;
- Un critère global est utilisé, mais la méthode envisagée ne converge pas quand elle est appliquée à la classification automatique;

Les méthodes hiérarchiques ne permettent pas d'atteindre un optimum global pour les raisons suivantes :

- Lors de la recherche des paires à fusionner, elles recherchent un minimum sur les distances actuellement disponibles. Il s'agit au plus d'un critère local.
- La nouvelle matrice des distances est calculée, à chaque itération à partir de la fusion décrite ci-dessus, par un critère local. Ce critère ne peut que renforcer l'erreur, ou plutôt l'éloignement de l'optimum théorique. Les autres méthodes utilisent un critère de calcul, qui tient également compte et propage l'erreur commise par la fusion.
- Si les groupes ne sont pas distincts, les résultats obtenus tiennent compte des points intermédiaires à l'intérieur de l'un ou l'autre des groupes et propagent des erreurs supplémentaires.

Les résultats sont difficiles à interpréter, l'utilisateur doit examiner une *hiérarchie de classifications* et déterminer *arbitrairement* à quel *niveau* (nombre de groupes) il s'intéresse. Il n'a aucune indication fournie par la méthode pour l'aider dans son choix. Même un test statistique tel que le test F ne lui apporterait aucune aide, car il est bien probable que la classification proposée à tout niveau ne donne pas de groupes homogènes. L'interprétation du dendogramme est avant tout une affaire visuelle et subjective. Elle ne peut pas conduire à une vision optimale, sous le sens où nous l'entendons, des résultats. L'interprétation d'un dendogramme avec plusieurs centaines de données est un exercice très difficile. Pour notre exemple avec six données, c'est chose aisée. Mais imaginons un tel graphe couvrant plusieurs pages !!!

Pour pallier à ce problème, les méthodes hiérarchiques sont complétées par une technique qui permet de déterminer la classification obtenue pour un nombre de groupes fixé. Ce nombre ne peut être déterminé que subjectivement. Une distorsion supplémentaire de la réalité est introduite. Mais en revanche, les résultats sont "quantifiables", c'est-à-dire que nous obtenons la liste des groupes avec les entités qu'ils contiennent.

Les méthodes de partitionnement tendent à minimiser un critère global. Cependant, comme elles travaillent avec une heuristique de calcul, elles ne trouvent qu'un optimum local. Il y a deux raisons principales. Nous allons les illustrer par rapport à la méthode PAM. Ces remarques peuvent être facilement

généralisées aux autres techniques de cette famille.

Nous avons vu que "PAM" se déroule en deux phases. Dans ces deux étapes de calcul, nous trouvons deux heuristiques. Seule la détermination du premier médioïde dans "BUILD" utilise un critère global. Ce premier objet est défini comme étant celui pour lequel la somme des distances est minimale sur l'ensemble. Par la suite, cette procédure cherche les m-1 autres objets représentatifs, en utilisant à chaque fois un critère local. Seul l'objet testé, en ce moment des calculs, est considéré. Les tests se font de plus sur des localisations précises, les médioïdes déjà connus.

Par la procédure de "SWAP", nous ne considérons que des paires d'objets, un médioïde et un autre, pour trouver quelles permutations sont favorables, nous localisons en deux points sur les n, le test d'optimalité.

Nous pouvons résumer pour toutes les méthodes de partitionnement. Elles nécessitent un point de départ pour la classification. Il s'agit souvent d'un élément représentatif pour chacun des groupes. Ce choix peut être faux et même si par la suite la méthode cherche à réallouer les entités en fonction du critère à optimiser, il n'y a aucune garantie d'atteindre l'optimum global.

Les méthodes de partitionnement ont au moins l'avantage d'être de pures techniques de calcul, et ne donnent qu'un résultat final : les m groupes demandés, avec les entités contenues. Elles sont souvent complétées par des résultats statistiques complémentaires, tels que variance à l'intérieur des groupes, dissimilarité moyenne ou totale pour l'ensemble des groupes ou/et chacun des groupes... Ainsi l'analyste dispose d'éléments de décision, qui doivent l'aider, par exemple à fixer le nombre de groupes à retenir.

Dans toutes les méthodes étudiées, le nombre de groupes n'est jamais déterminé de manière optimale. Il est très souvent laissé à l'appréciation de l'utilisateur. Diverses méthodes permettant de le trouver ont été proposées. Aucune ne s'est révélée satisfaisante.

Les méthodes hiérarchiques ont l'avantage d'être très rapides et faciles à comprendre conceptuellement pour des non-mathématiciens. Leur concept de regroupement par paire semble très "naturel" et explique certainement leur large diffusion.

Chapitre 4

Approches par la programmation mathématique

Nous allons aborder quatre approches du pmblème de classifieation automatique par la programmation mathématique. Nous exposons le principe de ces méthodes, et en raison de leur complexité, nous ne présentons qu'une partie de la résolution de notre exemple. Nous déterminons immédiatement la complexité et la performance de chaque méthode envisagée.

Nous terminons ce chapitre en examinant de plus près la définition des normes L_p. Nous essayons de trouver un algorithme qui détermine quand et avec quelle méthode, pour quelles données, utiliser une norme plutôt qu'une autre.

La programmation mathématique vise à optimiser une fonction-objectif, soit un critère numérique, sous un ensemble de contraintes décrivant le problème. L'optimisation consiste à trouver le maximum, respectivement le minimum global de la fonction considérée. Nous devons tenir compte de la réalisabilité de la solution par rapport au domaine "réel", où le problème a un sens. Pour un problème économique par exemple, nous ne pouvons que déterminer des solutions strictement positives ou nulles.

Le problème de classification automatique peut être vu généralement comme étant :

- L'optimisation d'un critère d'homogénéité des groupes;
- Sous les contraintes qu'une entité ne doit appartenir qu'à un et un seul groupe et que toutes doivent être classées;

Pour représenter un problème de classification sous une forme résoluble par une technique de programmation mathématique, il est nécessaire que le nombre de groupes soit fixé.

4.1 Classification et programmation entière

Plusieurs auteurs ont démontré que le problème de classification automatique pouvait être modélisé comme un problème de programmation entière. Cette technique d'optimisation consiste à n'utiliser comme variables que des 0 et des 1. Il existe de nombreuses variantes du problème de classification automatique sous cette forme. Nous nous proposons d'en aborder deux. Elles diffèrent par la fonction-objectif à minimiser : l'une cherche à optimiser la somme totale des distances aux médianes, l'autre la somme totale des carrés dans les groupes.

Pour ces deux approches, nous utiliserons les notations suivantes :

N = {1,2,...,n} : l'ensemble des n éléments à classer

n_j : nombre d'objets classés dans le jème groupe

m : nombre de groupes à obtenir

X : matrice des données entières où :

$x_{ij} = \begin{cases} 1 \text{ si l'élément } i \in \text{ au groupe } j \\ 0 \text{ sinon} \end{cases}$

d_{ij} matrice des distances

4.1.1 Détermination de la médiane d'un groupe minimisant la somme totale des distances à la médiane de l'échantillon

Nous allons examiner deux variantes de cette méthode : celle en programmation entière pure proposée par Rao (1971) et celle réduite par un lagrangien reprise par Arthanari et Dodge (1981).

Nous cherchons à classer les n entités dans m groupes en introduisant la contrainte qu'un élément donné i ne peut appartenir qu'à un seul groupe j. Nous pouvons considérer que m peut être égal à n, tout en sachant que (n-m) groupes peuvent rester vide. Nous identifions les m groupes non-vides par leur médiane. La matrice des coûts C est donnée par les dissimilarités entre les entités.

Nous avons à résoudre le problème suivant :

$$\text{minimiser} \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} d_{ij} \qquad (4.1.1)$$

et x_{ij} comme défini ci-dessus

sous contraintes :

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i=1,\ldots,n \qquad (4.1.2)$$

qui signifie qu'une entité ne peut appartenir qu'à un seul groupe.

$$\sum_{j=1}^{n} x_{jj} = m \qquad (4.1.3)$$

Cette contrainte exprime le fait que seuls m groupes sont non-vides.

$$x_{jj} \geq x_{ij} \qquad i,j=1,\ldots,n \qquad (4.1.4)$$

assure que le groupe j n'est formé que quand l'entité correspondante est une médiane, donc $x_{jj}=1$, $n_j > 0$.

Remarque : Le problème sous sa forme actuelle est très complexe. Il contient notamment n^2 contraintes du type (4.1.4). Nous voyons ainsi que la complexité croît exponentiellement par rapport au nombre d'entités à classer.

4.1.1.1 Résolution par la programmation entière pure

Pour être sûrs que la résolution du problème sous forme entière pure nous conduise bien à l'optimum, il convient de considérer la condition requise émise par Rao (1971) :

- "Dans une solution optimale, chaque groupe consiste en au moins un élément, qui par convention est appelé le leader du groupe; de telle sorte que la distance entre ce leader et tout élément qui n'appartient pas au groupe n'est pas moindre que celle entre cette entité donnée et tout autre élément de son groupe".

Cette condition de chaînage entre le leader (médiane) et les entités classées dans son groupe n'est pas une condition nécessaire, mais elle est suffisante.

Reprenons la formulation de notre problème sous forme matricielle :

```
Min CX
sc     AX = b
       Xi = 0 ou 1 pour tout i, sauf le dernier
où
       A est une matrice (n+1)x(n(n-1)+2)
       X est un vecteur-colonne n(n-1)+2
       b est un vecteur colonne n+1 donné par
            (1,1,...,m)
       C est un vecteur-ligne n(n-1)+2
```

Chaque élément du vecteur X représente une classification potentielle. Ainsi le $i^{ème}$ élément, vaut 1 ou 0, selon que ce groupe particulier est utilisé ou non dans une solution optimale. C_i représente l'évaluation de la fonction-objectif pour chaque groupe potentiel.

Chaque colonne de A représente les coefficients pour un groupe particulier et chacune de ses lignes, à l'exception de la dernière, correspond à une entité. Une colonne de A à un 1 dans une ligne, si l'entité correspondante appartient à un groupe donné.

Soit les entités $i_1, i_2, \ldots, i_n$ et j telles que :

$$d_{jj} = 0 < d_{ji1} < d_{ji2} \cdots < d_{jin}$$

Nous pouvons obtenir les classifications suivantes, par la condition de chaînage avec l'entité j comme médiane : $(j,i_1),(j,i_1,i_2)$. La classification (j,i_2) est exclue puisque $d_{ji2} > d_{ji1}$. Pour une médiane donnée, nous avons n-1 possibilités de former un groupe. Comme les n entités sont des leaders potentiels, nous avons n(n-1)+1 groupes possibles, y compris celui composé de

toutes les entités. La dernière colonne de A doit ne contenir que des zéros de façon à limiter ces possibilités à m. De plus, il convient de ne considérer les groupes identiques qu'une seule fois, avec le meilleur lesder.

Le problème ainsi défini se résout par l'algorithme du "set partioning" par énumération décrit par Garfinkel et Nemhauser (1972).

Nous utilisons les notations complémentaires suivantes :

Soit S une solution partielle telle que

$$S+ = \{ i \mid x_i = 1, i \in S \} \text{ et}$$
$$z(S) = \sum_{i \in S} + Ci$$

Soit Q(S) l'ensemble des contraintes satisfaites par S.

Algorithme

1. Générer le problème de programmation entière à partir des n entités et de la matrice des distances.

2. Chercher la solution optimale par l'algorithme de partitionnement en six étapes :

   Etape 1 : Soit S = 0 et zbar = ∞

   Etape 2 : (choix de la prochaine liste)

   Soit $i^* = \min \{i \mid i \in Q(S)\}$

   Initialiser un indicateur en haut de la liste i^* (élément avec le coût inférieur)

   Faire l'étape 3.

   Etape 3 : (test pour une variable supplémentaire)

   Commencer à la position indiquée dans la liste i^* et examiner les colonnes de la liste dans l'ordre. Si une colonne j est trouvée telle que :

   $$Q(S) \cap Pj = 0 \text{ et } z(S) + Cj < zbar,$$

   faire l'étape 4.

   $$Si\ z(s) + Cj \geq zbar$$

   ou si la liste i^* est épuisée ou vide, faire l'étape 5.

   Etape 4 : (test pour la solution)

   $$Soit\ S+=S+ \cup \{j\}.$$

   Si toutes les contraintes sont satisfaites, une meilleure solution a été trouvée, posons zbar=z(S) et faisons l'étape 5. Sinon faire l'étape 2.

Etape 5 : (backtracking)

Si S+=0 faire l'étape 6 sinon soit [k] le dernier élément introduit dans S+.

Posons que S+=S+-[k] et i* la liste dans laquelle [k] a été trouvé, et plaçons un indicateur directement à côté de [k] dans la liste i*. Faire l'étape 3.

Etape 6 : (test de fin)

Si zbar = ∞, il n'existe pas de solution de partitionnement, sinon la solution qui a généré zbar est optimale.

Pour notre exemple, nous pouvons montrer que la création des listes se passe de la manière suivante :

- Soit l'entité {1} choisie comme première médiane, nous pouvons former le premier groupe dont elle est l'unique objet.

- La condition de chaînage nous indique dans quel ordre les solutions utilisant l'entité {1} sont générées. Ainsi la deuxième nous permet de créer un groupe de deux entités, en y classant {1} et {3}. La distance est alors de 4.5. La troisième solution pour {1} est un groupe composé de {1},{3} et {5}. La somme des distances est alors de 10.0. Et finalement la sixième solution, avec {1} comme médiane est, les entités sont énumérées dans leur ordre d'entrée, {1,3,5,2,4,6}.

Finalement nous obtenons ainsi 32 solutions potentielles, pour six entités. Comme nous voulons obtenir deux groupes, ces 32 solutions sont à considérer sous forme de deux listes, l'une contenant toutes les solutions où {1} apparaît.

Nous pouvons représenter partiellement ces solutions, sous forme de tableau de zéros et de uns.

solution/ entité	1	2	...	6	...	10	....	32
	1	1	1	1		0		0
	2	0	0	1		1		1
	3	0	1	1		0		0
	4	0	0	1		1		0
	5	0	0	1		0		0
	6	0	0	1		1		1
somme des distances	0	4.5		38.5		7.0		4.0

Nous allons pointer successivement sur chacune des solutions, dans les deux listes. Au départ dans la liste 1, nous pointons sur la solution 1. Dans la liste 2, nous prenons la solution 7. Comme les six entités ne sont à ce moment pas classées, nous devons prendre la solution suivante dans la liste 2. Une fois une solution trouvée dans la liste 2 telle que combinée avec la solution 1 toutes les entités sont classées, nous calculons la somme des distances totale et enregistrons la position courante des deux pointeurs. Puis nous continuons à parcourir la liste 2. A la prochaine solution complète, nous calculons à nouveau la somme des distances et comparons ce résultat avec le précédent. Si le dernier est meilleur, nous le mémorisons.

Une fois que nous avons atteint la solution 32, nous déplaçons le pointeur

dans la liste 1 sur la solution 2. Et nous reprenons le parcours de la liste 2 à la solution 7. L'algorithme s'arrête quand pour la solution 6 de la liste 1, nous avons atteint la solution 32. A ce moment, les pointeurs mémorisés nous permettent de récupérer les solutions contenant les groupes tels que la somme des distances est minimale. Dans notre cas, nous obtenons (1,3,5) et (2,4,6).

Nous pouvons très facilement esquisser les problèmes soulevés par cet algorithme. Nous générons 32 solutions potentielles pour classer seulement six entités dans deux groupes. Nous remarquons que les besoins de stockage de données sont gigantesques. Il faut au moins $2m(n-1)n^3$ itérations pour parvenir à la solution. Chacune des itérations ne se composent que de six opérations au plus. La phase de préparation des données se compose de n(n-1) opérations.

4.1.1.2 Réduction par la méthode de Lagrange

Nous avons vu que le principal problème venait de l'énumération complète des solutions. En modélisant notre problème sous forme d'un lagrangien, nous pouvons éviter cette énumération. Nous allons parcourir partiellement la région réalisable en faisant converger vers l'optimum une fonction sous-gradiente à partir d'une solution réalisable initiale. A chaque itération, nous déterminerons l'ensemble des médianes optimisant localement (et provisoirement) la fonction-objectif. Cette réduction ne se fait pas sans risques, car nous n'avons par cette approche :

- Aucune certitude que la fonction sous-gradiente converge monotonement vers l'optimum de la fonction-objectif originale,

- Aucune garantie que les valeurs trouvées pour Xj forment une solution réalisable pour le problème original.

L'algorithme est le suivant :

1. Trouver la solution réalisable initiale et calculer la valeur de la fonction-objectif pour cette solution (LBARRE).

2. Tant que (LBARRE-LU)/LU > EPSI parcourir l'espace réalisable par le lagrangien

   LU : fonction-objectif pour le lagrangien

   EPSI : valeur de l'erreur admissible entre LBARRE et LU.

2.a Calculer les nouveaux multiplicateurs Ui i=1,...,n

$$Un(1,..n) = Un-1(1,...,n) + TP\ Vk(n-1)(1,...,n)$$

Ui représente l'ensemble des multiplicateurs actuellement utilisés dans la fonction sous-gradiente et Vk[i] vaut 1 si l'entité i n'appartient pas à la solution locale actuellement atteinte. Le facteur TP représente l'accroissement relatif de la valeur de la fonction-objectif au point actuellement atteint par rapport à la norme du vecteur Vk élevée au carré.

2.b Calculer CBARRE tel que

$$CBARRE_{ij} = \begin{cases} d_{ij}-U_i, \text{ si } d_{ij}-U_i < 0 \\ 0 \text{ sinon} \end{cases}$$

pour tout i ≠ j

2.c Calculer $CBAR_J = \sum_i CBARREij$

2.d Déterminer les m médianes en fonction de $CBAR_J$

2.e Assigner les X_{ij} au groupe du leader j si

$CBARREij < 0$

2.f Calculer la valeur de LU

Le détail et la démonstration des diverses étapes de l'algorithme sont donnés dans Dodge et Arthanari (1981).

Sa complexité est de $5kn^2$, où k est le nombre d'itérations nécessaire pour obtenir la convergence entre LBARRE et LU. Les besoins en mémoire sont également moindres, mais restent toutefois importants.

Nous supposons que la solution initiale est donnée par la partition (1,2,3) et (4,5,6), déterminée arbitrairement. Les médianes initiales sont déterminées en prenant la somme des distances minimale à l'intérieur de chacun des groupes. Nous avons ainsi (1) et (6). Nous pouvons calculer la somme totale des distances de cette solution initiale que nous avons appelée LBARRE. Elle vaut 30.0.

Nous commençons les calculs au gradient zéro avec U_0 = (0,0,0,0). Ainsi CBARRE = 0, pour tout i,j et $CBAR_J$ est également nul pour tout j. Nous pouvons choisir n'importe quelle entité comme médiane. Nous utilisons toutefois celles déterminées lors de l'initialisation de la solution. Le vecteur VK vaut ainsi (0,1,1,1,1,0), puisque seuls (1) et (6) font partie de la solution courante.

Nous pouvons maintenant calculer les nouveaux multiplicateurs $U_1 = U_0 + 2 * (0,1,1,1,1,0) = (0,2,2,2,2,0)$. TP a été fixé à 2. Nous pouvons déterminer que CBARRE est nul en tout point, puisque pour tout i,j la différence entre le multiplicateur et la distance est plus grande que zéro. Le processus se répète jusqu'à ce que le critère d'arrêt ait été atteint.

Nous avons pu nous rendre compte que cette méthode ne converge pas. Ainsi, nous avons dans notre réalisation informatique renforcé ce critère d'arrêt, afin d'éviter que notre programme ne boucle.

4.1.2 Minimisation de la somme des carrés dans les groupes

Nous considérons une variante de la fonction-objectif à minimiser. Il ne s'agit plus de la distance à la médiane, mais de la somme totale des carrés entre les groupes. Le problème principal réside dans le fait que la fonction-objectif considérée est non-linéaire. La contribution de la $j^{ème}$ colonne de la matrice X à l'accroissement de la fonction-objectif sera donnée par :

$$C_{ikj} = d_{ikj} - (\sum_{l=1}^{k} d_{ilj})^2/k+1 + (\sum_{l=1}^{k-1} d_{ilj})^2 / k \quad (4.1.7)$$

où k=1,2,...,n-1

Nous utilisons sans changement notable la condition de chaînage et l'algorithme de partitionnement vus ci-dessus. Il suffit de changer le calcul du "coût économique" d'une solution partielle générée. La complexité de cette formulation n'est en rien modifiée par ces changements, car l'énumération complète des solutions partielles reste nécessaire. Nous renonçons à illustrer par un exemple cette variante due uniquement à l'implantation d'une fonction-objectif différente.

4.1.3 Critique

Nous avons déterminé ci-dessus la complexité de ces deux algorithmes. Nous avons vu que pour l'algorithme de partitionnement par énumération, elle est particulièrement pénalisante au point de vue performance. Cependant la réalisation de cet algorithme donne un programme court, avec des instructions très simples (quelques additions ou soustractions à partir de données élémentaires ou de données structurées à un niveau, quelques tests sur ces mêmes données, et la gestion de deux "pointeurs"). Le problème principal consiste à stocker la matrice constituant le système d'équations. Puisque cette méthode ne recourt qu'à des opérations élémentaires et qu'elle procède par énumération complète, elle limite les risques de propagation d'erreur.

La méthode basée sur l'utilisation d'une fonction sous-gradiente de multiplicateurs de Lagrange est bien moins complexe. Certes, il n'est pas possible de déterminer précisément la valeur du facteur k. Cet algorithme utilise également des opérations simples avant tout.

Il n'y a pas de garantie que la fonction choisie converge vers la solution optimale. Nous avons pu le vérifier par simulation. Le problème est la détermination de l'ensemble des multiplicateurs d'une itération à l'autre. L'accroissement n'est plus suffisant après quelques itérations pour que toutes les entités soient affectées à un groupe.

Chacune des méthodes nécessite de la part de l'utilisateur qu'il fixe le nombre de groupes à obtenir. Ainsi, l'une des réserves émises quant à la qualité de la solution fournie par les méthodes classiques se retrouve pour une formulation en programmation entière.

4.2 L'approche par la programmation dynamique

4.2.1 Algorithme

Cette méthode examine un système qui évolue d'un état vers un autre en fonction d'actions accomplies à chaque étape d'une prise de décision. Sur la base de la décision prise, nous avons un résultat "économique", qui peut être un bénéfice, une perte, le coût de l'action, etc.... L'action optimale, c'est-à-dire celle qui permettra d'optimiser le résultat économique, est l'objectif à rechercher.

- Le concept d'action est le classement d'une ou plusieurs entités, dans un groupe donné à une étape connue du processus. Ainsi, les actions de la première étape consistent à classer un certain nombre d'entités dans le premier groupe. A la deuxième, nous traitons les entités du deuxième groupe, en tenant compte des premières actions, et ainsi de suite jusqu'à ce que les n entités aient trouvé leur place dans l'un des m groupes.

Notons que le processus commence à une étape 0, pour laquelle nous définissons un état fictif. A cette étape, aucune entité n'est classée. A chacune des étapes du regroupement, nous ne pouvons classer au maximum qu'un nombre d'entités donné par les formes de distribution.

Entre deux étapes successives, les états sont connectés par des arcs. Pour qu'un arc soit réalisable, il faut absolument qu'une entité contenue dans un état donné de l'étape K-1, le soit aussi dans l'état connecté de l'étape K, pour $1 \leq K \leq M_0-1$. Pour K=0, un arc existe, connectant cet état particulier à tous ceux de l'étape K=1.

L'algorithme utilisé exploite le réseau des actions réalisables par "parcours arrière" (backward value iteration algorithm). Il calcule à chaque itération le résultat de la formule récurrente suivante :

$$W_k^*(z) = \begin{cases} 0 \text{ pour } k = M_0 \\ \min_z \left[T(y-z) + W_{k+1}^*(y)\right] \\ \text{pour } k = M_0, \ldots, 0 \end{cases} \qquad (4.2.1)$$

où

M = nombre de sous-ensembles non-vides et disjoints dans lesquels les n éléments sont classés.

M_0 = M si N ≥ 2M, et N - M si N < 2M

K = index de la variable d'étape.

z = variable d'état représentant un ensemble donné d'entités à l'étape K.

y = variable d'état représentant un ensemble donné d'entités à l'étape K+1.

y-z = sous-ensemble de toutes les entités contenues dans y, mais qui ne le sont pas dans z.

T(y-z) = le coût de transition de l'étape K + 1 à l'étape K.

L'homogénièté d'une partition sera mesurée par le critère :

$$w = \sum_{k=1}^{m} T(y_k) \text{ avec } T(y_k) = (1/n_k) \sum_{i,j \in y_k} (d_{ij})^2 \qquad (4.2.2)$$

L'objectif de cette formulation est de minimiser la somme des carrés à l'intérieur des groupes et de maximiser la somme des carrés entre les groupes, c'est-à-dire d'obtenir les groupes les plus homogénes et le plua disjoints possibles.

4.2.2 Critique

La complexité de cette méthode est une combinatoire entre le nombre d'entités et le nombre de groupes. Pour classer 9 traitementa dans 3 groupes, il faut 3 étapes avec à l'étape 1 456 états, et à l'étape 2 encore 174. Il faut examiner par la procédure de "parcours arrière" 174 possibilités de classification entre les étapes 3 et 2, et entre les étapes 1 et 0. Entre les étapes 2 et 1, il faut examiner de nombreux arcs, pour chacun des 174 états.

En comparaison, l'algorithme de programmation entière, basé aur le partitionnement par énumération, génère 74 classifications potentielles, grâce à la condition de chaînage des entités. Et on n'examine les possibilités que pour les groupes de la liste 1, soit dans notre cas 8.

Il est également possible d'exploiter le réseau par un algorithme de parcours avant ("Forward Value Iteration"), c'est-à-dire de partir depuis l'étape 0 (aucune entité classée). Cette version n'est pas favorable, puisque la complexité est ainsi augmentée. En effet, noua aurions 456 "chemins" différents au dèpart, au lieu des 174 exploités en procédure de parcours arrière.

Le problème de la atabilité ne semble pas se poser. Comme nous traitons des variables discrètes, noua ne faisons aucune approximation. Selon Bellman et Dreyfuss (1965, p.343-349), il se pose dans les cas, où les variables discrètes servent à représenter une fonction d'une variable continue. Ils ont d'ailleurs démontré que la programmation dynamique fournit de bons résultats, comparé aux méthodes d'optimisation classiques (calcul de variations, calcul différentiel, ...).

Nous pouvons nous attendre à ce que la solution proposée par l'algorithme décrit ci-dessus, aoit la solution optimaledu problème.

4.3 Approche par la programmation linéaire

L'algorithme vu ci-dessua permet de résoudre notamment le problème de la "route la plus courte" ou de l'allocation minimale de ressources. Ce genre de problème se résout également par la programmation linéaire.

4.3.1 Algorithme

Pour résoudre le problème ainsi posé, nous utilisons la propriété de complémentarité entre le problème primal d'un modèle linéaire et son dual. Nous nous servons d'une méthode similaire à celle utilisée pour résoudre un problème de transport.

Le primal s'exprime sous la forme :

Minimiser Cf

sc $Af = b$

$f \geq 0$

Alors que le dual est :

Maximiser $b'w$

sc $A'w \leq C'$

w sans restriction dans le signe

Avec :

f = $(f_{0,1},\ldots,f_{i,*})$, le vecteur des arcs liant les états dans une solution réalisable

C : le coût de transition d'un état à l'autre

A : la matrice linéaire représentant le réseau

b'= (1,0,0,...,-1)

w = $(w_0,\ldots,w_*)$ le vecteur des variables duales

Posons sous une forme partiellement étendue le problème dual :

$$\begin{aligned}
&\text{Maximiser } w_0 - w_* \\
&\text{sc} \\
&\qquad w_0 - w_1 \leq C_{0,1} \\
&\qquad w_0 - w_2 \leq C_{0,2} \\
&\qquad w_0 - w_3 \leq C_{0,3} \\
&\qquad\quad \ldots \\
&\qquad w_1 - w_6 \leq C_{1,6} \\
&\qquad\quad \ldots \\
&\qquad w_1 - w_* \leq C_{1,*} \\
&\qquad w \text{ sans restriction dans le signe}
\end{aligned}$$

De la même manière que nous le ferions dans un problème de transport, nous pouvons déterminer simplement la valeur de w_0, puis résoudre séquentiellement le système ci-dessus. La valeur pour w_0 est donnée par :

$$w_0 = \min_{(0,k)\ \varepsilon\ A} C_{0,k}$$

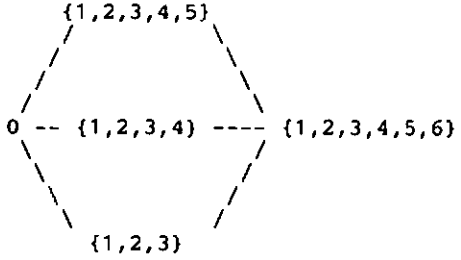
La résolution du système dual permet de déterminer les arcs à éliminer du réseau, si la différence $C_{i,j}-[w_i-w_j]$ est plus grande que zéro.

Ce procédé de calcul est répété jusqu'à ce qu'une "route" soit trouvée entre le node 0 et le node *, au travers du réseau restreint formé ainsi.

L'algorithme complet, pour représenter le problème de classification automatique en modèle de programmation linéaire est :

1. Trouver les formes de distribution des m objets en k groupes
2. Générer le réseau des états.
3. Déterminer le coût de transition le long de chacun des arcs réalisables.
4. Appliquer la méthode primale-duale jusqu'à ce que la route soit trouvée entre le node 0 et le node *.

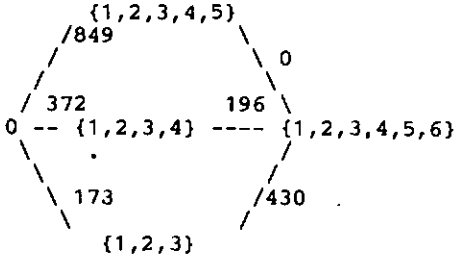
Nous avons le réseau d'états partiel suivant pour notre problème de référence.



Sur chacun des arcs générés, nous pouvons calculer le coût de transition d'un état à l'autre. Il s'agit du coût en distance par l'ajout des entités supplémentaires pour arriver à la solution sur laquelle l'arc pointe. Ainsi le coût de transition pour l'arc, qui relie la solution initiale, aucune entité classée, à la solution {1,2,3} est égale à $(8.5^2 + 4.5^2 + 9.0^2)$, soit 173.5.

Pour calculer le coût le long de l'arc entre {1,2,3} et {1,2,3,4,5,8}, nous considérons que nous formons pour ce passage le groupe {4,5,6}. Ce coût de transition est la somme des carrés dans le groupe ainsi déterminé. Il est égal à $(15^2 + 3^2 + 14^2)$, soit 430.

Nous représentons rapidement quelques coûts de notre réseau.



Le coût le long de l'arc reliant {1,2,3,4,5} à {1,2,3,4,5,6} est nul, car le groupe

formé ne contient que l'entité {6}.

Nous pouvons poser le programme dual partiel suivant :

$$
\begin{array}{lll}
\max & w_0 - w_* & \\
sc & w_0 - w_1 & \leq 849 \\
 & w_0 - w_2 & \leq 372 \\
 & w_0 - w_3 & \leq 173 \\
 & w_1 - w_* & \leq 0 \\
 & w_2 - w_* & \leq 196 \\
 & w_3 - w_* & \leq 430 \\
 & w \text{ sans restriction dans le signe} &
\end{array}
$$

Nous déterminons la valeur de w_0, soit le minimum aur {849,372,173} =173. Puis nous résolvons séquentiellement toutes les équations. Ici, la résolution n'a pas grand sens, vu que nous avons négligé les 90% du résesu au moins. Une fois toutes les valeurs pour w_i connues, nous pouvons par le calcul $C_{ij} - (w_i - w_j)$, éliminer tous les arcs pour lesquels cette différence est plus grande que zéro. Finalement, nous trouvons la solution habituelle, soit les groupes {1,3,5} et {2,4,6}.

4.3.2 Critique

Les problèmes de complexité soulevés par le modèle de programmation dynamique ne sont pas résolus par l'application de cette méthode. En effet, il s'agit dans une première étape de déterminer également le réseau complet et de calculer le coût de transition le long de tous les arcs réalisables, en exploitation avant du réseau. Nous avons vu précédemment que cette solution n'était pas recommandable, qu'il valait mieux exploiter par la procédure de retour arrière. Ensuite seulement, nous commençons à résoudre le problème proprement dit. La résolution du dual conduit à considérer un système d'équations très nombreuses, une par arc réalisable. Le réseau est parcouru en entier au minimum deux fois au cours de la procédure complète.

4.4 Approche Branch and Bound

4.4.1 Algorithme

Envisageons la façon suivante de classer les n entités en m groupes : Choisissons n=9 et m=3 pour limiter notre problème. Prenons tout d'abord les m groupes vides, choisissons ensuite de classer le premier élément de N et classons-le dans le groupe J_1. Examinons maintenant les possibilités de classer le deuxième élément. Nous pouvons soit le classer dans J_1 soit dans J_2 ...

Ce qui nous donne deux solutions partielles. Cherchons ensuite à classer le troisième élément de la liste. Il peut s'ajouter dans la première solution issue du classement du deuxième : soit dans J_1, soit dans J_2.

Mais il est également possible de reprendre à partir de la 2^e solution (de classification pour l'élément 2). Auquel cas, notre troisième élément peut prendre place soit dans J_1, soit dans J_2, soit dans J_3. En poursuivant ce raisonnement, nous pouvons construire l'arbre des solutions partielles suivant (seule une partie de l'arbre est représentée) :

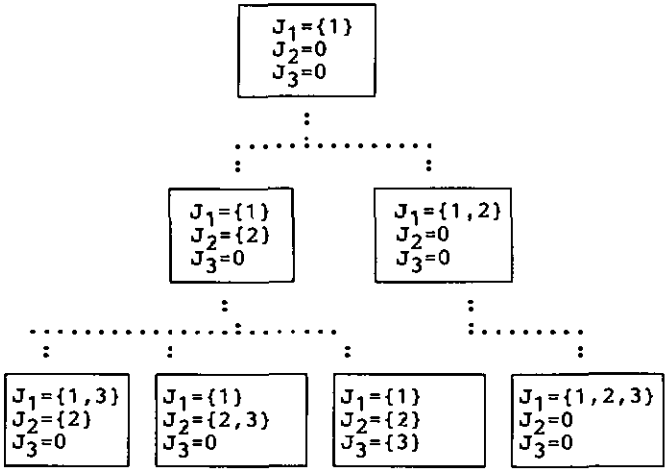


Figure 4.4.1 : Arbre des solutions de classification

Le Branch and Bound est une méthode permettant de construire un "arbre" en élaguant les branches inutiles, et déterminant finalement la solution optimale su problème.

Considérons le problème de partitionner N en m groupes tels que la fonction-objectif suivante soit minimisée :

$$C'(J) = \sum_{i=1}^{m} r(J_i) \qquad (4.4.1)$$

où $J=(J_1,J_2,\ldots,J_m)$ le sous-ensemble des m groupes
et r le critère d'homogénéité des groupes

L'algorithme de "Branch and Bound" se caractérise par le calcul de *bornes* sur les solutions partielles et par des règles de construction de ces dernières. Ces bornes correspondent à la variation que subit le coût total d'une solution partielle lors de l'sjout d'un élément supplémentaire. Pour notre problème de classification, nous considérerons les deux bornes suivantes :

- La première borne est le coût de non-classification des n-q objets. Il s'agit d'un *coût d'opportunité*, car par rapport à l'objectif fixé, nous dépensons une certaine somme à ne pas avoir tous les objets classés et notre fonction-objectif minimisée.

- Pour chaque solution partielle, nous pouvons déterminer de manière

précise quel est *l'accroissement de la fonction-objectif*, lorsqu'un des objets non-encore classés vient prendre place dans l'un ou l'autre des groupes. Ce coût sera nul, si nous générons le descendant d'ordre (q+1,a+1) par notre première règle de branchement.

Nous pouvons connaître à tout moment la valeur de la fonction-objectif pour toute solution partielle. La borne totale d'une solution partielle sera la somme de la fonction-objectif partielle et du maximum entre le coût d'opportunité et l'accroissement de la fonction-objectif décrits ci-dessus.

Le coût d'opportunité est donné par la formule :

$$\delta(\bar{N}_q) = (b-1)/2\left(\sum_{i=1}^{m-r} a_i\right) + (b/a)\sum_{i=m-r+1}^{m} a_i \qquad (4.4.2)$$

où b=(1/m), r=1-int(1/m)m, a_i = les d_{ij} des éléments non classés, triés par ordre croissant et "int" la partie entière.

et l'accroissement de la fonction-objectif par :

$$v(N_q) = \sum_{j \in N_q} \mu_j \qquad (4.4.3)$$

avec

$$\mu_j = \max\{0, \min_{1 \le k \le n}\left(\frac{S(J_k;j)-S(J_k)}{j-q+q_k} - \frac{S(J_k)}{q_k}\right)\} \qquad (4.4.4)$$

où

$$S(J_k) = q_k r(J_k), \quad S(J_k;j) = \sum_{i \in J_k} d_{ij} \qquad (4.4.5)$$

Nous considérerons la borne totale suivante pour chacun des noeuds de l'arbre généré.

$$\alpha(J) = C'(J) + \max(v(\bar{N}_q), \delta(\bar{N}_q)) \qquad (4.4.6)$$

Nous pouvons représenter quelques étapes du calcul pour notre exemple. Nous avons au départ une classification composée de deux groupes J_1 et J_2 vides. Il s'agit de l'état d'origine.

Nous décidons de commencer en classant l'entité (1). Nous la plaçons par simplicité dans le groupe J_1. Nous obtenons la situation :

$J_1 = \{1\}$
$J_2 = \{\ \}$

Nous devons classer l'objet numéro 2 dans l'un des deux groupes ci-dessus. Par les règles de branchement, nous générons deux descendants : Pour l'un nous avons :

$J_1 = \{1\}$
$J_2 = \{2\}$

Et l'autre :

$J_1 = \{1,2\}$
$J_2 = \{\ \}$

Nous évaluons ces deux noeuds afin de déterminer lequel sera branché, pour réaliser la classification de l'entité (3). Nous illustrerons les calculs pour le premier des noeuds branchés.

$$\text{Le coût d'opportunité } \delta(N) = (1/2) \sum_{i=1}^{2-0} (a_i) + 0$$

Nous ne calculons que la première partie de cette somme, car la seconde est nulle. Les a_i représentent les distances entre les éléments 3,4,5,6 dans leur ordre croissant, soit (3.0,4.5,5.5,..)

Le coût d'opportunité vaut 3.75.

L'accroissement potentiel de la fonction-objectif est facile à calculer. Nous déterminons cet accroissement dû au classement potentiel de l'entité (3). Tout d'abord le coût de transition par rapport au groupe J_1, nous le notons C pour plus de facilité.

$$C = \frac{4.5^2 + 8.5^2 + 9.0^2 - 8.5^2}{3-2+2} - \frac{8.5^2}{2} = -2.35$$

Pour le deuxième groupe, nous avons par définition un coût nul, puisque nous ajoutons une seule entité à un groupe vide. Ainsi, le coût d'opportunité d'ajouter l'entité 3, sera nul. Nous devons procéder de même pour les entités 4,5 et 6. Et nous additionnons finalement tous ces coûts partiels. Puis la borne totale est calculée. Ensuite le descendant le plus favorable est branché.

Cette démarche se poursuit jusqu'à ce que toutes les six entités sont classées. Nous déterminons la somme des carrés totale dans les groupes pour la première classification complète. Finalement, nous utilisons une méthode de backtracking pour améliorer la solution.

L'algorithme complet est :

1. Choisissons tout d'abord la classification vide représentée par le noeud (0).

2. Brancher ce noeud en appliquant la règle de branchement qui convient :

- Si $0 \leq q < n - m + s$ et $0 \leq s < m$, générer les s+1 descendants $s_1, s_2, \ldots, s_{s+1}$; soit s groupes d'ordre (q+1,s) et un d'ordre (q+1,a+1);

- Si $q = n - m + s$ et $a < m$, générer le descendant s_1 de a en assignant le $(q+i)^e$ élément au $a+1^e$ groupe, pour $1 \leq i \leq m-s$. s_1 est un noeud d'ordre (n,m), soit une classification complète.

- Sinon générer les m descendants de s. Chacun d'eux est d'ordre (q+1,m).

3. Calculer les bornes des descendants du noeud branché.

4. Choisir comme prochain noeud, le descendant pour lequel la borne est minimale.

5. Brancher ce noeud et générer ses descendants, si l'on obtient une classification complète, faire 6, sinon faire 3.

6. Calculer les bornes des derniers descendants générés.

7. Comparer la valeur de la fonction-objectif pour la classification complète actuelle et les bornes des noeuds non encore branchés. S'il existe un noeud pour lequel la borne est inférieure à la valeur de la fonction-objectif, reprendre en 4. Sinon stop, la classification complète actuelle est la classification optimale en m groupes pour l'ensemble N.

4.4.2 Critique

Cet algorithme est d'une complexité combinatoire entre le nombre d'éléments à classer et le nombre de groupes à obtenir. Il faut au départ $m^2(n-1)$ itérations pour obtenir la première classification complète. L'optimisation augmente fortement ce nombre d'itérations, qui peut dans le cas le moins favorable atteindre C^n_m. A chaque itération, on exécute $m^2q(n-q)$ opérations arithmétiques simples pour calculer les bornes des fils générés, plus m^2 calculs de minmax pour déterminer le noeud à brancher. Le tout est inclus dans deux boucles répétitives [DO WHILE], l'une pour la recherche de la solution initiale, l'autre pour l'optimisation de la solution.

Dans le meilleur des cas, la complexité de l'algorithme est de l'ordre de $m^3q(n^2-nq-q)$ opérations. Il est nécessaire de stocker de très nombreux résultats intermédiaires lors d'une réalisation informatique. Un logiciel réalisé selon cette approche "Branch and Bound" sera par conséquent peu performant.

La faiblesse principale de cet algorithme tient dans le fait qu'il faut développer une branche de l'arbre en profondeur pour trouver une première classification complète. Ceci nécessite déjà $m^2(n-1)$ itérations. C'est à partir de cette solution initiale qu'une technique de "backtracking" va calculer l'optimum. Deuxièmement, le choix des éléments à classifier est effectué sans souci d'optimisation. Le professeur Rousseeuw a montré que cette méthode était utilisable avec au maximum 50 objets à classer.

4.5 Application d'une norme L_p à la classification

La norme Lp est un outil mathématique qui s'applique à un espace de dimension n. C'est-à-dire que ce dernier possède un nombre n de vecteurs linéairement indépendants.

Lorsque nous choisissons un intervalle I arbitrairement dans R, nous pouvons montrer que les fonctions appartenant à l'espace des fonctions continues sur I peuvent être intégrées.

On pose de plus que :

$$L_p = ||f||_p = \left(\int |f(x)|^p dx\right)^{1/p} \qquad (4.5.1)$$

Par simple substitution, nous pouvons montrer que :

$p=1$: $||f||_1$ est la norme moyenne

$p=2$: $||f||_2$ est la norme moyenne quadratique

L'intérêt principal de ces normes mathématiques, en classification automatique est qu'elles nous donnent diverses unités de mesure de distances éprouvées.

La norme L_1 serait celle dont l'usage est à recommander en priorité. Nous pensons toutefois que si les données ont été centrées et réduites la norme L_2 est parfaitement utilisable. Toutes nos simulations nous ont montré que la meilleure combinaison est d'utiliser la distance de Manhattan avec la somme des distances aux médianes comme fonction-objectif.

Chapitre 5

Réalisation informatique

5.1 Analyse et structure d'un logiciel basés sur les méthodes de programmation mathématique

Pour les besoins de cette étude, nous avons développé les algorithmes nous intéressant sur des machines de type VAX-11. Les méthodes classiques, auxquelles nous confrontons les algorithmes issus de la programmation mathématique, ont presque toutes été implantées à partir d'une librairie de routines statistiques. Tous les algorithmes de programmation mathématique ont été réalisés par nos soins.

Toute la programmation a été réalisée en langage FORTRAN, en utilisant les extensions du constructeur quand cela nous permettait d'améliorer la clarté et la lisibilité du code produit. Nous garantissons de cette façon que les différences relevées dans les performances des algorithmes ne sont dues qu'à leur complexité.

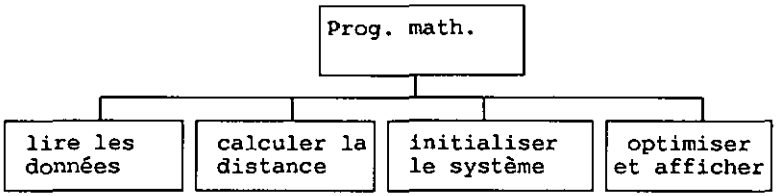
Nous avons relevé précédemment la complexité des méthodes issues de la programmation mathématique. Elle pose des problèmes lors la conception informatique de ces algorithmes. Deux sont à résoudre en priorité : la taille des tableaux de données et une bonne structuration des programmes de sorte à réduire le "swapping". En outre, afin de disposer d'une capacité de traitement suffisante, il est nécessaire de "swapper" nous-même une partie des données. Nous revenons ci-dessous sur le problème plus spécifique de fournir une interface convenable à l'utilisateur.

Nous avons fait un usage intensif de l'instruction COMMON pour la déclaration des données. Nous nous sommes rendu compte que la grande place occupée en mémoire provoquait des résultats "étranges", sans cette précaution. Nous avons poussé ce concept jusqu'à déclarer toutes les données dans un fichier séparé inclus lors de la compilation de tout module.

Pour réduire la taille des données en mémoire et de là le "swapping" de notre programme, nous avons dû recourir à l'utilisation de fichiers intermédiaires. Tous les résultats inutiles y sont écrits de sorte à libérer de la mémoire. Cela a pour conséquence de nombreux accès en mémoire secondaire (disque magnétique) et le ralentissement des calculs !!! Mais l'effet est pire si nous laissons le système d'exploitation "swapper" tout notre programme. Quand cela est possible nous avons regroupé les sous-programmes en espérant réduire le nombre de pages nécessaires. Finalement, nous avons conclu que ces programmes ne peuvent être réalisés qu'en mode batch, sur un système multi-utilisateurs !!! Nous aurions pu éviter ces problèmes en travaillant sur PC avec un disque virtuel ou de la mémoire étendue. Cependant nous ne disposions pas de telles possibilités lors de la première phase de développment réalisée sur

VAX.

Tous les programmes ont été analysés selon une approche "Top-down". La structure est très semblable de l'un à l'autre, aussi nous contentons nous de présenter un diagramme de Jsckson succinct valable pour toutes les méthodes de programmation mathématique.



Le calcul de la distance comprend à choix l'utilisation de la norme L_1 ou de L_2. L'initialisation du système d'équations varie d'une méthode à l'autre et peut se composer de un à trois modules. Un programme compte six à dix modules.

Les noms symboliques sont standardisés et respectent la dénomination msthématique utilisée au chapitre 4.

Le problème principal de cette réalisation est l'espace mémoire utilisé. Nous avons pu bénéficier du fait que la plupart des données entières ont une valeur comprise entre 1 et 32297. Nous les avons déclarées en INTEGER*2, économisant ainsi la moitié de la mémoire occupée par ce type de données.

Pour réaliser une version commerciale de ces programmes, nous devons nous pencher plus sur ce problème. Il s'agit de voir quelles sont les données à conserver impérativement en mémoire et comment ?

Nous pouvons nous demander à juste titre s'il est nécessaire de garder trace de la matrice des observations après calcul de la distance. De plus, la matrice des dissimilarités elle-même ne doit pas être conservée sous forme de tableau à deux dimensions. Il suffit d'une dimension et d'accéder à ce tableau par une fonction.

Enfin, il convient de se pencher sur les résultats à fournir. Pour le moment, nous obtenons seulement la répartition des entités dans les groupes. Cette évaluation devrait permettre une décision sur le maintien ou non de certaines données en mémoire.

5.2 Mode d'emploi du programme

Nous devons traiter d'abord le problème de la réalisation sous forme de package de ces programmes batch. Comment en fait concilier cette nécessité technique avec le besoin de réaliser un interface correct pour l'utilisateur ?

La solution la plus simple est représentée par l'état actuel de la réalisation. L'utilisateur prépare ses données selon un modèle standard dans un fichier qui porte un nom défini d'avance. Ensuite, il appelle la procédure, lançant l'exécution du programme choisi, par un symbole prédéfini dans le système. Les

résultats sont écrits dans un fichier portant lui aussi un nom standard. La maintenance est simple, c'est un avantage non négligeable.

L'inconvénient majeur est le peu d'égards pour l'utilisateur qui se voit imposer les contraintes suivantes :

1. Entrée des données avec l'éditeur;
2. Nom imposé du fichier qui ne représente rien pour lui;
3. Le fichier de résultat porte lui aussi un nom peu représentatif;

Nous pouvons imaginer la mise à disposition d'un interface interactif qui remplirait les fonctions suivantes :

1. Entrée interactive des données ou par fichier de l'utilisateur;
2. Saisie du nom à donner au fichier des données et au fichier de résultats;
3. Sélection de l'imprimante sur laquelle le fichier des résultats sera finalement imprimé;
4. Sélection de l'algorithme;

L'utilisateur appelle le package par un symbole. Il répond aux questions suivantes s'il désire entrer les données interactivement :

Voulez-vous entrer les données par le clavier ou par un fichier (C/F) : ?

Nombre de traitements ?

Nombre de groupes à obtenir ?

Choix de la métrique : 1 = norme L1, 2 = norme L2 ?

Nombre d'observations par traitement ?

Saisie des données pour l'entité X :

Voulez-vous sauver vos données dans un fichier ?

[Si c'est le cas, on demande le nom du fichier]

Nom du fichier des données ?

Nom du fichier de résultats ?

Nom de l'imprimante à utiliser ?

Si l'utilisateur choisit de préparer son fichier des données par un autre moyen (éditeur, autre programme de statistique), il aura le dialogue suivant :

Voulez-vous entrer les données par le clavier ou par un fichier (C/F) : ?

Nom du fichier des données ?

Nom du fichier de résultats ?

Nom de l'imprimante à utiliser ?

Le fichier des données devra alors suivre le modéle suivant :

ligne 1 : nombre de traitement

ligne 2 : nombre de groupes

ligne 3 : choix de la métrique 1 = norme L1 2 =norme L2

ligne 4 : nombre d'observations

ligne 5 : observations pour l'entité 1

ligne 6 : observations pour l'entité 2

....

ligne X : observations pour l'entité n

Toutes les saisies de données numériques (respectivement lecture) sont réalisées en format libre FORTRAN.

Lora d'une saisie interactive, la routine écrit toutes les données dans un fichier qui respecte ce format. Si l'utilisateur ne veut pas la sauvegarde des données, le package utilise alora le nom standard INPUT.DAT. Ce fichier est effacé à la fin de la procédure de classification. Nous avons pensé utile que l'utilisateur puisse sauver ses données saisies interactivement dans un fichier de son choix, afin de lui permettre d'effectuer une deuxiéme classification avec une autre méthode.

Le principal problème réside dans les difficultés de maintenir cet interface, car il contient des instructions en langage de commande. Ces instructions, de même que certaines définitions système, changent parfois!!!

Finalement ce concept a été abandonné, nous avons choisi de réaliser les programmes types de cette étude aur PC. Leur mode d'emploi figure dans l'annexe 1. Ces programmes peuvent être portés très facilement sous tout environnement disposant d'un compilateur FORTRAN.

5.3 Quelques considérations sur des produits disponibles à l'Université de Neuchâtel

Nous présentons rapidement les produits informatiques qui ont servi d'étalons aux programmes décrits ci-dessus. Cette aection et la suivante, qui présentera quelques autres réalisations courantes, montrerons que seules les méthodes classiques sont implantées dans des progiciels statistiques.

Parmi les produits disponibles à l'Université de Neuchâtel, il convient de distinguer les réalisations internes des produits commerciaux. Nous examinons d'abord les fruits des efforts des collaborateurs de notre institution.

5.3.1 Réalisations internes

La première présentation concerne le package ANAFAC. Il permet de réaliser une analyse factorielle . Il offre les deux analyses suivantes : analyse factorielle en composantes principales ou des correspondances. Il est complété par des modules de dessin des plans factoriels sur imprimante ou sur plotter.

Les données sont fournies au moyen d'un fichier standardisé. Ce format a été défini pour toutes les réalisations internes étudiées. Les résultats sont écrits dans divers fichiers : résultats numériques, paramètres de la session, données binaires, paramètres de commande du plotter. Ce progiciel permet le calcul de projections d'observations, respectivement de variables, supplémentaires après l'analyse demandée. Il est conversationnel.

La seconde réalisation interne est le progiciel CLAS, qui offre diverses méthodes de classification automatique. Au point de vue interface avec l'utilisateur, il est comparable avec ANAFAC. Il offre les possibilités suivantes :

Classification hiérarchique ascendante

Classification hiérarchique rapide sur données

Classification hiérarchique rapide sur distances

Partition rapide sur données

Partition rapide sur distances

Méthode séquentielle adaptative

Méthode du K-mean

Méthode de H-mean

Méthode des nuées dynamiques

Lissage typologique

Dans le cas d'une classification hiérarchique, il permet le dessin du dendogramme sur une imprimante ou un plotter. Il offre également la possibilité d'exclure du traitement certaines variables ou observations.

Il existe encore le programme CORRES qui effectue une analyse factorielle des correspondances standard selon l'école française. Il est également conversationnel et lit les données dans un fichier du type standardisé.

Tous ces programmes peuvent être exécutés interactivement ou en mode batch.

5.3.2 Les packages commerciaux

L'éventail des progiciels statistiques internes est complété par toute une palette de logiciels achetés.

Nous commençons par celui qui nous paraît le plus performant. Il s'agit de la librairie de routines IMSL. Ce concept de librairie de méthodes est très flexible, il s'adapte à toutes les situations ou presque. La documentation très complète permet une mise en oeuvre rapide (les méthodes utilisées nous ont demandé au plus une demi-journée de programmation !!!) et la compréhension aisée de la démarche proposée par la méthode. Cependant, cette librairie n'est pas accessible pour tout le monde. Son utilisation demande de bonnes connaissances du langage de programmation FORTRAN. Elle ne contient que des méthodes classiques (hiérarchiques, K-mesn). Une application réalisée avec cette librairie a la structure suivante :

Saisie des données (ou lecture du fichier des données)

Appel à la routine de calcul de la distance

Appel à la routine de calcul de la roéthode

Appel à la routine d'interprétation des résultats

Appel à la routine d'impression des résuitats ou impression self-made des résultats

Malgré les avantages énormes apportés par cette solution modulaire, il existe un inconvénient majeur : celui de la maintenance des applications réalisées. Ce produit est comme tous les autres logiciels sujet de temps en temps à des révisions. La dernière nous a valu la mauvaise surprise de voir tous les noms des routines changer !!!

Le progiciel interactif P-STAT-78 offre lui la possibilité de réaliser une analyse factorielle en composantes principales ou itérative, mais ne propose aucune procédure de classification automatique. C'est avant tout un puissant inatrument de maintenance et de manipulation de fichiers de données. Il permet de lire des données à partir de nombreux types de fichiers créés par d'autres logiciels statistiques ou par programmes utilisateur. Les diverses étapes de la procédure de calcul se décrivent par des commandes mnémoniques, qui peuvent être enregistrées dans un fichier. L'exécution peut ae faire interactivement ou en mode batch.

Nous avons examiné le progiciel SPSS/PC+, qui offre la classification automatique et l'analyse factorielle. Il inclut les méthodes suivantes de classification hiérarchique : la méthode du chaînage complet, la méthode du chaînage aimple, la méthode de Ward, la méthode de la moyenne des distances dans les groupes, la méthode de la moyenne des distances entre les groupes, la méthode des centroïdes, la méthode des médianes, la méthode de partition rapide. Le logiciel est totalement interactif.

5.4 Autres logiciels

Au cours de cette recherche, nous avons pu consulter les documentations de nombreux autres logiciels, sans pouvoir malheureusement tous les tester. Nous avons vérifié notre affirmation du chapitre 1 : aucun des logiciels examinés n'offrait de méthodes de classification basées sur la programmation mathématique. L'ensemble examiné regroupe des produits disponibles pour systèmes centraux, pour atations de travail et pour micro-ordinateurs.

L'ensemble représente une dizaine de logiciels.

Tous ces produits se veulent résolument interactifs. Ils sont mis en oeuvre par des commandes avec une syntaxe à respecter. Ces commandes doivent être interprétées avant l'exécution effective de l'algorithme. Il y a là une certaine perte de temps dû au contrôle de syntaxe. Ces produits sont faciles à utiliser, car leurs documentations sont très complètes et souvent bien illustrées par des exemples éducatifs.

5.5 Domaine d'application comparé

Nous voulons faire ici le parallèle entre tous les produits examinés ci-dessus. Nous sommes toutefois conscients que notre produit n'est pour l'instant qu'un prototype.

Notre travail doit porter sur le stockage des données, afin d'augmenter la capacité de traitement. Il nous faut ajouter certaines fonctions qui manquent encore, telles que standardisation des données, lecture d'une matrice de dissimilarités au lieu d'une matrice de données brutes, calcul d'informations concernant les données brutes, puis les données classées.

Le principal avantage de progiciels tels que PSTAT-78 ou SPSS/PC+ est qu'ils permettent une fois la classification effectuée de continuer le travail sans relire les données.

Cet avantage est obtenu aux dépends de la convivialité. A l'usage, la saisie des lignes de commandes s'avère fastidieuse et peu pratique. Il est difficile de connaître la syntaxe complète des commandes. Il est nécessaire de travailler en permanence avec un aide-mémoire à portée de main.

Le meilleur interface n'est pas réalisé actuellement dans ce domaine. Aucun des produits utilisés ou examinés pour cette étude n'offre de solution de saisie en "mode page". Une telle réalisation s'éloigne très rapidement des concepts qui prévalent lors de la réalisation d'une application scientifique : la portabilité, la facilité de maintenance. En ce sens, le programme PAM est une réussite, car nous avons pu l'utiliser tel quel sur PC, de même que sur les VAX. Il nous a suffit de le compiler et de le lier à l'environnement !!! Ce qui est aussi le cas de notre réalisation finale.

En principe, notre produit peut s'utiliser avec n'importe quelles données, tout comme les logiciels examinés ci-dessus. Les problèmes liés à la nature des données et à leur échelle (chapitre 2) sont toutefois à prendre en compte, comme nous le montrerons dans le chapitre 7.

Chapitre 6

Comparaison analytique des méthodes de classification

Nous montrerons ici que les deux courants principaux dans la "philosophie" des méthodes d'analyse des données, soit la classification automatique et l'analyse factorielle ne sont pas à utiliser l'une pour l'autre. En effet, si l'analyse factorielle peut montrer que les données ont une atructure de groupe par leur position dans le plan, elle ne procède nullement à une classification.

6.1 L'analyse des correspondances

Il s'agit en réalité d'une variante de la seconde méthode que nous présenterons : l'analyse factorielle. Aussi nous contenterons-nous d'un exposé sbrégé dans notre seconde partie.

La principale caractéristique de cette technique statistique est qu'elle utilise la distance du chi-carré pour mesurer la proximité de deux entités. Cette particularité en restreint le champ d'application. Les conditions suivantes doivent être respectées :

$$x_i = \sum_j x_{ij} \qquad (6.1)$$

et

$$x_j = \sum_i x_{ij} \qquad (6.2)$$

Elle analyse les profils des lignes et des colonnes de la matrice des données. La proximité des entités est déterminée en appliquant la distance du chi-carré aux profils, notés P dans les formules suivantes. Finalement cette même distance est utilisée sur les profils des observations.

Nous disposons à ce moment de l'analyse de la proximité entre les entités d'une part, et de celle entre les observations d'autre part.

Les étapes de cette méthode peuvent être résumées de la façon suivante :

- La matrice des données brutes doit vérifier toutes les conditions nécessaires à l'utilisation de la méthode.

- A partir de la matrice des données, calculer la matrice Y dont les éléments sont données par

$$Y_{ij} = \frac{p_{ij}}{p_i (p_j)^{\frac{1}{2}}} - p_j^{\frac{1}{2}}$$

- Calculer $C = \sum[p_{ij}/(p_i(p_j)^{\frac{1}{2}}-p_j)]^{\frac{1}{2}}(p_{ij}'/(p_i(p_j)^{\frac{1}{2}}-p_j^{\frac{1}{2}}))$
- Chercher les valeurs et les vecteurs propres de C
- Classer les valeurs propres par ordre décroissant et "renuméroter" les vecteurs propres.
- Sélectionner les cinq premières valeurs et premiers vecteurs propres.
- Projeter les entités sur les axes factoriels.
- Projeter les observations sur les axes factoriels.
- Calculer les contributions des facteurs aux entités et aux observations.

6.2 L'analyse factorielle en composantes principales

L'analyse factorielle en composantes principales est fondée sur le traitement des données brutes. Cette méthode représente les positions des données par rapport à des axes : les axes factoriels. Les entités semblables se trouvent dans le même sous-espace. Les axes factoriels sont déterminés à partir des données centrées et réduites. La distance utilisée est euclidienne.

Les étapes de cette méthode sont :

- Après saisie des données brutes, il convient d'opérer la transformation nécessaire, voir au chapitre 2, soit conserver les données inchangées, soit les centrer, ou les centrer et les réduire.
- Calculer la matrice $C = (1/n)X'X$
- Chercher les valeurs et les vecteurs propres de C
- Classer les valeurs propres par ordre décroissant et "renuméroter" les vecteurs propres.
- Sélectionner les cinq premières valeurs et premiers vecteurs propres.
- Projeter les entités sur les axes factoriels.
- Projeter les observations sur les axes factoriels.
- Calculer les contributions des facteurs aux entités et aux observations.

6.3 La classification automatique

Nous allons présenter par une série de tableaux récapitulatifs les divers

éléments de comparaison retenus. Nous espérons obtenir des critères de choix, qui permettront de décider dans quels cas utiliser plutôt l'analyse factorielle ou au contraire la classification automatique.

Classification automatique	Analyse factorielle
Trouver une typologie	
Ajustement à un modèle	Ajustement à un modèle
Prédiction	Prédiction
Tests d'hypothèses	
Exploration des données	Exploration des données
Génération d'hypothèses	
Réduction des données	Réduction des données
	Classer les données

Tableau 6.1 : Buts des méthodes

Nous pouvons voir sur ce tableau une première divergence entre les deux méthodes. La classification automatique, par le regroupement des données dans des groupes, permet d'atteindre l'un des buts relevés. En revanche, l'analyse factorielle doit permettre d'obtenir la classification des données. Pour l'une des écoles, la classification est le moyen, alors que pour l'autre, elle est l'un des objectifs. Mais il s'agit plutôt d'un objectif secondaire, puisque c'est grâce à la représentation des positions des données dans les plans factoriels, que nous verrons s'il peut y avoir des groupes à rechercher. L'analyse factorielle peut s'employer comme complément à la classification automatique (ou l'inverse), avec pour but de donner une indication sur le nombre de groupes à rechercher lors de la classification des données. Nous avons relevé que le principal problème des méthodes de classification réside dans la détermination de ce dernier. Aucune solution n'est proposée pour résoudre valablement ce problème. Recourir à l'analyse factorielle n'est sans doute pas la moins bonne.

Dans le tableau suivant, nous chercherons à montrer quels sont les outils mathématiques et statistiques, qui servent de base pour ces méthodes. Nous montrerons aussi certains éléments concernant les données à fournir pour débuter une analyse. Nous pensons mettre en lumière les limites (ou restrictions) à l'utilisation d'une ou l'autre des techniques statistiques présentées.

Critères	Méthodes hiérar.	Méthodes partition	Méthodes de R.O.	Analyse factorie
Type des données	quantit.	quantit.	quantit.	quantit. (1)
Transformat. des données	aucune (2)	aucune (2)	aucune (2)	oui
Traitement d'autres types	possible (3)	possible (3)	possible (3)	possible (3)
Nombre de groupes	peut être trouvé(4	doit être donné	doit être donné	est trouvé
Données de base de cal.	matrice distances	matrice distances	matrice distances	variante /covar
Représentation graphique	possible	possible rés final	possible partie	possible rés.fin-
Diverses distances math.	oui	oui	oui	non
Analyse du résultat	visuelle "donnée"	données	données	visuelle
Optimisation	partiel locale	locale	globale	aucune (5)

Tableau 6.2 : Caractéristiques mathématiques des méthodes de classification

(1) Sauf analyse factorielle des correspondances qui restreint à des données positives

(2) Opérées par la méthode uniquement, mais les données doivent éventuellement être centrées, ou centrées et réduites à cause du problème d'échelles de mesure.

(3) Au chapitre 2, nous avons brièvement présenté ce problème de transformation des données qualitatives en données quantitatives par exemple. Il en résulte en général une perte d'information.

(4) Certaines réalisations de ces méthodes nécessitent que le nombre de groupes soit également donné au départ par l'utilisateur.

(5) Ces méthodes optimisent toutefois un critère au sens de la régression des données.

Dans le tablesu 3, nous analysons le comportement des méthodes envisagées, en fonction des buts à atteindre, cf tablesu 6.1.

Critères	Méthodes hiérar.	Méthodes partition	Méthodes de R.O.	Analyse factorielle
Prédiction	possible	possible	non (1)	possible
Trouver une typologie	oui	oui, plus leaders .	oui	oui
Ajustement à un modèle	oui	oui	oui	oui
Hypothèses (tests)	oui	oui	oui (2)	oui
Exploration des données	oui	oui	oui	oui
Réduction des données	oui	oui	oui	oui
Classer les données	technique utilisée	technique utilisée	technique utilisée	objectif à atteindre
Génération hypothèses	oui	oui	oui	oui

Tableau 6.3 : Les méthodes face aux buts à atteindre

(1) Pour obtenir la classification d'une entité supplémentaire, c'est-à-dire sa prédiction, il est nécessaire de procéder à nouveau à toute la procédure en raison de l'optimisation d'une fonction-objectif globale.

(2) Les tests d'hypothèses sont possibles dans la mesure où ils portent sur le nombre de groupes, ou s'il s'agit de tester l'aspect de la classification en fonction d'une valeur prise par l'une des observations. Mais dans les deux cas, la procédure complète de classification devra être exécutée, pour tout nouveau test.

Dans notre quatrième tableau nous mettons en évidence les caractéristiques informatiques, et les constatations "techniques" que nous avons pu relever lors de nos séries de tests. La rapidité, avec laquelle un programme de classification (ou d'analyse factorielle) va permettre de trouver la solution du problème posé, dépend de nombreux facteurs. Nous définissons certains de ces critères :

- Complexité algorithmique : la fonction de complexité d'un algorithme est le nombre d'opérations de base nécessaires pour résoudre le problème posé. Toutes ces opérations de base sont pondérées par un facteur 1, c'est-à-dire sans tenir compte de leur temps d'exécution relatif par le processeur. Cette fonction est souvent déterminée en rapport avec la longueur en bits des données en entrée décrivant le problème.

- Complexité mémoire : Elle dépend de l'espace mémoire requis pour résoudre un problème. Elle se compose de toute la taille occupée par les données, aussi bien en mémoire vive qu'en mémoire auxiliaire ou

périphérique. En général, cette complexité peut jouer un rôle déterminant sur la performance d'un programme, plus particulièrement dans un environnement multi-utilisateurs.

- Performance : C'est une mesure de la vitesse d'exécution d'un programme. Pour l'évaluer, il convient de tenir compte des facteurs suivants :

- Complexité algorithmique : plus elle est grande, moins rapide sera le programme;

- Complexité mémoire : elle a les mêmes effets sur la performance;

- Le genre des opérations de base; comme nous le montrerons ci-dessous, elles ne demandent pas le même temps d'exécution de la part du processeur. Il convient donc de les pondérer par un facteur relatif de rapidité.

- La structuration des données; l'accès a des données atomiques est très rapide, par contre des données très structurées, tableaux multidimensionnels, structures complexes de type enregistrement ralentissent l'exécution.

- Fautes de page : Dans un système d'exploitation en mémoire virtuelle, un programme et sa zone de données sont découpés en pages d'une taille maximale prédéfinie. Seule la page active est chargée en mémoire centrale, les autres sont en mémoire auxiliaire (ou tampon) , ou encore sur disque (mémoire périphérique de masse). Une faute de page se produit au moment où le système d'exploitation doit aller chercher des pages sur disque pour les amener en mémoire centrale. Un gros programme, code très long ou zone de données très importante, est en général pénalisé dans sa performance à cause des fautes de page.

- Besoin de stockage des données en mémoire périphérique, dans certains cas, il est indispensable d'utiliser des fichiers sur disque afin de créer un programme qui ait des capacités de traitement suffisantes quant au nombre de données. Le programme est ensuite écrit de façon à paginer lui-même les données.

Opérations	Pondération
Initialisation, addition, soustraction et comparaison de données simples	1
Multiplication et division de données simples	10
Initialisation, addition, soustraction et comparaison de structures de données	10
Multiplication et division de structures de données	100
Appel de fonctions "systèmes" simples	100
Opérations avec des structures de données à plusieurs niveaux	1000
Appel de fonctions "systèmes" complexes	1000
Initialisation d'un fichier sans clé	1000
Addition à un fichier, modification et annulation en mémoire dynamique	1000
Initialisation d'un fichier avec une clé	10000
lecture consécutive	100000
lecture sélective	1000000

Tableau 6.4 : Pondération des opérations de base sur un ordinateur

Critères	Méthodes hiérarch	Méthodes partition	Méthodes de R.O.	Analyse factorielle
Complexité algorithmique	modérée	modérée	très élevée	modérée
Complexité mémoire	modérée	modérée	très élevée	modérée
Nombre de fautes page	modéré	modéré	élevé	modéré
Stockage des données	mémoire dynamique	mémoire dynamique	mémoires péri/dyn	mémoire dynamique
Nombre accès disque	faible à modéré	faible à modéré	élevé + élevé	faible à modéré
Type d'opérations	assez complexe	assez complexe	simple	assez complexe
Performance	élevée	élevée	faible	élevée

Tableau 6.5 : Caractéristiques informatiques des méthodes de classification

Chapitre 7

Comparaison des méthodes par la simulation

Les études de comportement de ces méthodes sont à notre connaissance peu nombreuses. Nous présenterons les plus intéressantes afin de disposer d'un élément de comparaison. Notre étude porte sur trois points : la convergence des méthodes, l'application à des jeux de données publiés par divers chercheurs, la mise en évidence d'erreurs types.

7.1 Evaluation par l'analyse de la variance du choix d'une norme, d'une méthode

Par cette analyse, nous cherchons à déterminer la convergence des méthodes. Selon Bellman (1965), le problème est l'exactitude de la solution. Il se pose avant tout lors de la représentation d'une équation fonctionnelle d'une variable continue par un ensemble discret. Sauf dans quelques cas spéciaux, où le problème est posé dans ces termes (ensemble de valeurs discrètes), il ne peut s'agir que d'une approximation. Nous devons nous poser la question de la vraisemblance de la solution obtenue par la résolution du problème approché et celle du problème original. Bellman (1965) s'était attaché à examiner ce rapport entre un problème original résolu par les techniques d'équations différentielles et une approximation résolue par les techniques de la programmation dynamique. Aucun de ses exemples ne convient à notre domaine.

Le problème de classification est représenté par un ensemble discret de valeurs. Nous estimons intéressant de déterminer la capacité des méthodes de représenter une réalité fictive continue dans une première phase de simulation. Dans une seconde phase, nous générons des valeurs discrètes qui représentent les valeurs morales d'une analyse statistique en sciences sociales.

Nous allons examiner le comportement des méthodes sélectionnées dans un cadre bien précis. Il s'agit de jeux de données types, générés par simulation de telle sorte que la classification optimale nous soit connue. Ces expériences sont répétées plusieurs milliers de fois, et nous déterminons le pourcentage de réussite des méthodes. Les jeux de données s'appliquent avant tout aux cas dans lesquels nous disposons de l'ensemble de la population, et non d'un échantillon.

Pour chacune de ces phases, les expériences ont été effectuées en utilisant la norme L_2 et la norme L_1.

Pour notre première phase de simulation, nous avons généré des dérivés de lois normales par la méthode de Box et Muller. Une paire de variables normales de même loi est donnée par :

$$X_1 = \mu + (-2\log_e U_1)\ \sigma 2\cos^2 \pi U_2 \qquad (7.1)$$

et

$$X_2 = \mu + (-2\log_e U_1)\ \sigma 2\sin^2 \pi U_2 \qquad (7.2)$$

où U_1 et U_2 sont deux variables indépendantes de loi uniforme [0,1].

Notre modèle de simulation comporte trois lois de moyenne variable, fixant une distance de base, et d'écart-type égal à un. Pour chacune de ces trois lois, nous générons trois entités, en variant le nombre d'observations par entité, entre un minimum de cinq et un maximum de vingt. Une expérience porte sur 100 échantillons. Nous disposons de résultats obtenus sur 10'000 échantillons analysés par la norme L_2 et de 2'000 traités par la norme L_1. Nous avons aussi déterminé le temps de calcul moyen pour chacune des expériences. Ce temps ne donne qu'une indication, car les simulations ont eu lieu sur des machines de puissance différente.

Notre modèle doit permettre de donner des mesures de :

- la convergence des méthodes;
- la qualité de traitement sur des données centrées et réduites, (voir discussion en 2.1.4), c'est-à-dire avoir une estimation de la sensibilité des méthodes à la perte d'information qui suit la standardisation;
- la sensibilité à une variation de la distance entre les traitements analysés;
- la capacité des normes mathématiques de représenter correctement la distance entre les entités;

Les résultats du tableau ci-dessous sont les moyennes obtenues. Avant d'en tirer des conclusions, nous présentons sur un deuxième tableau les extrêmes obtenus lors de cette première phase.

Méthodes	L_2	L_1	Cpu moyen en minutes
Single linkage	51	52	1
Complete linkage	56	55	1
Average between cluster sum of ..	58	59	1
Average within cluster sum of ..	55	54	1
Méthode de Ward	61	62	1
K-mean	58	54	1
K-median	64	65	1
PAM	62	63	1
Branch and Bound version 1	54	58	11
Branch and Bound version 2	62	64	13
Progra. entière réd. par Lagrange	30	29	9
Setpartionning méthode médiane	98	98	33
Setpartionning WCSS	98	98	35
Progr. dynamique	99	99	45
Progr. linéaire	99	99	75

Tableau 7.1 : Résultats pour la méthode "normale".

Méthodes	L_2	L_1
Single linkage	0 100	0 100
Complete linkage	0 100	3 100
Average between cluster sum of ..	0 100	4 100
Average within cluster sum of ..	0 100	4 100
Méthode de Ward	0 100	7 100
K-mean	0 100	6 100
K-median	1 100	10 100
PAM	0 100	4 100
Branch and Bound version 1	0 100	5 100
Branch and Bound version 2	0 100	9 100
Progra. entière réd. par Lagrange	0 90	2 63
Setpartionning méthode médiane	87 100	84 100
Setpartionning WCSS	91 100	84 100
Progr. dynamique	92 100	88 100
Progr. linéaire	92 100	88 100

Tableau 7.2 : Extrêmes pour le modèle normal

Un lecteur averti sera aurpris de ne pas trouver de résultats pour les méthodes d'analyse factorielle. Malheureusement, sur la base des outputs obtenus, nous n'avons jamais été en mesure de retrouver la classification des entités. Nous avons parfois pu émettre des suppositions. Par contre, les informations fournies sont très complètes, et permettent en tout cas de déterminer le nombre de groupes, qu'il faut rechercher.

Quatre méthodes de programmation mathématique, se détachent : les méthodes de programmation entière, la programmation dynamique et la programmation linéaire. Leur plus grande efficacité mathématique est obtenue au prix d'un temps de calcul très élevé.

Si nous nous attendions à ce que la méthode de programmation entière réduite par les multiplicateurs de Lagrange ne donne pas de bons résultats, nous sommes aurpris, en revanche, par ceux de la méthode de Branch and Bound. Si pour la méthode de Lagrange, noua avons pu vérifier empiriquement, la raison pour laquelle elle ne convergeait pas, nous ne pouvons que constater le fait pour la méthode de Branch and Bound. Pour tous les exemples, où une telle approche était proposée, sa convergence était démontrée. Pour la méthode de Lagrange, nous avons pu observer, grâce à un outil de mise au point de programmes (Debugger), que le point faible se situait au slade 2.a de l'algorithme, le calcul des nouveaux multiplicsteurs. L'accroissement donné par TP*VKn-1[1,...,n) tend très rapidement vers zéro. Les multiplicsteurs ne changent plus, et restent sur le même point de la fonction sous-gradiente.

Les résultats des méthodes classiques ne sont pas une surprise, dans la mesure où elles ne visent pas à optimiser un critère global. Nous avons également la confirmation que la moins bonne de ces méthodes est celle du "single linkage". Le choix de la norme mathématique ne semble pas jouer un rôle déterminant sur lea résultats.

Les deux tableaux suivants présentent les résultats obtenus pour les échantillons de type "valeur morale" générés par une méthode de Monte-Carlo. Ces échantillons sont construits de manière similaire à ceux de la première phase.

Méthodes	L_2	L_1	Cpu moyen en minutes
Single linkage	69	68	1
Complete linkage	71	72	1
Average between cluster sum of ..	73	74	1
Average within cluster sum of ..	72	72	1
Méthode de Ward	77	78	1
K-mean	75	74	1
K-median	80	81	1
PAM	79	81	1
Branch and Bound version 1	78	77	11
Branch and Bound version 2	82	82	13
Progra. entière réd. par Lagrange	65	65	9
Setpartionning méthode médiane	99	99	33
Setpartionning WCSS	99	99	35
Progr. dynamique	99	99	45
Progr. linéaire	99	99	75

Tableau 7.3 : Résultats pour la méthode de Monte-Carlo.

Méthodes	L_2	L_1
Single linkage	31 100	33 100
Complete linkage	35 100	33 100
Average between cluster sum of ..	40 100	42 100
Average within cluster sum of ..	49 100	46 100
Méthode de Ward	50 100	47 100
K-mean	46 100	43 100
K-median	51 100	53 100
PAM	49 100	52 100
Branch and Bound version 1	50 100	49 100
Branch and Bound version 2	53 100	52 100
Progra. entière réd. par Lagrange	21 90	22 63
Setpartionning méthode médiane	94 100	91 100
Setpartionning WCSS	93 100	92 100
Progr. dynamique	95 100	94 100
Progr. linéaire	96 100	95 100

Tableau 7.4 : Extrêmes pour le modèle de Monte-Carlo

A nouvesu, les quatre meilleures méthodes sont la programmation entière, dynamique et linéaire, su prix toujours d'un temps de calcul très élevé. Par contre, les résultats des méthodes classiques et de "Branch and Bound" sont meilleurs. La méthode de Lagrange ne donne pas satisfaction. Dans ce cas également, la méthode du "single linkage" s'avère la moins bonne des méthodes

classiques. Le choix de la norme n'a pas d'influence sur les résultats.

Finalement dans une troisième phase, nous avons généré des échantillons des deux modèles présentés ci-dessus, mais en contaminant systématiquement une des observations. Notre but était de voir la réaction des méthodes à cette "contamination". Nous pensions que toutes ne réagiraient pas de la même façon. Le résultat obtenu contredit en partie cette supposition. Presque toutes les méthodes ont détecté et traité la contamination de manière identique : le traitement contaminé, est isolé dans un des groupes. Nous avons obtenu en général la classification suivante :

Groupe 1 : entités 1,2,3,4,5,6

Groupe 2 : entités 7,8

Groupe 3 : entité 9 (celle contaminée)

Le modèle généré devait donner comme classification, sans contamination :

Groupe 1 : entités 1,2,3

Groupe 2 : entités 4,5,6

Groupe 3 : entité 7,8,9

Nous avons observé certaines différences dans la façon de classer les traitements non contaminés, dans les deux groupes restants. Si les méthodes de programmation entière et dynamique donnent un résultat constant, que nous discuterons plus bas, les autres ont des réactions plus diversifiées. Par exemple, la méthode hiérarchique du "single linkage" donne au moins trois variantes de solutions. Dans l'une, nous obtenons la classification suivante :

Groupe 1 : entités 1,2,3

Groupe 2 : entités 4,5,6,7,8

Groupe 3 : entité 9

La deuxième variante nous donne :

Groupe 1 : entités 1,2,3,4,5

Groupe 2 : entités 6,7,8

Groupe 3 : entité 9

Et la troisième :

Groupe 1 : entités 1

Groupe 2 : entités 2,3,4,5,6,7,8

Groupe 3 : entité 9

Il ne s'agit que des variantes pour lesquelles nous avons obtenu une fréquence moyenne d'au moins 10 %.

Les autres méthodes classiques fournissent des résultats absolument comparables, sauf la méthode de K-median, qui trouve dans 40 % des cas en moyenne, la aolution originale, c'est-à-dire qu'elle ne reconnaît pas la contamination. Nous avons constaté qu'elle n'avait jamais isolé le traitement contaminé et qu'elle restituait fidèlement le troisième groupe, avec les entités 7,8 et 9. L'effet de la contamination est que les groupes 1 et 2 contiennent des solutions fort diverses.

Les deux méthodes de "Branch and Bound" se comportent exactement comme les méthodes classiques. La méthode de Lagrange donne des résultats complètement aberrants.

Dernier aspect intéressant, les deux méthodes de programmation entière semblent insensibles à la contamination. Elles trouvent dans 80 % des cas, que le traitement contaminé devait faire partie du même groupe que les entités 7 et 8, soit la solution du problème original. Par contre, la méthode de programmation dynamique repère la contamination et fournit la solution, que nous avons présentée comme cas général.

Nous avons procédé à ces expériences en utilisant les normes L_1 et L_2. Nous n'avons à nouveau constaté aucune différence significative entre les résultats.

Pour conclure nous présentons brièvement quelques travaux semblables. Toutefois aucune de ces études ne s'est attachée aux méthodes de programmation mathématique.

Dans Everitt (1974) nous trouvons une étude comparable à la notre. Diverses méthodes de classification furent soumises à un test basé sur la génération aléatoire d'observations normales pour former des jeux de données à deux dimensions. Il était possible d'examiner visuellement comment se présentaient les données. Dans une première phase, Everitt a examiné le comportement des méthodes en présence d'un seul groupe. Dans une deuxième, ses jeux de données devaient en former deux. Dans ce cas, ils furent générés avec 25 entités et suffisamment distincts. Les distances furent calculées par la norme L_2. Everitt a soumis trois méthodes hiérarchiques, single linkage , centroïdes et Ward à ces tests. Les résultata sont pour Everitt peu satisfaisants. Les méthodes ne retrouvèrent jamais les groupes originaux. Pour les tests avec deux groupes, elles donnèrent les résultats suivants :

Méthode	Groupe 1	Groupe 2
Single linkage	23	24
Centroïdes	17	30
Ward	13	37

Selon Everitt, l'interprétation d'un dendogramme s'avère un exercice des plus difficiles et risque de conduire à des conclusions erronées. Il a constaté que ces méthodes hiérarchiques donnaient de bons résultats sur des groupes aphériques bien distincts les uns des autres. Lors des tests avec le premier type de données, ces méthodes tendraient à montrer qu'il existe deux groupes principaux et quelques données isolées.

Dans leur étude sur les critères d'arrêts des méthodes hiérarchiques, Cooper et Milligan (1985) ont remarqués que les méthodes retrouvaient les groupes attendus dans 50 à 70 % des cas. Pour cette étude, les jeux de données comportaient de 2 à 5 groupes.

7.2 Résultats obtenus sur des problèmes connus

Toutes les données reprises ici ont servi pour présenter la plupart des algorithmes classiques. Nous avons limité notre choix à des jeux de données restreinta aux limites de nos programmes.

Notre premier jeu provient de Hartigan (1975, p. 28). Il a'agit de données sur la criminalité dans diverses villes des Etats-Unla en 1970, tirées de "United States Statistical Abstract 1970". Nous n'avons repris que celles concernant les hult premières villes et avons cherché à obtenir deux groupes.

City	Murd.	Rape	Robb.	Assau	Burgl	Larce	Auto
Atlanta	16.5	24.8	106	147	1112	905	494
Boston	4.2	13.3	122	90	982	669	954
Chicago	11.6	24.7	340	242	808	609	645
Dallas	18.1	34.2	184	293	1668	901	602
Denver	6.9	41.5	173	191	1534	1368	780
Detroit	13.0	35.7	477	220	1566	1183	788
Hartford	2.5	8.8	68	103	1017	724	468
Honolulu	3.6	12.7	42	28	1457	1102	637

Tableau 7.5 : Crimes dans huit villes des Etats-Unis en 1970

Voici les résultats que nous avons obtenus :

Méthode de Single Linkage

Groupe 1 : Atlanta, Chicago, Dallas, Denver, Detroit,

Hartford, Honolulu

Groupe 2 : Boston

Méthode de Complete Linkage

Groupe 1 : Atlanta, Boston, Chicsgo, Hartford

Groupe 2 : Dallas, Denver, Detroit, Honolulu

Méthode de Average Between

Groupe 1 : Dallas, Denver, Detroit, Honolulu

Groupe 2 : Atlanta, Boston, Chicago, Hartford

Méthode de Average Within

Groupe 1 : Atlanta, Boston, Chicago, Hartford

Groupe 2 : Dallas, Denver, Detroit, Honolulu

Méthode de Ward

Groupe 1 : Dallas, Denver, Detroit, Honolulu

Groupe 2 : Atlanta, Boston, Chicago, Hartford

Méthode de K-mean

Groupe 1 : Dallas, Denver, Detroit, Honolulu

Groupe 2 : Atlanta, Boston, Chicago, Hartford

Méthode de K-median

Groupe 1 : Atlanta, Boston, Chicago, Hartford

Groupe 2 : Dallas, Denver, Detroit, Honolulu

Méthode de PAM

Groupe 1 : Atlanta, Boston, Chicago, Hartford

Groupe 2 : Dallas, Denver, Detroit, Honolulu

Méthode de Branch and Bound (version 1)

Groupe 1 : Atlanta, Boston, Chicago, Hartford

Groupe 2 : Dallas, Denver, Detroit, Honolulu

Méthode de Branch and Bound (version 2)

Groupe 1 : Atlanta, Boston, Chicago, Hartford

Groupe 2 : Dallas, Denver, Detroit, Honolulu

Méthode de Lagrange

Groupe 1 : Atlanta, Chicago, Dallas, Denver, Honolulu

Groupe 2 : Boston, Detroit, Hartford

Méthode de Setmed

Groupe 1 : Atlanta, Boston, Chicago, Hartford

Groupe 2 : Dallas, Denver, Detroit, Honolulu

Méthode de Setcar

Groupe 1 : Atlanta, Boston, Chicago, Hartford

Groupe 2 : Dallas, Denver, Detroit, Honolulu

Méthode de programmation dynamique

Groupe 1 : Atlanta, Boston, Chicago, Hartford

Groupe 2 : Dallas, Denver, Detroit, Honolulu

Méthode de programmation linéaire

Groupe 1 : Atlanta, Boston, Chicago, Hartford

Groupe 2 : Dallas, Denver, Detroit, Honolulu

La majorité de nos méthodes trouvent une classification identique. Seules "single linkage" et "Lagrange" fournissent un résultat différent, qui est assurément faux. Nous voyons que même avec un exemple simple, une "mauvaise" méthode ne permet pas de déterminer la solution optimale. Nous ne pouvons pas simplement comparer nos résultats avec ceux de Hartigan, car il a cherché à obtenir, trois groupes, pour les différentes catégories de crimes. De plus, il a utilisé la technique des profils, que nous n'avons pas abordée ici. Nous présentons ses résultats, afin de disposer d'un élément de discussion.

City	M	Ra	Ro	Al	B	L	A-T
Atlanta	+	=	-	-	-	=	-
Boston	-	-	-	-	-	-	+
Chicago	=	=	=	=	-	-	=
Dallas	+	=	-	=	=	=	=
Denver	-	+	-	=	=	+	=
Detroit	=	=	+	=	=	=	=
Hartford	-	-	-	-	-	-	-
Honolulu	-	-	-	-	=	=	=

Tableau 7.6 : Profils des villes pour les crimes
d'après Hartigan

Un "+" dénote une appartenance au groupe de degré élevé pour la catégorie de crime envisagée, un "=" un degré moyen, et un "-" un degré faible. Sur la base des profils, les villes de Atlanta et de Boston doivent être classées ensembles. Le regroupement de Dallas, Denver et Detroit est certainement correct. Les cas de Honolulu et de Chicago sont plus douteux. Il faut probablement chercher une classification à trois groupes. Les profils montrent que Chicago se trouve dans le groupe des villes à plus forte criminalité et Honolulu dans celui des villes à faible taux. Si nous examinons les valeurs disponibles pour chacune des catégories de crimes, nous pouvons remarquer que pour les catégories "Burglary" et "Larceny",

les taux pour Honolulu sont plus élevés que pour Chicago. L'effet est que la distance mathématique entre Honolulu et une ville à faible taux, telle que Hartford sera grande, alors qu'elle est petite avec une ville avec fort taux. En conséquence, mathématiquement, Honolulu est une ville à forte criminalité. Nous pouvons penser que la classification trouvée par presque toutes les méthodes est correcte.

Sur la base de ces mêmes données, nous avons cherché dans une deuxième analyse à obtenir trois groupes. En voici les résultats :

Méthode de Single Linkage

Groupe 1 : Honolulu, Detroit, Denver, Dallas

Groupe 2 : Atlanta, Chicago, Hartford

Groupe 3 : Boston

Méthode de Complete Linkage

Groupe 1 : Atlanta, Boston, Chicago, Hartford

Groupe 2 : Denver, Detroit, Honolulu

Groupe 3 : Dallas

Méthode de Average Between

Groupe 1 : Atlanta, Hartford

Groupe 2 : Boston, Chicago

Groupe 3 : Dallas, Denver, Detroit, Honolulu

Méthode de Average Within

Groupe 1 : Denver, Detroit, Honolulu

Groupe 2 : Dallas

Groupe 3 : Atlanta, Boston, Chicago, Hartford

Méthode de Ward

Groupe 1 : Atlanta, Hartford

Groupe 2 : Boston, Chicago

Groupe 3 : Dallas, Denver, Detroit, Honolulu

Méthode de K-mean

Groupe 1 : Atlanta, Boston, Hartford

Groupe 2 : Chicago

Groupe 3 : Dallas, Denver, Detroit, Honolulu

Méthode de K-median

Groupe 1 : Atlanta, Dallas, Hartford

Groupe 2 : Denver, Detroit, Honolulu

Groupe 3 : Boston, Chicago

Méthode de Pam

Groupe 1 : Atlanta, Boston, Hartford

Groupe 2 : Chicago

Groupe 3 : Dallas, Denver, Detroit, Honolulu

Méthode de Branch and Bound (version 1)

Groupe 1 : Atlanta, Boston, Hartford

Groupe 2 : Chicago

Groupe 3 : Dallas, Denver, Detroit, Honolulu

Méthode de Branch and Bound (version 2)

Groupe 1 : Atlanta, Boston, Hartford

Groupe 2 : Chicago

Groupe 3 : Dallas, Denver, Detroit, Honolulu

Méthode de Lagrange

Groupe 1 : Atlanta, Honolulu

Groupe 2 : Dallas, Denver, Detroit,

Groupe 3 : Boston, Chicago, Hartford

Méthode de Setmed

Groupe 1 : Boston, Chicago

Groupe 2 : Atlanta, Hartford

Groupe 3 : Dallas, Denver, Detroit, Honolulu

Méthode de Setcar

Groupe 1 : Boston, Chicago

Groupe 2 : Atlanta, Hartford

Groupe 3 : Dallas, Denver, Detroit, Honolulu

Méthode de programmation dynamique

Groupe 1 : Boston, Chicago

Groupe 2 : Dallas, Denver, Detroit, Honolulu

Groupe 3 : Atlanta, Hartford

Méthode de programmation linéaire

Groupe 1 : Boston, Chicago

Groupe 2 : Dallas, Denver, Detroit, Honolulu

Groupe 3 : Atlanta, Hartford

Sur la base des profils, il paraît difficile d'admettre que Honolulu se trouve dans le même groupe que Dallas. C'est une des solutions que nous obtenons par près de la moitié des méthodes. Elle est due aux valeurs pour les crimes "Burglary" et "Larceny", qui prennent une part importante dans le calcul des dissimilarités. Ces deux catégories seules classent Honolulu dans le même groupe que Dallas. Elles déterminent d'ailleurs l'ensemble de la classification, pour toutes les méthodes. Les données doivent être centrées et réduites, pour éviter cet effet de taille. Il est intéressant de voir que les diverses méthodes n'y réagissent pas de la même façon, ce que nous voulions montrer ici. A priori, la solution fournie par les méthodes, que la simulation a mis en évidence, doit être la meilleure. Toute autre est incorrecte.

Voici la solution obtenue par la programmation dynamique avec les données centrées et réduites :

Groupe 1 : Dallas, Denver, Detroit, Chicago

Groupe 2 : Atlanta, Hartford, Boston, Honolulu

et

Groupe 1 : Chicago

Groupe 2 : Dallas, Denver, Detroit

Groupe 3 : Atlanta, Hartford, Honolulu, Boston

L'analyse des profils permet de confirmer ces résultats. Le groupe des 4 villes est celles dont les profils ont une majorité de "-". Chicago est caractérisé par une majorité de "=". Cette ville doit être isolée dans un groupe. Ce que nous obtenons. Le cas du groupe de "Dallas" est clair, même sans standardiser les données.

Notre deuxième exemple nous vient également de Hartigan (1975, p 88). Il s'agit de données sur la valeur nutritive de certains aliments, d'après "The Yearbook of Agriculture 1959".

Aliments	Energie	Teneur protéines	Teneur calcium
Boeuf braisé	11	29	1
Hamburger	8	30	1
Roast Beef	13	21	1
Steak de boeuf	12	27	1
Rôti de boeuf	6	31	2
Poulet frit	4	29	1
Poulet rôti	5	36	1
Coeur de boeuf	5	37	2

Tableau 7.6 : Valeur nutritive de huit aliments

Ces données furent classées par Hartigan en trois groupes par la méthode de K-mean. Il obtint les résultats suivants :

Hartigan (K-mean)

Groupe 1 : Roast Beef

Groupe 2 : Hamburger, Steak de boeuf, boeuf braisé

Groupe 3 : Rôti de boeuf, poulet rôti, coeur de boeuf, poulet frit

Nous avons obtenu les résultats suivants :

Méthode de Single Linkage

Groupe 1 : Coeur de boeuf, poulet rôti

Groupe 2 : boeuf braisé, hamburger, steak de

boeuf, rôti de boeuf, poulet frit

Groupe 3 : roast beef

Méthode de Complete Linkage

Groupe 1 : Coeur de boeuf, poulet rôti

Groupe 2 : boeuf braisé, hamburger, steak de boeuf, rôti de boeuf, poulet frit

Groupe 3 : roast beef

Méthode de Average Between

Groupe 1 : Coeur de boeuf, poulet rôti

Groupe 2 : boeuf braisé, hamburger, steak de boeuf

rôti de boeuf, poulet frit

Groupe 3 : roast beef

Méthode de Average Within

Groupe 1 : Coeur de boeuf, poulet rôti

Groupe 2 : boeuf braisé, hamburger, steak de boeuf

rôti de boeuf, poulet frit

Groupe 3 : roast beef

Méthode de Ward

Groupe 1 : Coeur de boeuf, poulet rôti

Groupe 2 : hamburger, rôti de boeuf, poulet frit

Groupe 3 : boeuf braisé, roast beef, steak de boeuf

Méthode de K-mean

Groupe 1 : Boeuf braisé, hamburger, steak de boeuf

Groupe 2 : rôti de boeuf, poulet frit, poulet rôti,

coeur de boeuf

Groupe 3 : roast beef

Méthode de K-median

Groupe 1 : boeuf braisé, hamburger, steak de boeuf,

poulet frit

Groupe 2 : rôti de boeuf, poulet rôti, coeur de boeuf

Groupe 3 : roast beef

Méthode de Pam

Groupe 1 : Boeuf braisé, roast beef, steak de boeuf

Groupe 2 : hamburger, rôti de boeuf, poulet frit

Groupe 3 : poulet rôti, coeur de boeuf

Méthode de Branch and Bound (version 1)

Groupe 1 : Boeuf braisé, hamburger, rôti de boeuf,

poulet frit

Groupe 2 : roast beef, steak de boeuf

Groupe 3 : poulet rôti, coeur de boeuf

Méthode de Branch and Bound (version 2)

Groupe 1 : boeuf braisé, hamburger, rôti de boeuf

poulet frit

Groupe 2 : roast beef, steak de boeuf

Groupe 3 : poulet rôti, coeur de boeuf

Méthode de Lagrange

Groupe 1 : boeuf braisé

Groupe 2 : hamburger, rôti de boeuf, poulet frit

Groupe 3 : roast beef, steak de boeuf

La méthode n'a pas convergé sur un exemple aussi simple de telle sorte que, les aliments "coeur de boeuf" et "poulet rôti" ne sont pas classés.

Méthode de Setmed

Groupe 1 : boeuf braisé, steak de boeuf, roast beef

Groupe 2 : hamburger, rôti de boeuf, poulet frit

Groupe 3 : poulet rôti, coeur de boeuf

Méthode de Setcar

Groupe 1 : boeuf braisé, steak de boeuf, roast beef

Groupe 2 : hamburger, rôti de boeuf, poulet frit

Groupe 3 : poulet rôti, coeur de boeuf

Méthode de programmation dynamique

Groupe 1 : boeuf braisé, steak de boeuf, roast beef

Groupe 2 : hamburger, rôti de boeuf, poulet frit

Groupe 3 : poulet rôti, coeur de boeuf

Méthode de programmation linéaire

Groupe 1 : boeuf braisé, steak de boeuf, roast beef

Groupe 2 : hamburger, rôti de boeuf, poulet frit

Groupe 3 : poulet rôti, coeur de boeuf

Nous obtenons à nouveau des solutions forts divergentes les unes des autres. Une constatation immédiate, la méthode de Lagrange n'est absolument pas fiable, puisque même sur des données aussi simples, elle ne converge pas.

Comme nous étions en droit de l'attendre, l'implantation de la méthode de K-mean dans la librairie de routine est fidèle à celle utilisée par Hartigan. La méthode de Ward nous donne un résultat différent des autres méthodes hiérarchiques. Il s'agit du même que ceux obtenus par les méthodes de programmation dynamique, de programmation linéaire et de programmation entière . La méthode de partitionnement autour des médioïdes (PAM) reprodult également la même classification. Cette solution est probablement la solution optimale au sens d'un critère global.

Pour prouver cette affirmation, nous avons calculé à posteriori, pour chacune des méthodes et sa classification, quelle est la distance totale dans les groupes, en anglais "within clusters sum of distances".

Nous obtenons la matrice des distances suivante en utilisant la norme L_2.

Boeuf braisé	0.0	4.0	10.0	3.0	8.0	4.0	13.0	14.0
Hamburger		0.0	14.0	7.0	4.0	5.0	9.0	10.0
Roast Beef			0.0	7.0	18.0	17.0	23.0	24.0
Steak de boeuf				0.0	11.0	10.0	16.0	17.0
Rôti de boeuf					0.0	5.0	7.0	8.0
Poulet frit						0.0	8.0	9.0
Poulet rôti							0.0	1.0
Coeur de boeuf								0.0

Tableau 7.7 : Matrice des distances avec L2 pour les huit aliments

Méthodes	WCSS
Single linkage	62
Complete linkage	62
Average between	62
Average within	62
Méthode de Ward	35
K-mean	49
K-median	49
PAM	35
Branch and Bound	38
Branch and Bound	38
Lagrange	??
Setpartionning 1	35
Setpartionning 2	35
Progr. dynamique	35
Progr. linéaire	35

Tableau 7.8 : Somme totale des distances pour la classification obtenue sur l'exemple de la valeur nutritive

Nous voyons que la seule solution est celle pour laquelle nous avons la somme des distances dans les groupes minimale, c'est-à-dire 35. Toute autre solution est non-optimale. Cet exemple nous confirme les résultats théoriques de nos simulations. Les méthodes de programmation entière, programmation dynamique et programmation linéaire donnent une solution optimale globalement. La méthode de Branch and Bound n'est pas une approche très satisfaisante. Les méthodes hiérarchiques sont en général très loin de l'optimum. Le critère proposé par la méthode de Ward et celui de la méthode de partition autour des médioïdes semblent assez bons pour que dans certains cas ils permettent de trouver le minimum de la fonction-objectif considérée. Les méthodes de K-mean et de K-median proposent une alternative meilleure que les méthodes hiérarchiques, bien que non satisfaisantes mathématiquement.

7.3 Examen d'erreurs-type

Nous allons examiner quelles peuvent être les erreurs-type de ces méthodes.

Pour les méthodes hiérarchiques, le critère utilisé localement pour assigner une entité à un groupe est insuffisant. La formule de récurrence ne compense en aucun cas cette faiblesse. Sur le seul exemple des valeurs nutritives, nous pouvons relever les erreurs de classification suivantes : "steak de boeuf" et "boeuf braisé" mal classés, soit deux erreurs, sur huit données à regrouper. Ce qui représente une erreur de 25 % de l'échantillon considéré. Ces erreurs se traduisent en terme de coût sur l'optimum théorique par un écart de 27, soit presque un doublement. L'esprit de ces techniques est la cause de ces classifications erronées. Elles travaillent sur des paires d'entités, soit des traitements et/ou des groupes partiels.

Une méthode hiérarchique trouvera immédiatement qu'il faut former un groupe avec "coeur de boeuf" et "poulet rôti", puisque de toutes les paires, c'est celle qui a la plus petite distance. Normalement, la seconde est "boeuf braisé" et "roast beef". Mais comme la matrice des distances est partiellement recalculée en fonction du premier groupe, cette combinaison correcte passe totalement inaperçue. Si nous examinons attentivement la matrice des distances, nous remarquons que "roast beef" est l'aliment le plus éloigné de la première paire trouvée. Ces méthodes vont l'isoler, de sorte que la prochaine paire formée, est "boeuf braisé" et "hamburger". Puis comme "roast beef" est l'aliment le plus éloigné de "Hamburger", il reste isolé lorsqu'un cinquième élément est pris en considération. Cet élément est le "steak de boeuf", puisqu'il est le plus proche à la fois des deux éléments formant notre deuxième paire. Comme il est relativement éloigné de la première, il prend place dans le deuxième groupe. Une paire correcte, au point de vue optimalité, a été ignorée de toute considération par quatre des méthodes envisagées.

Les méthodes de K-mean et de K-median isolent également l'élément "roast beef".

Nous avons constaté que le critère utilisé la somme des distances, ignore totalement que la meilleure paire pour "roast beef" est d'être associé avec "boeuf braisé". Ainsi tel que l'a montré Hartigan (1975, p.88), nous retrouvons comme classification initiale, les groupes suivants :

Groupe 1 : roast beef, poulet frit

Groupe 2 : hamburger, steak de boeuf

Groupe 3 : boeuf braisé, rôti de boeuf, poulet rôti, coeur de boeuf

Le processus heuristique de réassignation des entités ne concerne que deux éléments, de sorte que nous obtenons les classifications proposées ci-dessus, qui sont éloignées de 33 % environ en terme de coût de l'optimum.

Le processus de branch and bound, dans ses deux versions, assigne l'aliment "boeuf braisé" au mauvais groupe. Il n'y a ici qu'une seule erreur, qui se traduit par 10 % de coût supplémentaire. Il est probable que l'erreur provienne du critère d'arrêt du "backtracking" de cette méthode.

Il est plus difficile de s'imaginer, quelles sont les erreurs-type des méthodes de Ward et de PAM. Comme il s'agit de méthodes heuristiques, nous pensons que leur critère d'arrêt est trop restrictif. Pour des cas où les groupes ne sont pas aussi nettement séparés, tel qu'un examen attentif de la matrice des distances permet de le voir, leurs calculs s'arrêtent trop tôt pour obtenir l'optimum effectif. Il ne faut pas oublier que la méthode de Ward souffre du même vice de conception que les autres techniques hiérarchiques, même si son critère d'agglomération est meilleur.

Nous ne reviendrons pas sur les problèmes de non-convergence de la

méthode de Lagrange. Nous soulignerons toutefois qu'elle se traduit par deux faits : des erreurs de classification, du même type que celles constatées pour les méthodes de branch and bound et des cas, la majorité, où des éléments ne sont pas du tout classés.

Chapitre 8

Conclusion et recommandations

Après avoir cherché à améliorer par des astuces de programmation les performances des méthodes de programmation entière, nous avons conclu que tous nos efforts ne auffisaient pas. Noua avons certes, par une meilleure structuration des données et du code, réussi à réduire le temps de calcul pour classer 100 entités en 10 groupes de 15h à 14h. Et pour des cas, avec moins de groupes, le gain de temps est bien plus apectaculaire. Nous avons repris notre étude de ces méthodes au point de vue conceptuel et en présentons une nouvelle approche. Puis, vu les résuitata obtenus, nous nous sommes penchés sur la méthode de programmation dynamique.

8.1 Etude des possibilités offertes par la programmation entière

Nous avons souligné maintes fois que cette définition du problème pour la programmation entière n'est pas satisfaisante conceptuellement.

Au départ, il est sans doute exact de considérer que les n entités puissent être des objets représentatifs. L'effet de cette "explosion" conduit à considérer des équations redondantes dans le système. Soit le sous-ensemble {1,2,3} formé d'objeta appartenant au même groupe, il est représenté trois fois, si la condition de chaînage est astisfaite, avec chacune des trois entités à tour de rôle comme lesder. Mais au point de vue optimalité, seule une des trois solutions doit être retenue, celle dont la distance est la plus petite. L'effet est encore plus grave, si au lieu de la distance à la médiane, nous utilisons la somme des carrés dans les groupes. Nous avons généré dans ce cas trois solutions strictement identiques quant à leur fonction-objectif.

Cette réflexion nous conduit à reconsidérer le partage des solutions générées en m listes pour satisfaire la définition originale de l'algorithme de partionnement par énumération développé par Garfinkel et Nemhauser.

Imaginons que les trois solutions ci-dessus se trouvent dans des listes différentes. L'algorithme de partitionnement va détecter la redondance et nous courons le risque que la solution la meilleure ne soit pas retenue dans les calculs ultérieurs. Le résultat final risque ainsi d'être incorrect. Nous en déduisons les modifications suivantes à apporter à cette représentation par la programmation entière :

1. Seule la liste 1, celle contenant les solutions pour lesquelles l'objet 1 est lesder potentiel, a une longueur fixe. La deuxième commence à la position longueur_ liste_1 + 1. La dernière se termine obligatoirement à la position n*(n-1). Les autres commencent juste après la position couramment pointée dans la liste de rang inférieur. Les bornes des listes se déplacent

ainsi dynamiquement en cours d'évaluation du système d'équations.

2. Si une redondance est détectée, la solution la meilleure au point de vue optimalité est privilégiée, c'est-à-dire que si elle se trouve dans une liste de rang plus élevée, elle est en quelque sorte "poussée" vers la liste inférieure, et vice-versa.

L'effet sur la performance d'une telle redéfinition est mauvais, mais nous pensons améliorer la capacité de l'algorithme à converger. Nous verrons ci-dessous sur la base d'un exemple comment ces modifications se répercutent sur la complexité de l'algorithme.

Nous avons vu ci-dessus quelles sont les limites à nos possibilités de développement de notre implantation de la méthode issue de la programmation entière. Toutefois, cette première étude nous a montré comment nous pourrions réduire la complexité de cet algorithme. Il s'agit de générer un ensemble de solutions potentielles restreint. Dans la formulation de base, nous utilisons les n entités pour cette génération. Mais le résultat final de la classification, m groupes, ne retient nécessairement que m entités caractéristiques, que nous avons appelées médianes ou leaders.

Nous savons maintenant que nous pouvons économiser beaucoup d'opérations en ne retenant que m entités comme médianes pour notre échantillon. Ne risquons-nous pas par cette réduction apparemment prometteuse de perdre la capacité de la méthode à générer une solution optimale dans 98% des cas en moyenne ? Nous avons pu voir par des essais que la détermination des entités représentatives jouait un rôle capital pour la qualité de la solution. La moindre erreur dans cette étape de l'algorithme conduit assurément à une solution non-optimale.

Le problème auquel nous pouvons faire face sera résolu par une première approche simplifiée. Elle permettra de se rendre compte si notre postulat se vérifie au moins au point de vue performance. Nos simulations nous ont montré que les méthodes classiques qui cherchent des objets représentatifs ne sont pas satisfaisantes au point de vue optimalité de la solution obtenue. Nous avons pu vérifier que celle développée par le professeur Rousseeuw donne des résultats intéressants pour notre tentative de réduction. Ses algorithmes de recherche des "médioïdes" fournissent des leaders corrects dans la plupart des cas que nous avons réexaminés. En revanche, l'affectation des autres entités aux groupes nous paraît l'élément faible de cette technique. Pour bien des cas analysés, les leaders sont déterminés correctement, alors que le résultat final de la classification n'est pas l'optimum global. Nous allons essayer de combiner la partie forte des deux algorithmes. Nous concluons que l'algorithme présenté par Rousseeuw permet avant tout de déterminer les objets représentatifs. La classification ensuite est opérée par simple affectation des entités restantes au leader dont chacune est proche, au sens de la distance utilisée.

Notre nouvelle approche se définit par les étapes ci-dessous, nous oublions les problèmes d'échange de données :

1. Trouver le premier ensemble de médioïdes par l'algorithme de BUILD (cf méthode PAM chapitre 2).

2. Améliorer l'ensemble des médioïdes trouvés par l'algorithme de SWAP (cf méthode PAM).

3. Générer les solutions potentielles pour les médioïdes déterminés par les étapes 1 et 2 en utilisant la condition de chaînage définie par Rao pour la

programmation entière (cf chapitre 4).

4. Résoudre le problème de programmation entière ainsi défini par l'algorithme de partitionnement par énumération.

Comparons les effets sur la performance de cette réduction, pour un échantillon de 9 objets à partager en trois groupes, avec celle des deux méthodes originales .

La méthode originale considère que les leaders potentiels sont les neufs objets. Pour chaque objet, nous générons 8 solutions de classification, au total 72. La liste 1, la seule qui est fixe, compte les 8 solutions dont l'élément 1 est le leader. La liste 2 commence à la position 9, la liste 3 se termine à la position 72. La 8n de la liste 2, respectivement le début de la liste 3, se déplace dynamiquement entre ces deux positions. Nous devons examiner très souvent l'ensemble des solutions entre ces deux positions connues. Pour une solution de la liste 1, nous examinons au total $85^{*(8-1)}$ équations dans les deux autres listes.

La méthode réduite trouve par ses deux premières étapes les trois objets représentatifs, au prix d'opérations peu coûteuses, mais nombreuses (voir la méthode de PAM ci-dessous). Par notre étape trois, et la règle de chaînage de Rao, nous générons un système de 24 équations. Le nombre de leaders est limité au strict nécessaire, nous n'avons pas de redondance d'équations et nous pouvons à nouveau fixer le début et la fin de trois listes pour utiliser pleinement l'algorithme de partitionnement. Pour une solution de la liste 1, nous examinons 82 solutions issues des listes 2 et 3.

La méthode PAM détermine rapidement les trois objets représentatifs, puis chacun des six objets est affecté au groupe du médioïde dont il est le plus proche individuellement. Cette dernière phase ne considère qu'un critère très local. Nous la considérons comme le point faible de cette méthode. La détermination des objets représentatifs est assez coûteuse. Elle comporte deux boucles principales à n opérations. Chacune ces boucles principales comporte une à deux boucles secondaires avec au minimum un passage et au maximum n passages. Mais les opérations sont toutes de type élémentaire. L'affectation des entités non représentatives s'effectue par une double boucle de n(n-m) itérations.

8.2 Possibilités offertes par la programmation dynamique

Cette méthode, la plus performante quant à sa convergence, est aussi l'une des plus coûteuses en temps de calcul et en besoin de stockage. Nous allons examiner où se trouvent les possibilités de l'améliorer.

Contrairement aux autres méthodes, elle définit de quelle manière la classification peut se dérouler par la notion des formes de distribution des entités. Nous disposons de guides dans la recherche des états à connecter d'une étape à l'autre, car il n'est pas possible de relier un état d'une forme donnée à un issu d'une autre même à une étape ultérieure.

Mais pour une forme de distribution donnée, à une étape k de l'algorithme, aucune règle ne fixe comment générer les différents états. Nous pouvons trouver les connections entre les états d'une étape à l'autre et calculer le coût de

transition dû à l'établissement de ces ilens par un ensemble de règles. Nous remarquons une redondance des solutions de classification, car les états sont déterminés par des combinaisons des entités. L'état formé par {1,2,3} se représente également par {2,3,1}, ou {3,2,1}, et encore par {3,1,2} ou {2,1,3}. Alors que le coût de transition, par la somme des carrés dans les groupes, est strictement identique dans les 5 cas.

Avec cette formulation, nous générons un réseau formé par 456 élala pour l'étape 1, formation du premier groupe, 174 pour l'étape 2, formation du deuxième groupe, puis un à la dernière étape, si nous voulons classer neuf objets dans trois groupes. Par le retour arrière, nous devons examiner les 174 possibilités de former le groupe 2. Puis pour chaque élat nous devons examiner toutes les possibilités de former le premier groupe dans la forme de distribution qui définit la relation entre les états. Ce nombre varie en fonction de la forme de distribution. Les ares à examiner sont très nombreux. Le réseau comporte 632 arcs au total.

Jensen (1969) s'est rendu compte de ce problème de redondance et propose une formulation alternative pour la réduire. Il examine deux aources de redondanee : d'une part celle que nous svons relevée ci-dessus, et celle due aux formes de distribution. La forme {4,3,2} est équivalente de {4,2,3}. Il suggère une régle fort simple pour éliminer cette dernière cause. Il auffit de fixer que le nombre d'entités à classer lors d'une étape postérieure doit être plus petit ou égal à celui de l'élape en cours. Il estime supprimer la plus grande partie des redondances également au nivesu des solutions. Il ne donne aucune indication précise quant à la façon de réduire le nombre des états redondanta. Il faut appliquer un critère sélectif. Selon ses études sur l'efficacité d'un tel critère aur la performance de l'algorithme, il lui semble que plus la taille du problème augmente, moins il est intéressant marginalement de procéder de la sorte. Il conseille de ne s'occuper en définitive nullement de ce problème de la redondanee tant au nivesu des formes de distribution qu'au nivesu des solutions. Nous préciserons que se comparaison se base par rapport à une méthode énumérant toutes les possibilités de classification pour obtenir les m groupes recherchés.

Il reconnaît que les besoins de stockage dc eette formulation conduisent à atteindre rapidement les ilmites des mémoires rapides des ordinsteurs et qu'il est nécessaire de travailler avec la mémoire auxiliaire lente. Il ne se soucie par eontre pas du temps de calcul nécessaire.

Nous estimons qu'il est au contraire intéressant d'étudier les moyens de réduire eette redondance des solutions. L'algorithme de programmation dynsmique a fournit la meilleure convergenee dans nos aimulations. Par eonvention, sachant que notre programme n'est pas parfait, nous l'avons fixée à 99% pour nos résultats. Jensen pense qu'elle converge dans tous les cas. Les études de Bellman propres à eette technique de calcul vont dans ce sens également.

Nous savons que contrairement à la méthode de programmation entière, nous pouvons "diriger" l'exploration des solutions de classification qui forment le réseau, en exploitant les formes de distribution.

Nous proposons d'aborder conjointement l'élimination des deux aources de redondance relevées ci-dessus. La régle pour éliminer la redondanee des formes de distribution est connue. Nous avons utilisé eette réduction dans notre implantation de l'algorithme.

L'élimination des redondances des solutions passe par un changement de

fonction-objectif. Nous utiliserons la distance aux médianes vue en programmation entière. Nous introduisons également le concept des "lesders" potentiels et la règle de chaînage des entités. Nous avons montré ci-dessus que cette formulation contenait encore des redondances au point de vue de la classification. Conceptuellement, il n'y a pas de redondance puisque les solutions se distinguent par la valeur de la distance à leur médiane. Dans l'application à la programmation dynamique, nous ne courons pas le risque d'ignorer une solution. Si nous considérons les entités {1}, {2}, {3}, en admettant que $d_{12} \leq d_{23} \leq d_{1k}$,et que {1} est le leader potentiel, nous n'aurons que la solution {1,2,3}. Le raisonnement est également valable si {2} ou {3} est considéré comme médiane potentielle. Nous obtenons pour ces trois entités, trois possibilités de classification, distinguables par leur médiane et la distance à cette dernière. La formulation originale de programmation dynamique, avec la somme des carrés, considère cinq solutions avec la même valeur pour la distance.

Dans notre exemple, les formes de distribution suivantes sont possibles, en respectant la règle éliminant les redondances :

{7, 1, 1}

{6, 2, 1}

{5, 3, 1}

{5, 2, 2}

{4, 4, 1}

{4, 3, 2}

{3, 3, 3}

Nous voyons que certaines se recoupent lors de la même étape, ou sur deux voire plusieurs étapes consécutives. Il n'est pas nécessaire de générer tous les états de toutes les formes. Pour l'étape 1, nous n'avons besoin que de 4 au lieu des 7 énumérées. A chacune de ces formes correspondent 9 médianes potentielles. Par la règle de chaînage, nous ne pouvons générer qu'une seule solution par médiane. Nous obtenons 36 états pour cette étape, au lieu des 456 nécessaires dans la formulation d'origine.

Pour générer les états de la seconde étape, nous avons deux alternatives. D'une part, nous pouvons considérer que nous avons le nombre d'entités classées pour les formes ci-dessus :

{6}

{8}

{8}

{7}

{6}

{7}

{6}

Là nous n'avons en réalité que trois cas différents, avec également neuf médianes potentielles. Nous formons 27 états, au lieu de 174. L'étape 3 ne contient qu'un seul état. Nous avons dans cette formulation un réseau de 85 arcs.

La deuxième possibilité, plus élégante quant à l'utilisation de la fonction-objectif, génère toutefois quelques arcs supplémentaires. Lors de la première étape, nous ne changeons pas la façon de procéder. Pour la seconde, nous considérons cette fois le nombre d'entités classées, lors de cette phase. Nous utilisons le vecteur de distribution ci-dessous :

(1)

(2)

(3)

(2)

(4)

(3)

(3)

Nous distinguons 4 formes de distribution. Pour chacune nous pouvons identifier neuf objets représentatifs. Nous obtenons 36 possibilités de classification pour cette étape.

Dans ces deux formulations, nous sommes loin du nombre d'états énorme nécessités par celle d'origine. La pratique nous montrera si "l'appauvrissement" de l'information empêche la méthode de converger.

Les deux formulations, par un usage adéquat des formes de distribution à la génération du réseau, nous permettent de diriger son exploration, en évitant de pointer sur une solution qui n'est pas correcte. Au pire, nous examinons pour chacune des solutions pointées 9 arcs. Pour la première, nous examinons d'office les 27 de la seconde étape lors du premier passage dans l'algorithme (étape finale—>étape précédente). Puis, pour le passage entre la seconde et la première, nous en évaluons 27*9, 243. Entre la première et l'état initial, nous n'examinerons que celles qui sont les meilleures. Le nombre de routes partielles restantes, respectivement les états sélectionnés comme noeuds, est au minimum de une par forme de distribution. Nous parcourrons au minimum 9 arcs. Le maximum est en tout cas inférieur aux 36 états de cette première étape.

Raisonnablement au moins la moitié des arcs sera éliminée, nous aurions vraiment dans le pire des cas 18*9, 162 arcs, à examiner. Au total, nous évaluons entre 432 arcs, le maximum, et 279 au minimum. Nous sommes là toujours très loin du nombre d'arcs qui composent le réseau généré avec la formulation de Jensen. Dans cette dernière, rien que le passage entre l'étape finale et la seconde requière déjà l'examen de 174 arcs. Nous savons ensuite que chacun de ces 174 arcs doit être connecté avec les états de la première étape. Cette opération n'est pas très aisée.

D'une part, nous devons examiner tous les arcs existants pour une forme de distribution afin de savoir si la connection est possible. D'autre part, nous devons déterminer les éléments de transition (ceux qui sont classés par ce lien) et le coût de cette action. Nous retrouvons ce second problème dans notre

première formulation, mais à une plus petite échelle. Par simplification, nous supposerons que le nombre d'arcs possibles entre ces deux étapes par forme de distribution est le carré du nombre d'objets, dans notre cas 81. Nous devons évaluer 14094 arcs entre la seconde et la première étape de l'algorithme avec la formulation d'origine. A partir des 456 états de la première étape, là aussi nous supposons que la moitié est éliminée, il ne reste que 228 arcs à examiner. Au total, sous la condition que notre restriction ci-dessus soit correcte, nous parcourons un réseau de 14496.

La seconde définition de notre nouvelle formulation nous oblige à examiner au départ 36 liens. Puis, nous pouvons nous restreindre pour chacune des formes à n'en examiner que 8 par état de la deuxième étape. Nous savons que seuls ceux pour lesquels le leader est différent sont connectables. Nous devons toutefois examiner si l'intersection des deux sous-ensembles est bien vide.

Nous évaluons au maximum pour cette action (passage de la seconde étape à la première) 288 états. Puis dans le dernier passage, nous pouvons en compter au pire 18, si seule la moitié des liens est éliminée. Au total, nous ne parcourons plus que 342 arcs. Un autre avantage de cette méthode est à relever : nous pouvons déterminer dès la génération du réseau tous les coûts de transition, car nous accédons d'office à la matrice des distances. Ceci n'est pas possible avec la première formulation, puisqu'elle se base sur le nombre total d'entités classées à une étape, et non pas sur celles qui sont effectivement assignées à un nouveau groupe.

A titre de comparaison, nous reprenons les éléments dégagés ci-dessus pour la méthode de programmation entière. La formulation d'origine modifiée examine pour chacune des 8 solutions formant la liste 1 $85^{(8-1)}$ solutions, soit $88^{(8-1)}$ recherches de liens, soit en clair 1'835.008 "arcs". La formulation réduite par la recherche des objets représentatifs 83 examens, 512 en clair. L'algorithme de PAM comporte quant à lui au maximum $2*2*82 + 2*8^{(8-3)}$ boucles, soit en clair 336 boucles d'opérations simples. Pour les méthodes de programmation mathématique, nous n'avons pas relevé le nombre d'opérations d'examen de liens, mais elles se composent de doubles boucles à maximum n passages. En règle générale, nous avons remarqué dans toutes les méthodes que chacune des ces boucles de calcul se composait en moyenne d'une dizaine d'instructions simples. Aussi, toujours sans compter, les opérations nécessaires de traitement des données entrantes et de communication du résultat final, nous estimons que nos méthodes exécutent au total le nombre d'opérations suivantes :

- programmation dynamique : 11741760
- programmation dynamique, réduite 1 : 349920
- programmation dynamique, réduite 2 : 277020
- programmation entière, complète : $1.49*10^9$
- programmation entière, réduite : 414720
- méthode de PAM : 30240

La formulation semble dans ce cas suffisamment prometteuse pour que nous tentions une dernière réduction. En utilisant le concept de chaînage des entités à un leader, nous avons pour le moment considéré que toute entité en est un. Nous avons conservé la définition d'origine développée par Rao. Mais nous pouvons très bien considérer que seuls m objets représentatifs sont intéressants, comme nous l'avons fait ci-dessus pour la programmation entière.

Le nouvel algorithme s'exprime par les étapes suivantes, nous ne les détaillons pas ici avec les équations :

1. Déterminer les objets représentatifs (médioïdes) par "bswap", la méthode développée par Rousseeuw.

2. Déterminer les formes de distribution pour la méthode de programmation dynamique.

3. Générer l'ensemble des états (solutions potentielles) en appliquant la règle de chaînage sur chacune.

4. Optimiser par l'algorithme de programmation dynamique en parcours arrière.

Nous réduisons de la sorte très fortement le nombre des états générés pour chacune des formes de distribution. Toutefois nous devons prendre en considération deux cas limites qui peuvent se présenter :

- Une entité est plus proche de deux médianes qu'une autre;

- Une entité est équidistante de deux leaders;

Ces deux cas sont traités lors de la création du réseau, de telle sorte qu'une entité donnée ne peut appartenir qu'au groupe de sa meilleure médiane.

Nous allons illustrer chacun des problèmes avec un exemple. Nous traiterons tout d'abord du problème de l'équidistance. Soit le fichier de données suivant :

```
1. 1. 1. 1. 1.
2. 2. 2. 2. 2.
3. 3. 3. 3. 3.
4. 4. 4. 4. 4.
9. 9. 9. 9. 9.
```

Nous voulons obtenir trois groupes à partir de ces données. Nous avons deux cas d'équidistance : d'une part l'entité {2} par rapport à {1} et {3} et d'autre part {3} avec {4} et {2}. Supposons que {2}, {4} et {9} sont les objets représentatifs pour cet échantillon. L'entité {1} ne peut faire partie que du groupe dont {2} est le médioïde. Par contre, l'objet {3} peut appartenir à la fois au groupe de {2} et à celui de {4}, ce qui n'est pas envisageable en classification automatique. Nous avons dû trouver une règle qui élimine ce genre de problème :

- Si un objet est équidistant de deux médioïdes, dont l'un est justement celui pour lequel nous construisons la solution; alors, à moins qu'il ne soit le dernier élément manquant au groupe et à la fois le dernier des objets que nous puissions classer, il n'entre pas dans la solution en cours.

Le deuxième cas est assez rapidement illustré, soit les quatre entités suivantes et leur distances :

```
A 0.
B 9. 0.
C 13. 11. 0.
D 15. 19. 12. 0.
```

A et C sont les objets représentatifs de l'échantillon.

Nous constatons que {B} est l'élément le plus proche des deux médioïdes. Nous risquons, lors de la construction du réseau des états de générer les solutions {A,B} {C,B}, pour la même forme de distribution. Le résultat étant que l'algorithme ne trouve pas de solution optimale au problème. Pour résoudre ce cas limite, nous avons dû introduire deux règles :

- Une entité ne peut pas appartenir à deux solutions de la même forme de distribution, sauf si nous créons les états de l'étape 1, ou s'il s'agit de la dernière pour obtenir le nombre d'éléments d'un groupe potentiel.

- Une entité à classer ne peut appartenir qu'au groupe de la médiane dont elle est le plus proche.

Enfin, nous avons encore explicitement exprimé dans notre implantation la règle que :

- Une médiane ne peut appartenir au groupe d'une autre médiane.

Nous avons alors procédé à une nouvelle série de simulations qui nous a montré que les qualités de calcul de l'algorithme original étaient préservées, mais que les temps de calcul étaient en revanche très réduits. Cette nouvelle formulation ne nous a pas entièrement satisfaits, car sa *complexité mémoire est encore trop importante*. Pour traiter de grands problèmes, nous serions contraints de stocker une partie des données en mémoire auxiliaire [=disque].

L'examen détaillé de l'algorithme d'optimisation original nous a montré que nous réalisions pratiquement un parcours d'arbre en largeur [breath search] pour examiner les états à lier, ce qui oblige à mémoriser tous les résultats intermédiaires. Nous avons donc décidé d'utiliser la technique de la recherche en profondeur [depth search]. De cette manière, nous n'avons plus besoin de mémoriser en permanence tout le réseau des états et réduisons la complexité mémoire.

Nous exposons en détail cet algorithme :

Soit d une mesure de la distance entre les objets

1. Déterminer les objets représentatifs (médioïdes)

   Le premier médioïde est déterminé par :

   Trouver i | $\sum d_{ij} = \min_i \delta d_{ij}$ i,j = 1,...,n (8.1)

   Les médioïdes suivants sont déterminés par la formule itérative :

   Trouver i | Ci = max(C_j) j=1,..,n (8.2)

   avec $C_j = \sum_q \max(d_{pq} - d_{pj}, 0)$ (8.3)

   q=1,...,n(-médioïdes)
   p | p ε {médioïdes)

   L'ensemble des médioïdes est amélioré par une série de permutations par la procédure itérative

   Tant que l'on effectue une permutation :

   Trouver i | $C_i > 0$ (8.4)

   avec $C_i = \sum_q (d_{pq} - d_{qi})$ (8.5)

   q=1,.,n(-médioïdes,i)
   tel que q est plus proche de p que des autres médioïdes
   p= 1 médioïde donné
   i= 1 objet non médioïde

2. Déterminer les formes de distribution des n objets en m groupes de telle sorte que :

   $n_1 \geq n_2 \geq \ldots \geq n_m$ (8.6)

3. Générer les états forme par forme et étape par étape en calculant la formule récurrente, et en utilisant les règles de construction énoncées ci -dessus :

   Pour f variant de 1 au nombre de forme faire :
   Pour x variant de 1 à m :

$$W_{k+1}^*(z) = \begin{cases} 0 \text{ pour } k+1 = M_0 \\ \min_z [T(z-y) + W_k^*(y)] \\ \text{pour } k = 1,..M_0-1 \end{cases} \quad (8.7)$$

où

M = nombre de sous-ensembles non-vides et disjoints dans lesquels les n éléments sont classés.
M_0 = M si N ≥ 2M, et N - M si N < 2M
f = index de la forme de distribution
x = index de l'objet représentatif

k = index de la variable d'étape.
z = variable d'état représentant un ensemble donné d'entités à l'étape k+1, ordonnées en fonction de leur distance par rapport au médioïde de l'étape k+1.
y = variable d'état représentant un ensemble donné d'entités à l'étape k. Le nombre d'entités dans z et y est donné par la forme de distribution.
z-y = sous-ensemble de toutes les entités contenues dans z, mais qui ne le sont pas dans y.
T(z-y) = le coût de transition de l'étape k à l'étape k+1.

L'homogénéité d'une partition est mesurée par le critère, soit la somme des carrés dans les groupes :

$$W = \sum_{k=1}^{m} T(y_k) \text{ avec } T(y_k) = (1/n_k) \sum_{i,j \in y_k} (d_{ij})^2 \qquad (8.8)$$

ou la somme des distances aux médianes

$$W = \sum_{k=1}^{m} T(y_k) \text{ avec } T(y_k) = \sum_{\substack{i,j \in y_k \\ i = \text{médiane}}} (d_{ij}) \qquad (8.9)$$

Si $W_k^* < W'$, alors $W' = W_k^*$ et mémoriser la solution qui a généré W_k^*.

A la fin des itérations, la solution mémorisée est la solution optimale.

Par rapport aux méthodes de partitionnement et hiérarchiques, nous cherchons à minimiser par la formule récurrente sur l'ensemble des objets contenus dans une solution et en examinant tous les branchements possibles à partir d'une étape.

Une fois, les objets représentatifs déterminés, nous n'effectuons pas une simple affectation, mais un véritable calcul d'optimisation par rapport à PAM.

Par rapport à la programmation dynamique, nous éliminons toutes les équations redondantes, en introduisant les concepts d'objets représentatifs et de chaînage des autres. Nous supprimons également la redondance des formes de distribution par la règle 8.6. Nous utilisons un parcours en profondeur et ne générons les solutions que lorsque nous avons besoin.

8.3 Résultats des nouveaux concepts

Nous avons réalisé une nouvelle phase de simulation restreinte, par rapport à celle du chapitre 7, avec les deux modèles de générateurs. Enfin, nous avons soumis les nouvelles méthodes de programmation dynamique à des tests sur des

cas limites, puisqu'elles sont plus rapides que la nouvelle mouture de l'algorithme de partitionnement. Nous avons cette fois travaillé sur un AT-286, équipé d'un co-processeur mathématique.

Nous avons repris la méthode de PAM, comme exemple de méthode classique, à laquelle nous avons confronté les deux nouvelles versions de la programmation entière décrites dans la section 1, et les deux algorithmes dynamiques décrits ci-dessus.

Le premier tableau présente les résultats des simulations, le second donne les temps de calcul obtenus avec divers échantillons.

méthode	Normal	Monte-Carlo
PAM	62	65
Rao (listes modifiées)	98	98
Pamrao (Rao + Rousseeuw	98	98
Dynamique (+Rao)	100	100
Dynamique (Rao + Rous.)	100	100

Modèle	Pam	Rao	Pamrao	Dynarao	Newdyn
6/2	1"	1"	1"	1"	0"57
10/2	1"	1"	1"	1"	1"
20/2	2"	15"	3"	1"	1"
6/3	1"	1"	1"	1"	0"81
9/3	1"55	3"	1"	1"60	0"90
12/3	1"79	12"	2"	2"37	0"90
18/3	2"76	118"	4"	11"03	2"35
30/3	5"60	2800"	50"	25"	5"87

Les temps de calculs sont pour ces petits jeux de données acceptables, même si nous pouvons déjà remarquer une explosion pour la méthode de programmation entière et la méthode de programmation dynamique originales. Cette explosion se reproduit également pour "pamrao" et "newdyna", dès que le nombre de données dépasse 20 et/ou le nombre de groupes dépasse 3. Mais pour cette dernière méthode le gain est excellent par rapport à la méthode originale de programmation dynamique. Pour classer 62 entités dans 3 groupes, les calculs durent environ 38 heures, alors qu'avec la méthode réduite, il ne faut que 23 minutes.

Finalement nous avons, pour compléter notre évaluation, repris les données de Hartigan dont nous nous étions servis au chapitre 7, soit les crimes dans les

villes et la valeur nutritive des aliments, sans standardiser les données. Nous obtenons avec nos trois nouvelles définitions les résultats ci-dessous :

Crimes (2 groupes) :

Groupe 1 : Hartford, Atlanta, Boston, Chicago

Groupe 2 : Dallas, Denver, Detroit, Honolulu

Crimes (3 groupes) :

Groupe 1 : Denver, Dallas, Detroit, Honolulu

Groupe 2 : Chicago, Boston

Groupe 3 : Atlanta, Hartford

Enfin, les trois groupes pour les aliments sont :

Groupe 1 : steak de boeuf, boeuf braisé, roast beef

Groupe 2 : hamburger, rôti de boeuf, poulet frit

Groupe 3 : coeur de boeuf, poulet rôti

Comme nous nous y attendions les résultats sont les mêmes. En standardisant les données des "crimes", nous obtenons également la solution proposée au chapitre 7.

8.4 Recommandations

Les méthodes dites classiques se sont révélées très inférieures aux méthodes de programmation mathématique lors de tous nos tests. L'écart entre les différentes méthodes classiques est même très important. Nous pourrions recommander l'abandon des méthodes hiérarchiques et de partitionnement en raison de ces mauvais résultats. Nous sommes d'avis que les ajouts de critères, ou une nouvelle formule de récurrence ne changeront rien. Nous avons pu montrer que la faible convergence de ces méthodes provient en réalité de la réduction des calculs à des paires, soit d'entités ou de sous-classes. Cette réduction conduit souvent à ignorer une solution correcte au point de vue optimalité. Sous certaines conditions, la méthode développée par le professeur Rousseeuw donne des résultats satisfaisants. Cette recommandation est toutefois une utopie, car chaque recherche a son but et les sommes d'efforts investis dans ces méthodes rendent leur abandon impossible.

Toutes les méthodes issues de la programmation mathématique ne donnent pas satisfaction. Ainsi, nous avons constaté, que les méthodes de Branch and Bound et la réduction par un lagrangien de la formulation en programmation entière ne donnent pas entière satisfaction quant à leur convergence. La deuxième est d'ailleurs de loin la plus mauvaise des méthodes que nous avons examinées. Nous recommandons l'abandon de la recherche sur ces deux méthodes.

Les formulation de programmation entière et de programmation dynamique

obtiennent de loin les meilleurs résultsts. Leur convergence est de 100%.

Maigré les faibles performances obtenues, nous pensons que la voie de la programmation mathématique est la solution d'avenir de la classification automatique. Elle ne résout pas le problème du nombre de groupes optimum pour le jeu de données. Même si la méthode de calcul est bonne, nous n'avons pas cependant résolu tous les problèmes de "mauvaise" utilisation. Pour le moment, nous ne sommes pas encore capable de déterminer avec certitude le nombre des groupes à chercher, nous ne savons pas quelle est la meilleure fonction-objectif à minimiser, et dans quel cas faut-il standardiser les données.

La réduction de la complexité de l'algorithme de programmation dynamique a donné de bonnes performances au point de vue temps de calcul par rapport à l'original, mais nous sommes loin de la vitesse des méthodes classiques. Dès que le nombre de groupes recherché dépasse 2, et que le nombre d'objets à classer dépasse 30 nous avons déjà trop de solutions à calculer. En revanche, le gain par rapport à l'algorithme original est très important : il faut environ 23 minutes avec la nouvelle définition pour former trois groupes à partir des 62 entités d'un de nos jeux de test. Avec l'original, il nous a fallu patienter 36 heures. Nous avons eu l'occasion de tester brièvement une machine équipée d'un processeur i486 (nouvelle génération), les performances sont dix à quinze fois meilleures, que ce que nous avons obtenu jusqu'à présent. L'amélioration de la vitesse de calcul des PC de pointe va permettre de compenser la lenteur relative de ces algorithmes. Nous n'avons pas cherché à utiliser ce nouvel algorithme aur des machines encore plus performantes (mini ou mainframes).

Annexe 1 : Mode d'emploi des programmes PC

Sur demande, nous fournissons certains des programmes développés dans le cadre de cette étude :

RAOMEDIA : programme selon l'algorithme de programmation entière complet

PAMRAO : programme selon l'algorithme de programmation entière réduit

DYNARAO : programme selon l'algorithme de programmation dynamique incluant la réduction par la règle de chaînage

NEWDYNA : réduction de l'algorithme de programmation dynamique par la règle de chaînage et la détermination des objets représentstifs

Tous ces programmes ont été développés sous DOS 3.30, en utilisant les librairies d'émulation et le modèle large de mémoire. L'usage de l'émulation nous permet d' exploiter un co-processeur mathématique, s'il est présent. La configuration matérielle requise pour les utiliser est :

- 640 Kbytes de mémoire vive; [dont 500 libres]
- un disque dur;
- un écran monochrome;
- en option un co-processeur mathématique;

Le co-processeur réduit, selon nos tests faits avec un AT-286 à 12Mhz, les temps de calcul au tiers de ceux obtenus en configuration simple. Par contre, nous n'avons pas vu de différence de précision des calculs. Les programmes ne sont pas protégés, ils peuvent être copiés sur un disque dur, ou une disquette de sauvegarde, par la commande COPY de DOS, ou toute commande équivalente d'un gestionnaire de fichier [PC-Tools,Norton-Commander, etc].

Voici le mode d'emploi des programmes de programmation entière, dont le dialogue est en français :

- Saisie des données par clavier ou fichier;

  Dans les 2 cas, le programme va demander, respectivement lire dans le fichier les données dans l'ordre suivant :

- nombre d'objets à classer;
- nombre de groupes à former;
- norme à utiliser pour le calcul de la distance;
- nombre de caractéristiques des objets;

- les éléments de la matrice des données, chaque ligne contenant les observations pour un objet;

- Choix de l'unité de sortie, fichier ou écran;

En cas de travail avec des fichiers, il faut impérativement donner le nom complet du fichier, selon les règles de DOS, s'il ne se trouve pas dans le répertoire des programmes. Pour les personnes en ayant l'habitude, la commande PATH, pour donner le chemin d'accès aux programmes, peut être utilisée sans restriction.

Ces deux programmes permettent de lire une matrice de distances, dans ce cas, elle doit être préparée sous forme du triangle inférieur, avec les zéros. Les limites des programmes sont données à l'écran et des tests sont faits pour vérifier qu'elles sont respectées. En cas de saisie interactive, les données peuvent être corrigées. Lors d'une lecture aur un fichier, l'erreur est signalée et le programme s'arrête.

Pour les deux autres programmes, si les données sont entrées par un fichier, il faut que ce fichier soit rentré comme une matrice :

```
X11 X12 X1n
X21 X22 X2n


Xm1 Xm2 Xmn
```

Ensuite le dialogue suivant guide l'utilisateur dans le choix des options :

1. Give the number of objects :

   Give the number of variables :

   do you want to read your data from the keyboard Y/N

   Y = entrée des données aous forme de dialogue

   N = lecture des données depuis le fichier dont le nom est donné plus tard

   Do you want your output on screen, printer, or in a file

   Please give the nsme of the device (con, prn, filename) :

   con = sortie à l'écran

   prn = sortie sur l'imprimante lpt1

   filename = écriture dans un fichier, donner le nom selon les régles de DOS

2. Give the number of clusters :

Choose between L1 or L2 :

1 : = L1

2 : = L2

Do you want to standardize your data :

Y = yes

N = No

Which objective function do you wish to use

Total distance to the medians = 1

Within clusters sum of squares = 2

Average distance to the medians = 3

3. Après la sortie du résultat :

Do you want another run ?

Y = yes = continuer de travailler avec le programme

N = no = fin du travail

Do you want to use the same dataset ?

Y = yes : continuer avec les mêmes données, dans ce cas le dialogue reprend au point 2.

N = no : travailler avec de nouvelles données, reprise avec le point 1.

Pour toute remarque, ou pour disposer des informations liées aux développements prévus après cette étude, vous pouvez nous contacter aux adresses suivantes :

Prof. Y Dodge

Groupe de Statistique

Division économique de l'Université

Pierre-à-Mazel 7

2000 Neuchâtel

Thierry Gafner

Sous le Chêne 2

2043 Boudevilliers

Annexe 2 : Exemple économique de la banque d'Iran

Nous présentons ci-dessous la classification obtenue pour des données économiques publiées par la banque d'Iran. Il s'agit de 55 indicateurs relevés pour chacune des 23 régions iraniennes. Nous avons effectué les calculs avec notre nouvel algorithme de programmation dynamique, en appliquant d'office les deux règles présentées au chapitre 8. Nous avons cherché à classer les données dans 2 à 6 groupes, en utilisant la distance de Manhattan et la distance totale aux médianes. Tous les calculs ont été effectués sur un AT-286 avec un coprocesseur.

Nous pouvons remarquer que l'effet de la classification est à chaque fois de sortir une ou plusieurs données du groupe 1, dont l'entité {2} est la médiane, seul le passage entre 5 et 6 groupes produit un effet différent. Nous voyons que entre le formation du 3^e groupe et du 5^e, nous n'avons formé que des groupes à une entité (singleton cluster), qui est la médiane nouvellement déterminée.

```
*******************************************************
*                                                     *
*  Dynamic Programing  Cluster Analysis Algorithm     *
*  reduced with the procedure BSWAP developped by     *
*  Prof. P. Rousseeuw and L. Kaufmann Brussels  and   *
*  the medians'rule developped by Rao for             *
*  the Integer Programing Algorithm                   *
*   author  : T. Gafner Univesity of Neuchâtel        *
*                                                     *
*                                                     *
*   Program's Limits are :                            *
*             Objects    : 100                        *
*             Clusters   : 10                         *
*             Variables  : 60                         *
*             Distances  : L1 or L2                   *
*                                                     *
*******************************************************

Summary of options :

Objects   :  23
Variables :  55
Clusters  :   2
Distance  :  Manhattan
Standardisation of variables
Total distance to the medians


Begin of computations at : 17:23:31

Cluster  1     Objects    2  4 10  9  7 11 15 20 12 19 13 18 17 16 22
                          3 14  6 21 23  8  5

Cluster  2     Objects    1


Total distance to the medians
    1163.389000

End of computations at : 17:24: 9


Summary of options :

Objects   :  23
Variables :  55
Clusters  :   3
Distance  :  Manhattan
Standardisation of variables
Total distance to the medians


Begin of computations at : 17:27: 8

Cluster  1     Objects    2  4 10  7 11 18 22  3  6  8  5

Cluster  2     Objects   17 14  9 16 21 13 20 15 12 23 19
```

```
Cluster  3     Objects    1


Total distance to the medians
     995.439000

End of computations at : 17:29:54


Summary of options :

Objects    :  23
Variables :  55
Clusters  :   4
Distance  :  Manhattan
Standardisation of variables
Total distance to the medians


Begin of computations at : 17:30:13

Cluster  1     Objects    2  4 10  7 11 18 22  3  6  8

Cluster  2     Objects   17 14  9 16 21 13 20 15 12 23 19

Cluster  3     Objects    5

Cluster  4     Objects    1


Total distance to the medians
     911.936200

End of computations at : 17:39:44


Summary of options :

Objects    :  23
Variables :  55
Clusters  :   5
Distance  :  Manhattan
Standardisation of variables
Total distance to the medians


Begin of computations at : 18: 2:41

Cluster  1     Objects    2  4 10  7 11 18 22  3  6  9

Cluster  2     Objects   17 14 16 21 13 20 15 12 23 19

Cluster  3     Objects    8

Cluster  4     Objects    5

Cluster  5     Objects    1


Total distance to the medians
```

```
     835.488500

End of computations at : 18:22: 9


Summary of options :

Objects    :  23
Variables :  55
Clusters  :   6
Distance  :  Manhattan
Standardisation of variables
Total distance to the medians


Begin of computations at : 18:22:56

Cluster  1     Objects     2  4 10  7 11 18 22  3  6

Cluster  2     Objects    17 14  9 16 13 20 15 19

Cluster  3     Objects    21 23 12

Cluster  4     Objects     8

Cluster  5     Objects     5

Cluster  6     Objects     1


Total distance to the medians
     767.606800

End of computations at : 18:47:17
```

Bibliographie

ARTHANARI T.S., DODGE Y., (1981). Mathematical Programming in Statistics,Wiley, New York.

BELLMAN R.E., DREYFUS S.E. (1965). La programmation dynamique et ses applications, Dunod, Paris.

Classification Society of America, Journal of Classification, Springer International, vol 4 no 1, 1987.

COOPER M.C.,MILLIGAN G. W. (1985). An Examination of Procedures Determining the Number of Clusters in a Data Set, Psychometrica,50, 159-179.

EVERITT B. (1974). Cluster Analysis, Social Science Research Council, London.

GARFINKEL R. S., NEMHAUSER G.L. (1972). Integer Programming, Wiley,New York.

GRAF-JACOTTET M. (1979). Classification automatique aspects mathématiques, Cahier de Méthodes Quantitatives, Université de Neuchâtel.

HARTIGAN J.A. (1975). Clustering Algorithms, Wiley, New York.

HASTINGS N.A.J. (1973). Dynamic Programming with Management Applications, Butterworths, London.

International Federation of Classification Society, Actes du congrès international de juillet 1987 à Aix-La-Chapelle.

JENSEN R.E. (1969). A Dynamic Programming Algorithm for Cluster Analysis, Operation Research 17, 1034.

METZGER-VOIDE A-C. (1983). Clasfac-Apl un package interactif d'analyse statistique multivariable, Thèse de docorat Université de Neuchâtel.

RAO M.R. (1971). Cluster Analysis and Mathematical Programming, Journal of the American Statiscal Association 66, 622.

ROUSSEEUW P. (1987). Cluster Analysis, Wiley, New York.

SPÄTH N. (1980). Cluster Analysis Algorithms, Ellis Horwood, London.

SPÄTH N. (1986). Anti-clustering : Maximising the variance criterion, Control and Cybenetics No. 2/3.

TRICOT M., DONEGANI M. (1987). Optimisation en classification automatique : sur une famille d'indices de proximité en classification hiérarchique ascendante, EPFL-DMA rapport 8702.

TRYON R.C., BAILEY D.E. (1967). Cluster Analysis, Wiley, New York.

VINOD H.D. (1969). Integer Programming and the Theory of Grouping, Journal of the American Statiscal Association 64, 506.

Table d'index